



Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Αυτοματισμού

Πτυχιακή Εργασία

Θέμα :

« Επαναπρογραμματιζόμενο Σύστημα ασφάλειας με χρωματικό κώδικα»



Επιβλέπων Καθηγητής : Μιχάλης Παπουτσιδάκης

Φοιτητές : Λούντζη Αθηνά Α.Μ 35158

Ράπτης Ιωάννης Α.Μ 33823

Αιγάλεω, Σεπτέμβριος 2013

Πίνακας Περιεχομένων

Περίληψη.....	6
Abstract.....	7
Κεφάλαιο 1: Η τεχνολογία και η χρήση του μικροελεγκτή arduino	8
1.1 Παρουσίαση των μικροελεγκτών	8
1.1.1 Πλεονεκτήματα των μικροελεγκτών	8
1.1.2 Κατηγορίες μικροελεγκτών	9
1.1.3 Εργαλεία ανάπτυξης και κατασκευαστές μικροελεγκτών.....	10
1.2 Arduino	11
1.2.1 Ιστορία	11
1.2.2 Η πλακέτα arduino	12
1.2.3 Arduino Duemilanove/UNO	12
1.2.3.1 Flash memory	12
1.2.3.2 SRAM memory	12
1.2.3.3 EEPROM memory	13
1.2.3.4 FTDI	13
1.2.3.5 Pins πλακέτας arduino	13
1.2.4 Άλλες εκδόσεις arduino	15
1.2.5 Το software για τον προγραμματισμό του arduino	18
1.2.6 Arduino shields	19
1.3 Προγραμματισμός του ελεγκτή Arduino	22
1.3.1 Το περιβάλλον Arduino IDE (integrated development environment)	22
1.3.2 Blinking Led.....	24
1.3.3 Επεξήγηση του Sketch	26

1.3.4 Μεταβλητές και τύποι δεδομένων	27
1.3.4.1 Ονομασία μεταβλητών	27
1.3.4.2 Τύποι δεδομένων των μεταβλητών.....	28
1.3.4.3 Πίνακες (array)	28
1.3.4.4 Qualifiers μεταβλητών.....	29
1.3.4.5 Προκαθορισμένες μεταβλητές	30
1.3.4.6 Εμβέλεια μεταβλητών	30
1.3.5 Εκχώρηση τιμών	30
1.3.6 Μαθηματικοί τελεστές	30
1.3.7 Σύνθετοι τελεστές	31
1.3.7.1 Τελεστές προσάυξησης και μείωσης	31
1.3.7.2 Συσχετιστικοί τελεστές	32
1.3.7.3 Λογικοί τελεστές	33
1.3.8 Σειρά προτεραιότητας των τελεστών	33
1.3.9 Εντολές ελέγχου	33
1.3.9.1 Εντολή if else.....	34
1.3.9.2 Εντολή switch	35
1.3.10 Βρόχοι.....	36
1.3.10.1 For	36
1.3.10.2 While	37
1.3.10.3 Do	37
1.3.10.4 Continue	38
1.3.11 Συναρτήσεις	38
1.3.11.1 Βασικές συνάρτησης setup & loop.....	38
1.3.11.2 Δημιουργία νέων συναρτήσεων	38

1.3.11.3 Καλώντας τις συναρτήσεις	39
1.3.11.4 Επιστροφές τιμών συνάρτησης	39
1.3.11.5 Παράμετροι συνάρτησης	39
1.3.11.6 Ψηφιακές συναρτήσεις	40
1.3.11.6.1 pinMode()	40
1.3.11.6.2 digitalWrite().....	41
1.3.11.6.3 digitalRead()	41
1.3.11.7 Αναλογικές συναρτήσεις	41
1.3.11.7.1 analogRead()	42
1.3.11.7.2 analogWrite()	42
1.3.11.7.3 analogReference()	42
1.3.12 map()	43
1.3.13 constrain()	43
1.3.14 Προχωρημένες συναρτήσεις	44
1.3.14.1 Συναρτήσεις Timing	44
1.3.14.1.1 delay()	44
1.3.14.1.2 delayMicroseconds()	44
1.3.14.1.3 millis()	44
1.3.14.1.4 macros().....	45
1.3.14.2 Συναρτήσεις random.....	45
1.3.14.2.1 random()	45
1.3.14.2.2 randomSeed().....	45
1.3.15 Hardware Διακοπές	46
1.3.16 Σειριακή βιβλιοθήκη arduino.....	46
1.3.16.1 begin()	47

1.3.16.2 available()	47
1.3.16.3 read()	47
1.3.16.4 print()	47
1.3.16.5 println()	48
1.3.17 Εισαγωγή βιβλιοθήκης	48
1.3.18 Comments	48
1.4 Εφαρμογές arduino	49
1.4.1 Εφαρμογές του arduino στην μουσική	49
1.4.1.1 Laser harp	49
1.4.1.2 Soundmachine	50
1.4.2 Εφαρμογές του arduino στον μοντελισμό	52
1.4.3 Εφαρμογές του arduino για την διακόσμηση	52
1.4.3.1 Led cube	52
1.4.4 Εφαρμογές του arduino στο σπίτι	53
1.4.4.1 Arduino security alarm	53
1.4.4.2 RGB combination door lock	54
1.4.5 Εφαρμογές arduino σε παιχνίδια	54
1.4.5.1 Marble labyrinth ελεγχόμενο με χρήση the WiiFit	54
1.4.6 Εφαρμογές του arduino στις τέχνες (ζωγραφική)	55
1.4.6.1 Senseless drawing bot	56
1.4.6.2 LumiBots	56
1.4.7 Εφαρμογές του arduino στην ρομποτική	57
1.4.8 Mind-control	57
Κεφάλαιο 2: ΕΦΑΡΜΟΓΗ - ΣΧΕΔΙΑΣΜΟΣ - ΚΑΤΑΣΚΕΥΗ –ΛΕΙΤΟΥΡΓΙΑ	59
2.1 Η ιδέα της εφαρμογής – περιγραφή	59

2.2 Σχεδιασμός μηχανικού μέρους	59
2.2.1 Υλικά κατασκευής-προδιαγραφές.	59
2.3 Η μηχανική κατασκευή.....	60
2.4 Σχεδιασμός του λογισμικού	76
2.4.1 Τι είναι η Processing	77
2.4.2 Αλγόριθμος processing	77
2.4.2.1 Αλγόριθμος κυρίου μέρους (processing)	77
Κεφάλαιο 3:	
Μελλοντικές επεκτάσεις συστήματος	91
3.1 Μελλοντικές επεκτάσεις σε θέματα τροφοδοσίας	91
3.1.1 Χρήση μπαταρίας	91
3.1.2 Χρήση ανανεώσιμων πηγών ενέργειας	91
3.2 Μελλοντικές επεκτάσεις ως προς τη χρήση συσκευών δράσης	92
3.3 Μελλοντικές επεκτάσεις γενικής χρήσης συστήματος	92
Βιβλιογραφία - πηγές πληροφοριών	93

Περίληψη

Στη παρούσα πτυχιακή εργασία αναπτύσσεται η κατασκευή και ο προγραμματισμός μιας ηλεκτρονικής κλειδαριάς με χρωματικό κώδικα. Το πληκτρολόγιο δεν περιέχει αριθμούς αλλά θα κατασκευαστεί από RGB leds που θα τροφοδοτούνται από έναν μικροεπεξεργαστή. Επομένως αντί για πληκτρολόγηση αριθμών, ο κωδικός θα είναι ένας μοναδικός συνδυασμός χρωμάτων. Τα κατασκευαστικά μέρη της πτυχιακής θα είναι, ένα πληκτρολόγιο, μια πλακέτα και ο μικροεπεξεργαστής.

Στόχος μας είναι με τον κατάλληλο προγραμματισμό να γίνεται η αναγνώριση της σωστής ή μη πληκτρολόγησης του κωδικού.

Abstract

In this thesis develops the construction and programming of an electronic lock with a color code. The keyboard does not contain numbers but will be built by RGB LEDs that will be powered by a microprocessor. So instead of typing numbers, the code will be a unique combination of colors. The components of the thesis will be, a keyboard, a plaque and the microprocessor.

Our goal is, with proper programming of the microprocessor to identify if we have entered the correct password or not.

ΚΕΦΑΛΑΙΟ 1:

Η ΤΕΧΝΟΛΟΓΙΑ ΚΑΙ Η ΧΡΗΣΗ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ

ARDUINO

1.1 Παρουσίαση των μικροελεγκτών

Ένας μικροελεγκτής είναι ένα ενσωματωμένο τσιπ (ολοκληρωμένο κύκλωμα) που αποτελεί συχνά μέρος ενός συστήματος. Ο μικροελεγκτής περιλαμβάνει CPU, RAM, ROM, θύρες εισόδου/εξόδου και timers σαν έναν απλό τυπικό υπολογιστή, αλλά επειδή είναι σχεδιασμένα να εκτελούν μόνο μία συγκεκριμένη εργασία για τον έλεγχο ενός απλού συστήματος, είναι πολύ μικρότερα και απλούστερα σχεδιασμένα ώστε να μπορούν να περιλαμβάνουν όλες τις λειτουργίες που απαιτούνται σε ένα μόνο ολοκληρωμένο κύκλωμα.

Ο μικροελεγκτής διαφέρει από τον μικροεπεξεργαστή στον οποίο του δόθηκε έμφαση στην υπολογιστική ισχύ. Έτσι αν συνδυαστεί με τις κατάλληλες εξωτερικές περιφερειακές συσκευές μπορεί να πραγματοποιήσει μία πληθώρα γενικών εργασιών. Σε αντίθεση ο μικροελεγκτής είναι σχεδιασμένος για πιο εξειδικευμένες εργασίες, έχει πολύ μικρότερες δυνατότητες συνεργασίας με τα εξωτερικά περιφερειακά αφού υστερεί κατά πολύ σε υπολογιστική ισχύ. Στον σχεδιασμό των μικροελεγκτών δόθηκε περισσότερη έμφαση στο να απαιτούν πολύ μικρότερο αριθμό ολοκληρωμένων κυκλωμάτων για τη λειτουργία μίας συσκευής, το χαμηλό κόστος κατασκευής τους και τον εύκολο προγραμματισμό εξειδικευμένων εργασιών.

1.1.1 Πλεονεκτήματα των μικροελεγκτών

Ακολουθούν τα πλεονεκτήματα των μικροελεγκτών:

- Αυτονομία, μέσο της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιώντας το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλές ταχύτητες λειτουργίας.

- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσης τους για τη σύνδεση εξωτερικών περιφερειακών συσκευών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.
- Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών νμικροεπεξεργαστών, αν και στους πρώτους συναντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (π.χ. οι σειρές από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου φον Νόιμαν.

1.1.2 Κατηγορίες μικροελεγκτών

Λόγω του ισχυρότατου ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή, η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές. Έτσι διακρίνονται οι εξής κυρίως κατηγορίες:

- Μικροελεγκτές (καμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα για να μην μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως για παράδειγμα οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel, Dallas κ.α.).
- Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου.

Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.

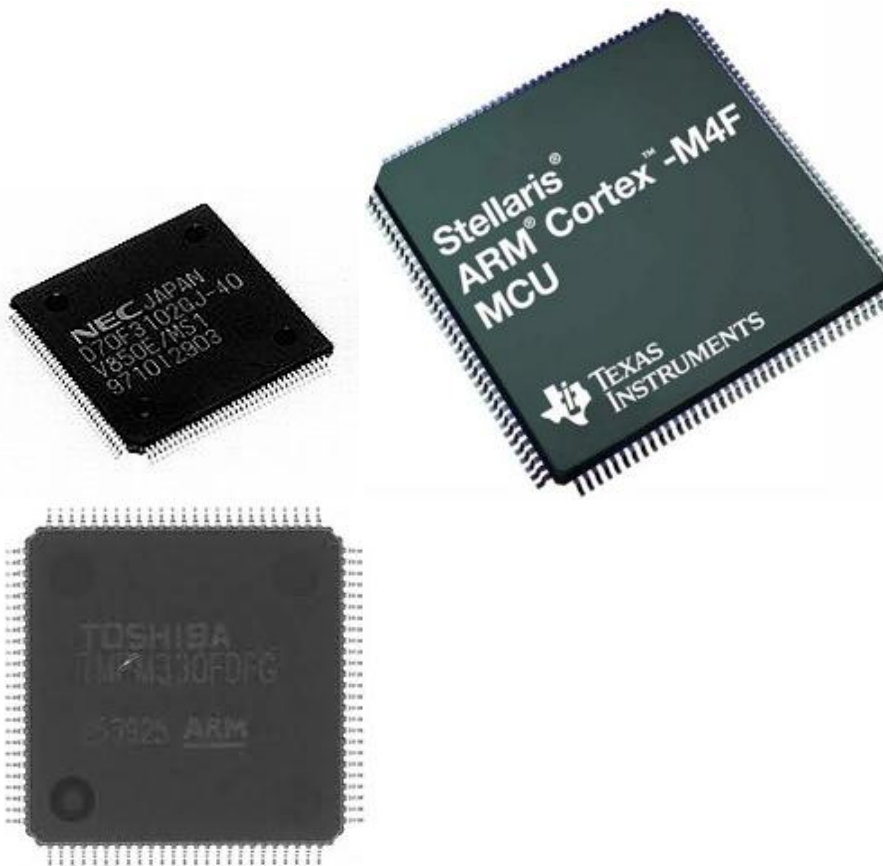
- Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή. Για παράδειγμα μεταξύ των μικροελεγκτών τύπου ARM ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί φυσικά να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).
- Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware.

Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

1.1.3 Εργαλεία ανάπτυξης και κατασκευαστές μικροελεγκτών

Η επιτυχία μίας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), τη δυνατότητα προγραμματισμού της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές τα εργαλεία αυτά δεν αποτελούνται ποτέ μόνο από λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας μεταξύ τους. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες.

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται μεγαλύτερη ταχύτητα ή μικρότερο μέγεθος χρησιμοποιούμενης μνήμης μπορεί να χρησιμοποιηθεί η Assembly. Όμως οι μεγαλύτερες δυνατότητες σε λειτουργικότητα και η ευκολία προγραμματισμού σε C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την assembly από τις περισσότερες εφαρμογές. Οι σημαντικότεροι κατασκευαστές μικροελεγκτών είναι η ARM η οποία δεν κατασκευάζει αλλά παραχωρεί δικαιώματα χρήσης του πυρήνα, η Atmel όπου και θα ασχοληθούμε εκτενέστερα την συνέχεια, η Epson, η Freescale Semiconductor (πρώην Motorola), η Hitachi, η Maxim (μετά την εξαγορά της Dallas), η Microchip, η NEC, η Toshiba και η Texas Instrument.



Εικόνα 2.1.3.1 - Μικροεπεξεργαστές από τις εταιρίες NEC, Texas Instruments και Toshiba

1.2 Arduino

1.2.1 Ιστορία

Το 2005 στην Ivrea της Ιταλίας κατασκευάζεται μία συσκευή η οποία θα είχε την δυνατότητα να ελέγχει και να αλληλεπιδρά σύμφωνα με το περιβάλλον. Σκοπός των κατασκευαστών ήταν αυτή η συσκευή να κοστίζει λιγότερο σε σχέση με άλλες παρόμοιων δυνατοτήτων. Η ομάδα αποτελούνταν από τους Massimo Banzi, David Cuartielles, Tom Igoe, David Mellis και Gianluca Martino. Το όνομα της συσκευής έχει τις ρίζες του από τον Arduino of Ivrea, έναν βασιλιά της Ιταλίας του ενάτου αιώνα όπου κατοικούσε στην ίδια πόλη. Η συσκευή ονομάστηκε “Arduino” που αντιστοιχούσε σε ένα ιταλικό ανδρικό όνομα και

σήμαινε “ισχυρός φίλος”.

Το arduino αναπτύχθηκε σύμφωνα με την πλατφόρμα Wiring, μία πτυχιακή εργασία του Hernando Barragan από το Interaction Design Institute Ivrea. Είχε ως στόχο να είναι μία ηλεκτρονική εκδοχή της Processing που θα χρησιμοποιούσε ένα περιβάλλον προγραμματισμού δικό της αλλά θα έμοιαζε σχεδιαστικά και συντακτικά με αυτό της Processing. Όχι πολύ καιρό πριν, αυτοί που εργάζονταν πάνω στον τομέα του hardware σήμαινε ότι κατασκεύαζαν κυκλώματα από το μηδέν, χρησιμοποιώντας εκατοντάδες διαφορετικές ηλεκτρονικές διατάξεις όπως αντιστάσεις, πυκνωτές, πηνία, τρανζίστορ, και πολλά άλλα. Τα κυκλώματα αυτά ήταν ενσύρματα με σκοπό να πραγματοποιήσουν συγκεκριμένες εργασίες. Όταν απαιτούνταν αλλαγές στις εργασίες τους, τότε έπρεπε να γίνουν και κάποιες χρονοβόρες αλλαγές στον σχεδιασμό των κυκλωμάτων, όπως αποσυγκολλήσεις και συγκολλήσεις των απαιτούμενων διατάξεων, αποσυνδέσεις και συνδέσεις καλωδίων κ.α.. Με την εμφάνιση της ψηφιακής τεχνολογίας και των μικροεπεξεργαστών, αυτές οι λειτουργίες οι οποίες πρώτα έπρεπε να γίνουν με καλώδια και διατάξεις τώρα αντικαταστάθηκαν από τα λογισμικά προγράμματα.

Το λογισμικό είναι πιο εύκολο να τροποποιηθεί από ότι το hardware. Με μερικά πατήματα πλήκτρων, μπορεί να αλλάξει ριζικά η λογική μίας συσκευής και να δημιουργηθούν επιπλέον ακόμη δύο ή τρεις δοκιμαστικές εκδόσεις. Η διαδικασία αυτή θα απαιτούσε το ίδιο χρονικό διάστημα όσο και ο χρόνος που απαιτείται για να κολληθεί ένα ζεύγος αντιστάσεων. Όπως είναι φυσικό μαζί με τον χρόνο μειώθηκε και το κόστος για τις τροποποιήσεις καθώς και για τις δοκιμαστικές εκδόσεις.

Όπως το περιγράφει ο δημιουργός του, το Arduino είναι μία open-source (ανοικτού κώδικα) πλατφόρμα «πρωτοτυποποίησης» ηλεκτρονικών κυκλωμάτων βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software που προορίζεται για οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα.

Το Arduino αποτελείται από δύο κύρια μέρη, την πλακέτα Arduino το οποίο είναι το κομμάτι του hardware πάνω στο οποίο εργάζεται ο κατασκευαστής όταν πραγματοποιεί μία κατασκευή ενώ το δεύτερο τμήμα είναι το Arduino IDE, το κομμάτι του λογισμικού που τρέχει στον υπολογιστή. Το IDE χρησιμοποιείται για να δημιουργηθεί ένα sketch (ένα μικρό πρόγραμμα στον υπολογιστή) που

φορτώνεται στον μικροελεγκτή της πλακέτα Arduino. Το sketch λέει στην πλακέτα arduino τι πρέπει να κάνει.

1.2.2 Η πλακέτα arduino

Η πλακέτα Arduino είναι ένα μικρό κύκλωμα (πλακέτα) που περιέχει ένα ολοκληρωμένο σύστημα υπολογιστή χρησιμοποιώντας ένα μικρό chip (ολοκληρωμένο κύκλωμα) που είναι ο μικροελεγκτής. Αυτός ο υπολογιστής είναι τουλάχιστον χίλιες φορές λιγότερο ισχυρός από ένα MacBook, αλλά είναι πολύ φθηνότερος και πολύ χρήσιμος για την κατασκευή ηλεκτρονικών συσκευών. Μάλιστα κάποιος θα μπορούσε να ισχυριστεί ότι λειτουργικά το Arduino μοιάζει πολύ με το NXT Brick των Lego Mindstorms NXT. Άλλωστε η ρομποτική είναι μία από τις πολλές κατηγορίες στις οποίες το Arduino διαπρέπει. Η ομάδα του Arduino έχει τοποθετήσει σε αυτήν την πλακέτα όλα τα απαραίτητα στοιχεία που απαιτούνται για τον μικροελεγκτή ώστε να μπορεί να λειτουργεί σωστά και να μπορεί να επικοινωνεί με τον υπολογιστή και άλλες σειριακές συσκευές.

1.2.3 Arduino Duemilanove/UNO

Η δημοφιλέστερη έκδοση του arduino είναι η Duemilanove/UNO που βασίζεται στο ολοκληρωμένο ATmega328, έναν 8-bit RISC μικροελεγκτή, ο οποίος χρονίζει στα 16MHz. Το ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

1.2.3.1 Flash memory

Η μνήμη flash έχει χωρητικότητα 32Kb, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware αυτό που στην ορολογία του arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash δεν χάνει τα περιεχόμενά της με την απώλεια της τροφοδοσίας ή κάνοντας reset το μικροελεγκτή. Επίσης, ενώ η μνήμη Flash υπό κανονικές συνθήκες δεν προορίζεται για χρήση runtime, μέσα από τα προγράμματα λόγω της μικρής συνολικής

μνήμης που είναι διαθέσιμη σε αυτά (2Kb SRAM + 1Kb EEPROM), έχει σχεδιαστεί μία βιβλιοθήκη που επιτρέπει την χρήση runtime στον χώρο που περισσεύει από την αποθήκευση των sketch (30Kb μείον το μέγεθος του προγράμματος σε μεταγλωττισμένη μορφή).

1.2.3.2 SRAM memory

Η μνήμη SRAM (static random access memory)είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν μεταβλητές, πίνακες κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο arduino σταματήσει ή αν γίνει reset. Στο ATmega328 η SRAM μνήμη καταλαμβάνει χώρο 2048

bytes κατά την διάρκεια μίας κανονικής λειτουργίας και όλες οι μεταβλητές φορτώνονται σε αυτή καθ' όλη την διάρκεια της λειτουργίας του microcontroller.

1.2.3.3 EEPROM memory

Το τελευταίο μέρος της μνήμης είναι η EEPROM και καταλαμβάνει 1024 bytes, αρκετά μικρή για μνήμη που χρησιμοποιείται μόνο για ανάγνωση (read-only). Η EEPROM έχει όριο ζωής καθώς δε μπορεί να επαναπρογραμματιστεί για περισσότερες από 100.000 φορές. Είναι μία byte addressable μνήμη, γεγονός που καθιστά λίγο δυσκολότερο να τεθεί σε χρήση αφού απαιτείται ειδική βιβλιοθήκη ώστε να μπορέσει κάποιος να έχει πρόσβαση σε αυτή.

1.2.3.4 FTDI

Εκτός όμως από το ATmega 328 το arduino χρησιμοποιεί και ένα FTDI ολοκληρωμένο. Οι μικροελεγκτές ATmega προγραμματίζονται χρησιμοποιώντας σειριακή επικοινωνία με τους υπολογιστές, έτσι το FTDI αναλαμβάνει την εργασία της μετατροπής της σειριακής θύρας σε USB.

1.2.3.5 Pins πλακέτας arduino

Η πλακέτα arduino διαθέτει :

- 14 ψηφιακές I/O θύρες (εισόδου & εξόδου). Σύμφωνα με το πρόγραμμα που θα φορτωθεί στον μικροελεγκτή αυτές οι θύρες μπορούν να εργαστούν σαν εισοδοί ή εξοδοί ψηφιακών σημάτων.
- Οι ψηφιακές θύρες 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές θύρες εξόδου με το σύστημα PWM (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Το PWM παίρνει ένα εύρος τιμών από το 0 έως το 255. Δεν είναι πραγματικά αναλογικό σύστημα, έτσι θέτοντας στην έξοδο την τιμή 127, δεν σημαίνει ότι η έξοδος θα παρέχει 2.5V αντί της κανονικής τιμής των 5V, αλλά ότι θα δίνει έναν παλμό που η τάση του θα εναλλάσσεται με μεγάλη συχνότητα και για ίσα χρονικά διαστήματα μεταξύ των τιμών 0V και 5V με σκοπό η μέση τιμή να ισούται με 2,5V.
- Οι θύρες 0 και 1 χρησιμοποιούνται επίσης και για να λαμβάνουν (RX) και να μεταδίδουν (TX) TTL σειριακά δεδομένα. Έτσι, όταν για παράδειγμα το πρόγραμμα στέλνει δεδομένα σειριακά, τότε αυτά προωθούνται στην θύρα USB μέσω του ελεγκτή Serial-Over-USB όπως επίσης και στο pin 0 για να τα διαβάσει ενδεχομένως μία άλλη συσκευή (π.χ. ένα δεύτερο arduino στη δικιά του θύρα 1). Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμα ενεργοποιηθεί το σειριακό interface, καταλαμβάνονται δύο ψηφιακές θύρες εισόδου/εξόδου.
- Οι θύρες 2 και 3 λειτουργούν και ως εξωτερικά interrupt (interrupt 0 και 1 αντίστοιχα). Με άλλα λόγια, μπορούν να ρυθμιστούν μέσα από το πρόγραμμα ώστε να λειτουργούν αποκλειστικά ως

ψηφιακές εισοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές τάσης, η κανονική ροή του προγράμματος να σταματάει άμεσα και να εκτελείται μία συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

- 6 αναλογικές θύρες εισόδου αριθμημένες από το 0 έως το 5. Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter). Για παράδειγμα, αν τροφοδοτηθεί ένα από αυτά τα pin με μία τάση η οποία μπορεί να κυμανθεί με ένα ποτενσιόμετρο από 0V ως μία τάση αναφοράς V_{ref} (η οποία αν δεν γίνει κάποια αλλαγή είναι προρυθμισμένη στα 5V), τότε μέσα από το πρόγραμμα μπορεί να «διαβαστεί» η τιμή της θύρας ως ένας ακέραιος αριθμός χωρητικότητας 10-bit, από το 0 (όταν η τάση στο pin είναι 0V) μέχρι το 1023 (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μία εντολή όπως για παράδειγμα στα 1.1V. Ένας άλλος τρόπος όπου η τάση αναφοράς μπορεί να δηλωθεί από τον προγραμματιστή είναι τροφοδοτώντας με μία εξωτερική τάση αναφοράς τη θύρα με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτηθεί η θύρα AREF με 3.3V και στην συνέχεια εκτελεσθεί η εντολή να διαβαστεί κάποιο pin αναλογικής εισόδου στο οποίο εφαρμόζετε τάση 1.65V, το Arduino θα επιστρέψει την τιμή 512.

- Δίπλα από της θύρες αναλογικής εισόδου, υπάρχει μία ακόμα συστοιχία από 6 pin με την σήμανση POWER. Η λειτουργία του καθενός pin έχει ως εξής:

- Το πρώτο, με την ένδειξη RESET, όταν γειωθεί (με οποιοδήποτε από τα 3 pin με την ένδειξη GND που υπάρχουν στο arduino) έχει ως αποτέλεσμα την επανεκκίνηση του arduino.

- Το δεύτερο με την ένδειξη 3.3V, μπορεί να τροφοδοτήσει διατάξεις, συσκευές ή αισθητήρες με τάση 3.3V. Η τάση αυτή δεν προέρχεται από την εξωτερική τροφοδοσία αλλά παράγεται από τον ελεγκτή Serial-over-USB και έτσι η μέγιστη ένταση που μπορεί να παρέχει είναι μόλις 50mA.

- Η τρίτη θύρα με την ένδειξη 5V, μπορεί να χρησιμοποιηθεί και αυτή για την τροφοδότηση διαφόρων εξαρτημάτων, συσκευών ή αισθητήρων με τάση 5V. Ανάλογα με τον τρόπο τροφοδοσίας του ίδιου του Arduino, η τάση αυτή προέρχεται είτε άμεσα από την θύρα USB (που ούτως ή άλλως παρέχει τάση 5V), είτε από την εξωτερική τροφοδοσία αφού αυτή περάσει από ένα ρυθμιστή τάσης για να την «σταθεροποιήσει» στα 5V.

- Το τέταρτο και το πέμπτο pin με την ένδειξη GND είναι οι γειώσεις.

- Το έκτο και τελευταίο pin, με την ένδειξη V_{in} έχει διπλό ρόλο. Σε συνδυασμό με το pin γείωσης δίπλα του, μπορεί να λειτουργήσει ως μέθοδος εξωτερικής τροφοδοσίας του Arduino στην περίπτωση που δεν βολεύει να χρησιμοποιηθεί η υποδοχή του φικ των 2.1mm. Αν όμως υπάρχει ήδη συνδεδεμένη εξωτερική τροφοδοσία μέσω του φικ, τότε μπορεί να χρησιμοποιηθεί αυτό το pin για να τροφοδοτήσει εξαρτήματα και συσκευές με την πλήρη τάση της εξωτερικής τροφοδοσίας (7~12V), πριν αυτή περάσει από τον ρυθμιστή τάσης όπως γίνεται με το pin των 5V.

- Η φόρτωση του sketch πραγματοποιείται μέσω μίας USB θύρας που διαθέτει η πλακέτα arduino. Έτσι οι πληροφορίες που προέρχονται από την USB θύρα του υπολογιστή εισέρχονται στην USB θύρα του arduino και στην συνέχεια οδηγούνται στο FDTI ολοκληρωμένο για να διαμορφωθούν σε μία κατάλληλη μορφή ώστε ο μικροελεγκτής να μπορέσει να τις διαβάσει.

- Πάνω στην πλακέτα του arduino υπάρχει ένας διακόπτης micro-switch και 4 smd (μικροσκοπικά) LED επιφανειακής στήριξης. Η λειτουργία του διακόπτη (που έχει την σήμανση RESET) και του

ενός LED με την σήμανση POWER είναι προφανής. Τα δύο LED με τις σημάνσεις TX και RX, χρησιμοποιούνται ως ένδειξη λειτουργίας του σειριακού interface, καθώς ανάβουν όταν το arduino στέλνει ή λαμβάνει (αντίστοιχα) δεδομένα μέσω της USB. Τα LED αυτά ελέγχονται από τον ελεγκτή Serial-over-USB και συνεπώς δεν λειτουργούν όταν η σειριακή επικοινωνία γίνεται αποκλειστικά μέσω των ψηφιακών pin 0 και 1.

- Τέλος, υπάρχει το LED με τη σήμανση L. Η βασική λειτουργία του LED στην πλακέτα Arduino είναι για να αναβοσβήνει συνήθως για δοκιμαστικό σκοπό. Οι κατασκευαστές σκέφτηκαν να ενσωματώσουν ένα LED στην πλακέτα το οποίο το σύνδεσαν στη ψηφιακή θύρα 13. Έτσι ακόμα και αν δεν έχει συνδεθεί τίποτα πάνω στο φυσικό pin 13, αναθέτοντας του την τιμή HIGH μέσα από το πρόγραμμα, θα ανάψει το ενσωματωμένο LED L.

- Η πλακέτα μπορεί να τροφοδοτηθεί από μία USB θύρα ενός υπολογιστή ή από ένα 9 volt τροφοδοτικό συνεχούς ρεύματος με βύσμα 2.1mm barrel tip. Ο θετικός πόλος θα πρέπει να βρίσκεται στη εσωτερική πλευρά και ο αρνητικός στην εξωτερική πλευρά του βύσματος.



