



Εισαγωγή στο Arduino

Μαριάντζελα Κομνηνού
ΙΤΥΕ-ΛΙΟΦΑΝΤΟΣ

Η προσπάθεια αυτή έγινε στο πλαίσιο του προγράμματος πρακτικής άσκησης φοιτητών του Πανεπιστημίου Πατρών ως φοιτήτρια του Τμήματος της Επιστήμης των Υλικών. Η πρακτική έγινε στο Ινστιτούτο Τεχνολογίας Υπολογιστών με συνεργασία των Διευθύνσεων Τηλεματικής & Εφαρμογών Περιφερειακής Ανάπτυξης, Εκπαιδευτικής Τεχνολογίας και Τεχνικής Υποστήριξης Συστημάτων.

Η επίβλεψη της πρακτικής άσκησης έγινε από τον Διευθυντή της Διεύθυνσης Τηλεματικής και Εφαρμογών Περιφερειακής Ανάπτυξης καθηγητή του Τμ. Μηχανικών Η/Υ και Πληροφορικής κ. Γιάννη Γαροφαλάκη τον οποίο ευχαριστώ θερμά για την ευκαιρία που μου έδωσε να εκπαιδευτώ σε ένα δομημένο και επαγγελματικό περιβάλλον ανάμεσα σε εξαιρετικούς επιστήμονες.

Επίσης ευχαριστώ τα στελέχη των Διευθύνσεων του ΙΤΥΕ που με βοήθησαν στην μελέτη του υλικού, στην διόρθωση και μορφοποίηση των κειμένων, στην σύνδεση των υλικών, στην παροχή εξοπλισμού, και στην μεταφορά τεχνογνωσίας στην αξιοποίηση λογισμικού σχεδίασης, σε λογισμικό προσομοίωσης, σε πληθώρα βάσεων δεδομένων με αισθητήρες και μικροηλεκτρονικά.

Κατά τη διάρκεια της πρακτικής άσκησης ο σκοπός μας ήταν να δημιουργηθούν οι πρώτες σημειώσεις/οδηγοί για την αξιοποίηση των Edulabs που φτιάχτηκαν πιλοτικά για τα Δημόσια Σχολεία. Το υλικό θα αναρτηθεί στο Αποθετήριο Εφαρμογών του Υπουργείου Παιδείας και Θρησκευμάτων και των Εποπτευόμενων Φορέων (<https://github.com/itminedu>).

Πρώτα δημιουργήσαμε ένα οδηγό για τα Βασικά της γλώσσας C στον προγραμματισμό της πλατφόρμας μικροϋπολογιστών της οικογένειας των Arduino. Για τον προγραμματισμό και τα παραδείγματα χρησιμοποιήθηκαν ανοιχτά/ελεύθερα λογισμικά όπως ο compiler TDM-GCC MinGW (<https://sourceforge.net/projects/tdm-gcc/>) και το περιβάλλον Code::Blocks v.16.01 (<http://www.codeblocks.org/> Open source, cross platform, free C, C++ and Fortran IDE).

Κατόπιν με ένα δεύτερο οδηγό περιγράψαμε τις ιδιαίτερες πτυχές του προγραμματισμού σε Arduino IDE (www.arduino.cc), τα βασικά ηλεκτρονικά και ηλεκτρολογικά υλικά που χρησιμοποιούνται στην πλατφόρμα και δημιουργήσαμε παραδείγματα χρήσης/υλοποίησης με αυξανόμενο βαθμό δυσκολίας.

Έπειτα, δημιουργήθηκε ένας Οδηγός «Ξεκινήστε γρήγορα με το Arduino» στον οποίο είναι συγκεντρωμένες οι εντολές της C, του Arduino και βασικά ηλεκτρονικά.

Τέλος με τα στοιχεία αυτά φτιάχτηκαν οδηγίες υλοποίησης για 4 παραδείγματα / πειράματα στην πράξη.

Μέσα από την πρακτική άσκηση βελτίωσα τις γνώσεις μου, αξιοποιώντας αυτά που έμαθα στη σχολή μου υπό την οπτική του μηχανικού, του προγραμματιστή και του εκπαιδευτή.

Ευχαριστώ ιδιαίτερα τον καθηγητή μου κ. Δημήτρη Αλεξανδρόπουλο στο Τμήμα Επιστήμης των Υλικών και τον καθηγητή Γιάννη Γαροφαλάκη που με επέβλεπαν ως υπεύθυνοι της πρακτικής εργασία μου στο ΙΤΥΕ.

Μαρία-Αγγελική Κομνηνού

Πάτρα, Δεκέμβριος 2016-Φεβρουάριος 201

Περιεχόμενα

| | |
|---|----|
| Εικόνες | 4 |
| Πίνακες | 4 |
| 1. Βασικά Στοιχεία του Arduino | 5 |
| 1.1 Μικροελεγκτής – Η «καρδιά» του Arduino | 5 |
| 1.2 Είσοδοι & Έξοδοι..... | 6 |
| 1.3 Τροφοδοσία | 8 |
| 1.4 Ηλεκτρονικά εξαρτήματα που συνδέονται στο Arduino | 10 |
| 2. Περιβάλλον Προγραμματισμού | 13 |
| 2.1 Άλλα περιβάλλοντα προγραμματισμού..... | 16 |
| 3. Εντολές Προγραμματισμού | 19 |
| 3.1 Σταθερές..... | 19 |
| HIGH LOW..... | 19 |
| INPUT OUTPUT | 19 |
| LED_BUILTIN..... | 19 |
| 3.2 Ψηφιακές Λειτουργίες (Digital Functions) | 20 |
| pinMode() | 20 |
| digitalWrite()..... | 21 |
| digitalRead() | 21 |
| 3.3 Αναλογικές Λειτουργίες (Analog) | 22 |
| analogRead() | 22 |
| analogWrite() | 22 |
| 3.4 Χρονικές Λειτουργίες | 23 |
| millis()..... | 23 |
| micros() | 24 |
| delay() | 24 |
| 3.5 Δομή προγράμματος για Arduino | 25 |
| setup() | 25 |
| loop()..... | 26 |
| 4. Επικοινωνία του χρήστη με το Arduino | 27 |
| 4.1 Σειριακά pins | 28 |
| 4.2 Αρχικοποίηση Σειριακής..... | 29 |
| 4.3 Εκτύπωση στην Σειριακή..... | 29 |
| 4.4 Διάβασμα από τη Σειριακή | 30 |
| 5. Παραδείγματα με Εκτελέσιμα Προγράμματα σε Arduino | 32 |
| 6. Βιβλιογραφία | 42 |

Εικόνες

| | |
|---|----|
| Εικόνα 1 Πλακέτα Arduino Uno..... | 5 |
| Εικόνα 2 Ανάλυση στοιχείων σε Duemilanova Arduino Board..... | 7 |
| Εικόνα 3 Προγραμματιστικό περιβάλλον Arduino IDE..... | 13 |
| Εικόνα 4 Συνδέσεις breadboard πλακέτας..... | 33 |
| Εικόνα 5 PWM με analogWrite..... | 37 |
| Εικόνα 6 Διαδοχική ενεργοποίηση LED (αριστερά το λευκό, δεξιά στη φάση 3 το πράσινο)..... | 39 |
| Εικόνα 7 Σχηματικό σύνδεσης Push Button..... | 41 |
| Εικόνα 8 Διαδρομή ρεύματος με αντιστάτη..... | 41 |

Πίνακες

| | |
|--|----|
| Πίνακας 1-1 Διάφορα είδη Arduino Boards..... | 9 |
| Πίνακας 1-2 Βασικά εξαρτήματα για Arduino..... | 11 |

1. Βασικά Στοιχεία του Arduino

Το Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «πρωτοτυποποίησης» συστήματος ελέγχου ηλεκτρονικών βασισμένη σε ευέλικτο και εύκολο στη χρήση υλικό και λογισμικό. Η πλατφόρμα απευθύνεται σε οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά συστήματα ή περιβάλλοντα ελέγχου και αντίδρασης σε εξωτερικά ερεθίσματα ή συνθήκες αλλά και σε αυτούς που έχουν τη διάθεση να φτιάξουν τις δικές τους κατασκευές από το μηδέν.

Το Arduino ουσιαστικά πρόκειται για μία ηλεκτρονική πλακέτα που βασίζεται στον μικροελεγκτή ATmega της Atmel και του οποίου το λογισμικό αλλά κι όλα τα σχέδια που χρειάζεται για την λειτουργία του, διανέμονται ελεύθερα και δωρεάν ώστε να δίνεται η δυνατότητα να κατασκευαστεί από τον καθένα. Πάνω του μπορούμε να συνδέουμε πολλές μονάδες εισόδου (π.χ. αισθητήρες, επαφές) και εξόδου (π.χ. Led, μοτέρ), να επεξεργαζόμαστε τα δεδομένα τους και να στέλνονται οι κατάλληλες οδηγίες στις εξόδους, ώστε να εφαρμόζεται μία αντίδραση στο ερέθισμα. Μπορούμε να πούμε ότι το Arduino λειτουργεί σαν υπολογιστής ή καλύτερα σαν ελεγκτής καταστάσεων δράσης και αντίδρασης.



Εικόνα 1 Πλακέτα Arduino Uno

Λόγω του ότι απευθύνεται και σε αρχάριους -οι οποίοι μπορεί να μην έχουν όλες τις απαραίτητες γνώσεις να κατασκευάσουν εξ αρχής την πλακέτα- κυκλοφορούν έτοιμες πλακέτες Arduino διαφόρων ειδών. Για τους αρχάριους υπάρχει Arduino Starter Kit, το οποίο περιέχει διάφορα εξαρτήματα κι αισθητήρες που μπορούν να χρειαστούν στα πρώτα «πειράματα»/εφαρμογές με Arduino.

Ας δούμε περιληπτικά τι περιλαμβάνει μία πλακέτα Arduino.

1.1 Μικροελεγκτής – Η «καρδιά» του Arduino

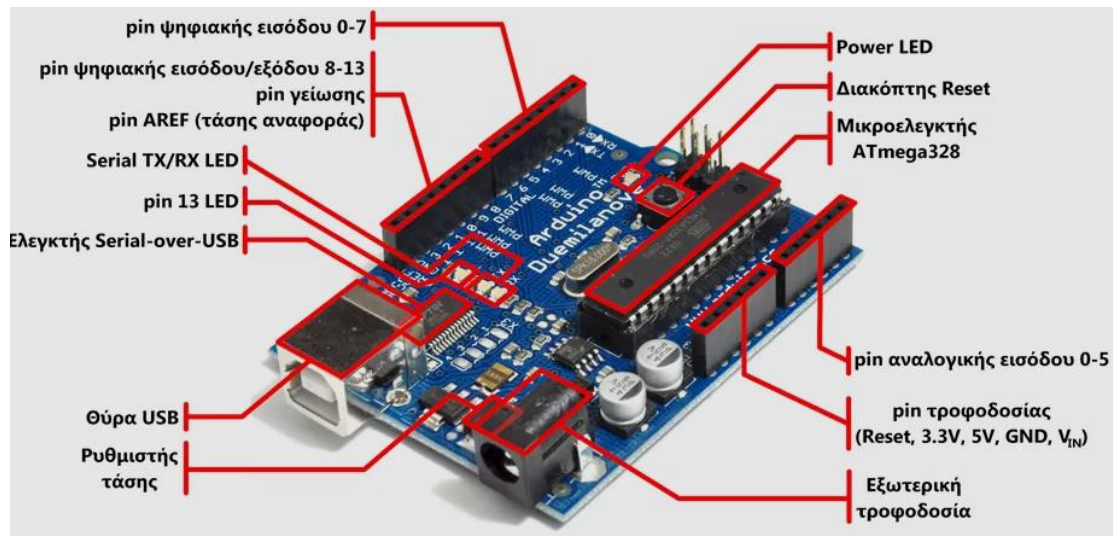
Το Arduino βασίζεται κυρίως σε επεξεργαστές (CPU) ATMEL όπως π.χ. ο ATmega328, που είναι ένας 8-bit RISC μικροελεγκτής, που έχει ρολόι που «τρέχει» στα 16MHz. Οι επεξεργαστές διαθέτουν ενσωματωμένη μνήμη τριών τύπων και ο ATmega328 έχει:

- 2Kb μνήμης SRAM που είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματά μας ώστε να αποθηκεύουν μεταβλητές, πίνακες κ.λπ. κατά την διάρκεια που «τρέχει» το πρόγραμμα (runtime). Αυτή η μνήμη όμως, χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή αν γίνει reset, όπως άλλωστε συμβαίνει και σε έναν υπολογιστή.
- 1Kb μνήμης EEPROM η οποία μπορεί να χρησιμοποιηθεί για απευθείας εγγραφή/ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τα προγράμματά μας κατά το runtime. Σε αντίθεση με την SRAM, η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset, έτσι μπορούμε να το παρομοιάσουμε με το σκληρό δίσκο ενός υπολογιστή.
- 32Kb μνήμης Flash, από τα οποία τα 2Kb χρησιμοποιούνται από το λογισμικό συστήματος (firmware) του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware, αυτό που στη «γλώσσα» του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των δικών μας προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer ενώ, τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για να αποθηκεύονται αυτά ακριβώς τα προγράμματα, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Σημειώνεται πως ούτε η μνήμη Flash χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset.

Υπάρχουν πολλών ειδών πλακέτες (boards) για διαφορετικές χρήσεις και με διαφορετικούς μικροεπεξεργαστές (CPUs) όπως ATmega168P, ATmega328P, ATmega2560, ATmega32u4, ATSAM3X8E και πολλά άλλα. Οι διαφορές τους είναι στην ταχύτητα της CPU (8MHz-84MHz), στις αναλογικές εισόδους (4 ως 16) και εξόδους (0 ως 2), τις ψηφιακές εισόδους (14 ως 54) και εξόδους (4 ως 15) το μέγεθος των τριών διαφορετικών ειδών της μνήμης, την τάση τροφοδοσίας και λειτουργίας, κλπ.

1.2 Είσοδοι & Έξοδοι

Συνήθως τα περισσότερα Arduino διαθέτουν σειριακό interface. Ο μικροελεγκτής ATmega υποστηρίζει σειριακή επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή Serial-over-USB ώστε να συνδέεται με τον υπολογιστή μέσω USB. Αυτή η σύνδεση χρησιμοποιείται με σκοπό τη μεταφορά των προγραμμάτων (upload) που γράφονται από τον υπολογιστή στο Arduino αλλά και για αμφίδρομη επικοινωνία Arduino και υπολογιστή μέσα από το πρόγραμμα, όσο αυτό εκτελείται.



Εικόνα 2 Ανάλυση στοιχείων σε Duemilanova Arduino Board

Αναλυτικότερα, στην αριστερή πλευρά του Arduino (βλ. Εικόνα 2) είναι τοποθετημένα 14 θηλυκές θύρες (pins), αριθμημένα από 0 ως 13, που μπορούν να λειτουργούν ως ψηφιακές εισοδοι και έξοδοι (Input/Output-IO), για τη συγκεκριμένη πλακέτα στα 5V και καθένα μπορεί να παρέχει ή να δεχτεί το πολύ 40mA ρεύμα.

