Εξομοίωση PID Ελεγκτή μέσω Arduino για ελεγχό κινητήρα

Τ.Ε.Ι. ΠΕΙΡΑΙΑ
ΤΜΗΜΑ: Η.Υ.Σ

Άρης Λάμπρης
Σταύρος Κοκόζης

Εισαγωγή

Η παρούσα πτυχιακή εργασία πραγματεύεται την εξομοίωση του PID μικροελεγκτή μέσω του ολοκληρωμένου κυκλώματος Arduino με σκοπό την οδήγηση και έλεγχο ενός κινητήρα DC. Στην πτυχιακή αυτή εργασία επιτυγχάνουμε να πραγματοποιήσουμε έλεγχο ταχύτητας και θέσης ενός κινητήρα συνεχούς τάσης. Πρόκειται επομένως για ένα σύστημα αυτόματου ελέγχου (Σ.Α.Ε) και την υλοποίηση του μέσω της πολύ διαδεδομένης και εύκολης στην χρήση της πλατφόρμας Arduino.

Στο πρώτο κεφάλαιο της εργασίας θα περιγράψουμε αναλυτικά την μητρική πλακέτα Arduino Uno: ιστορική αναδρομή, το ολοκληρωμένο ATmega328 και την μνήμη που χρησιμοποιεί, τις εισόδους/εξόδους του, την τροφοδοσία του, τα ενσωματωμένα πάνω στην πλακέτα κουμπιά και led, το ολοκληρωμένο περιβάλλον ανάπτυξής του (IDE) και την σύνδεση με τον υπολογιστή, την γλώσσα προγραμματισμού που χρησιμοποιεί προκειμένου να δημιουργήσουμε τα προγράμματά μας, διάφορα παραδείγματα και υλοποιήσεις προγραμμάτων για την κατανόηση της εύκολης και ευέλικτης χρήσης του Arduino.Τέλος, γίνεται αναφορά στις διάφορες κατηγορίες μητρικών πλακετών Arduino και των χαρακτηριστικών τους, καθώς και η παρουσίαση κάποιων επεκτάσεων υλικού ονομαζόμενα ως Shields που αυξάνουν τις λειτουργίες ενός Arduino.

Στο δεύτερο κεφάλαιο γίνεται αναφορά στον ελεγκτή PID. Γίνεται ιστορική αναδρομή του ελεγκτή, κάποιες βασικές σε βρόχους ελέγχου. Στη συνέχεια περιγράφεται η λειτουργία και η χρησιμότητα ενός PID ελεγκτή, έπειτα αναλύεται ο κάθε όρος ενός PID χωριστά, καθώς και οι διάφοροι συνδυασμοί των τριών όρων του (Αναλογικός-Ολοκληρωτικός-Παραγωγικός). Επίσης, αναφέρονται παραδείγματα γραμμένα σε Matlab και γραφήματα για να γίνει πιο εύκολα κατανοητή η λειτουργία και η χρησιμότητα ενός PID ελέγχου σε ένα σύστημα αυτοματισμού. Τέλος, γίνεται αναφορά στα χαρακτηριστικά ενός PID ελεγκτή και παρατάσσονται οι διάφορες μέθοδοι σχεδίασης ελεγκτών PID.

Στο τρίτο και τελευταίο κεφάλαιο περιγράφεται αναλυτικά το πειραματικό μέρος, όπου υλοποιείται το σύστημα ελέγχου της θέσης και της ταχύτητας του κινητήρα. Τα επιμέρους στοιχεία που απαρτίζουν το σύστημα ελέγχου, καθώς και περιγραφή της λειτουργίας και συνδεσμολογίας τους. Επιπλέον γίνεται εξομοίωση μέσω της μητρικής πλακέτας Arduino Uno του αλγόριθμου ελέγχου PID, η ανάλυση του κώδικα που το αφορά και τα τελικά συμπεράσματα.

Τέλος, σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε την καθηγήτριά μας Αναστασία Βελώνη του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Πειραιά, κυρίως για την εμπιστοσύνη που μας έδειξε, την υπομονή και την υποστήριξή της κατά τη διάρκεια υλοποίησης της πτυχιακής μας εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1-ARDUINO UNO

ΚΕΦΑΛΑΙΟ 2- Ο ΕΛΕΓΚΤΗΣ PID

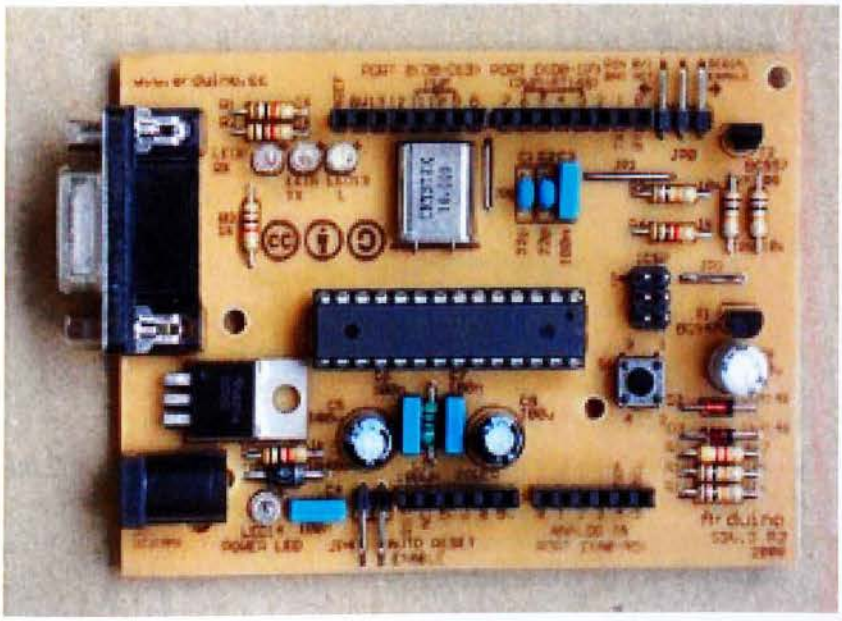
ΚΕΦΑΛΑΙΟ 3- ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ

ΚΕΦΑΛΑΙΟ 1

Ο Μικροελεγκτής Arduino

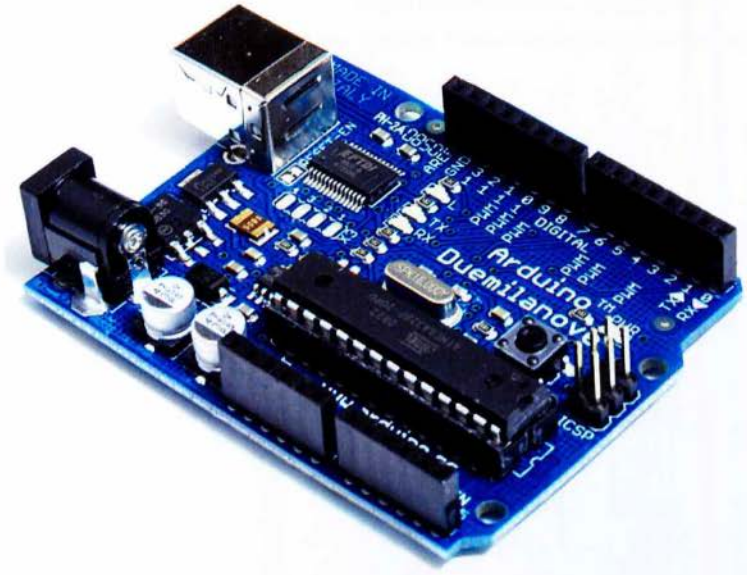
1.1 Ιστορία του Μικροελεγκτή "Arduino"

Το 2005, στην Ivrea της Ιταλίας ένα έργο άρχισε να δημιουργείτε, μια συσκευή για τον έλεγχο σχεδίων, χτισμένο από μαθητές με λιγότερα έξοδα από ότι με άλλα πρωτότυπα συστήματα που ήταν διαθέσιμα εκείνη τη χρονική στιγμή. Από το Μάιο του 2011, περισσότερα από 300,000 Arduino ήταν διαθέσιμα «στην άγρια φύση» της αγοράς. Οι εφευρέτες Massimo Banzi και David Cuartielles, ονόμασαν το έργο τους Arduin of Ivrea. Το «Arduino» είναι επίσης ένα ιταλικό όνομα, που σημαίνει «γενναίος φίλος».
Το έργο Arduino είναι μια παραγόμενη έκδοση της πλατφόρμας ανοικτού κώδικα Wiring Platform. Ο Κολομβιανός καλλιτέχνης και προγραμματιστής Hernando Barragan δημιούργησε γλώσσα προγραμματισμού «Wiring» ως μια Μεταπτυχιακή διπλωματική εργασία στο Interaction Design Institute of Ivrea (IDII) υπό την εποπτεία του Massimo Banzi και του Casey Reas. Η Wiring βασίστηκε στην επεξεργασία και το ολοκληρωμένο περιβάλλον ανάπτυξης που είχε δημιουργηθεί από τον Casey Reas και τον Ben Fry.



1.2 Τι είναι το Arduino

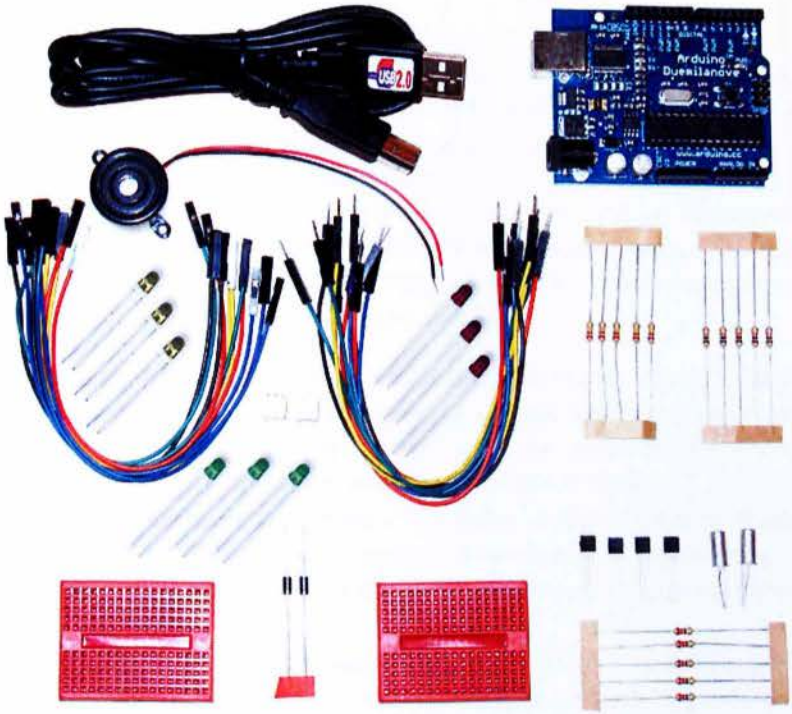
Το Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «προτυποποίησης» ηλεκτρονικών βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software που προορίζεται για οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα.



Στην ουσία, πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στον μικροελεγκτή ATmega της Atmel και του οποίου όλα τα σχέδια, καθώς και το software που χρειάζεται για την λειτουργία του, διανέμονται ελεύθερα και δωρεάν ώστε να μπορεί να κατασκευαστεί από τον καθένα (απ' όπου και ο περίεργος -για hardware- χαρακτηρισμός «ανοικτού κώδικα»). Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, αφού ο χρήστης μπορεί να συνδέσει επάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει

τον μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου.

Μάλιστα κάποιος θα μπορούσε να ισχυριστεί - και θα ήταν ένας αρκετά πετυχημένος παραλληλισμός - ότι λειτουργικά το Arduino μοιάζει πολύ με το NXT Brick των Lego Mindstorms NXT. Άλλωστε η ρομποτική είναι μια από τις πολλές εφαρμογές στις οποίες το Arduino διαπρέπει.
Το Arduino βέβαια, δεν είναι ούτε ο μοναδικός, ούτε και ο καλύτερος δυνατός τρόπος για την δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Όμως το κύριο πλεονέκτημά του είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια ανάλογου μεγέθους online γνωσιακή βάση. Έτσι, παρότι ένας έμπειρος ηλεκτρονικός μπορεί να προτιμήσει διαφορετική πλατφόρμα ή εξαρτήματα ανάλογα με την εφαρμογή που έχει στον νου του, το Arduino, με το εκτενές documentation, καταφέρνει να κερδίσει όλους αυτούς των οποίων οι γνώσεις στα ηλεκτρονικά περιορίζονται στα όσα λίγα έμαθαν στο σχολείο.



Ακριβώς επειδή απευθύνεται κυρίως σε **αρχάριους** των ηλεκτρονικών και επειδή, παρά τις αναλυτικότατες οδηγίες που υπάρχουν, δεν έχουν όλοι τις γνώσεις και τα μέσα να κατασκευάσουν μια ηλεκτρονική πλακέτα, κυκλοφορούν έτοιμες, προκατασκευασμένες πλακέτες **Arduino** που μπορείτε να προμηθευτείτε για περίπου **€25**. Με λίγα χρήματα παραπάνω μάλιστα, οι περισσότεροι προμηθευτές διαθέτουν **Arduino Starter Kit**, τα οποία, εκτός από το ίδιο το **Arduino**, περιέχουν διάφορα άλλα εξαρτήματα και εργαλεία που μπορεί να χρειαστείτε για τις πρώτες σας εφαρμογές (όπως το απαραίτητο καλώδιο **USB** για την σύνδεση με τον υπολογιστή, **καλώδια**, **LEDs**, **διακόπτες**, **ποτενσιόμετρα**, **αντιστάσεις**, **διόδους**, **τρανζίστορς** κ.λπ.).

1.3 Γιατί Arduino;

Υπάρχουν πολλοί άλλοι μικροελεγκτές και μικροεπεξεργαστικές πλατφόρμες διαθέσιμα για φυσική επεξεργασία και υπολογισμούς. Μερικά είναι, **Parallel Basic Stamp**, **Netmedia's BX-24**, **Phidgets**, **MIT's Handyboard**, και πολλά άλλα που προσφέρουν παρόμοιες λειτουργίες με το **Arduino**. Όλα αυτά τα εργαλεία απομονώνουν τις χαοτικές λεπτομέρειες του μικροελεγκτικού προγραμματισμού και μετατρέπουν την διαδικασία σε ένα εύκολο στη χρήση περιβάλλον λειτουργίας. Το Arduino επίσης απλοποιεί την διαδικασία χειρισμού μικροελεγκτών, αλλά προσφέρει και μερικά άλλα πλεονεκτήματα για καθηγητές, μαθητές και ενδιαφερόμενους ερασιτέχνες από τα άλλα μικροελεγκτικά συστήματα:

- **Χαμηλό κόστος** - Οι **Arduino πλακέτες** είναι σχετικά φθηνές σε σχέση με άλλες μικροελεγκτικές πλατφόρμες. Οι λιγότερο ακριβή έκδοση του **Arduino** τμήματος μπορεί να συναρμολογηθεί με το χέρι, ακόμα και ένα πλήρες συναρμολογημένο Arduino κοστίζει λιγότερο από 39 ευρώ
- **Διασταυρώμενη πλατφόρμα** – Το Arduino λογισμικό τρέχει σε Windows, Macintosh OSX και Linux λειτουργικά συστήματα. Τα περισσότερα μικροελεγκτικά συστήματα απλώς περιορίζονται σε λειτουργικά συστήματα Windows
- **Απλό, κατανοητό προγραμματιστικό περιβάλλον** – Το προγραμματιστικό περιβάλλον Arduino είναι **εύκολο-στη-χρήση** για αρχάριους, ακόμα είναι

ευέλικτο αρκετά και για προχωρημένους χρήστες. Για καθηγητές, είναι βολικά βασισμένο στο Processing προγραμματιστικό περιβάλλον, έτσι οι μαθητές που μαθαίνουν να προγραμματίζουν σε αυτό το περιβάλλον θα είναι γνώριμοι με την όψη και την αίσθηση ενός Arduino

- **Ανοιχτού κώδικα και επεκτάσιμου λογισμικού** – Το Arduino λογισμικό είναι δημοσιευμένο ως ένα ανοιχτού κώδικα εργαλείο, διαθέσιμο για επέκταση από έμπειρους προγραμματιστές. Η γλώσσα μπορεί να επεκταθεί μέσα από τις βιβλιοθήκες της γλώσσας προγραμματισμού C++, και για άτομα που θέλουν να κατανοήσουν τις τεχνικές λεπτομέρειες μπορούν εύκολα να μετεπηδήσουν από Arduino σε AVR C γλώσσα προγραμματισμού στην οποία είναι βασισμένο. Ομοίως, μπορείτε να προσθέσετε AVR-C κώδικα απευθείας στα Arduino προγράμματά σας εάν το επιθυμείτε.
- **Ανοιχτού κώδικα και επεκτάσιμου υλικού** – Το Arduino είναι βασισμένο στης Atmel τους μικροελεγκτές ATMEGA8 και ATMEGA168. Τα σχέδια για τα λειτουργικά τμήματα είναι δημοσιευμένα κάτω από ένα Δημιουργική Κοινή Άδεια (Creative Common License), με αυτό τον τρόπο έμπειροι σχεδιαστές και κατασκευαστές κυκλωμάτων μπορούν να δημιουργήσουν τις δικές τους εκδόσεις λειτουργικών τμημάτων, επεκτείνοντας και βελτιώνοντας τις δυνατότητες του Arduino. Ακόμα και ένας σχετικά άπειρος χρήστης μπορεί να χτίσει την μητρική πλακέτα για να κατανοήσει πως λειτουργεί ένα Arduino, καθώς και για να μειώσει το κόστος.

1.4 Μικροελεγκτής και Μνήμη του Arduino

Το **Arduino** βασίζεται στον **ATmega328**, έναν **8-bit RISC** μικροελεγκτή, τον οποίο χρονίζει στα **16MHz**. Ο **ATmega328** διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- **2Kb** μνήμης **SRAM** που είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματά σας 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**.

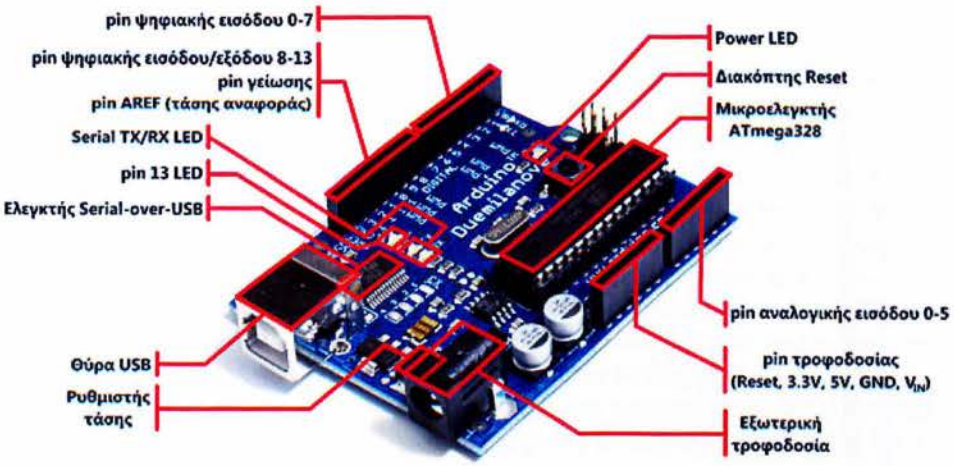
- **1Kb** μνήμης **EEPROM** η οποία μπορεί να χρησιμοποιηθεί για «ωμή» εγγραφή/ανάγνωση δεδομένων (χωρίς **datatype**) ανά byte από τα προγράμματά σας κατά το **runtime**. Σε αντίθεση με την **SRAM**, η **EEPROM**

δεν χάνει τα περιεχόμενά της με απώλεια **τροφοδοσίας** ή **reset** οπότε είναι το ανάλογο του **σκληρού δίσκου**.

- **32Kb** μνήμης **Flash**, 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 δικών σας 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 σας. Η μνήμη **Flash**, όπως και η **EEPROM** δεν χάνει τα περιεχόμενά της με απώλεια **τροφοδοσίας** ή **reset**. Επίσης, ενώ η μνήμη **Flash** υπό κανονικές συνθήκες δεν προορίζεται για χρήση **runtime** μέσα από τα προγράμματά σας, λόγω της μικρής συνολικής μνήμης που είναι διαθέσιμη σε αυτά (**2Kb SRAM** + **1Kb EEPROM**), έχει σχεδιαστεί μια βιβλιοθήκη που επιτρέπει την χρήση όσου χώρου περισσεύει (**30Kb** μείον το μέγεθος του προγράμματός σας σε **μεταγλωττισμένη** μορφή).

1.5 Είσοδοι - Έξοδοι

Το **Arduino** διαθέτει **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. Η σύνδεση αυτή χρησιμοποιείται για την μεταφορά των προγραμμάτων που σχεδιάζονται από τον υπολογιστή στο **Arduino** αλλά και για αμφίδρομη επικοινωνία με τον υπολογιστή μέσα από το πρόγραμμα την ώρα που εκτελείται.



Στην πάνω πλευρά του **Arduino** βρίσκονται **14 θηλυκά pins**, αριθμημένα από **0** ως **13**, που μπορούν να λειτουργήσουν ως **ψηφιακές είσοδοι** και **έξοδοι**. Λειτουργούν στα **5V** και καθένα μπορεί να παρέχει ή να δεχτεί το πολύ **40mA**.
Ως **ψηφιακή έξοδος**, ένα από αυτά τα **pins** μπορεί να τεθεί από το πρόγραμμά σας σε κατάσταση **HIGH** ή **LOW**, οπότε το **Arduino** θα ξέρει αν πρέπει να διοχετεύσει ή όχι ρεύμα στο συγκεκριμένο **pin**. Με αυτόν τον τρόπο μπορείτε λόγου χάρη να ανάψετε και να σβήσετε ένα **LED** που έχετε συνδέσει στο συγκεκριμένο **pin**. Αν πάλι ρυθμίσετε ένα από αυτά τα **pins** ως **ψηφιακή είσοδο** μέσα από το πρόγραμμά σας, μπορείτε με την κατάλληλη εντολή να διαβάσετε την κατάστασή του (**HIGH** ή **LOW**) ανάλογα με το αν η **εξωτερική συσκευή** που έχετε συνδέσει σε αυτό το **pin** διοχετεύει ή όχι ρεύμα στο **pin** (με αυτόν τον τρόπο λόγου χάρη μπορείτε να «**διαβάζετε**» την κατάσταση ενός **διακόπτη**).
Μερικά από αυτά τα **14 pins**, εκτός από **ψηφιακές είσοδοι/έξοδοι** έχουν και δεύτερη λειτουργία.

Συγκεκριμένα:

- Τα **pins 0** και **1** λειτουργούν ως **RX** και **TX** της **σειριακής** επικοινωνίας, όταν το πρόγραμμά σας ενεργοποιεί την **σειριακή θύρα**. Έτσι, όταν λόγου χάρη το πρόγραμμά σας 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**). Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμά σας ενεργοποιήσετε το **σειριακό interface**, χάνετε 2 ψηφιακές εισόδους/εξόδους.
- Τα **pins 2** και **3** λειτουργούν και ως **εξωτερικές διακοπές** (**interrupts 0** και **1** αντίστοιχα). Με άλλα λόγια, μπορείτε να τα ρυθμίσετε μέσα από το πρόγραμμά σας ώστε να λειτουργούν αποκλειστικά ως **ψηφιακές είσοδοι** στις οποίες όταν συμβαίνουν **συγκεκριμένες αλλαγές**, η κανονική ροή του προγράμματος σταματάει ***άμεσα*** και εκτελείται μια **συγκεκριμένη**

συνάρτηση. Τα **εξωτερικά interrupts** είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν **συγχρονισμό** μεγάλης ακρίβειας.

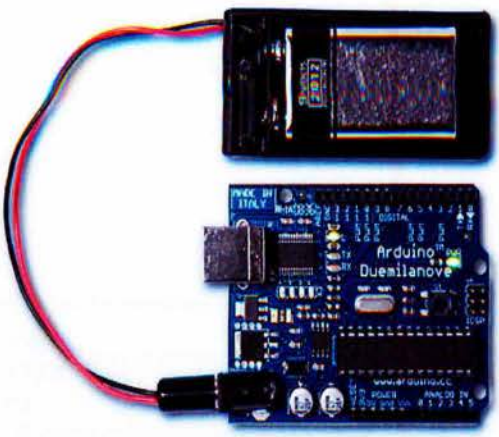
- Τα **pins 3, 5, 6, 9, 10** και **11** μπορούν να λειτουργήσουν και ως **ψευδό-αναλογικές έξοδοι** με το σύστημα **PWM (Pulse Width Modulation)**, δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Έτσι, μπορείτε να συνδέσετε λόγου χάρη ένα **LED** σε κάποιο από αυτά τα **pins** και να ελέγξετε πλήρως την **φωτεινότητά** του με **ανάλυση 8bits (256** καταστάσεις από **0-σβηστό** ως **255-πλήρως αναμμένο**) αντί να έχετε απλά την δυνατότητα **αναμμένο-σβηστό** που παρέχουν οι υπόλοιπές **ψηφιακές έξοδοι**. Είναι σημαντικό να καταλάβουμε ότι το **PWM** δεν είναι πραγματικά **αναλογικό σύστημα** και ότι θέτοντας στην **έξοδο** την **τιμή 127**, δεν σημαίνει ότι η **έξοδος** θα δίνει **2.5V** αντί της κανονικής τιμής των **5V**, αλλά ότι θα δίνει ένα **παλμό** που θα εναλλάσσεται με **μεγάλη συχνότητα** και για **ίσους** χρόνους μεταξύ των **τιμών 0** και **5V**.

  Στην κάτω πλευρά του **Arduino**, με τη σήμανση **ANALOG IN**, θα βρείτε μια ακόμη σειρά από **6 pins**, 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. Για παράδειγμα, μπορείτε να τροφοδοτήσετε ένα από αυτά με μια τάση την οποία μπορούμε να κυμάνουμε με ένα **ποτενσιόμετρο** από **0V** ως μια **τάση αναφοράς Vref** η οποία, αν δεν κάνετε κάποια αλλαγή είναι **προρυθμισμένη** στα **5V**. Τότε, μέσα από το πρόγραμμά μας μπορούμε να «**διαβάσουμε**» την τιμή του **pin** ως ένα ακέραιο αριθμό **ανάλυσης 10-bits**, από **0** (όταν η τάση στο **pin** είναι **0V**) μέχρι **1023** (όταν η **τάση** στο **pin** είναι **5V**).

  Η **τάση αναφοράς** μπορεί να ρυθμιστεί με μια εντολή στο **1.1V**, ή σε όποια τάση επιθυμείτε (μεταξύ **2V** και **5V**) τροφοδοτώντας εξωτερικά με αυτή την τάση το **pin** με την σήμανση **AREF** που βρίσκεται στην απέναντι πλευρά της πλακέτας. Έτσι, αν τροφοδοτήσετε το **pin AREF** με **3.3V** και στην συνέχεια δοκιμάσετε να διαβάσετε κάποιο **pin αναλογικής εισόδου** στο οποίο εφαρμόζετε τάση **1.65V**, το **Arduino** θα σας επιστρέψει την **τιμή 512**. Τέλος, καθένα από τα **6** αυτά **pins**, με κατάλληλη εντολή μέσα από το πρόγραμμα μπορεί να μετατραπεί σε **ψηφιακό pin εισόδου/εξόδου** όπως τα **14** που βρίσκονται στην απέναντι πλευρά και τα οποία περιγράφηκαν πριν. Σε αυτή την περίπτωση τα **pins** μετονομάζονται από **0~5** σε **14~19** αντίστοιχα.