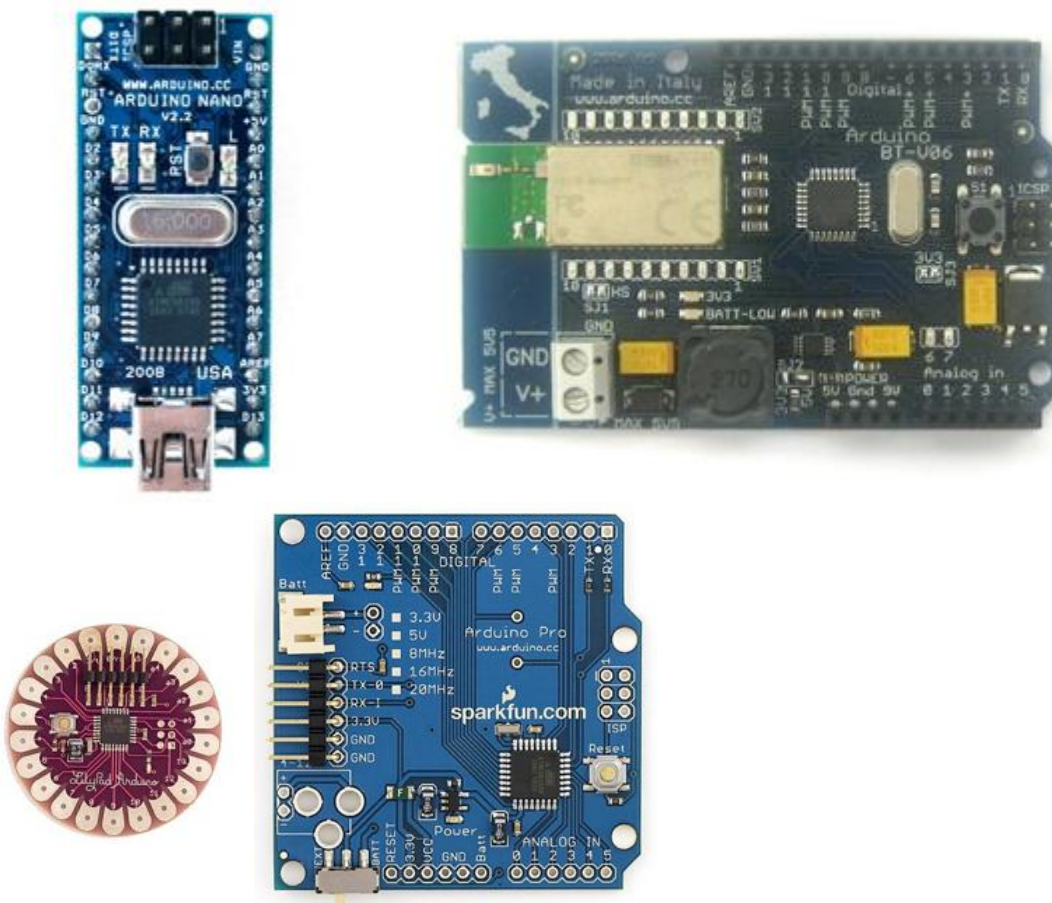
Εικόνα 2.2.3 - Το arduino UNO

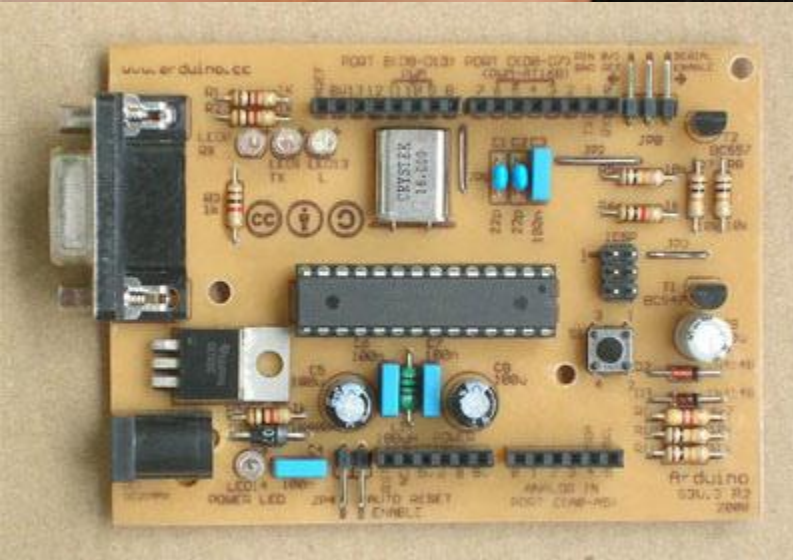
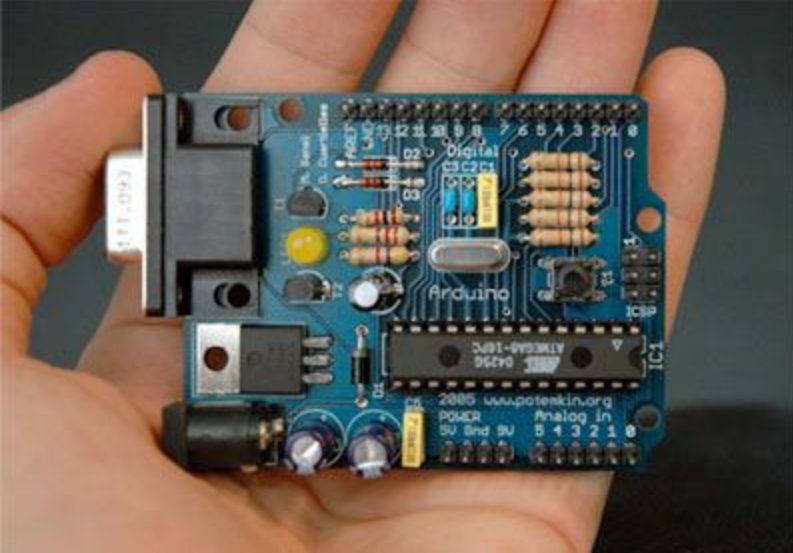
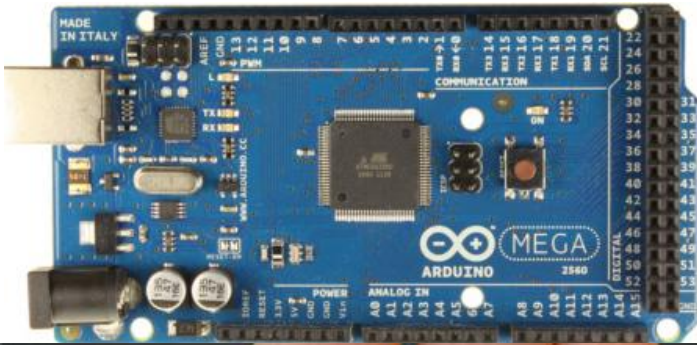
1.2.4 Άλλες εκδόσεις arduino

Σήμερα εκτός από την έκδοση arduino Duemilanove/UNO η οποία και αναλύθηκε παραπάνω, υπάρχουν άλλες οχτώ διαφορετικές πλακέτες arduino.

- Nano: Είναι μία μικρότερη έκδοση του arduino η οποία συνδέεται στον υπολογιστή μέσω καλωδίου mini USB B.
- Bluetooth: Ο ελεγκτής arduino BT περιέχει μία Bluetooth πλακέτα η οποία επιτρέπει την ασύρματη επικοινωνία και προγραμματισμό του μέσω του υπολογιστή.

- LilyPad: Αυτή η έκδοση είναι σχεδιασμένη για να χρησιμοποιείται στα ρούχα. Έχει μοβ χρώμα και μπορεί να ραφτεί εύκολα πάνω σε ύφασμα.
- Pro: Η συγκεκριμένη πλακέτα είναι σχεδιασμένη για προχωρημένους χρήστες που έχουν σκοπό να τη χρησιμοποιήσουν κάπου μόνιμα. Είναι φθηνότερη από την Duemilanove και συνδέεται εύκολα με μπαταρίες αλλά απαιτούνται επιπλέον ηλεκτρονικές διατάξεις για την χρήση της. • Pro-mini: Όπως και στην Pro έκδοση έτσι και σε αυτήν σχεδιάστηκε για προχωρημένους χρήστες με την διαφορά ότι η πλακέτα καταλαμβάνει μικρότερο χώρο. Η Pro-mini είναι και αυτή φθηνότερη έκδοση αλλά χρειάζεται επίσης περαιτέρω εργασία για την χρησιμοποίησή της.
- Serial: Αυτή είναι η βασική έκδοση arduino που χρησιμοποιεί το πρωτόκολλο RS232 για την επικοινωνία και τον προγραμματισμό του. Το πλεονέκτημα της είναι ότι μπορεί εύκολα να κατασκευαστεί από έναν χρήστη.
- Serial Single Sided: η έκδοση αυτή σχεδιάστηκε με σκοπό να κατασκευαστεί στο χέρι. Είναι λίγο μεγαλύτερο από τα προηγούμενα arduino, παρ' όλα αυτά παραμένει συμβατή με τις περισσότερες κατασκευές που σχεδιάστηκαν για να προεκτείνουν της δυνατότητες της Duemilanove.
- Mega: Το Arduino mega όπως και το Duemilanove, συνδέεται μέσω USB και έχει το ίδιο μέγεθος. Η διαφορά τους είναι ότι ο μικροελεγκτής διαθέτει μεγαλύτερη μνήμη και έχει περισσότερες θύρες εισόδου/εξόδου. Εκτός όμως από τις εκδόσεις arduino υπάρχουν και πολλές άλλες κατασκευές που λειτουργούν με παρόμοιο τρόπο όπως το Freduino, το Sanguino, το Bare Bones Board, το LEDuino και το Miduino.





Εικόνα 2.2.4 - Από πάνω αριστερά προς τα κάτω: arduino nano, bluetooth,LiliPad, pro, mega, serial, serial single sided, mini pro.

1.2.5 Το software για τον προγραμματισμό του arduino

Το arduino IDE (Integrated Development Environment -Ολοκληρωμένο Περιβάλλον Ανάπτυξης) είναι ένα ειδικό πρόγραμμα βασισμένο σε java που εκτελείται στον υπολογιστή και επιτρέπει να γραφούν τα sketches για την πλακέτα arduino σε μία απλή γλώσσα που διαμορφώθηκε μέσα από την γλώσσα Processing. Πατώντας ένα κουμπί, το πρόγραμμα φορτώνει το sketch στον μικροελεγκτή του arduino. Η γλώσσα του arduino βασίζεται στη γλώσσα Wiring, μία παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR lib. Λόγω της καταγωγής της από την γλώσσα προγραμματισμού C, στη γλώσσα του arduino μπορούν να χρησιμοποιηθούν ουσιαστικά οι ίδιες βασικές εντολές και συναρτήσεις, με την ίδια σύνταξη, οι ίδιοι τύποι δεδομένων και οι ίδιοι τελεστές με την C. Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του arduino.

Τα βήματα με τα οποία μπορεί να προγραμματιστεί το arduino Duemilanove/UNO είναι:

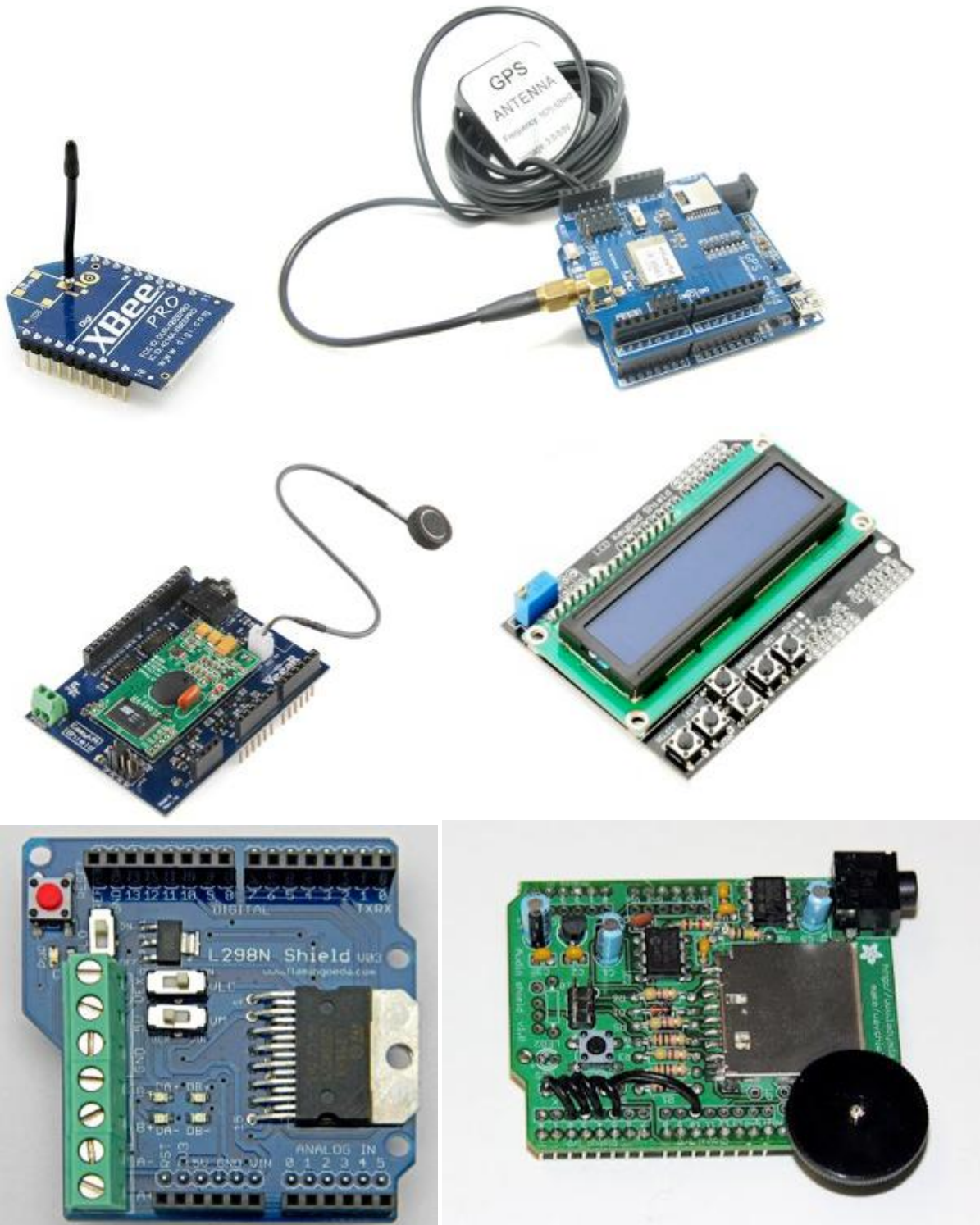
1. Συνδέεται πρώτα η πλακέτα με τον υπολογιστή μέσω USB θύρας
2. Γράφεται ο κώδικας του sketch ο οποίος θα προγραμματίσει τον μικροελεγκτή ώστε να εκτελέσει τις επιθυμητές εργασίες
3. Φορτώνεται το sketch στον μικροελεγκτή του arduino μέσω της USB θύρας. Απαιτούνται μερικά δευτερόλεπτα έως ότου να τελειώσει ο μεταφορά και ο προγραμματισμός του μικροελεγκτή και στη συνέχεια το arduino κάνει μία επανεκκίνηση ώστε ο μικροελεγκτής να διαβάσει τον καινούριο κώδικα.
4. Η πλακέτα μετά την επανεκκίνηση μπορεί να εκτελέσει το καινούριο sketch το οποίο γράφτηκε και φορτώθηκε στον μικροελεγκτή.

1.2.6 Arduino shields

Πέραν όμως της μεγάλης ποικιλίας των πλακετών arduino, υπάρχει και μία μεγάλη ποικιλία από πλακέτες οι οποίες μπορούν να “κουμπώσουν” και να συνδεθούν με την πλακέτα arduino, με σκοπό την προέκταση των δυνατοτήτων της. Κάποιες από αυτές είναι:

- Xbee shield. Το Xbee είναι μία κατασκευή η οποία επιτρέπει σε ένα arduino να επικοινωνήσει ασύρματα με έναν υπολογιστή σε απόσταση έως και 100 μέτρων. Στην πραγματικότητα η ασύρματη αυτή επικοινωνία επιτυγχάνεται από δύο πομποδέκτες. Ο κάθε πομποδέκτης αποτελείται από μία πλακέτα XBee Explorer USB η οποία είναι ένας μετατροπέας της USB θύρας σε σειριακή και ένα Xbee antenna. Το xbee antenna είναι υπεύθυνο για να εκπέμπει και να λαμβάνει σειριακά ηλεκτρικά σήματα τα οποία έχουν διαμορφωθεί σε ηλεκτρομαγνητικά στην συχνότητα των 2.4GHz.

- **Motor Controller.** Η ελεγκτές κινητήρων είναι πλακέτες οι οποίες χρησιμοποιούνται με σκοπό τον έλεγχο των κινητήρων. Εξαιτίας του ότι η πλακέτα arduino δεν έχει την δυνατότητα να τροφοδοτήσει με την απαιτούμενη ισχύ τους κινητήρες, παρουσιάστηκε η ανάγκη κατασκευής κυκλώματος όπου θα «οδηγεί» τους κινητήρες αυτούς.
- **Voice Recognition Shield.** Είναι μία shield η οποία μαζί με το κατάλληλο software έχει την δυνατότητα να αναγνωρίσει μία ποικιλία φωνητικών εντολών που δίνονται από κάποιον χρήστη και να τις προωθήσει για να πραγματοποιήσει το arduino συγκεκριμένες ενέργειες.
- **LCD shield.** Το arduino σε συνδυασμό με μία LCD shield έχει την δυνατότητα να εμφανίσει διάφορα μενού ή μηνύματα σε μία οθόνη. Για παράδειγμα, οι χρήστες μπορούν μέσω της LCD να ενημερώνονται για τα αποτελέσματα που λαμβάνει ένα arduino από τους αισθητήρες που είναι συνδεδεμένοι. Έτσι δε χρειάζεται να συνδεθεί το arduino με τον υπολογιστή για να διαβαστούν τα αποτελέσματα από το serial monitor.
- **GPS shield.** Η συγκεκριμένη πλακέτα επικοινωνεί με τουλάχιστον 3 δορυφόρους και επιστρέφει στο arduino έναν αριθμό μεταβλητών που αντιστοιχούν σε συντεταγμένες.
- **Wave shield.** Είναι και αυτή μία πολύ ενδιαφέρον κατασκευή που δίνει την δυνατότητα στο arduino να αναπαράγει μουσικά αρχεία μορφοποιημένα σε WAVE(.wav).
- **BlinkM.** Το BlinkM είναι ένα RGB led. Στην πραγματικότητα αποτελείται από τρεις διόδους led (red, green, blue) και εκμεταλλεύοντας την PWM δυνατότητα του arduino πραγματοποιεί μία μίξη των τριών βασικών χρωμάτων. Το αποτέλεσμα είναι να εκπέμπει μία πολύ μεγάλη ποικιλία φωτεινών χρωμάτων. Η λίστα των arduino shields είναι αρκετά μεγάλη για να καλύψει τις απαιτητικές ανάγκες των σχεδιαστών για την πραγματοποίηση νέων ιδεών. Παρόλα αυτά όμως συνεχίζονται να σχεδιάζονται και να κατασκευάζονται καινούρια shields από διάφορες εταιρείες, έτσι μέρα με την μέρα οι σχεδιαστές έχουν την δυνατότητα να κατασκευάσουν όλο και περισσότερες πρωτότυπες ιδέες.



Εικόνα 2.2.6 - Arduino Shields από πάνω αριστερά προς τα κάτω: Xbee shield, GPS shield, voice recognition shield, LCD shield , motor shield, wave shield

1.3 Προγραμματισμός του ελεγκτή Arduino

1.3.1 Το περιβάλλον Arduino IDE (integrated development environment)

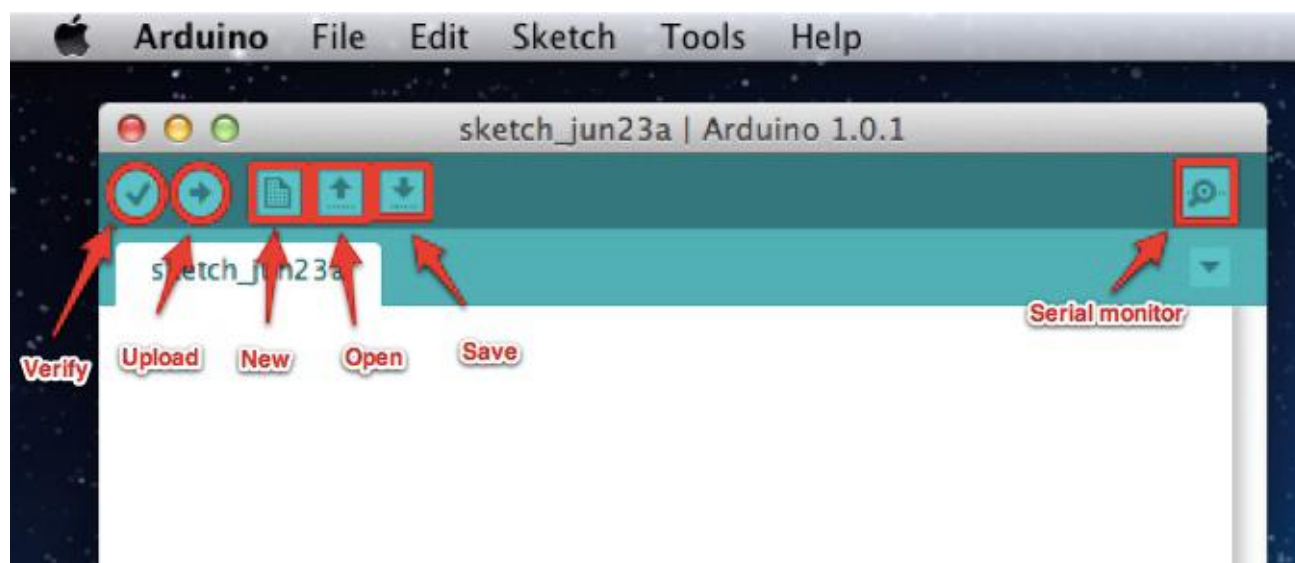
Το Arduino IDE είναι αρκετά απλούστερο σε αντίθεση με άλλα περιβάλλοντα ανάπτυξης λογισμικού όπως το Eclipse , το Xcode και το Visual Studio. Κυρίως αποτελείται από έναν editor (κειμενογράφο), έναν compiler, ένα loader και ένα serial monitor. Δεν περιέχει προχωρημένες λειτουργίες όπως debugging ή code completion, δίνει μόνο τη δυνατότητα για μερικές ρυθμίσεις στα «preferences».



Εικόνα 2.3.1.1 - Το περιβάλλον προγραμματισμού arduino

Στην συνέχεια ακολουθεί η εικόνα του toolbar του arduino IDE που δίνει άμεση πρόσβαση στις λειτουργίες που χρειάζονται περισσότερο για την ανάπτυξη των εφαρμογών που επιθυμεί

να αναπτύξει ο χρήστης.



Εικόνα 2.3.1.2 - Το toolbar του arduino IDE

Με το κουμπί Verify ο χρήστης μπορεί να μεταγλωττίσει (compile) τον κώδικα που βρίσκεται εκείνη την στιγμή στον editor. Το κουμπί verify εκτός από τον συντακτικό έλεγχο, αφού μεταγλωττιστή ο κώδικας στη συνέχεια τον μετατρέπει σε μορφή κατάλληλη για να «φορτωθεί» στον μικροελεγκτή του arduino. Το κουμπί New δημιουργεί ένα καινούριο πρόγραμμα διαγράφοντας οτιδήποτε υπάρχει στον editor. Πριν όμως πραγματοποιήσει αυτήν την ενέργεια, δίνει στον χρήστη την ευκαιρία να αποθηκεύσει το υπάρχον sketch (πρόγραμμα) στην περίπτωση που έχει κάνει κάποιες αλλαγές σε αυτό. Με το κουμπί Open μπορεί ο χρήστης να ανοίξει ένα υπάρχον πρόγραμμα από το σύστημα του. Το κουμπί Save αποθηκεύει τις αλλαγές που έχουν γίνει στο πρόγραμμα που επεξεργάστηκε στον Editor. Το κουμπί Upload όπως και το κουμπί Verify μεταγλωττίζει τον υπάρχον κώδικα στον editor. Με τη διαφορά όμως ότι αφού ελέγξει για τυχόν συντακτικά λάθη και το μετατρέψει σε μορφή κατάλληλη για το arduino, στη συνέχεια θα τον προωθήσει στην θύρα που έχει επιλέξει ο προγραμματιστής από το μενού Tools > Serial Port ώστε να «φορτωθεί» στον μικροελεγκτή.

Ο υπολογιστής μπορεί να επικοινωνήσει με το arduino μέσω σειριακής σύνδεσης. Επιλέγοντας το κουμπί Serial Monitor ανοίγει ένα serial monitor παράθυρο όπου επιτρέπει να παρακολουθεί ο προγραμματιστής τα δεδομένα που στέλνονται προς το arduino αλλά και τα δεδομένα που στέλνονται από το arduino προς τον υπολογιστή. Επιλέγοντας από το μενού το Sketch πέρα από τις λειτουργίες μου αναφέρθηκαν παραπάνω για την εντολή Verify/compile, εμφανίζονται και κάποιες άλλες ενδιαφέρουσες λειτουργίες. Η επιλογή Show Sketch folder είναι μία συντόμευση η οποία ανοίγει σε ένα παράθυρο την διεύθυνση στην οποία το λειτουργικό σύστημα αποθηκεύει τα αρχεία των εφαρμογών. Η επιλογή Add file επιτρέπει στον χρήστη να ανοίξει ένα αρχείο το οποίο μπορεί να βρίσκεται οπουδήποτε μέσα στο σύστημα και να το αποθηκεύσει στον ίδιο φάκελο όπου ανήκει και η εφαρμογή. Τελευταία είναι η επιλογή Import Library. Ο προγραμματιστής έχει την δυνατότητα να εισάγει στο πρόγραμμα που αναπτύσσει οποιαδήποτε βιβλιοθήκη, είτε αυτή είναι του arduino είτε κάποια που δημιούργησε ο ίδιος. Δίπλα από το Sketch στο μενού υπάρχει η επιλογή Tools. Σε αυτή

την στήλη του μενού ο προγραμματιστής έχει την δυνατότητα να επιλέξει την θύρα (serial port) με την οποία θα επικοινωνήσει ο υπολογιστής με το arduino, καθώς επίσης και μια συγκεκριμένη έκδοση arduino έχει (board). Εκτός όμως από αυτές τις δύο βασικές λειτουργίες υπάρχουν και κάποιες άλλες. Το Auto Format το οποίο μορφοποιεί των κώδικα που βρίσκεται στον editor κατάλληλα για να διαβάζεται ευκολότερα. Το Archive Sketch το οποίο μετατρέπει την εφαρμογή που αναπτύσσει ο προγραμματιστής σε ένα αρχείο zip και το αποθηκεύει. Ακόμα σε μερικές περιπτώσεις όταν ένα αρχείο ανοιχθεί στο arduino IDE υπάρχει η περίπτωση να περιέχει χαρακτήρες οι οποίοι δεν είναι ASCII, επιλέγοντας το Fix Encoding & Reload ο κώδικας θα ανανεωθεί αντικαθιστώντας τους περιέργους χαρακτήρες σε χαρακτήρες UTF-8 έκδοση. Τέλος η επιλογή Burning Bootloader απευθύνεται σε προχωρημένους χρήστες όπου τους δίνεται η δυνατότητα να χρησιμοποιήσουν την μνήμη που αντιστοιχεί στον bootloader.

1.3.2 Blinking Led

Μία κοινή τακτική στα βιβλία προγραμματισμού για αρχάριους είναι να δείξουν σε ένα από τα εισαγωγικά κεφάλαια στους αναγνώστες τους πως να εμφανίσουν στην οθόνη τους το μήνυμα «hello world!». Δυστυχώς όμως στον προγραμματισμό των μικροελεγκτών δεν είναι και τόσο εύκολο σε αντίθεση με τις κοινές γλώσσες προγραμματισμού. Έτσι στους μικροελεγκτές συνηθίζεται να δημιουργούν ένα πρόγραμμα το οποίο αναβοσβήνει ένα LED (blinking a LED [1]). Αυτός είναι ο τρόπος με τον οποίο οι μικροελεγκτές λένε «hello world!».

Όπως αναφέρθηκε παραπάνω η πλακέτα arduino έχει ενσωματωμένο ένα LED το οποίο είναι συνδεδεμένο μεταξύ της γείωσης και της θύρας 13. Έτσι χρησιμοποιώντας την συγκεκριμένη θύρα ο προγραμματιστής δε χρειάζεται να συνδέσει τις απαραίτητες ηλεκτρονικές διατάξεις (LED, αντιστάσεις, καλώδια) για να πραγματοποιήσει το ακόλουθο παράδειγμα. Εκτελώντας το arduino IDE σε επίπτωση που ο editor δεν είναι καθαρός θα πρέπει να δημιουργηθεί ένα καινούριο αρχείο. Εκτός από τον τρόπο που περιγράψαμε παραπάνω χρησιμοποιώντας το κουμπί New από το toolbar, μπορεί να γίνει και επιλέγοντας το New από το μενού. Στην συνέχεια αναπτύσσεται και αποθηκεύεται το sketch που βρίσκεται ακριβώς από κάτω δίνοντας του ένα όνομα όπως για παράδειγμα «blinking led».

// παράδειγμα sketch: Blinking LED

```
const int LED = 13; //το LED συνδέεται στο ψηφιακή θύρα 13

void setup()
{
  pinMode(LED, OUTPUT); // η θύρα 13 δηλώνετε ως θύρα έξοδου
}

void loop() {
  digitalWrite(LED, HIGH); // Η θύρα LED πηγαίνει σε κατάσταση ...
```



```

//...high(+5Volt)

delay(1000); // Το πρόγραμμα περιμένει για ένα δευτερόλεπτο

digitalWrite(LED, LOW);//Η θύρα LED πηγαίνει σε κατάσταση low(0Volt)

delay(1000); // Το πρόγραμμα περιμένει για ένα δευτερόλεπτο

}

```

Από την στιγμή που ο κώδικας έχει γράψει στον editor, το επόμενο βήμα είναι να γίνει ο έλεγχος του πατώντας το κουμπί Verify. Στην περίπτωση που όλα είναι σωστά και δεν υπάρχουν συντακτικά λάθη τότε εμφανίζεται στο κάτω μέρος του Arduino IDE το μήνυμα «Done compiling». Το μήνυμα που εμφανίζεται σημαίνει ότι το Arduino IDE τροποποίησε τον κώδικα σε ένα εκτελέσιμο αρχείο το οποίο μπορεί να «τρέξει» στο arduino, κάτι αντίστοιχο με τα .exe αρχεία για τα Windows και τα .app αρχεία για τα Mac Os X. Στην περίπτωση όμως που υπάρχουν λάθη στον κώδικα, τότε στη θέση του μηνύματος «Done compiling» εμφανίζεται ένα αντίστοιχο μήνυμα σφάλματος. Αφού έχει μεταγλωττιστεί σωστά ο κώδικας, μπορεί πλέον πατώντας το κουμπί Upload να «φορτωθεί» στην πλακέτα arduino. Αυτό θα επαναφέρει το arduino στις αρχικές του ρυθμίσεις, αναγκάζοντας το να σταματήσει οποιαδήποτε ενέργεια πραγματοποιούσε μέχρι εκείνη την στιγμή και να «ακούσει» τις οδηγίες που θα λάβει από την USB θύρα. Μόλις ο μικροελεγκτής σταματήσει να λειτουργεί, το arduino IDE στέλνει το sketch που έχει δημιουργήσει ο προγραμματιστής προς την πλακέτα, το οποίο θα αποθηκευτεί στην μνήμη του μικροελεγκτή και τελικά θα εκτελεστεί. Στη συνέχεια θα εμφανιστούν μερικά μηνύματα στην μαύρη περιοχή που βρίσκεται στο κάτω μέρος του arduino IDE και ακριβώς από πάνω το μήνυμα «Done uploading» το οποίο ενημερώνει τον προγραμματιστή ότι η διαδικασία τελείωσε με επιτυχία. Τα δύο LED RX και TX στην πλακέτα αναβοσβήνουν κάθε φορά που ένα byte λαμβάνεται ή στέλνεται από το arduino. Έτσι κατά την διάρκεια της διαδικασίας της «φόρτωσης» του sketch προς τον μικροελεγκτή τα δύο αυτά LED θα πρέπει αναβοσβήνουν συνεχώς.

Εάν δεν παρατηρηθεί να αναβοσβήνουν τα LED ή εμφανιστεί ένα μήνυμα σφάλματος αντί για το «Done uploading» στην μαύρη περιοχή, τότε υπάρχει πρόβλημα επικοινωνίας μεταξύ του υπολογιστή και του arduino. Ο προγραμματιστής θα πρέπει να σιγουρευτεί από το μενού ότι έχει επιλέξει τις σωστές ρυθμίσεις της πλακέτας (tools > boards) και της θύρας (tools > serial port) διαλέγοντας την σωστή έκδοση του arduino και την σωστή θύρα που είναι συνδεδεμένο το arduino με τον υπολογιστή. Από την στιγμή που ο κώδικας «φορτώθηκε» στον μικροελεγκτή τότε αυτός παραμένει εκεί έως ότου να αντικατασταθεί από ένα άλλο sketch. Το υπάρχον sketch διατηρείται στον μικροελεγκτή του arduino μετά από reset (επιαναφορά αρχικών ρυθμίσεων) ή μετά από τη διακοπή της παροχής ρεύματος (turned off), όπως για παράδειγμα τα δεδομένα τα οποία αποθηκεύονται στον σκληρό δίσκο του υπολογιστή. Με την προϋπόθεση ότι όλα έγιναν με επιτυχία τότε το LED «L» που είναι ενσωματωμένο στο Arduino θα πρέπει να αναβοσβήνει ανά ένα δευτερόλεπτο. Αυτό που μόλις δημιουργήθηκε είναι ένα μικρό πρόγραμμα υπολογιστή ή sketch όπως ονομάζονται τα προγράμματα του arduino. Όπως αναφέρθηκε και πριν, το arduino είναι ένας μικρός υπολογιστής και μπορεί να προγραμματιστεί ώστε να κάνει ότι επιθυμεί ο προγραμματιστής.

