ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη PID Controller με Arduino και Matlab

Νικητοπούλου Ν. Ελευθερία
Χαρμπή Π. Κυριακή

Εισηγητής: Δρ. Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΜΑΪΟΣ 2020

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη PID Controller με Arduino και Matlab

Ελευθερία Ν. Νικητοπούλου
Α.Μ. 43443

Κυριακή Π. Χαρμπή
Α.Μ. 43450

Εισηγητής:

Δρ. Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι Νικητοπούλου Ελευθερία του Νικολάου, με αριθμό μητρώου 43443 και Χαρμπή Κυριακή του Παναγιώτη, με αριθμό μητρώου 43450 φοιτητές του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβουμε την εκπόνηση της Πτυχιακής Εργασίας μας, δηλώνουμε ότι ενημερωθήκαμε για τα παρακάτω:

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

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

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

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστούμε θερμά,

Τον Καθηγητή μας Κ. Έλληνα για την ευγενική του συνεργασία κατά την διάρκεια εκπόνησης της πτυχιακής μας εργασίας και την οικογένεια μας για την πολύτιμη συμπαράσταση της κατά την διάρκεια των σπουδών μας.

ΠΕΡΙΛΗΨΗ

Βασικός σκοπός της εργασίας είναι η ανάπτυξη ενός PID Controller για τον έλεγχο της θερμοκρασίας χρησιμοποιώντας το MATLAB και το Arduino. Αρχικά, παρουσιάζονται τα βασικά χαρακτηριστικά του μικροελεγκτή Arduino καθώς και τα πλεονεκτήματα του συγκριτικά με τους υπόλοιπους μικροελεγκτές της αγοράς. Στην συνέχεια, αναφέρουμε την λειτουργικότητα των διάφορων ειδών PID ελεγκτών και πως ο κάθε ένας ελεγκτής επηρεάζει ένα κλειστό κύκλωμα. Ακόμα, γίνεται παρουσίαση του περιβάλλοντος MATLAB και των δυνατοτήτων του. Έπειτα, παρουσιάζεται η δημιουργία της κατασκευής μας, η σύνδεση του MATLAB με τον Arduino καθώς και όλα τα μοντέλα που αναπτύχθηκαν στο Simulink του Matlab με σκοπό την εύρεση των κατάλληλων όρων του PID Controller για την επίτευξη της λειτουργίας του συστήματος σύμφωνα με τις επιθυμητές προδιαγραφές.

ABSTRACT

The major purpose of this Project is the configuration of PID Controller for the temperature control using MATLAB and Arduino. At first, the basic characteristics of the microcontroller Arduino are presented and its advantages in compare with the rest microcontrollers in the market. Next, the functionality of the different types of PID controllers is reported and how each controller affects a closed system. Furthermore, the MATLAB environment and its capabilities are presented. Finally, we present the creation of our construction, the connection of MATLAB with Arduino as well as all the models developed in Simulink of Matlab in order to find the appropriate terms of the PID Controller to achieve the operation of the system according to the desired specifications.

Πίνακας περιεχομένων

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

PID: Proportional Integral Derivative

CPU: Central Processing Unit

RAM: Random - Access Memory

ROM: Read Only Memory

EEPROM: Electrically Erasable Programmable Read Only Memory

USB: Universal Serial Bus

PWM: Pulse Width Modulation

ICSP: In Circuit Serial Programming

MHz: Megaheartz

ADK: Accessory Development Kit

UART: Universal Asynchronous Receiver-Transmitter

Wi-Fi: Wireless Fidelity

GPS: Global Positioning System

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

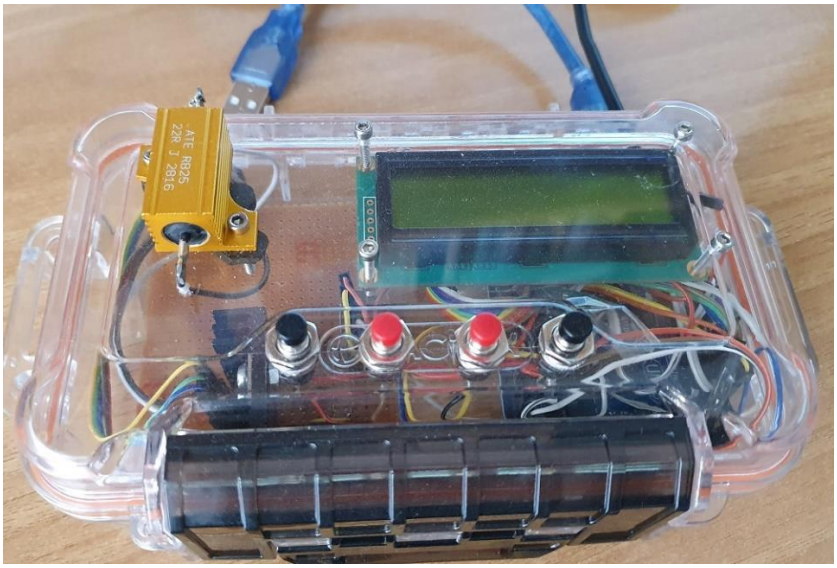
1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η εργασία φέρει τον τίτλο «Ανάπτυξη PID Controller με Arduino και Matlab». Είναι φανερό από τον τίτλο πως θα αναφερθούμε ξεχωριστά στον Arduino, στο MATLAB αλλά και στην συνύπαρξη αυτών των δύο για την ανάπτυξη ενός PID Controller.

Στόχος μας είναι να δημιουργήσουμε ένα «κουτί» μέσα στο οποίο θα διατηρείται η θερμοκρασία σταθερή, σύμφωνα με την επιθυμητή τιμή (Set Point) που εμείς θα ορίσουμε, και ταυτόχρονα η απόκριση του συστήματος να είναι όσο το δυνατόν πιο βέλτιστη/ομαλή γίνεται. Για να μπορέσουμε να ελέγξουμε αυτή την θερμοκρασία, θα χρειαστούμε feedback. Η ανατροφοδότηση αυτή θα επιτευχθεί χρησιμοποιώντας έναν αισθητήρα που θα μετράει την πραγματική θερμοκρασία του συστήματος. Το σύστημα θα υπολογίζει την διαφορά μεταξύ της επιθυμητής τιμής και της ανάδρασης από την έξοδο (error=setpoint – feedback) και μεταβάλλοντας τους όρους του PID Controller (K_p,K_i,K_d) μπορούμε να αλλάξουμε την έξοδο σύμφωνα με την ανατροφοδότηση. Το άθροισμα όλων αυτών των τιμών, των P, I, D ελεγκτών αποτελεί τον PID Controller. Είναι δική μας δουλειά να βρούμε τις σωστές σταθερές για καθένα από αυτά τα στοιχεία PID.

Για την κατασκευή του «κουτιού» θα χρειαστούμε ορισμένα εξαρτήματα. Αρχικά, πέρα από την πλακέτα του Arduino, θα χρειαστούμε στο κύκλωμά μας, μια αντίσταση η οποία θα θερμαίνεται και θα πρέπει να διατηρεί σταθερή την θερμοκρασία της και ίση με το Set Point και έναν αισθητήρα θερμοκρασίας όπου θα ελέγχει αυτή την θερμοκρασία. Ακόμα, θα χρειαστούμε μια LCD Display η οποία θα μας φανεί ιδιαίτερα χρήσιμη στην απεικόνιση των τιμών που θέλουμε να υπολογίσουμε (temperature, Kp,Ki,Kd). Τέλος, προγραμματίζουμε τον Arduino να εκτελέσει, με τον κατάλληλο κώδικα, τον έλεγχο των τιμών P, I, D για την δημιουργία του PID ελεγκτή.

Το MATLAB είναι ιδιαίτερα χρήσιμο για αυτήν την εργασία. Το πιο σημαντικό που διαθέτει είναι το *Support Package for Arduino Hardware* το οποίο μας επιτρέπει να δημιουργούμε και να εκτελούμε μοντέλα Simulink στον Arduino. Επομένως, μπορούμε να τρέξουμε απευθείας προγράμματα στον Arduino και έπειτα με την σειρά του, μέσω του Tx, να μεταφέρει στο MATLAB τις τιμές του αισθητηρίου της θερμοκρασίας. Με αυτόν τον τρόπο θα μπορούμε να απεικονίσουμε την απόκριση του συστήματος και να καταλήξουμε στις βέλτιστες τιμές των παραμέτρων του ελεγκτή PID.



Εικόνα 1: Κατασκευή-"Κουτί" Ελέγχου Θερμοκρασίας

1.2 Ιστορική Αναδρομή

1.2.1 Ιστορική Αναδρομή Arduino

Η επιθυμία για την δημιουργία συστημάτων που θα προσέφεραν πολλές δυνατότητες αλλά ταυτόχρονα θα ήταν σε μικρό μέγεθος οδήγησε στην κατασκευή κυκλωμάτων που συμπεριλάμβαναν όλες τις λειτουργίες του υπολογιστή. Η τεχνολογία αυτή, έφερε ως αποτέλεσμα, την ανάπτυξη του μικροεπεξεργαστή.

Ο μικροελεγκτής θεωρείται μια παραλλαγή ενός μικροεπεξεργαστή, καθώς είναι ένας τύπος μικροεπεξεργαστή που μπορεί να λειτουργήσει με ελάχιστα εξωτερικά συστήματα λόγω των πολλών υποσυστημάτων που διαθέτει. Ένας μικροελεγκτής είναι ένα ολοκληρωμένο κύκλωμα το οποίο,

συμπεριφέροντας όπως ένας τυπικός υπολογιστής, περιλαμβάνει διάφορα περιφερειακά όπως CPU, RAM, ROM για αποθήκευση δεδομένων και λογισμικού αντίστοιχα, μνήμη flash για μόνιμη αποθήκευση, θύρες εισόδου/εξόδου, μετατροπέα αναλογικού σε ψηφιακό σήμα και timers.

Το 2005, για λόγους εκπαιδευτικούς, κάνει την εμφάνισή του το Arduino, μια συσκευή κατάλληλα σχεδιασμένη για τον έλεγχο διαδραστικών προγραμμάτων από μαθητές. Ο καθηγητής Massimo Banzi στην πόλη της Ivrea, στην Ιταλία, με την βοήθεια του David Cuartielles αποφάσισαν να φτιάξουν έναν μικροελεγκτή εύκολο στην χρήση αλλά και φτηνότερο από όλα τα συστήματα εκείνης της περιόδου. Οι ιδρυτές, ονόμασαν το σχέδιο τους Arduino, από τον ιστορικό χαρακτήρα Arduin της Ivrea, όπου και ξεκίνησαν να παράγουν τις πρώτες πλακέτες.