1.6 Τροφοδοσία

Το **Arduino** μπορεί να τροφοδοτηθεί με ρεύμα είτε από τον **υπολογιστή** μέσω της σύνδεσης **USB**, είτε από **εξωτερική** τροφοδοσία που παρέχεται μέσω μιας υποδοχής **φις των 2.1mm** (θετικός πόλος στο κέντρο) και βρίσκεται στην κάτω-αριστερή γωνία του **Arduino**.



Για να μην υπάρχουν προβλήματα, η **εξωτερική τροφοδοσία** πρέπει να είναι από **7V** ως **12V** και μπορεί να προέρχεται από ένα κοινό μετασχηματιστή του εμπορίου, από μπαταρίες ή οποιαδήποτε άλλη πηγή DC.
Δίπλα από τα **pins αναλογικής εισόδου**, υπάρχει μια ακόμα συστοιχία από **6 pins** με την σήμανση **POWER**.

Η λειτουργία του καθενός έχει ως εξής:

- Το **πρώτο**, με την **ένδειξη RESET**, όταν **γειωθεί** (σε οποιοδήποτε από τα **3 pins** 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 τα εξαρτήματά σας 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**.

- Το **τρίτο**, με την **ένδειξη 5V**, μπορεί να τροφοδοτήσει τα εξαρτήματά σας 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 λειτουργεί στα **5V**), είτε από την **εξωτερική** τροφοδοσία αφού αυτή περάσει από ένα **ρυθμιστή τάσης** για να την «**φέρει**» στα **5V**.

- Το **τέταρτο** και το **πέμπτο pin**, με την **ένδειξη GND**, είναι φυσικά γειώσεις.

- Το **έκτο** και τελευταίο **pin**, με την **ένδειξη Vin** έχει διπλό ρόλο. Σε συνδυασμό με το **pin γείωσης** δίπλα του, μπορεί να λειτουργήσει ως μέθοδος **εξωτερικής τροφοδοσίας** του **Arduino**, στην περίπτωση που δεν σας βολεύει να χρησιμοποιήσετε την υποδοχή του **φις των 2.1mm**. Αν όμως έχετε ήδη συνδεδεμένη εξωτερική τροφοδοσία μέσω του φις, μπορείτε να χρησιμοποιήσετε αυτό το **pin** για

να **τροφοδοτήσετε** εξαρτήματα με την **πλήρη** τάση της **εξωτερικής** τροφοδοσίας (7V~12V), πριν αυτή **περάσει** από τον **ρυθμιστή** τάσης όπως γίνεται με το **pin** των **5V**.

1.7 Ενσωματωμένα κουμπιά και LED

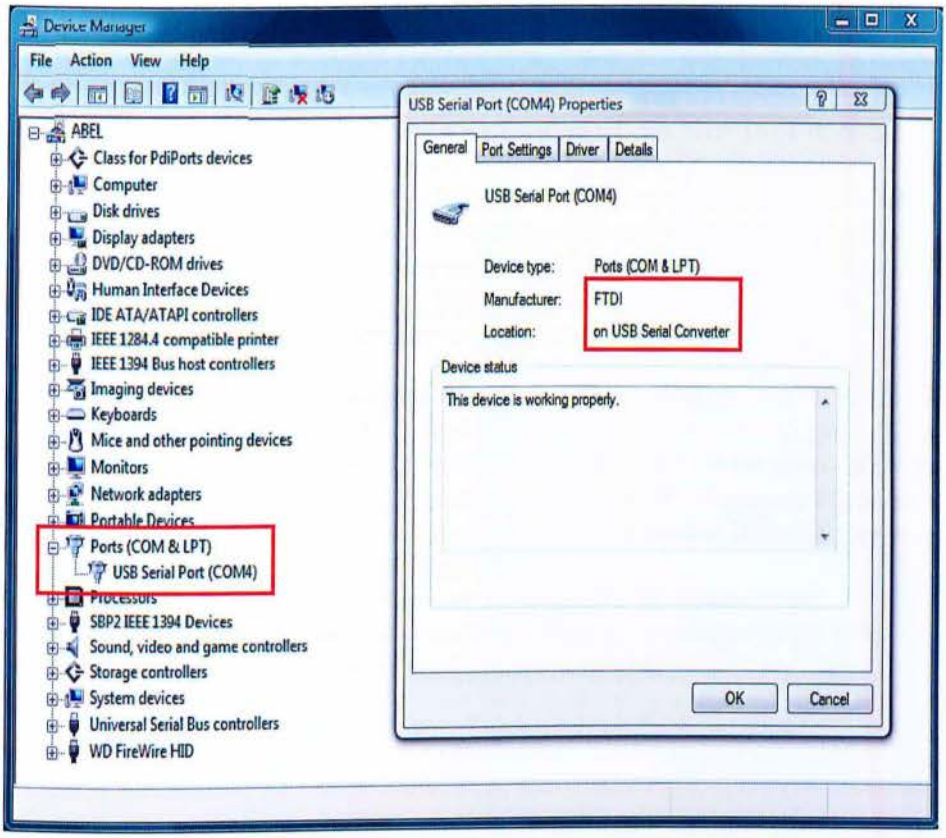
Πάνω στην πλακέτα του **Arduino** υπάρχει ένας **διακόπτης micro-switch** και **4** μικροσκοπικά **LEDs** επιφανειακής στήριξης.
Η λειτουργία του διακόπτη (που έχει την σήμανση **RESET**) και του ενός **LED** με την σήμανση **POWER** είναι μάλλον προφανής.
Τα δύο **LEDs** με τις σημάνσεις **TX** και **RX**, χρησιμοποιούνται ως ένδειξη λειτουργίας του **σειριακού interface**, καθώς **ανάβουν** όταν το **Arduino στέλνει** ή **λαμβάνει** (αντίστοιχα) δεδομένα μέσω **USB**. Σημειώστε ότι τα **LEDs** αυτά ελέγχονται από τον **ελεγκτή Serial-over-USB** και συνεπώς δεν λειτουργούν όταν η **σειριακή** επικοινωνία γίνεται **αποκλειστικά** μέσω των **ψηφιακών pins 0** και **1**.
Τέλος, υπάρχει το **LED** με την σήμανση **L**. Η **βασική δοκιμή λειτουργίας** του **Arduino** είναι να του αναθέσετε να **αναβοσβήνει** ένα **LED** (θα το δείτε αυτό στην συνέχεια όταν θα φτιάξετε την πρώτη εφαρμογή σας). Για να μπορείτε να το κάνετε αυτό από την πρώτη στιγμή, χωρίς να συνδέσετε τίποτα πάνω στο Arduino, οι κατασκευαστές του σκέφτηκαν να **ενσωματώσουν** ένα **LED** στην **πλακέτα**, το οποίο σύνδεσαν στο **ψηφιακό pin 13**. Έτσι, ακόμα και αν δεν έχετε συνδέσει τίποτα πάνω στο φυσικό **pin 13**, αναθέτοντάς του την τιμή **HIGH** μέσα από το πρόγραμμά σας, θα ανάψει αυτό το **ενσωματωμένο LED**.

1.8 Arduino IDE και σύνδεση με τον υπολογιστή

Ότι χρειάζεστε για την διαχείριση του **Arduino** από τον υπολογιστή σας το παρέχει το **Arduino IDE**, την τελευταία έκδοση του οποίου μπορείτε να κατεβάσετε από το επίσημο site για καθένα από τα τρία δημοφιλέστερα λειτουργικά συστήματα.

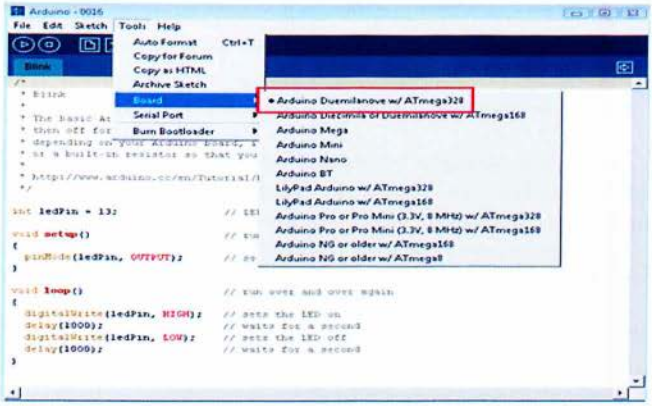
Το **Arduino IDE** είναι βασισμένο σε **Java** και συγκεκριμένα παρέχει:

- ένα πρακτικό περιβάλλον για την συγγραφή των προγραμμάτων σας (τα οποία ονομάζονται **sketchs** στην ορολογία του **Arduino**) με συντακτική χρωματική σήμανση
- αρκετά έτοιμα παραδείγματα
- μερικές **έτοιμες βιβλιοθήκες** για προέκταση της γλώσσας και για να χειρίζεστε εύκολα μέσα από τον κώδικά σας τα εξαρτήματα που συνδέετε στο **Arduino**
- τον **compiler** για την **μεταγλώττιση** των **sketchs** σας
- ένα **serial monitor** που παρακολουθεί τις επικοινωνίες της **σειριακής (USB)**, αναλαμβάνει να **στείλει αλφαριθμητικά** της επιλογής σας στο **Arduino** μέσω αυτής και είναι ιδιαίτερα χρήσιμο για το **debugging** των **sketchs** σας
- και την επιλογή να **"ανεβάσετε"** το **μεταγλωττισμένο sketch** στο **Arduino**



Αν ο οδηγός του ελεγκτή Serial-over-USB είναι σωστά εγκατεστημένος στα Windows, το Arduino θα αναγνωρίζεται από τον Device Manager όπως στην εικόνα. Εκεί μπορείτε να δείτε και τον αριθμό της εικονικής σειριακής θύρας που του ανατέθηκε.

Για τα δύο τελευταία χαρακτηριστικά βέβαια, το **Arduino** πρέπει να έχει **συνδεθεί** σε μια από τις **θύρες USB** του **υπολογιστή** και, λόγω του **ελεγκτή Serial-over-USB**, θα πρέπει να αναγνωριστεί από το λειτουργικό σας σύστημα ως **εικονική σειριακή θύρα**.



Για την σύνδεση θα χρειαστείτε ένα καλώδιο **USB** από **Type A** σε **Type B**, όπως αυτό των εκτυπωτών. Για την αναγνώριση από το λειτουργικό θα χρειαστεί να εγκαταστήσετε τον **οδηγό** του **FTDI chip** (δηλαδή του **ελεγκτή Serial-over-USB**) ο οποίος υπάρχει στον φάκελο **drivers** του **Arduino IDE** που κατεβάσατε. Την τελευταία έκδοση αυτού του οδηγού μπορείτε επίσης να κατεβάσετε για κάθε λειτουργικό σύστημα **από το site της FTDI**. Σημειώστε ότι στους τελευταίους πυρήνες του **Linux** υπάρχει εγγενής υποστήριξη του συγκεκριμένου ελεγκτή.

Αν όλα έγιναν σωστά, το **κεντρικό** παράθυρο του **Arduino IDE** θα εμφανιστεί όταν το εκτελέσετε και στο μενού **Tools** –> **Serial Port** θα πρέπει να εμφανίζεται η **εικονική σειριακή θύρα** (συνήθως **COM#** για τα **Windows**, **/dev/ttyusbserial##** για το **MacOS** και **/dev/ttyusb##** για το **Linux**). Επιλέξτε αυτή την **εικονική θύρα** και στην συνέχεια επιλέξτε τον **τύπο** του **Arduino** σας (π.χ. **Arduino Duemilanove w/ ATmega328**) από το μενού **Tools** –> **Board**.
Το **Arduino** είναι πλέον έτοιμο να δεχτεί τα **sketches** σας. Αν εμφανίστηκε οποιοδήποτε πρόβλημα διαβάστε τις αναλυτικές οδηγίες εγκατάστασης για κάθε λειτουργικό σύστημα στη διεύθυνση **http://arduino.cc/en/Guide/HomePage**.

1.9 Γλώσσα προγραμματισμού

Η **γλώσσα** του **Arduino** βασίζεται στη γλώσσα **Wiring**, μια 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 libc**.
Λόγω της καταγωγής της από την **C**, στην **γλώσσα** του **Arduino** μπορείτε να χρησιμοποιήσετε ουσιαστικά τις ίδιες **βασικές εντολές** και **συναρτήσεις**, με την ίδια σύνταξη, τους ίδιους **τύπων δεδομένων** και τους ίδιους **τελεστές** όπως και στην **C**. Πέρα από αυτές όμως, υπάρχουν κάποιες **ειδικές εντολές**, **συναρτήσεις** και **σταθερές** που βοηθούν για την διαχείριση του **ειδικού hardware** του **Arduino**.

Οι πιο σημαντικές από αυτές επεξηγούνται στον πίνακα που ακολουθεί:

Όρισμα	Είδος	Τύπος	Παράμετροι	Περιγραφή
LOW	Σταθερά	int	-	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
HIGH	Σταθερά	int	-	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
INPUT	Σταθερά	int	-	Έχει την τιμή 0 και είναι αντίστοιχη του λογικού false.
OUTPUT	Σταθερά	int	-	Έχει την τιμή 1 και είναι αντίστοιχη του λογικού true.
pinMode	Εντολή	-	(*pin, mode*)	Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode* (INPUT ή OUTPUT αντίστοιχα).
digitalWrite	Εντολή	-	(*pin, pinstatus*)	Θέτει την κατάσταση *pinstatus* (HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*.
digitalRead	Συνάρτηση	int	(*pin*)	Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου.
analogReference	Εντολή	-	(*type*)	Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο *type* για να καθορίσει την τάση αναφοράς (V_{ref}) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα)
analogRead	Συνάρτηση	int	(*pin*)	Επιστρέφει έναν ακέραιο από 0 έως 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο *pin* αναλογικής εισόδου στην κλίμακα 0 ως V_{ref}.

analogWrite	Εντολή	-	(*pin*, *value*)	Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με *value* 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο).
millis	Συνάρτηση	unsigned long	()	Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν.
delay	Εντολή	-	(*time*)	Σταματά προσωρινά την ροή του προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). Σημειώστε ότι παρά την προσωρινή παύση, συναρτήσεις των οποίων η εκτέλεση ενεργοποιείται από interrupt θα εκτελεστούν κανονικά κατά την διάρκεια μιας delay.
attachInterrupt	Εντολή	-	(*interrupt*, *function*, *triggermode*)	Θέτει σε λειτουργία το συγκεκριμένο *interrupt*, ώστε να ενεργοποιεί την συνάρτηση *function*, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο *triggermode*: • LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW) • RISING (όταν από LOW γίνει HIGH) • FALLING (όταν από HIGH γίνει LOW) • CHANGE (όταν αλλάξει κατάσταση γενικά)
detachInterrupt	Εντολή	-	(*interrupt*)	Απενεργοποιεί το συγκεκριμένο *interrupt*.
noInterrupts	Εντολή	-	()	Σταματά προσωρινά την λειτουργία όλων των interrupt
interrupts	Εντολή	-	()	Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts.
Serial.begin	Μέθοδος κλάσης	-	(*datarate*)	Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud)
Serial.println	Μέθοδος κλάσης	-	(*data*)	Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data* μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό.

Επιπλέον, στην **γλώσσα** του **Arduino** κάθε πρόγραμμα αποτελείται από **δύο βασικές ρουτίνες** ώστε να έχει την γενική δομή:

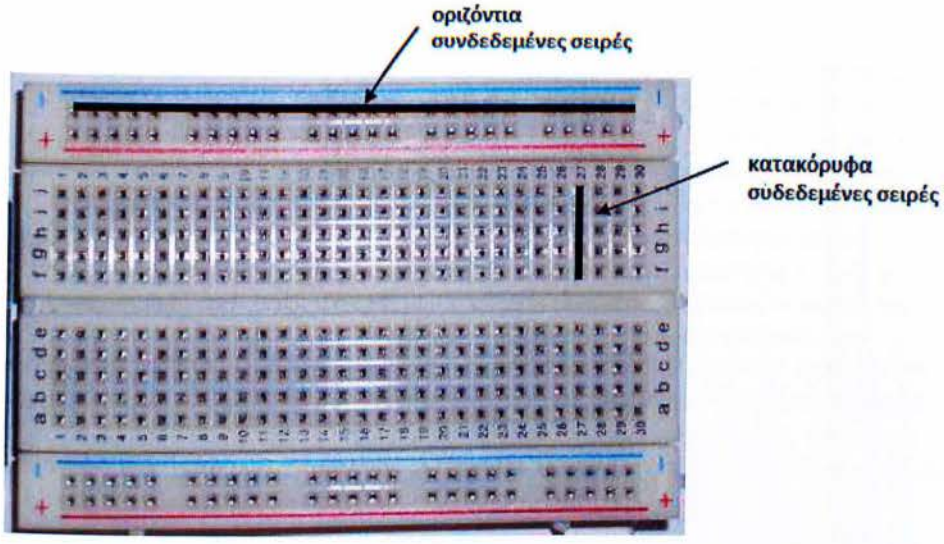
```
// Ενσωματώσεις βιβλιοθηκών, δηλώσεις μεταβλητών...

void setup()
{
  //...
}

void loop()
{
  // ...
}

// Υπόλοιπες συναρτήσεις...
```

Η **βασική ρουτίνα setup()** εκτελείται **μια** φορά **μόνο** κατά την **εκκίνηση** του προγράμματος ενώ η **βασική ρουτίνα loop()** περιέχει τον **βασικό** κορμό του προγράμματος και η εκτέλεσή της **επαναλαμβάνεται συνέχεια** σαν ένας **βρόγχος while (true)**.
Αν και πρόκειται μόνο για τις πιο βασικές λειτουργίες της γλώσσας του **Arduino**, με αυτές και με λίγες βασικές γνώσεις **C** θα μπορέσετε να δημιουργήσετε το **sketch** ακόμα και για κάποιο **αρκετά περίπλοκο project**.

Για το πλήρες **reference** πάντως, επισκεφτείτε **την σχετική σελίδα** ενώ ακόμα περισσότερες πληροφορίες μπορείτε να βρείτε **στο site της Wiring** καθώς και **στο εγχειρίδιο της βιβλιοθήκης AVR Libc**.

1.10 Πλακέτα τύπου Solderless

Μία πλακέτα τύπου Solderless, δηλαδή πλακέτα στην οποία τα ηλεκτρονικά στοιχεία δεν χρειάζονται ηλεκτρικές κολλήσεις μεταξύ τους, είναι ένα απαραίτητο εργαλείο για την ταχύ προτυποποίηση ηλεκτρονικών κυκλωμάτων. Στοιχεία και σύρμα ωθούνται μέσα σε τρύπες. Γραμμές και στήλες από τρύπες είναι εσωτερικά συνδεδεμένα για να κάνουν τις συνδέσεις ευκολότερες. Σύρματα συνδέονται από την πλακέτα στις εισόδους/εξόδους ακροδεκτών στην μητρική πλακέτα του Arduino. Οι συνδέσεις καλό είναι να γίνονται χρησιμοποιώντας μικρού μήκους συμπαγές σύρμα 22 g με εξοχή από το πλαστικό μόνωσης περίπου 0.25 cm στο κάθε άκρο. Παρακάτω υπάρχει μια εικόνα που δείχνει ποιες σειρές είναι συνδεδεμένες εσωτερικά. Τα ζεύγη των οριζόντιων σειρών στην κορυφή και στο τέλος είναι χρήσιμα για να συνδέσουμε πηγές ρεύματος και γειώσεις. Ως σύμβαση έχουμε τις κόκκινες σειρές για τροφοδοσία 5V και τις μπλε για την γείωση των ηλεκτρονικών στοιχείων. Οι σειρές τροφοδοσίας μερικές φορές ονομάζονται «δίαυλους ενέργειας».



Προσοχή: Χρησιμοποιούμε μόνο συμπαγές σύρμα στην πλακέτα. Νηματοειδή σύρμα μπορεί να διαχωριστεί και να εισχωρήσει σε τρύπες της πλακέτας μόνιμα.

1.11 Hello World!

Έφτασε η στιγμή να δημιουργήσετε το **πρώτο** σας **sketch**, το οποίο -παραδοσιακά- πρέπει να εξάγει το μήνυμα «**Hello World**». Βέβαια μέχρι να προσθέσετε εσείς μια, το **Arduino** δεν διαθέτει **οθόνη** ώστε να εμφανίσει κάποιο μήνυμα. Η μόνη συσκευή εξόδου που είναι ενσωματωμένη στην πλακέτα του **Arduino** είναι το **LED** του **pin 13**. Έτσι, το **Arduino** σας θα χαιρετίσει την οικουμένη **αναβοσβήνοντας** απλά το **LED** του.
Ανοίξτε το **IDE** του **Arduino** και -για να μην πληκτρολογείτε- επιλέξτε **File** –> **Sketchbook** –> **Examples** –> **Digital** –> **Blink**. Θα ανοίξει ένα **sketch** με τον παρακάτω κώδικα:

```
1  int ledPin = 13;
2  void setup()
3  {
4    pinMode(ledPin, OUTPUT);
5  }
6  void loop()
7  {
8    digitalWrite(ledPin, HIGH);
9    delay(1000);
10   digitalWrite(ledPin, LOW);
11   delay(1000);
12 }
```

Όπως κάθε "Hello World" πρόγραμμα, το sketch αυτό είναι αρκετά σαφές.

Αρχικά, στην ρουτίνα setup() ρυθμίζεται το pin στο οποίο είναι συνδεδεμένο το LED ως pin εξόδου (γραμμή 4) . Στην συνέχεια η κύρια ρουτίνα loop(), η εκτέλεση της οποίας επαναλαμβάνεται συνέχεια, ανάβει το LED (γραμμή 8) και στην συνέχεια το σβήνει (γραμμή 10). Δύο εντολές delay ρυθμίζουν τον χρόνο που το LED θα μένει αναμμένο ή σβηστό στις γραμμές 9 και 11 (1000ms δηλαδή 1 δευτερόλεπτο). Για να δείτε το πρόγραμμα στην πράξη, εφόσον έχετε ήδη συνδέσει το Arduino με τον υπολογιστή επιλέξτε File –> Upload to I/O Board (εναλλακτικά πατήστε Ctrl-U ή κάντε κλικ στο ανάλογο εικονίδιο της toolbar). Με αυτή την ενέργεια, το sketch θα μεταγλωττιστεί και θα σταλεί αυτόματα στο Arduino, γεγονός που μπορείτε να επαληθεύσετε από την δραστηριότητα των TX και RX LEDs πάνω στην πλακέτα του Arduino. Τα προγράμματα που «ανεβάζετε» στο Arduino εκτελούνται αυτόματα από τον bootloader αμέσως μετά την λήψη τους και έτσι, χωρίς καθυστέρηση, θα πρέπει να δείτε το LED με τη σήμανση 13 να ανάβει και να σβήνει συνεχόμενα με περίοδο 2 δευτερολέπτων, δηλαδή όπως ακριβώς ορίζει το sketch.
Αν επιμένετε ότι ένα LED που αναβοσβήνει δεν αποτελεί πρέπον χαιρετισμό και θέλετε σώνει και καλά να δείτε το "Hello World" γραμμένη, υπάρχει μια λύση. Μπορείτε να το στείλετε μέσω της σειριακής (USB) στον υπολογιστή και να το δείτε στην οθόνη σας. Και σαν bonus, το Arduino θα στέλνει και την κατάσταση του LED στον υπολογιστή . Προσθέστε απλά τις γραμμές:

```
Serial.begin(9600);
Serial.println("Hello World! - Are you happy now?");
```

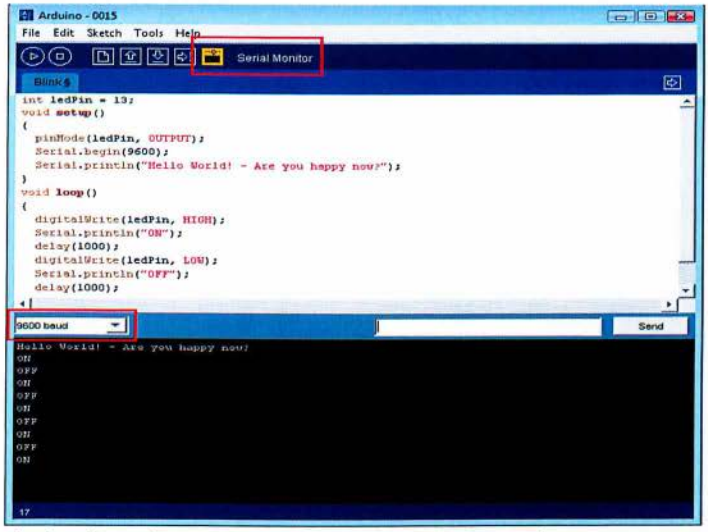
αμέσως μετά την γραμμή με την εντολή pinMode και πριν κλείσει το άγκιστρο της ρουτίνας setup(). Επίσης, προσθέστε την γραμμή:

Serial.println("ON");

και την γραμμή:

Serial.println("OFF");

αμέσως μετά την πρώτη και την δεύτερη digitalWrite αντίστοιχα.



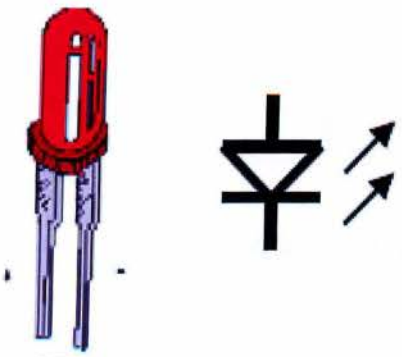
Κάπως έτσι μοιάζει το παράθυρο του Arduino IDE αφού μετατραπεί σε σειριακή κονσόλα. Αν κατά την εκτέλεση του ενημερωμένου «Hello World» λαμβάνετε ακαταλαβίστικους χαρακτήρες, βεβαιωθείτε ότι έχετε ρυθμίσει την κονσόλα στα 9600 baud, όπως φαίνεται στην εικόνα.

Αφού κάνουμε τις αλλαγές επιλέγουμε όπως και πριν το Upload to I/O Board από το IDE για να γίνει ξανά μεταγλώττιση και να ανέβει το νέο binary στο Arduino. Αμέσως μετά, κάντε κλικ στο τελευταίο κουμπί της toolbar με επεξήγηση Serial Monitor για να μετατρέψετε το κάτω τμήμα του παραθύρου του IDE σε σειριακή κονσόλα και σύντομα θα δούμε το Arduino να μας στέλνει τα μηνύματά του.

Μπορούμε να πειραματιστούμε με το sketch, να φτιάξετε ωραία pattern με τα οποία να αναβοσβήνει το LED, να το βάλετε να στέλνει διαφορετικές πληροφορίες στην σειριακή κ.λπ.