1.3.3 Επεξήγηση του Sketch

// παράδειγμα sketch: Blinking LED

Τα σχόλια είναι απαραίτητα όταν αναπτύσσονται εφαρμογές, ιδιαίτερα όταν αυτές αποτελούνται από πολλές γραμμές κώδικα. Ο μεταγλωττιστής όταν συναντάει κείμενο το οποίο στα αριστερά του ξεκινάει με “//” το αγνοεί. Αφήνοντας σχόλια στα κυριότερα σημεία του κώδικα, γίνετε ευκολότερος ο τρόπος για να διαβαστεί από έναν προγραμματιστή που τον ανοίγει για πρώτη φορά αλλά και για τον ίδιο τον προγραμματιστή που τον έγραψε όταν τον ξαναδιαβάσει μετά από ένα χρονικό διάστημα.

```
const int LED = 13;
```

const int σημαίνει ότι η LED είναι μία integer μεταβλητή η οποία δεν πρόκειται να αλλάξει ποτέ η τιμή της και θα αντιστοιχεί στον αριθμό 13. Έτσι ενημερώνεται το arduino όπου βλέπει την λέξη LED να την διαβάζει σαν τον ακέραιο αριθμό 13.

```
void setup()
```

Στην γραμμή αυτή δηλώνουμε στο arduino ότι το block που ακολουθεί ονομάζεται setup().

```
{
```

Με αυτήν την ανοιχτή αγκύλη ένα block κώδικα ξεκινάει.

```
pinMode(LED, OUTPUT); //set the digital pin as output
```

Το pinMode λέει στο arduino πως θα διαμορφώσει μία καθορισμένη θύρα (pin). Μία ψηφιακή θύρα μπορεί να χρησιμοποιηθεί ως είσοδος (INPUT) ή ως έξοδος (OUTPUT). Στην συγκεκριμένη περίπτωση θέλουμε μία θύρα εξόδου για να ελέγξουμε το LED, έτσι τοποθετούμε τον αριθμό της θύρας και την κατάσταση της θύρας μέσα στην παρένθεση. Η pinMode είναι μία συνάρτηση και οι λέξεις (ή οι αριθμοί) μέσα στην παρένθεση είναι τα ορίσματα. Η INPUT και η OUTPUT είναι σταθερά ορίσματα της γλώσσας του arduino.

```
}
```

Αυτή η αγκύλη δηλώνει το τέλος της συνάρτησης setup().

```
void loop() {
```

Η συνάρτηση loop() είναι υπεύθυνη για την διαδραστική λειτουργία του μικροελεγκτή. Αυτή θα επαναλαμβάνεται ξανά και ξανά έως ότου να σταματήσει να παρέχεται ρεύμα στην πλακέτα.

```
digitalWrite(LED, HIGH); //turns the LED on
```

Η συνάρτηση ενεργοποιεί η απενεργοποιεί την θύρα η οποία έχει οριστεί ως θύρα εξόδου. Το πρώτο όρισμα (στην συγκεκριμένη περίπτωση LED) ορίζει σε πιο pin (θύρα) του arduino θα πραγματοποιηθεί κάποια ενέργεια (άρα εδώ που το όρισμα είναι το LED απευθύνεται στην θύρα 13 όπως δηλώθηκε στην αρχή του κώδικα). Το δεύτερο όρισμα δηλώνει αν η θύρα θα πάει σε

κατάσταση HIGH (+5 volt) η σε κατάσταση LOW (0 volt). Σε αυτή την περίπτωση η θύρα θα πάει σε κατάσταση HIGH και το LED που βρίσκεται στο pin 13 θα ανάψει.

```
delay(1000); // waits for a second
```

Η delay(argument) είναι μία συνάρτηση στην οποία λέει στο arduino να μη κάνει απολύτως τίποτα και να περιμένει έως ότου να περάσουν όσα milliseconds αντιστοιχούν στο όρισμα (argument). Έτσι το led που βρίσκεται συνδεδεμένο στην θύρα 13 παραμένει αναμμένο έως ότου η κατάσταση του pin αλλάξει από HIGH σε LOW.

```
digitalWrite(LED, LOW); // turns the led off
```

Σε αυτήν την γραμμή κώδικα χρησιμοποιείται η ίδια συνάρτηση με πριν με την διαφορά ότι το δεύτερο όρισμα είναι το LOW. Το αποτέλεσμα θα είναι η θύρα 13 να επιστρέφει στην προηγούμενη κατάσταση της που ήταν η LOW. Άρα στο LED δεν θα παρέχονται πλέον τα +5 volt με αποτέλεσμα να σβήσει.

```
delay(1000); // waits for a second
```

Η συνάρτηση delay καλείται για ακόμη μία φορά και το led θα παραμείνει σβηστό για 1000 milliseconds (1 second).

```
}
```

Η κλειστή αγκύλη δηλώνει το τέλος της συνάρτησης loop() και ο κώδικας που βρίσκεται μέσα σε αυτήν θα επαναλαμβάνεται από την αρχή, έως ότου να σταματήσει να παρέχεται ρεύμα στον ελεγκτή.

1.3.4 Μεταβλητές και τύποι δεδομένων

Όταν δηλώνουμε μία καινούρια μεταβλητή τότε χρειαζόμαστε τουλάχιστον δύο είδη πληροφορίας. Τον τύπο και το όνομα της μεταβλητής. Αυτή η ενέργεια ονομάζεται ορισμός (declaring[2]) μίας μεταβλητής όπου δεσμεύει έναν χώρο στην μνήμη του μικροελεγκτή με σκοπό την αποθήκευση πληροφοριών.

1.3.4.1 Ονομασία μεταβλητών

Για την ονομασία των μεταβλητών υπάρχουν κάποιοι κανόνες που θα πρέπει να τηρηθούν. Πρώτα απ όλα το όνομα μίας μεταβλητής ή συνάρτησης μπορεί να περιέχει γράμματα, αριθμούς, το σύμβολο του δολαρίου και underscores. Κενά και περίεργους χαρακτήρες όπως για παράδειγμα @ ή & δεν επιτρέπονται, όπως επίσης δεν επιτρέπεται και να ξεκινάει με αριθμό το όνομα μίας μεταβλητής. Παράδειγμα επιτρεπτών ονομάτων από μεταβλητές είναι myVariable, mYnAlUe, DELAY_TIME, ενώ τα ακόλουθα ονόματα 1stValue, delay time, και time&money δεν επιτρέπονται. Τα ονόματα των μεταβλητών συνήθως αντιπροσωπεύουν τον λόγο της ύπαρξής τους με σκοπό να είναι πιο κατανοητή η χρήση τους.

1.3.4.2 Τύποι δεδομένων των μεταβλητών

Κάθε δεδομένο το οποίο αποθηκεύεται στο arduino πρέπει να αντιστοιχεί σε ένα τύπο. Ανάλογα τις ανάγκες των δεδομένων ο τύπος μπορεί να είναι ένας από τους παρακάτω.

- Boolean: οι τιμές αυτών των τύπων καταλαμβάνουν μόνο ένα byte στην μνήμη και μπορεί να αντιστοιχεί σε true ή false.
- Char: οι μεταβλητές αυτές καταλαμβάνουν ένα byte στην μνήμη και αποθηκεύουν αριθμούς από το -128 έως το 127. Τις περισσότερες φορές αυτοί οι αριθμοί αντιπροσωπεύουν χαρακτήρες κωδικοποιημένους σε ASCII. Στο παράδειγμα που ακολουθεί, οι μεταβλητές c1 και c2 έχουν τις ίδιες τιμές.

```
Char c1 = 'A';
```

```
Char c2 = 65;
```

- Byte: είναι μεταβλητές που καταλαμβάνουν ένα byte στην μνήμη και παίρνουν τιμές από το 0 έως το 255.
- Int: οι μεταβλητές αυτές καταλαμβάνουν δύο byte χώρο στην μνήμη και αποθηκεύουν τιμές από το -32,768 έως το 32,767.
- Για μεγαλύτερους αριθμούς χρησιμοποιείται η long. Καταλαμβάνει 4 byte στην μνήμη και αποθηκεύει τιμές από το -2,147,483,648 έως το 2,147,483,647.
- Η float και double μεταβλητές είναι ίδιες και μπορούν να χρησιμοποιηθούν για να δηλώσουν αριθμούς κινητής υποδιαστολής. Και οι δύο τύποι καταλαμβάνουν 4 byte στην μνήμη και αποθηκεύουν τιμές από το -3.4028235E+38 έως το 3.4028235E+38.

1.3.4.3 Πίνακες (array)

Ο ορισμός ενός πίνακα είναι παρόμοιος με τον ορισμό μίας μεταβλητής. Η απλούστερη σύνταξη για τον ορισμό ενός πίνακα είναι: «τύπος δεδομένων» «όνομα πίνακα» [«αριθμός στοιχείων»];

Ο τύπος δεδομένων αντιστοιχεί στον τύπο των δεδομένων που θα αποθηκευτούν στον πίνακα ενώ μέσα στις αγκύλες δηλώνεται ο αριθμός των δεδομένων που θα αποθηκευτούν μέσα στον πίνακα. Για παράδειγμα η εντολή `int myArray[3];` θα δημιουργήσει 3 νέα στοιχεία με το όνομα `myArray[0]`, `myArray[1]` και `myArray[2]`. Το πρώτο στοιχείο σε έναν πίνακα βρίσκεται πάντα στην θέση 0. Για να εκχωρηθούν οι τιμές στον πίνακα υπάρχουν δύο τρόποι. Ο πρώτος τρόπος σύνταξης είναι:

```
«όνομα πίνακα»[«θέση στοιχείου»] = «τιμή στοιχείου»
```

Η τιμή που βρίσκεται στα δεξιά του τελεστή εκχώρησης αποθηκεύεται στην θέση του πίνακα που ορίζεται μέσα στις αγκύλες. Ο δεύτερος τρόπος σύνταξης είναι:

```
«τύπος δεδομένων» «όνομα πίνακα»[] = {«τιμή 1ου στοιχείου, τιμή 2ου στοιχ ... »};
```

Ο τρόπος αυτής της εκχώρησης χρησιμοποιείται μόνο κατά τον ορισμό του πίνακα. Μέσα στις αγκύλες {} περιέχονται τα στοιχεία χωρισμένα με κόμμα που πρόκειται να αποθηκευτούν μέσα στον πίνακα, το πρόγραμμα μετράει τα στοιχεία και γνωρίζει τον αριθμό των θέσεων του πίνακα. Όταν καλείται το όνομα του πίνακα και μέσα στις αγκύλες [] ο αριθμός μίας θέσης του, τότε επιστρέφει η τιμή που είναι αποθηκευμένη στην συγκεκριμένη θέση. Ένα string είναι ένας πίνακας από char μεταβλητές. Η σύνταξη ενός char πίνακα μπορεί να διαφέρει λίγο. Στο παράδειγμα που ακολουθεί όλοι οι πίνακες έχουν το ίδιο περιεχόμενο.

```
char string1[8] = { 'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0' };
```

```
char string2[] = "Arduino";
```

```
char string3[8] = "Arduino";
```

```
char string4[] = { 65, 114, 100, 117, 105, 110, 111, 0 };
```

Τα string πρέπει πάντα να τερματίζονται με ένα μηδενικό byte. Όταν χρησιμοποιούνται double quotes για την δημιουργία ενός string τότε το μηδενικό byte προστίθεται αυτόματα. Αυτός είναι και ο λόγος που πρέπει να προσθέτουμε ακόμη μία μονάδα στο μέγεθος των περιεχομένων του πίνακα.

1.3.4.4 Qualifiers μεταβλητών

Όταν δηλωθεί μία μεταβλητή μπορούν να χρησιμοποιηθούν τα variable qualifiers. Σκοπός τους είναι η παραλλαγή της συμπεριφοράς μίας μεταβλητής για να αλλάξουν κάποιες λειτουργίες στον τρόπο που χρησιμοποιείται. Η const ή constant μετατρέπει μία μεταβλητή σε κατάσταση read-only (μόνο ανάγνωση) και δεν μπορεί να αλλάξουν οι πληροφορίες της από την στιγμή που οριστεί. Για παράδειγμα:

```
const int red = 9;
```

η μεταβλητή red θα περιέχει τον ακέραιο αριθμό 9 και δε θα μπορεί να αλλάξει την τιμή αυτή ποτέ ξανά. Για να μετατραπεί το εύρος των τιμών που παίρνουν οι μεταβλητές ενός τύπου δεδομένων χρησιμοποιείται το unsigned. Το unsigned είναι και αυτό ένα variable qualifier το οποίο επιτρέπει στις μεταβλητές να παίρνουν μόνο θετικές τιμές. Έτσι αφού η μεταβλητή δεν περιμένει ποτέ να πάρει αρνητικό πρόσημο, διπλασιάζει το εύρος των θετικών τιμών. Για παράδειγμα:

```
unsigned int maxThreshold = 1024;
```

Πιο συγκεκριμένα τοποθετώντας τη λέξη unsigned μπροστά από μία integer μεταβλητή κατά τον ορισμό της, τότε αυτή η μεταβλητή θα παίρνει μόνο θετικές τιμές μεταξύ του 0 και του 65,535.

1.3.4.5 Προκαθορισμένες μεταβλητές

Όπως ο χρήστης δηλώνει κάποιες μεταβλητές μόνο για ανάγνωση (constant) έτσι και το περιβάλλον ανάπτυξης arduino εφαρμογών έχει κάποιες δικές του μεταβλητές με προκαθορισμένες τιμές. Οι πρώτες από αυτές τις προκαθορισμένες μεταβλητές είναι η true και η false βασισμένες από την boolean λογική και τους λογικούς πίνακες αληθείας. Η κατάσταση false εύκολα μπορεί να οριστεί ως μηδέν ή μερικές φορές ως off. Από την άλλη πλευρά η κατάσταση true είναι οτιδήποτε άλλο εκτός του false όπως για παράδειγμα 1, «on» ή κάποιος άλλος αριθμός εκτός του μηδενός. Ένα άλλο ζευγάρι προκαθορισμένων μεταβλητών είναι το INPUT και το OUTPUT. Αυτές οι δύο μεταβλητές καθαρίζουν τον τρόπο λειτουργίας μίας ψηφιακής θύρας (digital pin) χρησιμοποιώντας την συνάρτηση pinMode. Επίσης υπάρχουν ακόμη δύο προκαθορισμένες μεταβλητές η HIGH και η LOW όπου κατά προσέγγιση αντιπροσωπεύουν τις τιμές +5 και 0 volt ή on και off. Χρησιμοποιούνται με σκοπό να δηλώσουν την κατάσταση μίας θύρας.

1.3.4.6 Εμβέλεια μεταβλητών

Στις προηγούμενες παραγράφους περιγράφηκε ο τρόπος δήλωσης μίας μεταβλητής, ένα άλλο επίσης σημαντικό είναι το μέρος στο οποίο θα δηλωθεί αυτή. Η τοποθεσία που θα οριστεί μία μεταβλητή δηλώνει και την εμβέλεια της. Αυτό είναι γνωστό και ως variable scope. Οι μεταβλητές στο παράδειγμα blinking led δηλώθηκαν έξω από τις συναρτήσεις setup() και loop() στην αρχή του κώδικα. Αυτές που δηλώνονται εκτός κάποιας συνάρτησης είναι γνωστές και ως global μεταβλητές (variables) και μπορούν να χρησιμοποιηθούν σε όλο το sketch. Σε αντίθεση, αν κάποια μεταβλητή έχει δηλωθεί μέσα σε μία συνάρτηση για παράδειγμα μέσα στην setup(), τότε αυτή θα είναι διαθέσιμη μόνο μέσα στο block που δηλώθηκε και δε θα μπορούσε να χρησιμοποιηθεί έξω από αυτό. Αυτές είναι γνωστές ως local μεταβλητές (variables).

1.3.5 Εκχώρηση τιμών

Για να γίνει η εκχώρηση τιμών στις μεταβλητές χρησιμοποιείται ο τελεστής εκχώρησης «=». Αυτός είναι το μαθηματικό σύμβολο της ισότητας αλλά δεν λειτουργεί με τον ίδιο ακριβώς τρόπο όπως στα μαθηματικά. Ο τελεστής λέει στον compiler να πάρει τις τιμές που βρίσκονται στα δεξιά του και να τις εκχωρήσει στην μεταβλητή που βρίσκεται στα αριστερά του.

1.3.6 Μαθηματικοί τελεστές

Υπάρχουν πέντε βασικοί τελεστές που χρησιμοποιούνται για να κάνουν τις βασικές αριθμητικές πράξεις.

Πρώτος τελεστής είναι αυτός της πρόσθεσης «+». Έτσι αν επιθυμεί ο προγραμματιστής να προσθέσει δύο στοιχεία και να εκχωρήσει σε μία μεταβλητή το αποτέλεσμα τους, θα το κάνει κάπως έτσι:

```
myValue = 4 + 38;
```

Εκκινώντας από τα δεξιά το arduino θα προσθέσει το 4 με το 38 και θα εκχωρήσει το άθροισμα τους στην μεταβλητή myValue που βρίσκεται στα αριστερά του τελεστή εκχώρησης.

```
myValue = 255;
```

```
newValue = myValue - 128;
```

Ο τελεστής της αφαίρεσης «-» στο παραπάνω παράδειγμα θα αφαιρέσει από την μεταβλητή myValue η οποία ισούται με το 255 τον αριθμό 128 και το αποτέλεσμα θα το εκχωρήσει στη μεταβλητή newValue.

```
float pi = 3.14159;
```

```
int diameter = 5;
```

```
float C;
```

```
C = pi * diameter;
```

Στην συγκεκριμένη περίπτωση ο τελεστής του πολλαπλασιασμού «*» αγνοεί τον τύπο της μεταβλητής diameter που είναι integer και πολλαπλασιάζει τις δύο μεταβλητές μεταξύ τους σαν να είναι float. Στην συνέχεια εκχωρεί το αποτέλεσμα στην μεταβλητή C.

```
int myValue;
```

```
myValue = 9 / 5;
```

Ο τελεστής διαίρεσης «/» πραγματοποιεί μία διαίρεση μεταξύ των τελεστών και επιστρέφει το αποτέλεσμα. Στο παραπάνω παράδειγμα ο τελεστής διαίρεσης θα κάνει την διαίρεση και θα επιστρέψει την τιμή 1, εξαιτίας του ότι η αριθμητική πράξη γίνεται μεταξύ ακέραιων μεταβλητών. Έτσι η μεταβλητή myValue θα περιέχει την τιμή 1 αντί για 1.8 στην περίπτωση που οι μεταβλητές ήταν αριθμοί κινητής υποδιαστολής. Τέλος ο τελεστής υπολοίπου «%» επιστρέφει το υπόλοιπο μίας διαίρεσης μεταξύ ακέραιων τελεστών. Η έκφραση δηλαδή 9%5 θα επέστρεφε την τιμή 0.8 επειδή είναι το υπόλοιπο της διαίρεσης.

1.3.7 Σύνθετοι τελεστές

1.3.7.1 Τελεστές προσαύξησης και μείωσης

Μία συνηθισμένη εργασία είναι να προστίθεται ή να αφαιρείται το 1 από μία ακέραια μεταβλητή.

```
myValue = myValue +1;
```

Το παραπάνω παράδειγμα θα μπορούσε να αντικατασταθεί χρησιμοποιώντας έναν σύνθετο τελεστή. Στην περίπτωση δηλαδή που θέλουμε να αυξήσουμε μία μεταβλητή κατά ένα, για συντομία θα

μπορούσε να γραφεί το όνομα της μεταβλητής και ο τελεστής ++. Η γραμμή κώδικα που ακολουθεί θα είχε ακριβώς το ίδιο αποτέλεσμα με την παραπάνω.

```
myValue++;
```

Στην περίπτωση που η μεταβλητή έπρεπε να μειωθεί κατά 1, θα χρησιμοποιούνταν ο αντίστοιχος τελεστής --. Η σύνταξη της εντολής θα ήταν:

```
myValue--;
```

Αυτοί οι δύο τελεστές χρησιμοποιούνται μόνο για να αυξήσουν ή να μειώσουν μία μεταβλητή κατά μία μονάδα. Αν ο προγραμματιστής επιθυμεί να αυξήσει ή να μειώσει την μεταβλητή με αριθμό μεγαλύτερο από το 1 τότε μπορεί να χρησιμοποιήσει τους σύνθετους τελεστές εκχώρησης +=, -=, *=, /=. Για παράδειγμα αν θέλει να αυξήσει μία μεταβλητή κατά 10 μπορεί να εκτελέσει την ακόλουθη εντολή:

```
myValue +=10;
```

Η δήλωση αυτή είναι η ίδια με την `myValue = myValue + 10;`. Δηλαδή πρόσθεσε την μεταβλητή `myValue` με το 10 και στη συνέχεια εκχωρεί το αποτέλεσμα της πρόσθεσης στην ίδια μεταβλητή. Επίσης μπορεί η μεταβλητή να πολλαπλασιαστεί με έναν αριθμό και το αποτέλεσμα να αποθηκευτεί στην ίδια. Για παράδειγμα:

```
myValue *=1,5;
```

Η μεταβλητή `myValue` πολλαπλασιάζεται με το 1.5 και το αποτέλεσμα εκχωρείται πάλι στην `myValue`, έτσι αν η αρχική τιμή της ήταν 10, τώρα θα ισούται με το 15. Παρόμοια λειτουργούν και οι υπόλοιποι σύνθετοι τελεστές εκχώρησης -=, /=.

1.3.7.2 Συσχετιστικοί τελεστές

Το `arduino` μπορεί να χρησιμοποιηθεί για την λήψη αποφάσεων, έτσι αν ο προγραμματιστής θέλει να ελέγξει κατά ποσό ή όχι μία μεταβλητή ανέκτησε την επιθυμητή τιμή, θα πρέπει να χρησιμοποιήσει μία έκφραση σαν αυτή που ακολουθεί.

```
myValue == 5
```

Στο παράδειγμα αυτό χρησιμοποιείται ο τελεστής συσχέτισης `==`, ο οποίος ελέγχει εάν μία μεταβλητή ή τιμή που βρίσκεται στο αριστερό μέρος είναι ίση με μία μεταβλητή ή τιμή που βρίσκεται στο δεξιό μέρος. Στην περίπτωση του παραδείγματος η έκφραση θα πρέπει να επιστρέψει `true` εάν η μεταβλητή `myValue` περιέχει την τιμή 5, ενώ `false` αν περιέχει διαφορετική τιμή. Ένας παρόμοιος τελεστής σύγκρισης που επιστρέφει ακριβώς τα αντίθετα αποτελέσματα με τον προηγούμενο είναι το `!=`. Σύμφωνα με αυτόν τον τελεστή όταν το αριστερό στοιχείο είναι διαφορετικό από το δεξιό τότε επιστρέφει `true`, ενώ όταν έχουν την ίδια τιμή επιστρέφει `false`.

Άλλοι συσχετιστικοί τελεστές είναι το `>` και το `<`. Ο τελεστής `>` επιστρέφει `true` όταν συγκρίνει δύο στοιχεία και το αριστερό είναι μεγαλύτερο από το δεξιό στοιχείο. Αντίθετα ο τελεστής `<` επιστρέφει `true` όταν το αριστερό στοιχείο είναι μικρότερο από το δεξιό. Στις περιπτώσεις που τα στοιχεία είναι ίσα μεταξύ τους τότε επιστρέφεται η τιμή `false`. Τέλος υπάρχει ακόμη ένα ζευγάρι τελεστών

σύγκρισης . Ο τελεστής \geq ο οποίος επιστρέφει true στην περίπτωση που το αριστερό στοιχείο είναι μεγαλύτερο ή ίσο με το δεξιό και false εάν το αριστερό είναι μικρότερο. Ο τελεστής \leq επιστρέφει την τιμή true εάν το αριστερό στοιχείο είναι μικρότερο ή ίσο με το δεξιό στοιχείο, σε διαφορετική περίπτωση θα επιστρέψει false.

1.3.7.3 Λογικοί τελεστές

Οι συγκριτικοί τελεστές δουλεύουν ιδανικά όταν θέλουμε να συγκρίνουμε δύο τιμές μεταξύ τους, υπάρχουν όμως περιπτώσεις που απαιτούν πιο πολύπλοκες συγκρίσεις. Σε τέτοιες περιπτώσεις χρησιμοποιούνται οι λογικοί τελεστές. Για παράδειγμα όταν υπάρχουν δύο ξεχωριστές συγκρίσεις και αυτές θα πρέπει να κάνουν κάτι μόνο στην περίπτωση που είναι και οι δύο αληθείς. Τότε η έκφραση θα πρέπει να γραφεί κάπως έτσι.

```
myValue >= 10 && myValue <=20
```

Η παραπάνω έκφραση χρησιμοποιεί το λογικό AND «&&» και επιστρέφει true μόνο όταν οι δύο εκφράσεις στα αριστερά και στα δεξιά του λογικού τελεστή επιστρέψουν true. Άρα για να συμβεί αυτό, η τιμή της μεταβλητής myValue θα πρέπει να είναι μεγαλύτερη του 9 και μικρότερη του 21. Μόνο όταν η τιμή της μεταβλητής βρίσκεται μέσα στο εύρος τιμών 10 έως 20 θα επιστρέψει true. Όταν ο προγραμματιστής όμως επιθυμεί να επιστραφεί η τιμή true στην περίπτωση που τουλάχιστον μία από τις δύο συνθήκες είναι true τότε θα πρέπει να χρησιμοποιήσει το λογικό OR «||». Η έκφραση αυτή θα επιστρέψει true όταν μία ή και οι δύο από τις εκφράσεις που βρίσκονται αριστερά και δεξιά του λογικού τελεστή επιστρέψουν true. Για παράδειγμα:

```
myValue < 10 || myValue > 20
```

Τέλος υπάρχει και ο λογικός τελεστής NOT (!). Ο τελεστής αυτός αντιστρέφει την τιμή μίας έκφρασης boolean. Δηλαδή στο επόμενο παράδειγμα:

```
!myValue
```

θα επιστρέψει true μόνο όταν η μεταβλητή myValue είναι false. Στην περίπτωση που η μεταβλητή είναι true τότε η έκφραση θα επιστρέψει false.

1.3.8 Σειρά προτεραιότητας των τελεστών

Όταν οι πράξεις γίνονται πιο πολύπλοκες τότε ακολουθείται μία σειρά προτεραιότητας για τις πράξεις. Πιο αναλυτικά, πρώτα πραγματοποιούνται οι πράξεις οι οποίες βρίσκονται μέσα στις παρενθέσεις ή αγκύλες «[], ()». Στην συνέχεια πραγματοποιούνται οι πράξεις με τους τελεστές μοναδιαίας αύξησης ++ και μείωσης --. Αμέσως μετά, την προτεραιότητα έχουν οι πράξεις του πολλαπλασιασμού, τις διαίρεσης και του υπολοίπου (*, /, %) ενώ μετά σειρά έχουν οι προσθέσεις και οι αφαιρέσεις (+, -). Αφού τελειώσουν οι αριθμητικές πράξεις πραγματοποιούνται οι συγκριτικές εντολές και στην συνέχεια οι λογικές. Τελευταίοι σε προτεραιότητα είναι οι τελεστές εκχώρησης δηλαδή το «=» καθώς και οι σύνθετοι (+=, -=, *=, /=).

1.3.9 Εντολές ελέγχου

Υπάρχουν δύο ειδών εντολών διαθέσιμες στο arduino C που ελέγχουν την ροή του προγράμματος. Οι εντολές υπό συνθήκη (conditional statement) οι οποίες παίρνουν μία απόφαση ανάλογα με το εάν ή όχι μία συνθήκη επιστρέφει true και οι επαναληπτικές καταστάσεις (iterative statement) οι οποίες επαναλαμβάνουν ένα block εντολών αρκετές φορές έως ότου μία συνθήκη επιστρέψει false. Στις καταστάσεις υπό συνθήκη περιλαμβάνονται η if – else και η switch εντολές ελέγχου, οι οποίες επιτρέπουν την εκτέλεση ορισμένων ενεργειών σύμφωνα με την κατάσταση της συνθήκης. Στις επαναλαμβανόμενες καταστάσεις περιλαμβάνονται η for και η while, οι οποίες συχνά αποκαλούνται και loops (βρόχοι) επειδή όπως και η συνάρτηση loop() επαναλαμβάνουν ότι βρίσκεται μέσα στο βρόχο έως ότου να επιστρέψει false η υπεύθυνη συνθήκη.

1.3.9.1 Εντολή if else

Η εντολή if είναι η απλούστερη μορφή ελέγχου και από τις σημαντικότερες εντολές του arduino για την λήψη αποφάσεων. Η βασική σύνταξη είναι:

```
if(συνθήκη){  
  
εντολές  
  
}
```

Η λέξη if ακολουθείται από ένα ζευγάρι παρενθέσεων (), και μέσα σε αυτές βρίσκεται η συνθήκη. Για παράδειγμα εάν το πρόγραμμα πρέπει να τρέξει μία εντολή όταν η μεταβλητή myValue ισούται με το 5, η εντολή θα είναι:

```
if (myValue == 5)
```

Στην περίπτωση που η συνθήκη επιστρέψει true τότε θα εκτελεστεί η άμεσος επόμενη εντολή ή το block που ακολουθεί μέσα στις αγκύλες {}. Μία κοινή χρήση της συνθήκης if είναι ο έλεγχος μίας ψηφιακής θύρας του arduino και η εκτέλεση μίας εντολής σύμφωνα με την συνθήκη που βρίσκεται. Όταν όμως η συνθήκη if επιστρέψει false τότε η εντολή ή το block που ακολουθεί θα παρακαμφθούν. Στην περίπτωση αυτή ίσως ο προγραμματιστής επιθυμεί να εκτελέσει μία άλλη εντολή, τότε μπορεί να χρησιμοποιηθεί προαιρετικά η εντολή else. Παρακάτω φαίνεται ο τρόπος σύνταξης της if μαζί με την else.

```
If(συνθήκη){  
  
εντολή  
  
}  
  
else {
```

εντολή

}

Έτσι στην περίπτωση που η συνθήκη της if είναι μη αληθής τότε θα εκτελεστεί η έκφραση ή το block που βρίσκεται ακριβώς μετά την else. Αντίστοιχα όταν η συνθήκη της if είναι αληθής τότε η εντολή else αγνοείται.

1.3.9.2 Εντολή switch

Στη περίπτωση που χρησιμοποιηθούν πολλές συνθήκες if με σκοπό να καλυφθεί ένα εύρος τιμών ελέγχου μίας μεταβλητής, τότε το πρόγραμμα γίνεται πολύπλοκο. Η εντολή switch μπορεί να απλοποιήσει αλλά και να επιταχύνει την διαδικασία εκτέλεσης των εντολών. Η σύνταξη της είναι:

```
switch(μεταβλητή){  
  
case «περίπτωση τιμής μεταβλητής»:  
  
εντολή 1;  
  
case «περίπτωση τιμής μεταβλητής 2»:  
  
εντολή 2;  
  
case «περίπτωση τιμής μεταβλητής 3»:  
  
εντολή 3;  
  
...  
  
...  
  
default :  
  
εντολή χ;  
  
}
```

Μετά την λέξη switch αναφέρεται το όνομα της μεταβλητής που θα ελεγχθεί μέσα στις παρένθεσης (). Στην συνέχεια μέσα στο block καταγράφονται οι περιπτώσεις (cases) της μεταβλητής, έτσι όταν η τιμή της μεταβλητής αντιστοιχίσει με μία περίπτωση τότε εκτελείται η αντίστοιχη εντολή. Όταν όμως η μεταβλητή δεν αντιστοιχίσει σε κάποια περίπτωση, τότε εκτελείται η default εντολή. Η default μπορεί να παραληφθεί. Μετά το τέλος των εντολών κάθε περίπτωσης μπορεί να προστεθεί η εντολή break;

```
switch(μεταβλητή){
```

case « περίπτωση τιμής μεταβλητής»:

εντολή 1;

break;

...