Το “Serial Arduino” ήταν το πρώτο Arduino που κατασκευάστηκε και περιλάμβανε μια ATmega8 με άμεση σύνδεση RS-232 με τον μικροελεγκτή και όλα τα περιφερειακά του. Οι επόμενες εκδόσεις περιλάμβαναν FTDI USB μετατροπέα. Έπειτα, ακολούθησε το “Arduino Extreme” το οποίο αύξησε την ποσότητα των επιφανειακών εξαρτημάτων. Ακολούθως, έρχεται το Arduino Nuova Generazione για να μετατρέψει το ATmega8 σε ATmega168, μεταβαίνοντας σε έναν απλούστερο μετατροπέα USB. Οι βελτιώσεις συνεχίστηκαν με το Diecimila, με το Duemilanove και το Arduino Uno όπου ο μικροελεγκτής αναβαθμίστηκε σε ATmega328. Η τελευταία έκδοση του Arduino, Arduino Leonardo, εξαλείφει πλήρως τον μετατροπέα αναβαθμίζοντας τον σε ATmega3204, το οποίο περιέχει έναν ελεγκτή USB. Τέλος, έχουν φτιαχτεί εξειδικευμένες εκδόσεις για μικρότερες αλλά και μεγαλύτερες εγκαταστάσεις. Οι εκδόσεις Mini και Nano επιτρέπουν μικρότερες εγκαταστάσεις, ενώ οι ATmega1260 και ATmega2560 είναι κατάλληλες για μεγαλύτερη επεκτασιμότητα.

1.2.2 Ιστορική Αναδρομή Matlab

Το Matlab (Matrix Laboratory) είναι ένα πολυδιάστατο αριθμητικό περιβάλλον πληροφορικής και γλώσσα προγραμματισμού τέταρτης γενιάς. Ξεκίνησε να αναπτύσσεται στα τέλη της δεκαετίας του 1970 από τον Cleve

Moler, πρόεδρο του τμήματος Πληροφορικής στο Πανεπιστήμιο του Νέου Μεξικού**.** Το σχεδίασε για να δώσει στους μαθητές του πρόσβαση στο LINPACK και το EISPACK χωρίς να χρειάζεται να μάθουν το Fortran. Εξαπλώθηκε σύντομα και σε άλλα πανεπιστήμια και βρήκε ένα ισχυρό κοινό στην κοινότητα των εφαρμοσμένων μαθηματικών. Ο Jack Little, μηχανικός, εκτέθηκε σε αυτό κατά τη διάρκεια μιας επίσκεψης του Moler στο Πανεπιστήμιο του Στάνφορντ το 1983. Αναγνωρίζοντας τις εμπορικές δυνατότητές του, ενώθηκε με τον Moler και τον Steve Bangert. Έγραψαν εκ νέου το MATLAB στην γλώσσα C και ίδρυσαν το MathWorks το 1984 για να συνεχίσουν την ανάπτυξή του. Αυτές οι ανασυνταγμένες βιβλιοθήκες ήταν γνωστές ως JACKPAC. Το 2000, το MATLAB ξαναγράφηκε για να χρησιμοποιήσει ένα νεότερο σύνολο βιβλιοθηκών. Το MATLAB υιοθετήθηκε για πρώτη φορά από ερευνητές και επαγγελματίες στη μηχανική ελέγχου συστημάτων και γρήγορα εξαπλώθηκε σε πολλούς άλλους τομείς. Τώρα χρησιμοποιείται επίσης στην εκπαίδευση, ιδίως στη διδασκαλία της γραμμικής άλγεβρας και της αριθμητικής ανάλυσης, και είναι δημοφιλής στους επιστήμονες που ασχολούνται με την επεξεργασία εικόνων.

1.2.3 Ιστορική Αναδρομή PID Controller

Ο συνεχής έλεγχος, πριν γίνει κατανοητός και εφαρμόσιμος ο ελεγκτής PID, έχει τις ρίζες του στον φυγοκεντρικό ρυθμιστή, ο οποίος χρησιμοποιεί περιστρεφόμενα βάρη για τον έλεγχο μια διαδικασίας. Αυτό το είδος ελέγχου εφευρέθηκε τον 17ο αιώνα από τον Christiaan Huygens, ο οποίος ήθελε να ρυθμίσει το χάσμα μεταξύ των μυλόπετρών στους ανεμόμυλους ανάλογα με την ταχύτητα της περιστροφής με σκοπό να αντισταθμίσει την μεταβλητή ταχύτητα της τροφοδοσίας σιτηρών.

Όταν εφευρέθηκε η σταθερή ατμομηχανή υψηλής πίεσης, υπήρξε ανάγκη για τον αυτόματο έλεγχο της ταχύτητας. Δημιουργήθηκε τότε λοιπόν από τον James Watt, ο αυτοδιαμορφωμένος ρυθμιστής “conical pendulum", ένα σύνολο περιστρεφόμενων χαλύβδινων σφαιρών που συνδέονται με έναν κάθετο άξονα με βραχίονες σύνδεσης.

Ωστόσο, αυτό το είδος ελέγχου (γνωστός ως αναλογικός έλεγχος) παρέμενε μεταβλητός υπό συνθήκες μεταβαλλόμενου φορτίου αφού το σφάλμα μεταξύ της επιθυμητής ταχύτητας και της πραγματικής ταχύτητας αυξανόταν με την αύξηση του φορτίου. Τον 19ο αιώνα, ο Τζέιμς Κλερ Μέξγουελ περιέγραψε για πρώτη φορά την θεωρητική βάση για την λειτουργία των ρυθμιστών. Διερεύνησε τη μαθηματική βάση για τη σταθερότητα του ελέγχου και έκανε βήματα προς μια λύση, αλλά έκανε έκκληση στους μαθηματικούς να εξετάσουν το πρόβλημα. Το πρόβλημα αυτό, εξετάστηκε περαιτέρω από τον Edward Routh το 1874, τον Charles Sturm, και το 1895, τον Adolf Hurwitz. Όλοι οι παραπάνω βοήθησαν στην καθιέρωση των κριτηρίων σταθερότητας ελέγχου, όμως βελτιώθηκαν περαιτέρω, ιδίως από τον Αμερικανό επιστήμονα Willard Gibbs που ανέλυσε θεωρητικά τον ρυθμιστή του Watt.

Περίπου εκείνη την περίοδο, η εφεύρεση της τορπίλης Whitehead έθεσε ένα πρόβλημα ελέγχου το οποίο απαιτούσε ακριβή έλεγχο του βάθους λειτουργίας. Ύστερα από την ανεπαρκή προσπάθεια ελέγχου με την χρήση μόνο ενός αισθητήρα πίεσης βάθους, συνδυάστηκε η μέτρηση βάθους με ένα εκκρεμές που μετρούσε το εμπρόσθιο και το οπίσθιο βήμα της τορπίλης για να πραγματοποιηθεί ο έλεγχος του εκκρεμούς και του υδροστάτη. Ο έλεγχος πίεσης παρείχε μόνο αναλογικό έλεγχο, όπου αν το κέρδος ήταν πολύ υψηλό, θα καθιστούσε το σύστημα ασταθές στην διατήρηση βάθους. Το εκκρεμές μείωσε τις ταλαντώσεις ανιχνεύοντας τον ρυθμό αλλαγής του βάθους. Αυτός ο έλεγχος σήμερα είναι γνωστός ως διαφορικός. Αυτή η εξέλιξη έφερε την ονομασία «Το Μυστικό».

Ένα ακόμα πρώιμο παράδειγμα του PID Controller, αναπτύχθηκε για την καθοδήγηση των πλοίων το 1911 από τον Elmer Sperry. Η προσέγγιση του ήταν περισσότερο διαισθητική και όχι μαθηματική.

Το 1922 αναπτύχθηκε για πρώτη φορά επίσημη θεωρητική ανάλυση για τον PID ελεγκτή από τον μηχανικό Nicolas Minorsky. Ο Minorsky ερεύνησε και σχεδίασε την αυτόματη καθοδήγηση πλοίου του Πολεμικού Ναυτικού των ΗΠΑ, στηριζόμενος στην ανάλυση του σε παρατηρήσεις ενός πηδαλιούχου. Σημείωσε πως το πηδάλιο οδήγησε το πλοίο βάσει του συνδυασμού του τρέχοντος σφάλματος πορείας, του παρελθοντικού σφάλματος αλλά και του

ρυθμού αλλαγής αυτού. Στην συνέχεια, αυτή η θεωρία μοντελοποιήθηκε μαθηματικά από τον ίδιο. Στόχος του ήταν η σταθερότητα. Παρότι ο αναλογικός έλεγχος παρείχε σταθερότητα έναντι των μικρών διαταραχών, δεν επαρκούσε για την αντιμετώπιση μιας σταθερής διαταραχής και ήταν απαραίτητη η προσθήκη του ολοκληρωτικού όρου. Τέλος, προστέθηκε ο διαφορικός όρος για την βελτίωση της σταθερότητας και του ελέγχου.

Οι δοκιμές πραγματοποιήθηκαν στο USS New Mexico με τους ελεγκτές να ελέγχουν την γωνιακή ταχύτητα του πηδαλίου. Έπειτα από τις δοκιμές, διαπιστώθηκε πως ο έλεγχος PI απέδωσε σταθερή στροφή ± 2 °, ενώ η προσθήκη του διαφορικού όρου οδήγησε σε σφάλμα εκτροπής ± 1/6 °. Τελικά όμως, το Πολεμικό Ναυτικό δεν υιοθέτησε το σύστημα λόγω της αντίστασης του προσωπικού.

ΚΕΦΑΛΑΙΟ 2

Η Λειτουργία και τα Χαρακτηριστικά του Arduino σε συνδυασμό με την Λειτουργία του PID Controller

2.1 Εισαγωγή

2.1.1 Εισαγωγή στους Μικροελεγκτές

Η επιθυμία των κατασκευαστικών συστημάτων για την δημιουργία συστημάτων, τα οποία θα προσέφεραν περισσότερες λειτουργικές δυνατότητες και ταυτόχρονα θα ήταν σε μικρότερο μέγεθος, οδήγησε στην κατασκευή ολοκληρωμένων κυκλωμάτων που συμπεριλάμβαναν όλες τις λειτουργίες ενός τυπικού υπολογιστή. Η τεχνολογία αυτή, έφερε ως αποτέλεσμα την ανάπτυξη του μικροεπεξεργαστή (Singh, 2013).

Ο μικροελεγκτής θεωρείται μια παραλλαγή ενός μικροεπεξεργαστή, καθώς είναι ένας τύπος επεξεργαστή που μπορεί να λειτουργήσει με ελάχιστα εξωτερικά συστήματα λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Ένας μικροελεγκτής είναι ένα ολοκληρωμένο κύκλωμα, το οποίο συμπεριφέρεται ως ένας τυπικός υπολογιστής καθώς περιλαμβάνει διάφορα περιφερειακά όπως CPU, RAM, ROM για αποθήκευση δεδομένων και λογισμικού αντίστοιχα, μνήμη flash για μόνιμη αποθήκευση, θύρες εισόδου/εξόδου, μετατροπέα αναλογικού σε ψηφιακά σήματα και timers (Mehta, 1996).