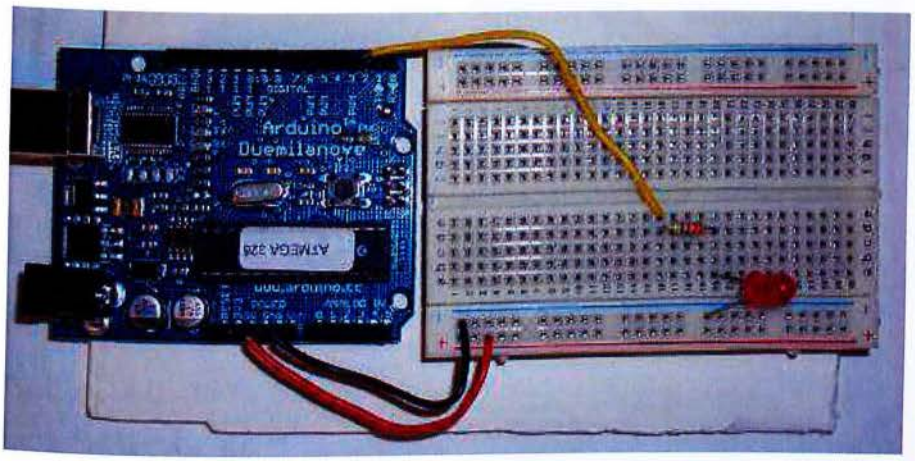
1.12 Αναβοσβήνοντας ένα LED

Μια δίοδος εμπέμποντος φωτός (Light Emitting Diode) είναι χρήσιμη για να ελέγχει τι μπορεί να κάνει ένα Arduino. Για αυτό τον σκοπό, θα χρειαστούμε ένα LED, μία αντίσταση 330 ohm και μερικά μικρού μήκους σύρματα. Η παρακάτω εικόνα είναι ένα σχέδιο ενός LED και το σύμβολό της που χρησιμοποιείται σε ηλεκτρονικά σχέδια.

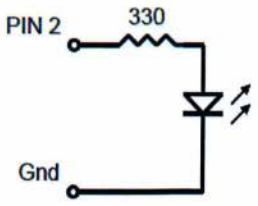


LED

Χρησιμοποιώντας συμπαγές σύρμα, συνδέουμε τον ακροδέκτη των 5V τροφοδοσίας από το Arduino στο κάτω κόκκινο δίαυλο τροφοδοσίας της πλακέτας και τον ακροδέκτη γειώσεως από το Arduino στο κάτω μπλε δίαυλο τροφοδοσίας της πλακέτας. Συνδέουμε το οδοντωτό ή επίπεδο μέρος του LED (το οδοντωτό ή επίπεδο είναι στο άκρο που περιβάλλει την βάση LED, κοιτάξτε προσεκτικά γιατί μπορεί να είναι δύσκολο να βρεθεί) στο δίαυλο γειώσεως και το άλλο άκρο σε μία ελεύθερη τρύπα στο κυρίως τμήμα της πλακέτας. Τοποθετούμε την αντίσταση με τέτοιο τρόπο ώστε το ένα άκρο της να είναι στην ίδια στήλη με το LED και το άλλο άκρο σε μία ελεύθερη στήλη. Από αυτή την στήλη, συνδέουμε ένα σύρμα στην ψηφιακή ακίδα 2 της μητρικής πλακέτας του Arduino. Το στήσιμο μας θα είναι το παρακάτω.



Για να ελέγξουμε εάν το LED λειτουργεί, προσωρινά αποσυνδέουμε το σύρμα από τον ακροδέκτη 2 του Arduino board και το συνδέουμε στο 5V δίαυλο τροφοδοσίας. Το LED θα πρέπει να ανάψει. Εάν όχι, δοκιμάζουμε να αλλάξουμε τον προσανατολισμό του LED. Τοποθετούμε το σύρμα πάλι στον ακροδέκτη 2. Στο LED, το ρεύμα τρέχει από την άνοδο (+) στην κάθοδο (-) όπου είναι σημαδεμένο από την εγκοπή. Το κύκλωμα το οποίο δημιουργήσαμε, παρουσιάζεται στο παρακάτω σχέδιο.



Δημιουργούμε και τρέχουμε το παρακάτω Arduino πρόγραμμα:

```
Void setup ( )

{

   pinMode (2, OUTPUT ) ;

   digitalWrite (2, HICH ) ;

   delay ( 1000 );

   digitalWrite(2, LOW ) ;

}

void loop ( )

{}
```

Άναψε το LED για ένα δευτερόλεπτο; Πιέζουμε το ενσωματωμένο κουμπί reset του Arduino για να τρέξουμε το πρόγραμμα ξανά. Τώρα δοκιμάζουμε αυτό το πρόγραμμα, το οποίο θα ανάψει το LED με συχνότητα 1Hz. Όλα μέτα το σύμβολο // στη γραμμή είναι σχόλια, όπως επίσης και το κείμενο μεταξύ /* και */ στη αρχή. Είναι πάντα χρήσιμο και ωφέλιμο να προσθέτουμε σχόλια σε ένα πρόγραμμα.

```
/*--------------------------------

Blinking LED, 1Hz on pin 2

--------------------------------*/

void setup ( )    // μιας φοράς εκτέλεση

{

  pinMode ( 2, OUTPUT ) ;   // ορίζουμε τον ακροδέκτη 2 ως έξοδο
```

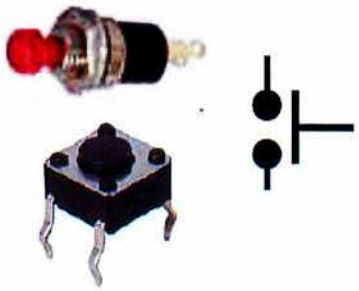
```
}

void loop ( )   // ατέρμονος βρόγχος εκτέλεσης

{

digitalWrite ( 2, HIGH ) ;  // ακροδέκτης 2 σε κατάσταση HIGH ( LED on )

delay ( 500 ) ;     // αναμονή 500 ms σε κατάσταστη on

digitalWrite ( 2, LOW ) ;  // ακροδέκτης 2 σε κατάσταση LOW (LED off )

delay ( 500 );  // αναμονή 500 ms σε κατάσταστη off

}
```

Η εντολή pinMode θέτει το LED ακροδέκτη 2 σαν έξοδο. Η πρώτη digitalWrite εντολή λέει στο Arduino να θέσει τον ακροδέκτη 2 του σε κατάσταση HIGH,ή 5V. Αυτό στέλνει ρεύμα από τον ακροδέκτη 2, μέσω της αντίστασης, μέσω του LED στη γείωση. Η delay (500) εντολή περιμένει για 500 microseconds. Η δεύτερη digitalWrite εντολή θέτει τον ακροδέκτη 2 σε κατάσταση LOW ή 0V διακόπτωντας την παροχή ρεύματος και ως εκ τούτου να σβήνει το LED. Ο κώδικας εντός των παρενθέσεων ορίζει την loop ( ) συνάρτηση και επαναλαμβάνεται συνεχώς, και γιαυτό το LED αναβοσβήνει συνεχώς.

Αυτή η άσκηση επιδεικνύει πως το Arduino μπορεί να ελέγξει τον έξω κόσμο. Με την σωστή διεπαφή κυκλώματος, ο ίδιος κώδικας μπορεί να εκκινήσει και να σταματήσει έναν κινητήρα, ρελέ, ηλεκτρομαγνήτες, πνευματικές βαλβίδες ή οποιαδήποτε άλλη συσκευή on/off κατάστασης.

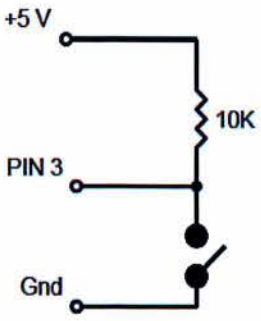
1.13 Διαβάζοντας ένα Διακόπτη

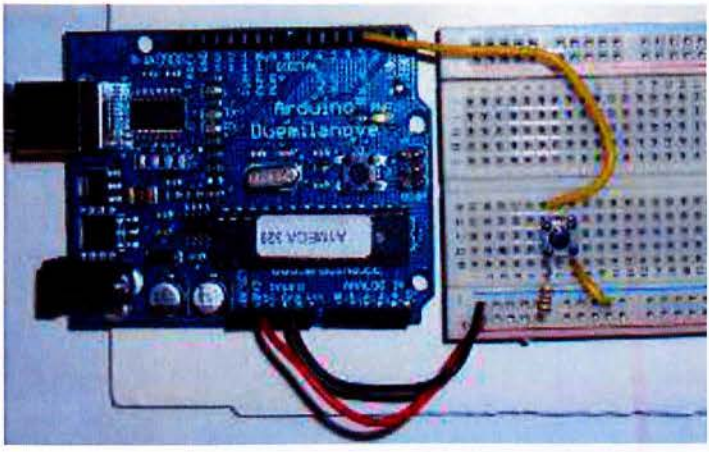
Η άσκηση αυτή επιδεικνύει πώς το Arduino μπορεί να ελέγξει επίσης τον εξωτερικό κόσμο. Πολλές εφαρμογές απαιτούν την ανάγνωση της κατάστασης ενός αισθητήρα, συμπεριλαμβανομένου και την κατάσταση ενός διακόπτη. Η εικόνα παρακάτω δείχνει έναν τύπου μπουτόν διακόπτη και το σχηματικό σύμβολό του.



Σημειώνεται ότι το σύμβολο αυτό αντιπροσωπεύει ένα διακόπτη του οποίου οι επαφές είναι κανονικά ανοιχτές, αλλά όταν πατηθεί οι επαφές του βραχυκυκλώνονται.

Σε αυτή την άσκηση, το Arduino θα διαβάζει την κατάσταση ενός κανονικά-ανοιχτού μπουτόν και θα εμφανίζει τα αποτελέσματα στον υπολογιστή κάνοντας χρήση της εντολής serial.println( ). Θα χρειαστούμε ένα διακόπτη, μία αντίσταση 10 KOhm και μερικά κομμάτια συρμάτινων καλωδίων. Εάν δεν διαθέτουμε διακόπτη, το αντικαθιστούμε με δύο καλώδια και ενώνουμε τα ελεύθερα άκρα τους για να δημιουργήσουμε ένα κλειστό διακόπτη. Η εικόνα δείχνει παρακάτω το σχηματικό για το κύκλωμα στα αριστερά.





Δημιουργούμε και τρέχουμε το παρακάτω Arduino πρόγραμμα

```
void setup ( )

{

    Serial.begin ( 9600 ) ;

}
```

```
void loop ( )
{
    Serial.println ( digitalRead ( 3 ) ) ;
    delay ( 250 ) ;
}
```

Ανοίγουμε το παράθυρο της Σειριακή Οθόνης. Όταν το διακόπτης είναι ανοιχτός, θα πρέπει να δούμε μία σειρά από άσσους (1) στην οθόνη. Όταν κλείσει, η σειρά των άσσων (1) αλλάζει σε σειρά από μηδενικά (0). Από πλευρά υλικού (hardware), όταν ο διακόπτης είναι ανοιχτός, δεν διαρρέει ρεύμα διαμέσου της αντίστασης. Όταν δεν διαρρέει ρεύμα μέσω μίας αντίστασης, δεν υπάρχει πτώση τάσης διαμέσου της αντίστασης, το οποίο σημαίνει ότι η τάση σε κάθε άκρο είναι ίδια. Στο κύκλωμά μας, όταν ο διακόπτης είναι ανοιχτός, ο ακροδέκτης 3 διαρρέεται από τάση 5V το οποίο ο υπολογιστής το διαβάζει ως μία κατάσταση 1. Όταν ο διακόπτης είναι κλειστός, ο ακροδέκτης 3 είναι απευθείας συνδεδεμένος με την γείωση, το οποίο είναι στα 0V. Ο υπολογιστής το διαβάζει αυτό ως μία κατάσταση 0.

Τώρα ας προσπαθήσουμε να δοκιμάσουμε το παρακάτω πρόγραμμα, το οποίο αποτελεί ένα παράδειγμα πως μπορούμε να κάνουμε τον υπολογιστή να περιμένει έναν αισθητήρα να αλλάξει κατάσταση.

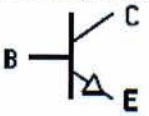
```
void setup ( )
{
   Serial.begin ( 9600 )
}
void loop ( )
{
  while (digitalRead (3) = = HIGH) ;
  Serial.println ("Somebody closed the switch!") ;
  while (digitalRead(3)==LOW) ;
  Serial.println ("The switch is now open!") ;
}
```

Προσέχουμε την δραστηριότητα στο παράθυρο της Σειριακής Οθόνης καθώς πιέζουμε και απελευθερώνουμε τον διακόπτη.

1.14 Ελέγχοντας έναν μικρό DC κινητήρα

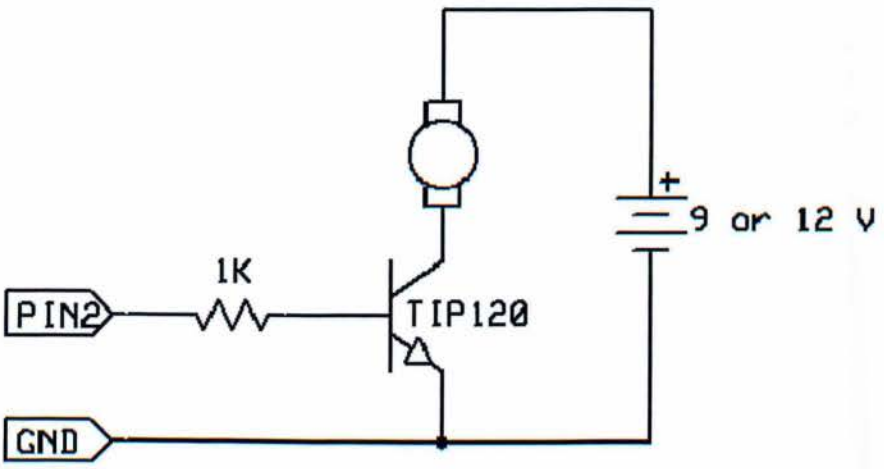
Το Arduino μπορεί να χειριστεί έναν μικρό DC κινητήρα μέσω ενός τρανζίστορ διακόπτη. Θα χρειαστούμε ένα TIP120 τρανζίστορ, μία αντίσταση 1K, μία μπαταρία 9V και ένα κινητήρα.

Οι ακροδέκτες του TIP120 είναι αυτοί και στο σχηματικό οι ακροδέκτες είναι

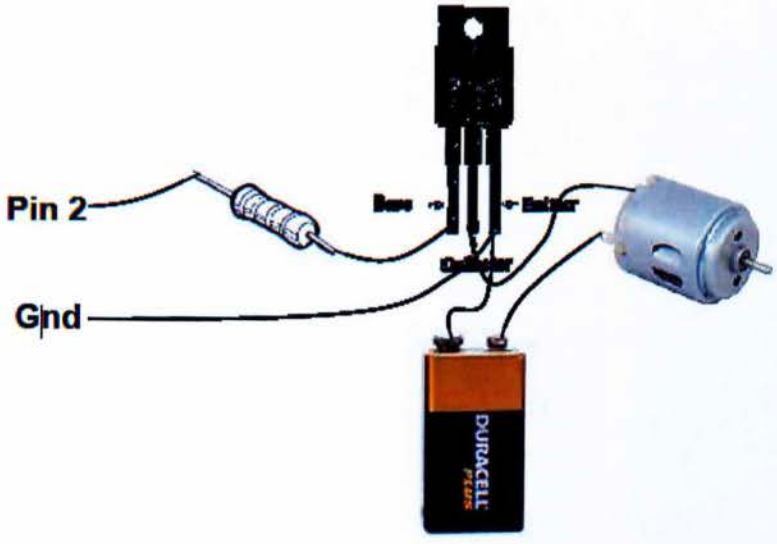


.

Παρακάτω είναι το σχηματικό διάγραμμα για το πώς θα συνδέσουμε τον κινητήρα.



Και παρακάτω είναι ένα εικονογραφημένο διάγραμμα για το πώς θα συνδέσουμε τα στοιχεία. Οι συνδέσεις μπορούν να γίνουν με κολλήσεις ή μπορούν να γίνουν δίχως κολλήσεις (soldered) με την χρήση ενός solderless breadboard.



Ο ακροδέκτης 2 μπορεί να είναι οποιοσδήποτε από τους ψηφιακούς I/O ακροδέκτες του Arduino. Συνδέουμε το πλην της μπαταρίας στον εκπομπό του τρανζίστορ (E pin) και επίσης συνδέουμε τον εκπομπό του τρανζίστορ στη γείωση (Gnd) της Arduino πλακέτας.

Για να ελέγξουμε εάν όλα λειτουργούν, παίρνουμε έναν βραχυκυκλωτήρα και βραχυκυκλώνουμε τον συλλέκτη (C) στους ακροδέκτες του εκπομπού (E). Ο κινητήρας θα πρέπει να εκκινήσει την λειτουργία του. Στη συνέχεια, αποσυνδέουμε την αντίσταση 1K από τον ακροδέκτη 2 και βραχυκλώνουμε με 5V. Ο κινητήρας θα πάρει μπρος. Τοποθετούμε την αντίσταση πάλι στον ακροδέκτη 2 και τρέχουμε το επόμενο πρόγραμμα:

```
Void setup ( )

{

   pinMode ( 2, OUTPUT ) ;

   digitalWrite ( 2, HIGH ) ;

   delay ( 1000 ) ;

}

Void loop ( )

{}
```

Ο κινητήρας θα πρέπει να εκκινήσει για 1 δευτερόλεπτο και μετά να σταματήσει την λειτουργία του.

Επίσης για να κάνουμε τον κινήτηρα να τρέχει και να σταματάει κάθε ένα δευτερόλεπτο τρέχουμε τον παρακάτω κώδικα:

```
void setup ( )

{

  pinMode ( 2, OUTPUT ) ;

}

void loop ( )

{

digitalWrite ( 2, HIGH ) ;

delay ( 1000 ) ;

digitalWrite ( 2, LOW ) ;
```

```
delay ( 1000 ) ;

}
```

1.15 Εξφαλμάτωση

Εάν υπάρχει κάποιο συντακτικό λάθος στο πρόγραμμα κατά την πληκτρολόγηση, ένα μήνυμα λάθους θα εμφανιστεί στο κάτω μέρος του παραθύρου. Γενικά, παρατηρώντας το σφάλμα θα αποκαλύψει το πρόβλημα. Εάν συνεχίσουμε να αντιμετωπίζουμε προβλήματα, μπορούμε να δοκιμάσουμε τις εξής λύσεις:

- Επανεκκίνηση του Arduino προγράμματος
- Έλεγχος του USB καλωδίου, εξασφαλίζοντας ότι οι δύο απολήξεις του καλωδίου είναι σωστά κουμπωμένες στη θύρα USB του υπολογιστή και του Arduino
- Επανεκκίνηση του υπολογιστή γιατί μερικές φορές η σειριακή θύρα μπορεί έχει μπλοκάρει
- Εάν η σειριακή θύρα είναι ήδη σε χρήση, μήνυμα σφάλματος εμφανίζεται κατά την μεταφόρτωση του προγράμματος στο Arduino

1.16 Βιβλιοθήκες

Οι βιβλιοθήκες είναι μια συλλογή κώδικα που κάνει διαδικασίες όπως την σύνδεση ενός σένσορα, μια οθόνη και άλλα εξαρτήματα ευκολότερες. Οι βιβλιοθήκες (libraries) παρέχουν επιπλέον λειτουργικότητα για χρήση στα προγράμματά μας (sketches), π.χ. δουλεύοντας με υλικό (hardware) ή διαχειρίζοντας δεδομένα. Για παράδειγμα, μια βιβλιοθήκη Liquid Crystal κάνει ευκολότερο τον χειρισμό μίας LCD οθόνης όταν την συνδέουμε με ένα Arduino. Για να χρησιμοποιήσουμε μια βιβλιοθήκη σε ένα sketch, ακολουθούμε τα βήματα, επιλέγουμε από το μενού Sketch --> Import Library. Με αυτό τον τρόπο θα εισάγουμε μία ή περισσότερες #include δηλώσεις στη αρχή του sketch μας και θα μεταγλωττιστεί μαζί με το sketch. Επειδή οι βιβλιοθήκες μεταφορτώνονται στην πλακέτα με το sketch, αυξάνουν τον χώρο σε bytes που καταλαμβάνουν. Εάν ένα sketch δεν χρειάζεται περαιτέρω μια βιβλιοθήκη, απλά διαγράφουμε την #include δηλώσεις στην αρχή του κώδικά μας.

Υπάρχει μία λίστα από βιβλιοθήκες. Μερικές βιβλιοθήκες εμπεριέχονται με το λογισμικό του Arduino. Άλλες μπορούν να βρεθούν μέσα από ένα μεγάλο εύρος πηγών στο διαδίκτυο.

1.17 Κατηγορίες μητρικών πλακετών Arduino

Η επιλογή της πλακέτας έχει δύο αποτελέσματα: θέτει τις παραμέτρους (π.χ. την ταχύτητα του επεξεργαστή και το baud rate) που χρησιμοποιούνται κατά την μεταγλώττιση και την μεταφόρτωση των προγραμμάτων μας (sketches), και τέλος θέτει το αρχείο και τις ρυθμίσεις συγχώνευσης που χρησιμοποιούνται από τον bootloader. Μερικοί ορισμοί που χρησιμοποιούν οι διάφορες πλακέτες διαφέρουν μόνο στο τέλος, επομένως ακόμα και αν έχουμε μεταφορτώσει πετυχημένα το sketch μας με μια συγκεκριμένη επιλογή, θα ήταν καλό να την ελέγξουμε πριν γράψουμε το sketch στον bootloader.

- Arduino Uno

Microcontroller	ATmega328
Operating	Voltage 5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 used by bootloader
SRAM	2KB (ATmega328)
EEPROM	1KB (ATmega328)
Clock Speed	16MHz

Arduino Uno Front Arduino Uno Back

- Arduino Leonardo:

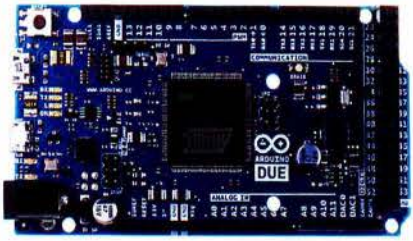
Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
Analog Input Pins	12
PWM Channels	7
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 4KB used by bootloader
SRAM	2.5KB (ATmega328)
EEPROM	1KB (ATmega328)
Clock Speed	16MHz

Arduino Leonardo Front

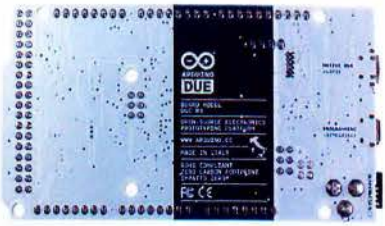
Arduino Leonardo Back

❖ Arduino Due:

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 5V Pin	800 mA
DC Current for 3.3V Pin	800 mA
Flash Memory	512KB all available for applications
SRAM	96KB (two banks:64KB & 32KB)
Clock Speed	84MHz



Arduino Due Front



Arduino Due Back

❖ Arduino Robot:

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage	5V through flat cable
Digital I/O Pins	5
PWM Channels	6
Analog Input Channels	4 (of the Digital I/O pins)
Analog Input Channels (multiplexed)	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM (internal)	1 KB (ATmega32u4)
EEPROM (external)	512 Kbit (I2C)
Clock Speed	16 MHz
Keypad	5 keys
Knob	potentiomenter attached to analog pin
Full color LCD	over SPI communication
SD card reader	for FAT16 formatted cards
Speaker	8 Ohm
Digital Compass	provides deviation from the geographical north in degrees
I2C soldering ports	3

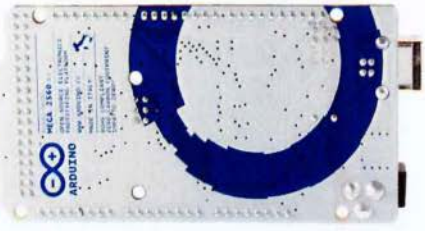
Arduino Robot Front

Arduino Robot Back

❖ Arduino Mega 2560:

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Arduino Mega 2560 Front



Arduino Mega 2560 Back

❖ Arduino Mega ADK

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

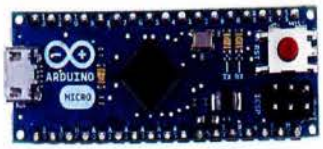
Arduino Mega ADK Front

Arduino Mega ADK Back

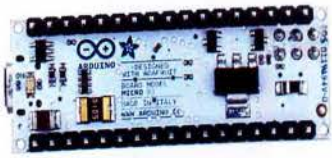
❖ Arduino Micro:

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by

	bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz



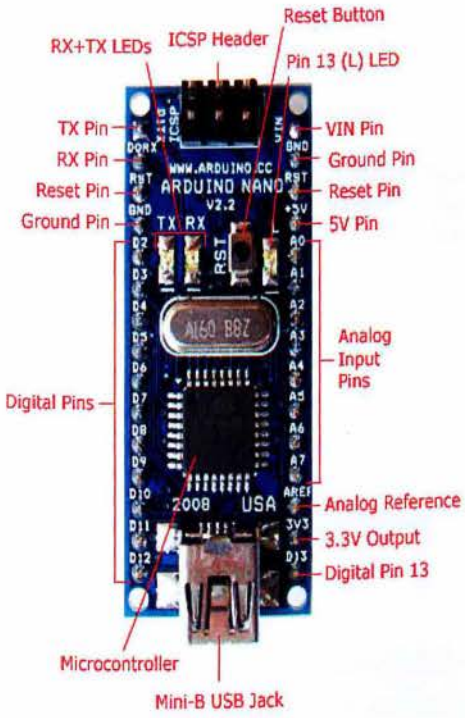
Arduino Micro Front



Arduino Micro Back\

- Arduino Nano

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"

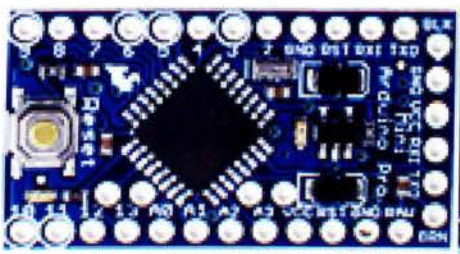


Arduino Nano Front

Arduino Nano Back

❖ Arduino Pro Mini

Microcontroller	ATmega168
Operating Voltage	3.3V or 5V (depending on model)
Input Voltage	3.35 -12 V (3.3V model) or 5 - 12 V (5V model)
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (of which 2 KB used by bootloader)
SRAM	1 KB
EEPROM	512 bytes
Clock Speed	8 MHz (3.3V model) or 16 MHz (5V model)

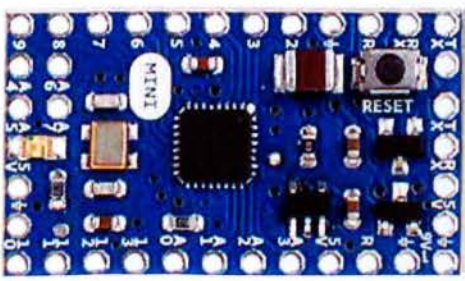


Arduino Mini Front

Arduino Mini Back

❖ Arduino Mini

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	7-9 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8 (of which 4 are broken out onto pins)
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