}

Η συγκεκριμένη εντολή προκαλεί άμεση έξοδο από την switch. Έτσι αν μετά την εκτέλεση μίας περίπτωσης δεν υπάρχει λόγος να συνεχίσει να συγκρίνει την μεταβλητή με άλλες περιπτώσεις, τότε χρησιμοποιώντας την εντολή break; μπορεί το πρόγραμμα να διαφύγει από το block της switch.

1.3.10 Βρόχοι

1.3.10.1 For

Ο βρόχος for επιτρέπει στο arduino να επαναλάβει εκτελέσιμες γραμμές κώδικα που βρίσκονται μέσα σε block για συγκεκριμένο αριθμό. Αυτό που κάνει την εντολή for μοναδική είναι ότι βασίζεται σε έναν μετρητή ο οποίος αυξάνεται κάθε φορά που ο βρόχος επαναλαμβάνεται. Ο μετρητής έχει την δυνατότητα να χρησιμοποιηθεί και μέσα στον βρόχο όπως μία τοπική μεταβλητή. Η βασική σύνταξη του βρόχου for είναι: for(ορισμός μεταβλητής ; έλεγχος μεταβλητής ; αύξηση μεταβλητής) {εντολές

}

Η αρχή του βρόχου αποτελείται από τρία μέρη, τον ορισμό μίας μεταβλητής , την συνθήκη που θα ελέγξει τη μεταβλητή και τον τρόπο με τον οποίο θα αυξάνεται η μεταβλητή. Στο πρώτο μέρος ορίζουμε ή αρχικοποιούμε την μεταβλητή, η εντολή αυτή εκτελείται μόνο την πρώτη φορά. Στο δεύτερο μέρος έχουμε μία λογική έκφραση η οποία επιστρέφει μία τιμή boolean. Αν ο έλεγχος επιστρέψει true τότε συνεχίζεται η εκτέλεση του βρόχου, διαφορετικά σταματάει. Στο τρίτο μέρος αυξάνει την τιμή της μεταβλητής κάθε φορά που ολοκληρώνεται η εκτέλεση των εντολών του βρόχου. Ένα παράδειγμα του βρόχου for είναι:

```
for(int i = 0; i <5; i++){
```

```
.....
```

```
}
```

Στο πρώτο μέρος ορίζουμε μία καινούρια integer μεταβλητή και της εκχωρούμε την τιμή 0

```
(int i = 0;)
```

Στην συνέχεια ελέγχουμε αν η μεταβλητή «i» είναι μικρότερη από το 5 για να συνεχιστεί η εκτέλεση του βρόχου (i < 5;)

Ενώ στο τρίτο μέρος αυξάνουμε την τιμή της μεταβλητής κατά μία μονάδα, κάθε φορά που ο βρόχος επαναλαμβάνεται (i++)

Έτσι το εσωτερικό που βρόχου θα εκτελεστεί 5 φορές εάν δεν υπάρξει κάποια προσαύξηση ή μείωση της μεταβλητής «i» μέσα στο block.

1.3.10.2 While

Ο βρόχος while χρησιμοποιείται για να επαναλάβει ένα block για όσο χρόνο μία συγκεκριμένη συνθήκη είναι true. Ακολουθεί η σύνταξη του βρόχου while:

```
while(συνθήκη) {  
  
εντολές  
  
}
```

Το προηγούμενο παράδειγμα θα μπορούσε να ξαναγραφεί χρησιμοποιώντας τη white:

```
int i = 0;  
  
while( i < 5 ){  
  
...  
  
i++;  
  
}
```

Στην πρώτη γραμμή ορίζεται η integer μεταβλητή $i = 0$. Στην επόμενη γραμμή δημιουργείται ο βρόχος while και συγκρίνει αν η μεταβλητή είναι μικρότερη του 5. Στην περίπτωση που η συνθήκη επιστρέψει true θα εκτελεστεί ο κώδικας μέσα στο block, ενώ στην τελευταία γραμμή του block θα αυξηθεί η τιμή της μεταβλητής κατά μία μονάδα. Τέλος θα επιστρέψει στην αρχή του βρόχου while, θα επαναλάβει τον έλεγχο της συνθήκης και μέχρι αυτή να επιστρέψει false θα επαναλαμβάνει την εκτέλεση των εντολών μέσα στο block.

1.3.10.3 Do

Οι βρόχοι for και while αξιολογούν μία συνθήκη πριν εκτελέσουν τις εντολές που βρίσκονται μέσα στο block. Στην περίπτωση όμως που δεν ικανοποιείται η συνθήκη τότε το block δε θα τρέξει ποτέ. Μερικές φορές είναι χρήσιμο ο κώδικας που βρίσκεται μέσα στο block να εκτελείται τουλάχιστον μία φορά. Αυτό επιτυγχάνεται χρησιμοποιώντας το βρόχο do, ο οποίος ενεργεί σαν μία while με την

διαφορά ότι πρώτα εκτελεί τον κώδικα που βρίσκεται μέσα στο block και στην συνέχεια ελέγχει αν η συνθήκη επιστρέφει true με σκοπό να τον επαναλάβει. Η βασική της σύνταξη είναι:

```
do {  
  
εντολές  
  
} while( συνθήκη);
```

1.3.10.4 Continue

Η εντολή continue; χρησιμοποιείται μέσα σε βρόχους με σκοπό την έναρξη της επόμενης επανάληψης του. Δηλαδή όταν το πρόγραμμα διαβάσει την εντολή continue; θα σταματήσει την εκτέλεση των εντολών που ακολουθούν μέσα στο βρόχο και θα επιστρέψει στην αρχή. Εκεί θα ελέγξει αν η συνθήκη συνεχίσει να επιστρέφει true ώστε να επαναλάβει τον κώδικα μέσα στο βρόχο ακόμη μία φορά.

1.3.11 Συναρτήσεις

1.3.11.1 Βασικές συνάρτησης setup & loop

Τα προγράμματα που αναπτύσσονται για να εφαρμοστούν στο arduino εκτός της βασικής βιβλιοθήκης arduino απαιτούν και δύο συναρτήσεις, την setup() και την loop(). Κάθε sketch πρέπει να περιλαμβάνει αυτές τις δύο συναρτήσεις ώστε να μπορέσει να τρέξει η εφαρμογή, έστω και αν είναι άδειες. Το επόμενο παράδειγμα είναι τεχνικά ένα ολοκληρωμένο sketch που τρέχει κανονικά, αν και δεν κάνει απολύτως καμία ενέργεια.

```
void setup(){  
  
}  
  
void loop(){  
  
}
```

Η setup() καλείται μόνο μία φορά κατά την έναρξη του sketch και χρησιμοποιείται με σκοπό να πραγματοποιήσει διάφορες εντολές , όπως είναι η έναρξη μίας σειριακής επικοινωνίας με μία σειριακή συσκευή, η εκχώρηση αρχικών τιμών σε αισθητήρες ή να εκτελέσει ορισμένες εργασίες που απαιτούνται για την ομαλή λειτουργία του προγράμματος. Η loop() δίνει την δυνατότητα στο sketch να τρέχει επαναλαμβανόμενα έως ότου να σταματήσει η παροχή του ρεύματος στο arduino. Η έναρξη της σηματοδοτείται με το που εκτελέσει όλες τις εντολές τις η setup(). Με το που θα

εκτελεστεί η τελευταία εντολή της συνάρτησης loop(), τότε το πρόγραμμα θα ξεκινήσει και πάλι να εκτελεί τις εντολές της loop() από την πρώτη γραμμή του block.

1.3.11.2 Δημιουργία νέων συναρτήσεων

Σε ένα sketch, ο προγραμματιστής εκτός από τις ήδη υπάρχουσες συναρτήσεις που βρίσκονται στις βιβλιοθήκες, του δίνεται η δυνατότητα να χρησιμοποιήσει και δικές του. Αν στο πρόγραμμα επαναλαμβάνονται κάποιες γραμμές κώδικα, τότε ο προγραμματιστής θα μπορούσε να δημιουργήσει μία νέα συνάρτηση και να συμπεριλάβει μέσα σε αυτήν τον επαναλαμβανόμενο κώδικα. Έτσι κάθε φορά που απαιτούνται οι συγκεκριμένες γραμμές κώδικα, αντί να τις γράφει επανειλημμένα μπορεί απλά και εύκολα να καλεί την συνάρτηση που δημιούργησε. Για την δημιουργία μίας συνάρτησης απαιτείται να δηλωθούν ο τύπος της τιμής που θα επιστρέφει, το όνομα της συνάρτησης και το σύνολο των παραμέτρων που απαιτούνται για την εκτέλεση των εντολών μέσα στη συνάρτηση. Ο ορισμός μίας συνάρτησης τηρεί το ακόλουθο σχήμα:

```
«τύπος επιστρεφόμενης τιμής» «όνομα συνάρτησης» («λίστα παραμέτρων») {  
  
«εντολές συνάρτησης»  
  
}
```

Η κάθε συνάρτηση που προστίθεται στο sketch πρέπει να δημιουργείται έξω από τις υπόλοιπες συναρτήσεις που υπάρχουν. Τις περισσότερες φορές δεν παίζει κάποιο ρόλο σε πιο ακριβώς σημείο θα τοποθετηθούν αλλά συνηθίζεται οι καινούριες συναρτήσεις να μπαίνουν αμέσως μετά το τέλος της συνάρτησης loop().

1.3.11.3 Καλώντας τις συναρτήσεις

Μία συνάρτηση καλείται όταν ο προγραμματιστής θέλει να εκτελέσει ένα σύνολο ομαδοποιημένων εντολών που βρίσκονται μέσα σε αυτήν. Η συνάρτηση καλείται με την ακόλουθη μορφή:

```
«όνομα συνάρτησης» («παράμετροι»);
```

Όταν καλείται η συνάρτηση τότε το πρόγραμμα πηδάει από το σημείο που καλείται στο σημείο που δημιουργήθηκε για να εκτελέσει της εντολές που βρίσκονται μέσα σε αυτήν. Μόλις το πρόγραμμα τελειώσει όλες τις εντολές που βρίσκονται μέσα στην συνάρτηση, τότε θα συνεχίσει από το σημείο που συνάντησε το κάλεσμα της συνάρτησης.

1.3.11.4 Επιστροφές τιμών συνάρτησης

Σε αρκετές περιπτώσεις ο σκοπός της συνάρτησης είναι να εκτελέσει διάφορες πράξεις και να επιστρέψει ένα αποτέλεσμα σε μία κατάλληλη μορφή ώστε το πρόγραμμα να την χρησιμοποιήσει κάπου αλλού. Η ενέργεια του καλέσματος μίας συνάρτησης και η επιστροφή μίας τιμής από την ίδια

συνάρτηση είναι γνωστή ως function return. Στην δήλωση της συνάρτησης το πρώτο πράγμα που χρειάζεται να δηλωθεί είναι ο τύπος της τιμής που θα επιστραφεί. Επίσης μέσα στην συνάρτηση θα χρησιμοποιηθεί η εντολή return «όνομα μεταβλητής»; η οποία επιστρέφει την τιμή της μεταβλητής στο σημείο που καλέστηκε η συνάρτηση. Ο τύπος της συνάρτησης θα πρέπει να είναι ο ίδιος με τον τύπο της μεταβλητής που επιστρέφεται. Στην περίπτωση που η συνάρτηση δε επιστρέφει καμία τιμή τότε η συνάρτηση δηλώνεται ως void.

1.3.11.5 Παράμετροι συνάρτησης

Οι παράμετροι συνάρτησης είναι ένα είδος σαν τις μεταβλητές αλλά δηλώνονται μέσα στις παρενθέσεις όπου ορίζεται η συνάρτηση και χωρίζονται με κόμμα. Αν απαιτούνται κάποιες μεταβλητές για τις πράξεις μέσα στην συνάρτηση, των οποίων οι εμβέλεια τους δεν επιτρέπει την χρήση τους, μπορούν να δηλωθούν ως παράμετροι συνάρτησης. Έτσι κατά το κάλεσμα της συνάρτησης μέσα στις παρενθέσεις μπορούν να εισαχθούν τα απαιτούμενα στοιχεία και στην συνέχεια με το που διαβάσει το πρόγραμμα το όρισμα της συνάρτησης, αντιστοιχεί τα στοιχεία με νέες μεταβλητές τοπικής εμβέλειας.

Ένα παράδειγμα δημιουργίας και κάλεσμα συνάρτησης:

```
void loop(){  
  
int a = 5;  
  
int b = 4;  
  
int c;  
  
c = add(a, b);  
  
}  
  
int add(int num1, int num2){  
  
return num1+num2;  
  
}
```

Στην πρώτη σειρά εκτελείται επανειλημμένα η συνάρτηση loop. Μέσα στο block της loop, ορίζονται τρεις τοπικές μεταβλητές a b c και εκχωρούνται οι τιμές 5, 4 στις δύο πρώτες. Αφού οριστούν οι μεταβλητές μετά καλείται η συνάρτηση add με παραμέτρους τις μεταβλητές a, b. Έτσι η τιμή που θα επιστραφεί από την συνάρτηση add() θα εκχωρηθεί στην μεταβλητή c. Όταν το πρόγραμμα συναντήσει το κάλεσμα της συνάρτησης αμέσως πηγαίνει στο σημείο όπου ορίζεται. Η συνάρτηση που καλείται ορίζεται ως int επειδή θα επιστρέψει έναν ακέραιο αριθμό. Το όνομα της είναι add και μέσα στην παρένθεση θα δημιουργήσει δύο νέες μεταβλητές τοπικής εμβέλειας. Στις μεταβλητές αυτές (num1 και num2) θα καταχωρηθούν οι τιμές που αντιστοιχούν στις παραμέτρους που βρίσκονται στο κάλεσμα της συνάρτησης. Στην τελευταία σειρά επιστρέφεται η τιμή που αντιστοιχεί

στο άθροισμα των δύο τοπικών μεταβλητών. Έτσι η συνάρτηση add επιστρέφει μία τιμή που στην συνέχεια εκχωρείται στην μεταβλητή c.

1.3.11.6 Ψηφιακές συναρτήσεις

1.3.11.6.1 pinMode()

Πριν χρησιμοποιηθεί μία ψηφιακή θύρα (digital pin) θα πρέπει πρώτα το arduino να ενημερωθεί με ποιον τρόπο, δηλαδή εάν η θύρα θα είναι εισόδου ή εξόδου. Για να γίνει αυτό καλείται η συνάρτηση pinMode() η οποία ενημερώνει στον μικροελεγκτή τον αριθμό της θύρας που να χρησιμοποιηθεί καθώς και την κατάσταση της (εισόδου/εξόδου). Η σύνταξη είναι αρκετά απλή:

```
pinMode(«αριθμός θύρας», «κατάσταση»);
```

Μέσα στις παρενθέσεις υπάρχουν δύο παράμετροι οι οποίες χωρίζονται με κόμμα. Η πρώτη αντιστοιχεί στον αριθμό της θύρας που ο προγραμματιστής επιθυμεί να χρησιμοποιήσει. Η τιμή αυτή μπορεί να κυμαίνεται από το 0 έως το 13 ή από το A0 μέχρι το A5 στην περίπτωση που επιθυμεί να χρησιμοποιήσει της αναλογικές εισόδους ως ψηφιακές εισόδου/εξόδου. Η δεύτερη τιμή ορίζει την κατάσταση που θα εργάζεται στο κύκλωμα η συγκεκριμένη θύρα. Αυτό επιτυγχάνεται χρησιμοποιώντας τις δύο προκαθορισμένες από το arduino σταθερές, την INPUT και την OUTPUT. Όταν η θύρα ορίζεται ως INPUT, τότε δέχεται εισερχόμενα σήματα και καταλαβαίνει μόνο δύο καταστάσεις, την HIGH και την LOW. Αντίστοιχα όταν η θύρα ορίζεται ως OUTPUT τότε έχει την δυνατότητα να παρέχει στην έξοδο έως και 40 milli amps σε ένα άλλο κύκλωμα, ισχύς που είναι αρκετή για να ανάψει ένα LED.

1.3.11.6.2 digitalWrite()

Αν μία θύρα έχει οριστεί ως OUTPUT τότε η ρύθμιση της εξόδου της σε on(+5Volt) ή off (0Volt) κατάσταση πραγματοποιείται χρησιμοποιώντας την συνάρτηση digitalWrite(). Η σύνταξη της είναι:

```
digitalWrite(«αριθμός θύρας», «κατάσταση θύρας»);
```

Εδώ όπως και στην pinMode() υπάρχουν δύο παράμετροι που απαιτούνται για την συνάρτηση. Η πρώτη δηλώνει τον αριθμό της θύρας που παίρνει τιμές από 0 έως 13 ή από A0 έως A5. Η δεύτερη τιμή αντιστοιχεί στις δύο καταστάσεις που μπορεί να ρυθμιστεί η έξοδος χρησιμοποιώντας δύο προκαθορισμένες μεταβλητές του arduino, την HIGH και την LOW. Η κατάσταση HIGH παρέχει στην έξοδο την +5VDC τάση της πλακέτας ενώ η LOW συνδέει το κύκλωμα που βρίσκεται στην έξοδο της θύρας με την γείωση (0 Volt).

1.3.11.6.3 digitalRead()

Όταν μία ψηφιακή θύρα ρυθμιστεί έως INPUT (εισόδου) τότε για να διαβαστεί η είσοδος της χρησιμοποιείται η συνάρτηση digitalRead().

Η σύνταξη της είναι: digitalRead(«αριθμός θύρας»);

Η συγκεκριμένη συνάρτηση χρησιμοποιεί μόνο μία παράμετρο που αντιστοιχεί στον αριθμό της θύρας εισόδου. Η συνάρτηση σύμφωνα με την είσοδο της, επιστρέφει μία τιμή HIGH ή LOW. Η τιμή αυτή μπορεί να αποθηκευτεί ή να συγκριθεί με μία άλλη μεταβλητή.

1.3.11.7 Αναλογικές συναρτήσεις

Οι αναλογικές συναρτήσεις είναι λίγο διαφορετικές από τις ψηφιακές. Σε αντίθεση από τις ψηφιακές, δε χρειάζεται το arduino να ενημερωθεί για τον αριθμό της αναλογικής θύρας που πρόκειται να χρησιμοποιηθεί όπως συμβαίνει με την συνάρτηση pinMode(). Επιπλέον αντί για τιμές HIGH και LOW χρησιμοποιούν ακέραιους αριθμούς εύρους από το 0 έως το 1023 για τις τιμές εισόδου και από 0 έως 255 για τις τιμές εξόδου.

1.3.11.7.1 analogRead()

Η analogRead() λειτουργεί παρόμοια με την digitalWrite. Η σύνταξη της συνάρτησης είναι:

```
analogRead(«αριθμός θύρας»);
```

Όταν καλείται η παραπάνω συνάρτηση, απαιτείται να προσδιοριστεί ο αριθμός της αναλογικής θύρας από όπου αναμένεται να διαβαστούν τα δεδομένα. Η παράμετρος της analogRead παίρνει τιμές από το A0 έως το A5. Το γράμμα A μπαίνει μπροστά από τον αριθμό της θύρας με σκοπό να υπενθυμίσει τον προγραμματιστή ότι πρόκειται για Analog θύρα. Αφού η συνάρτηση διαβάσει το εισερχόμενο σήμα τότε επιστρέφει μία τιμή χωρητικότητας 10 bit που αντιπροσωπεύει έναν ακέραιο αριθμό εύρους από το 0 έως το 1023. Η τιμή που επιστρέφεται αντιστοιχεί στην στάθμη του αναλογικού σήματος εισόδου της θύρας.

1.3.11.7.2 analogWrite()

Η συνάρτηση analogWrite() επιτρέπει την πρόσβαση στο pulse width modulation hardware του μικροελεγκτή στο arduino. Η βασική σύνταξη της συνάρτησης είναι:

```
analogWrite(«αριθμός θύρας», «κατάσταση παλμού»);
```

Χρησιμοποιώντας αυτή την συνάρτηση απαιτούνται δύο παράμετροι. Η πρώτη παράμετρος ορίζει τον ρυθμό της αναλογικής θύρας εξόδου. Για το arduino duemilanove/UNO η πρώτη παράμετρος μπορεί να πάρει μόνο τις ακόλουθες τιμές 3, 5, 6, 9, 10 και 11 οι οποίες αντιστοιχούν σε θύρες που έχουν την ένδειξη PWD πάνω στην πλακέτα. Άλλες εκδόσεις arduino μπορεί να έχουν περισσότερες ή λιγότερες PWD θύρες. Η δεύτερη παράμετρος ορίζει την κατάσταση του παλμού η οποία εκφράζεται με έναν integer αριθμό εύρους από το 0 έως το 255 καταλαμβάνοντας χωρητικότητα 8 bit. Όταν η τιμή ισούται με το 0 τότε ο παλμός βρίσκεται στη κατάσταση off ή 0% ή 0 volt καθ' όλη την διάρκεια της περιόδου. Ανάλογα όταν η τιμή ισούται με το 255 τότε σε αυτήν την περίπτωση ο παλμός βρίσκεται στη κατάσταση on ή 100% ή 5 volt καθ' όλη την διάρκεια της περιόδου. Αν όμως η τιμή βρίσκεται στο μέσο μεταξύ των τιμών 0 και 255 τότε σε κάθε περίοδο ο παλμός θα βρίσκεται

στην κατάσταση off κατά την μίση διάρκεια της περιόδου και στην υπόλοιπη διάρκεια στην κατάσταση on.

1.3.11.7.3 analogReference()

Χρησιμοποιώντας την συνάρτηση analogRead(), το arduino είναι προετοιμασμένο να δεχτεί ένα εύρος τάσης από 0 έως +5 volt. Όμως όλο και περισσότεροι αισθητήρες χρησιμοποιούν χαμηλότερη τάση λειτουργίας. Έτσι όταν ένας σένσορας στέλνει μέγιστη τιμή μικρότερη των 5 volt τότε το arduino (που περιμένει μέγιστη τιμή 5 volt) θα μεταφράσει λανθασμένα την πληροφορία του αισθητήρα. Ο μικροελεγκτής της ATMEL έχει την δυνατότητα να αλλάξει την τάση αναφοράς καλώντας την συνάρτηση analogReference(). Η σύνταξη της συνάρτησης είναι:

```
analogReference(«επιλογή αναφοράς»);
```

Μέσα στις παρενθέσεις ορίζεται το σημείο αναφοράς που θα έχει ο μικροελεγκτής. Υπάρχουν τρεις επιλογές τιμής αναφοράς οι οποίες είναι:

- **DEFAULT.** Επιλέγοντας την προκαθορισμένη τιμή, το arduino χρησιμοποιεί σαν τάση αναφοράς τα 5 volt (ή τα 3.3 volt ανάλογα τον τύπο του arduino).
- **External.** Η πλακέτα του arduino διαθέτει ένα pin δίπλα από το pin 13 που ονομάζεται AREF. Έτσι όταν καλείται η συνάρτηση analogReference(External); η θύρα AREF χρησιμοποιείται με σκοπό ο μικροελεγκτής να διαβάσει μία εξωτερική τάση αναφοράς διαφορετική από την προκαθορισμένη που είναι τα 5 volt.
- **INTERNAL.** Δηλώνοντας την παράμετρο INTERNAL, το arduino χρησιμοποιεί ως τάση αναφοράς τα 1,1 volt. Η συνάρτηση αυτή εκτελείται μόνο μία φορά στο sketch πριν την χρήση της συνάρτησης analogRead(). Συνηθίζεται να καλείται μέσα στην συνάρτηση setup().

1.3.12 map()

Το εύρος της τιμής που επιστρέφει μία αναλογική θύρα είναι αρκετά μεγάλο (0 έως 1023). Το γεγονός αυτό πολλές φορές δυσκολεύει τον έλεγχο της τιμής εισόδου, έτσι απαιτείται η μείωση του εύρους της με σκοπό την διευκόλυνση της επεξεργασία της. Ένας άλλος λόγος μείωσης του εύρους είναι η εξοικονόμηση αποθηκευτικού χώρου που απαιτεί μία μεταβλητή από την μνήμη. Η συνάρτηση map() επιστρέφει μία νέα τιμή σύμφωνα με το επιθυμητό εύρος. Η σύνταξη της είναι: map(«αρχική τιμή», «ελ. τιμή αρχικού εύρους», «μεγ. τιμή αρχικού εύρους», «ελ. τιμή νέου εύρους», «μεγ. τιμή νέου εύρους»); Η map() όταν καλείται απαιτεί 5 παραμέτρους. Η πρώτη αντίστοιχη σε μία τιμή που ανήκει στο αρχικό επιτρεπτό εύρος τιμών, και επιθυμείτε να μετατραπεί σε μία νέα τιμή σύμφωνα με ένα νέο εύρος. Η δεύτερη και η τρίτη παράμετρος ορίζει την ελάχιστη και την μέγιστη επιτρεπτή τιμή του αρχικού εύρους. Η τέταρτη και η πέμπτη παράμετρος ορίζει την ελάχιστη και τη μέγιστη τιμή του νέου εύρους τιμών. Η map() στην ουσία επιστρέφει μία νέα τιμή η οποία ανήκει σε ένα νέο εύρος τιμών και είναι αντίστοιχη της αρχικής τιμής (πρώτη παράμετρος) που ανήκει στο αρχικό εύρος τιμών.

1.3.13 constrain()

Η συνάρτηση constrain() ελέγχει μία τιμή εάν βρίσκεται μέσα στα όρια ενός εύρους τιμών. Στην περίπτωση που η τιμή δεν ανήκει μέσα στα όρια των επιτρεπτών τιμών τότε επιστρέφεται η ελάχιστη ή η μέγιστη επιτρεπτή τιμή, ανάλογα εάν η τιμή είναι μικρότερη ή μεγαλύτερη από τα επιθυμητά όρια. Όταν όμως η τιμή ανήκει μέσα στα όρια τότε επιστρέφεται χωρίς κάποια αλλαγή. Η σύνταξη της είναι:

```
constrain(«τιμή», «ελαχ. επιτρεπτή τιμή», «μεγ. επιτρεπτή τιμή»);
```

Η συνάρτηση χρησιμοποιεί τρεις παραμέτρους. Η πρώτη παράμετρος είναι η τιμή που θα ελεγχθεί. Η δεύτερη είναι το κατώτατο και η τρίτη το ανώτατο όριο της επιτρεπόμενης τιμής. Η συνάρτηση επιστρέφει πάντα μία τιμή που ανήκει μέσα στα όρια του ορίστηκαν στην συνάρτηση.

1.3.14 Προχωρημένες συναρτήσεις

Μέχρι τώρα εκτός από τη δομή και την σύνταξη ενός arduino προγράμματος που είναι νβασισμένο στη γλώσσα προγραμματισμού C αναφέρθηκαν και οι βασικές συνάρτησης που χρησιμοποιούνται για το «διάβασμα» και το «γράψιμο» ψηφιακών και αναλογικών δεδομένων εισόδου και εξόδου. Εκτός όμως από αυτές τις συναρτήσεις, η βιβλιοθήκη του arduino περιλαμβάνει και άλλες εξίσου σημαντικές και χρήσιμες οι οποίες θα αναλυθούν στην συνέχεια.

1.3.14.1 Συναρτήσεις Timing

Επειδή ο μικροελεγκτής του arduino μπορεί να «κινηθεί» πιο γρήγορα απ' όσο τουλάχιστον μπορεί να αντιληφθεί ο άνθρωπος, είναι αρκετές φορές σκόπιμο να επιβραδυνθούν οι ενέργειες που πραγματοποιεί. Αυτό επιτυγχάνεται καλώντας τις κατάλληλες συναρτήσεις.

1.3.14.1.1 delay()

Η συνάρτηση delay δημιουργεί μία μικρή παύση στην ροής του προγράμματος. Η σύνταξη της είναι:

```
delay(«χρόνος»);
```

Η delay έχει μία παράμετρο η οποία αντιστοιχεί στον χρόνο που θα «παγώσει» την ροή του προγράμματος. Η μονάδα μέτρησης του χρόνου που χρησιμοποιείται είναι τα milliseconds. Έτσι όταν το πρόγραμμα καλέσει την εντολή delay(1000); τότε η ροή του θα σταματήσει για 1 δευτερόλεπτο, αφού 1 second ισούται με 1000 milliseconds.

1.3.14.1.2 delayMicroseconds()

Στην περίπτωση που η delay() κάνει μεγάλης διάρκειας παύσεις τότε μπορεί να χρησιμοποιηθεί η συνάρτηση delayMicroseconds(). Όπως και στην delay() έτσι και η delayMicroseconds() συνάρτηση έχει ακριβώς την ίδια σύνταξη με την διαφορά ότι η παράμετρος του χρόνου αντιστοιχεί σε microseconds. Ένα second ισούται με 1,000,000 microseconds.

1.3.14.1.3 millis()

Μέσα στον μικροελεγκτή της πλακέτας arduino υπάρχουν τρεις on-board hardware μετρητές (timers) οι οποίοι εκτελούνται πίσω από το κύριο πρόγραμμα με σκοπό να χειριστούν επαναλαμβανόμενες εργασίες όπως είναι η αύξηση της τιμής ενός χρονόμετρου. Η συνάρτηση millis() χρησιμοποιεί έναν από τους τρεις hardware timers που περιέχει τον χρόνο από το σημείο που ο microcontroller τέθηκε σε λειτουργία, είτε μετά από διακοπή ρεύματος, είτε μετά από reset. Η σύνταξη της δεν περιέχει κάποια παράμετρο, έτσι καλώντας απλά την συνάρτηση millis(); αυτή θα επιστρέφει μία τιμή σε milliseconds. Η τιμή αυτή μπορεί να εκχωρηθεί σε μία μεταβλητή και να χρησιμοποιηθεί όπως μία οποιαδήποτε άλλη μεταβλητή.

1.3.14.1.4 macros()

Όπως η συνάρτηση millis() επιστρέφει τον χρόνο λειτουργίας του μικροελεγκτή σε milliseconds, έτσι και η συνάρτηση macros() κάνει ακριβώς το ίδιο πράγμα με την διαφορά ότι η μονάδα μέτρησης του χρόνου που επιστρέφετε είναι σε microseconds.

1.3.14.2 Συναρτήσεις random

Πολλές φορές στο πρόγραμμα απαιτείται η δημιουργία τυχαίων αριθμών, όπως για παράδειγμα για την κατασκευή ενός ηλεκτρονικού ζαριού. Το arduino παρέχει την δυνατότητα χρησιμοποίησης ψευδο-τυχαίων (semi-random) αριθμών καλώντας τις απαιτούμενες συναρτήσεις.

1.3.14.2.1 random()

Η συνάρτηση random() επιστρέφει μία ψευδο-τυχαία (semi-random) τιμή, σύμφωνα με τις παραμέτρους. Αν δεν υπάρχουν παράμετροι τότε θα επιστρέψει μία τιμή τύπου long εύρους από το -2,147,483,648 έως το 2,147,483,647. Η κανονική σύνταξη της είναι:

```
random(«ελάχιστη τιμή», «μέγιστη τιμή»);
```

Η πρώτη παράμετρος ορίζει την ελάχιστη επιτρεπτή τιμή που μπορεί να επιστρέψει, ενώ αντίστοιχα η δεύτερη παράμετρος την μέγιστη. Στην περίπτωση που η ελάχιστη τιμή είναι το μηδέν τότε η πρώτη παράμετρος μπορεί να παραληφθεί. Έτσι το πρόγραμμα όταν δει μόνο μία παράμετρο,

καταλαβαίνει ότι η ελάχιστη επιτρεπτή τιμή είναι το μηδέν ενώ η μέγιστη ισούται με την παράμετρο της συνάρτησης.