Είναι απλούστερα σχεδιασμένος διότι είναι υπεύθυνος για την εκτέλεση μόνο μιας εργασίας ελέγχου ενός απλού συστήματος και έτσι περιλαμβάνει όλες τις απαιτούμενες λειτουργίες ενός ολοκληρωμένου κυκλώματος. Χρησιμοποιείται σε όλα τα ενσωματωμένα συστήματα ελέγχου χαμηλού και

μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρικές συσκευές και ηλεκτρονικά καταναλωτικά προϊόντα. Στην σημερινή εποχή, το κάθε προϊόν που αλληλοεπιδρά με έναν χρήστη περιλαμβάνει έναν μικροελεγκτή, ο οποίος παίζει το ρόλο του «εγκεφάλου» των ηλεκτρονικών κυκλωμάτων (Faugel, Bobkov, 2013).

Ο μικροελεγκτής είναι μια γρήγορη συσκευή, φυσικά όχι όσο ο υπολογιστής, με αποτέλεσμα κάθε εντολή που εκτελείται σε αυτόν να γίνεται χωρίς καθυστερήσεις. Έχει κάποια χαρακτηριστικά που τον καθιστά καταλληλότερο για την χρήση του σε εφαρμογές έναντι της χρήσης των επιμέρους στοιχείων που τον αποτελούν. Το πιο βασικό χαρακτηριστικό που λαμβάνει ένας σχεδιαστής υπόψιν είναι το χαμηλό του κόστος. Επίσης, ένας μικροελεγκτής παρέχει αυτονομία καθώς έχει ενσωματωμένα όλα τα περιφερειακά υποσυστήματα που χρειάζεται για να λειτουργήσει. Ακόμα, επιτυγχάνει έλεγχο και μετρήσεις σε πραγματικό χρόνο χωρίς την απαίτηση επιπλέον λογισμικού. Τέλος, η ολοκλήρωση των βασικών στοιχείων που τον αποτελούν, τον καθιστούν πλέον ένα υπολογιστικό σύστημα μικρού μεγέθους (Singh, 2013).

Όσο αφορά τον προγραμματισμό του μικροελεγκτή οι πιο διαδεδομένες γλώσσες είναι η C, η C++ και οι παραλλαγές τους. Η Assembly χρησιμοποιείται σε τμήματα του λογισμικού όπου απαιτείται μικρό μέγεθος χρησιμοποιούμενης μνήμης ή ταχύτητα. Όμως, η Assembly έχει εκτοπιστεί από τις περισσότερες εφαρμογές καθώς ο προγραμματισμός της C είναι ολοφάνερα πιο εύκολος από την Assembly και οι απαιτήσεις σε λειτουργικότητα ολοένα και μεγαλώνουν. Τέλος, μερικοί από τους γνωστότερους κατασκευαστές μικροελεγκτών είναι οι: Intel, Atmel, Microchip, Epson, NAC κ.ά.

2.1.2 Εισαγωγή στον Arduino

Το Arduino είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring. Αναφερόμενοι στην γλώσσα Wiring, αναφερόμαστε επί της ουσίας στην γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++.

Το Arduino βασίζεται στο 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. Ουσιαστικά, 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. Αφού κατασκευαστεί, μπορεί να φερθεί 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.

Εικόνα 2: Το λογότυπο του Arduino

2.2 Εκδόσεις Arduino

Υπάρχουν πολλές διαφορετικές εκδόσεις πλακετών του Arduino, κάθε μια από τις οποίες έχει διαφορετικά χαρακτηριστικά και χρησιμοποιείται αναλόγως για διαφόρους σκοπούς. Παρακάτω, θα βρείτε κάποια μοντέλα του Arduino που κυκλοφορούν στην αγορά. Στο δικό μας Project χρησιμοποιήσαμε τον Arduino UNO.

- Arduino UNO

Είναι η πιο διαδεδομένη πλακέτα με αρκετά χαμηλό κόστος και χρησιμοποιείται για απλές εφαρμογές. Χρησιμοποιεί τεχνολογία ATmega8U2 και είναι προγραμματισμένο ως σειριακός μετατροπέας. Έχει 14 ψηφιακές ακίδες εισόδου / εξόδου από τις οποίες 6 μπορούν να χρησιμοποιηθούν ως έξοδο PWM, 6 αναλογικές εισόδους, έναν ενισχυτή ήχου 16 MHz, μια σύνδεση USB, μια υποδοχή τροφοδοσίας, μια ICSP και ένα κουμπί Reset.

Εικόνα 3: Arduino UNO

- Arduino Mega 2560

Το Arduino Mega 2560 χρησιμοποιεί τεχνολογία Surface-mounted. Τα πλεονεκτήματα του είναι η ολική μνήμη του στα 256KB καθώς και οι 54 ψηφιακές ακίδες εισόδου/εξόδου, οι 16 αναλογικές είσοδοι καθώς και ένα ρολό 84MHz, 4 UARTs (hardware serial ports), 1 κρυσταλλικός ταλαντωτής 16 MHz, υποδοχή USB, υποδοχή τροφοδοσίας ρεύματος, 1 ICSP και ένα κουμπί Reset.

Εικόνα 4:Arduino Mega 2560

- ArduinoADK

Η συγκεκριμένη πλακέτα είναι βασισμένη στο Arduino Mega, όμως διαθέτει επιπλέον μια θύρα USB για σύνδεση με τα τηλέφωνα λογισμικού Android.

Εικόνα 5: ArduinoADK

- Arduino Leonardo

Είναι μια πλακέτα, η οποία με την εισαγωγή Atmega 32U4 chip μπορεί να χρησιμοποιηθεί σαν ψηφιακό πληκτρολόγιο ή ποντίκι και εξαλείφει την ανάγκη για συνδεσιμότητα μέσω USB. Έχει 20 ψηφιακές ακίδες εισόδου / εξόδου από τις οποίες 7 μπορούν να χρησιμοποιηθούν ως έξοδο PWM και 12 ως αναλογικές είσοδοι, έναν ταλαντωτή κρυστάλλου 16 MHz, μια σύνδεση micro USB, μια υποδοχή τροφοδοσίας, μια ICSP και ένα κουμπί Reset.

Εικόνα 6: Arduino Leonardo

❖ Arduino Yun

Αυτή η πλακέτα είναι βασισμένη στο Arduino Leonardo, όμως διαθέτει ενσωματωμένη κεραία Wi-Fi και υποδοχή για καλώδιο RJ45. Είναι ένας συνδυασμός Arduino και Linux.

Εικόνα 7: Arduino Yun

❖ Arduino Ethernet

Η πλακέτα αυτή επιτρέπει τη σύνδεση με το διαδίκτυο και η τροφοδοσία γίνεται μέσω καλωδίου Ethernet. Επίσης διαθέτει υποδοχή για κάρτα micro SD. Έχει 14 ψηφιακές ακίδες εισόδου / εξόδου, 6 αναλογικές εισόδους, έναν ταλαντωτή κρυστάλλου 16 MHz, μια υποδοχή τροφοδοσίας, μια ICSP και ένα κουμπί Reset.

Εικόνα 8: Arduino Ethernet

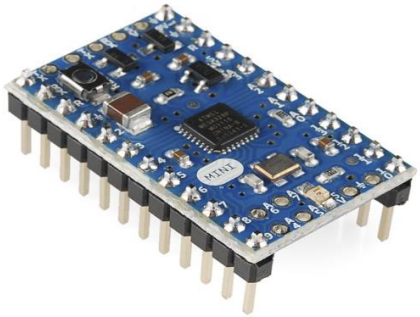
❖ Arduino Esplora

Η πλακέτα Esplora Περιέχει ενσωματωμένους αισθητήρες για ήχο, φως, θερμοκρασία και επιτάχυνση και έχει εμφάνιση που παραπέμπει σε χειριστήριο κονσόλας βιντεοπαιχνιδιών.

Εικόνα 9: Arduino Esplora

❖ Arduino Mini

Το Arduino Mini προορίζεται για εγκατάσταση σε αντικείμενα ή εκθέσεις και δεν διαθέτει τοποθετημένες ακίδες επιτρέποντας έτσι άμεση συγκόλληση των καλωδίων. Έχει 14 ψηφιακές ακίδες εισόδου / εξόδου από τις οποίες 6 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM, 8 αναλογικές εισόδους και έναν ταλαντωτή κρυστάλλου 16 MHz.



Εικόνα 10: Arduino Mini

2.3 Arduino Shields

Τα shields είναι τα εξαρτήματα του Arduino τα οποία είναι σχεδιασμένα έτσι ώστε να μπορούν να συνδεθούν πάνω απευθείας πάνω στα pin του Arduino επεκτείνοντας έτσι τις δυνατότητες του. Ορισμένα shields, διαθέσιμα στην αγορά, είναι τα ακόλουθα :

1. Arduino Wi-Fi Shield: Συνδέει ασύρματα το Arduino στο διαδίκτυο.
2. Arduino Ethernet Shield: Συνδέει το Arduino σε ένα LAN ή στο Internet μέσω ενός RJ45 καλώδιο.
3. GPS Shield: Δίνει στο Arduino την δυνατότητα εντοπισμού στίγματος.
4. Bluetooth Shield: Συνδέει το Arduino με μια άλλη συσκευή μέσω Bluetooth.
5. Motor Shield: Επιτρέπει την οδήγηση δύο DC κινητήρων από την ίδια συσκευή, ελέγχοντας την ταχύτητα και την κατεύθυνση του καθενός ξεχωριστά.

2.4 Χαρακτηριστικά Πλακετών Arduino

Η πιο διαδεδομένη πλακέτα Arduino είναι το Arduino Uno. Πολλές πλακέτες έχουν παρόμοια χαρακτηριστικά με κάποιες μικροδιαφορές αναλόγως με το πού θέλουμε να χρησιμοποιηθεί. Το Arduino Uno βασίζεται στον μικροεπεξεργαστή ATmega328P της Atmel, ο οποίος αποτελείται από έναν 8-bit μικροελεγκτή που είναι χρονισμένος στα 16MHz με δυνατότητα επέκτασης στα 20MHz. Μέσω αυτού του εξωτερικού 16MHz κρυστάλλου ενεργεί ως πηγή ρολογιού, στέλνοντας σήματα ON-OFF τα οποία αλλάζουν τη κατάσταση του συστήματος. Περιλαμβάνει 14 ψηφιακές ακίδες εισόδου / εξόδου (από τις οποίες 6 μπορούν να χρησιμοποιηθούν ως έξοδο PWM), 6 αναλογικές εισόδους, μια σύνδεση USB, μια υποδοχή τροφοδοσίας, μια κεφαλίδα ICSP και ένα κουμπί επαναφοράς(Reset). Περιέχει όλα τα απαραίτητα στοιχεία για την υποστήριξη του μικροελεγκτή. Το μόνο που χρειάζεται για να ξεκινήσει η λειτουργία του Arduino είναι απλώς να συνδεθεί με έναν υπολογιστή. Η σύνδεσή του στον υπολογιστή γίνεται μέσω της θύρας USB που διαθέτει και χρησιμοποιείται για τη μεταφορά των προγραμμάτων που θα «τρέξουν» στην πλακέτα καθώς και για την επικοινωνία του Arduino με τον υπολογιστή.