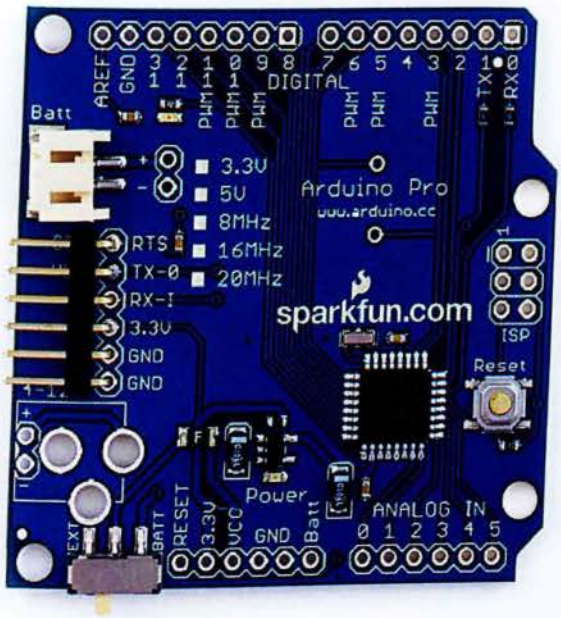


Arduino Mini Front

Arduino Mini Back

❖ Arduino Pro

Microcontroller	ATmega168 or ATmega328
Operating Voltage	3.3V or 5V
Input Voltage	3.35 -12 V (3.3V versions) or 5 - 12 V (5V versions)
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	8 MHz (3.3V versions) or 16 MHz (5V versions)

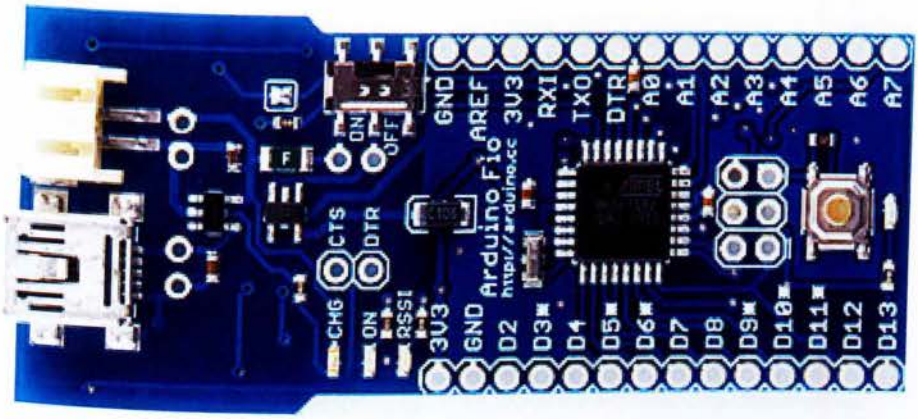


Arduino Pro

❖ Arduino Fio

Microcontroller	ATmega328P
Operating Voltage	3.3V

Input Voltage	3.35 -12 V
Input Voltage for Charge	3.7 - 7 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	8 MHz



Arduino Fio

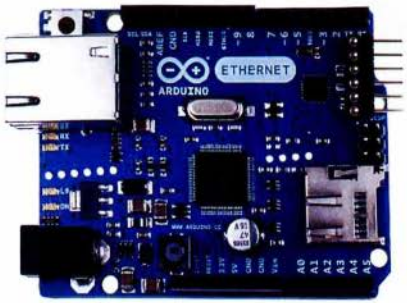
- Arduino Ethernet

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage Plug (recommended)	7-12V
Input Voltage Plug (limits)	6-20V
Input Voltage PoE (limits)	36-57V
Digital I/O Pins	14 (of which 4 provide PWM output)

Arduino Pins reserved:

10 to 13 used for SPI
4 used for SD card
2 W5100 interrupt (when bridged)

Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
W5100 TCP/IP Embedded Ethernet Controller	
Power Over Ethernet ready Magnetic Jack	
Micro SD card, with active voltage translators	



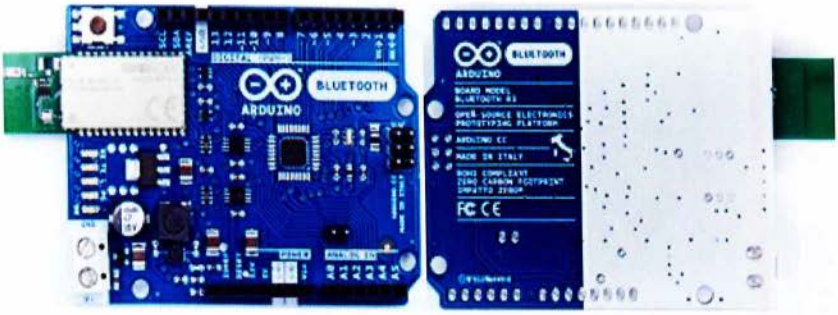
Arduino Ethernet Front

Arduino Ethernet Back

- Arduino BT

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	2.5-12 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	500 mA (with a 1.5A capable power source)

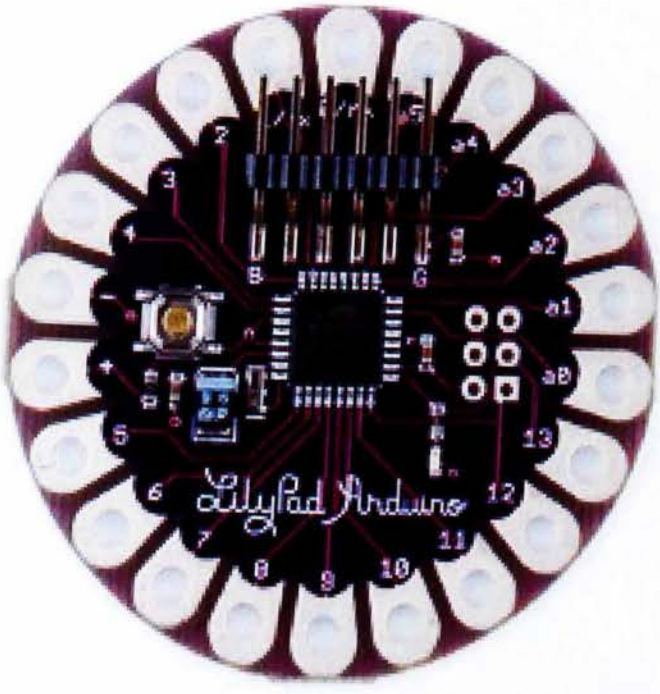
DC Current for 5V Pin	1000 mA (with a 1.5A capable power source)
Flash Memory	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
BT Module	2.1 WT11i-A-AI4



Arduino BT Front Arduino BT Back

- LilyPad Arduino

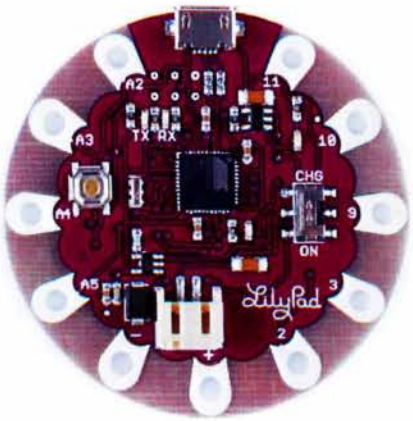
Microcontroller	ATmega168V or ATmega328V
Operating Voltage	2.7-5.5 V
Input Voltage	2.7-5.5 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (of which 2 KB used by bootloader)
SRAM	1 KB
EEPROM	512 bytes
Clock Speed	8 MHz



Lilypad Arduino

- Lilypad Arduino USB

Microcontroller	ATmega32u4
Operating Voltage	3.3V
Input Voltage	3.8V to 5V
Digital I/O Pins	9
PWM Channels	4
Analog Input Channels	4
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	8 MHz

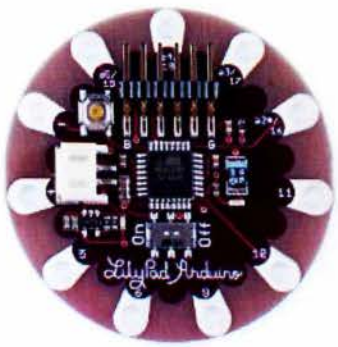


Lilypad Arduino USB Front

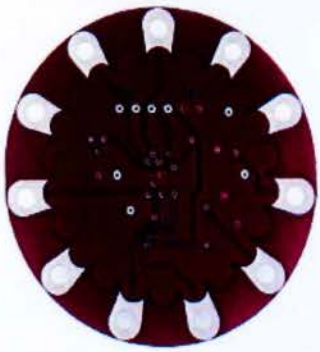
Lilypad Arduino USB Back

- Lilypad Arduino Simple

Microcontroller	ATmega328
Operating Voltage	2.7-5.5 V
Input Voltage	2.7-5.5 V
Digital I/O Pins	9 (of which 5 provide PWM output)
Analog Input Pins	4
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	8 MHz



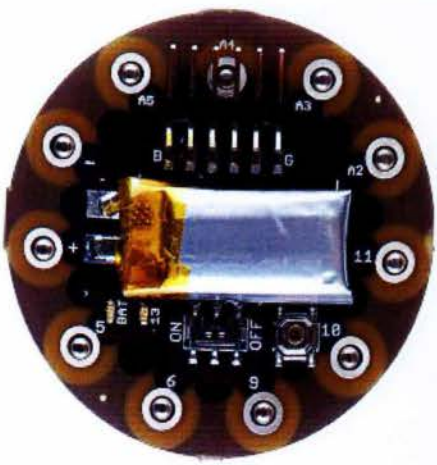
Lilypad Arduino Simple Front



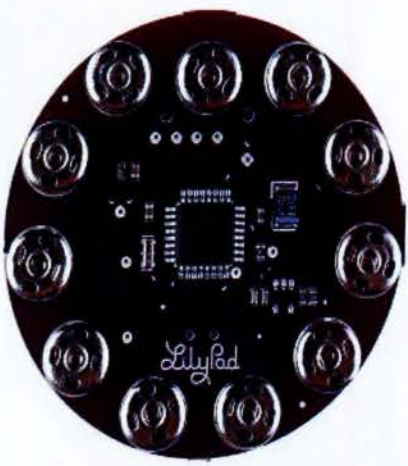
Lilypad Arduino Simple Back

❖ Lilypad Arduino SimpleSnap

Microcontroller	ATmega328
Operating Voltage	2.7-5.5 V
Input Voltage	2.7-5.5 V
Digital I/O Pins	9 (of which 5 provide PWM output)
Analog Input Pins	4
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	8 MHz



Lilypad Arduino SS Front

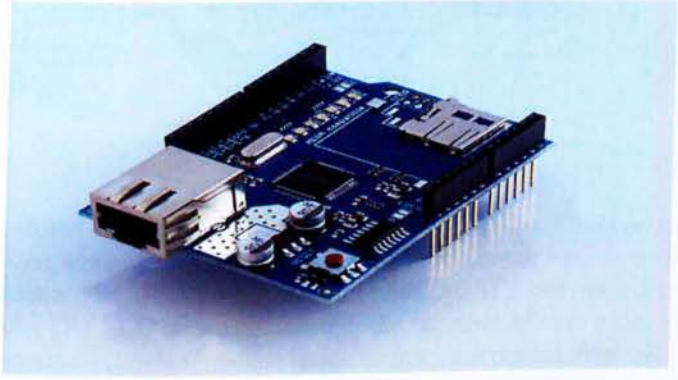


Lilypad Arduino SS Back

1.18 Arduino Shields

Τα **shields** είναι **ολοκληρωμένες πλακέτες** που είναι σχεδιασμένες ώστε να **κουμπώνουν** πάνω στο **Arduino** προεκτείνοντας την λειτουργικότητά του. Είναι η **hardware** αντίστοιχη έννοια των **plugins, addons** και **extensions** που υπάρχουν στο **software**.
Μερικά από τα πιο δημοφιλή **shields** που κυκλοφορούν στο εμπόριο για το **Arduino** είναι:

- **Ethernet shield**: Δίνει στο **Arduino** την δυνατότητα να **δικτυωθεί** σε ένα **LAN** ή στο **internet** μέσω ενός τυπικού **καλωδίου Ethernet**. Για περισσότερες πληροφορίες πάνω σε αυτό το shield και πως λειτουργεί μπορείτε να επισπευτείτε την διεύθυνση
  http://arduino.cc/en/Guide/ArduinoEthernetShield



Ethernet shield

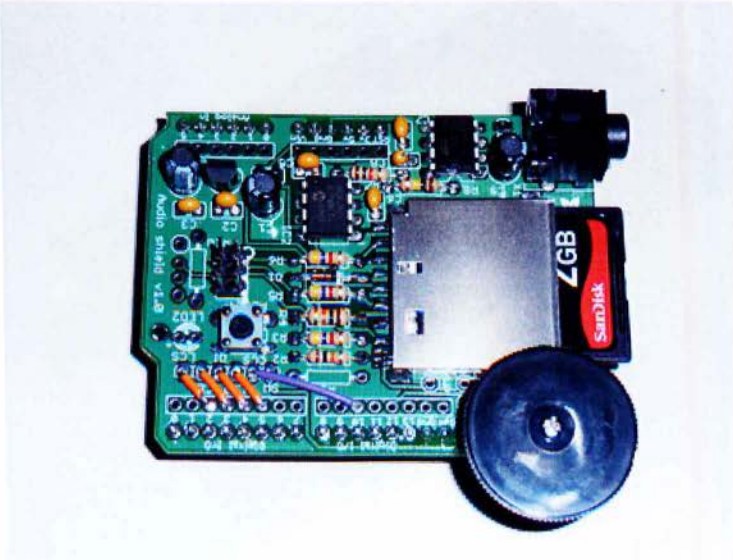
- **WiFi shield**: Όμοιο με το **Ethernet shield**, χωρίς φυσικά το καλώδιο. Επιτρέπει στο Arduino να συνδεθεί με το Internet χρησιμοποιώντας την WiFi βιβλιοθήκη και διαβάζει και γράφει δεδομένα σε μια SD κάρτα κάνοντας χρήση της βιβλιοθήκης SD. Για περισσότερες πληροφορίες πάνω σε αυτό το shield και πως λειτουργεί μπορείτε να επισπευτείτε την διεύθυνση
  http://arduino.cc/en/Guide/ArduinoWiFiShield

WiFi Shield

- Διάφορα **shield οθόνης**: Προσθέτουν οθόνη στο **Arduino**. Κυκλοφορούν από απλές οθόνες τύπου **calculator** μέχρι **OLED touchscreen** υψηλής ανάλυσης τύπου **iPhone**. TFT LCD Screen Shield: Η Arduino TFT οθόνη είναι μια φωτιζόμενη TFT LCD οθόνη με υποδοχή κάρτας microSD στο πίσω μέρος της. Μπορούμε να σχεδιάσουμε κείμενο, εικόνες, και σχέδια στην οθόνη με την TFT βιβλιοθήκη. Για περισσότερες πληροφορίες πάνω σε αυτό το shield και πως λειτουργεί μπορείτε να επισπευτείτε την διεύθυνση **http://arduino.cc/en/Guide/TFT**

TFT LCD shield

- **Wave shield**: Δίνει στο **Arduino** την δυνατότητα να παίζει **ήχους/μουσική** από **κάρτες SD**.

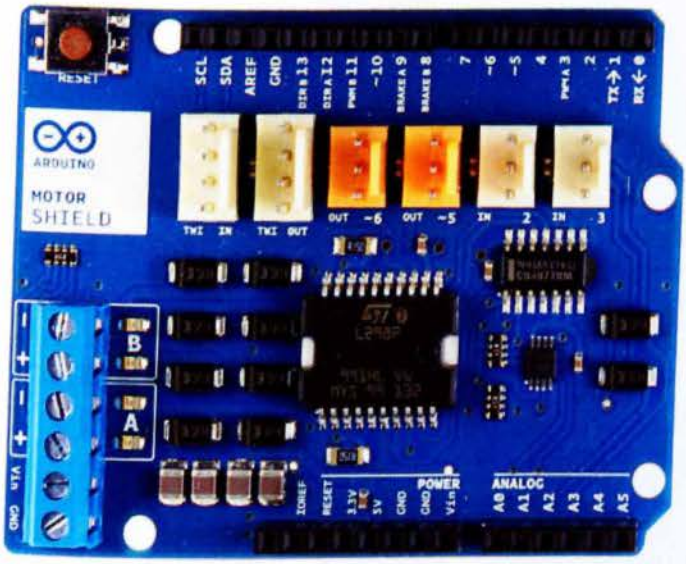


Wave shield

- **GSM shield**: Επιτρέπει στο Arduino να συνδεθεί στο internet, να στείλει και να λάβει SMS και να πραγματοποιεί φωνητικές κλήσεις χρησιμοποιώντας την GSM βιβλιοθήκη. Το shield αυτό λειτουργεί άψογα με την σειρά μητρικών πλακετών Arduino Uno. Το εξάρτημα αυτό θα λειτουργήσει με Arduino Mega, Mega ADK και Leonardo. Προς το παρόν δεν υποστηρίζεται από τα παραπάνω μοντέλα. Για περισσότερες πληροφορίες πάνω σε αυτό το shield και πως λειτουργεί μπορείτε να επισπευτείτε την διεύθυνση **http://arduino.cc/en/Guide/ArduinoGSMShield**

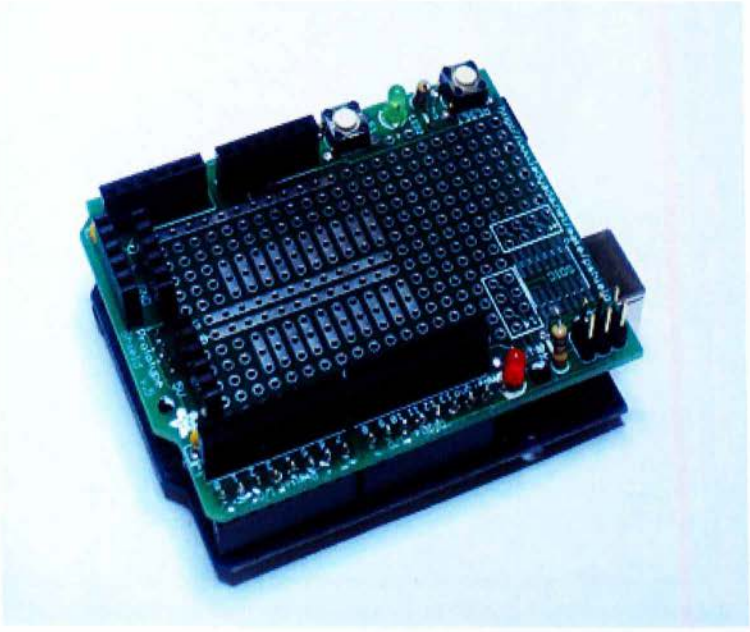
GSM Shield

- Διάφορα **Motor Shields**: Σας επιτρέπουν να **οδηγήσετε** εύκολα **μοτέρ** διάφορων τύπων (απλά **DC**, **servo**, **stepper** κ.λπ.) από το **Arduino**.



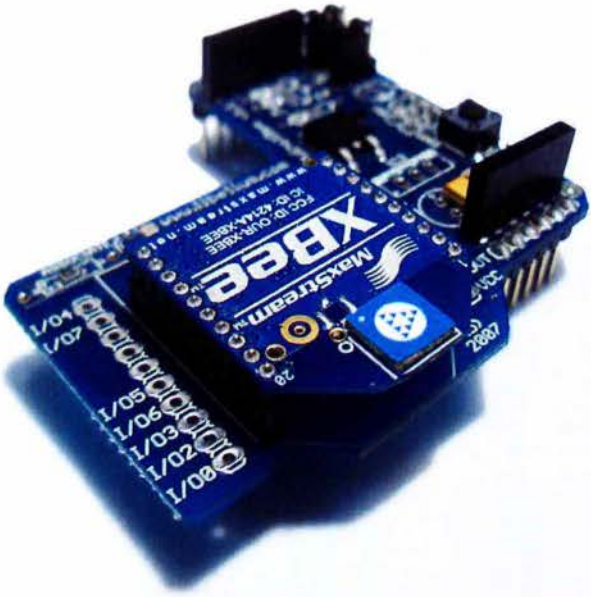
Motor Shield

- **ProtoShield**: Μια προσχεδιασμένη πλακέτα πρωτοτυποποίησης, συμβατή στις διαστάσεις του **Arduino** και χωρίς εξαρτήματα για να **φτιάξετε** το δικό σας **shield**.



Proto Shield

- **Wireless Shield**: Το Wireless Shield επιτρέπει στο Arduino να επικοινωνεί ασύρματα χρησιμοποιώντας το **Zigbee**. Μπορούμε μέσα από το shield να πραγματοποιήσουμε την επικοινωνία ανάμεσα σε δύο Arduino ασύρματα χωρίς καμία απολύτως ρύθμιση. Για περισσότερες πληροφορίες πάνω σε αυτό το shield και πως λειτουργεί μπορείτε να επισπευτείτε την διεύθυνση **http://arduino.cc/en/Guide/ArduinoWirelessShield**

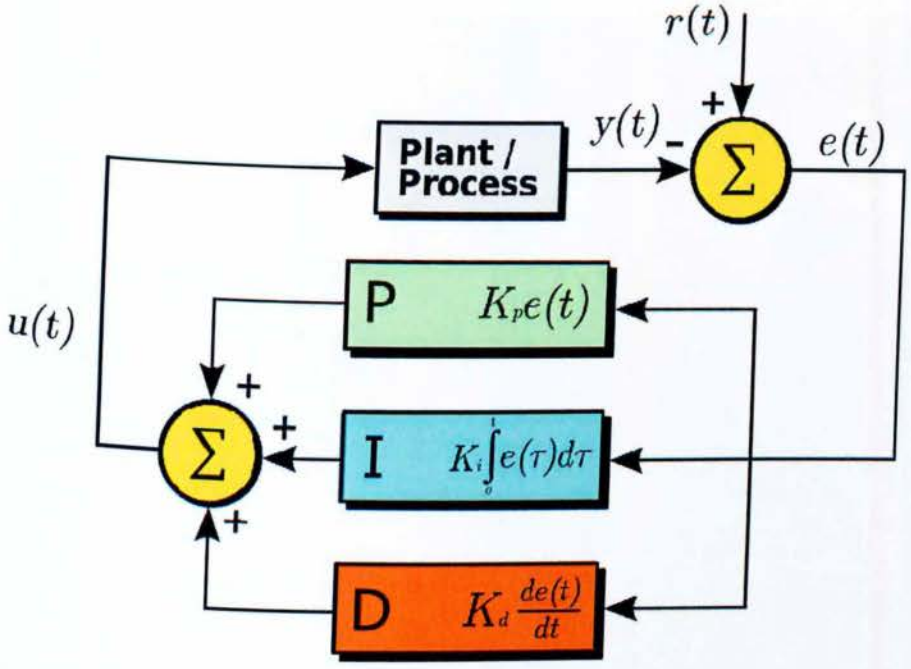


Wireless XBee Shield

Τα **shields** είναι σχεδιασμένα ώστε αφού κουμπωθούν πάνω στο **Arduino** να **προωθούν** τις **υποδοχές** του, ώστε να μπορείτε να **συνδέσετε** επιπλέον τα δικά σας **εξαρτήματα** ή να **κουμπώσετε** και **επόμενο shield**. Φυσικά, το κάθε **shield** χρησιμοποιεί ορισμένους από τους **πόρους** συνδεσιμότητας του **Arduino** και έτσι **δεν** μπορείτε να συνδέσετε **απεριόριστα shields**. Μάλιστα κάποια **shields** μπορεί να μην είναι συμβατά μεταξύ τους γιατί χρησιμοποιούν τα **ίδια pin** του **Arduino** για επικοινωνία με αυτό. Επίσης, επειδή κάποια **shields** δεν προωθούν τις συνδέσεις του **Arduino** (όπως π.χ. οι οθόνες οι οποίες δεν έχουν νόημα αν τις καλύψετε από πάνω με ένα επόμενο **shield**), υπάρχουν ειδικά **extender shields** που κουμπώνουν στο **Arduino** και δίνουν την δυνατότητα σε δύο άλλα **shields** να κουμπώσουν πάνω τους, λειτουργώντας σαν **πολύπριζα**.
Όπως και για το ίδιο το **Arduino**, το βασικό πλεονέκτημα των **shields** δεν είναι τόσο το προφανές πλεονέκτημα του **έτοιμου hardware** όσο ότι συνοδεύονται συνήθως από **έτοιμες βιβλιοθήκες** που σας επιτρέπουν να προγραμματίζετε τα **sketchs** σας σε **high level**. Έτσι, λόγου χάρη, δεν χρειάζεται να διαβάζετε **datasheets** ή να γίνετε ηλεκτρονικός για να συνδέσετε και να λειτουργήσετε ένα **GPS module** πάνω στο **Arduino**. Απλά συνδέετε το **shield**, εγκαθιστάτε τη βιβλιοθήκη που το συνοδεύει και χρησιμοποιείτε μια έτοιμη συνάρτηση -του στυλ **getLocation()**- για να πάρετε το γεωγραφικό στίγμα και να το επεξεργαστείτε περαιτέρω στο **sketch** σας.
Τα **shields** σας λύνουν τα χέρια όταν θέλετε να δημιουργήσετε εύκολα ένα πραγματικά πρακτικό project. Αυτός είναι και ο λόγος που δεν συνιστάται η αγορά κάποιας έκδοσης του **Arduino** που δεν είναι **100% συμβατή** με τα **shields**.

ΚΕΦΑΛΑΙΟ 2

Ο Μικροελεγκτής PID



2.1 Ιστορία του PID Ελεγκτή

Οι PID ελεγκτές χρονολογούνται το 1980 στην ρυθμιστική σχεδίαση (governor design). Οι PID ελεγκτές μεταγενέστερα αναπτύχθηκαν στην αυτόματη πηδαλιούχηση πλοίων. Ένα από τα παλαιότερα παραδείγματα ενός PID πρωτοτύπου αναπτύχθηκε από τον εφευρέτη Αμερικάνο Elmer Sperry το 1911, ενώ η πρώτη δημοσιευμένη θεωρητική ανάλυση ενός PID ελεγκτή είχε πραγματοποιηθεί από τον Ρώσο-Αμερικάνο μηχανικό Nicolas Minorsky. Ο Minorsky σχεδίαζε συστήματα με

αυτόματη πηδαλιούχηση για το Αμερικάνικο Ναυτικό, και η ανάλυσή του βασίστηκε στην παρατήρηση ενός πηδαλιούχου, παρατηρώντας ότι ο πηδαλιούχος χειριζόταν το πλοίο βασισμένος όχι μόνο στο τρέχον σφάλμα (current error), αλλά επίσης και στο προηγούμενο σφάλμα (past error) και στον τρέχον ρυθμό μεταβολής αυτού. Στη συνέχεια αυτό αποτυπώθηκε μαθηματικά από τον Minorsky. Στόχος του ήταν η ευστάθεια, όχι ο γενικός έλεγχος, ο οποίος σημαντικά απλοποίησε το πρόβλημα. Ενώ ο αναλογικός έλεγχος (proportional control) προσφέρει ευστάθεια έναντι μικρών διαταραχών, ήταν ανεπαρκής για την αντιμετώπιση μίας σταθερής διαταραχής, το οποίο απαιτούσε την πρόσθεση του ολοκληρωτικού όρου (integral term). Στο τέλος, ο παραγωγικός όρος (derivative term) προστέθηκε για την βελτίωση του ελέγχου. Δοκιμές συνεχίστηκαν στο πολεμικό πλοίο USS New Mexico, με τον ελεγκτή να ελέγχει την γωνιώδη επιτάχυνση (angular velocity) του πηδαλίου.