Αν είναι ψηφιακή έξοδος, το pin μπορεί να τεθεί από το πρόγραμμά μας σε κατάσταση HIGH ή LOW, οπότε το Arduino θα ξέρει αν πρέπει να διοχετεύσει ή όχι το ρεύμα λειτουργίας του στο συγκεκριμένο pin. Συνεπώς, μπορούμε να ανάψουμε και να σβήσουμε ένα LED συνδεδεμένο στο συγκεκριμένο pin. Αντιστοίχως, εάν ρυθμιστεί ένα από αυτά τα pin ως ψηφιακή είσοδος, τότε, μπορούμε με την κατάλληλη εντολή να διαβάσουμε την κατάστασή του (HIGH ή LOW).

Μερικά από αυτά τα 14 pin, εκτός από ψηφιακές εισοδοι/έξοδοι έχουν και δεύτερη λειτουργία. Συγκεκριμένα:

- Τα pin 0 και 1 λειτουργούν ως υποδοχή RX και αποστολή TX της σειριακής όταν το πρόγραμμά μας ενεργοποιεί την σειριακή θύρα. Συνεπώς, όταν το πρόγραμμα στέλνει δεδομένα στην σειριακή, αυτά προωθούνται και στην θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin 1).



Αν στο πρόγραμμά ενεργοποιήσουμε το σειριακό interface, χάνουμε αυτόματα 2 ψηφιακές εισόδους/εξόδους ή αν χρησιμοποιούμε τα pin 0 και 1 π.χ. για σύνδεση με αισθητήρες τότε δεν μπορούμε να έχουμε και λειτουργία σειριακής.

- Τα pin 2 και 3 λειτουργούν και ως εξωτερικά interrupts (interrupt 0 και 1 αντίστοιχα). Αυτό σημαίνει πως μπορούμε να τα ρυθμίσουμε μέσω του προγράμματος με σκοπό να λειτουργούν αποκλειστικά ως ψηφιακές εισοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική

ροή του προγράμματος σταματάει άμεσα κι εκτελείται μια συγκεκριμένη συνάρτηση.

Αξίζει να σημειωθεί πως είναι πολύ χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

- Τα pin 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές εξόδους με το σύστημα PWM (Pulse Width Modulation), δηλαδή με το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Συνεπώς, μπορούμε να συνδέσουμε παραδείγματος χάρη ένα LED σε κάποιο από αυτά τα pin και να ελέγχουμε πλήρως την φωτεινότητά του με ανάλυση 8bit (256 καταστάσεις από 0-σβηστό ως 255-πλήρως αναμμένο) αντί να μπορούμε απλά να το αναβοσβήνουμε όπως συμβαίνει με τις υπόλοιπες ψηφιακές εξόδους.

Στην δεξιά πλευρά του Arduino (Εικόνα 2), με τη σήμανση ANALOG IN, βρίσκεται ακόμη μια σειρά από 6 pin, αριθμημένα από το 0 ως το 5. Το καθένα από αυτά λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένο στον μικροελεγκτή. Παραδείγματος χάρη, μπορούμε να τροφοδοτήσουμε ένα από αυτά με μια τάση η οποία μπορεί να κυμαίνεται από 0V ως μια τάση αναφοράς V_r η οποία, αν δεν κάνουμε κάποια αλλαγή είναι προ ρυθμισμένη στην τάση λειτουργίας της πλακέτας, εδώ στα 5V, με τη χρήση π.χ. ενός ποτενσιόμετρου.

1.3 Τροφοδοσία

Στην πλειοψηφία τους, οι πλακέτες Arduino μπορούν να τροφοδοτηθούν από μπαταρία ή εξωτερικό τροφοδοτικό, ακόμα κι από το ίδιο USB που χρησιμοποιούμε για να μεταφέρουμε το πρόγραμμα. Η τάση που μπορούμε να εφαρμόσουμε στο Arduino είναι από 5 (ή 3.3 για κάποια boards) έως 12V χωρίς να αντιμετωπίζουμε κίνδυνο να καεί η πλακέτα.



Πάντα η τάση λειτουργίας είναι 3.3V ή 5V.

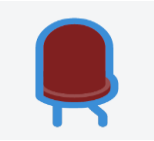
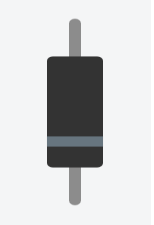
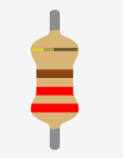
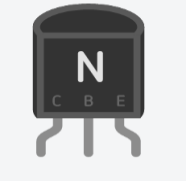

Παρακάτω παρατίθενται οι γνωστότερες πλακέτες Arduino με τους επεξεργαστές και τα χαρακτηριστικά τους


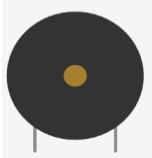
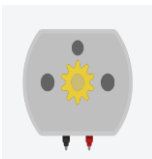


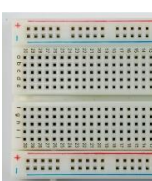
| Name | Processor | Operating/ Input Voltage | CPU Speed | Analog In/Out | Digital IO/PWM | EEPROM [kB] | SRAM [kB] | Flash [kB] | USB | UART |
|-------------------------------|--|--------------------------------------|------------------|---|-------------------|---|------------------------|---|---------|------|
| 101 | Intel® Curie | 3.3 V / 7-12V | 32MHz | 6/0 | 14/4 | - | 24 | 196 | Regular | - |
| Gemma | ATtiny85 | 3.3 V / 4-16 V | 8 MHz | 1/0 | 3/2 | 0.5 | 0.5 | 8 | Micro | 0 |
| LilyPad | ATmega168V ATmega328P | 2.7-5.5 V / 2.7-5.5 V | 8MHz | 6/0 | 14/6 | 0.512 | 1 | 16 | - | - |
| LilyPad SimpleSnap | ATmega328P | 2.7-5.5 V / 2.7-5.5 V | 8 MHz | 4/0 | 9/4 | 1 | 2 | 32 | - | - |
| LilyPad USB | ATmega32U4 | 3.3 V / 3.8-5 V | 8 MHz | 4/0 | 9/4 | 1 | 2.5 | 32 | Micro | - |
| Mega 2560 | ATmega2560 | 5 V / 7-12 V | 16 MHz | 16/0 | 54/15 | 4 | 8 | 256 | Regular | 4 |
| Micro | ATmega32U4 | 5 V / 7-12 V | 16 MHz | 12/0 | 20/7 | 1 | 2.5 | 32 | Micro | 1 |
| MKR1000 | SAMD21 Cortex-M0+ | 3.3 V / 5V | 48MHz | 7/1 | 8/4 | - | 32 | 256 | Micro | 1 |
| Pro | ATmega168 ATmega328P | 3.3 V / 3.35-12 V 5 V / 5-12 V | 8 MHz 16 MHz | 6/0 | 14/6 | 0.512 1 | 1 2 | 16 32 | - | 1 |
| Pro Mini | ATmega328P | 3.3 V / 3.35-12 V 5 V / 5-12 V | 8 MHz 16 MHz | 6/0 | 14/6 | 1 | 2 | 32 | - | 1 |
| Uno | ATmega328P | 5 V / 7-12 V | 16 MHz | 6/0 | 14/6 | 1 | 2 | 32 | Regular | 1 |
| Zero | ATSAMD21G18 | 3.3 V / 7-12 V | 48 MHz | 6/1 | 14/10 | - | 32 | 256 | 2 Micro | 2 |
| Due | ATSAM3X8E | 3.3 V / 7-12 V | 84 MHz | 12/2 | 54/12 | - | 96 | 512 | 2 Micro | 4 |
| Esplora | ATmega32U4 | 5 V / 7-12 V | 16 MHz | - | - | 1 | 2.5 | 32 | Micro | - |
| Ethernet | ATmega328P | 5 V / 7-12 V | 16 MHz | 6/0 | 14/4 | 1 | 2 | 32 | Regular | - |
| Leonardo | ATmega32U4 | 5 V / 7-12 V | 16 MHz | 12/0 | 20/7 | 1 | 2.5 | 32 | Micro | 1 |
| Mega ADK | ATmega2560 | 5 V / 7-12 V | 16 MHz | 16/0 | 54/15 | 4 | 8 | 256 | Regular | 4 |
| Mini | ATmega328P | 5 V / 7-9 V | 16 MHz | 8/0 | 14/6 | 1 | 2 | 32 | - | - |
| Nano | ATmega168 ATmega328P | 5 V / 7-9 V | 16 MHz | 8/0 | 14/6 | 0.512 1 | 1 2 | 16 32 | Mini | 1 |
| Yún | ATmega32U4 AR9331 Linux | 5 V | 16 MHz 400MHz | 12/0 | 20/7 | 1 | 2.5 16MB | 32 64MB | Micro | 1 |
| Arduino Robot | ATmega32u4 | 5 V | 16 MHz | 6/0 | 20/6 | 1 KB (ATmega32u4)/ 512 Kbit (I2C) | 2.5 KB (ATmega32u4) | 32 KB (ATmega32u4) of which 4 KB used by bootloader | 1 | 1 |
| MKRZero | SAMD21 Cortex-M0+ 32bit low power ARM MCU | 3.3 V | 48 MHz | 7 (ADC 8/10/12 bit)/1 (DAC 10 bit) | 22/12 | No | 32 KB | 256 KB | 1 | 1 |

Πίνακας 1-1 Διάφορα είδη Arduino Boards

1.4 Ηλεκτρονικά εξαρτήματα που συνδέονται στο Arduino

Για να χρησιμοποιηθούν τα ηλεκτρονικά εξαρτήματα με το Arduino δε χρειάζεται προηγούμενη εμπειρία. Παρακάτω παρατίθενται κάποια από τα βασικά εξαρτήματα ώστε να μπορούν να αναγνωριστούν και να κατανοηθεί η χρήση τους.

| | | |
|---|--|---|
| <p>LED</p>  | <p>Τι κάνει Εκπέμπει φως όταν μικρό ρεύμα περάσει μέσα του (μόνο στη μια κατεύθυνση)</p> <p>Πως το αναγνωρίζουμε Μοιάζει με μικρή λάμπα.</p> | <p>Αριθμός επαφών 2 (ένα μακρύτερο που συνδέεται με το θετικό πόλο)</p> <p>Τι να προσέξουμε -Λειτουργεί μόνο στη μια κατεύθυνση -Απαιτεί αντιστάτη</p> |
| <p>ΔΙΟΔΟΣ (DIODE)</p>  | <p>Τι κάνει Είναι το αντίστοιχο ηλεκτρονικό στοιχείο μιας μονόδρομης βαλβίδας. Επιτρέπει το ρεύμα να περνάει προς μια κατεύθυνση αλλά όχι και αντίστροφα.</p> <p>Πως το αναγνωρίζουμε Συνήθως είναι κύλινδρος με επαφές που προεξέχουν από τα δύο άκρα του</p> | <p>Αριθμός επαφών 2</p> <p>Τι να προσέξουμε Δουλεύει μόνο προς τη μια κατεύθυνση</p> |
| <p>ΑΝΤΙΣΤΑΤΗΣ (RESISTOR)</p>  | <p>Τι κάνει Περιορίζει την ποσότητα ρεύματος που περνά μέσα από ένα κύκλωμα.</p> <p>Πως το αναγνωρίζουμε Είναι κύλινδρος με εξογκωμένα άκρα και επαφές στα δύο άκρα του.</p> | <p>Αριθμός επαφών 2</p> <p>Τι να προσέξουμε Να επιλέγουμε το σωστό αντιστάτη ανάλογα το κύκλωμα.</p> |
| <p>ΤΡΑΝΖΙΣΤΟΡ (TRANSISTOR)</p>  | <p>Τι κάνει Χρησιμοποιεί ένα μικρό ρεύμα για να ενεργοποιήσει ή ενισχύσει ένα μεγαλύτερο ρεύμα.</p> <p>Πως το αναγνωρίζουμε Είναι συνήθως μισός μαύρος κύλινδρος με 3 ποδαράκια-επαφές</p> | <p>Αριθμός επαφών 3(βάση, συλλέκτης, εκπομπή)</p> <p>Τι να προσέξουμε Πρέπει να συνδέονται τα ποδαράκια-επαφές με το σωστό τρόπο. Συνήθως χρειάζεται ένας αντιστάτης στην επαφή της Βάσης</p> |
| <p>ΠΟΤΕΝΣΙΟΜΕΤΡΟ (POTENTIOMETER)</p>  | <p>Τι κάνει Παρέχει διαφορετική αντίσταση ανάλογα με τη θέση της γωνίας του δείκτη.</p> <p>Πως το αναγνωρίζουμε Έχει μία βίδα ή επιφάνεια που περιστρέφεται και έχει ένα βελάκι επάνω του.</p> | <p>Αριθμός επαφών 3</p> <p>Τι να προσέξουμε Να μην αγοραστεί κατά λάθος ποτενσιόμετρο λογαριθμικής κλίμακας.</p> |
| <p>ΚΟΥΜΠΙ (PUSH BUTTON)</p> | <p>Τι κάνει Ενεργοποιεί το κύκλωμα όταν πατηθεί.</p> <p>Πως το αναγνωρίζουμε Μικρό τετράγωνο με οδηγούς στο κάτω μέρος κι ένα κουμπί στην κορυφή.</p> | <p>Αριθμός επαφών 4</p> <p>Τι να προσέξουμε Είναι τετράγωνα οπότε μπορούν να εισαχθούν με 90 μοίρες γωνία.</p> |

| | | |
|--|--|--|
|  | | |
| ΠΙΕΖΟ ΣΤΟΙΧΕΙΟ (PIEZO ELEMENT)  | Τι κάνει Ένας παλμός ρεύματος το ενεργοποιεί. Η ροή των παλμών θα το αναγκάσει να εκπέμψει έναν ήχο. Πως το αναγνωρίζουμε Συνήθως είναι ένα μαύρο βαρελάκι κι άλλες φορές απλά ένας χρυσός δίσκος. | Αριθμός επαφών 2 Τι να προσέξουμε Είναι αδύνατο να μη χρησιμοποιηθεί σωστά. |
| DC ΜΟΤΕΡ  | Τι κάνει Περιστρέφεται όταν περνά ρεύμα. Πως το αναγνωρίζουμε Μοιάζει με μοτέρ | Αριθμός επαφών 2 Τι να προσέξουμε Πρέπει να χρησιμοποιείται τρανζίστορ το οποίο να ταιριάζει στο μοτέρ. |
| ΦΩΤΟΑΝΤΙΣΤΑΤΗΣ  | Τι κάνει Παράγει μια μεταβλητή αντίσταση η οποία εξαρτάται από την ποσότητα του φωτός που δέχεται. Πως το αναγνωρίζουμε Συνήθως είναι ένας μικρός δίσκος με διάφανη κορυφή και μια καμπυλωτή γραμμή από κάτω. | Αριθμός επαφών 2 Τι να προσέξουμε Πρέπει να είναι περσέει από έναν διαιρητή τάσης πριν δώσει αποτέλεσμα. |
| Καλώδια σύνδεσης  | Τι κάνει Καλώδια σύνδεσης Πως το αναγνωρίζουμε Χρωματιστά καλώδια με μαύρα άκρα | Αριθμός επαφών 2 Τι να προσέξουμε Υπάρχουν 3 ειδών αρσενικά στα 2 άκρα (M-M), αρσενικό-θηλυκό (M-F), και θηλυκά και στα 2 άκρα (F-F) |
| Βοηθητική πλακέτα  | Τι κάνει Βοηθητική πλακέτα (breadboard) Πως το αναγνωρίζουμε Ορθογώνια πλακέτα με τρύπες ομαδοποιημένα σε 4 σειρές και πολλές στήλες | Αριθμός επαφών Ανάλογα με το μέγεθος Τι να προσέξουμε Οι 2 πάνω και κάτω σειρές (με σύμβολα +, -) είναι συνδεδεμένα εσωτερικά σε γραμμές. Οι εσωτερικές 2 ομάδες είναι συνδεδεμένες κατά στήλες |