Για την εκπόνηση της πτυχιακής μας εργασίας χρησιμοποιήθηκε το Arduino Uno. Παρακάτω παρουσιάζονται τα τεχνικά του χαρακτηριστικά :

Πίνακας 1: Χαρακτηριστικά Arduino

Μικροελεγκτής	ATmega328	**Μνήμη Flash**	32KB
Τάση λειτουργίας	5V	**Μνήμη SRAM**	2KB
Τάση εισόδου	7-12V	**Μνήμη EEPROM**	1KB
Όρια τάσης εισόδου	6-20V	**Ταχύτητα ρολογιού**	16MHz
Ψηφιακοί ακροδέκτες I/O	14, (6 PWM έξοδοι)	**Ισχύς συνεχόμενου ρεύματος ανά ακροδέκτη**	40mA
Αναλογικοί ακροδέκτες εισόδου	6	**Ισχύς συνεχόμενου ρεύματος για ακροδέκτη τάσης 3.3V**	50mA

Η τροφοδοσία του Arduino πρέπει να κυμαίνεται από 7V-12V και μπορεί να προέρχεται από έναν κοινό μετασχηματιστή, μπαταρίες είτε μέσω της θύρας USB που διαθέτει και συνδέεται κατευθείαν με τον υπολογιστή. Οι ακροδέκτες τροφοδοσίας είναι οι ακόλουθοι :

- Vin: Ακροδέκτης για μη σταθεροποιημένη τάση. Η τάση της εισόδου της πλακέτας όταν χρησιμοποιείται εξωτερική πηγή τροφοδοσίας.
- 5V: Ακροδέκτης σταθεροποιημένης τάσης 5V ή η ρυθμισμένη τάση που δίνεται μέσω του Vin. Η τάση που χρησιμοποιείται από τον μικροελεγκτή.
- 3.3V: Η τάση αυτή παράγεται από το ολοκληρωμένο FTDI με μέγιστο ρεύμα 50mA.
- GND: Ακροδέκτης γείωσης.

Στον Πίνακα 1 παρατηρούμε πως το Arduino διαθέτει τρεις ομάδες μνήμης. Ας δούμε αναλυτικά την χρησιμότητα της κάθε μιας:

- Μνήμη Flash: **32kb**, εκ των οποίων τα 2kb χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Για την εγκατάσταση των προγραμμάτων μας στον μικροελεγκτή μέσω της θύρας USB είναι απαραίτητο το firmware ή αλλιώς bootloader. Τα υπόλοιπα 30kb της μνήμης είναι διαθέσιμα για την αποθήκευση του κώδικα που θέλουμε να ανεβάσουμε στο Arduino, αφού πρώτα μεταγλωττιστεί στον υπολογιστεί. Ακόμα, δεν χάνει τα περιεχόμενα της σε περιπτώσεις απώλειας τροφοδοσίας ή επανεκκίνησης.
- Μνήμη SRAM: **2kb.** Είναι η ωφέλιμη μνήμη που μπορεί να χρησιμοποιηθεί από τα προγράμματα για την αποθήκευση μεταβλητών. Η μνήμη SRAM, όπως και ο υπολογιστής, χάνει τα δεδομένα της όταν σταματήσει η παροχή ρεύματος στο Arduino ή γίνει Reset.
- Μνήμη EEPROM: **1kb.** Χρησιμοποιείται για εγγραφή και ανάγνωση των δεδομένων από τα προγράμματα ανά byte. Θα μπορούσαμε να πούμε ότι η EPPROM είναι το ανάλογο του σκληρού δίσκου, καθώς σε αντίθεση με την SRAM, δεν χάνει τα περιεχόμενα της σε περιπτώσεις απώλειας τροφοδοσίας ή επανεκκίνησης.

Το Arduino διαθέτει σειριακό interface. Ο μικροελεγκτής ATmega328 υποστηρίζει σειριακή επικοινωνία, η οποία προωθείται από τον Arduino μέσω ενός ελεγκτή Serial-over-USB και έτσι γίνεται εφικτή η σύνδεση με τον υπολογιστή μέσω USB. Η σύνδεση αυτή χρησιμοποιείται για τη μεταφορά των

προγραμμάτων που θα «τρέξουν» στην πλακέτα καθώς και για την επικοινωνία του Arduino με τον υπολογιστή.

Στο πάνω μέρος της πλακέτας βρίσκονται 14 θηλυκά pin τα οποία λειτουργούν σαν ψηφιακές είσοδοι και έξοδοι. Το κάθε ένα από αυτά τα pin λειτουργεί στα 5V και μπορεί να δεχτεί ή να παρέχει το πολύ 20mA. Από αυτούς τους 14 ακροδέκτες, οι οποίοι είναι αριθμημένοι από το 0 έως το 13, ορισμένοι εκτός από την λειτουργία I/O έχουν και άλλη χρησιμότητα. Ας δούμε αναλυτικά τις επιπρόσθετες λειτουργίες τους:

- Οι ακροδέκτες 0 και 1 λειτουργούν ως RX και TX της σειριακή θύρας όταν το πρόγραμμά μας ενεργοποιεί τη σειριακή θύρα. Έτσι, όταν το πρόγραμμά μας στέλνει δεδομένα στην σειριακή (UART), αυτά προωθούνται στην θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στον ακροδέκτη 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή. Αυτό όμως σημαίνει ότι αν στο πρόγραμμά μας ενεργοποιήσουμε το σειριακό interface, θα χάσουμε 2 ψηφιακές εισόδους/εξόδους.
- Οι ακροδέκτες 2 και 3 λειτουργούν και ως εξωτερικά interrupt. Μπορούμε δηλαδή, να τα ρυθμίσουμε μέσα από το πρόγραμμά μας ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, να σταματάει άμεσα η κανονική ροή του προγράμματος και να εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.
- Οι ακροδέκτες 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM, δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων, ώστε να έχουμε ψηφιακούς παλμούς.
- Ο ακροδέκτης 8 χρησιμοποιείται για τον υπολογισμό συχνότητας ενός σήματος.
- Οι ακροδέκτες 12 και 13 χρησιμοποιούνται για την επικοινωνία του Arduino με τις εξωτερικές περιφερειακές μονάδες.

Στην κάτω πλευρά του Arduino, θα βρούμε μια ακόμη σειρά από 6 ακροδέκτες, αριθμημένους από το 0 ως το 5 με την ένδειξη ANALOG IN. Ο καθένας από αυτούς λειτουργεί ως αναλογική είσοδος κάνοντας χρήση του ADC που είναι ενσωματωμένο στον μικροελεγκτή. Όμως, με την κατάλληλη εντολή μέσα από τον κώδικα μας, μπορούν να μετατραπούν σε ψηφιακούς ακροδέκτες, όπου και θα μετονομαστούν από 0-5 σε 14-19 αντίστοιχα.

Ακόμα, πάνω στην βρίσκονται ένας διακόπτη και 4 μικροσκοπικά LED. Ο διακόπτης, ο οποίος έχει την σήμανση Reset, χρησιμοποιείται για την εκκίνηση και τη λειτουργία της πλακέτας και το LED με την ένδειξη POWER μας δείχνει το αν βρίσκεται σε λειτουργία ή όχι η πλακέτα. Τα 2 LED με τις ενδείξεις TX και RX ανάβουν όταν το Arduino στέλνει ή λαμβάνει δεδομένα μέσω USB. Το 4^{Ο} LED ενσωματώθηκε από τους κατασκευαστές προκειμένου να μπορεί να γίνει οποιαδήποτε δοκιμή λειτουργίας μέσω του pin 13. Ακόμα και αν δεν έχουμε συνδέσει τίποτα στο pin 13, δίνοντας του την τιμή HIGH από το πρόγραμμα μας, το LED αυτό θα ανάψει.

Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino είναι μια εφαρμογή γραμμένη σε Java, η οποία χρησιμοποιείται σε πολλές πλατφόρμες, και προέρχεται από το IDE για τη γλώσσα προγραμματισμού Processing και το σχέδιο Wiring. Περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα με τα εξής χαρακτηριστικά: επισήμανση σύνταξης, συνδυασμός αγκύλων, αυτόματη εσοχή, καθώς επίσης είναι σε θέση να μεταγλωττίζει και να φορτώνει προγράμματα στην πλακέτα με ένα μόνο κλικ. Τα προγράμματα που γράφονται για το Arduino φέρουν την ονομασία Sketch.

Το περιβάλλον προγραμματισμού ενός Arduino ενδείκνυται για αρχάριους, αλλά είναι ταυτόχρονα και ευέλικτο και για πιο προχωρημένους χρήστες. Τα Arduino προγράμματα είναι γραμμένα σε γλώσσα προγραμματισμού C ή C++. Το Arduino IDE παρέχει μια βιβλιοθήκη λογισμικού που ονομάζεται «Wiring» ,από το αρχικό σχέδιο Wiring, γεγονός που καθιστά πολλές κοινές λειτουργίες εισόδου/εξόδου πολύ πιο εύκολες.