2.2 Βασικές γνώσεις για τον Βρόγχο Ελέγχου

Ένα οικείο παράδειγμα ενός βρόγχου ελέγχου αποτελεί η ενέργεια προσαρμογής κρύου-ζεστού στις βρύσες για την διατήρηση του νερού στην επιθυμητή θερμοκρασία. Τυπικά αυτό εμπλέκει την μύξη των δύο διαδικασιών ροής, το ζεστό και το κρύο νερό. Το άτομο ακουμπάει το νερό για να αισθανθεί ή να μετρήσει την

θερμοκρασία του. Βασισμένοι σε αυτή την ανάδραση, εκτελούμε μία ενέργεια ελέγχου για να προσαρμόσουμε τις ζεστές και το κρύες βαλβίδες νερού μέχρι η διαδικασία της θερμοκρασίας να σταθεροποιηθεί στην επιθυμητή τιμή.

Η αίσθηση της θερμοκρασίας του νερού αποτελεί μία μεταβλητή διαδικασία ή διαδικασία τιμών (process value-PV). Η επιθυμητή θερμοκρασία ονομάζεται επιθυμητή τιμή (setpoint-SP). Η είσοδος της διαδικασίας (η θέση της βαλβίδας του νερού) ονομάζεται χειριζόμενη μεταβλητή (manipulated value-MV). Η διαφορά ανάμεσα στη μετρήσιμη θερμοκρασία και την επιθυμητή τιμή είναι το σφάλμα (error-e) και ποσολογεί αν το νερό είναι πολύ ζεστό ή πολύ κρύο και κατά πόσο.

Αφού μετρηθεί η θερμοκρασία (PV), και στη συνέχεια υπολογιστεί το σφάλμα (e), ο ελεγκτής αποφασίζει πότε θα αλλάξει την θέση της βαλβίδας (MV) και κατά πόσο. Όταν ο ελεγκτής πρώτα ενεργοποιήσει την βαλβίδα, μπορεί να περιστρέψει την βαλβίδα του ζεστού ελάχιστα εάν θερμό νερό είναι το επιθυμητό αποτέλεσμα, ή μπορεί να περιστρέψει την βαλβίδα μέχρι το τέλος εάν πολύ ζεστό νερό είναι η επιθυμητή ενέργεια. Αυτό αποτελεί ένα παράδειγμα ενός απλού αναλογικού ελέγχου. Στην περίπτωση που το ζεστό νερό δεν δημιουργείται γρήγορα, ο ελεγκτής μπορεί να μπορεί να επιταχύνει την διαδικασία ανοίγοντας την βαλβίδα ζεστού νερού περισσότερο και περισσότερο καθώς ο χρόνος μεταβάλλεται. Αυτό αποτελεί ένα παράδειγμα ενός ολοκληρωτικού ελέγχου.

Πραγματοποιώντας μία αλλαγή που είναι πολύ μεγάλη όταν το σφάλμα είναι μικρό, ισούται με έναν υψηλού κέρδους ελεγκτή και θα οδηγήσει σε υπέρβαση (overshoot). Αν ο ελεγκτής έκανε συνεχείς αλλαγές οι οποίες θα ήταν πολύ μεγάλες και επαναλαμβανόμενα υπερέβαινε τον στόχο, η έξοδος θα ταλαντωνόταν γύρω από την επιθυμητή τιμή είτε συνεχόμενα, αυξάνοντας, ή σε ημιτονοειδές σήμα. Αν οι ταλαντώσεις αυξηθούν με το χρόνο τότε το σύστημα είναι ασταθές, εάν όμως μειώσουμε τις μεγάλες και απότομες αλλαγές στις τιμές του συστήματος, τότε το σύστημα θα γίνει και πάλι ευσταθές, καθώς θα μειωθούν οι ταλαντώσεις. Αν οι ταλαντώσεις διατηρηθούν σε ένα σταθερό μέγεθος τότε το σύστημα θα είναι οριακά ευσταθές.

Προς την επίτευξη μίας σταδιακής σύγκλισης στην επιθυμητή θερμοκρασία (SP), ο ελεγκτής μπορεί να αποσβέσει τις αναμενόμενες μελλοντικές ταλαντώσεις. Έτσι για να αντισταθμιστεί αυτό το αποτέλεσμα, ο ελεγκτής μπορεί να επιλέξει να μετριάσει τις προσαρμογές του. Αυτό μπορεί να θεωρηθεί ως μία μέθοδος παραγωγικού ελέγχου.

Αν ο ελεγκτής αρχίσει από μία σταθερή κατάσταση σε μηδενικό σφάλμα (PV=SP), τότε περαιτέρω αλλαγές από τον ελεγκτή θα είναι υπεύθυνες σε αλλαγές σε άλλες μετρημένες ή μη μετρημένες εισόδους στη διαδικασία που προσκρούουν στη διαδικασία, και ως εκ τούτου στη διαδικασία τιμών (PV). Μεταβλητές οι οποίες προσκρούουν στη διαδικασία εκτός από την χειριζόμενη μεταβλητή (MV) είναι γνωστές ως διαταραχές. Γενικά, οι ελεγκτές χρησιμοποιούνται για να απορρίπτουν διαταραχές και/ή εφαρμόζουν τις επιθυμητές αλλαγές τιμών. Αλλαγές στη

θερμοκρασία υδροδότησης συνιστούν μία διαταραχή στη θερμοκρασία της βρύσης της διαδικασίας ελέγχου.

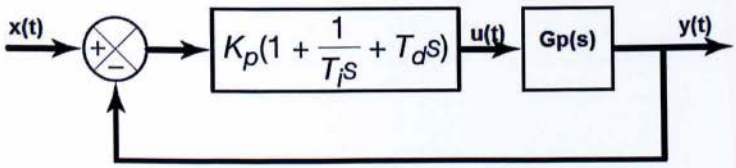
Θεωρητικά, οι ελεγκτές μπορούν να χρησιμοποιηθούν για να ελέγξουν οποιαδήποτε διαδικασία η οποία έχει μία μετρήσιμη έξοδο (PV), μία ιδανική τιμή για την έξοδο (SP) και μία είσοδο στη διαδικασία (MV) η οποία επηρεάζει το σχετικό PV. Οι ελεγκτές χρησιμοποιούνται στη βιομηχανία για να ρυθμίζουν θερμοκρασία, πίεση, ρυθμό ροής, χημικές συνθέσεις, ταχύτητα και πρακτικά κάθε άλλη μεταβλητή για την οποία ένα μετρήσιμο μέγεθος υπάρχει.

2.3 ΕΛΕΓΚΤΕΣ PID

Ο ελεγκτής τριών όρων είναι ουσιαστικά ένας ελεγκτής-αντισταθμιστής σειράς που τοποθετείται στον απ' ευθείας κλάδο του κλειστού συστήματος και ρυθμίζει το σήμα που οδηγεί το σύστημα λαμβάνοντας υπ' όψη την απόκλιση (σφάλμα) της εισόδου από την έξοδο. Η ονομασία του PID ελεγκτή προέρχεται από τα αρχικά των Αγγλικών λέξεων Proportional, Integral και Derivative, που αντιστοιχούν στην αναλογική, ολοκληρωτική και διαφορική δράση που συμπεριλαμβάνει ο ελεγκτής. Ανάλογα με τις ιδιότητες του συστήματος είναι δυνατό να χρησιμοποιηθούν διάφοροι συνδυασμοί των παραπάνω δράσεων. Πιο συχνά συναντάται ο PI ελεγκτής, κυρίως σε εφαρμογές με μικρές καθυστερήσεις χρόνου ή σε περιπτώσεις όπου ο θόρυβος μέτρησης είναι σημαντικός ώστε να αποτρέπεται η χρήση του διαφορικού όρου, ή τέλος όταν δεν απαιτείται το κλειστό σύστημα να είναι αρκετά γρήγορο.

Επίσης, σε συστήματα τύπου 1 (ή μεγαλύτερου), λόγω της εξασφάλισης του μηδενικού σφάλματος στη μόνιμη κατάσταση, πολλές φορές χρησιμοποιείται ο PD ελεγκτής.

Γενικά κάθε κλειστό σύστημα με ελεγκτή σειράς έχει την δομή του σχήματος (9.1) :



Σχήμα 9.1

Η $G_p(s)$ είναι η συνάρτηση μεταφοράς του υπό έλεγχο συστήματος και η $G_c(s)$ είναι η συνάρτηση μεταφοράς του ελεγκτή τριών όρων.

Ο ελεγκτής τριών όρων που ονομάζεται ελεγκτής PID περιγράφεται από το νόμο ελέγχου:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\mathrm{t})\mathrm{d}(\mathrm{t}) + T_\mathrm{d}\frac{de(t)}{dt}\right) = PID \qquad (9.1)$$

όπου:

$$K_{i=}\frac{1}{T_i}\int_0^t e(\mathrm{t})\mathrm{d}(\mathrm{t})$$

$$K_{d\_=}T_\mathrm{d}\frac{de(t)}{dt}$$

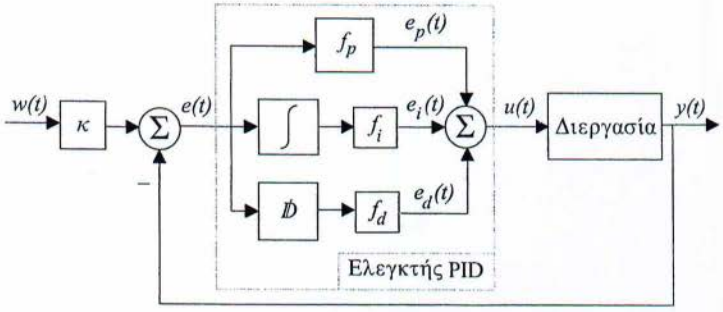
$K_{p=}$: το αναλογικό κέρδος (Proportional Gain), μία ρυθμιζόμενη παράμετρος
K_i: το ολοκληρωτικό κέρδος (Integral Gain), μια ρυθμιζόμενη παράμετρος
K_d: το παραγωγικό κέρδος (Derivative Gain), μια ρυθμιζόμενη παράμετρος
e: Error= SP-PV
t: χρόνος ή στιγμιαίος χρόνος (το τρέχον)
τ: μεταβλητή της ολοκλήρωσης, παίρνοντας τιμές από το 0 έως το τρέχον t
Ti: χρόνος που κάνει το σήμα εξόδου να πάρει τιμή ίση με την τιμή του σήματος εισόδου

όπου u(t) είναι η μεταβλητή ελέγχου και αποτελεί την έξοδο του ελεγκτή και είσοδο της διαδικασίας ενώ $e(t) = y_{set\ point} - y(t) = y_{sp} - y(t)$ είναι το σφάλμα μεταξύ της πραγματικής και της επιθυμητής τιμής της εξόδου y(t) της διαδικασίας υπό έλεγχο. Η μεταβλητή ελέγχου, δηλαδή η έξοδος του ελεγκτή είναι το άθροισμα τριών όρων (σχέση 9.1):

- του αναλογικού όρου P (Propotional - δηλαδή ανάλογο του σφάλματος)

- του ολοκληρωτικού όρου I (Integral - δηλαδή ανάλογο του ολοκληρώματος του σφάλματος)
- του διαφορικού όρου D (Derivative - δηλαδή ανάλογο της παραγώγου του σφάλματος).

Στο σχήμα (9.2) απεικονίζεται το δομικό διάγραμμα του ελεγκτή τριών όρων.



Σχήμα 9.2

Οι παράμετροι ενός βιομηχανικού ελεγκτή τριών όρων είναι :

1.Το αναλογικό κέρδος K_p (Proportional Gain)

2. Ο χρόνος ολοκλήρωσης T_i (Integral Action Time)

3. Ο χρόνος παραγώγισης T_d (Derivative Action Time)

Τα αποτελέσματα της επίδρασης καθενός από τους ελεγκτές, P, I και D σε ένα σύστημα κλειστού βρόχου, συνοψίζονται στον πίνακα (9.1) και εξηγούνται αναλυτικότερα παρακάτω.

Τύπος Ελεγκτή	Χρόνος Ανόδου	Υπερύψωση	Χρόνος Αποκατάστασης	Μόνιμο Σφάλμα
P	Μείωση	Αύξηση	Μικρή Αλλαγή	Μείωση
I	Μείωση	Αύξηση	Αύξηση	Εξάλειψη
D	Μικρή Αλλαγή	Μείωση	Μείωση	Μικρή Αλλαγή

Πίνακας 9.1

2.4 ΕΛΕΓΚΤΗΣ ΤΥΠΟΥ P

Με την κατάλληλη επιλογή του αναλογικού όρου (K_p) του ελεγκτή τύπου P (σχήμα 9.3) εξαλείφονται οι ταλαντώσεις στη μόνιμη κατάσταση και σταθεροποιείται το σήμα εξόδου του κλειστού συστήματος. Πολλές φορές όμως το σύστημα τείνει να εμφανίζει μόνιμο σφάλμα (proportional offset) το οποίο μπορεί μεν να μειωθεί αλλά δεν εξαλείφεται.

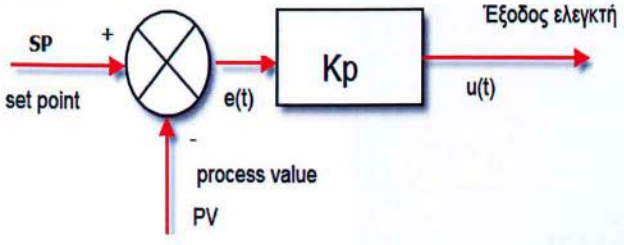
Είναι:

$$u(t) = K_p e(t) \qquad (9.2)$$

$$G_c(s) = \frac{U(s)}{E(s)} = K_p \qquad (9.3)$$

Με την αύξηση του Kp:

- Αυξάνονται οι ταλαντώσεις του συστήματος το οποίο αποσβένει αργότερα (δηλαδή η μεταβατική του συμπεριφορά χειροτερεύει).
- Αυξάνει η φυσική συχνότητα ταλαντώσεων του συστήματος ω_n και μειώνεται ο συντελεστής απόσβεσης J.



Σχήμα 9.3

2.5 ΕΛΕΓΚΤΗΣ ΤΥΠΟΥ I

Με την προσθήκη του ολοκληρωτικού όρου του ελεγκτή τύπου I το σφάλμα στη μόνιμη κατάσταση εξαλείφεται αλλά αυξάνεται ο χρόνος αποκατάστασης t_s του συστήματος. Είναι:

$$u(t) = K_i \int e(t)dt = \frac{K_p}{T_i} \int e(t)dt \qquad (9.4)$$

$$G_c(s) = \frac{U(s)}{E(s)} = \frac{K_i}{s} \tag{9.5}$$

Ο ολοκληρωτικός έλεγχος εισάγει ένα επιπλέον πόλο στο αρχικό σύστημα στο $s = 0$. Συνέπεια αυτού είναι:

- Η αύξηση του τύπου του συστήματος κατά ένα οπότε για οποιαδήποτε διαταραχή της μορφής της βηματικής συνάρτησης το σφάλμα στη μόνιμη κατάσταση μηδενίζεται.
- Ο αρχικός γεωμετρικός τόπος των ριζών της χαρακτηριστικής εξίσωσης μετατοπίζεται δεξιότερα, μειώνεται επομένως η ταχύτητα του συστήματος και χειροτερεύει η σχετική του ευστάθεια.

2.6 ΕΛΕΓΚΤΗΣ ΤΥΠΟΥ D

Με την προσθήκη του διαφορικού όρου του ελεγκτή τύπου D αυξάνεται η ευστάθεια του κλειστού συστήματος και βελτιώνεται η συμπεριφορά της βηματικής του απόκρισης στο μεταβατικό στάδιο. Είναι:

$$u(t) = K_d \frac{d(e(t))}{dt} = K_p T_d \frac{d(e(t))}{dt} \tag{9.6}$$

$$G_c(s) = \frac{U(s)}{E(s)} = K_d s \tag{9.7}$$

Ο διαφορικός έλεγχος εισάγει ένα επιπλέον μηδενικό στο αρχικό σύστημα στο $s = 0$. Συνέπεια αυτού είναι:

- Η μείωση του τύπου του συστήματος κατά ένα οπότε αυξάνει η πιθανότητα να υπάρχει σφάλμα στη μόνιμη κατάσταση.
- Βελτιώνεται η ταχύτητα απόκρισης του συστήματος.

Στο μηχανικό ισοδύναμο, ο διαφορικός όρος του ελεγκτή έχει την ίδια επίδραση με τη γραμμική (μη στατική) τριβή, η οποία είναι ανάλογη της ταχύτητας (παραγώγου της θέσης) και με αντίθετη φορά. Όπως λοιπόν η τριβή χρησιμοποιείται στις αναρτήσεις για να μειώσει την ταλάντωση, έτσι και ο διαφορικός όρος του ελεγκτή

χρησιμοποιείται για την αύξηση της απόσβεσης του κλειστού συστήματος. Είναι προφανές ότι ο D ελεγκτής δεν έχει καμία επίδραση στο σφάλμα στη μόνιμη κατάσταση αφού στη μόνιμη κατάσταση η παράγωγος είναι ίση με μηδέν. Όμως στην περίπτωση ρυθμιστικού ελέγχου (regulatory control) η χρήση διαφορικού όρου μπορεί να μειώσει σημαντικά το μέγιστο σφάλμα που προκύπτει όταν το σύστημα υπόκειται σε εξωτερικές διαταραχές. Η ιδιότητα αυτή οφείλεται στο γεγονός ότι ο διαφορικός όρος αντιλαμβάνεται τη μεταβολή στο σφάλμα (από την κλίση) πολύ πριν το σφάλμα γίνει αρκετά μεγάλο ώστε να γίνει σημαντική η αναλογική δράση. Το σημαντικότερο μειονέκτημα στη χρήση του διαφορικού όρου προέρχεται από την ενίσχυση του υψίσυχνου θορύβου της μέτρησης. Αυτός είναι και ο σημαντικότερος λόγος που αποφεύγεται η χρήση του. Για να ξεπεραστεί το πρόβλημα αυτό συνηθίζεται να χρησιμοποιείται κάποιο φίλτρο σε συνδυασμό με την παραγώγιση.

2.7 ΕΛΕΓΚΤΗΣ ΤΥΠΟΥ PI

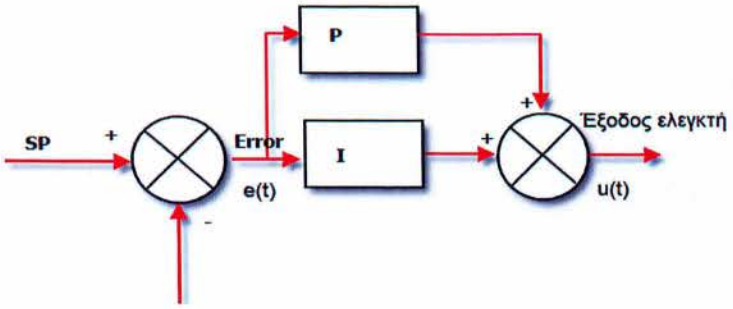
Ο αναλογικός – ολοκληρωτικός ελεγκτής (PI) παρέχει μηδενικό σφάλμα στη μόνιμη κατάσταση και βελτιώνει την ταχύτητα απόκρισης του συστήματος (σχήμα 9.4). Είναι:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int e(t)dt \qquad (9.8)$$

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} = \frac{sK_p + K_i}{s} \overset{(9.8)}{=} K_p(1 + \frac{1}{T_i s}) \qquad (9.9)$$

Ο PI έλεγχος εισάγει ένα επιπλέον μηδενικό στο αρχικό σύστημα στο $s = -\frac{K_i}{K_p}$ και ένα πόλο στο $s = 0$. Συνέπεια αυτού είναι:

- Μεγαλώνουν κατά ένα η τάξη και ο τύπος του συστήματος επομένως βελτιώνεται το σφάλμα στη μόνιμη κατάσταση.
- Χειροτερεύει η ευστάθεια του συστήματος (λόγω του πόλου στο $s = 0$). Για ορισμένες τιμές των K_p και K_i το σύστημα μπορεί να πέσει σε αστάθεια.



Σχήμα 9.4

2.8 ΕΛΕΓΚΤΗΣ ΤΥΠΟΥ *PD*

Ο αναλογικός – διαφορικός ελεγκτής (*PD*) επιτρέπει τη λειτουργία του συστήματος υπό έλεγχο με μεγαλύτερες τιμές κέρδους (σε σχέση με τον *P* έλεγχο) περιορίζει τις ταλαντώσεις του συστήματος και ελαττώνει το σφάλμα στη μόνιμη κατάσταση. Είναι:

$$u(t) = K_p e(t) + K_d \frac{d(e(t))}{dt} \tag{9.10}$$

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + K_d s \tag{9.11}$$

Ο *PI* έλεγχος εισάγει ένα επιπλέον μηδενικό στο αρχικό σύστημα στο $s = -\frac{K_p}{K_d}$.

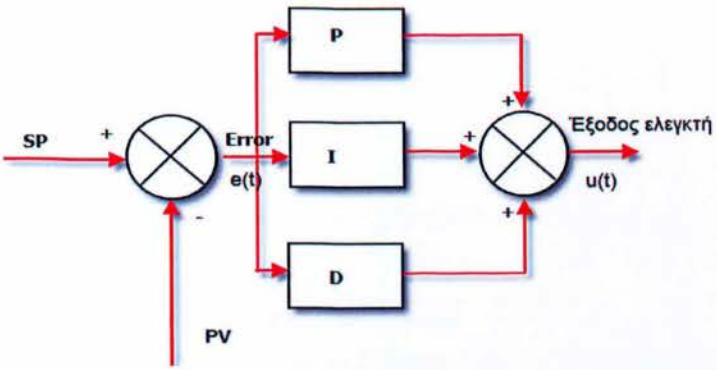
Συνέπεια αυτού είναι η μετατόπιση αριστερότερα του αρχικού γεωμετρικού τόπου των ριζών οπότε υπάρχει η δυνατότητα μείωσης των ταλαντώσεων και αύξησης του αναλογικού κέρδους .

2.9 ΕΛΕΓΚΤΗΣ ΤΥΠΟΥ *PID*

Ο αναλογικός – ολοκληρωτικός – διαφορικός ελεγκτής (*PID*) με την επιλογή των κατάλληλων K_p, K_i και K_d συνδυάζει μικρό χρόνο ανόδου, μικρή υπερύψωση, μικρό χρόνο αποκατάστασης και μηδενικό σφάλμα στη μόνιμη κατάσταση (σχήμα 9.5). Είναι:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{d(e(t))}{dt} \quad (9.12)$$

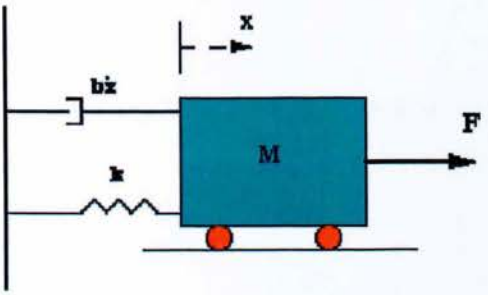
$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} = K_p(1 + \frac{1}{sT_i} + T_d s) \quad (9.13)$$



Σχήμα 9.5

2.10 Παράδειγμα

Ας υποθέσουμε ότι έχουμε ένα σύστημα που αποτελείται από μία μάζα M, ένα ελατήριο και έναν αποσβεστήρα b σε διαμήκη κίνηση λόγω μιας δύναμης F.



Η εξίσωση που περιγράφει το πρόβλημα είναι:
$Mx + bx + kx - F$ (1)
Ο μετασχηματισμός Laplace αυτής της εξίσωσης είναι:
$Ms^2X(s) + bsX(s) + kX(s) - F$
Η συνάρτηση μεταφοράς που προκύπτει ανάμεσα στην μετατόπιση X(s) και την είσοδο F(s) είναι

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

Έστω:

- M = 1kg
- b = 10 N.s/m
- k = 20 N/m
- F(s) = 1

Εάν αντικαταστήσουμε αυτές τις τιμές στην παραπάνω συνάρτηση μεταφοράς θα έχουμε:

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

Σκοπός αυτού του παραδείγματος είναι να δείξει πως οι ελεγκτές Kp, Kd, και Ki συνεργάζονται για να εξασφαλίσουν:

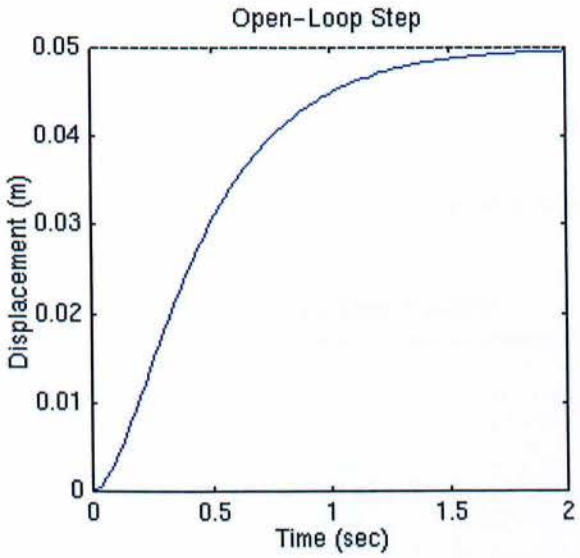
- Γρήγορο χρόνο ανύψωσης
- Ελάχιστη υπερύψωση
- Μηδενικό μόνιμο σφάλμα

2.11 Βηματική απόκριση ανοικτού βρόγχου

Ας ρίξουμε καταρχήν μία ματιά στην βηματική απόκριση ανοικτού βρόγχου. Δημιουργήστε ένα νέο m-file και προσθέστε τον ακόλουθο κώδικα:

```
num=1;
den=[1 10 20];
step(num,den)
```

Εκτελώντας αυτό το m-file στο παράθυρο εντολών του Matlab θα πρέπει να πάρετε το παρακάτω σχήμα.



Το κέρδος DC της συνάρτησης μεταφοράς του συστήματος προς έλεγχο είναι 1/20, και επομένως η τελική τιμή της εξόδου για μια μοναδιαία βηματική είσοδο είναι 0.05. Αυτή η τιμή αντιστοιχεί σε ένα μόνιμο σφάλμα 0.95, το οποίο είναι αρκετά μεγάλο. Επιπλέον ο χρόνος ανύψωσης είναι περίπου ένα δευτερόλεπτο, ενώ ο χρόνος αποκατάστασης είναι περίπου 1,5 δευτερόλεπτο. Ας σχεδιάσουμε έναν ελεγκτή ο οποίος θα μειώσει τον χρόνο ανύψωσης, θα μειώσει τον χρόνο αποκατάστασης και θα εξαλείψει το σφάλμα στη μόνιμη κατάσταση.