Πίνακας 1-2 Βασικά εξαρτήματα για Arduino

Επιπλέον υπάρχουν και πολλά ειδικά εξαρτήματα και αισθητήρες όπως ενδεικτικά τα παρακάτω:

| | | |
|--|---|--|
| <p>Αισθητήρας Θερμοκρασίας Υγρασίας</p>  | <p>Τι κάνει Μετρά Θερμοκρασία και Υγρασία περιβάλλοντος</p> <p>Πως το αναγνωρίζουμε Έχει ένα λευκό διάτρητο κουτί</p> | <p>Αριθμός επαφών 3</p> <p>Τι να προσέξουμε Υπάρχει και με μικρότερη ακρίβεια και είναι μπλε</p> |
| <p>Οθόνη</p>  | <p>Τι κάνει Δείχνει στην οθόνη δύο γραμμών χαρακτήρες και αριθμούς για να βλέπουμε μετρήσεις</p> <p>Πως το αναγνωρίζουμε Έχει μία οθόνη ορθογώνιου παραλληλόγραμμου πάνω σε πλακέτα</p> | <p>Αριθμός επαφών 16</p> <p>Τι να προσέξουμε Υπάρχουν με οπίσθιο φωτισμό και χωρίς. Επίσης υπάρχουν TFT οθόνες (επαφής ή όχι) διαφόρων μεγεθών που μπορούν να έχουν γραφικά και να απεικονίζουν περισσότερους χαρακτήρες</p> |
| <p>WiFi επικοινωνία</p>  | <p>Τι κάνει Δημιουργεί ασύρματη επικοινωνία wifi με access point και συνδέεται στο διαδίκτυο</p> <p>Πως το αναγνωρίζουμε Έχει στο ένα άκρο μια χρυσή τεθλασμένη γραμμή (ζικ-ζακ) που είναι η κεραία του</p> | <p>Αριθμός επαφών 24</p> <p>Τι να προσέξουμε Υπάρχουν πολλές εκδόσεις με διαφορετικά χαρακτηριστικά. Απαιτούνται ιδιαίτερες γνώσεις προγραμματισμού και δικτύων.</p> |
| <p>Αισθητήρας μονοξειδίου του άνθρακα</p>  <p>MQ7</p> | <p>Τι κάνει Αισθητήρας για ανίχνευση μονοξειδίου του άνθρακα</p> <p>Πως το αναγνωρίζουμε Γράφει MQ-7</p> | <p>Αριθμός επαφών 6</p> <p>Τι να προσέξουμε Χρειάζεται να συνδεθούν Pins μεταξύ τους</p> |
| <p>Αισθητήρας καπνού</p>  <p>MQ2</p> | <p>Τι κάνει Αισθητήρας για εύφλεκτα αέρια (Μεθάνιο, Βουτάνιο, LPG, καπνό)</p> <p>Πως το αναγνωρίζουμε Γράφει MQ-2</p> | <p>Αριθμός επαφών 4</p> <p>Τι να προσέξουμε Κάποιες υλοποιήσεις έχουν μεταβλητή αντίσταση σε ποτενσιόμετρο για να ρυθμίζεται η ευαισθησία του</p> |

Πίνακας 1-3 Μερικά ειδικά ηλεκτρονικά εξαρτήματα

2. Περιβάλλον Προγραμματισμού

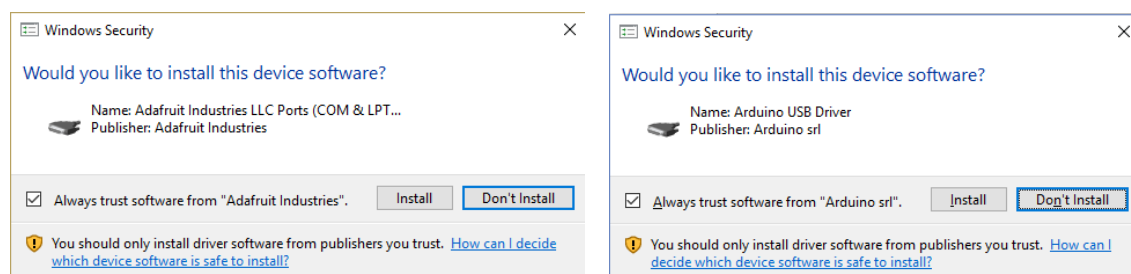
Το περιβάλλον προγραμματισμού που συνήθως χρησιμοποιείται για τα Arduino διατίθεται δωρεάν από τον ιστότοπο <https://www.arduino.cc/en/Main/Software> και είναι κατάλληλο για λειτουργικά συστήματα Windows, Mac OS X και Linux.

Η εγκατάσταση για περιβάλλον λειτουργικού συστήματος Windows γίνεται με τα ακόλουθα βήματα:

Βήμα 1. «Κατεβάζουμε» το λογισμικό για το λειτουργικό μας σύστημα από το <https://www.arduino.cc/en/Main/Software> από το σύνδεσμο Windows Installer

Βήμα 2. Εκτελούμε το αρχείο Arduino-rr-windows.exe που μόλις κατεβάσαμε (rr) είναι η τρέχουσα έκδοση π.χ. 1.8.1)

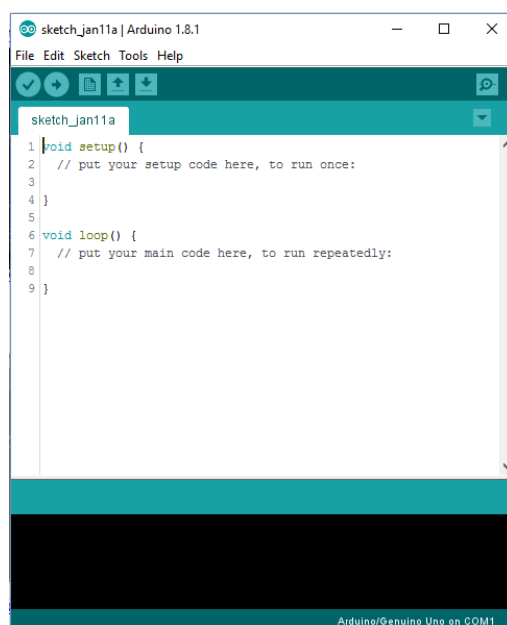
Βήμα 3. Επιβεβαιώνουμε πως θα εγκαταστήσουμε τους οδηγούς για τις σειριακές θύρες και τις USB θύρες.



Βήμα 4. Εκτελούμε την συντόμευση που έχει δημιουργηθεί στην επιφάνεια εργασίας

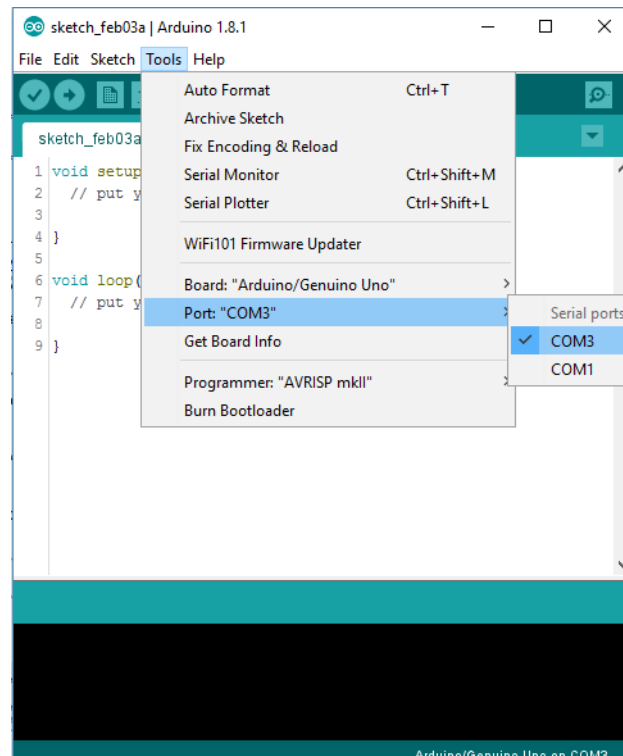


Βήμα 5. Συνδέουμε την πλακέτα Arduino μέσω του USB καλωδίου με τον υπολογιστή και εμφανίζεται το παράθυρο προσθήκης νέου υλικού και αναζητούμε τους κατάλληλους οδηγούς στο path C:\Program Files (x86)\Arduino\drivers



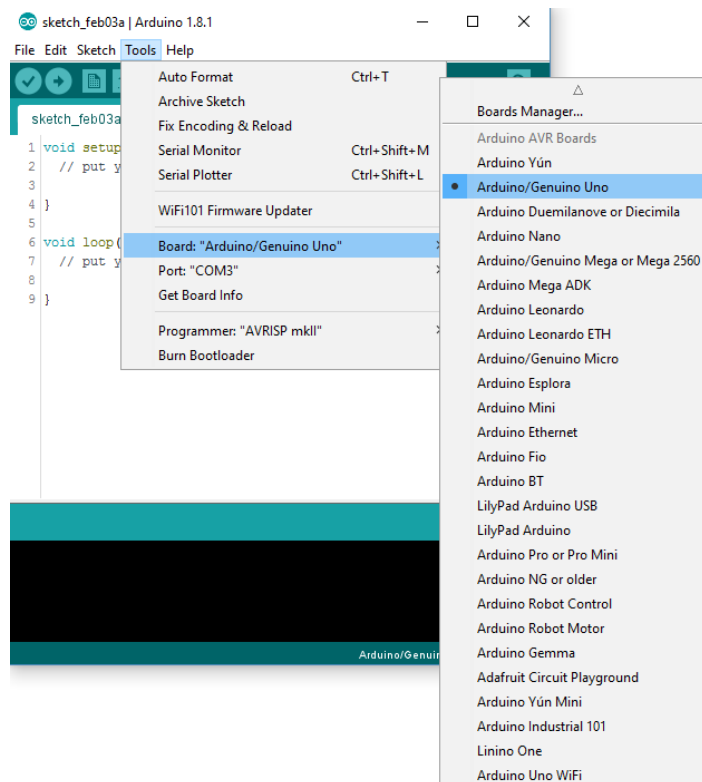
Εικόνα 3 Προγραμματιστικό περιβάλλον Arduino IDE

Βήμα 6. Μετά τη σύνδεση του Arduino board στην USB του υπολογιστή και την εκκίνηση του Arduino IDE επιλέγουμε στο Tools-Port την νέα σειριακή θύρα που έχει εμφανιστεί π.χ. COM3



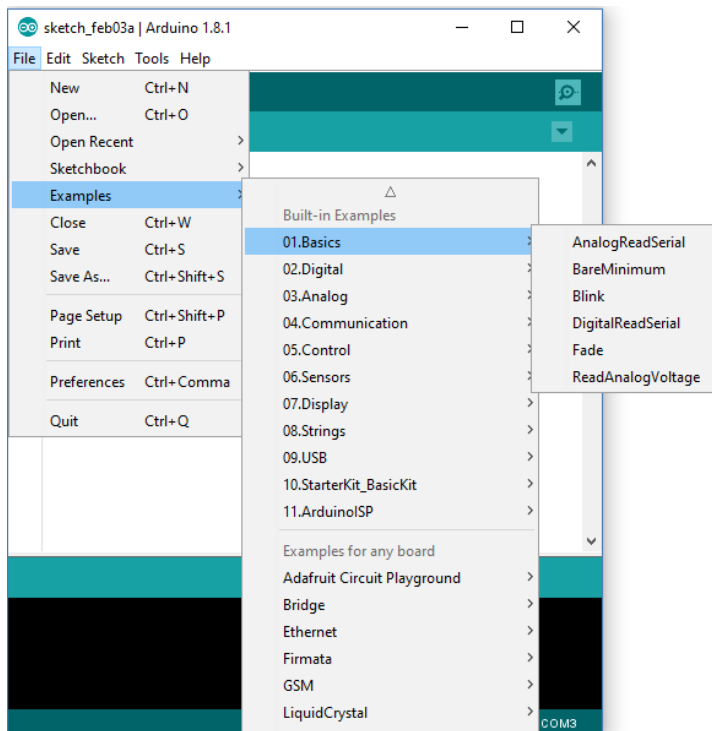
Εικόνα 4 Επιλογή σειριακής θύρας

Βήμα 7. Επιλέγουμε τον τύπο του Arduino board που έχουμε στη διάθεσή μας από το Tools-Board





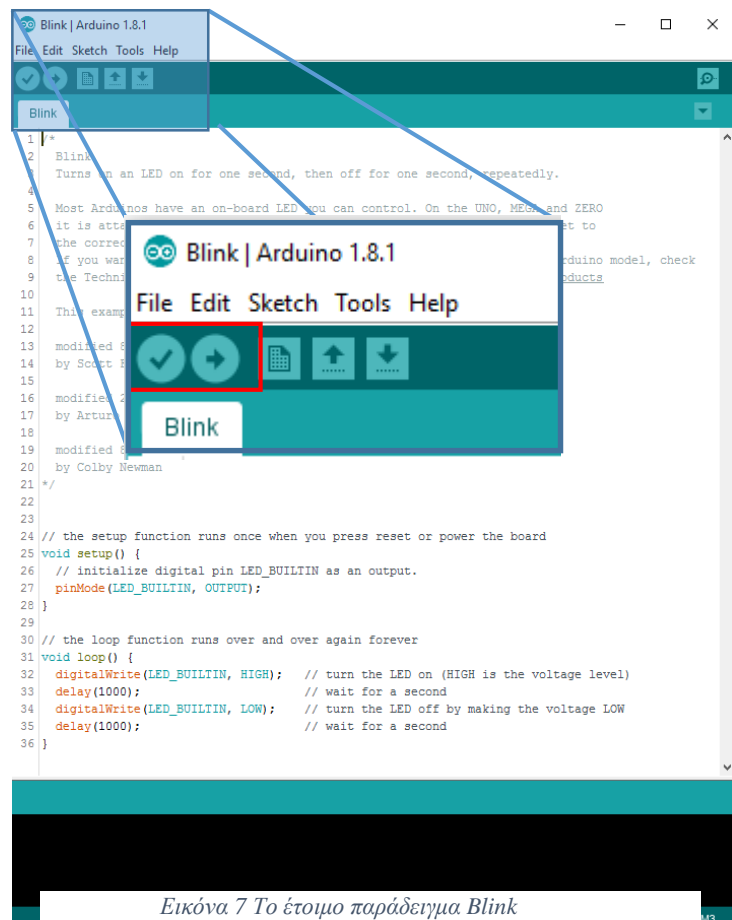
Εικόνα 5 Επιλογή του διαθέσιμου Arduino board

Στο περιβάλλον προγραμματισμού υπάρχουν αρκετά έτοιμα παραδείγματα με σχόλια στον κώδικα που μπορούμε να τα επιλέξουμε από το File-Examples. Για παράδειγμα στο Examples-01.Basics μπορούμε να επιλέξουμε το παράδειγμα Blink.