2.5 Πλεονεκτήματα Arduino

Υπάρχει πληθώρα διαφορετικών μικροελεγκτών και αναπτυξιακών στο εμπόριο για να ασχοληθεί κάποιος, αρκεί να βρει το κατάλληλο για εκείνον ανάλογα με τα κριτήρια της επιλογής του. Όλα αυτά τα εργαλεία που προαναφέραμε είναι απλά, ακόμα και για τον αρχάριο χρήστη, καθώς «κρύβουν» τις δύσκολες λεπτομέρειες της αρχιτεκτονικής και επιτρέπουν τον άμεσο προγραμματισμό του μικροελεγκτή, προσφέροντας τα πάντα σε ένα πακέτο έτοιμο για χρήση. Το Arduino απλοποιεί την διαδικασία του να δουλεύει κάποιος με μικροελεγκτές. Κάποια πλεονεκτήματα που προσφέρει σε σχέση με άλλους μικροελεγκτές για χρήση από δασκάλους, μαθητές και άλλους αναγράφονται παρακάτω (Elgar, 2000):

- Είναι φθηνός: Οι πλακέτες του Arduino είναι εξαιρετικά φθηνές σε σχέση με άλλες πλατφόρμες μικροελεγκτών. Ειδικά δε, μπορεί με τα σχηματικά που κυκλοφορούν στο Internet να κατασκευάσει κάποιος την φθηνότερη εκδοχή ενός Arduino. Ωστόσο, ακόμα και αν προμηθευτεί την έτοιμη αυτή πλακέτα, θα κοστίσει το μέγιστο 50 ευρώ ανάλογα με την έκδοση που θα επιλέξει.
- Πληθώρα Λειτουργικών Συστημάτων: Το περιβάλλον προγραμματισμού του Arduino είναι κατάλληλο για Windows, Macintosh OSX και για λειτουργικά συστήματα Linux.
- Απλό, ξεκάθαρο προγραμματιστικό περιβάλλον: Το περιβάλλον προγραμματισμού ενός Arduino ενδείκνυται για αρχαρίους χρήστες οι οποίοι επιχειρούν για πρώτη φορά να ασχοληθούν με το συγκεκριμένο αντικείμενο, ενώ ταυτόχρονα είναι ευέλικτο για πιο προχωρημένους χρήστες.
- Ανοιχτού λογισμικού και λογισμικού που επεκτείνεται και παραμετροποιείται: Το Software του Arduino διανέμεται με την μορφή εργαλείων ανοιχτού λογισμικού και είναι διαθέσιμο προς επέκταση για έμπειρους προγραμματιστές. Η γλώσσα προγραμματισμού του μπορεί να επεκταθεί μέσω των βιβλιοθηκών της C++ και εκείνοι που θέλουν να ασχοληθούν περισσότερο με τους μικροελεγκτές μπορούν να μεταβούν από τον Arduino στην AVR-C που είναι κατάλληλη για προγραμματισμό των Atmel μικροελεγκτών αλλά και η γλώσσα στην οποία βασίστηκε το

λογισμικό του Arduino. Ομοίως, μπορεί κάποιος να προσθέσει κώδικα της AVR-C στο πρόγραμμα που έχει δημιουργήσει για τον Arduino του.

- Ανοιχτού Υλικού το οποίο μπορεί́ να επεκταθεί: Το Arduino βασίζεται στους μικροελεγκτές της Atmel. Τα σχηματικά για τα αναπτυξιακά του είναι κάτω από την άδεια της Creative Commons, επιτρέποντας σε έμπειρους σχεδιαστές να κατασκευάσουν το δικό τους αναπτυξιακό, εξελίσσοντας το ήδη υπάρχον δίχως να έχουν νομικά προβλήματα. Επίσης, οι όχι τόσο έμπειροι χρήστες μπορούν να επιδιώξουν την αντιγραφή και κατασκευή της πλακέτας σε ράστερ , καταλαβαίνοντας έτσι ακόμα καλύτερα την λειτουργία ενός Arduino. Ακόμα, υπάρχουν οι πλακέτες επέκτασης, λεγόμενες ως shields για μεγαλύτερη επεκτασιμότητα σε αυτό που προσπαθούμε να υλοποιήσουμε. Μερικά shields που υπάρχουν διαθέσιμα στην αγορά τα αναφέραμε σε παραπάνω ενότητα.

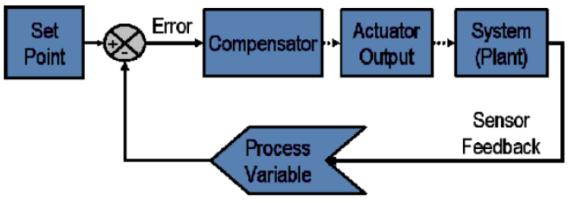
ΚΕΦΑΛΑΙΟ 3

Η Λειτουργία και τα Χαρακτηριστικά του PID Controller και η χρήση του MATLAB

3.1 Σύστημα ελέγχου

Σύστημα αυτόματου ελέγχου ονομάζεται ένα σύνολο φυσικών ή τεχνητών στοιχείων τα οποία είναι συνδεδεμένα μεταξύ τους με τέτοιο τρόπο ώστε να καθοδηγούν, ελέγχουν ή ρυθμίζουν τον εαυτό τους ή άλλα συστήματα ώστε να λειτουργούν με ένα προκαθορισμένο τρόπο. Τα συστήματα αυτόματου ελέγχου διακρίνονται σε δυο κατηγορίες ανάλογα με το αν έχουν ανατροφοδότηση ή όχι. Συγκεκριμένα υπάρχουν:

- Συστήματα ανοιχτού βρόγχου. Ονομάζεται το σύστημα στο οποίο η είσοδος δεν είναι συνάρτηση της εξόδου. Στο ανοιχτό σύστημα δεν υπάρχει ανατροφοδότηση, αλλά αντίθετα χρησιμοποιεί μια συσκευή που παράγει το σήμα εισόδου για να ελέγξει απευθείας την διεργασία.
- Συστήματα κλειστού βρόγχου. Ονομάζεται το σύστημα στο οποίο η είσοδος είναι συνάρτηση της εξόδου. Ένα σύστημα κλειστού βρόγχου χρησιμοποιεί τη μέτρηση του σήματος εξόδου και την ανατροφοδοτεί για να συγκριθεί με το σήμα αναφοράς (είσοδος – επιθυμητή έξοδος). Τα κλειστά συστήματα χαρακτηρίζονται από αυξημένη σταθερότητα και μειωμένη ευαισθησία. Έχουν τεράστιο εύρος λειτουργίας και είναι πολύ ακριβή. Γενικά, προτιμάμε τα κλειστά συστήματα σε σχέση με τα ανοιχτά διότι θέλουμε το σύστημα μας να οδηγείται προς την εξισορρόπηση.

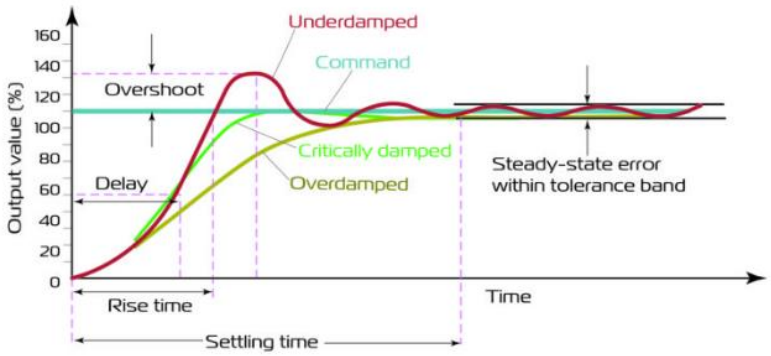


Εικόνα 11: Σύστημα Κλειστού Βρόγχου

- **Set Point** = Η επιθυμητή τιμή ενός φυσικού μεγέθους (π.χ. η θερμοκρασία ενός χώρου να γίνει 30 ^{0}C).
- **Process Variable =** Η μέτρηση του πραγματικού μεγέθους με το κατάλληλο αισθητήριο.
- **Compensator =** Το κύκλωμα που εξασφαλίζει τη διορθωτική κίνηση για την επίτευξη της επιθυμητής τιμής (PID controller).
- **Actuator =** Η εντολή του PID ενεργοποιεί το στοιχείο διόρθωσης (π.χ. μια αντίσταση, ένα καυστήρα, κλπ.).
- **Plant =** Το υπό έλεγχο σύστημα (π.χ. δωμάτιο, κλίβανος, κλπ.).

3.2 Συντονισμός συστήματος

Υπάρχουν τρία χαρακτηριστικά που δείχνουν πότε συντονίζεται σωστά ένα σύστημα: χρόνος απόκρισης *(Response time)*, χρόνος ρύθμισης *(Settling time)* και υπέρβαση *(Overshoot)*.



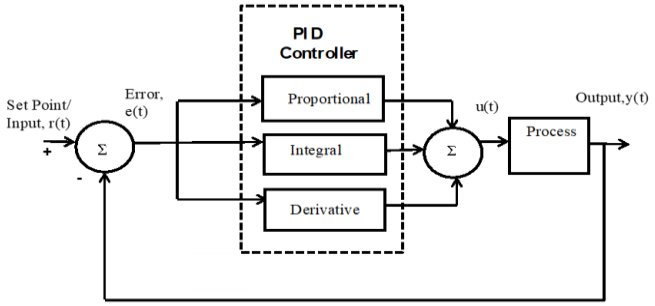
Εικόνα 12: Συντονισμός Συστήματος

Ο χρόνος ανόδου *(Rise Time)* είναι ο χρόνος που απαιτείται για να φτάσει το σύστημα από το 10% στο 90% της τελικής τιμής. Ο χρόνος ρύθμισης *(Settling time)* είναι ο χρόνος που απαιτείται για να φτάσει το σύστημα στο +-5% της τελικής τιμής. Η τιμή *Steady-State Error* είναι η διαφορά της επιθυμητής από την τελική τιμή. Ο στόχος του συντονισμού είναι να ελαχιστοποιηθεί ο χρόνος απόκρισης, ο χρόνος διευθέτησης και η υπέρβαση.

3.3 PID Controller

3.3.1 Η Έννοια και τα Χαρακτηριστικά του PID Ελεγκτή

Ο ελεγκτής PID αποτελεί αναπόσπαστο κομμάτι στα συστήματα αυτομάτου ελέγχου. Υπολογίζει συνεχώς μια τιμή σφάλματος e(t) ως τη διαφορά μεταξύ της επιθυμητής και της πραγματικής τιμής και εφαρμόζει διόρθωση στο σύστημα βασισμένη στους όρους P, I, D. Θα μπορούσαμε να πούμε πως είναι ένα μικρότερο σύστημα μέσα στο κυρίως το οποίο προσπαθεί να μηδενίσει την διαφορά ανάμεσα στη πραγματική και στην επιθυμητή τιμή της εξόδου έτσι ώστε το τελικό σφάλμα να τείνει να είναι μηδενικό. Η έξοδος του ελεγκτή PID σχηματίζεται από το άθροισμα τριών όρων, ενός όρου P (Proportional) ανάλογου του σφάλματος, ενός όρου I (Internal) ανάλογου του ολοκληρώματος του σφάλματος και ενός όρου D (Derivative) ανάλογου της παραγώγου του σφάλματος όπως φαίνεται στο παρακάτω διάγραμμα.



$$\mathrm{u}(t) = \mathrm{MV}(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$

K_p: Proportional gain, a tuning parameter

K_i: Integral gain, a tuning parameter (*Ki=Kp/Ti*)

K_d: Derivative gain, a tuning parameter (*Ki=Kp/*T_d)

e: Error $= SP - PV$

t: Time or instantaneous time (the present)

τ: Variable of integration; takes on values from time 0 to the present t.