2.12 Αναλογικός έλεγχος

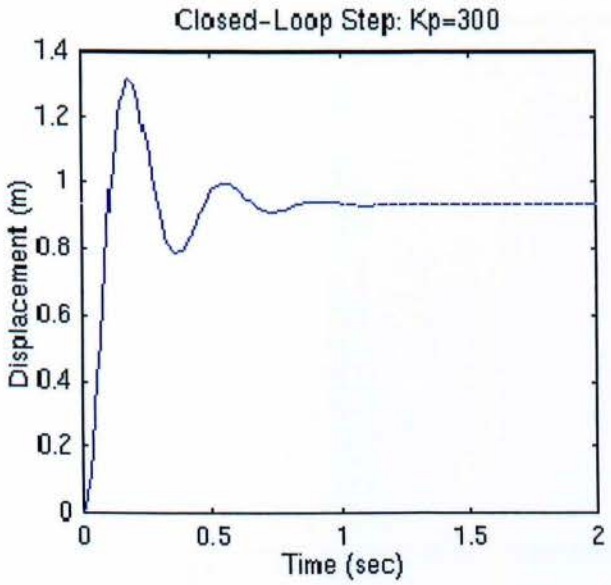
Από τον παραπάνω πίνακα φαίνεται ότι ο αναλογικός ελεγκτής (Kp) μειώνει τον χρόνο ανύψωσης, αυξάνει την υπέρβαση και μειώνει το μόνιμο σφάλμα. Η συνάρτηση μεταφοράς κλειστού βρόγχου του παραπάνω συστήματος με αναλογικό ελεγκτή γίνεται:

$$\frac{X(s)}{F(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

Έστω ότι η τιμή του αναλογικού κέρδους είναι 300 και ας αλλάξουμε τα παρακάτω στο m-file:

```
Kp=300;
num=[Kp];
den=[1 10 20+Kp];
t=0:0.01:2;
step(num,den,t)
```

Εκτελώντας αυτό το m-file στο παράθυρο εντολών του Matlab θα πρέπει να πάρετε το παρακάτω σχήμα

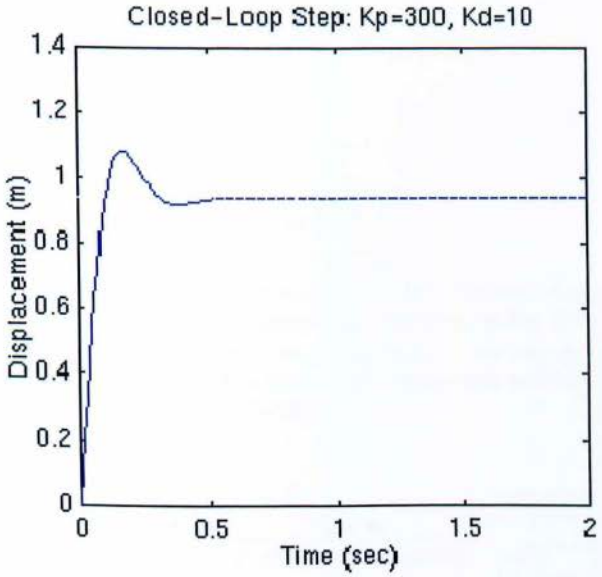


Σημείωση: Η συνάρτηση του Matlab, που καλείται cloop μπορεί να χρησιμοποιηθεί για να πάρουμε την συνάρτηση μεταφοράς κλειστού

βρόγχου απευθείας από την αντίστοιχη συνάρτηση μεταφοράς ανοιχτού βρόγχου (αντί να την υπολογίσουμε με το χέρι). Το ακόλουθο m-file χρησιμοποιεί την εντολή cloop, η οποία θα πρέπει να σας δώσει μία καμπύλη όμοια με αυτήν που απεικονίζεται παραπάνω.

```
num=1;
den=[1 10 20];
Kp=300;
[numCL,denCL]=cloop(Kp*num,den);
t=0:0.01:2;
step(numCL, denCL,t)
```

Η παραπάνω καμπύλη δείχνει ότι ο αναλογικός ελεγκτής μειώνει τόσο τον χρόνο ανύψωσης, όσο και το μόνιμο σφάλμα, αυξάνει όμως την υπερύψωση και μειώνει κατά ένα μικρό ποσοστό τον χρόνο αποκατάστασης.



Η παραπάνω καμπύλη δείχνει ότι ο διαφορικός ελεγκτής μειώνει τόσο την υπερύψωση όσο και τον χρόνο αποκατάστασης και έχει μικρή επίδραση στον χρόνο ανύψωσης καθώς και στο μόνιμο σφάλμα.

2.13 Αναλογικός-Διαφορικός έλεγχος

Τώρα, ας ρίξουμε μία ματιά σε έναν έλεγχο PD. Από τον πίνακα που απεικονίζεται παραπάνω βλέπουμε ότι ο διαφορικός ελεγκτής (Kd) μειώνει τόσο την υπερύψωση όσο και τον χρόνο αποκατάστασης. Η συνάρτηση μεταφοράς κλειστού βρόγχου του συστήματος που περιγράφουμε παραπάνω είναι:

$$\frac{X(s)}{F(s)} = \frac{K_D s + K_P}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Έστω ότι η τιμή του αναλογικού κέρδους Kp ισοδυναμεί με 300 όπως και προηγουμένως και ας δώσουμε στο διαφορικό κέρδος Kd την τιμή 10. Ας βάλουμε τις παρακάτω εντολές σε ένα αρχείο m-file και ας το εκτελέσουμε μέσα από το παράθυρο εντολών του Matlab.

```
Kp=300;
Kd=10;
num=[Kd Kp];
den=[1 10+Kd 20+Kp];
t=0:
```

2.14 Αναλογικός-Ολοκληρωτικός έλεγχος

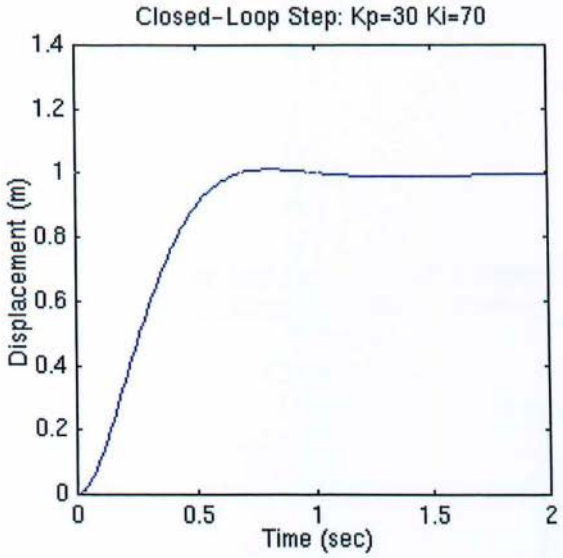
Πριν από την μελέτη του ελέγχου PID, ας ρίξουμε μία ματιά στον έλεγχο PI. Από τον παραπάνω πίνακα βλέπουμε ότι ο ολοκληρωτικός ελεγκτής (Ki) μειώνει τον χρόνο ανύψωσης, ενώ αυξάνει την υπέρβαση και τον χρόνο αποκατάστασης και εξαλείφει το μόνιμο σφάλμα. Στο σύστημα που περιγράψαμε παραπάνω, η συνάρτηση μεταφοράς κλειστού βρόγχου με PI έλεγχο, είναι:

$$\frac{X(s)}{F(s)} = \frac{K_p s + K_i}{s^3 + 10s^2 + (20 + K_p)s + K_i}$$

Ας μειώσουμε την τιμή του Kp στο 30 και ας δώσουμε στο ολοκληρωτικό κέρδος Ki την τιμή 70. Δημιουργήστε ένα νέο αρχείο mfile και καταχωρήστε τις ακόλουθες εντολές:

```
Kp=30;
Ki=70;
num=[Kp Ki];
den=[1 10 20+Kp Ki];
t=0:0.01:2;
step(num,den,t)
```

Εάν εκτελέσετε αυτό το αρχείο μέσα από το παράθυρο εντολών του Matlab, θα πρέπει να πάρετε το παρακάτω σχήμα:



Μειώσαμε την τιμή του αναλογικού κέρδους (Kp) γιατί ο ολοκληρωτικός ελεγκτής μειώνει επίσης τον χρόνο ανύψωσης και αυξάνει την υπερύψωση, όπως και ο αναλογικός (διπλή επίδραση). Η παραπάνω απόκριση δείχνει ότι ο ολοκληρωτικός ελεγκτής εξαλείφει το μόνιμο σφάλμα.

2.15 Αναλογικός-Ολοκληρωτικός-Διαφορικός έλεγχος

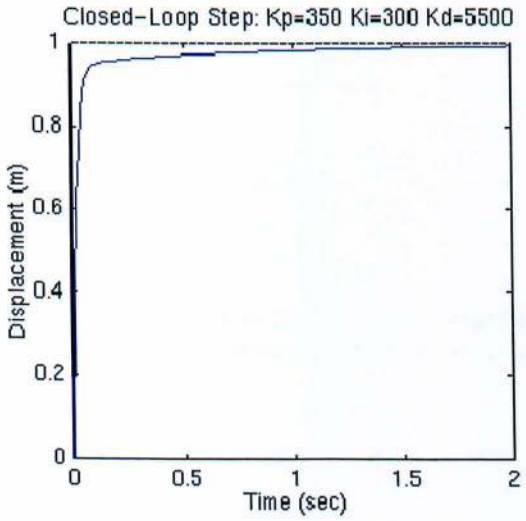
Τώρα, ας εξετάσουμε τον PID ελεγκτή. Η συνάρτηση μεταφοράς κλειστού βρόγχου του δοθέντος συστήματος είναι:

$$\frac{X(s)}{F(s)} = \frac{K_D s^2 + K_P s + K_I}{s^3 + (10 + K_D)s^2 + (20 + K_P)s + K_I}$$

Υστερα από πολλές προσπάθειες και δοκιμές που κατέληξαν σε λάθη, καταλήξαμε στα παρακάτω κέρδη, τα οποία μας δίνουν την επιθυμητή απόκριση: Kp=350, Ki=300 και Kd=50. Για να επιβεβαιώσουμε τις τιμές, εισάγουμε τις ακόλουθες

εντολές σε ένα αρχείο m-file και το εκτελούμε μέσα από το παράθυρο εντολών του Matlab.

```
Kp=350;
Ki=300;
Kd=50;
num=[Kd Kp Ki];
den=[1 10+Kd 20+Kp Ki];
t=0:0.01:2;
step(num,den,t)
```



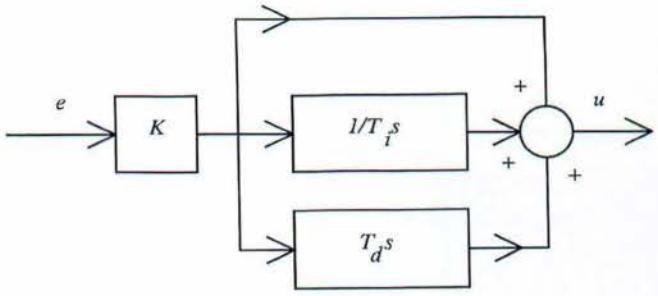
Τελικά, παρατηρούμε ένα σύστημα χωρίς υπέρβαση, με γρήγορο χρόνο ανύψωσης και καθόλου μόνιμο σφάλμα.

2.16 ΕΝΑΛΛΑΚΤΙΚΕΣ ΜΟΡΦΕΣ ΤΟΥ ΕΛΕΓΚΤΗ ΤΡΙΩΝ ΟΡΩΝ

Ο κλασικός αλγόριθμος του βιομηχανικού ελεγκτή μπορεί να εκφραστεί στις εξής μορφές:

1.Στην **παράλληλη μορφή** όπου ο ελεγκτής έχει συνάρτηση μεταφοράς αυτή της σχέσης (9.13) και το δομικό διάγραμμα υλοποίησης απεικονίζεται στο σχήμα (9.6). Η παράμετρος $K = K_p$ και συμβολίζει το αναλογικό κέρδος το οποίο επηρεάζει

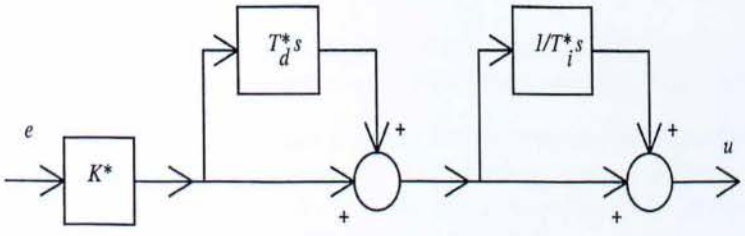
όλους τους όρους του ελεγκτή ενώ οι σταθερές χρόνου T_i και T_d δεν αλληλεπιδρούν.



Σχήμα 9.6

2. Στη **σειριακή μορφή** όπου ο ελεγκτής έχει συνάρτηση μεταφοράς αυτή της σχέσης (9.14) και το δομικό διάγραμμα υλοποίησης απεικονίζεται στο σχήμα (9.7).

$$G_c(s)=\frac{U(s)}{E(s)}=K^*(1+\frac{1}{T_i^* s})(1+T_d^* s) \qquad (9.14)$$



Σχήμα 9.7

Η σειριακή μορφή έχει το μειονέκτημα ότι οι σταθερές των όρων ολοκλήρωσης και παραγώγισης αλληλεπιδρούν και για το λόγο αυτό δεν χρησιμοποιείται συχνά.

3. Στην **αναλυτική μορφή** όπου ο ελεγκτής έχει συνάρτηση μεταφοράς αυτή της σχέσης (9.15).

$$G_c(s) = K_p + \frac{K_i}{s} + K_d s = \frac{k}{s}(s+z_1)(s+z_2) \qquad (9.15)$$

Η μορφή αυτή είναι χρήσιμη κατά την ανάλυση της ευστάθειας του κλειστού συστήματος και έχει το πλεονέκτημα ότι οι τρεις όροι εμφανίζονται ανεξάρτητα.

2.17 ΣΧΕΔΙΑΣΗ ΕΛΕΓΚΤΩΝ *PID*

Σχεδίαση ελεγκτή *PID* σημαίνει την κατάλληλη επιλογή των παραμέτρων του (tuning) ώστε το σύστημα υπό έλεγχο να λειτουργεί με τις επιθυμητές προδιαγραφές. Οι μέθοδοι υπολογισμού των παραμέτρων ενός ελεγκτή τριών όρων είναι οι **εμπειρικές** μέθοδοι και οι **υπολογιστικές** μέθοδοι. Στις εμπειρικές μεθόδους υπολογίζουμε τις παραμέτρους έχοντας τη βηματική απόκριση του συστήματος ενώ τις περισσότερες φορές είναι άγνωστοι οι επιμέρους συναρτήσεις μεταφοράς του κλειστού συστήματος. Ο σχεδιασμός του ελεγκτή γίνεται με την επιθυμία να έχουμε ικανοποιητική υπερύψωση, σχετικά ταχεία απόκριση και μικρό μόνιμο σφάλμα.

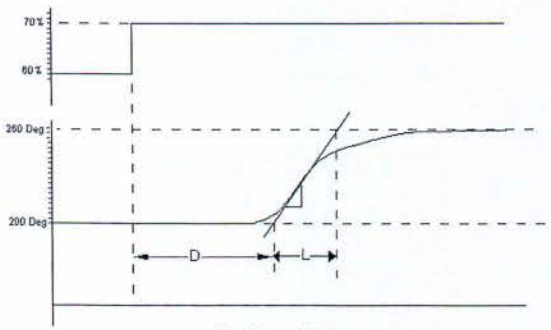
2.17.1 ΠΡΩΤΗ ΜΕΘΟΔΟΣ Ziegler-Nichols

Οι **Ziegler-Nichols** (1940) πρότειναν δύο πειραματικές μεθόδους με βασικό τους πλεονέκτημα ότι δεν απαιτείται η γνώση του αναλυτικού μοντέλου του συστήματος.

Η πρώτη μέθοδος Ziegler - Nichols στηρίζεται στον πειραματικό προσδιορισμό της βηματικής απόκρισης της διεργασίας. Η μέθοδος χαρακτηρίζεται από δύο παραμέτρους που αφορούν την καθυστέρηση της απόκρισης και τη μέγιστη ταχύτητα της βηματικής απόκρισης όπως φαίνεται στο σχήμα (9.8).

Στον πίνακα (9.3) που ακολουθεί δίνονται οι προτεινόμενες τιμές των παραμέτρων K_p, K_i και K_d του ελεγκτή PID βάσει των τιμών των D (=η καθυστέρηση νεκρού χρόνου) και L (=ο χρόνος από το σημείο τομής της εφαπτομένης με τον οριζόντιο άξονα έως την τιμή της εξόδου στη μόνιμη κατάσταση).
Η χρήση της μεθόδου ενδείκνυται σε συστήματα τύπου 0 με καθυστέρηση νεκρού χρόνου (dead time) για τα οποία η Σ.Μ μπορεί προσεγγιστικά να πάρει τη μορφή:

$$G(s) = \frac{K}{L}\frac{e^{-Ds}}{s+1/L} \tag{9.16}$$



Σχήμα 9.8

ΕΛΕΓΚΤΗΣ	K_p	K_i	K_d
P	L/D	0	0
PI	$0.9\,L/D$	$3.33/D$	0
PID	$1.2\,L/D$	$0.5/D$	$0.5/D$

Πίνακας 9.3

Βάσει των τιμών του Πίνακα 9.3 , η συνάρτηση μεταφοράς του ελεγκτή θα είναι της μορφής:

$$G_c(s) = 0.6L\frac{(s+\frac{1}{D})^2}{s} \tag{9.17}$$

Η σχέση (9.17) προκύπτει αν γράψουμε τη σχέση εισόδου – εξόδου του ελεγκτή η οποία είναι:

$$u(t) = 1.2\frac{L}{D}e(t) + 0.6\frac{L}{D^2}\int_0^t e(t)dt + 0.6L\frac{de(t)}{dt} \qquad (9.18)$$

2.17.2 ΔΕΥΤΕΡΗ ΜΕΘΟΔΟΣ Ziegler-Nichols

Η δεύτερη μέθοδος Ziegler - Nichols στηρίζεται βασίζεται στην απόκριση της διαδικασίας σε ημιτονοειδή διέγερση και εφαρμόζεται για συστήματα κάθε τύπου στα οποία η έξοδος μπορεί να οδηγηθεί σε αμείωτες ταλαντώσεις. Τα βήματα για τον καθορισμό των δύο παραμέτρων είναι:

- Με κλειστό βρόχο εφαρμόζουμε αναλογικό ελεγκτή στη διεργασία (τα κέρδη K_i και K_d θέτονται ίσα με μηδέν).
- Η εξωτερική είσοδος της διεργασίας τίθεται ίση με το βηματικό σήμα συνεχούς χρόνου.
- Το αναλογικό κέρδος K_p αυξάνεται έως ότου το σύστημα κλειστού βρόχου φτάσει στο όριο ευστάθειας, δηλ. έως ότου η βηματική απόκριση του εμφανίσει αμείωτες ταλαντώσεις.
- Στην κρίσιμη περίπτωση σημειώνονται οι τιμές της περιόδου της ταλάντωσης T_{cr} καθώς και του κέρδους που προκάλεσε την ταλάντωση K_{cr}.

Στον πίνακα (9.4) που ακολουθεί δίνονται οι προτεινόμενες τιμές των παραμέτρων K_p, K_i και K_d του ελεγκτή PID βάσει των τιμών των K_{cr} και T_{cr}.

ΕΛΕΓΚΤΗΣ	K_p	K_i	K_d
P	$0.5K_{cr}$	0	0
PI	$0.45K_{cr}$	$1.2/T_{cr}$	0
PID	$0.6K_{cr}$	$2/T_{cr}$	$0.125T_{cr}$

Πίνακας 9.4

Βάσει των τιμών του Πίνακα 9.4 , η συνάρτηση μεταφοράς του ελεγκτή θα είναι της μορφής:

$$G_c(s) = 0.075K_{cr}T_{cr}\frac{(s+\frac{4}{T_{cr}})^2}{s} \qquad (9.19)$$

Η σχέση (9.19) προκύπτει αν γράψουμε τη σχέση εισόδου – εξόδου του ελεγκτή η οποία είναι:

$$u(t) = 0.6K_{cr}e(t) + 1.2\frac{K_{cr}}{T_{cr}}\int_0^t e(t)dt + 0.075K_{cr}T_{cr}\frac{de(t)}{dt} \qquad (9.20)$$

Οι σχέσεις βαθμονόμησης που πρότειναν οι Ziegler και Nichols καθορίστηκαν εμπειρικά για να παρέχουν αποκρίσεις κλειστού βρόχου που έχουν λόγο απόσβεσης 50%. Όταν λαμβάνει χώρα μόνο αναλογικός έλεγχος, οι τιμές που υπολογίζονται από τον πίνακα (9.4) παρέχουν ένα περιθώριο ενίσχυσης ίσο με 2 για το K_p επειδή είναι ίσο με μισή φορά την τιμή του ορίου ευστάθειας K_{cr} . Όταν προστεθεί ολοκληρωτική δράση και ο ρυθμιστής γίνει PI, το K_p ελαττώνεται από $0.5K_{cr}$ σε $0.45K_{cr}$. Η σταθεροποιητική επίδραση της διαφορικής δράσης, οπότε και σχηματίζεται ρυθμιστής PID, επιτρέπει την αύξηση του K_p στο $0.6K_{cr}$.

Η παραπάνω μέθοδος παρουσιάζει ορισμένα σημαντικά μειονεκτήματα:

1. Η μεγάλη διάρκεια των πειραματικών ελέγχων μπορεί να οδηγήσει σε μειωμένη παραγωγή ή σε χαμηλή ποιότητα προϊόντος.
2. Σε πολλές εφαρμογές, δεν είναι επιτρεπτή η ταλάντωση που παρατηρείται στην περίπτωση της οριακής ευστάθειας, επειδή το σύστημα προς έλεγχο ωθείται στα όρια της ευσταθούς συμπεριφοράς και αν συμβούν εξωτερικές διαταραχές ή αλλαγές στη διεργασία κατά την διάρκεια του ελέγχου, είναι πιθανό να προκύψει ασταθής ή και επικίνδυνη λειτουργία .
3. Αυτή η διαδικασία εύρεσης των παραμέτρων δεν είναι δυνατή σε συστήματα με ολοκληρωτική συμπεριφορά επειδή η συμπεριφορά τους είναι γενικά ασταθής για υψηλές και χαμηλές τιμές του K_p, παρότι είναι ευσταθείς για ενδιάμεσες τιμές.
4. Για μοντέλα πρώτης και δεύτερης τάξης χωρίς χρονική καθυστέρηση, δεν υπάρχει τελική ενίσχυση επειδή το σύστημα κλειστού βρόχου είναι ευσταθές για όλες τις τιμές του K_p. Ωστόσο, στην πράξη είναι ασυνήθιστο να μην έχει τελική ενίσχυση ένας βρόχος ρύθμισης.

2.17.3 Η ΜΕΘΟΔΟΣ ΣΥΝΤΟΝΙΣΜΟΥ ΤΩΝ Chien ,Hrones και Reswick

Οι **Chien, Hrones και Reswick** πρότειναν μια τροποποίηση της πρώτης μεθόδου των Ziegler και Nichols ώστε το τελικό σύστημα να έχει μεγαλύτερο βαθμό απόσβεσης. Η μέθοδος αυτή επιτυγχάνει ταχύτερη απόκριση με δεδομένα ποσοστά υπερύψωσης (0% ή 20%). Η μέθοδος των Chien, Hrones και Reswick διαφέρει από τη μέθοδο των Ziegler και Nichols γιατί χρησιμοποιεί τρεις παραμέτρους της διαδικασίας αντί δύο.

Για το συντονισμό ενός ελεγκτή με τη μέθοδο *CHR* οι συντελεστές των Ziegler – Nichols επιλέγονται βάσει του πίνακα (9.5).

ΕΛΕΓΚΤΗΣ	K_p	T_i	T_d
P	$\frac{1}{K}(0.35+\frac{1}{\mu})$, $\mu = D/L$	0	0
PI	$\frac{1}{K}(0.083+\frac{0.9}{\mu})$	$\frac{3.33+0.31\mu}{1+2.2\mu}D$	0
PD	$\frac{1}{K}(0.16+\frac{1.24}{\mu})$	0	$\frac{0.27-0.088\mu}{1+0.13\mu}D$
PID	$\frac{1}{K}(0.25+\frac{1.35}{\mu})$	$\frac{2.5+0.46\mu}{1+0.61\mu}D$	$\frac{0.37}{1+0.19\mu}D$

Πίνακας 9.5

Η επιλογή του τύπου του ελεγκτή γίνεται βάσει του πίνακα (9.6)

ΤΥΠΟΣ ΕΛΕΓΚΤΗ	$R = L/D$
P	$R>10$
PI	$7.5<R<10$
PID	$3<R<7.5$

Πίνακας 9.6

Για απεριοδική απόκριση οι παράμετροι του ελεγκτή συντονίζονται βάσει του πίνα
(9.7)

ΕΛΕΓΚΤΗΣ	K_p	T_i	T_d
P	$0.3R/K$	0	0
PI	$0.35R/K$	$1.2L$	0
PID	$0.6R/K$	L	$0.5D$

Πίνακας 9.7

2.17.4 Η ΜΕΘΟΔΟΣ ΣΥΝΤΟΝΙΣΜΟΥ ΤΩΝ Cohen και Coon

Με αυτή τη μέθοδο η απόκριση του ανοιχτού βρόχου (έχοντας βγάλει τον ελεγκτή από το σύστημα) σε βηματική είσοδο μετράται και προσεγγίζεται με μικρές ευθείες γραμμές δίνοντας μία καμπύλη της μορφής του σχήματος (9.8), στην οποία ακολουθείται η ίδια διαδικασία με την Ziegler-Nichols, για την εξαγωγή της καμπύλης. Παράγει απόκριση κλειστού βρόχου με λόγο εξασθένισης (decay ratio). Το βασικό πλεονέκτημα της είναι η απλότητα υλοποίησης, αλλά το μειονέκτημα της είναι πως οι αποκρίσεις έχουν ταλαντώσεις.

Η μέθοδος των Cohen και Coon βασίζεται στην τοποθέτηση των επικρατούντων πόλων και συγκεκριμένα σε πρότυπο της διαδικασίας τριών παραμέτρων της κλασικής μορφής:

$$G_p(s) = K\frac{e^{-t_d s}}{\tau s + 1} \qquad (9.21)$$

Βασικός στόχος της μεθόδου είναι ο περιορισμός των διαταραχών του φορτίου στην έξοδο του συστήματος με την κατάλληλη τοποθέτηση των πόλων του κλειστού συστήματος ώστε να επιτυγχάνεται συντελεστής απόσβεσης 0.25. Έτσι με τη

μέθοδο αυτή έχουμε ελαχιστοποίηση του μόνιμου σφάλματος λόγω διαταραχών του φορτίου για ελεγκτές δύο και τριών όρων.