Εικόνα 6 Επιλογή έτοιμων παραδειγμάτων

Το παράδειγμα στην Εικόνα 7 έχει επιλεγεί από την ομάδα των Βασικών παραδειγμάτων. Περιέχει έτοιμο τον κώδικα με σχόλια. Για να τρέξει το πρόγραμμα πρώτα πατάμε το κουμπί  για να γίνει ο έλεγχος σφαλμάτων και να «μεταγλωττιστεί» όπως λέμε το πρόγραμμα C σε γλώσσα που καταλαβαίνει ο επεξεργαστής του Arduino και μετά το κουμπί  για να «φορτωθεί» το πρόγραμμα στη μνήμη του Arduino και να αρχίσει να εκτελείται. Όταν γίνει αυτό βλέπουμε το LED να αναβοσβήνει.

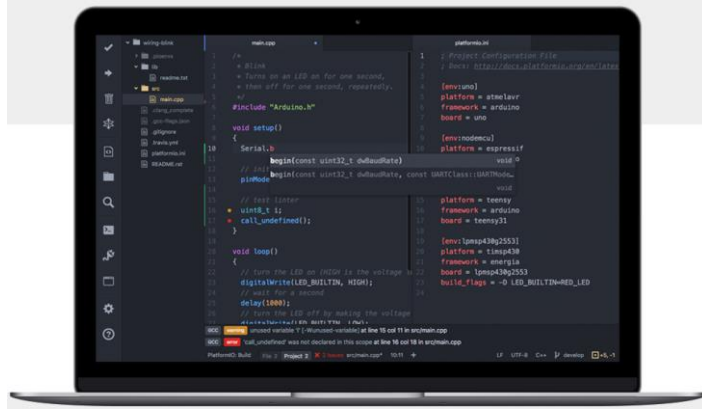


Εικόνα 7 Το έτοιμο παράδειγμα Blink

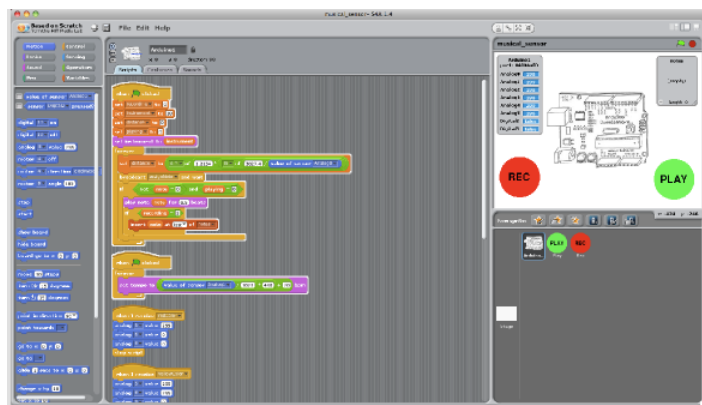
2.1 Άλλα περιβάλλοντα προγραμματισμού

Λόγω της ευκολίας δημιουργίας πληθώρας πρωτότυπων projects έχει δημιουργηθεί μεγάλη κοινότητα για τα Arduino και φυσικά έχουν δημιουργηθεί πολλά περιβάλλοντα προγραμματισμού με χρήση C/C++/C# για όσους γνωρίζουν τα βασικά του προγραμματισμού ή με προγραμματισμό με χρήση δομικών στοιχείων βασισμένα σε Scratch (programming building blocks) για όσους δεν έχουν καθόλου εμπειρία, αλλά και προσομοιωτών (simulators) για δοκιμές πριν την υλοποίηση, όπως:

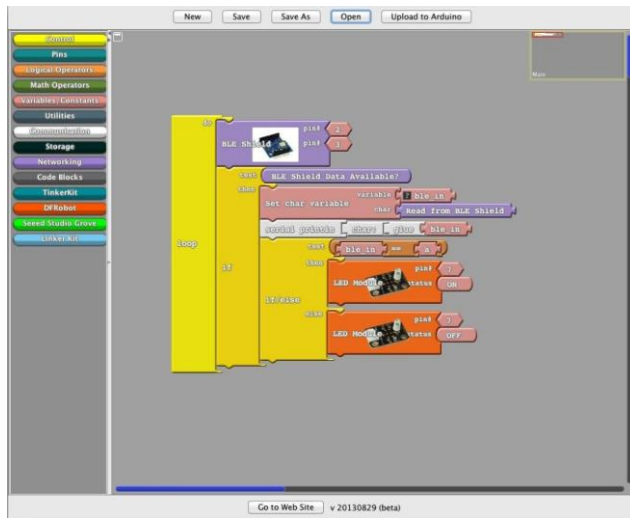
PlatformIO IDE <http://platformio.org/platformio-ide>



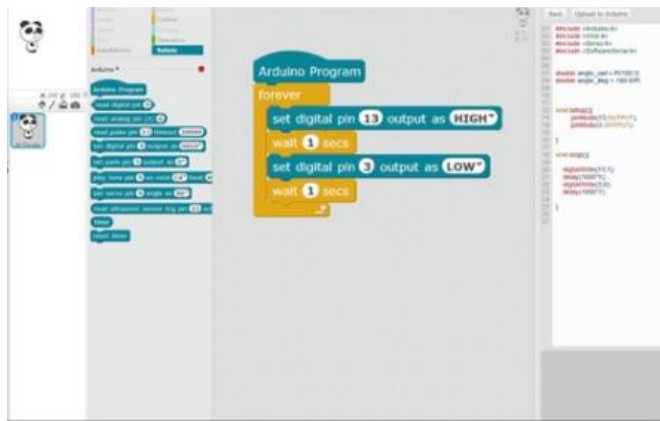
S4A <http://s4a.cat/>



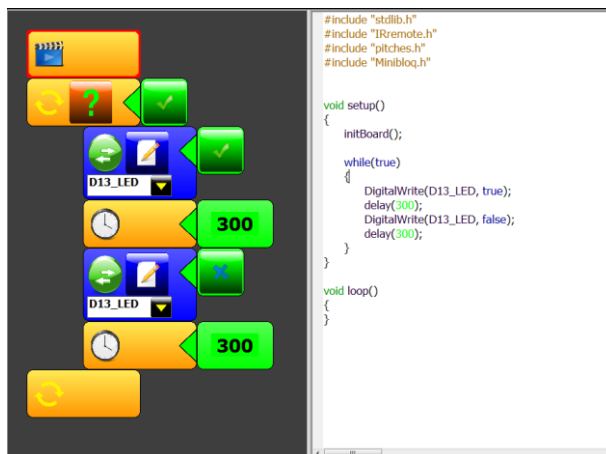
Ardublock <https://sourceforge.net/projects/ardublock/?source=navbar>



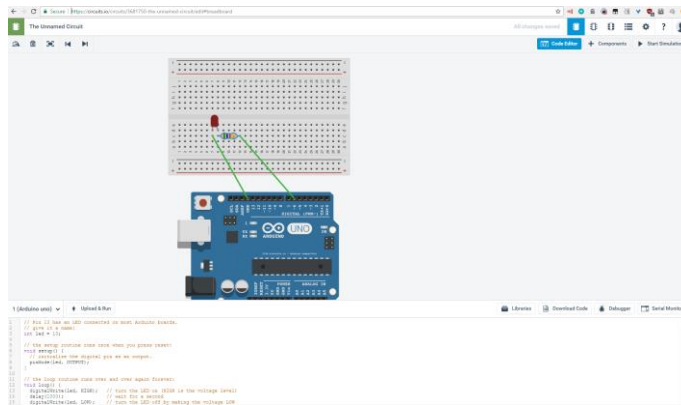
mBlock <http://www.mblock.cc/>



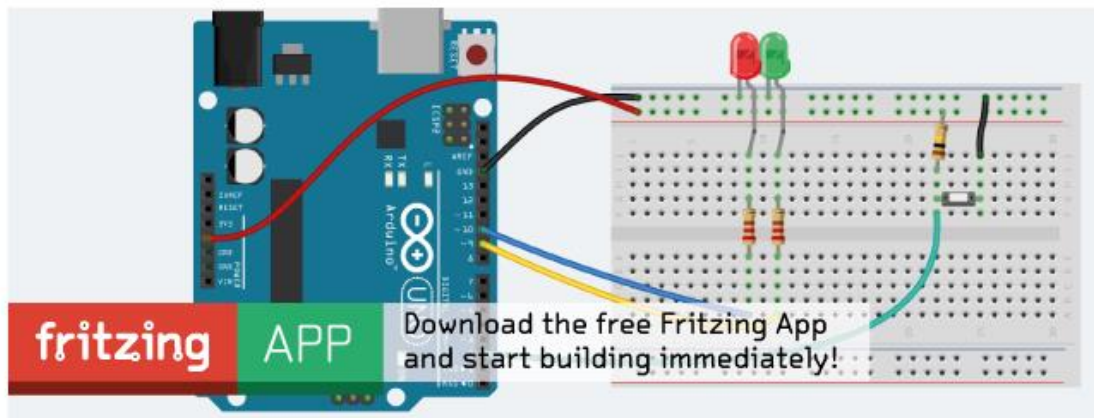
Minibloq <http://blog.minibloq.org/p/download.html>



Autodesk Circuits <https://circuits.io>



Ταυτόχρονα υπάρχουν και σχεδιαστικά εργαλεία που βοηθούν στη δημιουργία σχηματικών, όπως το fritzing (<http://fritzing.org/home/>)



Στη συνέχεια και τις παρούσες ανάγκες γίνεται χρήση του Arduino IDE και του Autodesk Circuits.

3. Εντολές Προγραμματισμού

Το Arduino IDE σαν προγραμματιστικό περιβάλλον χρησιμοποιεί την γλώσσα C.

Οι βασικές βιβλιοθήκες που υπάρχουν στο περιβάλλον αυτό έχει τις βασικές εντολές που αφορούν την πλακέτα του Arduino.

Παρακάτω περιγράφονται οι βασικές εντολές για τις πλακέτες Arduino.

3.1 Σταθερές

Οι σταθερές που έχουν οριστεί στο περιβάλλον προγραμματισμού είναι:

HIGH | LOW

Όταν διαβάζουμε ή γράφουμε σε ένα ψηφιακό pin υπάρχουν μόνο δύο πιθανές τιμές που μπορεί να πάρει. Αυτές είναι HIGH (με τιμή 1) ή LOW (με τιμή 0).

INPUT | OUTPUT

Τα pins μπορούν να ρυθμιστούν σαν είσοδος (INPUT) η έξοδος (OUTPUT), δηλαδή μπορούν να δέχονται ρεύμα ή να στέλνουν ρεύμα ως σήμα.

Εάν το pin έχει ρυθμιστεί ως είσοδος (INPUT) τότε έχουν μικρές απαιτήσεις από το κύκλωμα στο οποίο βρίσκονται ισοδύναμες με σειριακή αντίσταση 100 Mega Ohms μπροστά από το pin. Αυτό τα καθιστά πολύ χρήσιμα.

Εάν έχει ρυθμιστεί ως έξοδος (OUTPUT) τότε λέμε ότι είναι σε μια κατάσταση χαμηλής αντίστασης. Αυτό σημαίνει ότι ένα σημαντικό ποσό ρεύματος μπορεί να προσφερθεί σε άλλα κυκλώματα.

LED_BUILTIN

Τα περισσότερα Arduino έχουν ένα pin συνδεδεμένο με ένα, ενσωματωμένο σε αυτά, LED σε σειρά με έναν αντιστάτη. Η σταθερά LED_BUILTIN είναι το νούμερο του pin με το οποίο το ενσωματωμένο LED είναι συνδεδεμένο. Τα περισσότερα έχουν το LED συνδεδεμένο με το ψηφιακό pin 13.

3.2 Ψηφιακές Λειτουργίες (Digital Functions)

Όπως είπαμε τα διαθέσιμα pins στο Arduino είναι ψηφιακά και αναλογικά. Ανάλογα με το είδος τους υπάρχουν και οι κατάλληλες εντολές/συναρτήσεις.

```
pinMode()
```

Τα pins στο Arduino μπορούν να είναι είτε εισόδου ή εξόδου μόνο, ποτέ και τα δύο μαζί.

Με την εντολή αυτή ορίζεται αν ένα συγκεκριμένο pin θα είναι είτε ως input ή ως output.

```
pinMode (pin, mode);
```

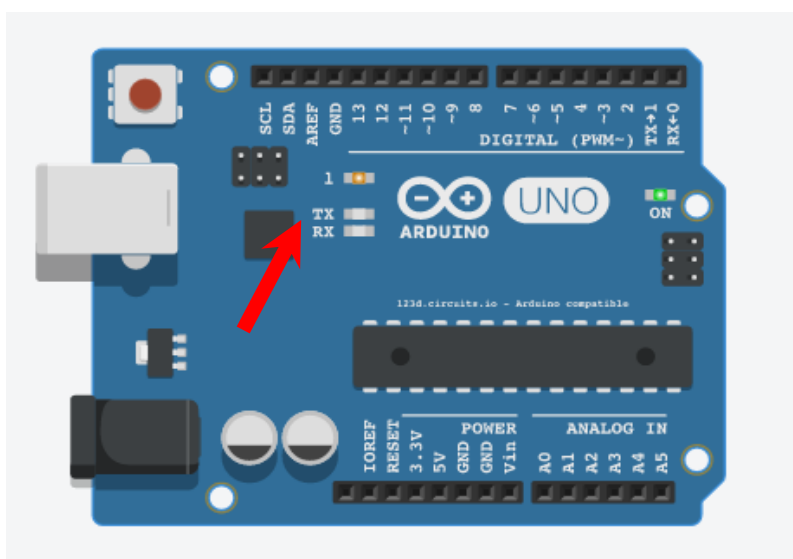
Παράδειγμα

```
int ledPin = 13; // το LED συνδέθηκε με το ψηφιακό pin 13

void setup() {
  pinMode(ledPin, OUTPUT); //θέτει το ψηφιακό pin ως έξοδο
}

void loop() {
  digitalWrite(ledPin, HIGH); // ενεργοποιεί το LED
  delay(1000);                // περιμένει για 1 δευτερόλεπτο
  digitalWrite(ledPin, LOW);  // απενεργοποιεί το LED
  delay(1000);                // περιμένει για 1 δευτερόλεπτο
}
```

Το παράδειγμα αυτό, όταν το τρέξουμε σε προσομοιωτή έχει το παρακάτω αποτέλεσμα.



Το ενσωματωμένο led στην πλακέτα του Arduino που αντιστοιχεί στο pin 13 ανάβει για ένα δευτερόλεπτο κι σβήνει για 1 δευτερόλεπτο συνεχώς.

`digitalWrite()`

Γράφει μια HIGH ή LOW τιμή σε ένα ψηφιακό pin.