Ο όρος P προσαρμόζει την έξοδο του ελεγκτή σύμφωνα με το σύνολο του μεγέθους του σφάλματος. Ο όρος I χρησιμοποιείται για την αφαίρεση του σφάλματος σταθερής κατάστασης του συστήματος ελέγχου και τη βελτίωση της απόκρισης σταθερής κατάστασης και ο όρος D χρησιμοποιείται για να προβλέψει την τάση του σφάλματος και να βελτιώσει την παροδική απόκριση του συστήματος. Κάθε ένας από αυτούς τους όρους μπορεί να καθοριστεί από τον χρήστη. Αυτοί οι όροι πρέπει να προσαρμοστούν για να βελτιστοποιήσουν την ακρίβεια του ελέγχου και η διαδικασία προσδιορισμού των τιμών αυτών των παραμέτρων είναι γνωστή ως PID Tuning. Αυτές οι λειτουργίες είναι αρκετές για τις περισσότερες διαδικασίες ελέγχου. Η δομή του ελεγκτή PID Controller προτείνεται για αυτό το έργο καθώς είναι απλή αλλά και η πιο εκτεταμένη μέθοδος ελέγχου που έχει χρησιμοποιηθεί στη βιομηχανία μέχρι στιγμής. Ο ελεγκτής PID Controller επιχειρεί συνεχώς να προσαρμόσει ένα κατάλληλο αναλογικό κέρδος (K_p), ολοκληρωτικό κέρδος (K_i) και διαφορικό κέρδος (K_d) για την επίτευξη της βέλτιστης απόδοσης ελέγχου.

Με την ρύθμιση των τριών παραμέτρων του μοντέλου, ένας ελεγκτής PID μπορεί να αντιμετωπίσει συγκεκριμένες απαιτήσεις επεξεργασίας. Σε ορισμένες εφαρμογές μπορεί να απαιτείται η χρήση μόνο ενός ή δύο όρων για την παροχή του κατάλληλου ελέγχου του συστήματος. Αυτό θα επιτευχθεί με το μηδενισμό των άλλων παραμέτρων. Ένας ελεγκτής PID ονομάζεται ελεγκτής PI, PD, P ή I όταν λείπουν οι αντίστοιχοι όροι. Πιο συχνά από όλους συναντάται ο PI ελεγκτής, σε εφαρμογές με μικρές καθυστερήσεις χρόνου ή σε περιπτώσεις όπου ο θόρυβος μέτρησης είναι σημαντικός ώστε να αποτρέπεται η χρήση του διαφορικού όρου, ή όταν δεν απαιτείται το κλειστό σύστημα να είναι αρκετά γρήγορο.

Τα αποτελέσματα καθενός από τους ελεγκτές P, I, D σε ένα σύστημα κλειστού βρόγχου φαίνονται 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

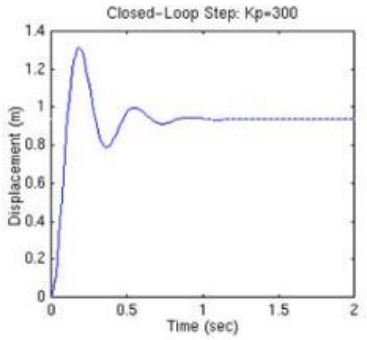
Πίνακας 2: Επίδραση ελεγκτών P, I, D

Έστω ότι έχουμε την συνάρτηση μεταφοράς: $$\frac{X(s)}{F(s)} = \frac{1}{s^2+10s+20}$$

Ας δούμε τις γραφικές παραστάσεις απόκρισης του συστήματος μεταβάλλοντας τους όρους K_p, K_i και K_d.

- P Controller - Αναλογικός Ελεγκτής

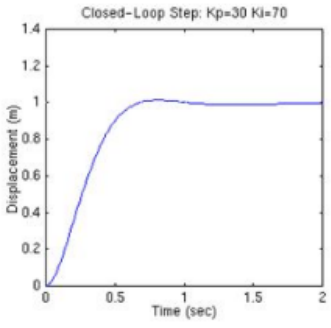
Επιλέγοντας K_p=300, από τη γραφική παράσταση της απόκρισης φαίνεται ότι ο αναλογικός ελεγκτής μειώνει το χρόνο ανόδου και το μόνιμο σφάλμα, αυξάνει την υπερύψωση και επιφέρει μικρή αλλαγή στο χρόνο αποκατάστασης.



Εικόνα 13: Απόκριση - P Controller

- PI Controller - Αναλογικός/Ολοκληρωτικός Ελεγκτής

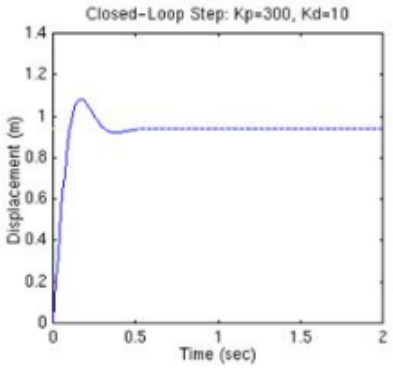
Επιλέγοντας K_p=30 και Ki=70, από τη γραφική παράσταση της απόκρισης φαίνεται ότι ο PI ελεγκτής, μειώνει το χρόνο ανόδου και μηδενίζει το μόνιμο σφάλμα.



Εικόνα 14: Απόκριση - PI Controller

- PD Controller – Αναλογικός/Διαφορικός Ελεγκτής

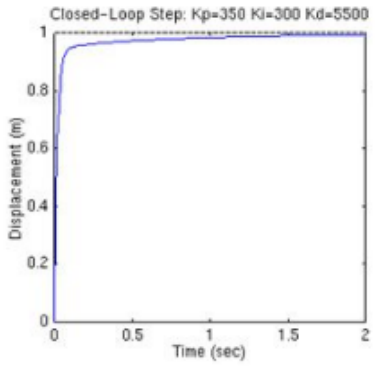
Επιλέγοντας K_p=300 και K_d=10, από τη γραφική παράσταση της απόκρισης φαίνεται ότι ο PD ελεγκτής μειώνει την υπερύψωση και το χρόνο αποκατάστασης ενώ έχει μικρή επιρροή στο χρόνο ανύψωσης και στο μόνιμο σφάλμα.



Εικόνα 15: Απόκριση PD Controller

- PID Controller – Αναλογικός/Ολοκληρωτικός/Διαφορικός Ελεγκτής

Επιλέγοντας K_p=350, K_i=300 και K_d= 5500, είναι φανερό από την γραφική παράσταση πως το σύστημα παρουσιάζει την βέλτιστη απόκριση.

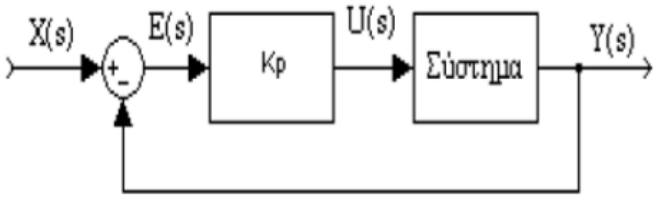


Εικόνα 16: Απόκριση PID Controller

3.3.2 Είδη Ελεγκτών PID

- ***Αναλογικός Ελεγκτής (P)***

Ο αναλογικός ελεγκτής πολλαπλασιάζει το σφάλμα που δέχεται το σύστημα και ουσιαστικά επιταχύνει την επίτευξη της τελικής τιμής. Έχει την παρακάτω μορφή:



Εικόνα 17: Αναλογικός Ελεγκτής

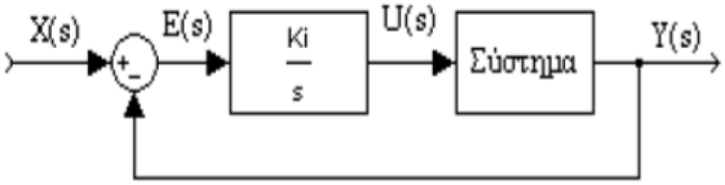
Συνάρτηση μεταφοράς: $\mathbf{P_{(s)} = K_p}$

Στα κλειστά συστήματα, με την σωστή επιλογή του αναλογικού όρου(K_p), σταθεροποιείται το σήμα εξόδου λόγω της εξάλειψης των ταλαντώσεων στη μόνιμη κατάσταση. Πολλές φορές όμως, δημιουργείται σφάλμα μόνιμης κατάστασης που παρά την μείωση του, δεν μπορεί να εξαλειφθεί πλήρως.

Εάν αυξήσουμε το κέρδος του ελεγκτή K_p, μπορεί να μειώνεται το σφάλμα στο σύστημα όμως ταυτόχρονα αυξάνονται οι ταλαντώσεις του.

- ***Ολοκληρωτικός Ελεγκτής (I)***

Ο ολοκληρωτικός ελεγκτής αθροίζει το σφάλμα στο χρόνο. Όσο το σφάλμα μειώνεται, το ολοκλήρωμα τείνει σε σταθερή τιμή που θα είναι η τελική τιμή. Έχει την παρακάτω μορφή:



Εικόνα 18: Ολοκληρωτικός Ελεγκτής

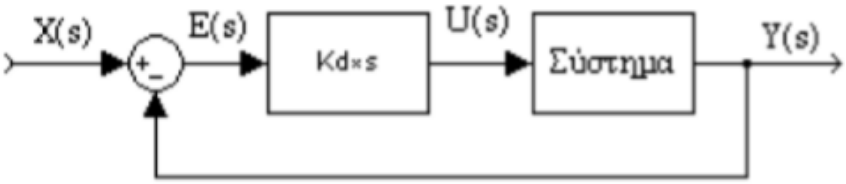
Συνάρτηση μεταφοράς: $\boldsymbol{I(s) = \frac{K_i}{s}}$

Προσθέτοντας τον ολοκληρωτικό όρο, το σφάλμα στην μόνιμη κατάσταση μηδενίζεται , όμως αυξάνεται ο χρόνος αποκατάστασης του συστήματος.

Το ολοκλήρωμα του σφάλματος με σταθερό κέρδος προσφέρει στο σύστημα την εκμηδένιση του μόνιμου σφάλματος σε βηματική απόκριση. Όμως, η φυσική συχνότητα του συστήματος αυξάνεται με αποτέλεσμα να μειώνεται ο συντελεστής απόσβεσης του συστήματος και να χειροτερεύει η ευστάθειά του.

- ***Διαφορικός Ελεγκτής (D)***

Προσθέτοντας τον διαφορικό όρο επιτυγχάνεται η αύξηση της ευστάθειας του κλειστού συστήματος και η βελτίωση της μεταβατικής απόκρισης. Έχει την παρακάτω μορφή:



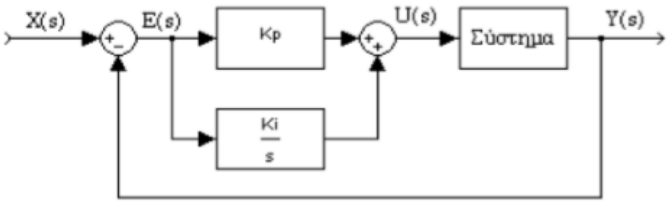
Εικόνα 19: Διαφορικός Ελεγκτής

Συνάρτηση μεταφοράς: $\mathbf{D_{(s)}=K_d*s}$

Η διαφόριση του σφάλματος με σταθερό κέρδος επιφέρει στο σύστημα βελτίωση της ταχύτητας απόκρισης αλλά δεν επηρεάζει το σφάλμα στην μόνιμη κατάσταση. Γενικότερα, αποφεύγεται η χρήση του είτε συνηθίζεται να χρησιμοποιείται κάποιο φίλτρο συνδυαστικό με την παραγώγιση.