Οι παράμετροι του ελεγκτή PID υπολογίζονται βάσει της μεθόδου Cohen-Coon από τους τύπους:

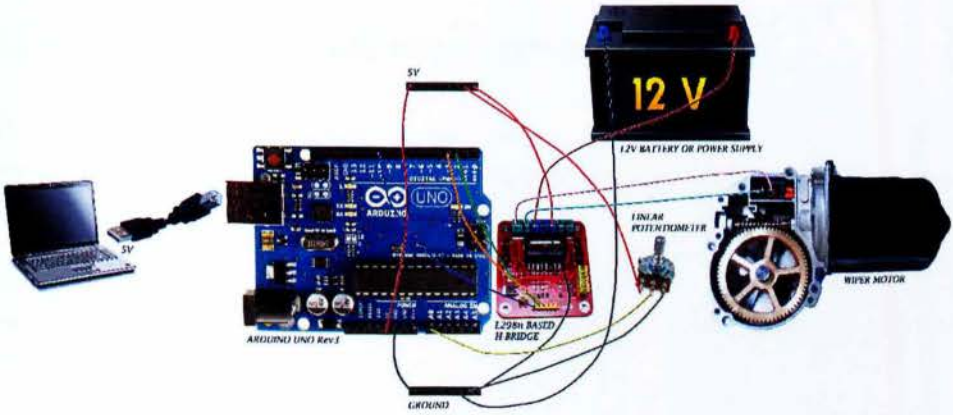
$$K_p = \frac{1}{K}\frac{\tau}{t_d}\left(\frac{4}{3}+\frac{t_d}{4\tau}\right)$$

$$T_i = t_d \frac{32+\frac{6t_d}{\tau}}{13+\frac{8t_d}{\tau}}$$

$$T_d = \frac{4t_d}{11+\frac{2t_d}{\tau}}$$

ΚΕΦΑΛΑΙΟ 3

ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ



3.1 Εισαγωγή

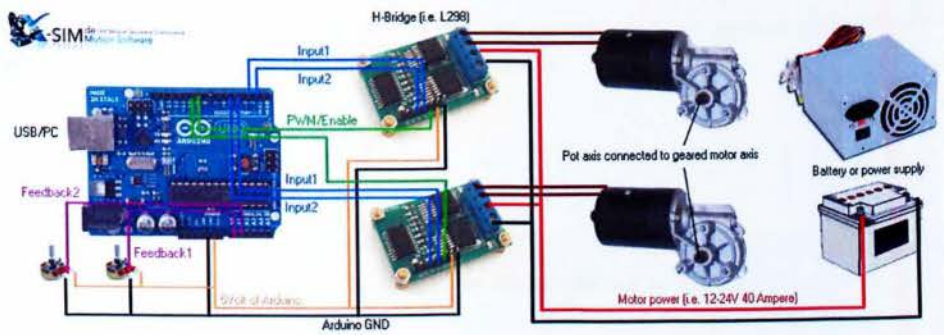
Στο πειραματικό μέρος θα αναλύσουμε το σύστημα ελέγχου μας ως εξής:

- ανάλυση της λειτουργίας του,
- τα μέρη από τα οποία αποτελείται το σύστημα,
- βήμα-βήμα πώς θα συνδέσουμε τα εκάστοτε μέρη,
- την σύνδεση με τον υπολογιστή και την επικοινωνία του με το Arduino,
- την φόρτωση του κώδικα στο Arduino και την εκτέλεση του, την ανάλυση του τμήματος του κώδικα που αντιστοιχεί στο κομμάτι του ελέγχου PID,
- την διεπαφή (interface-XSim Motion Converter) με την οποία ελέγχουμε τον κινητήρα DC,
- την πραγματοποίηση του ελέγχου PID χειροκίνητα.

3.2 Ανάλυση λειτουργίας του συστήματος

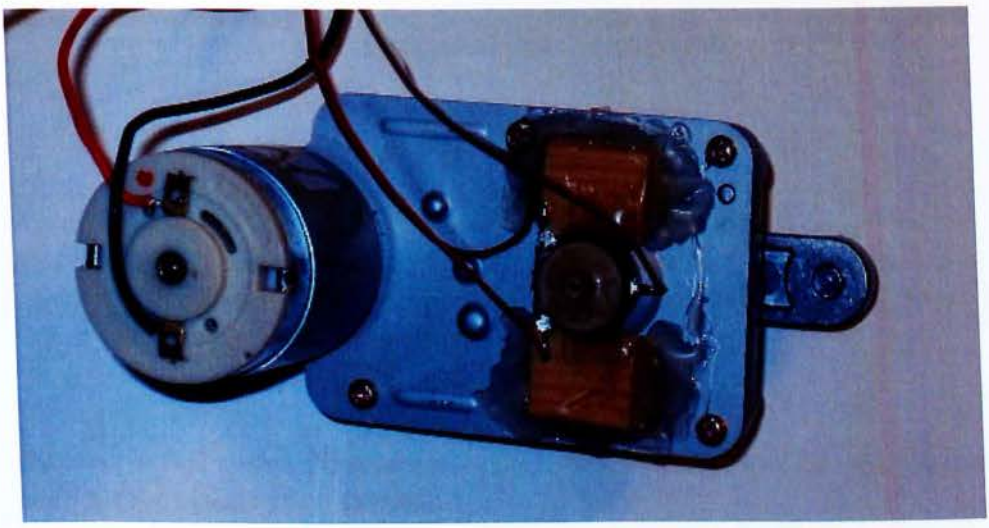
Στο project αυτό ουσιαστικά προσπαθούμε να επιτύχουμε εξομοίωση ενός σερβοκινητήρα χρησιμοποιώντας: DC κινητήρα με γρανάζια (ηλεκτρομειωτήρας, κινητήρας που συναντάται στους υαλοκαθαριστήρες αυτοκινήτων), arduino (αλγόριθμος PID και κατάλληλος χειρισμός του feedback), H-bridge (μεταβολή φοράς περιστροφής, και τροφοδοσία του κινητήρα καθώς το arduino δεν μπορεί να αντέξει σε πολλά ampere), ποτενσιόμετρο (παροχή feedback), μπαταρία (τροφοδοσία κινητήρα).

Σύστημα:



Για να δημιουργήσουμε το feedback, προσκολλούμε το ποτενσιόμετρο στον άξονα περιστροφής που είναι κάθετος του αρχικού άξονα περιστροφής του κινητήρα.

Προσκόλληση ποτενσιόμετρου στον κινητήρα:

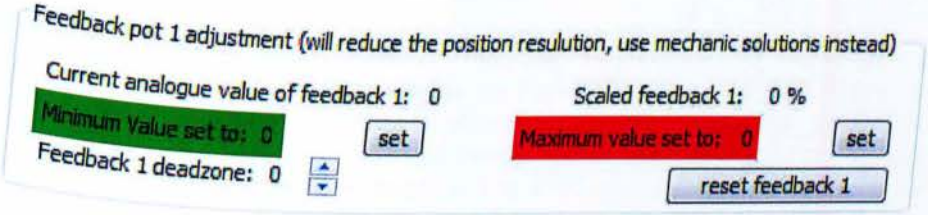


Με αυτό τον τρόπο επιτυγχάνουμε, ως ανατροφοδότηση από το ποτενσιόμετρο, μια αναλογική τιμή από 0-5V(όρια ποτενσιόμετρου) που μεταβάλλεται καθώς στρέφεται ο κινητήρας. Αυτό το αναλογικό μέγεθος μεταφράζεται από το A/D converter του arduino σε ψηφιακό παλμό PWM, η τιμή του οποίου κυμαίνεται από 0-1023, και αποτελεί το μέγεθος ανάλογα με το οποίο αυξομειώνεται η ταχύτητα του κινητήρα. Κάνοντας λοιπόν την διαίρεση 5V/1023 καταλαβαίνουμε ότι για κάθε μοναδιαία αλλαγή στο παλμό, η αλλαγή που πρέπει να έχουμε στον ποτενσιόμετρο πρέπει να είναι της τάξης των 0.0048V, περίπου 5millivolts.

Μετατροπή ποτενσιόμετρου από 47kOhm σε 21kOhm ώστε να επιτύχουμε feedback 0-5V (δεν είναι αναγκαία καθώς μπορεί να επιτευχθεί σε επίπεδο software).

Αν επιλέξουμε την μέθοδο του software, και όχι την μηχανική μετατροπή του ποτενσιόμετρου, θα περιοριστεί το εύρος των θέσεων που μπορεί να φτάσει ο κινητήρας. Για να ελαχιστοποιήσουμε το μειονέκτημα αυτό, οι τιμές που θα ορίσουμε πρέπει να μην απέχουν πολύ από τα άκρα (0 - 1023).

Στην παρακάτω εικόνα βλέπουμε τα σημεία στα όποια μπορούμε να εφαρμόσουμε αυτό τον περιορισμό μέσω software:



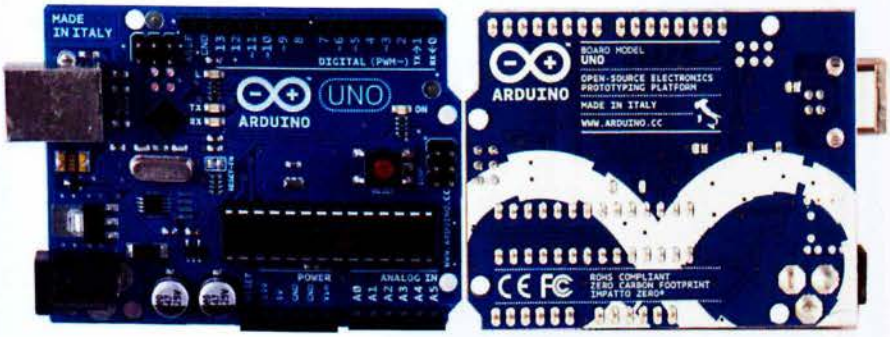
Minimum Value: Ορίζουμε μια τιμή για το ελάχιστο όριο περιστροφής.

Maximum Value: Ορίζουμε μια τιμή για το μέγιστο όριο περιστροφής.

Καταφέρνουμε έτσι να περιορίζουμε την γωνία περιστροφής του κινητήρα καθώς είτε ο κινητήρας θα σταματήσει να περιστρέφεται ή (επειδή το H-Bridge καθιστά εφικτή την αλλαγή της πόλωσης του κινητήρα) θα αρχίσει να περιστρέφεται αντίστροφα όταν η ανατροφοδότηση επιστρέψει τιμές εκτός του εύρους των 0-5V (η συμπεριφορά του εξαρτάται από τον κώδικα που έχουμε φορτώσει στο arduino).

3.3 Αποτελούμενα μέρη του συστήματος

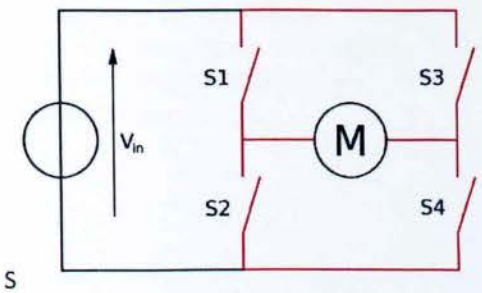
3.3.1 Arduino Uno Rev3 Board



Το Arduino αποτελεί την καρδιά του συστήματος καθώς όλες οι είσοδοι/έξοδοι, η ανάδραση, ο pid αλγόριθμος διαχειρίζονται από αυτό.

3.3.2 H-Bridge βασισμένο σε L298n

Το H-Bridge έχει διπλό ρόλο στο σύστημα μας. Πρώτον χρησιμοποιείται για να «οδηγήσει» τον κινητήρα, καθώς απαιτούνται περίπου 2amps για την κίνηση του. Το Arduino δεν θα μπορούσε να αντέξει σε τόσα ampere, καθώς το όριο του είναι 40mA. Δεύτερον μας δίνει την δυνατότητα να αντιστρέψουμε την φορά περιστροφής του κινητήρα, το οποίο επιτυγχάνεται όπως φαίνεται στην παρακάτω εικόνα:



Ένα H-Bridge αποτελείται από 4 διακόπτες (solid state ή μηχανικά), Όταν ο διακόπτες S1και S4 (σύμφωνα με την παραπάνω εικόνα) είναι κλειστοί (και οι S2 κα S3 είναι ανοιχτοί) μία θετική τάση θα εφαρμοστεί κατά μήκος του κινητήρα Ανοίγοντας τους διακόπτες S1 και S4 και κλείνοντας τους S2 και S3, αυτή η τάση αντιστρέφεται, επιτρέποντας αντίστροφη λειτουργία κίνησης του κινητήρα.

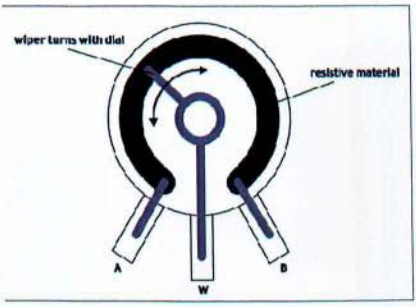
S1	S2	S3	S4	Αποτέλεσμα
1	0	0	1	Δεξιόστροφη κίνηση
0	1	1	0	Αριστερόστροφη κίνηση

3.3.3 Κινητήρας DC 12V

Πρόκειται για ένα κινητήρα που χρησιμοποιείται στους υαλοκαθαριστήρες αυτοκινήτων και λειτουργεί ως εξής. Μια σύνδεση μετατρέπει την περιστροφή του κινητήρα σε μια μπρος-πίσω κίνηση των υαλοκαθαριστήρων. Ένας ατέρμονος κοχλίας ελέγχει την δύναμη που δέχεται ο άξονας περιστροφής των υαλοκαθαριστήρων από τον κινητήρα. Ο κοχλίας καταφέρνει να μειώνει (ηλεκτρομειωτήρας) την ταχύτητα περιστροφής του κινητήρα 50 φορές πολλαπλασιάζοντας την ροπή 50 φορές.

3.3.4 Γραμμικό Ποτενσιόμετρο 47ΚΩ

Το ποτενσιόμετρο είναι αναλογικό ηλεκτρονικό εξάρτημα, που χρησιμοποιείται στα κυκλώματα ως μεταβλητή αντίσταση. Αποτελείται από αγώγιμη πλάκα σχήματος Ω, πάνω στην οποία γυρίζει, με τη βοήθεια ενός στροφέα, μια επαφή. Ανάλογα με την απόσταση της επαφής από την είσοδο του ρεύματος στο ποτενσιόμετρο μεταβάλλεται και η αντίσταση.

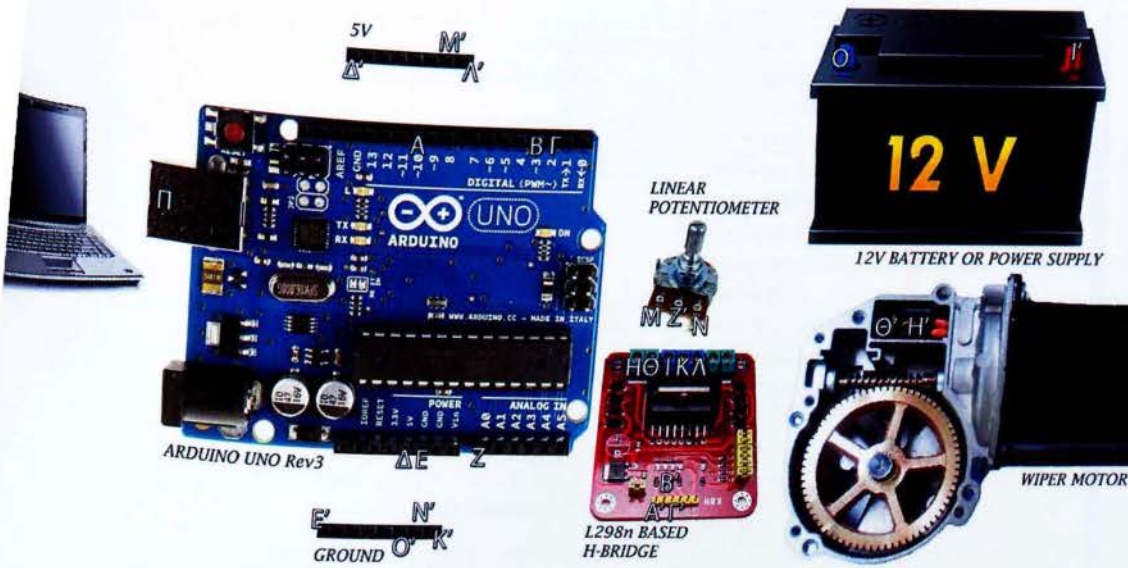


3.3.5 Τροφοδοσία

Το Arduino τροφοδοτείται από τα 5V της θύρας USB του υπολογιστή. Το H-Bridge χωρίζεται σε δύο μέρη. Στο λογικό, το οποίο τροφοδοτείται από τον ακροδέκτη των 5V του Arduino και το μέρος που τροφοδοτεί τον κινητήρα, το οποίο μπορεί να τροφοδοτείται από μια 12V μπαταρία αυτοκινήτου ή από τροφοδοτικό χαμηλού θορύβου.

3.4 Συνδεσμολογία κυκλώματος

Προς διευκόλυνση έχουμε χρησιμοποιήσει ζεύγη γραμμάτων (A-A', B-B' κ.ο.κ.), τα οποια συνδέονται φυσικά μεταξύ τους με χάλκινα καλώδια. Παρακάτω απεικονίζεται η εικόνα του κυκλώματος με τα ζεύγη γραμμάτων (επαφές καλωδίων), καθώς επίσης ακολουθεί και ένας πίνακας που αναφέρει τους ακροδέκτες αυτούς και την περιγραφή της χρησιμότητας τους.

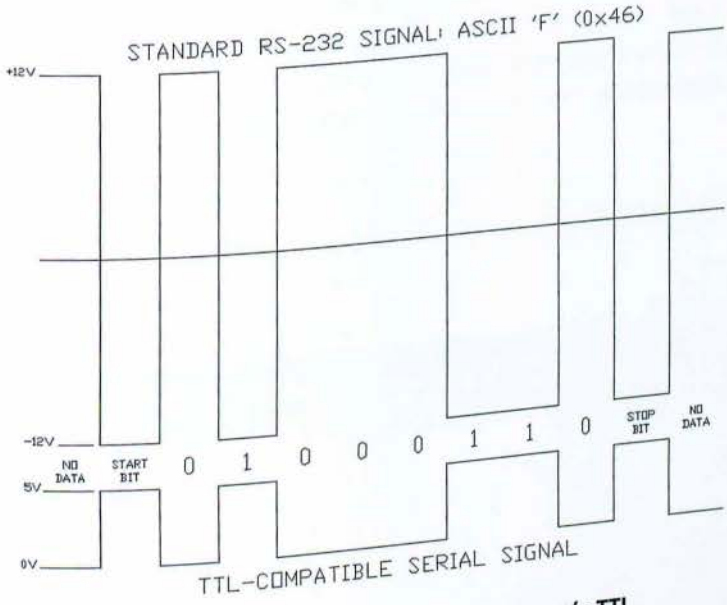


Σημείο 1	Σημείο 2	Περιγραφή
A: Arduino digital pin 10	A': H-Bridge ENCA pin	Σύνδεση των σημείων A-A', χρησιμοποιούμε το PWM enabled ακροδέκτη 10 του Arduino για να ελέγξουμε την ταχύτητα του κινητήρα
B: Arduino digital pin 3	B': H-Bridge IN1 pin	Σύνδεση των σημείων B-B', για την δεξιόστροφη κίνηση του κινητήρα, όταν το digital pin 3 του Arduino είναι HIGH
Γ: Arduino digital pin 2	Γ': H-Bridge IN2 pin	Σύνδεση των σημείων Γ-Γ', για την αριστερόστροφη κίνηση του κινητήρα, όταν το digital pin 2 του Arduino είναι HIGH
Δ: Arduino 5v pin	Δ': Σειρά 5V	Σύνδεση των σημείων Δ-Δ' για την δημιουργία μίας σειράς παροχής 5V από το Arduino, για την διευκόλυνση της συνδεσμολογίας
E: Arduino GND pin	E': Σειρά GROUND	Σύνδεση των σημείων E-E' για την δημιουργία μίας σειράς παροχής GROUND από το

		Arduino, για την διευκόλυνση της συνδεσμολογίας
Z: Arduino Analog In A3	Z': Ποτενσιόμετρο μεσαίο pin	Σύνδεση των σημείων Z-Z', για την λήψη της ανάδρασης από το ποτενσιόμετρο στο analog input 3 του Arduino
H: H-Bridge MotorA κλέμα	H': pin κινητήρα	Σύνδεση των σημείων H-H', για την τροφοδοσία του κινητήρα διαμέσου του H-Bridge, όποτε είναι αναγκαία η κίνηση του κινητήρα
Θ: H-Bridge MotorB κλέμα	Θ': pin κινητήρα	Σύνδεση των σημείων Θ-Θ', για την τροφοδοσία του κινητήρα διαμέσου του H-Bridge, όποτε είναι αναγκαία η κίνηση του κινητήρα
I: H-Bridge VMS κλέμα	I': +12V pin μπαταρίας	Σύνδεση των σημείων Γ-Γ', για την παροχή τροφοδοσίας +12V στο H-Bridge
K: H-Bridge GND κλέμα	K': Σειρά GROUND	Σύνδεση των σημείων K-K' στη σειρά παροχής GROUND
Λ: H-Bridge 5V κλέμα	Λ': Σειρά 5V	Σύνδεση των σημείων Λ-Λ', για την παροχή τροφοδοσίας 5V στο λογικό κομμάτι του H-Bridge που χρειάζεται για την επικοινωνία με το Arduino
M: Ποτενσιόμετρο αριστερό pin	M': Σειρά 5V	Σύνδεση των σημείων M-M' στη σειρά παροχής τροφοδοσίας 5V
N: Ποτενσιόμετρο δεξί pin	N': Σειρά GROUND	Σύνδεση των σημείων N-N' στη σειρά παροχής GROUND
O: -12V pin μπαταρίας	O': Σειρά GROUND	Σύνδεση των σημείων Γ-Γ', για την παροχή GROUND (-12V) στο H-Bridge
Π: Arduino USB θύρα	Π': Θύρα USB υπολογιστή	Σύνδεση των σημείων Π-Π', για την επικοινωνία του Arduino και του υπολογιστή, καθώς και την τροφοδοσία του Arduino

3.5 Σύνδεση και επικοινωνία Arduino/Υπολογιστή

Στο Arduino uno είναι σχεδιασμένο να αλληλοεπιδρά με σήματα που βρίσκονται μέσα στην κλίμακα 0 έως 5V DC των TTL chips. Ένα σήμα RS232 εναλλάσσεται ανάμεσα σε -12V DC και +12V DC πράγμα που θα μπορούμε να «κάψει» τα ολοκληρωμένα της πλακέτας του UNO. Επιπλέον, ένα RS232 σήμα είναι ανεστραμμένο (η κατάσταση αδράνειας, όταν δεν υπάρχουν δεδομένα ή όταν εμφανίζεται ένα λογικό 0, είναι +12V, ενώ το πλάτος του σήματος γίνεται -12V για λογικό 1). Συνεπώς, χρειάζεται ένα κύκλωμα που εκτός από την προσαρμογή του επιπέδου της τάσης σε ένα αποδεκτό εύρος, θα αντιστρέφει το σήμα έτσι ώστε ένα λογικό 0 να μετατρέπεται σε 0V DC και ένα λογικό 1 σε +5V DC.



Σύγκριση μεταξύ ενός RS232 σήματος και ενός TTL

Στο Arduino uno Rev3, το κλασσικό FTDI chip που είναι υπεύθυνο για την μετατροπή του TTL σε USB (δηλαδή για την επικοινωνία του ATMEGA328P και του υπολογιστή) έχει αντικατασταθεί από ένα ATMEGA16U2 chip. Αυτό έγινε λόγω της μεγαλύτερης ταχύτητας επικοινωνίας, του μικρότερου κόστους και της εξάλειψης της ανάγκης για εγκατάσταση οδηγών που χρειαζόντουσαν για το FTDI chip.

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να κατεβάσουμε το arduino environment για windows. Το λογισμικό παρέχεται δωρεάν και είναι ανοιχτού κώδικα. Μπορείτε να το κατεβάσετε από εδώ:

http://arduino.cc/en/Main/Software

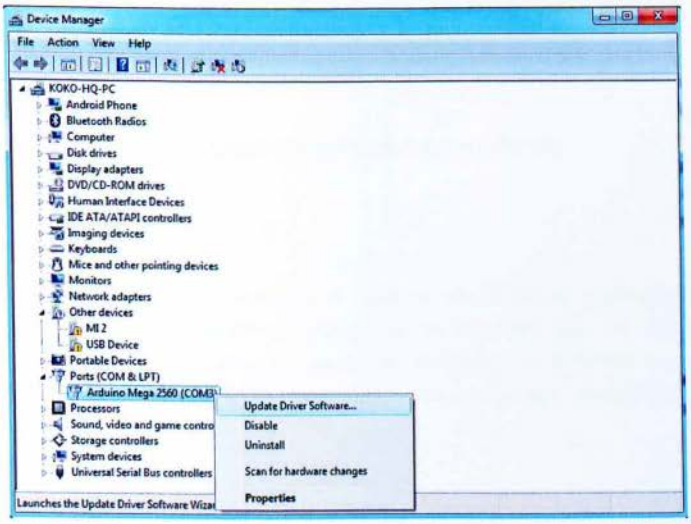
Stable version: http://arduino.googlecode.com/files/arduino-1.0.5-windows.zip

Beta version: http://downloads.arduino.cc/arduino-1.5.2-windows.zip

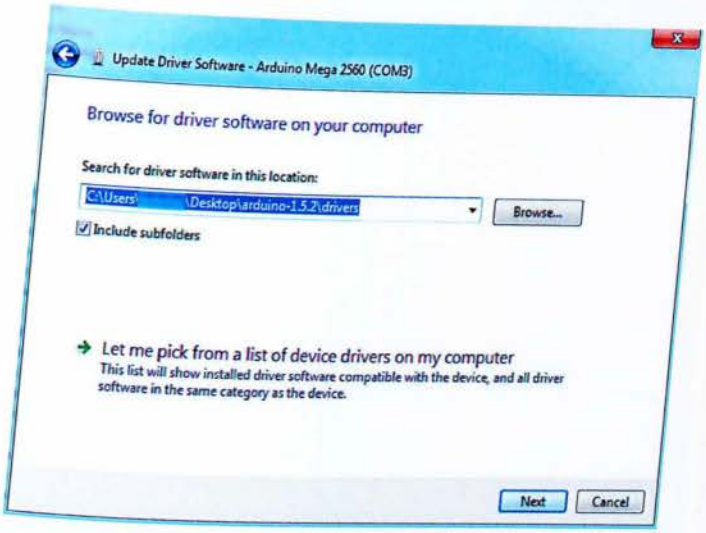
Αφού κάνουμε αποσυμπίεση το arduino environment, συνδέουμε το arduino με τον υπολογιστή χρησιμοποιώντας ένα καλώδιο USB τύπου A Male to B Male, μέσω του οποίου τροφοδοτούμε το arduino με 5V και ταυτόχρονα επιτρέπουμε στον υπολογιστή μας να το διαβάζει αλλά και να γράφει σε αυτό με την προϋπόθεση ότι έχουμε εγκαταστήσει σωστά τους ανάλογους drivers. Οι drivers βρίσκονται στο φάκελο drivers του arduino environment που αποσυμπιέσαμε προηγουμένως αλλά πρέπει να τους ορίσουμε χειροκίνητα στα windows για να τους εγκαταστήσει.

Πηγαίνουμε στο **device manager**.

Κάνουμε δεξί κλικ στην άγνωστη συσκευή, και επιλέγουμε **update driver software**.