`digitalWrite(pin, value);`

όπου pin είναι ο αριθμός του pin που έχει οριστεί ως OUTPUT και value παίρνει τιμή HIGH ή LOW

Παράδειγμα

```
int ledPin = 13; // το LED συνδέθηκε με το ψηφιακό pin 13

void setup() {
  pinMode(ledPin, OUTPUT); //θέτει το ψηφιακό pin ως έξοδο
}

void loop() {
  digitalWrite(ledPin, HIGH); // ενεργοποιεί το LED
  delay(1000);                // περιμένει για 1 δευτερόλεπτο
  digitalWrite(ledPin, LOW);  // απενεργοποιεί το LED
  delay(1000);                //περιμένει για 1 δευτερόλεπτο
}
```

`digitalRead()`

Διαβάζει την τιμή ενός συγκεκριμένου ψηφιακού pin που έχει ήδη οριστεί ως INPUT. Αυτή μπορεί να είναι είτε HIGH ή LOW.

`digitalRead(pin);`

Παράδειγμα

Σε αυτό το παράδειγμα θέτουμε το pin 13 να έχει την ίδια τιμή όπως το pin 7, δηλωμένο ως είσοδος.

```
int ledPin = 13; // Το LED συνδέθηκε στο ψηφιακό pin 13
int inPin = 7;   // Το μπουτόν συνδέθηκε στο ψηφιακό pin 7
int val = 0;     // μεταβλητή που αποθηκεύει την πραγματική τιμή

void setup() {
  pinMode(ledPin, OUTPUT); //θέτει το ψηφιακό pin ως έξοδο
  pinMode(inPin, INPUT);  // θέτει το ψηφιακό pin ως είσοδο
}

void loop() {
  val = digitalRead(inPin); // Διαβάζει το pin εισόδου
  digitalWrite(ledPin, val); //θέτει το LED στην τιμή του μπουτόν
}
```

3.3 Αναλογικές Λειτουργίες (Analog)

Οι αναλογικές θύρες έχουν κι αυτές τις δικές τους εντολές για το χειρισμό τους.

analogRead()

Διαβάζει την τιμή ενός συγκεκριμένου αναλογικού pin. Στην ουσία διαβάζει την τάση ρεύματος από 0 ως 5V σε τιμές ανάμεσα στους ακεραίους 0 ως 1023, έτσι ώστε η τιμή 1023 να σημαίνει 5V, ενώ η τιμή 818 αντιστοιχεί σε 4V ($4 \times 1023 / 5 = 818$).

```
analogRead(pin);
```

Παράδειγμα

```
int analogPin = 3; // το ποτενσιόμετρο συνδέεται στο αναλογικό pin 3
int val = 0; // μεταβλητή στην οποία αποθηκεύεται η τιμή που διαβάζεται

void setup() {
  Serial.begin(9600); // αρχικοποιείται η επικοινωνία στα 9600 bits
}

void loop() {
  val = analogRead(analogPin); // διαβάζει το pin εισόδου
  Serial.println(val); // τυπώνεται η τιμή της μεταβλητής
}
```

analogWrite()

Εκχωρεί μια αναλογική τιμή σε ένα pin. Μπορεί να χρησιμοποιηθεί για να ανάψει ένα LED με διαφορετική φωτεινότητα κάθε φορά ή να ρυθμιστεί ώστε να κινείται ένας κινητήρας με διάφορες ταχύτητες.

```
analogWrite(pin, value);
```

Παράδειγμα

Σε αυτό το παράδειγμα ορίζεται η έξοδος σε σχέση με την αναλογία του LED προς την τιμή που διαβάζει το ποτενσιόμετρο.

```
int ledPin = 9; //το LED συνδέεται στο ψηφιακό in 9
int analogPin = 3; // το ποτενσιόμετρο συνδέεται στο αναλογικό pin 3
int val = 0; // μεταβλητή στην οποία αποθηκεύεται η τιμή που διαβάζεται

void setup() {
  pinMode(ledPin, OUTPUT); // το pin τίθεται ως έξοδος
}

void loop() {
  val = analogRead(analogPin); // διαβάζει το pin εισόδου
  analogWrite(ledPin, val / 4); // οι τιμές του analogRead είναι 0-1023
                                // οι τιμές του analogWrite είναι 0-255
}
```

3.4 Χρονικές Λειτουργίες

Σε όλους τους μικροεπεξεργαστές και στα συστήματα ελέγχου χρειάζεται να εισάγουμε καθυστερήσεις ώστε ένας αισθητήρας να μετρήσει τιμή ή μετρήσεις χρόνου για να μετράμε διάρκεια στην οποία πρέπει να κάνει κάτι ο επεξεργαστής. Στις περιπτώσεις αυτές χρησιμοποιούνται οι παρακάτω εντολές.

| |
|----------|
| millis() |
|----------|

Επιστρέφει τον αριθμό των miliseconds από τη στιγμή που το Arduino άρχισε να τρέχει το συγκεκριμένο πρόγραμμα. Αυτός ο αριθμός θα υπερχειλίσει (θα γυρίσει στο μηδέν) περίπου μετά από 50 ημέρες.

Παράδειγμα

```
unsigned long time;

void setup() {
  Serial.begin(9600); // θέτει τη σειριακή επικοινωνία στα 9600 bits
}

void loop() {
  Serial.print("Time: ");
  time = millis() // τυπώνει το χρόνο απο την ώρα που ξεκίνησε το πρόγραμμα
  Serial.println(time);
  //περιμένει ένα δευτερόλεπτο για να μη σταλούν τεράστιες ποσότητες δεδομένων
  delay(1000);
}
```


micros()

Επιστρέφει τον αριθμό των microseconds από όταν το Arduino ξεκίνησε να τρέχει το συγκεκριμένο πρόγραμμα. Αυτός ο αριθμός θα υπερχειλίσει (θα γυρίσει στο μηδέν) περίπου μετά από 70 λεπτά.

Παράδειγμα

```
unsigned long time;

void setup() {
  Serial.begin(9600); // θέτει τη σειριακή επικοινωνία στα 9600 bits
}

void loop() {
  Serial.print("Time: ");
  time = micros() // τυπώνει το χρόνο απο την ώρα που ξεκίνησε το πρόγραμμα
  Serial.println(time);
  //περιμένει ένα δευτερόλεπτο για να μη σταλούν τεράστιες ποσότητες δεδομένων
  delay(1000);
}
```



Προφανώς για να μετρήσουμε το χρόνο ανάμεσα σε δύο συμβάντα τότε παίρνουμε μία μέτρηση χρόνου t_a για το γεγονός a από την αρχή εκτέλεσης του προγράμματος και μία μέτρηση t_b για το γεγονός b και αφαιρούμε τις δύο τιμές $t_b - t_a$.

delay()

Κάνει παύση στο πρόγραμμα για όσο χρόνο (σε milliseconds) ορίζεται στην κλήση από το πρόγραμμα.

delay(ms);

Παράδειγμα

```
int ledPin = 13; // το LED συνδέθηκε με το ψηφιακό pin 13

void setup() {
  pinMode(ledPin, OUTPUT); //θέτει το ψηφιακό pin ως έξοδο
}

void loop() {
  digitalWrite(ledPin, HIGH); // ενεργοποιεί το LED
  delay(1000);                // περιμένει για 1 δευτερόλεπτο
  digitalWrite(ledPin, LOW);  // απενεργοποιεί το LED
  delay(1000);                //περιμένει για 1 δευτερόλεπτο
}
```

3.5 Δομή προγράμματος για Arduino

Το κάθε πρόγραμμα Arduino (συνήθως λέγεται sketch) αποτελείται από δύο απαραίτητες συναρτήσεις (functions) οι οποίες είναι η `setup()` και η `loop()`.

setup()

Αυτή η συνάρτηση καλείται όταν ξεκινάει το πρόγραμμα και χρησιμοποιείται για την αρχικοποίηση των μεταβλητών, τη λειτουργία των pins, την αρχικοποίηση της σειριακής θύρας, τη χρήση βιβλιοθηκών κ.α.

Η εντολή `setup()` τρέχει μόνο μια φορά, αφότου γίνει reset ή εκκινηθεί το Arduino.

Παράδειγμα

```
int buttonPin = 3;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Σε αυτό το παράδειγμα, ασχολούμαστε με την εντολή **void setup()**. Αφού δηλωθεί η τιμή του `buttonPin` και το είδος αυτής, στη συνέχεια καλείται η εντολή **setup()** η οποία αρχικοποιεί την σειριακή επικοινωνία στα 9600 bits κι έπειτα θέτει το pin 3 ως εισαγωγή.

loop()

Αφού εκτελεστεί το τμήμα του κώδικα στη **setup()** (η οποία αρχικοποιεί και θέτει τις αρχικές τιμές), η λειτουργία της **loop()** επαναλαμβάνει συνεχώς τον κώδικα που είναι μέσα σε αυτήν επιτρέποντας στο πρόγραμμα να τροποποιείται και να ανταποκρίνεται διαρκώς ελέγχοντας ενεργά το Arduino, μέχρι να βγάλουμε την τροφοδοσία του ή πατήσουμε reset.

Παράδειγμα

```
const int buttonPin = 3;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop() {
  if (digitalRead(buttonPin) == HIGH)
    Serial.write('H');
  else
    Serial.write('L');

  delay(1000);
}
```

Σε αυτό το παράδειγμα αφού αρχικοποιηθεί και δηλωθεί ως σταθερά το pin και η εντολή **void setup()** εκτελέσει τις εντολές που εκτελούσε και στο προηγούμενο παράδειγμα, η επαναληπτική εντολή **void loop()** εάν διαβάσει ότι το pin έχει την τιμή 'HIGH' τότε θα εμφανίσει σειριακά το μήνυμα 'H', διαφορετικά θα εμφανίσει το μήνυμα 'L'. Στη συνέχεια θα γίνει παύση για 1000 milliseconds.

4. Επικοινωνία του χρήστη με το Arduino

Στην γλώσσα προγραμματισμού C η επικοινωνία του προγράμματος με το χρήστη γίνεται κυρίως με την εκτύπωση τιμών μεταβλητών στην οθόνη με την συνάρτηση `printf` και διάβασμα τιμών από το πληκτρολόγιο (με εμφάνιση στην οθόνη) με την συνάρτηση `scanf`.

Στο Arduino όμως δεν υπάρχει οθόνη και πληκτρολόγιο, οπότε ένας βασικός και απλός τρόπος επικοινωνίας με το Arduino είναι μέσω χρήσης LEDs, που ανάβουν όταν ισχύει μία συνθήκη, με την όποια ανάδραση απαιτείται να γίνεται μόνο με διακόπτες και ποτενσιόμετρα.

Αυτό όμως δεν είναι αρκετό αν πρέπει να υπάρχει περισσότερη πληροφόρηση όπως π.χ. όταν χρειάζεται να αποτυπωθεί η θερμοκρασία και υγρασία που μετρά ένας αισθητήρας.

Σε αυτή την περίπτωση που απαιτούνται περισσότερες πληροφορίες γίνεται χρήση της σειριακής θύρας για την επικοινωνία με τον υπολογιστή. Βέβαια υπάρχουν περιπτώσεις που χρησιμοποιούνται διάφορα είδη οθονών όπως υγρών κρυστάλλων για εμφάνιση 2 γραμμών και 16 χαρακτήρων (LCD 2x16), οθόνες TFT διαφόρων μεγεθών με δυνατότητα απεικόνισης γραφικών και δυνατότητα εισαγωγής ανάδρασης με οθόνες επαφής (TFT touch).

Επίσης σε ορισμένες περιπτώσεις χρησιμοποιείται ένα εξάρτημα που δίνει διασύνδεση με το διαδίκτυο στο Arduino είτε ασύρματη (WIFI ESP8266) ή ενσύρματη (Ethernet shield).

Τόσο η σύνδεση οθόνης σε Arduino όσο και η δημιουργία διασύνδεσης με το διαδίκτυο μέσω ασύρματης ή ενσύρματης σύνδεσης απαιτεί αρκετές γνώσεις σε προγραμματισμό, ηλεκτρονικά (χρειάζονται αρκετές συνδέσεις) και γνώσεις πρωτοκόλλου TCP/IP.

Στη συνέχεια θα χρησιμοποιήσουμε για την αμφίδρομη επικοινωνίας χρήστη-προγράμματος στο Arduino την σειριακή θύρα που διασυνδέεται με τον υπολογιστή μέσω της θύρας USB.

4.1 Σειριακά pins

Η σειριακή χρησιμοποιείται για την επικοινωνία της πλακέτας Arduino με τον υπολογιστή που έχει οθόνη και πληκτρολόγιο ή για την επικοινωνία του με άλλες συσκευές. Όλες οι πλακέτες Arduino έχουν τουλάχιστο μία σειριακή θύρα που βρίσκεται στα ψηφιακά Pins 0 (Rx Receive-λήψη) και 1 (Tx Transmit-αποστολή), ενώ άλλα όπως το Arduino Mega η Due έχουν τρεις επιπλέον.



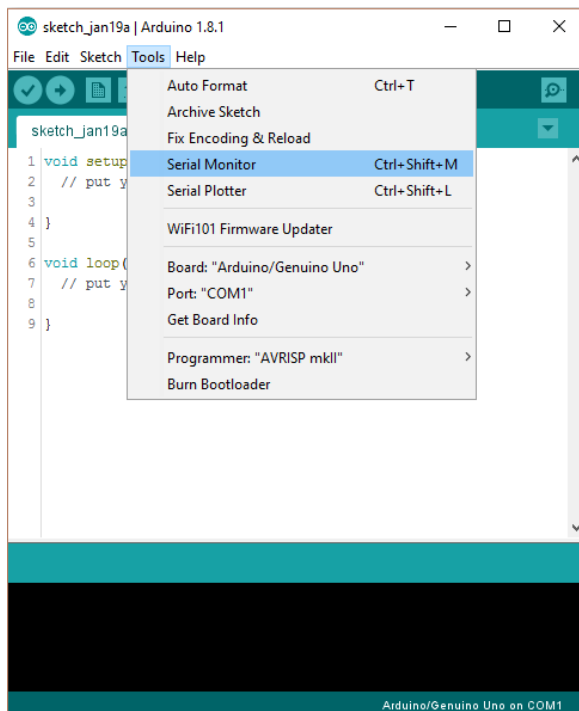
Αν με το πρόγραμμα ενεργοποιήσουμε το σειριακό interface σε πλακέτα Arduino που έχει μόνο ένα σειριακό στα pin 0 και 1, τότε αυτές οι 2 ψηφιακές εισοδοι/έξοδοι δεν μπορούν να χρησιμοποιηθούν για σύνδεση αισθητήρων ή αντίστροφα αν χρησιμοποιούμε τα pin 0 και 1 π.χ. για σύνδεση με αισθητήρες τότε δεν μπορούμε να έχουμε και λειτουργία σειριακής.

Η τάση που χρησιμοποιούν είναι 3.3V ή 5V ανάλογα με την πλακέτα μικροεπεξεργαστή Arduino που χρησιμοποιούμε.



Δεν συνδέονται ποτέ τα Pins αυτά κατευθείαν με την σειριακή θύρα του υπολογιστή γιατί η τάση στον υπολογιστή είναι +/-12V και θα καταστραφεί η πλακέτα του Arduino.

Το προγραμματιστικό περιβάλλον IDE Που χρησιμοποιούμε έχει ενσωματωμένο Serial Monitor.



Εικόνα 8 Serial Monitor on IDE

4.2 Αρχικοποίηση Σειριακής

Η σειριακή θύρα ενεργοποιείται στο τμήμα αρχικοποιήσεων στη συνάρτηση `void setup() { }` με την εντολή `Serial.begin(speed)`, όπου `speed = 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ή 115200 bps`.

Παράδειγμα

```
1 void setup() {
2   // Αρχικοποιεί την σειριακή θύρα με ταχύτητα επικοινωνίας 9600
3   Serial.begin(9600);
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

4.3 Εκτύπωση στην Σειριακή

Για να μπορέσει να εμφανιστεί μία εκτύπωση στην σειριακή θύρα χρησιμοποιείται η εντολή `print` και `println` που είναι η `print` με αλλαγή γραμμής στην εκτύπωση.