1.3.14.2.2 randomSeed()

Όπως αναφέρθηκε προηγούμενος οι τιμές του επιστρέφει η random() δεν είναι ακριβώς τυχαίες. Το γεγονός αυτό οφείλεται στον σχεδιασμό του arduino που είναι ένα ντετερμινιστικό σύστημα. Έτσι όταν χρησιμοποιείται η συνάρτηση, η φόρμουλα που παράγει τους τυχαίους αριθμούς επαναλαμβάνεται κάθε φορά που εκτελείται. Με αποτέλεσμα οι τιμές να είναι προβλεπόμενες αφού στην πραγματικότητα δεν είναι τυχαίες. Για να αποφευχθεί η ανάκτηση των ίδιων ψευδο-τυχαίων αριθμών χρησιμοποιείται η συνάρτηση randomSeed(), η

σύνταξη της είναι:

```
randomSeed(«τιμη»);
```

Τροφοδοτώντας την συνάρτηση με μία τιμή, μπορεί να κάνει την παραγωγή ψευδο-τυχαίων αριθμών αρκετά πιο απρόβλεπτη. Η τιμή μπορεί να αντιστοιχεί σε έναν integer ή ένα long αριθμό που ορίζεται μέσα στο πρόγραμμα. Για να δημιουργηθούν ακόμη πιο τυχαίοι αριθμοί, αντί να οριστεί η τιμή, μπορεί κάλλιστα να εισαχθεί από μία αναλογική θύρα (analog pin) χρησιμοποιώντας έναν αισθητήρα. Για παράδειγμα:

```
randomSeed(analogRead(A0));
```

1.3.15 Hardware Διακοπές

Οι λόγοι για να ζητήσουμε από το πρόγραμμα να διακόψει την ροή της εκτέλεσης του είναι πολλοί. Ένας λόγος είναι ότι η συνάρτηση loop() μπορεί να περιέχει πολλές γραμμές κώδικα με αποτέλεσμα να χρειάζεται αρκετό χρόνο για μία πλήρη επανάληψη των εντολών που περιέχει, έτσι είναι πιθανών όταν πατηθεί ένα button να μην γίνει αντιληπτή η ενέργεια. Ένα άλλο παράδειγμα είναι όταν χρησιμοποιείται ένας φωτοηλεκτρικός αισθητήρας ο οποίος αντιλαμβάνεται ένα αντικείμενο όταν πλησιάζει. Οι αποφάσεις του arduino για τις εντολές που πρέπει να δώσει στους κινητήρες θα πρέπει να ληφθούν άμεσα και με ακρίβεια. Η συνάρτηση attachInterrupt() ενεργοποιεί μία διακοπή της κανονικής ροής του κώδικα που εκτελεί το arduino με σκοπό να τρέξει μία συγκεκριμένη συνάρτηση. Η συνάρτηση καλείται όταν σκανδαλιστεί ο μικροελεγκτής σε ένα συγκεκριμένο pin. Η σύνταξη της είναι:

```
attachInterrupt(«διακόπτης», «συνάρτηση», «κατάσταση»);
```

Η πρώτη παράμετρος ορίζει τον διακόπτη που θα σκανδαλίσει τον microcontroller. Στο arduino UNO υπάρχει η δυνατότητα για την χρήση δύο διακοπών, συγκεκριμένα ο αριθμός 0 δηλώνει ότι ο διακόπτης αντιστοιχεί στο pin 2 και ο αριθμός 1 αντιστοιχεί στο pin 3. Η δεύτερη παράμετρος δηλώνει την συνάρτηση που θα εκτελεστεί όταν καλεστεί η attachInterrupt(). Τέλος η τρίτη παράμετρος δηλώνει με πιο τρόπο θα σκανδαλίσει ο διακόπτης τον μικροελεγκτή. Υπάρχουν τέσσερις πιθανές καταστάσεις.

1. LOW: σκανδαλίζει όταν ο διακόπτης είναι σε κατάσταση LOW(0 volt).

2. CHANGE: σκανδαλίζει όταν ο διακόπτης αλλάζει κατάσταση, είτε από LOW (0 volt) πηγαίνει σε HIGH (+5 volt) κατάσταση, είτε το αντίθετο.

3. RISING: σκανδαλίζει όταν ο διακόπτης από την κατάσταση LOW αλλάζει σε κατάσταση HIGH.

4. FALLING: σκανδαλίζει όταν ο διακόπτης από την κατάσταση HIGH πηγαίνει στη κατάσταση LOW.

Κατά την διάρκεια εκτέλεσης του προγράμματος και ενώ είναι ενεργοποιημένη η συνάρτηση διακοπής, δίνεται η δυνατότητα αλλαγής της κατάστασης με την οποία θα σκανδαλίζεται η διακοπή. Καλώντας την συνάρτηση detachInterrupt() θα απενεργοποιηθεί η συνάρτηση διακοπής. Έτσι αφού διακοπεί, η attachInterrupt() μπορεί να ξαναεκτελεστεί ορίζοντας τον επιθυμητό τρόπο σκανδαλισμού. Η συνάρτηση detachInterrupt() έχει μία παράμετρο που ορίζει τον διακόπτη που θα απενεργοποιηθεί, 0 για το διακόπτη στην θέση pin 2 και 1 για το διακόπτη στη θέση pin 3.

1.3.16 Σειριακή βιβλιοθήκη arduino

Το arduino χρησιμοποιείται σε πολλές stand-alone εφαρμογές ελέγχου οι οποίες δεν έχουν μεγάλες απαιτήσεις για την επεξεργασία δεδομένων, έτσι οι δυνατότητες του μικροελεγκτή είναι αρκετές για τον πλήρη έλεγχο. Σε άλλες όμως επιπτώσεις απαιτείται μεγαλύτερη υπολογιστική δύναμη με αποτέλεσμα να χρειάζεται την βοήθεια ενός υπολογιστή. Για να επιτευχθεί η επικοινωνία του arduino με τον υπολογιστή γίνεται χρήση της σειριακής βιβλιοθήκης. Η συγκεκριμένη βιβλιοθήκη εισάγεται αυτόματα στο sketch μαζί με την κανονική βιβλιοθήκη του arduino χωρίς να απαιτείται κάποια περαιτέρω εντολή στο πρόγραμμα όπως συμβαίνει με τις υπόλοιπες βιβλιοθήκες. Όταν καλείται η Serial τότε μία instance της Serial βιβλιοθήκης δημιουργείται. Έτσι οπότε καλείται μία από τις μεθόδους που ανήκουν στην Serial, θα προηγείται πάντα το όνομα της instance «Serial».

1.3.16.1 begin()

Με την Serial βιβλιοθήκη να περιλαμβάνεται στο sketch και μία instance που ονομάζεται Serial να έχει ήδη δημιουργηθεί, το πρώτο πράγμα που χρειάζεται να πραγματοποιηθεί για να μπορούν να χρησιμοποιηθούν οι Serial μέθοδοι είναι να καθοριστεί η ταχύτητα μεταφοράς δεδομένων. Για να επιτευχθεί αυτό καλείται η μέθοδος begin(), η σύνταξη της είναι:

```
Serial.begin(«ρυθμός δεδομένων»);
```

Η παράμετρος αντιστοιχεί στο μέτρο ταχύτητας μεταφοράς bit ανά δευτερόλεπτο ανάμεσα στο arduino και τον υπολογιστή και μπορεί να πάρει μία από τις ακόλουθες τιμές 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ή 115200. Γενικά χρησιμοποιείται η τιμή 9600 που είναι αρκετά κοινή και γρήγορη ταχύτητα μεταξύ συσκευών σειριακής επικοινωνίας για την μεταφορά δεδομένων.

1.3.16.2 available()

Η μέθοδος `available()` δεν έχει κάποια παράμετρο αλλά επιστρέφει τον αριθμό των διαθέσιμων byte που περιμένουν στο σειριακό buffer. Η hardware σειριακή θύρα του arduino περιέχει ένα buffer το οποίο μπορεί να αποθηκεύσει μέχρι και 128 byte πληροφορίες. Στην περίπτωση που καμία πληροφορία δεν περιέχεται στο buffer τότε η μέθοδος επιστρέφει την τιμή 0. Η σύνταξη της είναι:

```
Serial.available()
```

1.3.16.3 read()

Από την στιγμή που το πρόγραμμα γνωρίζει ότι το buffer περιέχει πληροφορίες, χρησιμοποιείται η μέθοδος `read()` για να τις διαβάσει. Η σύνταξη της είναι:

```
Serial.read()
```

Η `read()` επιστρέφει το πρώτο διαθέσιμο byte του buffer, στην πραγματικότητα διαβάζει μία ASCII τιμή από το 0 έως το 127 που αντιστοιχεί σε έναν ASCII χαρακτήρα. Τα byte λαμβάνονται ένα ένα κάθε φορά μέσω σειριακού hardware.

1.3.16.4 print()

Η μέθοδος `print()` χρησιμοποιείται για να εμφανίσει ASCII χαρακτήρες στην συσκευή που είναι σειριακά συνδεδεμένοι με το arduino. Η σύνταξη της είναι:

```
Serial.print(«δεδομένα»);
```

Η παράμετρος που περιέχει τα δεδομένα μπορεί να δηλωθεί με διάφορους τρόπους.

1.3.16.5 println()

Η μέθοδος `println()` όπως και η `print()` χρησιμοποιείται για την εμφάνιση χαρακτήρων. Έχουν την ίδια σύνταξη και η μονή τους διαφορά είναι ότι κάθε φορά που εμφανίζει τα δεδομένα η `println()` στη συνέχεια προσθέτει έναν ειδικό χαρακτήρα νέας σειράς «\n».

1.3.17 Εισαγωγή βιβλιοθήκης

Για να χρησιμοποιηθεί μία βιβλιοθήκη πρέπει να ενημερωθεί το arduino πια είναι αυτή η βιβλιοθήκη από τον προγραμματιστή. Υπάρχουν δύο τρόποι να γίνει αυτό. Ο πρώτος είναι να χρησιμοποιηθεί η εντολή `#include` πριν την εκτέλεση του κώδικα στην αρχή του sketch διευκρινίζοντας το όνομα της επιθυμητής βιβλιοθήκης που θα εισαχθεί στο πρόγραμμα. Η σύνταξη για την εισαγωγή μίας βιβλιοθήκης είναι:

```
#include<"name library".h>
```


Το «name library» δηλώνει το όνομα του αρχείου της βιβλιοθήκης που επιθυμεί ο προγραμματιστής να χρησιμοποιήσει. Ο δεύτερος τρόπος για την εισαγωγή μίας βιβλιοθήκης είναι από την εργαλειοθήκη του arduino IDE. Κάνοντας κλικ την επιλογή sketch από το μενού και επιλέγοντας το Import Library. Ο προγραμματιστής έχει την δυνατότητα να επιλέξει μία από της βιβλιοθήκες που περιλαμβάνει το arduino IDE. Στη συνέχεια το περιβάλλον είναι αυτό που αναλαμβάνει από μόνο του να εισάγει την βιβλιοθήκη στο sketch.

1.3.18 Comments

Τα comments (σχόλια) είναι περιοχές κειμένων που αγνοούνται από το πρόγραμμα αλλά περιέχουν χρήσιμες σημειώσεις που δίνουν κάποιες πληροφορίες για την λειτουργία του sketch με σκοπό να κάνει τον κώδικα πιο κατανοητό από τους ανθρώπους. Τα comments δεν καταλαμβάνουν χώρο μέσα στην μνήμη του μικροελεγκτή, έτσι μπορούν να χρησιμοποιηθούν άφοβα. Υπάρχουν δύο τρόποι σύνταξης comments στον κώδικα του arduino. Ο πρώτος είναι:

```
/*
```

```
«comments»
```

```
*/
```

Ότι βρίσκεται ανάμεσα στους ειδικούς χαρακτήρες /* και */ αγνοείται. Τα σχόλια μπορούν να αποτελούνται από όσες γραμμές επιθυμεί ο προγραμματιστής. Ο δεύτερος τρόπος χρησιμοποιεί δύο slash «//». Όταν το πρόγραμμα συναντήσει δύο slash καταλαβαίνει ότι το κείμενο που ακολουθεί στα δεξιά είναι comments και τα αγνοεί. Για παράδειγμα:

```
delay(1000); // παύση προγράμματος για ένα sec
```

Το πρόγραμμα θα εκτελέσει την συνάρτηση delay() κανονικά και δεν θα ασχοληθεί με το τι υπάρχει δεξιά των slash.

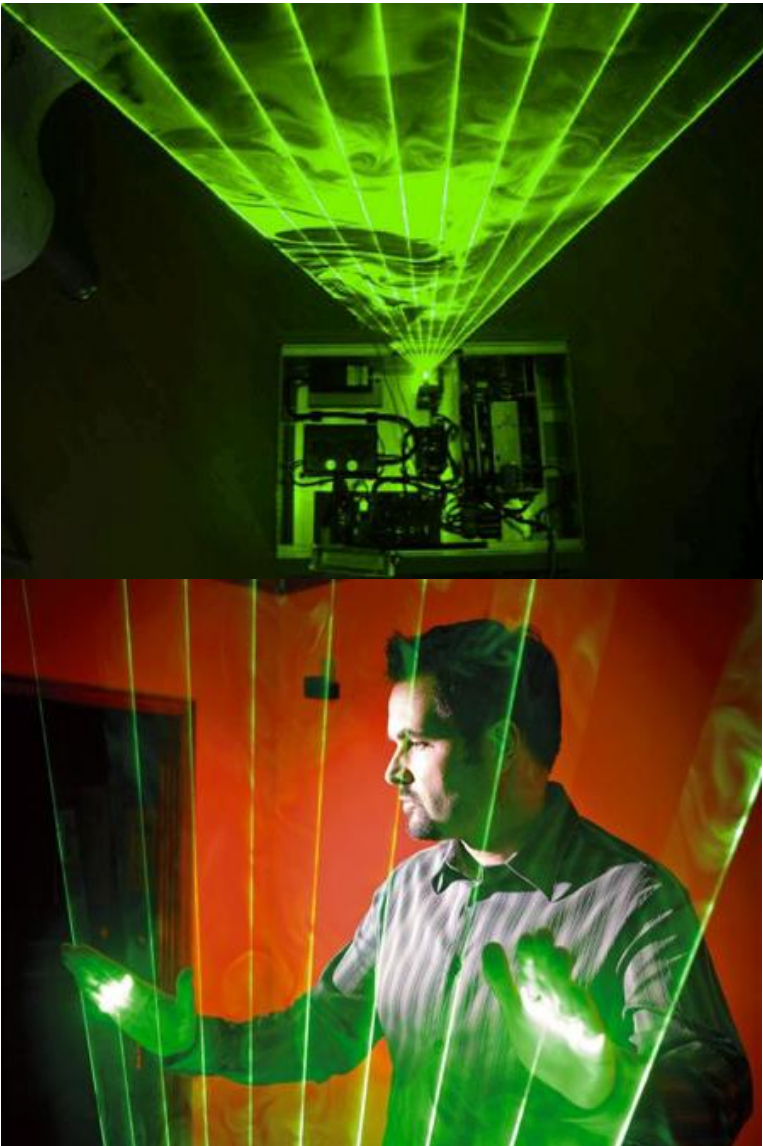
1.4 Εφαρμογές arduino

Όπως αναφέρθηκε και στα προηγούμενα κεφάλαια το arduino είναι ένας ελεγκτής που έχει την ικανότητα να επικοινωνήσει μέσω σειριακής θύρας με άλλες συσκευές. Γεγονός που του δίνει τη δυνατότητα να αλληλεπιδράσει σε συνεργασία διαφόρων προγραμμάτων του υπολογιστή. Οι γλώσσες προγραμματισμού στα οποία έχουν αναπτυχθεί τα προγράμματα ποικίλουν σύμφωνα με τις απαιτήσεις του προγραμματιστή. Ο πιο διαδεδομένος τρόπος για την ανάπτυξη προγραμμάτων που έχουν την ικανότητα να αλληλεπιδρούν με το arduino είναι η γλώσσα προγραμματισμού Processing (περισσότερες πληροφορίες για την processing θα αναπτυχθούν στην επόμενη ενότητα). Μερικές εφαρμογές που αναπτύχθηκαν με τον ελεγκτή arduino αλλά με τη βοήθεια άλλων προγραμμάτων είναι.

1.4.1 Εφαρμογές του arduino στην μουσική

1.4.1.1 Laser harp

Ίσως το πιο γνωστό project με την χρήση του arduino είναι το laser harp (laser άρπα). Πιθανόν να παρουσιάστηκε για πρώτη φορά από τον Bernard Szajner το 1981, αλλά έγινε δημοφιλές από τον Jean Michel Jarre όπου και το χρησιμοποιούσε στις συναυλίες του. Το 2009 ο Stephen Hobley επηρεασμένος από μία συναυλία του Jarre αποφάσισε να κατασκευάσει τη δική του laser άρπα. Το laser harp είναι ένα μουσικό όργανο που αλληλεπιδράει με το φως. Στην πραγματικότητα αποτελείται από μία συσκευή η οποία παράγει μία σειρά κάθετων φωτεινών γραμμών (laser) που ξεκινούν από το δάπεδο. Ο μουσικός διακόπτοντας τις δέσμες φωτός παράγει μία ποικιλία μουσικών ήχων. Ρόλο για την παραγωγή του ήχου δε παίζει μόνο η διακοπή της δέσμης αλλά και το ύψος που θα τοποθετηθεί το εμπόδιο από το δάπεδο. Η άρπα δεν παράγει από μόνη της τους ήχους, απαιτείται να συνδεθεί με ένα νέας τεχνολογίας synthesizer ώστε να λαμβάνει από το arduino σειριακά τα MIDI (Musical Instrument Digital Interface) δεδομένα που παράγει.



Εικόνα 2.4.1.1 - Επάνω: η συσκευή laser harp. Κάτω: Παίξιμο του μουσικού οργάνου laser harp

1.4.1.2 Soundmachine

Το Σεπτέμβριο στην έκθεση μηχανοκίνητων IAA 2011 της Φρανκφούρτης, η σχεδιαστική ομάδα The Product σε ένα ειδικό τμήμα που αφορούσε νέες τεχνολογίες οι οποίες αλληλεπιδρούν με τον άνθρωπο παρουσίασε την soundmachine. Η εξωτερική εμφάνιση της κατασκευής όπως φαίνεται και στην εικόνα 2.4.1.2 μοιάζει αρκετά με ένα κλασικό πικάπ, η λειτουργία της όμως είναι εντελώς διαφορετική. Η βελόνα του πικάπ έχει αντικατασταθεί με ένα led το οποίο φωτίζει την περιστρεφόμενη πλατφόρμα και έναν αισθητήρα φωτός που ανιχνεύει το αντανακλώμενο φως. Η κυκλική περιστρεφόμενη πλατφόρμα η οποία στην πραγματικότητα έχει αντικαταστήσει το βινύλιο αποτελείται από διάφορες χρωματιστές ακτίνες. Οι ακτίνες με τη σειρά τους ανάλογα από το χρώμα που έχουν, αντανακλούν το φως στον αισθητήρα ο οποίος μετατρέπει το φως σε ηλεκτρικά σήματα και τα στέλνει στο arduino. Ο μικροελεγκτής είναι υπεύθυνος για την μετατροπή των σημάτων στην κατάλληλη μορφή και στη συνέχεια για την αποστολή τους στον υπολογιστή. Από το σημείο αυτό και μετά την εργασία αναλαμβάνει μία ειδική εφαρμογή που έχει αναπτυχθεί σε processing όπου ανάλογα το φως που αντανακλάται από την πλατφόρμα, ο υπολογιστής παίζει τα ανάλογα μουσικά loops.



Εικόνα 2.4.1.2 - Επάνω: το εσωτερικό τμήμα της soundmachine. Κάτω: μουσικός που παίζει με την soundmachine

1.4.2 Εφαρμογές του arduino στον μοντελισμό

Μία άλλη διαδεδομένη εφαρμογή του arduino κυρίως ανάμεσα στους φίλους του μοντελισμού είναι η κατασκευή τηλεκατευθυνόμενου οχήματος. Για τη κατασκευή ενός τέτοιου οχήματος απαιτείται ένας μικροελεγκτής που θα είναι υπεύθυνος για τον έλεγχο του οχήματος (arduino), επίσης απαραίτητο είναι και ένα σύστημα για την ασύρματη επικοινωνία του arduino με την συσκευή που θα το τηλεκατευθύνει (όπως για παράδειγμα η shield x-bee). Ο δουλειά του μικροελεγκτή είναι να ελέγχει τους απαραίτητους κινητήρες και σερβοκινητήρες οι οποίοι τροφοδοτούνται από μία πηγή ρεύματος (συνήθως μπαταρίες) για την κατεύθυνση του σκάφους. Η τηλεκατεύθυνση μπορεί να προσαρμοστεί σε οποιαδήποτε μορφή σκάφους (αυτοκίνητο, πλοίο, αεροπλάνο). Εκτός όμως του βασικού εξοπλισμού, οι μοντελιστές έχουν την δυνατότητα να προσθέσουν περαιτέρω συσκευές ανάλογες των απαιτήσεων τους. Όπως για παράδειγμα είναι η shield GPS που ενημερώνει το arduino με τις συντεταγμένες του σημείου που βρίσκεται. Με τον τρόπο αυτό το όχημα γνωρίζοντας τον προορισμό του βάση των συντεταγμένων που του δόθηκαν, θα μπορεί να κινηθεί προς αυτόν χωρίς να χρειάζεται κάποιος να το κατευθύνει.

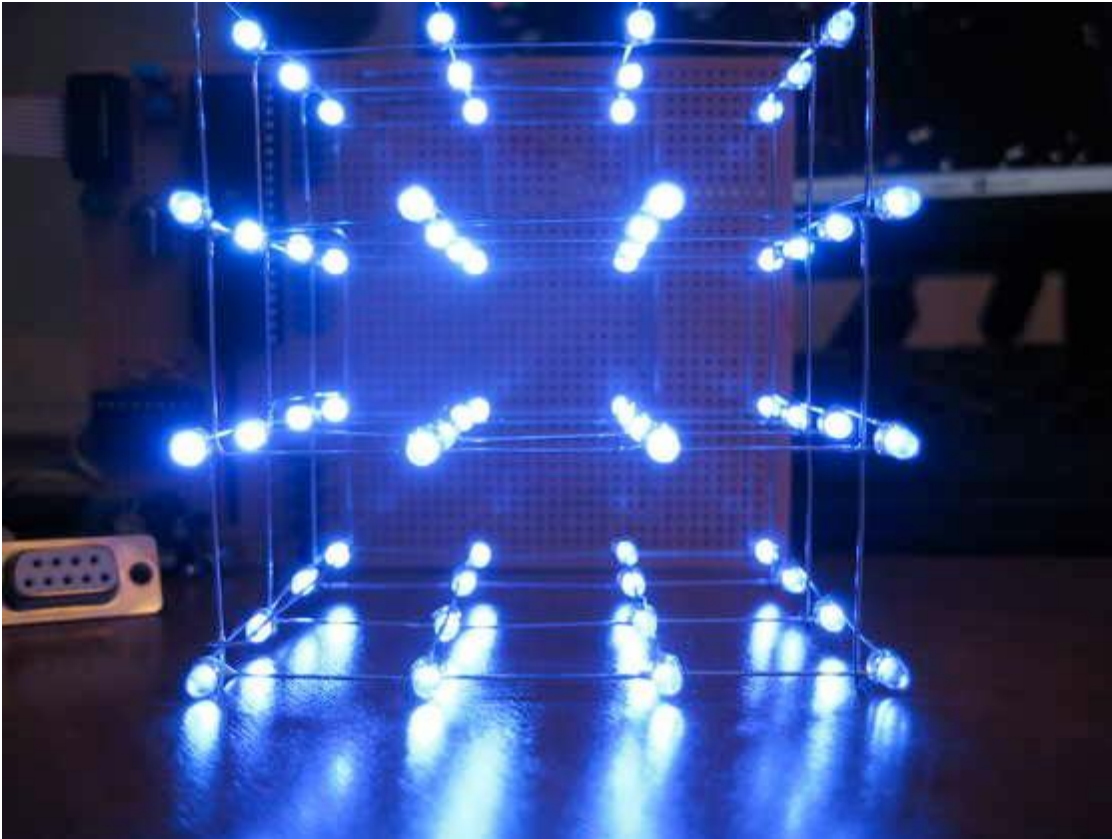


Εικόνα 2.4.2 - UAV κατασκευή χρησιμοποιώντας μικροελεγκτή arduino

1.4.3 Εφαρμογές του arduino για την διακόσμηση

1.4.3.1 Led cube

Μία ενδιαφέρον και σχετικά απλή εφαρμογή είναι η κατασκευή κύβου με led. Το led cube σχηματίζεται από led τα οποία είναι διατεταγμένα σε θέσεις ίσης απόστασης μεταξύ τους. Όπως φαίνεται στην εικόνα 2.4.3.1 ο κύβος είναι 4x4x4, δηλαδή αποτελείται από τέσσερις στρώσεις, και η κάθε μία με την σειρά της αποτελείται από τέσσερις γραμμές και τέσσερις στήλες. Τα led αναβοσβήνουν σύμφωνα με το arduino είτε τυχαία είτε με τέτοιο τρόπο ώστε να σχηματίσουν ένα σχήμα. Με την χρήση περισσότερων led είναι δυνατή η αναπαράσταση τρισδιάστατων γραμμάτων ή ακόμη και εικόνων. Τέτοιου είδους εφαρμογές συνήθως χρησιμοποιούνται για διακοσμητικούς ή διαφημιστικούς σκοπούς.



Εικόνα 2.4.3.1 - Διακοσμητικό led cube στο σκοτάδι

1.4.4 Εφαρμογές του arduino στο σπίτι

Το arduino επίσης μπορεί να χρησιμοποιηθεί για να διευκολύνει τον τρόπο διαβιώσεις των ανθρώπων μέσα στο σπίτι. Αυτό επιτυγχάνεται με διάφορες κατασκευές.

1.4.4.1 Arduino security alarm

Ο ελεγκτής έχει την δυνατότητα να συνδυαστεί με μία ποικιλία αισθητηρίων με σκοπό να λειτουργεί ως ένας συναγερμός σπιτιού. Έτσι ανιχνεύοντας για παράδειγμα την κίνηση, ελέγχοντας συγκεκριμένες πόρτες και παράθυρα ακόμη και την θερμοκρασία σε περίπτωση πυρκαγιάς μπορεί να γίνει ένα αξιόπιστο σύστημα συναγερμού, ενώ με ένα κατάλληλο πρόγραμμα στον υπολογιστή έχει την δυνατότητα ακόμη και να ενημερώνει απομακρυσμένα τους υπεύθυνους του κτιρίου.

1.4.5 Εφαρμογές arduino σε παιχνίδια

Όπως είναι φυσικό οι καινοτομίες του arduino δεν πέρασαν απαρατήρητες και από τους άνθρωπος που ασχολούνται με την κατασκευή πρωτότυπων παιχνιδιών. Η επικοινωνία του ελεγκτή με τον υπολογιστή έδωσε την δυνατότητα για την εύκολη κατασκευή παιχνιδιών που αλληλεπιδρούν με τον άνθρωπο.

1.4.5.1 Marble labyrinth ελεγχόμενο με χρήση the WiiFit

Το marble labyrinth είναι ένα είδος λαβύρινθου που περιέχει μία μεταλλική μπίλια. Σκοπός του παιχνιδιού είναι η μπίλια να διασχίσει μία συγκεκριμένη διαδρομή του λαβύρινθου για να φτάσει στον τερματισμό. Μέσα στους διαδρόμους υπάρχουν διάφορες τρύπες που θα πρέπει να τις προσπεράσει χωρίς να πέσει μέσα ώστε να μην χρειαστεί να ξαναρχίσει την διαδρομή από την αρχή. Για να μετακινηθεί η μπίλια, πρέπει ο παίχτης να αλλάζει συνεχώς το ύψος των πλευρών του λαβύρινθου από το δάπεδο, έτσι λόγω της βαρύτητας η μπίλια θα αναγκαστεί να κινηθεί προς την κατευθύνει της πιο χαμηλής πλευράς του λαβύρινθου.



Εικόνα 2.4.5.1 - Ο λαβύρινθος marble μαζί με το wiifit της NITENDO

Αντί ο χρήστης να χρειάζεται να σηκώσει ολόκληρη την πλατφόρμα του λαβύρινθου για να ελέγξει την κίνηση της μπίλιας, σταθεροποίησαν τον λαβύρινθο πάνω σε μία άλλη πλατφόρμα η οποία με την βοήθεια δύο σερβοκινητήρων μπορεί να αλλάξει να ύψος των πλευρών του λαβύρινθου και να αναγκάσει την μπίλια να κινηθεί. Ο παίχτης για να κατευθύνει την μπίλια πατάει πάνω σε ένα wiifit, μία επιδαπέδια πλατφόρμα της παιχνιδομηχανής wii της NITENDO. Η πλατφόρμα wiifit λειτουργεί

όπως ένα χειριστήριο, δηλαδή ανάλογα με το σημείο που ο παίχτης θα ρίξει το βάρους του, αυτή θα ερμηνεύσει την κίνηση σε μία κατεύθυνση (μπροστά, δεξιά, πίσω, αριστερά) . Το wiifit με την σειρά του στέλνει σειριακά τα δεδομένα στο arduino το οποίο ελέγχει τους σερβοκινητήρες και έτσι αλλάζει στάση ο λαβύρινθος. Η ιδέα για την κατασκευή του παιχνιδιού ανήκε στον Justin Stoffel. Το wiifit μπορεί να αντικατασταθεί με διαφορετικά τύπου χειριστήρια χωρίς ιδιαίτερη δυσκολία.

1.4.6 Εφαρμογές του arduino στις τέχνες (ζωγραφική)

1.4.6.1 Senseless drawing bot

Το senseless drawing bot είναι μία κατασκευή η οποία ζωγραφίζει τυχαίες καμπύλες μστους τοίχους. Όπως φαίνεται και στην εικόνα 1.4.6.1 η συσκευή αυτή είναι ένα μηχανοκίνητο όχημα το οποίο κινείται παράλληλα σε έναν τοίχο. Επάνω στο όχημα έχει τοποθετηθεί ένα διπλό εκκρεμές και στην άκρη του ένα σπρέι χρώματος. Κατά την κίνηση του οχήματος το εκκρεμές αρχίζει να ταλαντώνεται δεξιά και αριστερά ενώ παράλληλα ελέγχεται από έναν αισθητήρα για το εάν ξεπερνάει ένα κατώτατο και ένα ανώτατο όριο. Έτσι για παράδειγμα στην περίπτωση που η ταλάντωση του εκκρεμές είναι μικρότερη από την επιθυμητή τότε το όχημα αρχίζει και κινείται ρυθμικά αριστερά και δεξιά ώστε να αυξήσει την ταλάντωση του εκκρεμές. Αντίθετα όταν η ταλάντωση είναι μεγαλύτερη από την επιτρεπτή τότε το όχημα σταματάει. Όταν το εκκρεμές έχει την σωστή ταλάντωση, τότε με ένα σύστημα αυτοματισμού το σπρέι αρχίζει να ζωγραφίζει χρωματιστές καμπύλες σε όλο το μήκος του τοίχου. Αντικαθιστώντας το σπρέι με διάφορα χρώματα μπορεί να δημιουργηθεί ένα πρωτότυπο έργο τέχνης. Η κατασκευή αυτή προσομοιώνει την δημιουργία ενός γκραφίτι σε τοίχο απελευθερώνοντας τον δυναμισμό του σχεδιασμού του γκραφίτι καταργώντας παράλληλα τον ανθρώπινο παράγοντα.





Εικόνα 2.4.6.1 - Το senseless drawing bot ζωγραφίζει στο τοίχο

1.4.6.2 LumiBots

Έναν αξιοθαύμαστο και διαφορετικό τρόπο σχεδίασης σκέφτηκε και κατασκεύασε ο δημιουργός του lumiBots. Ο Mey Lean Kronemann πάνω σε μία φωσφορίζουσα επιφάνεια έβαλε να κινούνται 9 αυτόνομα robot τα οποία στο κάτω μέρος τους εκπέμπουν προς την φωσφορίζουσα επιφάνεια υπεριώδες φως. Έτσι καθώς τα ρομπότ κινούνται στο σκοτάδι, αφήνουν πίσω τους να φωσφορίζουν τα σημεία απ' όπου πέρασαν. Το κάθε ένα lumibot (ρομπότ) σχεδιάστηκε με σκοπό τα υλικά κατασκευής του να έχουν όσον το δυνατόν φθηνότερο κοστολόγιο. Αποτελείται από ένα ελεγκτή arduino, δύο αισθητήρες φωτός, δύο διακόπτες οι οποίοι ανιχνεύουν την επαφή του ρομπότ με τα τοιχώματα ή με ένα άλλο ρομπότ και τέλος ένα led που εκπέμπει UV φως και κάνει την επιφάνεια στην οποία κινείται να φωσφορίζει. Οι κινήσεις τους δεν είναι ούτε προγραμματισμένες ούτε προβλέψιμες, απλά βασίζονται σε δύο απλούς κανόνες. Ο πρώτος είναι να ακολουθούν μία φωσφορίζουσα γραμμή που υπάρχει στην επιφάνεια πάνω στην οποία κινούνται (όσο πι φωτεινή η γραμμή τόσο καλύτερα). Ο δεύτερος κανόνας αναγκάζει το όχημα να αλλάξει πορεία όταν αυτό έρχεται σε επαφή με ένα άλλο όχημα ή με τα τοιχώματα που υπάρχουν με σκοπό να περιορίσει την κίνηση του lumibot μέσα στην φωσφορίζουσα επιφάνεια. Όταν το lumibot δεν βρίσκει να ακολουθήσει φωσφορίζουσες γραμμές τότε αυτό από μόνο του κινείται κυκλικά σύμφωνα με μία ακτίνα που του έχει δοθεί. Όπως φαίνεται και στην παρακάτω εικόνα τα σχήματα που δημιουργούνται είναι εκπληκτικά.