Στην συνέχεια επιλέγουμε **Browse my computer for driver software**, εισάγουμε το path του φακέλου drivers και πατάμε επόμενο.



Αφού βεβαιωθούμε ότι βρέθηκε ο κατάλληλος driver για το μοντέλο του arduino που έχουμε, προχωράμε στην εγκατάσταση.

Τώρα είμαστε έτοιμοι να χρησιμοποιήσουμε το arduino environment για να φορτώσουμε τον πηγαίο κώδικα στο arduino.

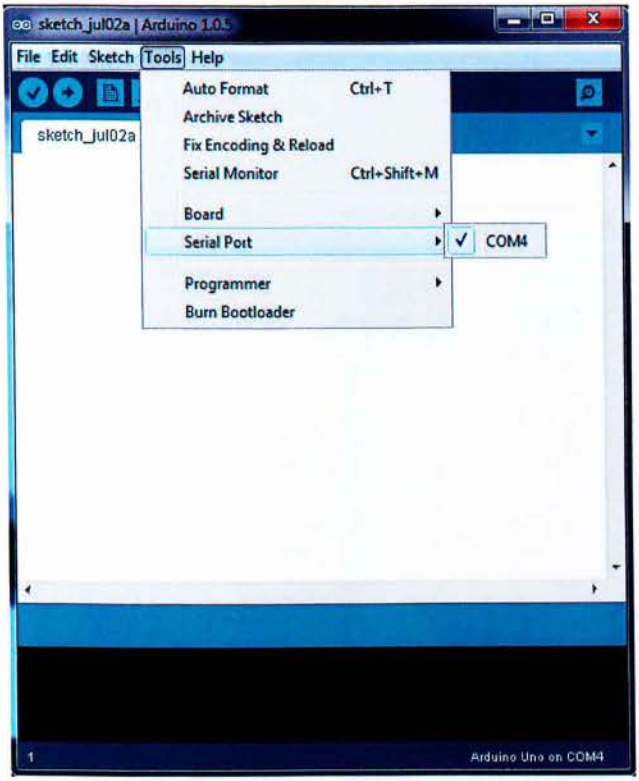
3.6 Φόρτωση και Εκτέλεση κώδικα στο Arduino

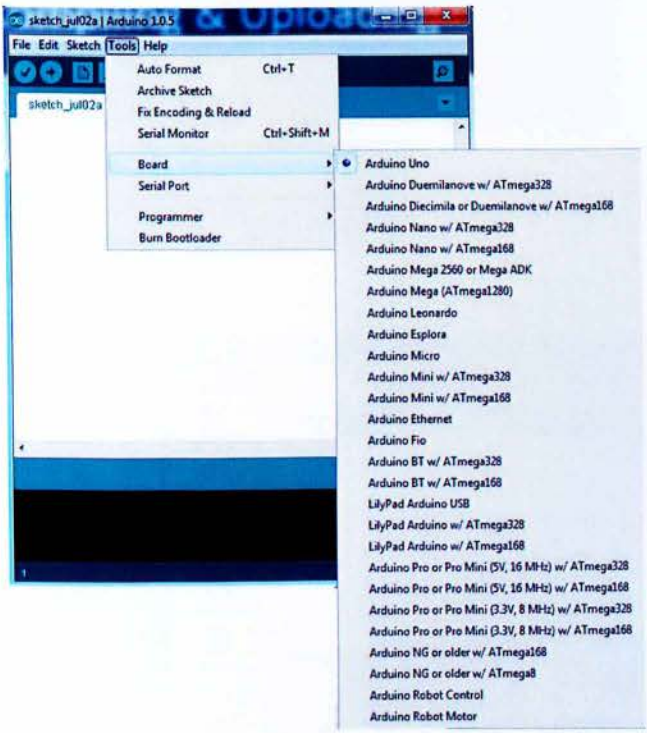
Για το compiling και την φόρτωση του κώδικα στο Arduino χρησιμοποιούμε το Arduino Environment, το οποίο μπορούμε να το κατεβάσουμε από την επίσημη ιστοσελίδα του Arduino δωρεάν. Τρέχοντας το Arduino.exe θα δούμε το παρακάτω παράθυρο στο οποίο μπορούμε να γράψουμε, τροποποιήσουμε, ανεβάσουμε κώδικα στο Arduino μας.

Δύο πράγματα που πρέπει να προσέξουμε πριν προχωρήσουμε στο compiling και την φόρτωση του κώδικα:

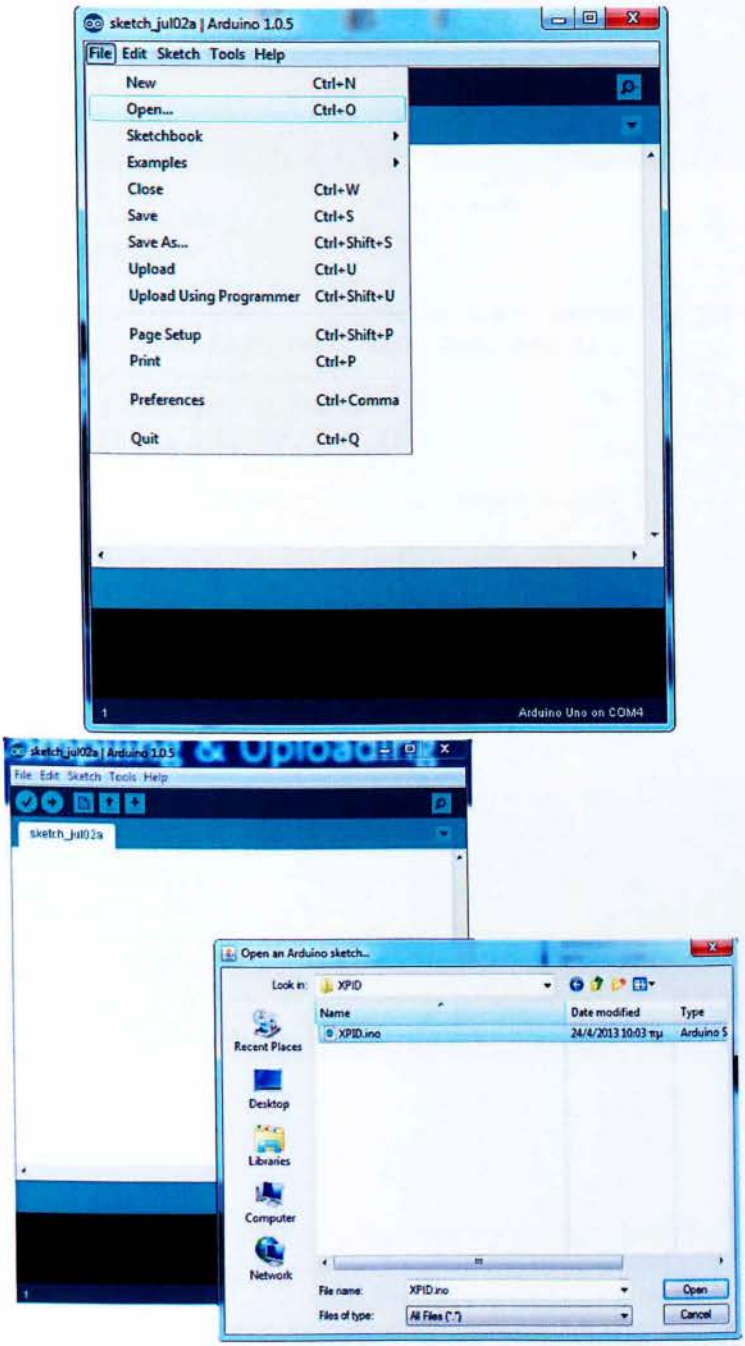
- Επιλογή της σωστής Arduino πλακέτας από το Tools -> Board
- Επιλογή της σωστής COM θύρας, που είναι συνδεδεμένο το Arduino.

Οι παρακάτω εικόνες θα μας βοηθήσουν να ρυθμίσουμε τις δύο παραπάνω πολύ σημαντικές παραμέτρους.

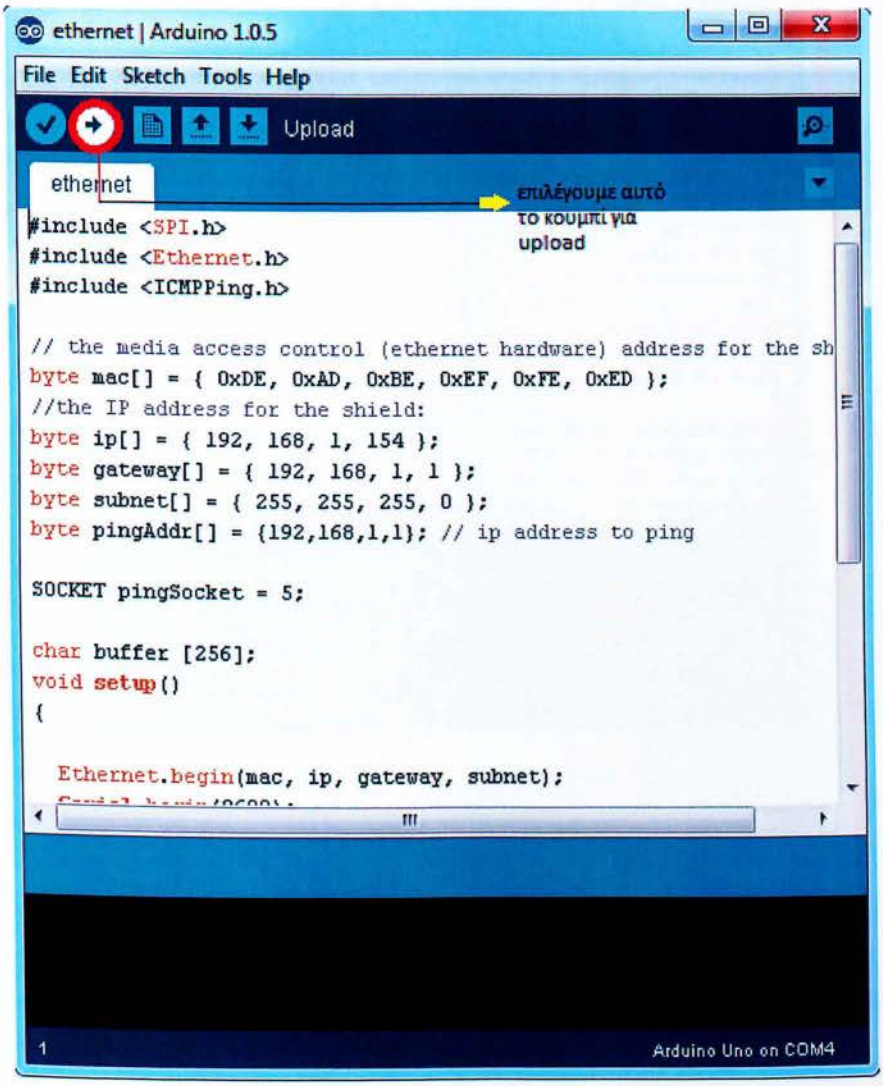

sketch_jul02a | Arduino 1.0.5
File Edit Sketch Tools Help
sketch_jul02a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
COM4
Programmer
Burn Bootloader
1
Arduino Uno on COM4


sketch_jul02a | Arduino 1.0.5
File Edit Sketch Tools Help
sketch_jul02a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

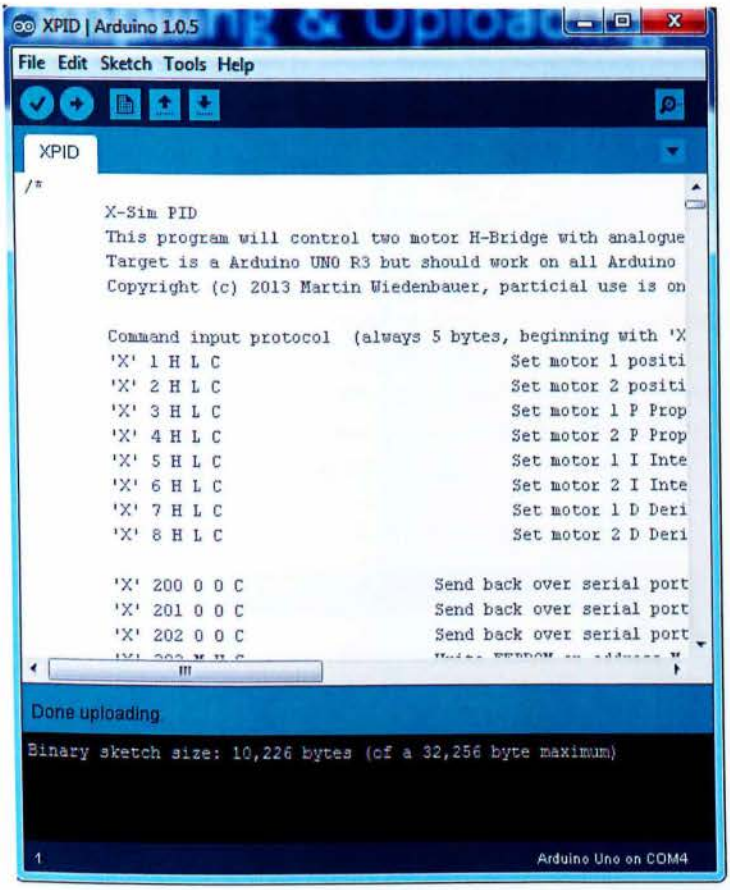
Ο κώδικας του Arduino βρίσκεται στο αρχείο XPID.ino. Αφού το ανοίξουμε, ακολουθώντας την διαδικασία που φαίνεται στις εικόνες είμαστε έτοιμοι να κάνουμε το compiling και την φόρτωση του κώδικα.



Για την διαδικασία του compiling και το uploading επιλέγουμε το κουμπί Upload, το οποίο φαίνεται στη διπλανή εικόνα.



Αν η διαδικασία ολοκληρωθεί επιτυχώς, θα δούμε την ένδειξη "Done Uploading" όπως φαίνεται και στην παρακάτω εικόνα.



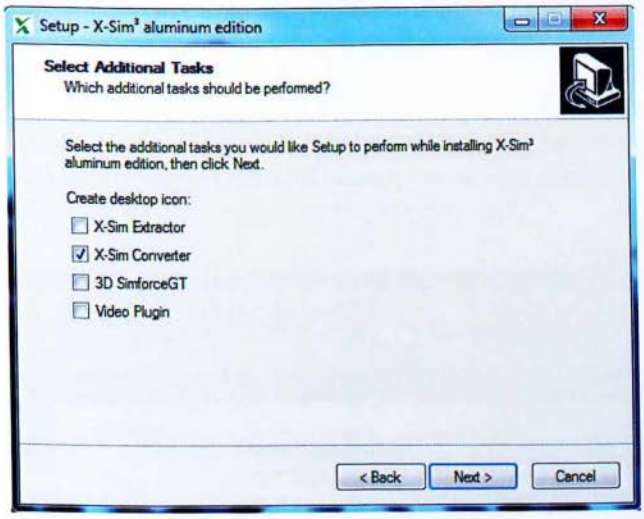
3.7 Γενικά-Η Διεπαφή X-Sim Convrerter

Το X-Sim είναι ένα λογισμικό ελέγχου που θα μας βοηθήσει να μετατρέψουμε τις αγνώστου μεγέθους τιμές, όπως η ταχύτητα σε μια κίνηση του υλικού (hardware). Επειδή κάθε προσομοιωτής ή υλικό έχει τα δικά του όρια μπορούμε να μεταβάλλουμε τα αποτελέσματα της εξόδου όπως επιθυμούμε. Ο μετατροπέας αρχίζει με την σελίδα " my simulator" που απλά δείχνει το φορτωμένο προφίλ και κάποιες σχετικές λεπτομέρειες με τον προσομοιωτή, όπως τις ώρες που ο εξομοιωτής λειτουργεί. Αυτή η εφαρμογή είναι σε θέση να λύσει τα περισσότερα από τα στοιχεία ελέγχου ενός ενεργοποιητή όπως συμπιεσμένο αέρα, υδραυλικά και ηλεκτρικά. Βασίζεται σε συστήματα προσέγγισης θέσης και χρησιμοποιεί μαθηματικά plugins για να αλλάξουμε το μέγεθος των τιμών εισόδου σε μία ενιαίου μεγέθους τιμή που μπορεί να μεταφερθεί σε έναν ενεργοποιητή με διαφορετική

ακρίβεια κίνησης. Έτσι είστε σε θέση να χρησιμοποιήσετε 8bits ανάλυση εισόδων σε ποτενσιόμετρα ή υψηλότερη ανάλυση με οποιοδήποτε ενεργοποιητή έχετε στην κατοχή σας.

3.7.1 Εγκατάσταση-Η Διεπαφή X-Sim Convrerter

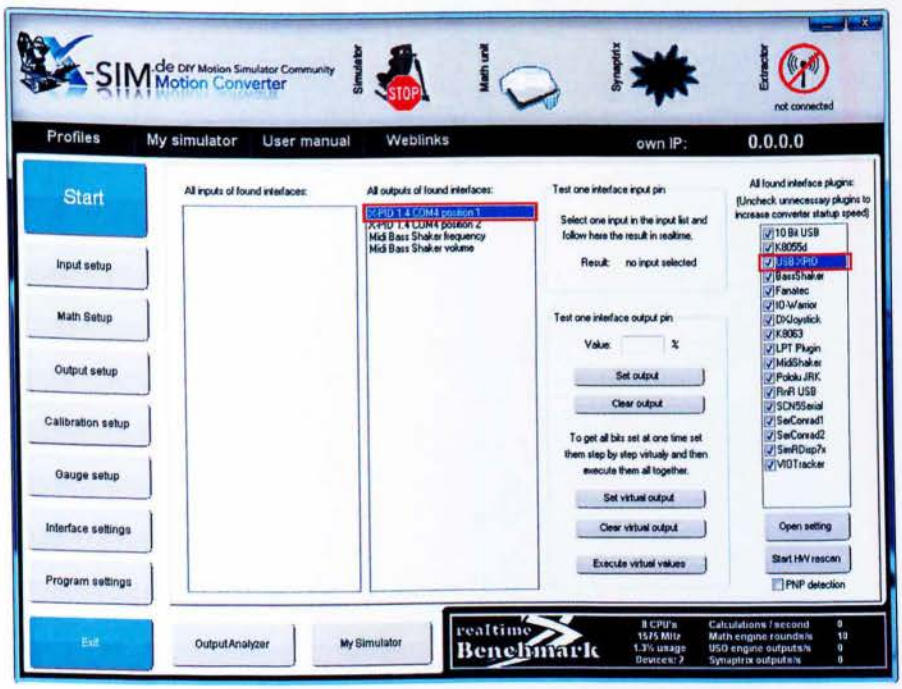
Κατεβάσουμε το πακέτο λογισμικού X-Sim από την ιστοσελίδα (απαιτείται εγγραφή μέλους): http://www.x-sim.de/forum//dm_eds/showcat.php?id=2. Κάνουμε αποσυμπίεση του αρχείου και τρέχουμε το εκτελέσιμο **X-Sim Setup 3.0.x.x beta aluminum.exe** για να προχωρήσουμε στην εγκατάσταση. Στο τελευταίο βήμα της εγκατάστασης επιλέγουμε μόνο το πρόγραμμα **X-Sim Converter** όπως φαίνεται στην παρακάτω εικόνα.



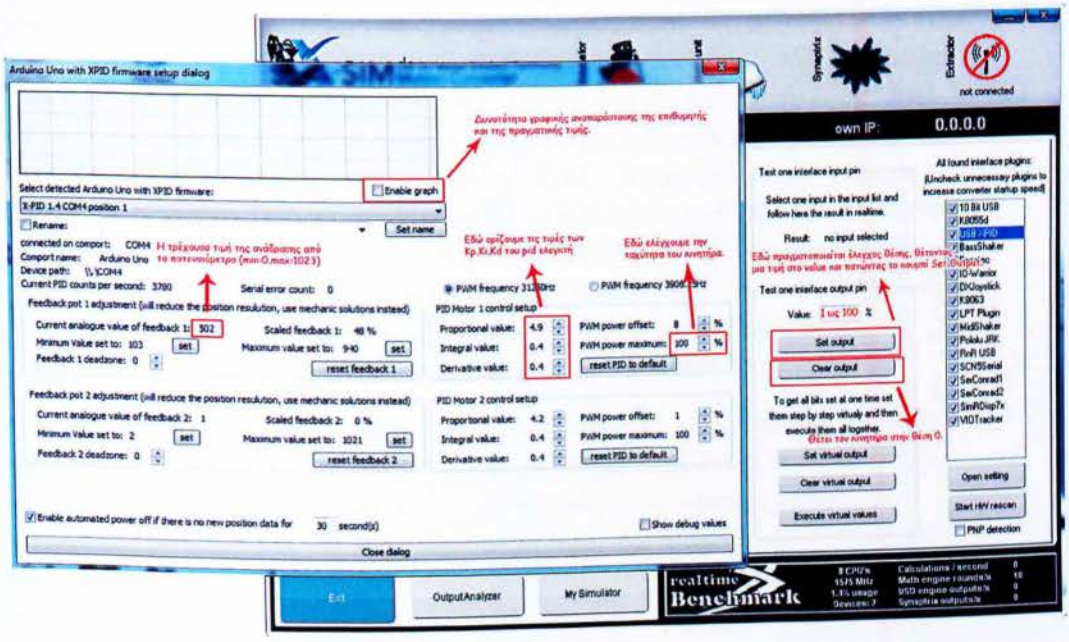
3.7.2 Τρόπος Λειτουργίας-Η Διεπαφή X-Sim Convrerter

Αφού ολοκληρωθεί η εγκατάσταση μπορούμε να τρέξουμε την διεπαφή X-Sim Converter από το εικονίδιο στην επιφάνεια εργασίας. Εφόσον έχει ολοκληρωθεί σωστά η συνδεσμολογία του συστήματος τότε ο κινητήρας πρέπει να κινηθεί στην αρχική του θέση (50% του εύρους των δυνατών θέσεων). Στο κεντρικό παράθυρο της εφαρμογής επιλέγουμε το κουμπί **Interface Settings**, όπως φαίνεται στην εικόνα:

Στη συνέχεια επιλέγουμε το interface **X-PID 1.4 COMx position 1** από τη λίστα και κάνουμε διπλό κλικ στο plugin **USB XPID** όπως φαίνεται στην εικόνα.



Παρακάτω βλέπουμε τις βασικές λειτουργίες που μας προσφέρει το X-Sim Converter, για έλεγχο της ταχύτητας και της θέσης του κινητήρα, καθώς και τον συντονισμό του PID.



3.7.3 Τρόπος Λειτουργίας-Δοκιμή των όρων του PID

Στην διεπαφή X-Sim Converter, έχουμε την δυνατότητα να δοκιμάσουμε την συμπεριφορά του συστήματος μας αλλάζοντας σε πραγματικό χρόνο τις τιμές P, I και D του PID αλγορίθμου. Σε αυτό το σημείο θα θέλαμε να επισημάνουμε πως, χρησιμοποιήσαμε έναν DC κινητήρα χωρίς την υποστήριξη ενσωματωμένης ανάδρασης. Έτσι το πρόβλημα αυτό λύθηκε προσκολλώντας ένα ποτενσιόμετρο στον άξονα περιστροφής του κινητήρα, με αυτό τον τρόπο αποκτήσαμε ανάδραση από το σύστημά μας. Από μηχανικής άποψης δεν αποτελεί την βέλτιστη λύση και γι' αυτό στις γραφικές μας θα παρατηρήσουμε αρκετές μικρές ταλαντώσεις. Οι ταλαντώσεις αυτές δεν προκαλούν ιδιαίτερο πρόβλημα στο σύστημά μας, δηλαδή το σύστημα παραμένει ευσταθές. Οι όροι του PID αλγόριθμου βοηθούν αρκετά στην εξομάλυνση αυτών των ταλαντώσεων, όμως επειδή όπως αναφέρθηκε και παραπάνω, το θέμα είναι μηχανικό και η παρουσία κάποιων μικρής κλίμακας ταλαντώσεων θα παραμένει εμφανής. Η μόνη λύση η οποία θα εξαφάνιζε την παρουσία των ταλαντώσεων αποτελεί η προσθήκη ενός κινητήρα με ενσωματωμένη ανάδραση.

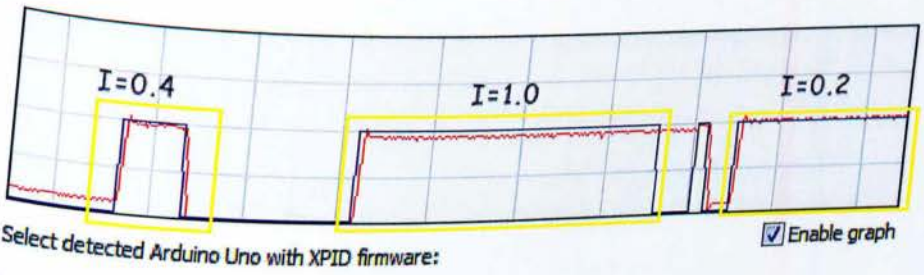
Οι όροι P, I και D μπορούν να πάρουν τιμές από 0 έως 10. Αυτό είναι ένα όριο που ορίζεται από την ίδια την διεπαφή. Πρέπει να επισημάνουμε ότι σε περίπτωση που θελήσουμε να θέσουμε τιμές που ενδέχεται να βγάλουν το σύστημα εκτός ευστάθειας, καλό θα είναι να περιορίσουμε το εύρος της ελάχιστής και της μέγιστης θέσης που μπορεί να προσεγγίσει το ποτενσιόμετρο, όπως αναφέρουμε και στην σελίδα 84.

Εφαρμόζοντας διαφορετικές τιμές για κάθε όρο κρατώντας τους άλλους δύο σταθερούς, πήραμε τα παρακάτω αποτελέσματα:

Για τον όρο P:

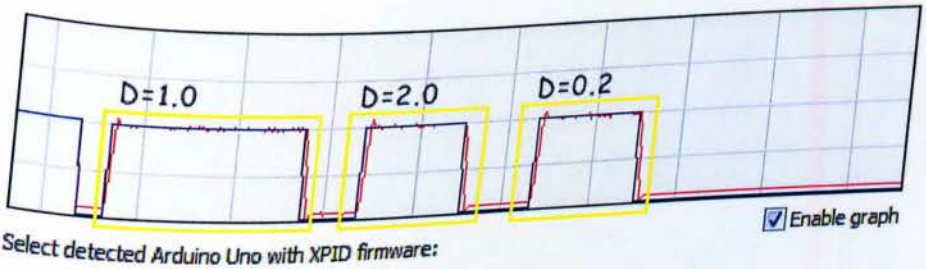
Παρατηρούμε ότι αλλάζοντας την τιμή του όρου P, έχουμε μεταβολές που αφορούν τον χρόνο ανύψωσης, την υπερύψωση και το μόνιμο σφάλμα.

Για τον όρο I:



Παρατηρούμε ότι αλλάζοντας την τιμή του όρου I, έχουμε μεταβολές που αφορούν το μόνιμο σφάλμα. Επίσης έχει μειωθεί η υπερύψωση.

Για τον όρο D:



Παρατηρούμε ότι αλλάζοντας την τιμή του όρου D, έχουμε μεταβολές που αφορούν το χρόνο αποκατάστασης και τις ταλαντώσεις της εξόδου.