Serial.print(val)
ή
Serial.print(val, format)

Όπου:

val η μεταβλητή που θα τυπωθεί και

format αν η μεταβλητή **val** είναι ακέραιος αριθμός αφορά τη βάση του αριθμητικού συστήματος που θα τυπωθεί (δηλαδή DEC δεκαδικό, BIN δυαδικό, HEX δεκαεξαδικό, OCT οκταδικό)

ή αν η μεταβλητή είναι πραγματικός αριθμός ο αριθμός των δεκαδικών ψηφίων

Παράδειγμα

Αν `int val = 78; float fl = 1.23456;`

- `Serial.print("Για ακέραιο:");` τυπώνει **Για ακέραιο:**
- `Serial.print(val, BIN)` τυπώνει **1001110**
- `Serial.print(val, OCT)` τυπώνει **116**
- `Serial.print(val, DEC)` τυπώνει **78**
- `Serial.print(val, HEX)` τυπώνει **4E**
- `Serial.println(fl, 0)` τυπώνει **1** και πάει στην επόμενη γραμμή
- `Serial.println(fl, 2)` τυπώνει **1.23** και πάει στην επόμενη γραμμή
- `Serial.println(fl, 4)` τυπώνει **1.2346** και πάει στην επόμενη γραμμή

Παράδειγμα

```
1 // Διαβάζει το αναλογικό Pin 0 και τυπώνει την τιμή σε διάφορα συστήματα
2
3 int analogValue = 0;
4
5 void setup() {
6   // αρχικοποίηση serial port στα 9600 bps:
7   Serial.begin(9600);
8 }
9
10 void loop() {
11   // διαβάζει το αναλογικό pin 0:
12   analogValue = analogRead(0);
13
14   // το τυπώνει σε διαφορετικά αριθμητικά συστήματα:
15   Serial.println(analogValue); // το τυπώνει σαν ASCII-encoded δεκαδικό
16   Serial.println(analogValue, DEC); // το τυπώνει σαν ASCII-encoded δεκαδικό
17   Serial.println(analogValue, HEX); // το τυπώνει σαν ASCII-encoded δεκαεξαδικό
18   Serial.println(analogValue, OCT); // το τυπώνει σαν ASCII-encoded οκταδικό
19   Serial.println(analogValue, BIN); // το τυπώνει σαν ASCII-encoded διαδυκό
20
21   // καθυστέρηση 10 milliseconds για επόμενο διάβασμα:
22   delay(10);
23 }
```

Έξοδος στο Serial monitor

```
1023
1023
3FF
1777
1111111111
```

4.4 Διάβασμα από τη Σειριακή

Όπως είπαμε δεν υπάρχει εντολή `scanf()` στο Arduino όπως στη γλώσσα C για υπολογιστές. Για να εισάγει λοιπόν ο χρήστης δεδομένα σε ένα πρόγραμμα Arduino μπορεί να διαβάσει δεδομένα από τη σειριακή θύρα με την εντολή `read()`.

Serial.read()

Η συνάρτηση δεν δέχεται παραμέτρους και επιστρέφει ως ακέραιο το πρώτο byte που είναι διαθέσιμο στον buffer της σειριακής θύρας ή -1 αν δεν υπάρχουν δεδομένα.

Για να γνωρίζουμε πόσα bytes δεδομένων υπάρχουν μπορούμε να χρησιμοποιήσουμε τη συνάρτηση `available()`.

Serial.available()

Η συνάρτηση δεν δέχεται παραμέτρους και επιστρέφει τον αριθμό των bytes που είναι διαθέσιμα για διάβασμα στο buffer της σειριακής θύρας. Στο buffer εισόδου αποθηκεύονται τα δεδομένα που έχουν φτάσει με τη σειρά στην είσοδο (Rx-Receive) της σειριακής και είναι διαθέσιμα για διάβασμα.

Παράδειγμα

```
1 int incomingByte = 0; // για το διάβασμα του byte των δεδομένων
2
3 void setup() {
4     Serial.begin(9600); // αρχικοποίηση της σειριακής θύρας στα 9600 bps
5 }
6
7 void loop() {
8
9     // διάβασε δεδομένα όταν υπάρχουν στο Buffer της σειριακής:
10    if (Serial.available() > 0) {
11        // διάβασε το εισερχόμενο byte:
12        incomingByte = Serial.read();
13
14        // εκτύπωσε αυτό που διαβάστηκε:
15        Serial.print("I received: ");
16        Serial.println(incomingByte, DEC);
17    }
18 }
19
```

Αν ως είσοδο δώσουμε **150** τότε ως **έξοδο** έχουμε:

I received: 49

I received: 53

I received: 48

Το 150 που δώσαμε ως είσοδο δεν είναι το νούμερο 150 αλλά οι 3 χαρακτήρες «1», «5» και «0» και αυτό που τυπώνεται στη σειριακή είναι η απεικόνιση των χαρακτήρων σε κωδικό ASCII.



Ότι εισέρχεται ή εξέρχεται στη σειριακή είναι χαρακτήρες και η απεικόνισή τους σε bytes είναι με τον ASCII Κώδικα. Αν θέλουμε σε πρόγραμμα να έχουμε την αντιστοίχιση σε αριθμό τότε αφαιρούμε τον αριθμό 48 που είναι το byte του ASCII κώδικα για το 0.

5. Παραδείγματα με Εκτελέσιμα Προγράμματα σε Arduino

Παράδειγμα 1^ο

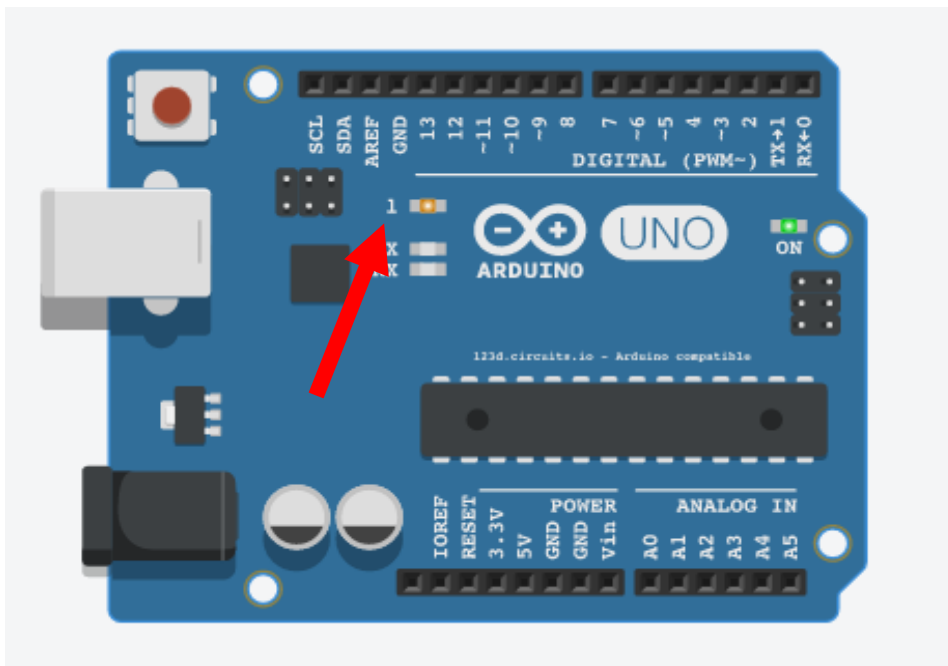
Σε αυτό το παράδειγμα θα δούμε το πρώτο και βασικό πρόγραμμα των Arduino, κάτι αντίστοιχο του hello world στη C, το οποίο ήδη έχει παρουσιαστεί και παραπάνω.

```
int ledPin = 13; // το LED συνδέθηκε με το ψηφιακό pin 13

void setup() {
  pinMode(ledPin, OUTPUT); //θέτει το ψηφιακό pin ως έξοδο
}

void loop() {
  digitalWrite(ledPin, HIGH); // ενεργοποιεί το LED
  delay(1000);                // περιμένει για 1 δευτερόλεπτο
  digitalWrite(ledPin, LOW);  // απενεργοποιεί το LED
  delay(1000);                // περιμένει για 1 δευτερόλεπτο
}
```

Το παράδειγμα αυτό, όταν πραγματοποιηθεί με προσομοιωτή έχει το παρακάτω αποτέλεσμα.

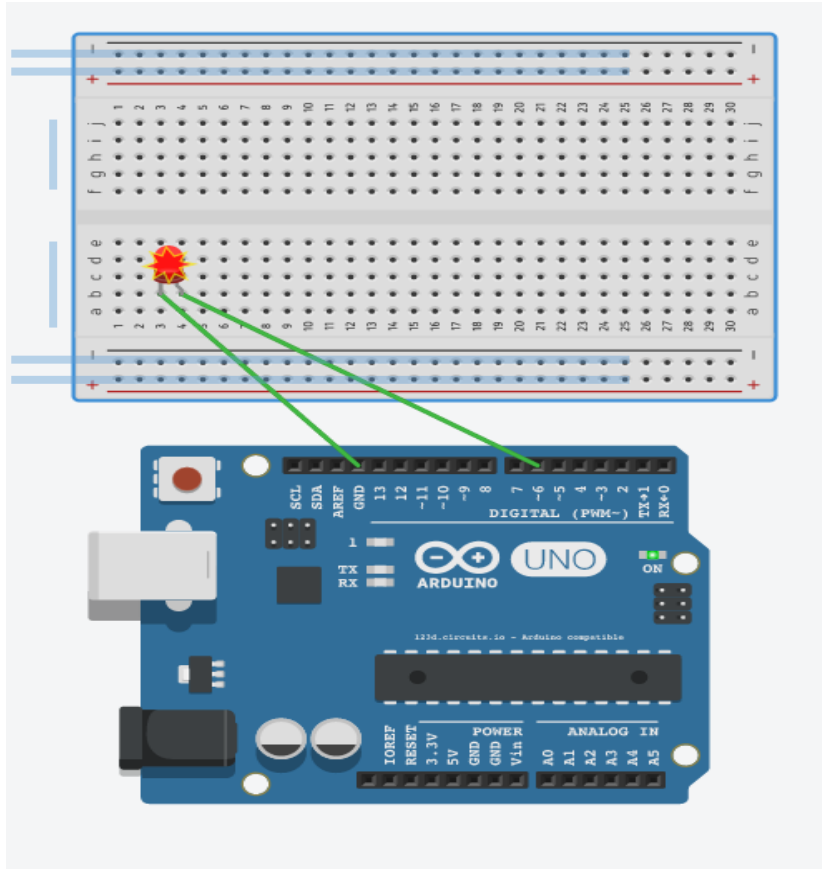


Το λαμπάκι στη θέση 1 ανάβει ένα δευτερόλεπτο και παραμένει σβηστό για 1 δευτερόλεπτο συνεχώς.

Παράδειγμα 2^ο

Σε αυτό το παράδειγμα θα συνδέσουμε ένα LED στο Arduino και θα το ρυθμίσουμε να αναβοσβήνει. Για τις συνδέσεις θα χρησιμοποιήσουμε ένα breadboard όπως παρακάτω. Τα breadboards είναι βοηθητικές πλακέτες που έχουν εσωτερικά συνδεδεμένες οπές, οπότε ότι βάλουμε σε αυτές να είναι σαν να συνδέονται μεταξύ τους. Οι συνδέσεις είναι σε κάθε μία από τις 4 γραμμές με το σύμβολο – για την γείωση, + για την τάση στο πάνω μέρος και αντίστοιχα στο κάτω και η κάθε σειρά στήλης (με τα νούμερα 1, 2, ..., 30) σε δύο μπλοκ πάνω και κάτω. Έτσι ότι ενώνεται στις οπές a, b, c, d, e, στη στήλη 3 είναι συνδεδεμένο και μεταξύ τους, ενώ είναι άλλη σύνδεση τα f, g, h, i, j της στήλης 3 στο πάνω μέρος.

```
1 // σύνδεση στο pin 6 του κόκκινου LED.
2 // ονομάζεται pinLed:
3 int pinLed = 6;
4
5 void setup() {
6 // αρχικοποίηση ψηφιακού pin ως έξοδος
7   pinMode(pinLed, OUTPUT);
8 }
9
10 void loop() {
11   digitalWrite(pinLed, HIGH); // ενεργοποιείται το κόκκινο LED (HIGH)
12   delay(1000);                // αναμονή για 1 δευτερόλεπτο
13   digitalWrite(pinLed, LOW);  // απενεργοποιείται το κόκκινο LED (LOW)
14   delay(1000);                // αναμονή για 1 δευτερόλεπτο
15 }
```

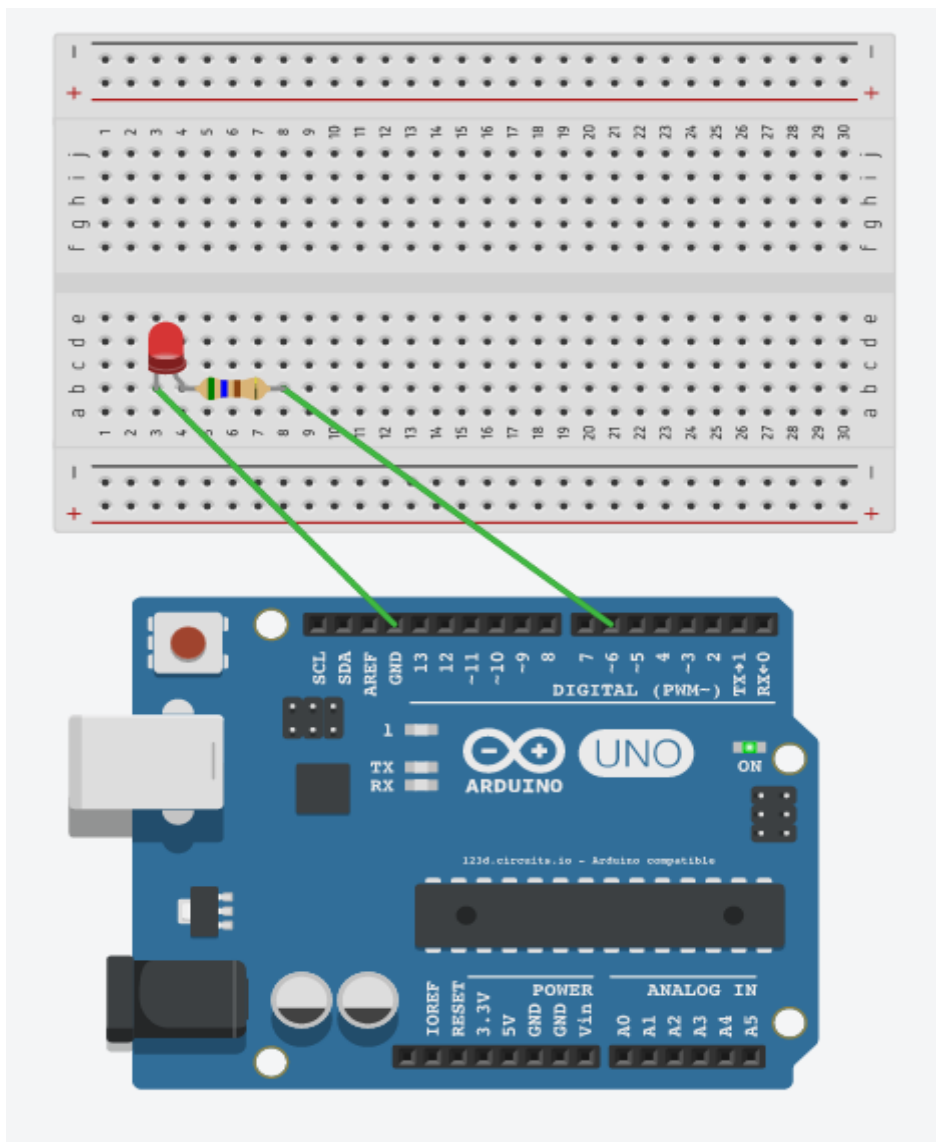


Εικόνα 9 Συνδέσεις breadboard πλακέτας



Προσέχουμε τα άκρα του LED να ακουμπάνε στην ίδια σειρά με το ίδιο γράμμα ειδήλλως θα καεί, ενώ το ίδιο ισχύει για όλα τα εξαρτήματα που θα προστεθούν στη συνέχεια.