Εικόνα 2.4.6.2 - Επάνω το lumiBot. Κάτω το αποτέλεσμα που φαίνεται πάνω στην φωσφορίζων επιφάνεια.

1.4.7 Εφαρμογές του arduino στην ρομποτική

Η μεγάλη ποικιλία των συμβατών αισθητήρων για arduino που διατίθενται στο εμπόριο κάνουν τον σχεδιασμό ρομποτικών συστημάτων πολύ ενδιαφέρον. Ενώ παράλληλα η ασύρματη επικοινωνία του arduino με έναν υπολογιστή μπορεί να αυξήσει τις δυνατότητες των ρομπότ σε πολύ μεγάλο βαθμό.

1.4.8 Mind-control

Μία από τις πιο εντυπωσιακές ρομποτικές κατασκευές είναι και αυτή η οποία παρουσιάζεται στο βιβλίο «Make a Mind-Controlled Arduino Robot» της oreilly. Στο βιβλίο αυτό περιγράφεται πως κατασκευάζεται ένα ρομπότ το οποίο θα ανταποκρίνεται σύμφωνα με την επιθυμία του ανθρώπου χωρίς να χρειαστεί να πατήσει κάποιο πλήκτρο. Πιο συγκεκριμένα εξηγείται ο τρόπος για την κατασκευή ενός ρομπότ το οποίο θα παίζει ήχους, θα αναβοσβήνει led και θα αντιδρά στα σήματα που θα δέχεται από το mindwave. Το mindwave είναι μία συσκευή η οποία τοποθετείται στο κεφαλή και ανιχνεύει την ηλεκτρική δραστηριότητα του ανθρώπινου εγκεφάλου. Η συσκευή αυτή μπορεί να ερμηνεύσει συγκεκριμένες ενέργειες και να τις μετατρέψει σε εντολές οι οποίες στην συνέχεια μπορούν να μεταβιβαστούν σειριακά στο arduino. Ο τρόπος λειτουργίας του mindwave είναι ακριβώς ο ίδιος με αυτόν του εγκεφαλογραφήματος.



Εικόνα 2.4.8 - Η συσκευή mindwave της neurosky η οποία ανιχνεύει την ηλεκτρική δραστηριότητα του ανθρώπινου εγκεφάλου.

Κεφάλαιο 2 : **ΕΦΑΡΜΟΓΗ - ΣΧΕΔΙΑΣΜΟΣ - ΚΑΤΑΣΚΕΥΗ -** **ΛΕΙΤΟΥΡΓΙΑ**

2.1 Η ιδέα της εφαρμογής - περιγραφή

Αρχικά η βασική ιδέα για τον σχεδιασμό του ηλεκτρολογίου προήλθε από το project [Arduino Monome clone](#)[3]. Σε αυτό το project χρησιμοποιήθηκε ένα Arduino και κάποια RGB leds με ένα ποτενσιόμετρο ώστε να δημιουργηθεί ένα πολύχρωμο ηλεκτρολόγιο. Κόψαμε το ηλεκτρολόγιο ώστε να αποτελείται από δύο σειρές και τέσσερις στήλες. Αυτό είχε σαν αποτέλεσμα να μειωθεί σημαντικά το κόστος για την κατασκευή του ηλεκτρολογίου και κάνει τον κώδικα που πολύ πιο εύκολο.

2.2 Σχεδιασμός μηχανικού μέρους

2.2.1 Υλικά κατασκευής-προδιαγραφές

Πιο αναλυτικά για την κατασκευή του ηλεκτρολογίου χρειαστήκαμε:

- Έναν μικροελεγκτή Arduino Uno
- 1 TIP120 transistor
- 1 δίοδο 1N4001
- 10 διόδους 1N4148
- 4 2n2222 transistors
- 1 ηλεκτρολόγιο Monome style ([Sparkfun Electronics](#) [4])
- 1 Keypad PC board ([Sparkfun Electronics](#) [5])
- 8 RGB LEDs ([Sparkfun Electronics](#) [6])
- 1 ελεγκτή τάσης 7805
- 4 αντιστάσεις 100 ohm
- 2 αντιστάσεις 150 ohm
- 8 αντιστάσεις 1 Kohm

2.3 Η μηχανική κατασκευή

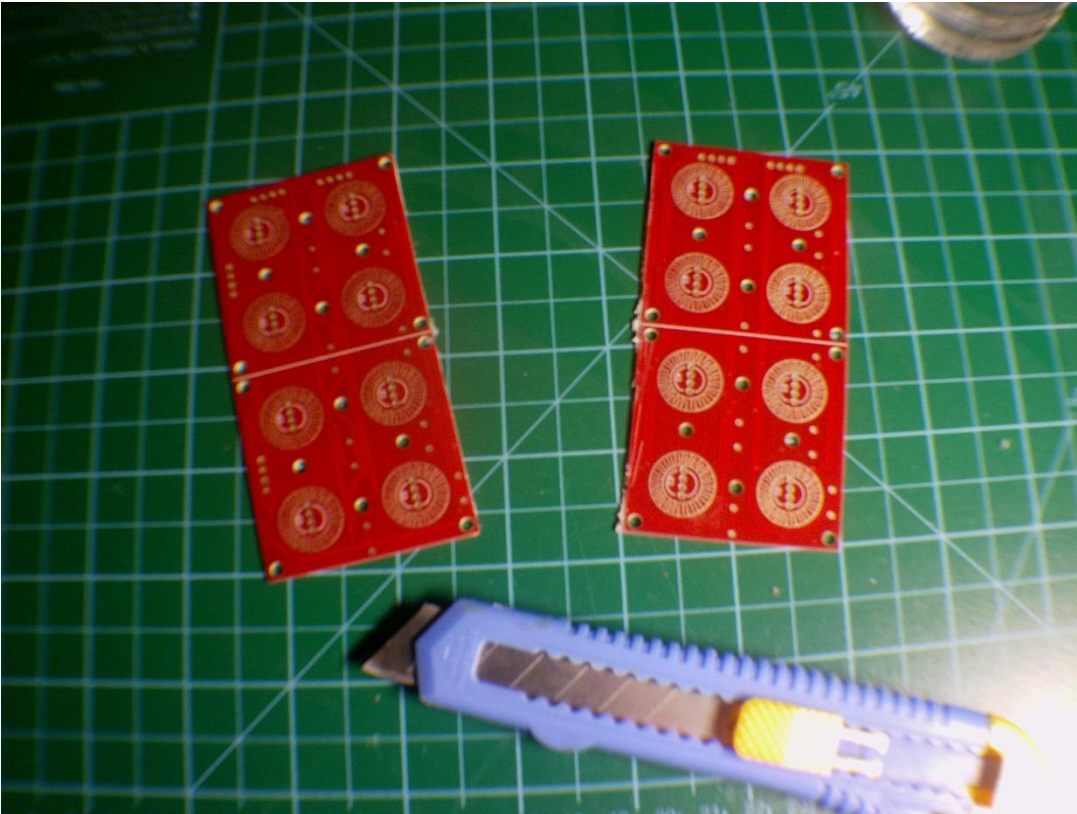
Το ηλεκτρολόγιο στη πραγματικότητα αποτελείται από δυο ξεχωριστά κυκλώματα τα οποία αλληλεπικαλύπτονται . Το κύκλωμα εισόδου είναι μια μήτρα απλού ηλεκτρολογίου. Για να διαβάσει το πάτημα κάθε κουμπιού το Arduino κάνει μία είσοδο του ηλεκτρολογίου λογικό 1 (high) και ελέγχει την τάση από τις τέσσερις γραμμές στη σειρά. Οι δίοδοι τον κύκλωμα , αποκλείουν την ανατροφοδότηση μεταξύ γραμμών και στηλών.

Τα RGB leds ανάβουν μέσω ενός εντελώς ξεχωριστού κυκλώματος . Η φωτεινότητα κάθε γραμμής των χρωματιστών leds , ελέγχεται από ένα ψηφιακό ποτενσιόμετρο. Το ψηφιακό ποτενσιόμετρο , λειτουργεί όπως ένα κανονικό ποτενσιόμετρο αλλά ελέγχεται ψηφιακά από τον Arduino.Εντωμεταξύ, κάθε στήλη από τα Leds ελέγχεται από ένα ξεχωριστό τρανζίστορ. Αλλάζοντας γρήγορα την αντίσταση και καθώς προχωράμε στις στήλες , κάθε led θα φαίνεται σαν να ελέγχεται ατομικά.

Η πόρτα του κυκλώματος είναι αρκετά απλή . Δεδομένου ότι περιέχει ένα πηνίο, θα το αντιμετωπίσουμε σαν το πηνίο ενός βηματικού κινητήρα και θα χρησιμοποιήσουμε ένα TIP120 τρανζίστορ να παρέχει το ρεύμα. Όταν η ισχύς αφαιρείται από ένα πηνίο, η κατάρρευση του μαγνητικού πεδίου δημιουργεί ένα ρεύμα μέσα από το πηνίο. Για να προστατέψουμε τον TIP120 από υπερθέρμανση, θα προσθέσουμε μια δίοδο για να χειριστούμε το κύμα που δημιουργήθηκε από την field breakdown.

Σημείωση: Η δίοδος είναι σε καλύτερη θέση παράλληλα με το πηνίο για να χειριστεί την παροδική αύξηση. Είναι σωστό, αλλά το κύκλωμα εδώ έχει λειτουργήσει τέλεια για αρκετούς μήνες, οπότε λειτουργεί με τον ένα ή τον άλλο τρόπο.

Τα ίχνη για τα button φαινόταν λίγο δύσκολο να ταιριάξουν με τα ίδια τα button, γι 'αυτό παραγγείλαμε αυτήν την πλακέτα που παράγει η εταιρεία Sparkfun για τα ηλεκτρολόγια της. Η Sparkfun παρέχει τη διάταξη για αυτά τα πλήκτρα στην βιβλιοθήκη τους, ώστε να μπορείτε να φτιάξουμε το δικό μας PCB. Για περισσότερη αξιοπιστία, θα προτιμούσαμε πιθανώς να παράγεται στο εμπόριο. Η πλακέτα δεν έχει σχεδιαστεί για να κοπεί σε κομμάτια, αλλά μετά από μια επανεξέταση των ιχνών και των μονοπατιών, αποφασίσαμε ότι θα μπορούσαμε να ξεπεράσουμε αυτό το εμπόδιο με την αφαίρεση ενός ζευγαριού γραμμών από την πλακέτα.



Εικόνα 2.1 - Κοπή πλακέτας

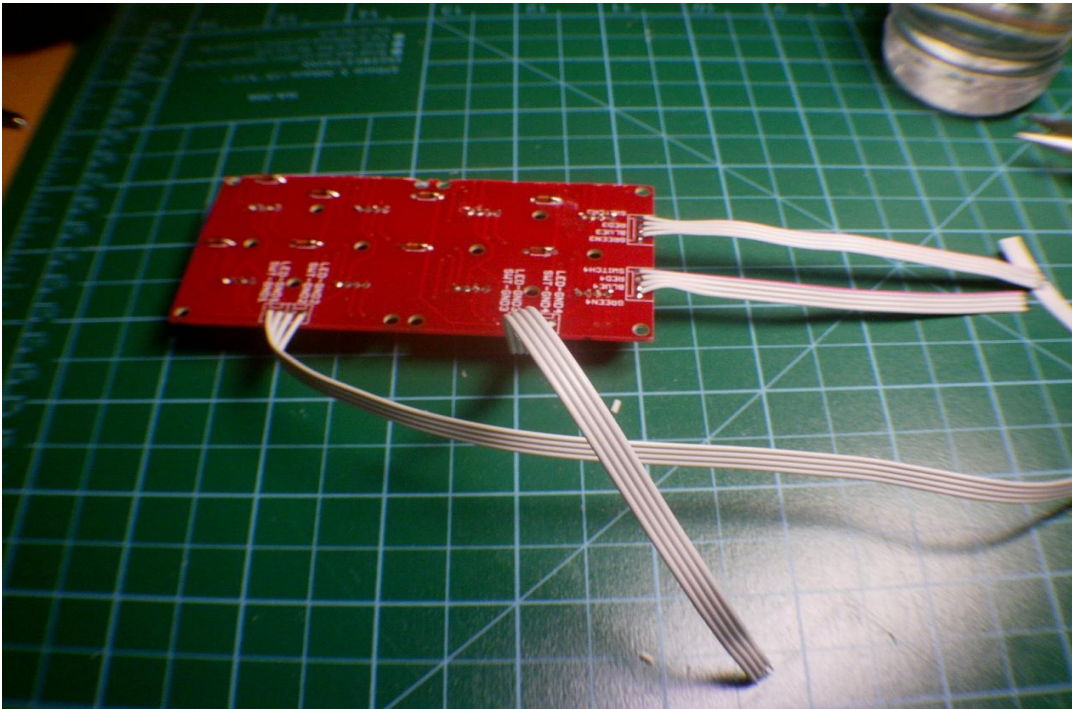
Χωρίζουμε προσεκτικά την πλακέτα κάτω από τη μέση ένα κοπίδι. Αν κοιτάξετε προσεκτικά, μπορείτε να δείτε πού μερικά από τα μονοπάτια έχουν πράγματι μειωθεί κατά το ήμισυ. (η για πιο επαγγελματική δουλειά με μια πριονοκορδέλα) .



Εικόνα 2.2 – Κοπή ηλεκτρολογίου

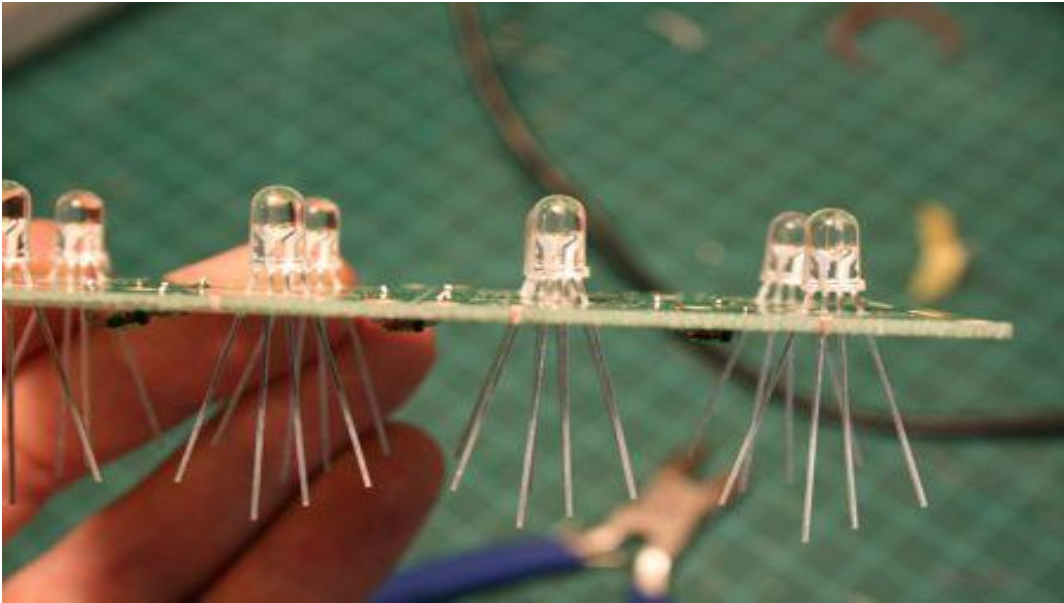
Η κοπή των button είναι πολύ πιο εύκολη. Τα button έχουν προ-χαραγμένες γραμμές που χρειάζονται μόνο ένα γρήγορο και δυνατό κτύπημα από ένα κοφτερό μαχαίρι ή ψαλίδι για να τα χωρίσει.

Το νέο μικρότερο PCB χρειάζεται μόνο μερικά τμήματα: μερικές 1N4148 διόδους και τα RGB LEDs. Η μεταξοτυπία στο ταμπλό δείχνει την κατεύθυνση και τη θέση των διόδων και των LED.



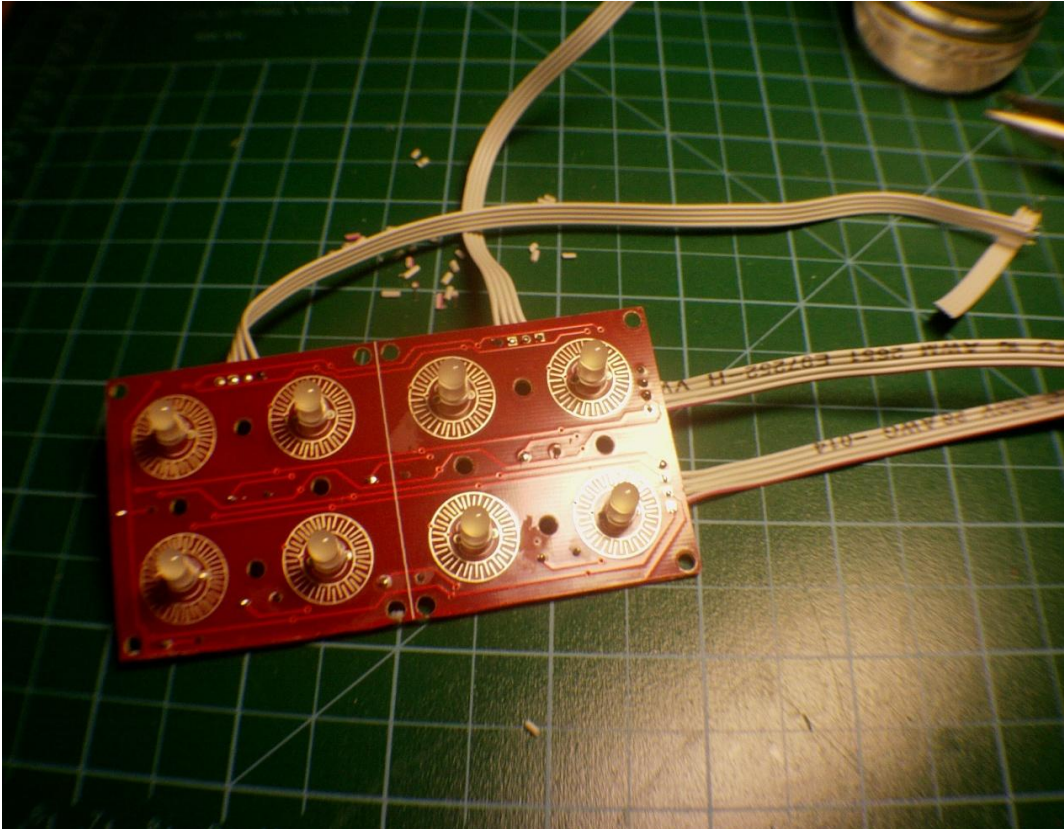
Εικόνα 2.4 – Σύνδεση καλωδίων σε πλακέτα

Μόλις κολλήσουμε τις 1N4148 διόδους, τις κόβουμε όσο πιο κοντά στην πλακέτα μπορούμε. Κόφτες με επίπεδη κεφαλή σαν αυτόν λειτουργούν εξαιρετικά καλά. Το πληκτρολόγιο θα καθίσει σε αυτήν την πλευρά της πλακέτας και θέλουμε να βεβαιωθούμε ότι μπορεί να καθίσει όσο το δυνατόν επίπεδα.



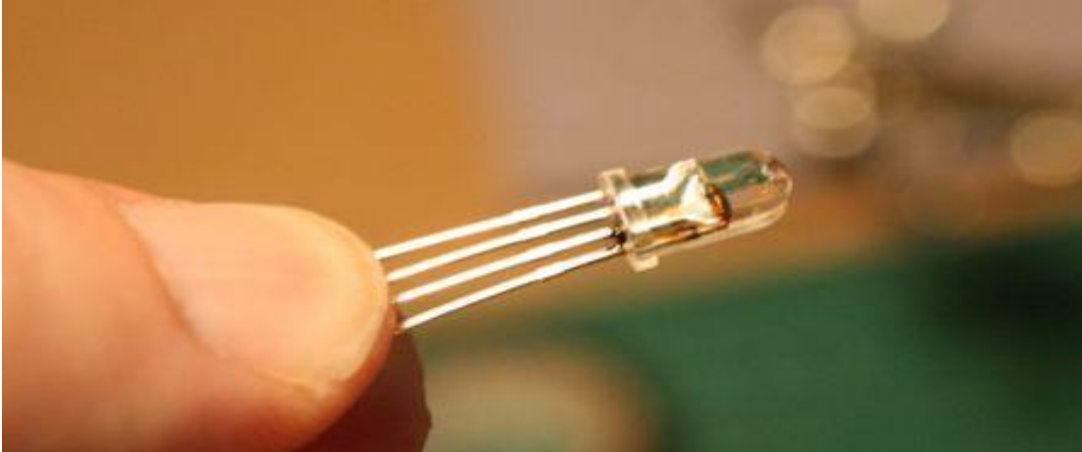
Εικόνα 2.5 – Τοποθέτηση LEDs σε πλακέτα

Τοποθετούμε τα LEDs στον προσανατολισμό που υποδεικνύεται από την μεταξοτυπία στο ταμπλό. Τα σπρώχνουμε προς τα κάτω στην πλακέτα μέχρι να εισαχθούν ακριβώς όπως αυτό. Αν τα αφήσουμε να κολλήσουν πολύ ψηλά, θα παρεμβαίνουν με τα κουμπιά του πληκτρολογίου να πιέζονται.



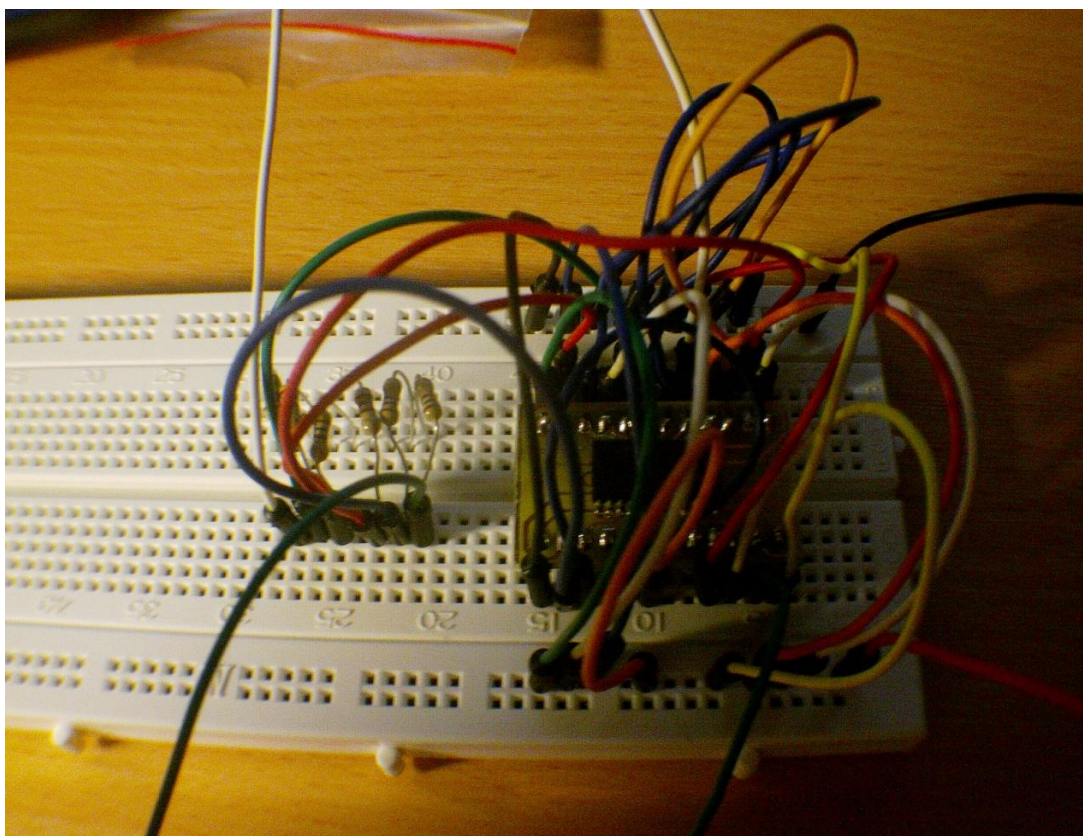
Εικόνα 2.6 –Συγκόλληση LEDs

Μόλις συγκολλήσουμε όλα τα LEDs στη θέση τους, τα καθαρίζουμε ψαλιδίζοντας οτιδήποτε δε μας χρειάζεται. Στη συνέχεια, θα χρειαστεί να προσθέσουμε καλώδιο από το ηλεκτρολόγιο έως την πλακέτα διεπαφής που θα κατασκευάσουμε. Χρησιμοποιήσαμε κάποια παλιά καλωδίωση CAT-5. Δεδομένου ότι κάθε άξονας της πλακέτας έχει οκτώ ακίδες, είναι ιδανικό για την εφαρμογή.



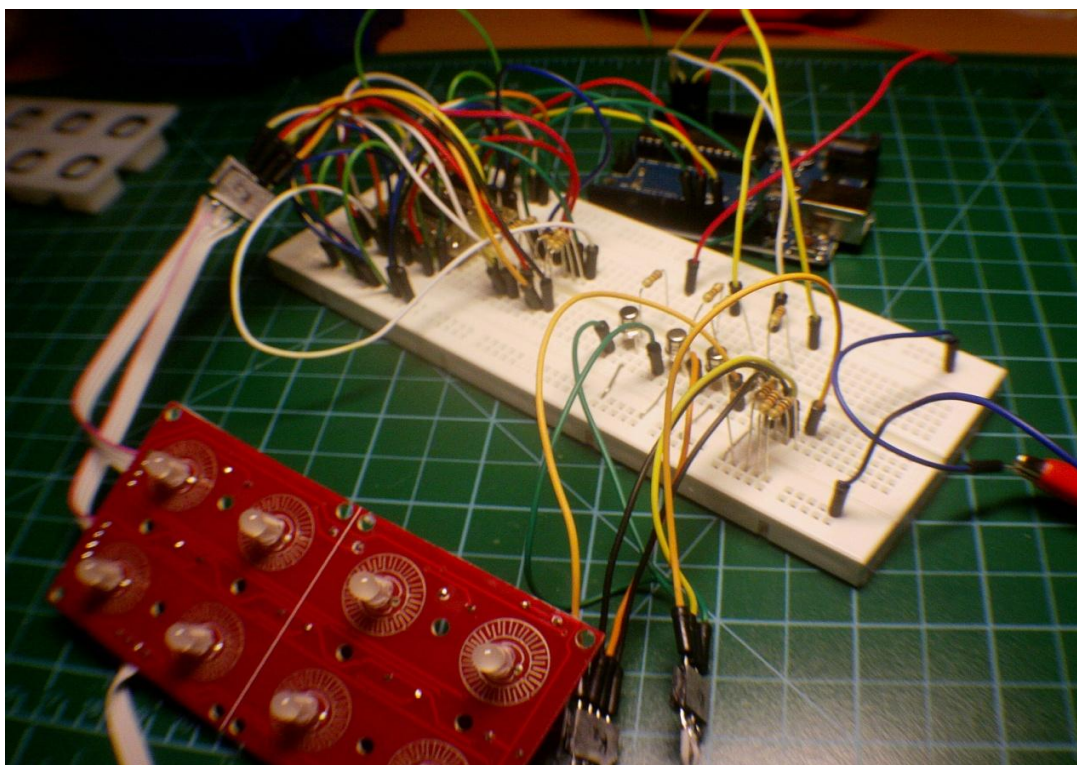
Εικόνα 2.7 – RGB LED

Κάθε RGB LED διαθέτει τρεις λυχνίες LED στο εσωτερικό της συσκευασίας. Μοιράζονται ένα κοινό τερματικό και διαθέτουν ένα μόνο ξεχωριστό αγωγό που βγαίνει έξω. Επειδή έχουν διαφορετικά χαρακτηριστικά - την φωτεινότητα, τις τρέχουσες και τις απαιτήσεις τάσης, περάσαμε αρκετό χρόνο δοκιμάζοντας διάφορους συνδυασμούς.

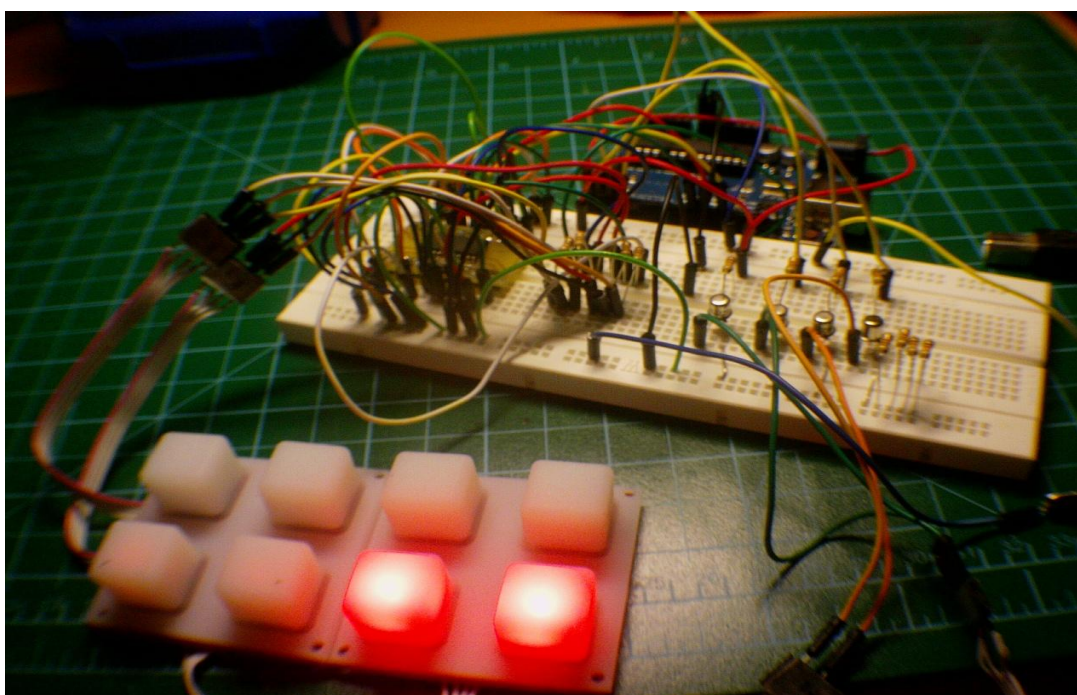


Εικόνα 2.8 –Κύκλωμα φωτεινότητας LED

Μετά από κάποια πειράματα, καταφέραμε να βρούμε το σωστό συνδυασμό για να δημιουργήσουμε κάποια με αρκετά λευκό φως. Οι απαιτήσεις θα ποικίλλουν μεταξύ των κατασκευαστών, αλλά για τα Sparkfun LEDs διαπιστώσαμε ότι ένα ζεύγος αντιστάσεων 100 ohm και μία αντίσταση 150 ohm μπορούν να συνδυάσουν το κόκκινο, το πράσινο και το μπλε αρκετά καλά.