- ***Αναλογικός - Ολοκληρωτικός Ελεγκτής (PI)***

Με τον συνδυασμό αναλογικού και ολοκληρωτικού ελέγχου επιτυγχάνεται η εξάλειψη του σφάλματος στην μόνιμη κατάσταση και αυξάνεται η ευστάθεια του συστήματος. Έχει την εξής μορφή:



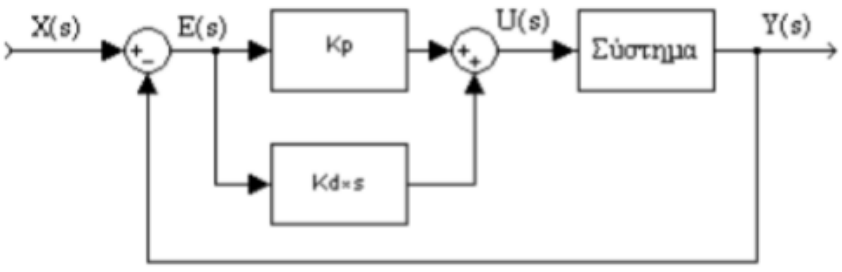
Εικόνα 20:Αναλογικός - Ολοκληρωτικός Ελεγκτής

Συνάρτηση μεταφοράς : $\mathbf{PI(s) = K_p\left(1 + \frac{1}{T_i * s}\right)}$

Με τον PI έλεγχο βελτιώνεται το σφάλμα στην μόνιμη κατάσταση. Αυξάνεται όμως η υπερύψωση των ταλαντώσεων και το σύστημα μπορεί να πέσει σε αστάθεια.

- ***Αναλογικός - Διαφορικός Ελεγκτής (PD)***

Με τον συνδυασμό αναλογικού και διαφορικού ελέγχου, έχουμε καλύτερη απόσβεση του συστήματος άρα και κατά συνέπεια μείωση των ταλαντώσεων, αλλά και ελάττωση του σφάλματος στην μόνιμη κατάσταση. Έχει την εξής μορφή:



Εικόνα 21: Αναλογικός - Διαφορικός Ελεγκτής

Συνάρτηση μεταφοράς: $\mathbf{PD_{(s)} = K_p + K_d * s}$

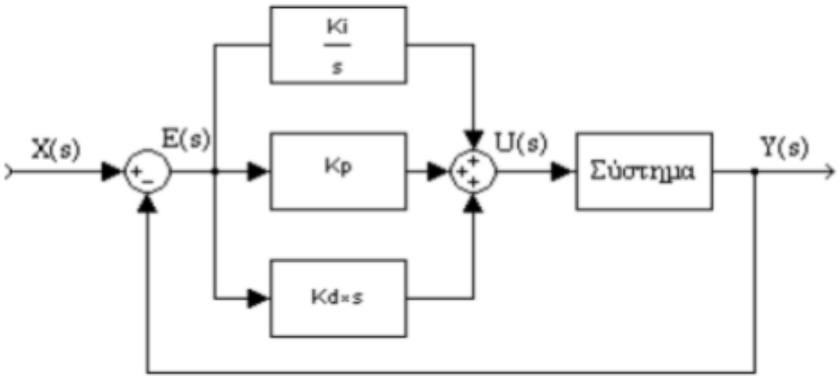
- ***Αναλογικός - Ολοκληρωτικός-Διαφορικός Ελεγκτής (PID)***

Ο ελεγκτής PID Controller είναι ευρέως διαδεδομένος και προτείνεται για αυτή την εργασία, λόγω της απλότητας του αλλά και της ορθής συμπεριφοράς του σε τεράστια κλίμακα συνθηκών. Ο PID Controller χρησιμοποιείται ευρέως στον έλεγχο κίνησης λόγω του απλού αλγόριθμου καθώς και της υψηλής αξιοπιστίας του. Όμως μερικά από τα ελεγχόμενα αντικείμενα του δεν έχουν ορισμένο κανένα ακριβές μαθηματικό μοντέλο, το οποίο να οδηγεί στην ρύθμιση των παραμέτρων PID. Έτσι, οι παράμετροι έχουν συνήθως κακή απόδοση και είναι δύσκολο να ικανοποιήσουν τον έλεγχο κίνησης υψηλής ακρίβειας του γραμμικού κινητήρα. Αν όμως χρησιμοποιηθεί ο αλγόριθμος, βασισμένος στο εκάστοτε σύστημα, για τον ορισμό των παραμέτρων PID, το σύστημα ελέγχου μπορεί όχι μόνο να διατηρήσει τα πλεονεκτήματα των απλών αρχών και τη χρήση του συμβατικού συστήματος ελέγχου PID, αλλά και να αποκτήσει χαρακτηριστικά όπως η ευελιξία και η προσαρμοστικότητα, τα οποία μπορούν να βελτιώσουν αποτελεσματικά την απόδοση του.

Κατά τη διάρκεια του περασμένου μισού αιώνα, οι ερευνητές αναζητούσαν την επόμενη βασική τεχνολογία για τον συντονισμό PID Controller και την ομαλή του υλοποίηση. Η λειτουργία Computerizing, επιτρέπει την αυτόματη διεξαγωγή προσομοιώσεων, γεγονός που διευκολύνει την αναζήτηση των καλύτερων δυνατών ρυθμίσεων PID για την εφαρμογή. Μια προσέγγιση βασισμένη στην προσομοίωση, δεν απαιτεί τεχνητή ελαχιστοποίηση του εύρους ελέγχου και βοηθά στη βελτίωση της βραδείας παροδικής απόκρισης του ελεγκτή (Microcontroller, 2009). Κατά την αντιμετώπιση των προβλημάτων PID Controller, είναι επιθυμητό να χρησιμοποιηθούν τυποποιημένες δομές PID για εύλογο φάσμα τύπων εγκαταστάσεων και λειτουργιών. Η προσαρμογή γύρω από τις τυποποιημένες δομές PID Controller, θα πρέπει επίσης να συμβάλει στη βελτίωση της οικονομικής αποδοτικότητας του ελέγχου και της συντήρησης PID. Με αυτόν τον τρόπο, μπορούν να αναπτυχθούν εξαιρετικά βέλτιστες μέθοδοι σχεδιασμού. Με τη ύπαρξη τεχνικών ταυτοποίησης του συστήματος, ολόκληρη η διαδικασία σχεδιασμού και συντονισμού PID Controller μπορεί να αυτοματοποιηθεί και μπορούν να διατεθούν αρθρωτά μπλοκ κώδικα για έγκαιρη εφαρμογή και προσαρμογή σε πραγματικό χρόνο (Aras, et al., 2009).

Ο αλγόριθμος PID είναι ο πιο δημοφιλής ελεγκτής ανάδρασης που χρησιμοποιείται ευρέως στο σύστημα ελέγχου. Η μέθοδος PID είναι ένα από τα πιο διαδεδομένα ανατροφοδοτούμενα συστήματα ελέγχου που έχει χρησιμοποιηθεί πριν από περισσότερα από 50 χρόνια. Ένα από τα πρώτα παραδείγματα ενός ελεγκτή τύπου PID αναπτύχθηκε από τον Elmer Sperry το 1911, ενώ η πρώτη δημοσιευμένη θεωρητική ανάλυση ενός ελεγκτή PID Controller, ήταν από τον Ρώσο-Αμερικάνο μηχανικό Nicolas Minorsky το 1922.

Η μορφή ενός PID ελεγκτή είναι η ακόλουθη:



Εικόνα 22: PID Ελεγκτής

$$\text{Συνάρτηση μεταφοράς : } \mathbf{PID(s)} = \frac{\mathbf{K_d * s^2 + K_p * s + K_i}}{\mathbf{s}}$$

Με την σωστή επιλογή των όρων K_d, K_p και K_i, ο PID Controller συνδυάζει μικρό χρόνο ανόδου, μικρή υπερύψωση, μικρό χρόνο αποκατάστασης και μηδενικό σφάλμα στην μόνιμη κατάσταση.

Συνοψίζοντας, έχοντας γνωρίζει τα διαφορετικά είδη ελεγκτών καταλήγουμε στα ακόλουθα:

- Με την αύξηση του αναλογικού κέρδους, το μόνιμο σφάλμα ελαττώνεται. Εάν όμως, το αυξήσουμε πολύ, το σύστημα γίνεται ασταθές.
- Με τον ολοκληρωτικό έλεγχο, επιτυγχάνεται μείωση του μόνιμου σφάλματος, όμως συχνά μειώνεται η ευστάθεια του συστήματος.

- Με τον διαφορικό έλεγχο, βελτιώνεται η ευστάθεια του συστήματος και το μόνιμο σφάλμα μένει ανεπηρέαστο.

3.3.3 Σχεδίαση Ελεγκτών PID

Ο σχεδιασμός του PID Controller αναφέρεται στην επιλογή κατάλληλων τιμών στον συντελεστή κάθε όρου (K_p, K_i, K_d), ώστε το σύστημα να ανταποκρίνεται θετικά υπό έλεγχο και να λειτουργεί σύμφωνα με τις επιθυμητές προδιαγραφές. Η κακή ρύθμιση του PID μπορεί να προκαλέσει πολλά προβλήματα στην απόκριση του συστήματος όπως είναι οι αργοί χρόνοι ύψωσης, οι ταλαντώσεις και μη επαρκής απόσβεση.

- **ΧΕΙΡΟΚΙΝΗΤΗ ΡΥΘΜΙΣΗ**

Οι μέθοδοι που χρησιμοποιούνται για τον υπολογισμό των τριών όρων ενός ελεγκτή είναι οι εμπειρικές και οι υπολογιστικές. Στις εμπειρικές μεθόδους υπολογίζουμε τις παραμέτρους έχοντας τη βηματική απόκριση του συστήματος. Η πιο φυσική μέθοδος είναι η χειροκίνητη ρύθμιση από έναν χρήστη. Τα κέρδη K_p, K_i, K_d αρχικά ισούνται με το μηδέν. Ύστερα, αυξάνουμε το αναλογικό κέρδος έως ότου το σύστημα παρουσιάσει ελαφριά ταλάντωση και ο χρόνος ανύψωσης γίνει ικανοποιητικός. Στην συνέχεια, αυξάνουμε το ολοκληρωτικό κέρδος μέχρι να εξαλειφθεί το σφάλμα μόνιμης κατάστασης, με γνώμονα πως όσο μεγαλύτερη είναι αυτή η τιμή τόσο πιο ασταθές θα είναι το σύστημα. Τέλος, εάν είναι απαραίτητο, αυξάνουμε το διαφορικό κέρδος ώστε να βελτιώσουμε την σταθερότητα του συστήματος με γνώμονα ότι το μεγάλο διαφορικό κέρδος θα προκαλέσει υπερβολική ανταπόκριση σε μεταβολές και θόρυβο. Οι τιμές αυτές υπολογίζονται έως ότου καταλήξουμε στις τιμές που θα προσφέρουν στο σύστημα μας: ικανοποιητική υπερύψωση, ταχεία απόκριση και μικρό μόνιμο σφάλμα.