Κατά την προσομοίωση παρατηρούμε πως το κόκκινο LED ανάβει πολύ έντονα που σημαίνει ότι είναι επικίνδυνο να καεί. Το ιδανικό λοιπόν, θα ήταν να μειώσουμε το ρεύμα που διαπερνά το LED προσθέτοντάς έναν αντιστάτη περίπου 500Ω ώστε να μειωθεί η τάση στο κόκκινο LED.

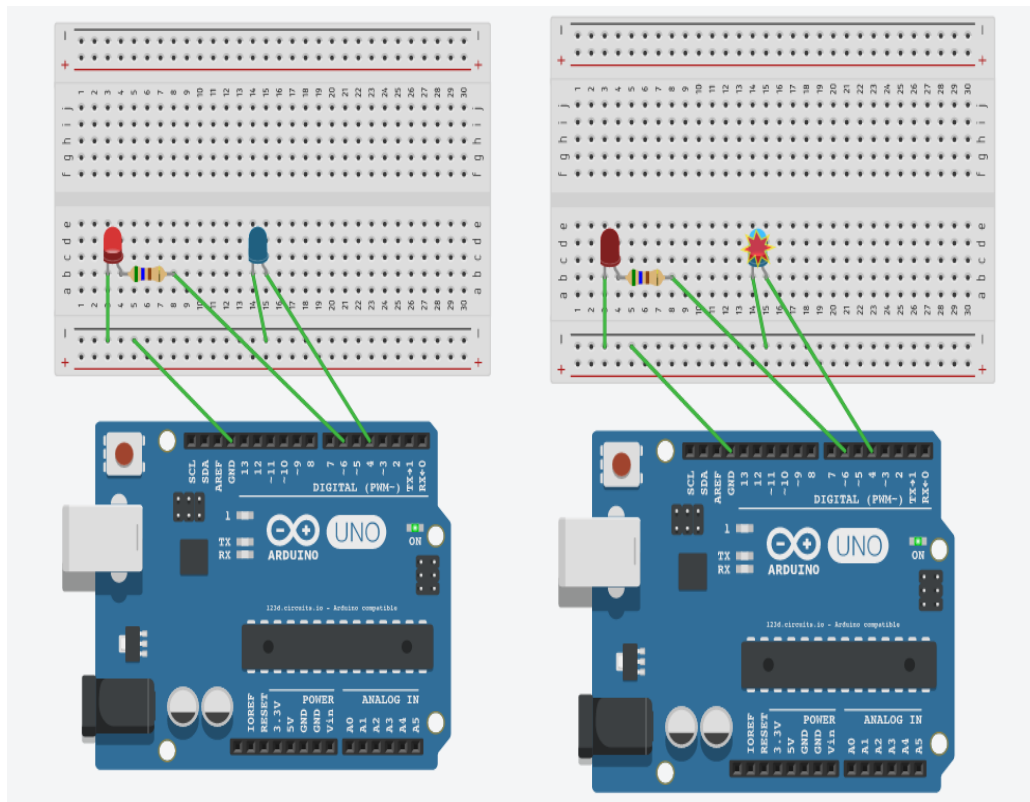


Παρ' όλο που προσθέτουμε την αντίσταση ο κώδικας δεν αλλάζει καθώς ο αντιστάτης είναι ηλεκτρονικό στοιχείο.

Παράδειγμα 3^ο

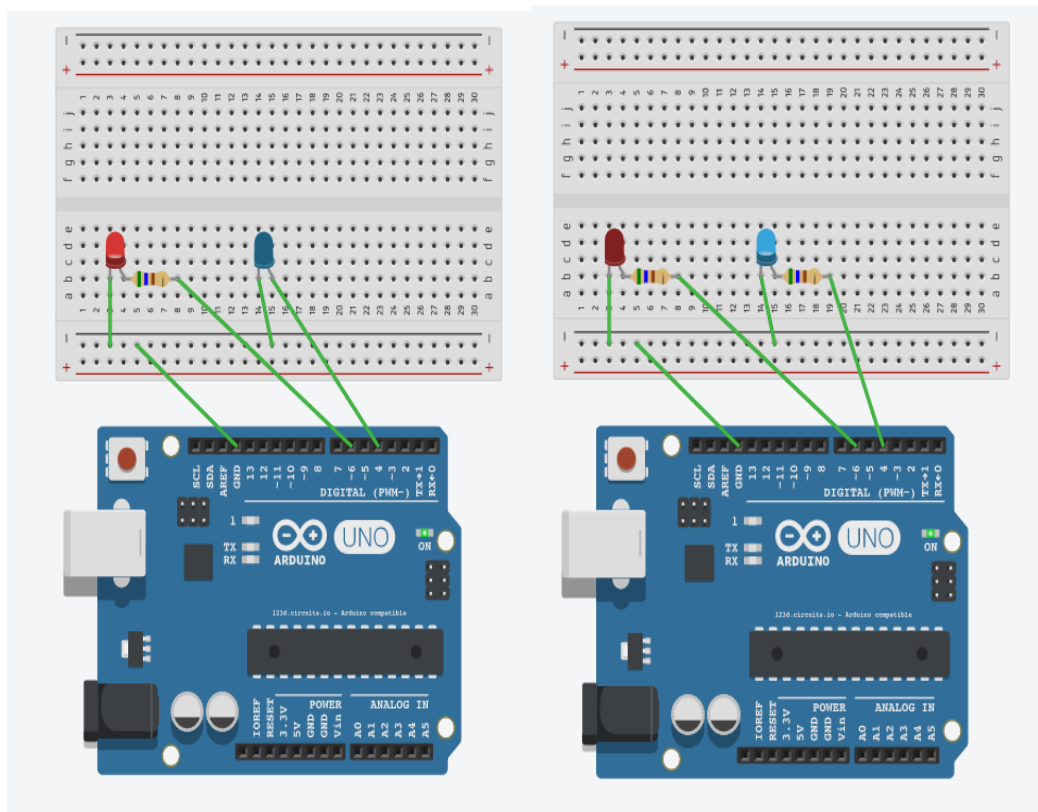
Τώρα θα συνδέσουμε δύο LED, ένα κόκκινο κι ένα μπλε.

```
1 // σύνδεση στο pin 6 του κόκκινου LED και στο pin4 του μπλε LED.
2 // ονομάζεται redpinLed το κόκκινο και bluepinLed το μπλε:
3 int redpinLed = 6;
4 int bluepinLed = 4;
5
6
7 void setup() {
8 // αρχικοποίηση του ψηφιακού redpinLed και του bluepinLed ως έξοδοι
9   pinMode(redpinLed, OUTPUT);
10  pinMode(bluepinLed, OUTPUT);
11 }
12
13 void loop() {
14   digitalWrite(redpinLed, HIGH); // ενεργοποιείται το κόκκινο LED (HIGH)
15   delay(1000); // αναμονή για 1 δευτερόλεπτο
16   digitalWrite(redpinLed, LOW); // απενεργοποιείται το κόκκινο LED (LOW)
17   delay(1000); // αναμονή για 1 δευτερόλεπτο
18   digitalWrite(bluepinLed, HIGH); // ενεργοποιείται το μπλε LED (HIGH)
19   delay(1000); // αναμονή για 1 δευτερόλεπτο
20   digitalWrite(bluepinLed, LOW); // απενεργοποιείται το μπλε LED (LOW)
21   delay(1000); // αναμονή για 1 δευτερόλεπτο
22 }
```



Παρατηρούμε ότι το κόκκινο LED που είναι συνδεδεμένο με αντιστάτη ανάβει χωρίς τον κίνδυνο να καεί, ενώ το μπλε ανάβει με τον κίνδυνο καψίματος.

Έτσι, θα προσθέσουμε έναν αντιστάτη και στο μπλε LED χωρίς να τροποποιηθεί ο κώδικας.



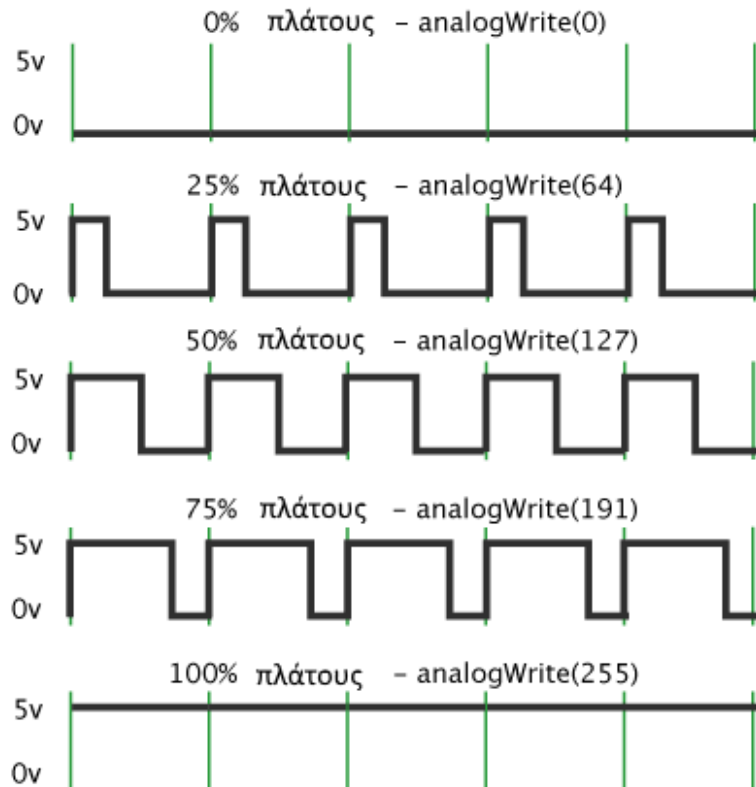
Στο παράδειγμα αυτό θα δείτε πως στο breadboard συνδέσαμε το pin GND του Arduino στη γραμμή – και την γείωση από κάθε led την συνδέσαμε με αυτή τη γραμμή, άρα με τη γείωση του Arduino. Αυτή είναι η χρήση του breadboard στην δημιουργία πρωτότυπων σχηματικών.

Παράδειγμα 4^ο

Σε αυτό το παράδειγμα έχουμε ένα LED του οποίου η φωτεινότητα θα αυξάνεται και θα μειώνεται. Χρησιμοποιούμε για το πετύχουμε αυτό την αναλογική μέθοδο εγγραφής (analogWrite) στο pin 6 του Arduino.

Η analogWrite(pin, value) στέλνει μια αναλογική τιμή με τη μορφή παλμού (Pulse Width Modulation ή PWM wave) στο pin 6. Η τεχνική PWM χρησιμοποιείται για να έχουμε αναλογικά αποτελέσματα με ψηφιακά μέσα. Η value παίρνει τιμή από 0 (0V) ως 255 (5V), οπότε έχουμε τη δυνατότητα να έχουμε αλλαγή στη φωτεινότητα.

Για όποιον θέλει να μάθει για την τεχνική αυτή αναφέρουμε πως υλοποιείται στέλνοντας ένα σήμα ρεύματος ανοίγοντας και κλείνοντας την τροφοδοσία για κάποιο χρονικό διάστημα στην ουσία στέλνοντας ένα παλμό. Αυτή η εναλλαγή On-Off μπορεί να εξομοιώσει τάσεις από 5V στο 0V. Ο χρόνος που είναι On, περνά δηλαδή τάση, είναι το πλάτος του παλμού. Αν ο παλμός είναι στο 100% του πλάτους έχουμε τάση 5V, αν είναι 50% τότε 2,5V και αν 0 τότε 0V (βλ. Εικόνα 10).