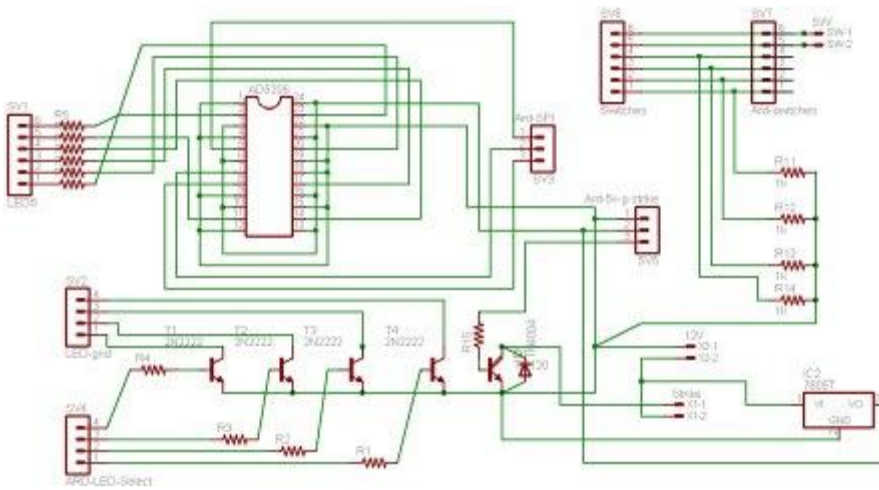


Εικόνα 2.9 –Κύκλωμα φωτεινότητας LED



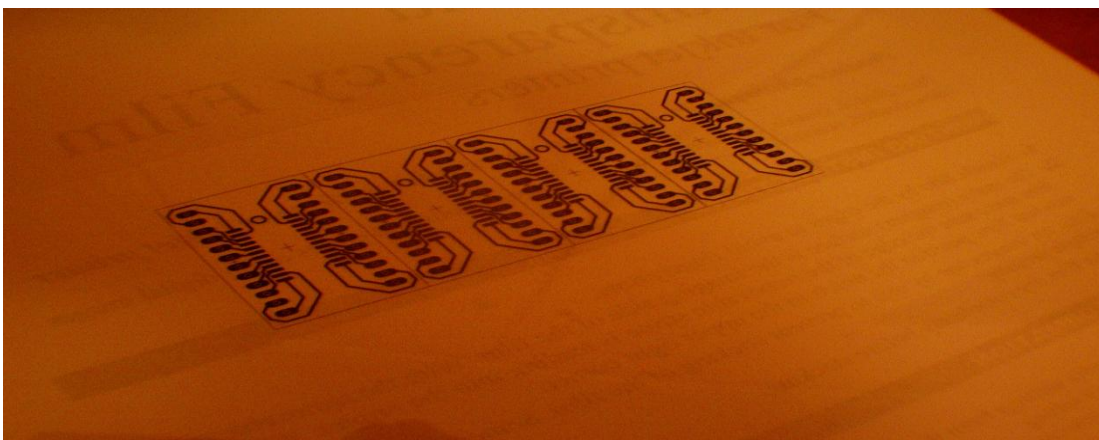
Εικόνα 2.10 –Κύκλωμα φωτεινότητας LED

Ο συνδυασμός χρωμάτων ήταν δύσκολο για τα μάτια μέχρι να βάλουμε το πληκτρολόγιο πάνω από το LED για να ελέγξουμε διπλά τα ευρήματά μας.

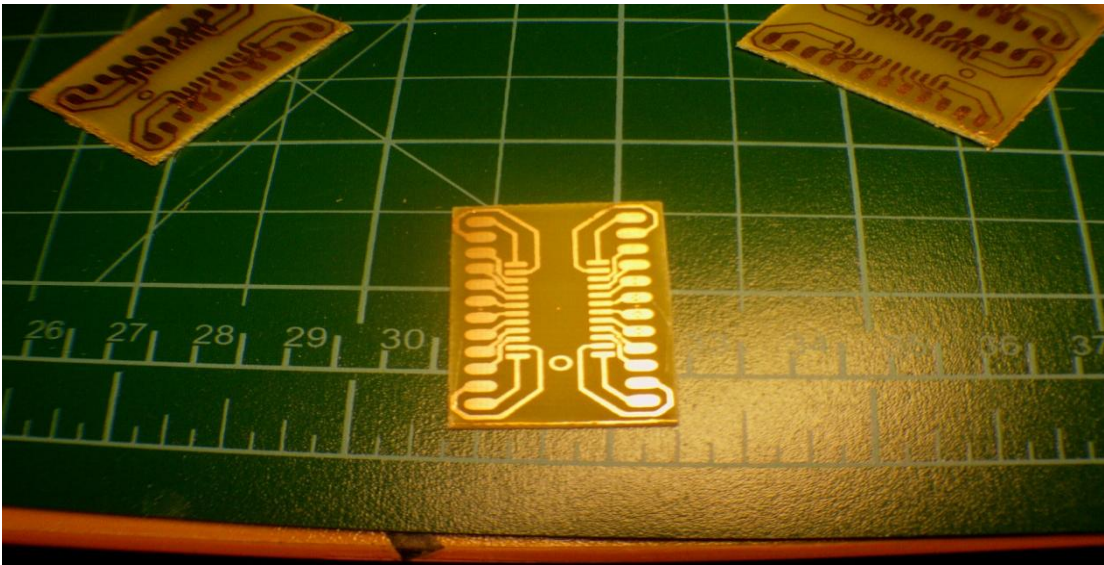


Εικόνα 2.11 – Σχέδιο κυκλώματος

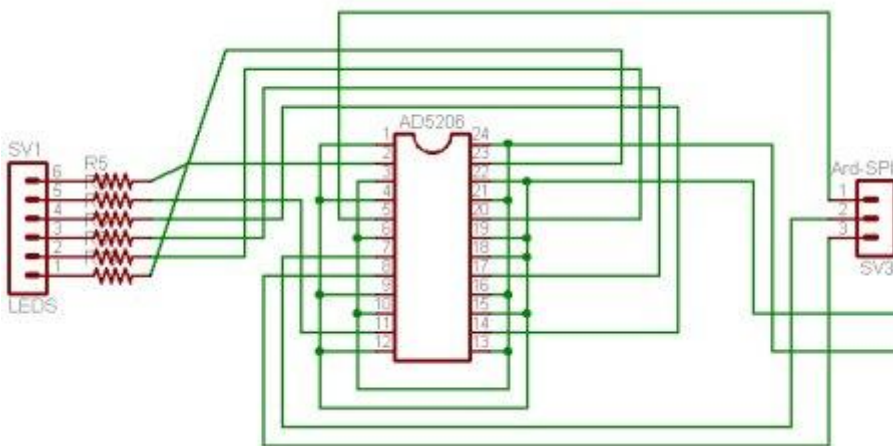
Το κύκλωμα έχει αφθονία στοιχείων, αλλά είναι αρκετά εύκολο να κατασκευαστεί. Θα χωρίσουμε τα πάντα από το τμήμα για να κρατήσουμε τα πράγματα εύκολα. Για τη σχεδίαση του κυκλώματος χρησιμοποιήσαμε το Eagle [7].



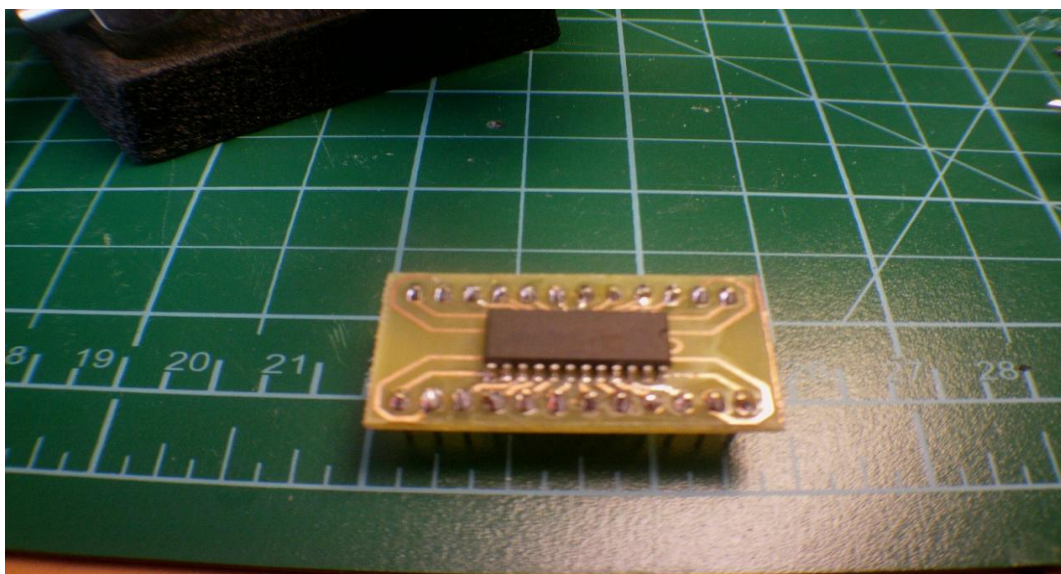
Εικόνα 2.12 – Τύπωση κυκλώματος για AD5206



Εικόνα 2.13 – Πλακέτα AD5206



Εικόνα 2.14 – Ηλεκτρονικό σχέδιο AD5206



Εικόνα 2.15 – Συγκόλληση AD5206 σε πλακέτα

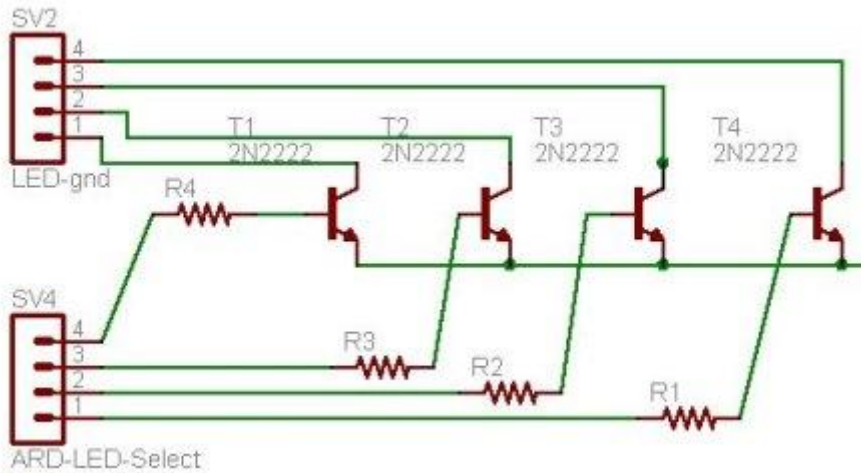
Το ψηφιακό ποτενσιόμετρο έχει έξι εξόδους. Κάθε μία από αυτές θα τροφοδοτήσει μια σειρά από κόκκινο, πράσινο ή μπλε LEDs, μέσω μίας φωτοαντίστασης. Η ψηφιακή καλωδίωση του ποτενσιόμετρου προέρχεται άμεσα από αυτόν τον τρόπο. Για περισσότερες πληροφορίες, μπορούμε να ανατρέξουμε εδώ [8].

- Συνδέστε AD5206 ακίδες 3, 6, 10, 13, 16, 21 και 24 5v.
- Συνδέστε τις ακίδες 1, 4, 9, 12, 15, 18, 19, και 22 στο έδαφος.
- Συνδέστε την ακίδα 5 στην ακίδα 10 του Arduino
- Συνδέστε την ακίδα 7 στην ακίδα 11 του Arduino
- Συνδέστε την ακίδα 8 στην ακίδα 13 του Arduino

Παίρνουμε τέσσερις αντιστάσεις 100 ohm και δύο 150 ohm αντιστάσεις. Τα τοποθετούμε στο breadboard στη σειρά με κάθε άκρη σε ένα ξεχωριστό διάλυο. (Απέναντι από το κέντρο της πλακέτας είναι πιο εύκολο), Συνδέουμε τα έξι LED που οδηγούν από το πληκτρολόγιο στο ένα άκρο της κάθε αντίστασης –τα κόκκινα παίρνουν τις 150 και τα μπλε και τα πράσινο τις 100. Εδώ είναι η σειρά σύνδεσης που χρησιμοποιούμε.

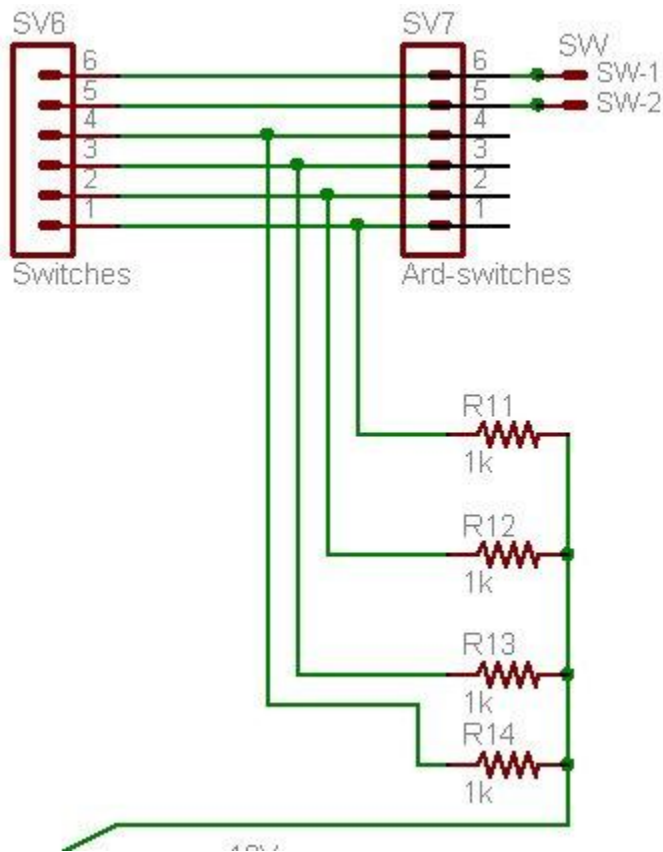
- RED3 σε μία αντίσταση 150 ohm στην ακίδα 14
- Green3 σε μία αντίσταση 100 ohm στην ακίδα 11
- BLUE3 σε μία αντίσταση 100 ohm στην ακίδα 2
- RED4 σε μία αντίσταση 150 ohm στην ακίδα 23

- GREEN4 σε μία αντίσταση 100 ohm στην ακίδα 20
- BLUE4 σε μία αντίσταση 100 ohm στην ακίδα 17



Εικόνα 2.16 – Συνδεσμολογία τρανζίστορ

Για τη γείωση των LED, θα πρέπει να χρησιμοποιήσουμε τέσσερα 2N2222 τρανζίστορ. Το Arduino θα ενεργοποιήσει κάθε τρανζίστορ ξεχωριστά μέσω μίας αντίστασης 1Kohm . Ο συλλέκτης του κάθε τρανζίστορ συνδέεται σε μία γραμμή γείωσης από το ηλεκτρολόγιο. Ο εκπομπός του κάθε τρανζίστορ είναι συνδεδεμένος με το έδαφος. Τα τέσσερα τρανζίστορ επιλέγουν γραμμές για να συνδεθούν με τις ακίδες του Arduino 0, 1, 2, και 3. Έχουν σημειωθεί ως Αναλογικά, αλλά δεν έχει σημασία.

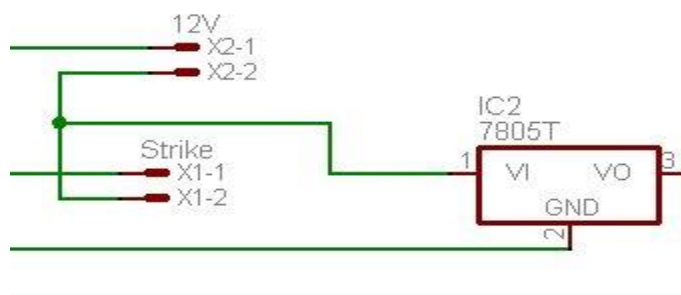


Εικόνα 2.17 – Ηλεκτρονικό σχέδιο πληκτρολογίου –αντιστάσεων

Η μήτρα πληκτρολογίου διακόπτη συνδέεται σε τέσσερις στήλες και δύο σειρές. Κάθε μία από τις τέσσερις στήλες παίρνει μία αντίσταση. Χρησιμοποιήσαμε 1Kohm αντιστάσεις για τις R11, R12, R13, και R14. Το ένα ποδαράκι της αντίστασης, συνδέεται με τις στήλες και το άλλο είναι γειωμένο.

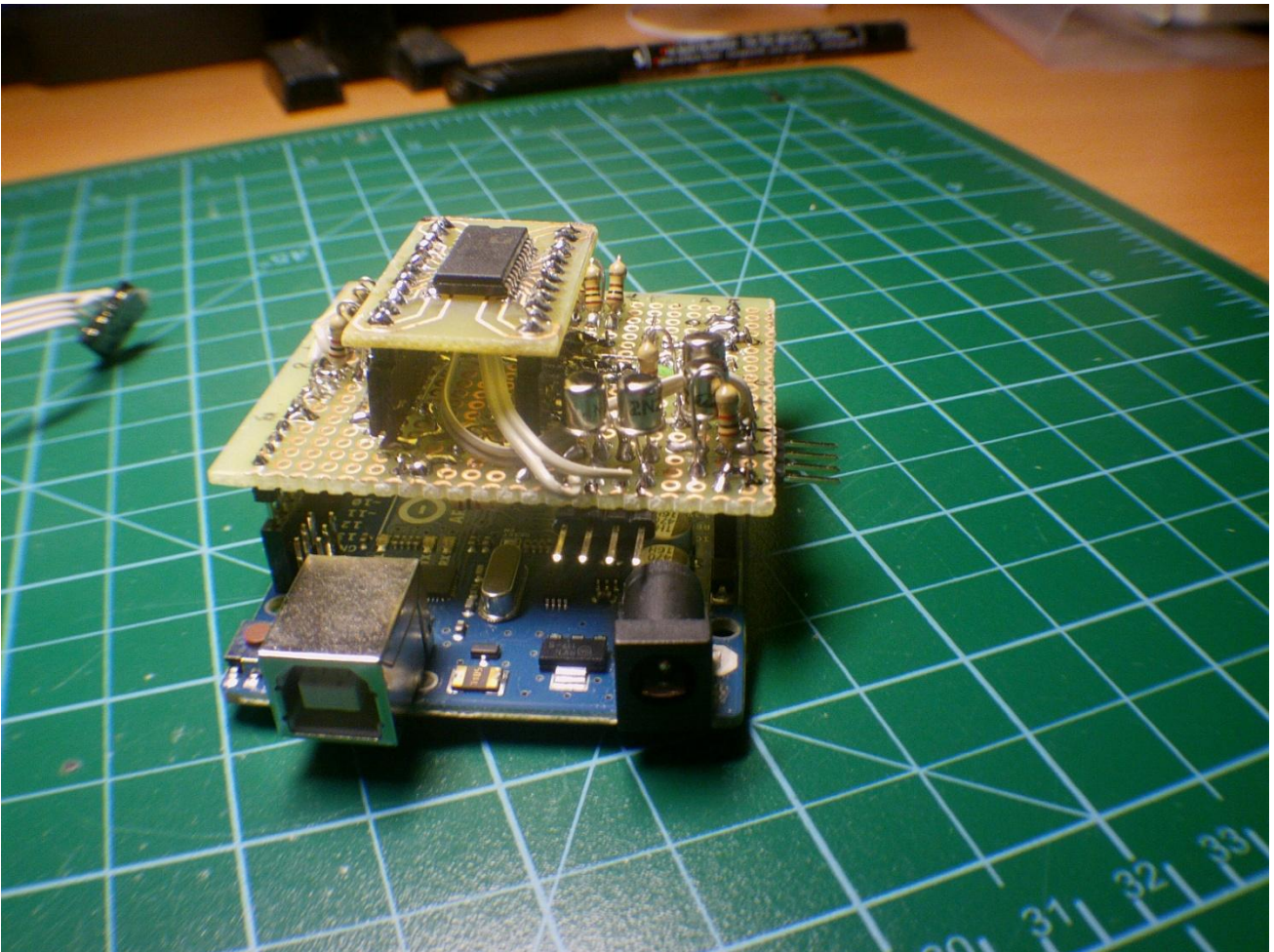
Οι ακίδες 2 και 3 του Arduino θα πρέπει να συνδεθούν με τις δύο μη γειωμένες γραμμές, οι οποίες σημειώνονται SWITCH3 και SWITCH4 στον πίνακα PC (5 και 6 στο schematic).

Οι ακίδες 6, 7, 8, και 9 του Arduino, πρέπει να συνδεθούν με τις τέσσερις γραμμές που αναγράφουν SWT-GND1, SWT-GND2, SWT-GND3 και SWT-GND4 (1-4 για το schematic).



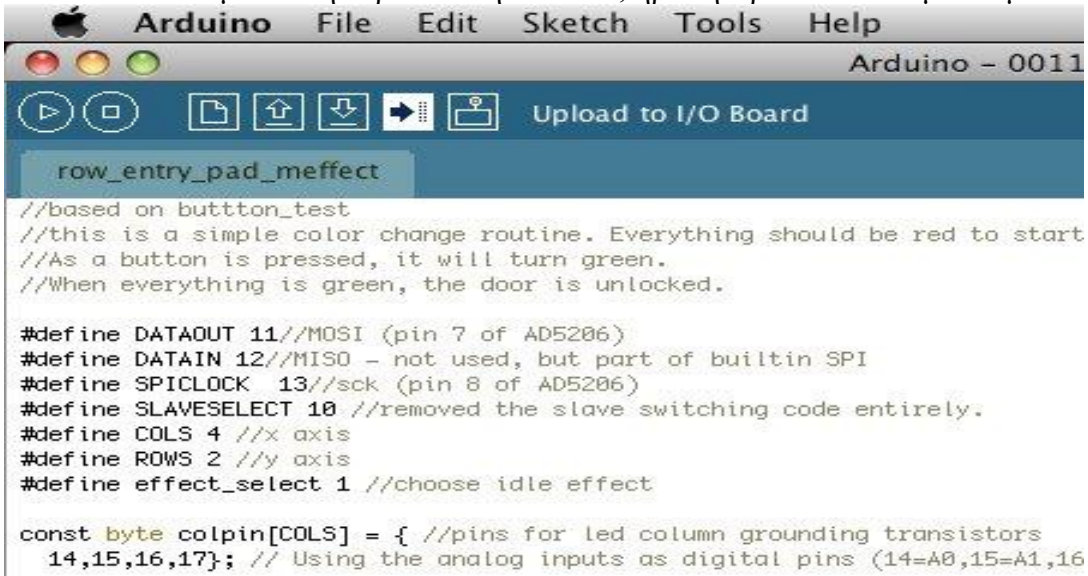
Εικόνα 2.17- Ηλεκτρονικό σχέδιο σύνδεσης κλειδαριάς

Η τελική έκδοση της πλακέτας παίρνει 12VDC είσοδο για να λειτουργεί η κλειδαριά της πόρτας. Προσθέσαμε έναν 7805 για να ρίξει τα 12V στα 5V για το Arduino.. Το Arduino έχει ένα ενσωματωμένο ρυθμιστή, αλλά οι 7805 είναι φθηνοί και βοηθούν στη μείωση του φορτίου στην κατασκευή του Arduino ως ρυθμιστή. Για την ανάπτυξη κώδικα, συνδέσαμε μόνο ένα LED με μία αντίσταση στη γραμμή εξόδου που θα ελέγχει την κλειδαριά της πόρτας.



Εικόνα 2.18 – Τελική συγκόλληση και συνδεσμολογία συστήματος

Με όλα καλωδιωμένα στην πρωτότυπη πλακέτα, ήρθε η ώρα να το δοκιμάσουμε.



```
Arduino File Edit Sketch Tools Help
Arduino - 0011
row_entry_pad_meffect
//based on button_test
//this is a simple color change routine. Everything should be red to start
//As a button is pressed, it will turn green.
//When everything is green, the door is unlocked.

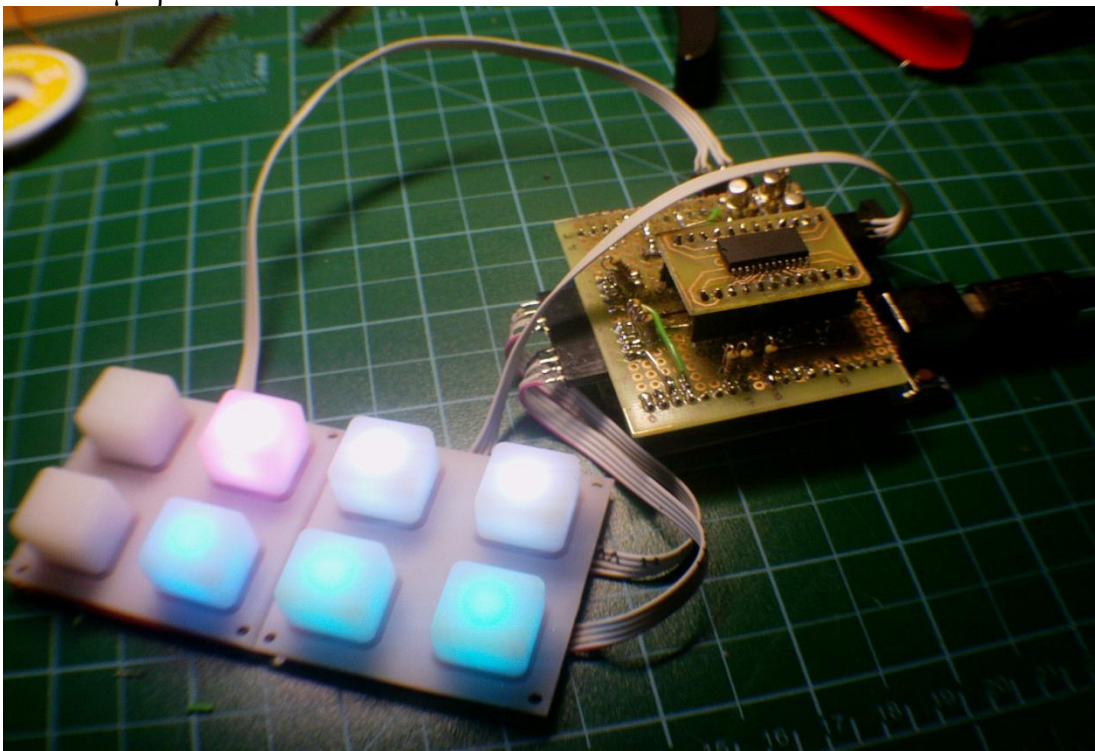
#define DATAOUT 11//MOSI (pin 7 of AD5206)
#define DATAIN 12//MISO - not used, but part of builtin SPI
#define SPICLOCK 13//sck (pin 8 of AD5206)
#define SLAVESELECT 10 //removed the slave switching code entirely.
#define COLS 4 //x axis
#define ROWS 2 //y axis
#define effect_select 1 //choose idle effect

const byte colpin[COLS] = { //pins for led column grounding transistors
  14,15,16,17}; // Using the analog inputs as digital pins (14=A0,15=A1,16
```

Εικόνα 2.19 – Προγραμματιστικό περιβάλλον Arduino

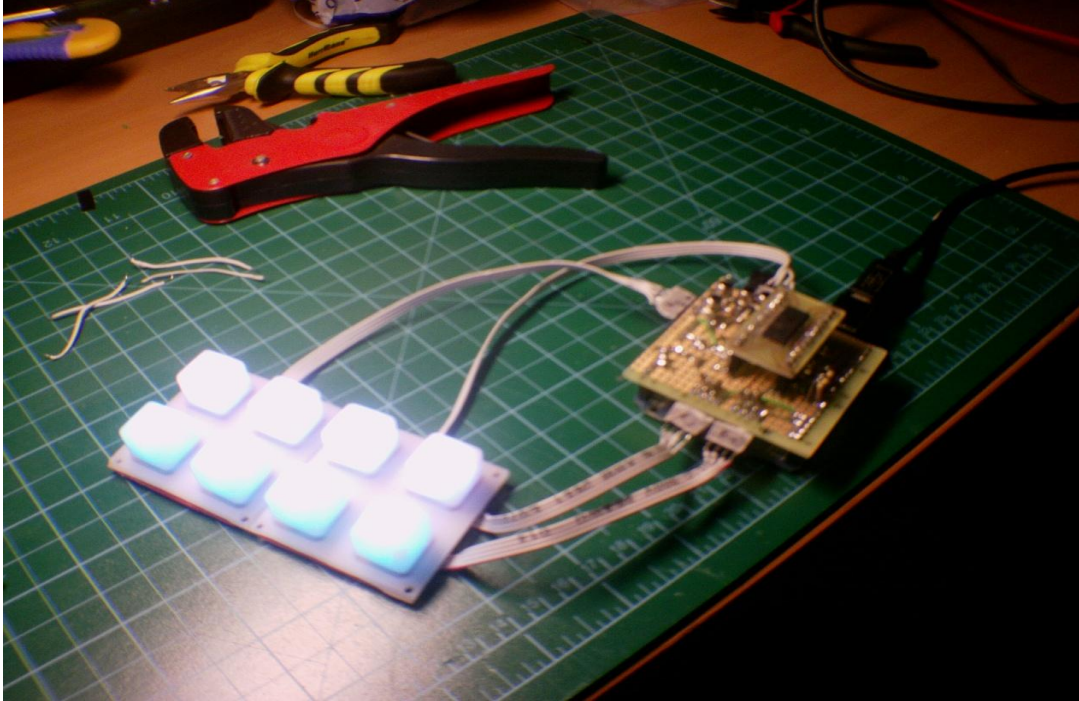
Ο προγραμματισμός του Arduino είναι εύκολος. Απλά μπορούμε να ‘κατεβάσουμε’ το πρόγραμμα [9].

Αφού ελέγχτηκαν τα κουμπιά μας, θέλαμε να δοκιμάσουμε και τα LEDs. Επομένως ‘Κατεβάσαμε’ την ρουτίνα RGB_light_fade [10] και ανεβάσαμε τη στο Arduino . Αυτό είναι το αγαπημένο μας demo γιατί δείχνει πραγματικά τις χρωματικές δυνατότητες ανάμειξης του ψηφιακού ποτενσιόμετρου.



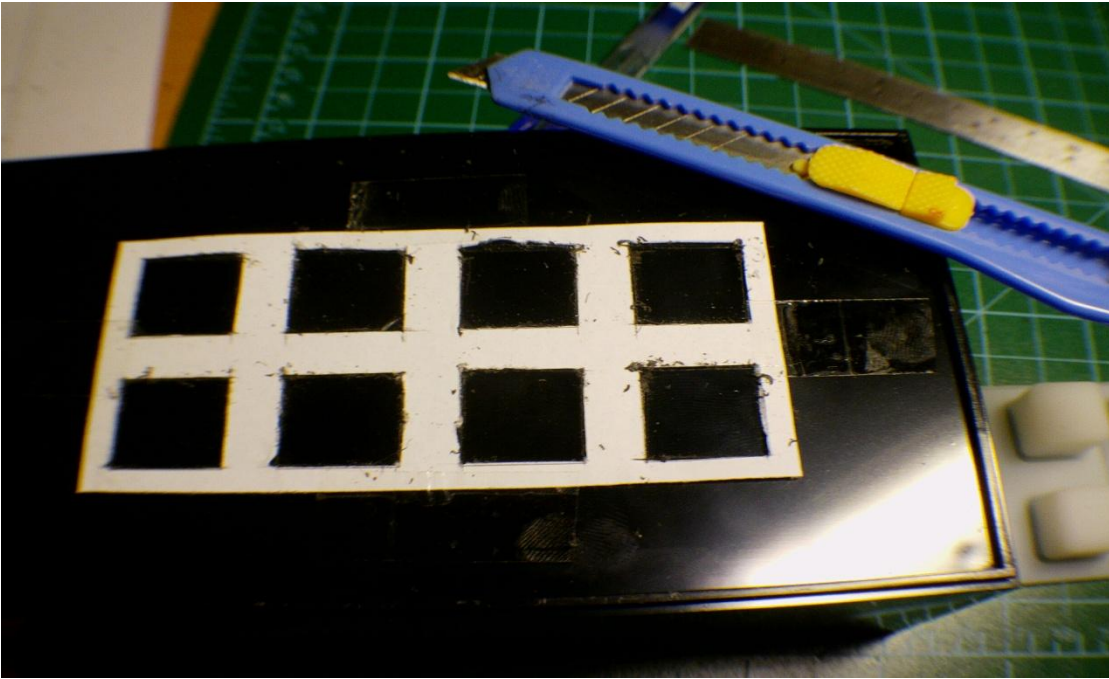
Εικόνα 2.20 – Σύστημα με demo κώδικα

Αφού δοκιμάσαμε το Arduino μας, και είδαμε ότι δουλεύει, ήρθε η ώρα να δοκιμάσουμε και το πρωτότυπο μας.

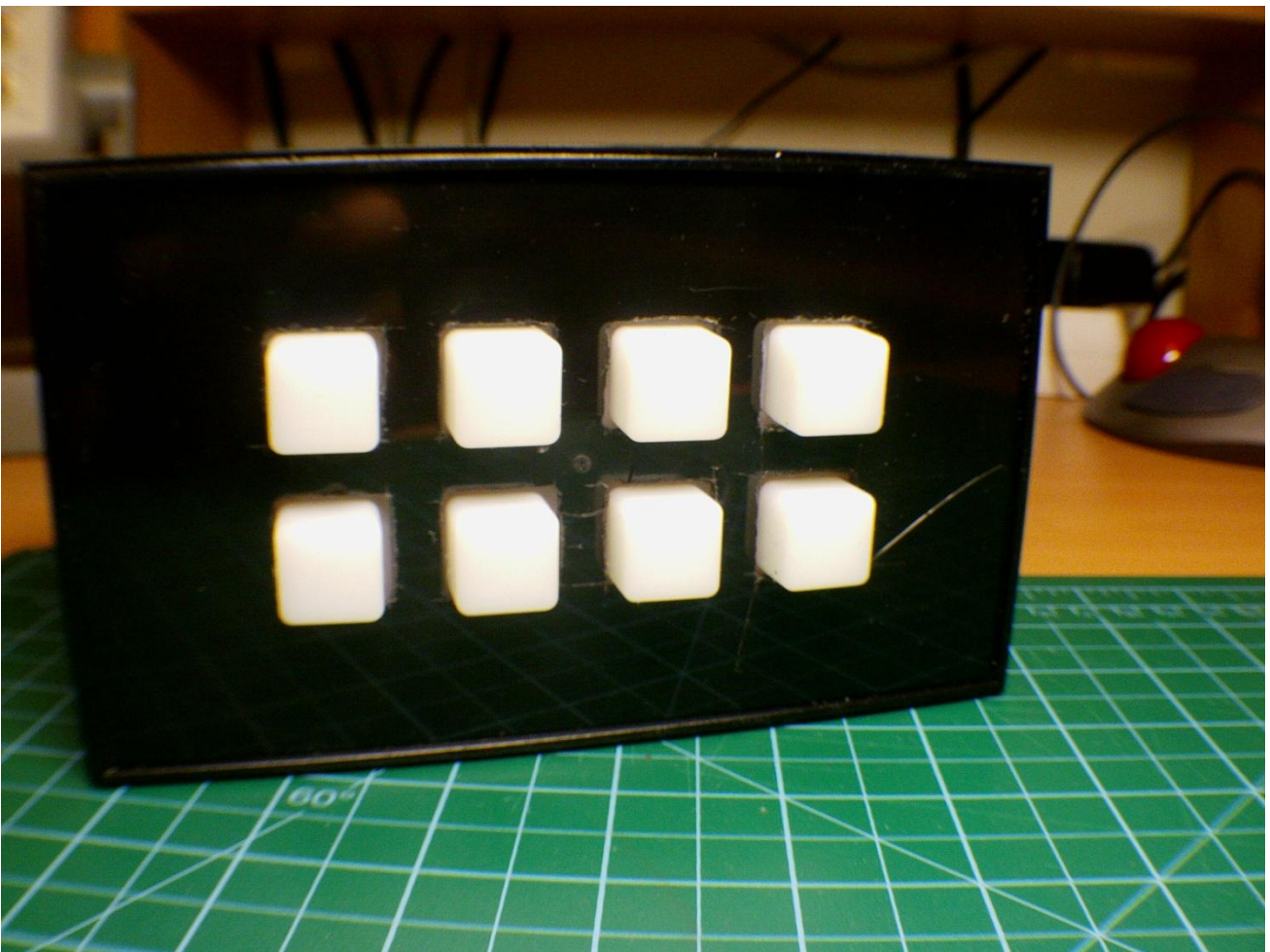


Εικόνα 2.20 – Σύστημα έχοντας φορτώσει τον τελικό κώδικα

Για την τελική κατασκευή, κόβουμε με ένα κοπίδι το κουτί ώστε να τοποθετήσουμε τα κουμπιά αλλά και ολόκληρο το σύστημα μας σωστά.



Εικόνα 2.21 – Κοπή πλαισίου για τοποθέτηση πληκτρολογίου

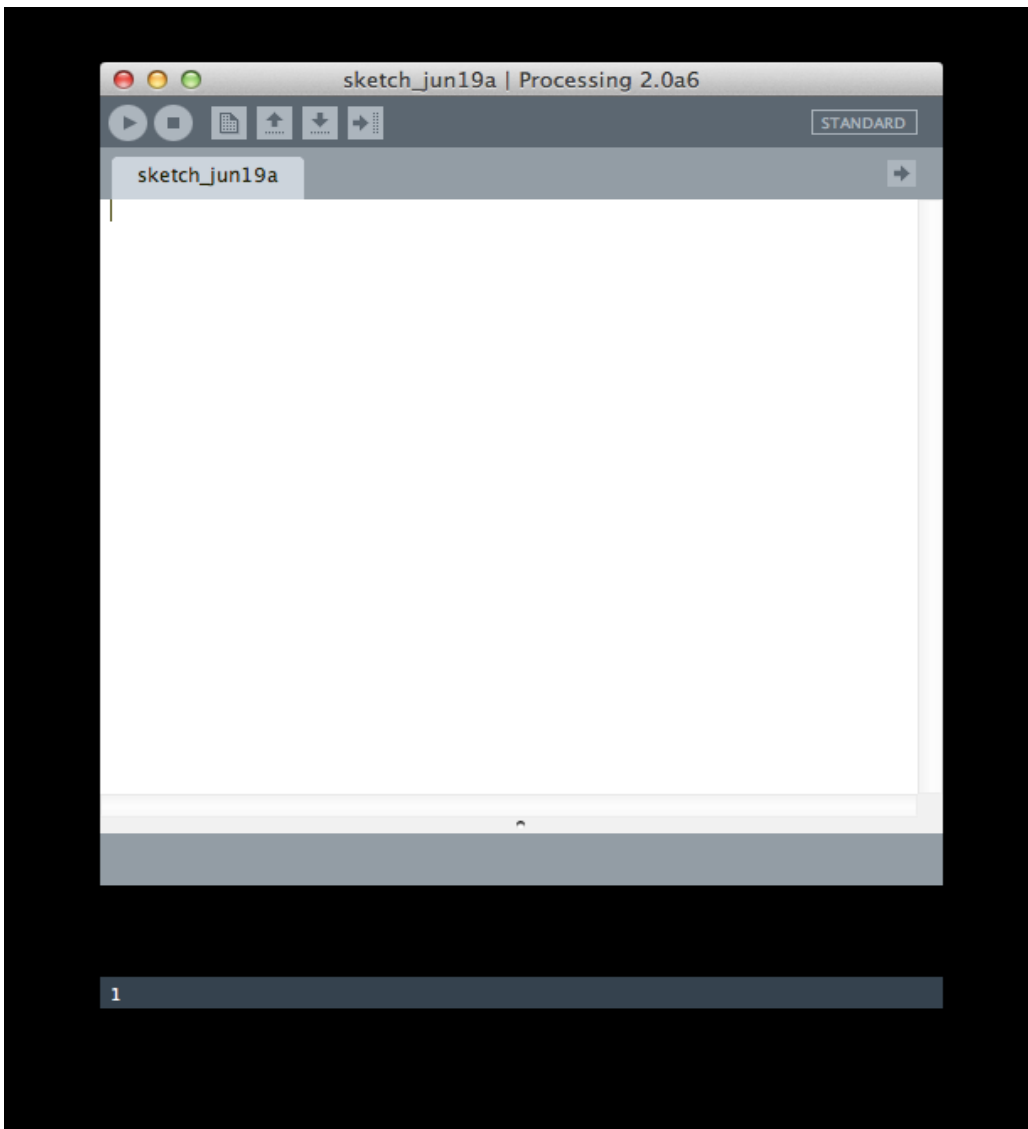


2.4 Σχεδιασμός του λογισμικού

Για την ανάπτυξη του λογισμικού θα χρησιμοποιηθεί η γλώσσα προγραμματισμού Processing.

2.4.1 Τι είναι η Processing

Η processing είναι μία γλώσσα προγραμματισμού ανοικτού κώδικα και παράλληλα ένα περιβάλλον προγραμματισμού για τα άτομα που θέλουν να δημιουργήσουν εικόνες (στάσιμες ή κινούμενες) και αλληλεπιδράσεις. Αρχικά αναπτύχθηκε για να χρησιμεύσει ως ένα sketchbook λογισμικού και να διδάξει βασικά στοιχεία του προγραμματισμού σε ένα οπτικό πλαίσιο. Όλα ξεκίνησαν το 2001 όταν δύο απόφοιτοι του πανεπιστημίου MIT, Benjamin Fry και Casey Reas ξεκίνησαν την ανάπτυξη της γλώσσας Processing πάνω σε Java. Παρόλο που η γλώσσα αναπτύχθηκε στη Java, το συντακτικό της είναι πιο απλό, με αποτέλεσμα γρήγορα να εξελιχθεί σαν ένα εργαλείο για τη δημιουργία επαγγελματικών εφαρμογών. Σήμερα, υπάρχουν δεκάδες χιλιάδες φοιτητές, καλλιτέχνες, σχεδιαστές, ερευνητές και ερασιτέχνες που χρησιμοποιούν την processing για εκμάθηση, προτυποποίηση και παραγωγή προγραμμάτων.



Εικόνα 2.4.1 - Το περιβάλλον processing IDE

Το σύστημα μας για την λειτουργία του, χρησιμοποιεί ένα απλό χρωματικό κώδικα. Αν εισαχθεί σωστά, οι ενδείξεις του πληκτρολογίου θα ανάψουν πράσινο και η κλειδαριά της πόρτας ανοίγει για 10 δευτερόλεπτα. Αν πατηθούν πάρα πολλά πλήκτρα, το πληκτρολόγιο κλειδώνει και ανάβει κόκκινο για 30 δευτερόλεπτα. Κατά τον κανονικό χρόνο αδράνειας, το πληκτρολόγιο προχώρα μέσα από κάθε πλήκτρο και τα leds ανάβουν μπλε μόνο για διασκέδαση.

2.4.2 Αλγόριθμος processing

2.4.2.1 Αλγόριθμος κυρίου μέρους (processing)

Ακολουθεί το κύριο μέρος του αλγορίθμου για το άνοιγμα της κλειδαριάς και την αποστολή των εντολών προς τον μικροελεγκτή:

```

#define DATAOUT 11
#define DATAIN 12
#define SPICLOCK 13
#define SLAVESELECT 10
#define COLS 4
#define ROWS 2
#define effect_select 1

const byte colpin[COLS] = {
  17,16,15,14};

const byte buttonWrite[ROWS] = {
  2, 3};
const byte buttonRead[COLS] = {
  9, 8, 7, 6};
boolean pressed[COLS][ROWS] = {
  0};
  const byte lock_pin = 4;
boolean lockState = 0;
boolean effect[COLS][ROWS] = {
  0};
byte effect_color = 3;
byte effect_state = 0;
byte effect_count = 0;

const byte red[2] = {
  5, 0};
const byte green[2] = {
  4, 1};
const byte blue[2] = {

```

```
2, 3};

byte rGrid[COLS][ROWS] = {
  0};
byte gGrid[COLS][ROWS] = {
  0};
byte bGrid[COLS][ROWS] = {
  0};

byte rCode[COLS][ROWS] = {
  0};
byte gCode[COLS][ROWS] = {
  0};
byte bCode[COLS][ROWS] = {
  0};
```

Αρχικά δηλώνουμε τις μεταβλητές.

AD 5206:#####

* Connect pot pin 5 to Arduino pin 10

* Connect pot pin 7 to Arduino pin 11

* Connect pot pin 8 to Arduino pin 13

* RED3 to a 150 ohm resistor to pot pin 14

* GREEN3 to a 100 ohm resistor to pot pin 11

* BLUE3 to a 100 ohm resistor to pot pin 2

* RED4 to a 150 ohm resistor to pot pin 23

* GREEN4 to a 100 ohm resistor to pot pin 20

* BLUE4 to a 100 ohm resistor to pot pin 17

- * Connect pins 1, 4, 9, 12, 15, 18, 19, and 22 to ground.
- * Connect AD5206 pins 3, 6, 10, 13, 16, 21 and 24 to 5v.

led grounds :#####

* pins 0 1 2 3 for led ground 1 2 3 4 (analog in)

Keyboard#####

* switch 3 adruino pin 2

* switch 4 adruino pin 3

* switch ground 1 -> pin6

* switch ground 2 -> pin7

switch ground 3 -> pin8

switch ground 4 -> pin9

```
byte COLORS = 4;//0 1 2 3
byte rColors[] = {0, 255, 0, 0 };
byte gColors[] = {0, 0, 255, 0 };
byte bColors[] = {0, 0, 0, 255};
byte colorcode[COLS][ROWS] = { 0 };
byte count;
```

Στο παραπάνω τμήμα του κώδικα, ορίζουμε τους συνδυασμούς χρωμάτων που θέλουμε.

```
void setup(){
  count = 0;
  colorcode[0][0] = 1;
  colorcode[1][0] = 3;
  colorcode[2][0] = 3;
  colorcode[3][0] = 1;
  colorcode[0][1] = 1;
  colorcode[1][1] = 2;
  colorcode[2][1] = 2;
```



```
colorcode[3][1] = 1;
```

Τον κωδικό τον εισάγουμε εδώ. 1= πράσινο , 2=μπλε , 3=κόκκινο , 4=κενό

```
Serial.begin(19200);
pinMode(lock_pin, OUTPUT);
digitalWrite(lock_pin, LOW);
for(int i = 0; i < ROWS; ++i){
  pinMode(buttonWrite[i], OUTPUT);
  digitalWrite(buttonWrite[i],LOW);
}
for(int j = 0; j<COLS; ++j) {
  pinMode(buttonRead[j], INPUT);
}
byte i;
byte clr;
pinMode(DATAOUT, OUTPUT);
pinMode(DATAIN, INPUT);
pinMode(SPICLOCK,OUTPUT);
pinMode(SLAVESELECT,OUTPUT);

for(byte c = 0; c < COLS; ++c){
  pinMode(colpin[c], OUTPUT);
  digitalWrite(colpin[c], LOW);
}

digitalWrite(SLAVESELECT,HIGH);
SPCR = (1<<SPE)|(1<<MSTR);
clr=SPSR;
clr=SPDR;
delay(10);

for (i=0;i<6;i++)
{
```

```

write_pot(i,0);
}
grid_init();
code_init();
effect_init();
}

void grid_init(){
for(byte c = 0; c < COLS; ++c){
for(byte r = 0; r < ROWS; ++r){
rGrid[c][r] = 0;
gGrid[c][r] = 0;
bGrid[c][r] = 0;
}
}
}

void effect_init(){
for(byte c = 0; c < COLS; ++c){
for(byte r = 0; r < ROWS; ++r){
effect[c][r] = 0;
}
}
effect[0][0] = 1;
}

```

Ρυθμίζουμε τις εισόδους και τις εξόδους των κουμπιών. Σετάρουμε τις γραμμές του πίνακα στις εξόδους και τις μηδενίζουμε και τις στήλες στις εισόδους. Αρχικοποιούμε τα leds των στηλών και τα απενεργοποιούμε.

Σβήνουμε όλες τις μεταβλητές του ποτενσιόμετρου, σβήνουμε τις μεταβλητές στα rgb leds και περνάμε τον κωδικό στη μνήμη και πλέον στις μεταβλητές δεν υπάρχουν δεδομένα.

```

void loop(){

  Serial.print(".");
  effect_count++;
  if(count > 25){
    color_effect(0, 0, 255, 20);
    grid_init();
    count = 0;
  }
}

```

Εάν ο κωδικός που έχουμε εισάγει είναι λανθασμένος, ανάβουν όλα τα leds κόκκινα για 20 δευτερόλεπτα.

```

for(byte r = 0; r < ROWS; ++r){
  digitalWrite(buttonWrite[r], HIGH);
  clear_pot();
  if(code_check()){
    count = 0;
    Serial.print("code matched!");
    open_door();
    grid_init();
  }
}

```

Αλλάζουμε την κατάσταση των σειρών δε high ώστε να διαβάζουν τις πιέσεις των κουμπιών. Ύστερα διαγράφει τις καταχωρήσεις του ποτενσιόμετρου μεταξύ των σειρών καθώς αλλιώς θα άναβε και η γραμμή που δεν διαβάζεται εκείνη τη στιγμή. Εάν ο κωδικός είναι σωστός ξεκλειδώνει η πόρτα και διαγράφει τον κωδικό , επιστρέφοντας στο σημείο ώστε να μπορείς να πληκτρολογήσεις ξανά νέο κωδικό.

```

for(byte c = 0; c < COLS; ++c){
  if(pressed[c][r] != digitalRead(buttonRead[c])){

```

```

pressed[c][r] = digitalRead(buttonRead[c]);
if(pressed[c][r]){

    if(count == 0){
        grid_init();
    }
    on_press(c, r);
    count++;
} else {
    on_release(r, c);
}
} else {
    if(pressed[r][c]){
        while_pressed(c, r);
    } else {
        while_released(c,r);
    }
}
if(count == 0){
    if(effect_count == 20) {
        idle_effect();
        effect_count = 0;
    }
}
write_pot(red[r],rGrid[c][r]);
write_pot(green[r],gGrid[c][r]);
write_pot(blue[r],bGrid[c][r]);

```

Μόλις αρχίσουμε να εισάγουμε έναν κωδικό αλλάζει η κατάσταση των κουμπιών. Στο πάτημα του κουπιού καλείται η ρουτίνα on_press και κατά την επαναφορά καλείται η on_release. Τέλος καταχωρεί την κατάσταση των χρωμάτων στον ψηφιακό καταχωρητή του ποτενσιόμετρου.

```
digitalWrite(colpin[c], HIGH);
```

```

delayMicroseconds(750);
digitalWrite(colpin[c], LOW);

}
digitalWrite(buttonWrite[r], LOW);
delay(4);

}

}

void on_press(byte c, byte r){
  Serial.print(c, DEC);
  Serial.print(" ");
  Serial.println(r, DEC);
  //set_lock();
  color_cycle(c, r);
}

void on_release(byte c, byte r){
}

void while_pressed(byte c, byte r){

}

void while_released(byte c, byte r){

}

void open_door(){

```

```

digitalWrite(lock_pin, HIGH);
color_effect(255, 0, 0, 5);
digitalWrite(lock_pin, LOW);
}

```

όσο εισάγουμε δεδομένα στο ποτενσιόμετρο μια στήλη ανάβει, εμφανίζεται και μετά πάλι σβήνει. Δίνουμε low στην κάθε σειρά. Σετάρουμε τη κλειδαριά .

```

char spi_transfer(volatile char data)
{
    SPDR = data;
    while (!(SPSR & (1<<SPIF)))
    {
    };
    return SPDR;
}

byte write_pot(byte address, byte value)
{
    digitalWrite(SLAVESELECT, LOW);
    spi_transfer(address % 6);
    spi_transfer(constrain(255-value,0,255));
    digitalWrite(SLAVESELECT, HIGH);
}

void rgb(byte c, byte r, byte R, byte G, byte B){
    rGrid[c][r] = R;
    gGrid[c][r] = G;
    bGrid[c][r] = B;
}

void clear_pot(){

```

```

byte i;
for (i=0;i<6;i++)
{
    write_pot(i,0);
}
}

```

Ξεκινά η μετάδοση, περιμένει να ολοκληρωθεί και επιστρέφει τα byte που έλαβε. Μόλις ολοκληρωθεί ελευθερώνει το chip , το σήμα και τελειώνει τη μετάδοση. Επίσης καθαρίζει όλους τους καταχωρητές του ποτενσιόμετρου.

```

void code_init(){
for(byte c=0; c < COLS; c++){
    for(byte r=0; r < ROWS; r++){
        rCode[c][r] = rColors[colorcode[c][r]];
        gCode[c][r] = gColors[colorcode[c][r]];
        bCode[c][r] = bColors[colorcode[c][r]];
    }
}
}

boolean code_check(){

for(byte c=0;c<COLS;c++){
    for(byte r=0; r < ROWS; r++){

        if(rGrid[c][r] != rCode[c][r]){
            return(0);
        }
        if(gGrid[c][r] != gCode[c][r]){
            return(0);
        }
    }
}
}

```

```

if(bGrid[c][r] != bCode[c][r]){
    return(0);
}
}
}
return(1);
}

```

Εδώ είναι που ορίζεται ο κωδικός που εισάγεται. Το c αντιστοιχεί σε στήλες και το r δηλώνει τις σειρές του πίνακα. Οι τιμές κυμαίνονται από 0-255. Οι μεταβλητές συνδυάζονται ώστε να δημιουργήσουν τα χρώματα. Ο κωδικός αποτελείται από τέσσερα χρώματα να εμφανίζονται σε κάθε σειρά και έπειτα τσεκάρει τα χρώματα που εμφανίζονται, ώστε να τα συγκρίνει με τον κωδικό που έχουμε εισάγει αρχικά.

```

void color_effect(byte dRed, byte dGreen, byte dBlue, byte time){
    byte c;
    for(byte r=0; r<ROWS; r++){
        write_pot(red[r],dRed);
        write_pot(green[r],dGreen);
        write_pot(blue[r],dBlue);
    }
    for(c=0;c<COLS;c++){
        digitalWrite(colpin[c], HIGH);
    }
    delay(time*1000);
    for(c=0;c<COLS;c++){
        digitalWrite(colpin[c], LOW);
    }
    clear_pot();
}

```

Επηρεάζει όλα τα κουμπιά του πληκτρολογίου ώστε να ανάβουν για η δευτερόλεπτα και μετά να σβήνουν.


```

void color_cycle(byte c, byte r){
  byte color = get_color(c, r);
  Serial.print("got color");
  Serial.print( color, DEC );
  if(color < COLORS){
    color++;
  } else {
    color = 1;
  }
  rgb(c, r, rColors[color], gColors[color], bColors[color]);
}

```

```

byte get_color(byte c, byte r){
  for(byte i=0; i < COLORS; i++){
    if(rGrid[c][r] == rColors[i]){
      if(bGrid[c][r] == bColors[i]){
        if(gGrid[c][r] == gColors[i]){
          return(i);
        }
      }
    }
  }
}

```

```

void idle_effect(){ //this is he eye candy while the keypad isn't in use.
  grid_init();
  if(effect_select==1)
  {
    for(byte r=0; r < ROWS; r++){
      for(byte c=0;c<COLS;c++){
        if(effect_state == 1){
          effect[c][r] = 1;

```

```

effect_state = 0;
return;
}
if(effect[c][r]) {
    rGrid[c][r] = rColors[effect_color];
    gGrid[c][r] = gColors[effect_color];
    bGrid[c][r] = bColors[effect_color];
    effect[c][r] = 0;
    effect_state = 1;
}

}
}
}
if(effect_select==2){

byte red = rColors[effect_color];
byte blue = bColors[effect_color];
byte green = gColors[effect_color];
if(red != 0)
    red = red - effect_state;

for(byte r=0; r < ROWS; r++){
    for(byte c=0;c<COLS;c++){

        rGrid[c][r] = rColors[effect_color];
        gGrid[c][r] = gColors[effect_color];
        bGrid[c][r] = bColors[effect_color];
        effect[c][r] = 0;
        effect_state = 1;

```

```
}  
}  
  
}  
}
```

Τέλος αναφορικά με τις παραπάνω τρεις ρουτίνες η λειτουργία τους είναι η εξής:

Η ρουτίνα “color_cycle” κάθε φορά που πατάμε ένα κουμπί τρέχει τη δεύτερη ρουτίνα η οποία συγκρίνει τα περιεχόμενα του πίνακα bGrid με τα αντίστοιχα των πινάκων που περιέχουν τις τιμές χρωμάτων rColors , bColors ,gColors (red, green, blue) και επιστρέφει το χρώμα που έχει το κουμπί που πατήσαμε. Αφού βρει τι χρώμα είχε, αυξάνει κατά ένα το χρώμα του κουμπιού έτσι ώστε από πράσινο να γίνει κόκκινο κτλ.

Και η λειτουργία της τρίτης ρουτίνας είναι το εφέ που τρέχει όταν η κλειδαριά, απλά τροφοδοτείται με ρεύμα.

Κεφάλαιο 3:

Μελλοντικές επεκτάσεις συστήματος

Όπως κάθε σύστημα που στηρίζεται σε αισθητήρες, μικροελεγκτές, αλλά και στο διαδίκτυο υπάρχουν σημαντικά θέματα λειτουργίας που λόγω της τεχνολογικής εξέλιξης, απαιτούν συνεχή βελτίωση. Ιδιαίτερα όταν πρόκειται για συστήματα που έχουν πρακτική εφαρμογή, τότε η λειτουργία τους θα πρέπει να είναι τέτοια ώστε να ανταποκρίνεται και στις πιο απαιτητικές συνθήκες, όπως και στους πιο απαιτητικούς χρήστες. Οι μελλοντικές

Επεκτάσεις του συστήματος, που αποτελούν το αντικείμενο αναφοράς του κεφαλαίου αυτού, ασχολούνται με :

1. Θέματα τροφοδοσίας
2. Θέματα χρήσης συσκευών δράσης
3. Θέματα γενικής χρήσης

3.1 Μελλοντικές επεκτάσεις σε θέματα τροφοδοσίας

Η αρχική έκδοση του συστήματος χρησιμοποιεί ηλεκτρικό ρεύμα μέσω του αντιστοιχου φορέα παροχής, για την τροφοδοσία της. Όπως είναι όμως φυσιολογικό, η χρήση του είναι μεν αξιόπιστη αλλά δεν καλύπτει τον χρήστη απόλυτα. Παρακάτω, αναλύονται ιδέες για τη βελτιστοποίηση του συστήματος όσο αφορά την τροφοδοσία του.

3.1.1 Χρήση μπαταρίας

Η χρήση μπαταρίας, είτε αλκαλικής είτε λιθίου, αποτελεί μια βελτιστοποίηση του συστήματος. Ο λόγος που χρειάζεται η παρουσία της στο σύστημα, προέκυψε κατά τη διάρκεια των πειραματικών δοκιμών και πιο συγκεκριμένα κατά τη διακοπή παροχής ρεύματος στο σύστημα μέσω πρίζας. Η διακοπή του ρεύματος στο σύστημα έχει τις παρακάτω επιπτώσεις :

1. Το σύστημα δε μπορεί να λειτουργήσει ούτε να ελέγξει τις συσκευές δράσης. Σε περίπτωση που οι συσκευές δράσης, τροφοδοτούνται από διαφορετική παροχή, η οποία δε σταματήσει να τροφοδοτείται με ρεύμα, τότε η κατάστασή τους θα παραμείνει σύμφωνα με την τελευταία λήψη εντολών από τη βάση. Το γεγονός αυτό μπορεί να προκαλέσει εσφαλμένη λειτουργία ενός άλλου υποσυστήματος, όπως για παράδειγμα, την αναίτια ενεργοποίηση ενός συνδεδεμένου με το πληκτρολόγιο μας, συστήματος ασφαλείας (συναγερμός) .

2. Ο χρήστης, με το που διαπιστωθεί απώλεια σύνδεσης, δεν μπορεί να γνωρίζει ποια είναι η αιτία, δηλαδή εάν πρόκειται για πρόβλημα στην επικοινωνία του συστήματος εσωτερικά ή στη παροχή ρεύματος.

Για να ελαχιστοποιηθούν οι πιθανότητες εμφάνισης τέτοιων παθογενειών στο σύστημα, η χρήση μπαταρίας αποτελεί μια λύση. Με τη παρουσία της, θα μπορεί να υπάρχει η λειτουργία της ειδοποίησης για διακοπή ρεύματος και ενεργοποίησης συγκεκριμένων ρυθμίσεων από τον χρήστη. Επομένως ο χρήστης θα λαμβάνει ειδοποίηση (αν φυσικά το επιθυμεί) για το εάν έχει διακοπεί το ρεύμα και θα έχει ρυθμίσει τη συμπεριφορά της συσκευής σε ανάλογη περίπτωση (αδρανοποίηση

των συσκευών δράσης κτλ). Στόχος της χρήσης της μπαταρίας δεν είναι η ολική αποκατάσταση της τροφοδοσίας, αλλά η ειδοποίηση του χρήστη για το γεγονός της διακοπής.

3.1.2 Χρήση ανανεώσιμων πηγών ενέργειας

Η χρήση της ηλιακής ή της αιολικής ενέργειας, ανάλογα πάντα και την τοποθεσία την οποία εγκαθίσταται το σύστημα μπορεί να αντικαταστήσει την παροχή ρεύματος από τους σχετικούς φορείς. Φυσικά υπάρχει και το κόστος της αρχικής εγκατάστασης, αλλά η απόσβεση είναι δεδομένη. Εκτός αυτού η συσκευή γίνεται αυτόνομη όσο αφορά τις ενεργειακές της ανάγκες, με αποτέλεσμα να μην απειλείται από θέματα παροχής ρεύματος.

3.2 Μελλοντικές επεκτάσεις ως προς τη χρήση συσκευών δράσης

Η εφαρμογή επιπλέον συσκευών δράσης θα βελτιώνει το ποσοστό αντίληψης της συσκευής και κατ' επέκταση του συστήματος. Για παράδειγμα, αντί για το υπάρχον πληκτρολόγιο, θα μπορούσε να χρησιμοποιηθεί ένα πληκτρολόγιο αφής. Επίσης θα μπορούσε να υπάρχει οθόνη και να σου δίνει την επιλογή να ορίσεις νέο κωδικό.

3.3 Μελλοντικές επεκτάσεις γενικής χρήσης συστήματος

Το σύστημα ασφαλείας με χρωματικό κώδικα, απαιτεί για την εγκατάσταση του σε κάθε χρήστη, ξεχωριστή αντιμετώπιση. Το γεγονός αυτό προκαλείται από τον πλουραλισμό αισθητήρων και συσκευών δράσης που υπάρχουν στην αγορά, αλλά και από το γεγονός της διαφορετικής αντίληψης από άνθρωπο σε άνθρωπο. Ανάλογα με τους αισθητήρες και τις συσκευές δράσης που θέλει ο χρήστης γίνεται και ο αντίστοιχος προγραμματισμός του συστήματος.

Η διαφορετική αντιμετώπιση του κάθε χρήστη προγραμματιστικά είναι ένα πρόβλημα για τον δημιουργό του συστήματος, καθώς συνέχεια θα χρειάζεται να επεμβαίνει τόσο στο λογισμικό μέρος όσο και στο υλικό μέρος του συστήματος. Για αυτό μια λύση θα ήταν η ανάπτυξη πλατφόρμας ανοιχτού λογισμικού. Οι χρήστες ανάλογα με τις απαιτήσεις τους θα μπορούν να συμβάλουν στην ανάπτυξη του συστήματος σε επίπεδο λογισμικού.

Βιβλιογραφία - πηγές πληροφοριών:

Αναφορές- links :

[1] <http://arduino.cc/en/Guide/Windows>

[2] <http://arduino.cc/en/Tutorial/HomePage>

[3] <http://createdigitalmusic.com/2008/08/arduinome-an-arduino-based-monome-clone-behind-the-scenes/>

[4] <https://www.sparkfun.com/products/7835>

[5] <https://www.sparkfun.com/products/8033>

[6] <https://www.sparkfun.com/products/105>

[7] <http://appliedelectronicsengineering.blogspot.gr/2013/04/download-cadsoft-eagle-professional-63.html>

[8] <http://www.arduino.cc/en/Tutorial/SPIDigitalPot>

[9] <http://www.arduino.cc/en/Main/Software>

[10] http://biobug.org/rgb-keypad/RGB-howto/RGB_light_fade.pde

Ιστοσελίδες:

- wiseGEEK <http://www.wisegeek.com/what-is-a-microcontroller.htm>
- Βικιπαιδεία <http://el.wikipedia.org/wiki/Μικροελεγκτής>
- wikipedia <http://en.wikipedia.org/wiki/Arduino>
- arduino <http://arduino.cc/>

Βιβλία:

- LEGO MINDSTORMS NXT: Mars Base Command by James Floyd Kelly ISBN-10: 1430238046
- Arduino A Quick-Start Guide by Maik Schmidt) ISBN-10: 1-934356-66-2
- Programming Interactivity, Second Edition by Joshua Noble ISBN: 978-1-449-31144-5
- Make: Arduino Bots and Gadgets by Kimmo and Tero Karvinen ISBN: 978-1-449-38971-0
- Getting Started with Arduino by Massimo Banzi (Second Edition) ISBN: 978-1-449-309879
- Arduino Cookbook by Michael Margolis ISBN: 978-1-449-31387-6
- Beginning Arduino Programming Writing Code for the Most Popular Microcontroller Board in the World by Brian Evans ISBN-13 (pbk): 978-1-4302-3777-8

- Arduino Robotics by John-David Warren, Josh Adams, and Harald Molle ISBN-13 (pbk):
978-1-4302-3183-7