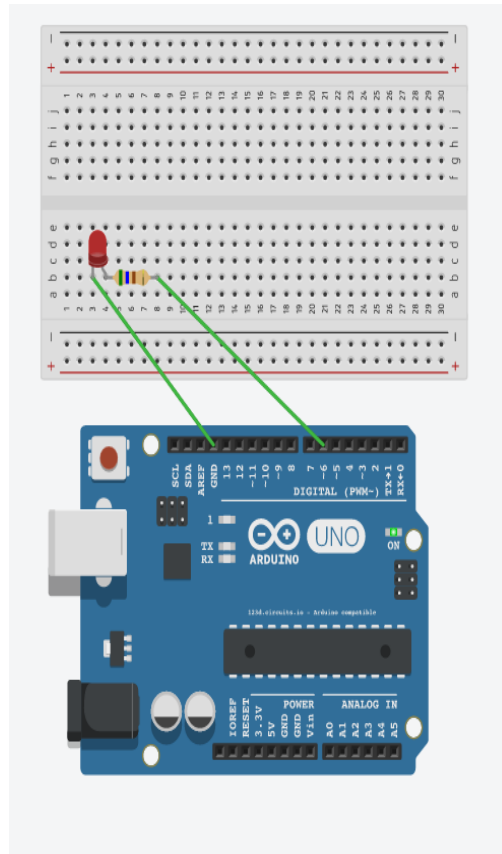
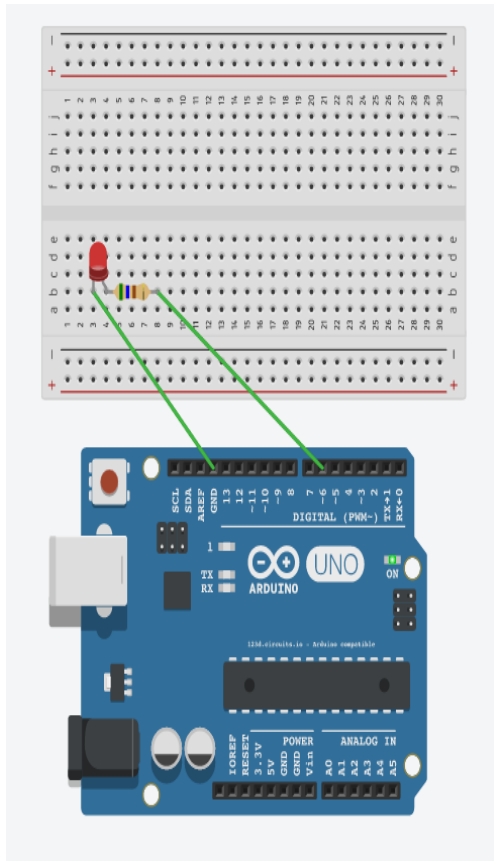


Εικόνα 10 PWM με analogWrite

```

1 // σύνδεση στο pin 6 του LED
2 // ονομάζεται redpinLed:
3 int pinLed = 6;
4 int brightness = 0; // πόσο φωτεινό είναι το LED
5 int fadeAmount = 5; // πόσο να μειωθεί η φωτεινότητα του LED
6
7 void setup() {
8 // αρχικοποίηση του ψηφιακού pinLed ως έξοδος
9   pinMode(pinLed, OUTPUT);
10 }
11
12 void loop() {
13   analogWrite(pinLed, brightness); // θέτει τη φωτεινότητα του LED
14   brightness = brightness + fadeAmount; // αλλάζει τη φωτεινότητα για την επόμενη επανάληψη
15
16 // αντιστρέφεται η διεύθυνση της μείωσης της φωτεινότητας
17 if (brightness <= 0 || brightness >= 255) {
18   fadeAmount = -fadeAmount;
19 }
20
21 delay(30); // αναμονή για 30 miliseconds
22 }
23

```

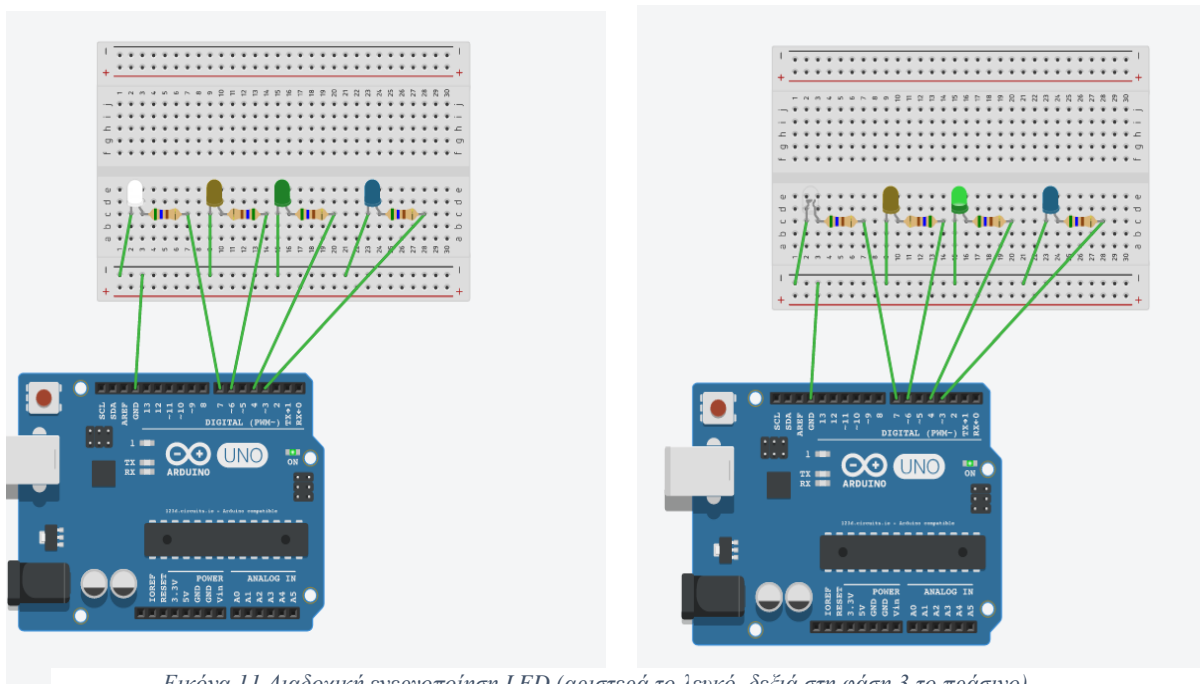



Στο παράδειγμα αυξάνεται η φωτεινότητα από το 0-255 (5V) με βήμα 5 και μετά κατεβαίνει στο 0 με το ίδιο βήμα.

Παράδειγμα 5^ο

```
1 // σύνδεση του διάφανου Led στο Pin 7, του κίτρινου στο 6, του πράσινου στο 4 και του μπλε στο 3
2 // ονοματίζονται τα Pin:
3 int clearpinLed = 7;
4 int yellowpinLed = 6;
5 int greenpinLed = 4;
6 int bluepinLed = 3;
7
8 void setup() {
9   // αρχικοποίηση όλων των Pin ως έξοδοι
10  pinMode(clearpinLed, OUTPUT);
11  pinMode(yellowpinLed, OUTPUT);
12  pinMode(greenpinLed, OUTPUT);
13  pinMode(bluepinLed, OUTPUT);
14 }
15
16
17 void loop() {
18  digitalWrite(clearpinLed, HIGH); // ενεργοποιείται το διάφανο LED
19  delay(1000); // παύση για ένα δευτερόλεπτο
20  digitalWrite(clearpinLed, LOW); // απενεργοποιείται το διάφανο LED
21  delay(1000); // παύση για ένα δευτερόλεπτο
22  digitalWrite(yellowpinLed, HIGH); // ενεργοποιείται το κίτρινο LED
23  delay(900); // παύση για 0.9 δευτερόλεπτα
24  digitalWrite(yellowpinLed, LOW); // απενεργοποιείται το κίτρινο LED
25  delay(900); // παύση για 0.9 δευτερόλεπτα
26  digitalWrite(greenpinLed, HIGH); // ενεργοποιείται το πράσινο LED
27  delay(700); // παύση για 0.7 δευτερόλεπτα
28  digitalWrite(greenpinLed, LOW); // απενεργοποιείται το πράσινο LED
29  delay(700); // παύση για 0.7 δευτερόλεπτα
30  digitalWrite(bluepinLed, HIGH); // ενεργοποιείται το μπλε LED
31  delay(300); // παύση για 0.3 δευτερόλεπτα
32  digitalWrite(bluepinLed, LOW); // απενεργοποιείται το μπλε LED
33  delay(300); // παύση για 0.3 δευτερόλεπτα
34 }
```

Σε αυτό το παράδειγμα συνδέουμε τέσσερα LED τα οποία θα αναβοσβήνουν το ένα μετά το άλλο με διαφορετική καθυστέρηση το καθένα.

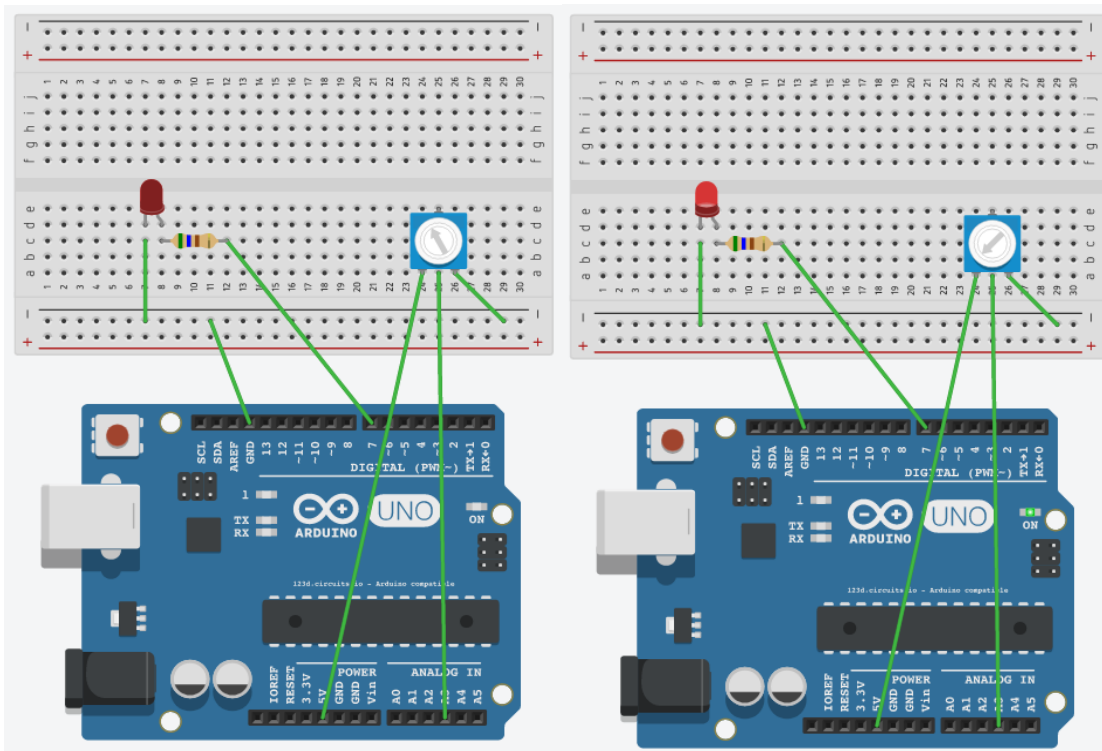


Εικόνα 11 Διαδοχική ενεργοποίηση LED (αριστερά το λευκό, δεξιά στη φάση 3 το πράσινο)

Παράδειγμα 6°

Εδώ χρησιμοποιείται ένα ποτενσιόμετρο ώστε να ρυθμίζεται η ένταση του LED μέσω αυτού.

```
1 // συνδέεται το LED στο pin 7
2 int pinled = 7;
3 // συνδέεται το ποτενσιόμετρο στο αναλογικό pin 3
4 int analogpin = 3;
5 // αρχικοποιείται η μεταβλητή val
6 int val = 0;
7
8 void setup() {
9 //αρχικοποιείται το ψηφιακό pin 7 ως έξοδος
10 pinMode(pinled, OUTPUT);
11 }
12
13 void loop() {
14 val=analogRead(analogpin)/4; /* διαβάζεται το αναλογικό pin 3
15                               η τιμή του είναι 0-1023
16                               στην μεταβλητή val αποθηκεύεται τιμή 0-255 */
17 // δίνεται τάση στο LED ανάλογη με την τιμή val από το ποτενσιόμετρο
18 analogWrite(pinled, val);
19 }
```

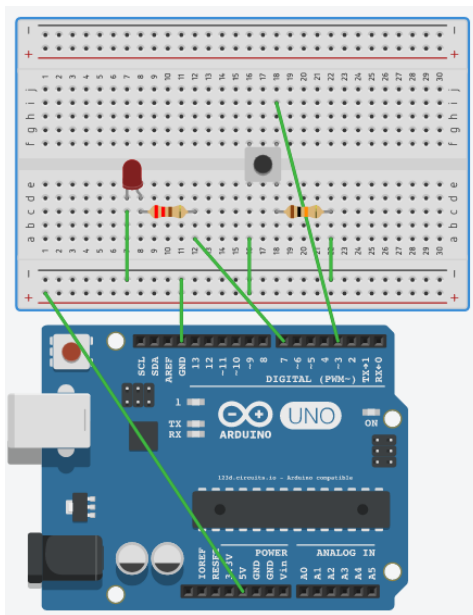


Στο παράδειγμα αυτό το ποτενσιόμετρο συνδέεται στην τάση 5V (αριστερός ακροδέκτης) στη γείωση (δεξιός ακροδέκτης) και από το μεσαίο ακροδέκτη διαβάζεται η τιμή που έχει η μεταβλητή του αντίστασης στο αναλογικό pin 3. Στρέφοντας το ποτενσιόμετρο αλλάζουμε την αντίστασή του την οποία διαβάζει το Arduino στο αναλογικό pin 3 (τιμές από 0-1023), την διαιρεί με το 4 για να δημιουργηθεί εύρος τιμών από 0-255 που μπορεί να γράψουμε στο pin 7 του Arduino που βρίσκεται το LED.

Παράδειγμα 7^ο

Σε αυτό το παράδειγμα θα χρησιμοποιήσουμε ένα κουμπί το οποίο όσο πατιέται θα ανάβει το LED μας.

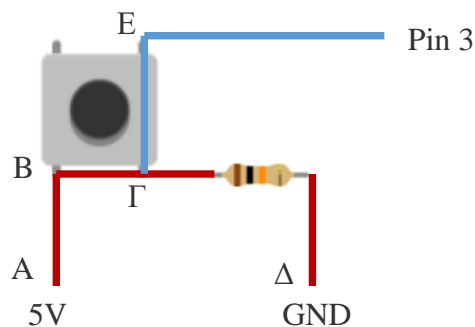
```
1 int pushbuttonPin = 3; // pin για το push button
2 int ledPin = 7; // LED pin
3
4
5 int buttonState = 0; // μεταβλητή που κρατά αν είναι πατημένο το κουμπί
6
7 void setup() {
8
9     pinMode(ledPin, OUTPUT); // το pin του LED ορίζεται OUTPUT
10    pinMode(pushbuttonPin, INPUT); // το pin του push button ορίζεται INPUT
11 }
12
13 void loop() {
14     // διάβασε αν είναι πατημένο το κουμπί
15     buttonState = digitalRead(pushbuttonPin);
16
17     /* εξέτασε αν το κουμπί είναι πατημένο.
18     // αυτό ισχύει αν η buttonState είναι HIGH */
19     if (buttonState == HIGH) {
20         // άναψε το LED
21         digitalWrite(ledPin, HIGH);
22     } else {
23         // σβήσε το LED
24         digitalWrite(ledPin, LOW);
25     }
26 }
```



Εικόνα 12 Σχηματικό σύνδεσης Push Button

Στο παράδειγμα αυτό χρησιμοποιήσαμε έναν αντιστάτη 220Ω για να μειώσουμε την τάση που περνά από το led και ένα αντιστάτη 10kΩ για να οδηγείται το ρεύμα από την διαδρομή που έχει μικρότερη αντίσταση όταν πατάμε το κουμπί (βλ. Εικόνα 12).

Έτσι όταν δεν πατάμε το κουμπί το ρεύμα πάει από το Α-Β-Γ-μέσω του αντιστάτη-στο Δ και τη γείωση (GND), ενώ όταν πατάμε το κουμπί επιλέγει τη διαδρομή Α-Β-Γ-Ε-Pin 3 γιατί η προηγούμενη διαδρομή έχει μεγαλύτερη αντίσταση για να «τρέξει» το ρεύμα (βλ. Εικόνα 13). Το ρεύμα πάντα ακολουθεί τη διαδρομή με τη λιγότερη αντίσταση.



Εικόνα 13 Διαδρομή ρεύματος με αντιστάτη

6. Βιβλιογραφία

1. Purdum Jack, “*Beginning C for Arduino*”, Apress, 2012
2. McEoberts Michael, “*Beginning Arduino*”, Apress, 2012, 2nd Edition
3. Evans Brian, “*Beginning Arduino Programming*”, Apress, 2011
4. Amariei Cornel, “*Arduino Development Cookboo*”, PACKT publishing, 2015
5. Langbridge James, “*Arduino Sketches*”, WILEY, 2015