- **ΜΕΘΟΔΟΣ Ziegler – Nichols**

Η μέθοδος συντονισμού Ziegler–Nichols, γνωστή και ως *μέθοδος ορίου ευστάθειας*, είναι μια από τις πιο γνωστές μεθόδους συντονισμού ενός ελεγκτή PID. Παρόλο που αναπτύχθηκε το 1942 από τους John G. Ziegler και Nathaniel B. Nichols, η μέθοδος αυτή χρησιμοποιείται ακόμα και σήμερα. Εφαρμόζεται στα συστήματα των οποίων η βηματική απόκριση μπορεί να οδηγηθεί σε αμείωτες ταλαντώσεις και βασίζεται στον έλεγχο μόνο με αναλογικό ρυθμιστή.

Η μέθοδος αυτή εκτελείται ρυθμίζοντας τα κέρδη K_i, K_d στο μηδέν. Στην συνέχεια, το αναλογικό κέρδος αυξάνεται μέχρι την τιμή που θα οδηγήσει το σύστημα σε ευστάθεια, η βηματική απόκριση του συστήματος δηλαδή να εμφανίσει αμείωτες ταλαντώσεις. Η τιμή αυτή του κέρδους ονομάζεται K_{cr} και της περιόδου της ταλάντωσης $T_{cr.}$ Οι παράμετροι του PID Controller μπορούν να προσδιοριστούν σύμφωνα με τον παρακάτω πίνακα βάσει των τιμών K_{cr} και T_{cr} :

Ελεγκτής	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 3: Παράμετροι Ziegler – Nichols

Ωστόσο, αυτή η μέθοδος παρουσιάζει ορισμένα μειονεκτήματα. Οι πειραματικοί έλεγχοι μπορεί να διαρκέσουν μεγάλα χρονικά διαστήματα με αποτέλεσμα την περιορισμένη παραγωγή. Επίσης, σε πολλές εφαρμογές, δεν είναι δυνατή η ταλάντωση που παρατηρείται στα όρια της ευστάθειας διότι μόλις δεχτεί το σύστημα εξωτερικές διαταραχές, παρουσιάζει ασταθής ή ακόμα και επικίνδυνη λειτουργία. Με άλλα λόγια, αυτοί οι κανόνες ρύθμισης δεν αποτελούν από μόνοι τους μια πλήρη μέθοδο συντονισμού, όμως μπορούν να χρησιμοποιηθούν στα πρώτα βήματα της διαδικασίας.

- **ΜΕΘΟΔΟΣ Tyreus – Luyben**

Η μέθοδος αυτή είναι παρόμοια με την μέθοδο Ziegler–Nichols αφού στηρίζεται στην τιμή αυτή του κέρδους K_{cr} και της περιόδου της ταλάντωσης $T_{cr.}$ Αυτό που

αλλάζει είναι το πως προσδιορίζονται οι παράμετροι του PID Controller. Οι τιμές αυτές φαίνονται στον παρακάτω πίνακα:

Ελεγκτής	K_p	T_i	T_d
PI	$K_{cr}/3.2$	$2.2T_{cr}$	0
PID	$K_{cr}/2.2$	$2.2T_{cr}$	$T_{cr}/6.3$

Πίνακας 4: Παράμετροι Tyreus – Luyben

- **ΜΕΘΟΔΟΣ Cohen – Coon**

Στόχος της μεθόδου αυτής είναι ο περιορισμός των διαταραχών στην έξοδο του συστήματος. Αυτό θα επιτευχθεί με τον κατάλληλο υπολογισμό των παραμέτρων συντονισμού έτσι ώστε να προσφέρουν μια απόκριση κλειστού συστήματος με συντελεστή απόσβεσης 0.25. Οι παράμετροι του PID Controller υπολογίζονται από τους εξής τύπους:

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

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

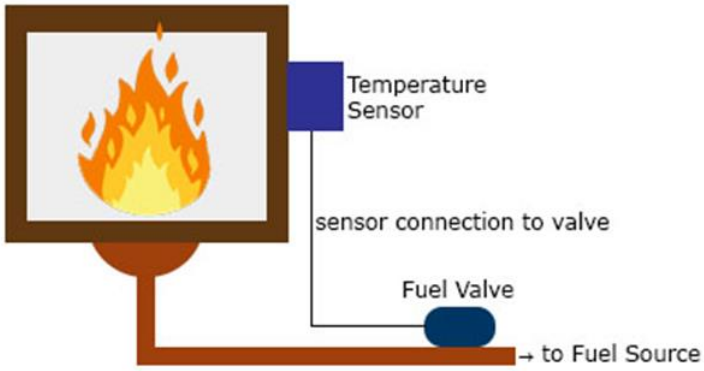
$$\boldsymbol{T_d = \frac{4t_d}{11+\frac{2t_d}{\tau}}}$$

3.3.4 Κλίβανος - PID CONTROLLER

Όπως αναφέρουμε παραπάνω, στα συστήματα ελέγχου, ένας ελεγκτής διορθώνει την έξοδο ενός συγκεκριμένου συστήματος σε μια επιθυμητή τιμή/είσοδο, παρά την παρουσία σφαλμάτων και διαταραχών. Ο πιο δημοφιλής τύπος ελεγκτή είναι ο PID Controller, τον οποίο και θα χρησιμοποιήσουμε σε αυτό το Project. Το όνομα αυτό προέρχεται από τις μεθόδους που χρησιμοποιούνται για την διόρθωση των διαταραχών στο σύστημα (Proportional, Integral, Derivative).

Σύστημα ανάδρασης είναι ένα σύστημα όπου μέρος της εξόδου ανατροφοδοτείται στην είσοδο.

Για την καλύτερη κατανόηση όσων αναφέρθηκαν παραπάνω, έχουμε ένα πρόγραμμα που ελέγχει τη φωτιά στο φούρνο:



Θέλουμε να διατηρήσουμε τη θερμοκρασία στο φούρνο σε ένα συγκεκριμένο σημείο ρύθμισης (Set Point). Ένας αισθητήρας εγκατεστημένος στον κλίβανο αναγνωρίζει την θερμοκρασία του φούρνου ανά πάσα στιγμή. Αυτός ο αισθητήρας, στην συγκεκριμένη περίπτωση, παρέχει την ανατροφοδότηση ως αναφορά στην απαιτούμενη αύξηση ή μείωση της θερμοκρασίας. Η διαφορά μεταξύ του αισθητήρα ανάδρασης και της επιθυμητής θερμοκρασίας που έχουμε δηλώσει ως Set Point, είναι το σφάλμα.

Proportional Control

Ο αναλογικός ελεγκτής (P) αναφέρεται σε έναν υπολογισμό που είναι ανάλογος με το πόσο είναι το σφάλμα.

Ας υποθέσουμε ότι ο ελεγκτής στο παραπάνω παράδειγμα είναι μια ηλεκτρονική βαλβίδα, υπεύθυνη για τον έλεγχο του καυσίμου στον κλίβανο. Εάν το σφάλμα είναι μικρό, η βαλβίδα θα απελευθερώσει μια μικρή ποσότητα καυσίμου έτσι ώστε το σημείο ρύθμισης και η ανάδραση να συμπίπτουν. Εάν το σφάλμα είναι μεγάλο, η βαλβίδα πρέπει να απελευθερώσει περισσότερο καύσιμο για να συμπίπτουν αυτές οι τιμές.

Integral Control

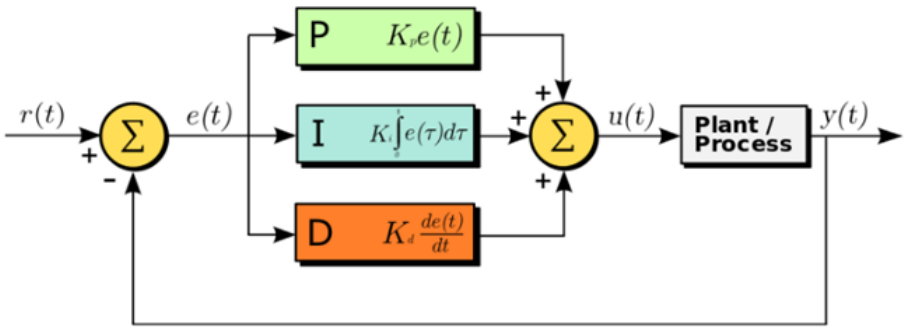
Ο αναλογικός ελεγκτής παράγει μια αντιστάθμιση στην διόρθωση του λόγω διαταραχών. Ο ολοκληρωτικός ελεγκτής (Integral) έχει την δυνατότητα να αφαιρέσει αυτή την μετατόπιση και να επαναφέρει το σφάλμα στο μηδέν. Ένας τέτοιος ελεγκτής είναι υπεύθυνος για προσαρμογές που βασίζονται στο συσσωρευμένο σφάλμα κατά την πάροδο του χρόνου. Χωρίς τον Integral Controller, το σύστημα δεν μπορεί να αντιμετωπίσει την προδιάθεση των σφαλμάτων.

Στο παραπάνω παράδειγμα, μπορεί να υπάρχει μια μετατόπιση εάν η βαλβίδα καυσίμου δεν επιστρέψει στην αρχική της θέση κατά την αυξομείωση της παροχής καυσίμου. Ο ολοκληρωτικός ελεγκτής θα το εντοπίσει αυτό και θα γυρίσει τη βαλβίδα καυσίμου στην αρχική της θέση.

Derivative Control

Τέλος, ο διαφορικός ελεγκτής (D) ασχολείται με τον ρυθμό αλλαγής του σφάλματος. Θα μπορούσαμε να πούμε δηλαδή πως εάν ο ολοκληρωτικός ελεγκτής εξετάζει το ιστορικό του σφάλματος, τότε ο διαφορικός ελεγκτής μπορεί και προβλέπει το σφάλμα. Βασικά, το ποσό της διόρθωσης θα βασίζεται στο πόσο γρήγορα αλλάζει το σφάλμα. Αυτός ο τύπος ελεγκτή λειτουργεί καλύτερα με δυναμικά σφάλματα στα οποία δεν μπορούν να ασχοληθούν τόσο οι αναλογικοί όσο και οι ολοκληρωμένοι ελεγκτές.

Στο παραπάνω παράδειγμα, ας υποθέσουμε πως η θερμοκρασία στον φούρνο αυξάνεται ξαφνικά από 130 °C σε 140 °C μέσα σε 2 δευτερόλεπτα ενώ έχουμε ορίσει ως Set Point τους 120 °C. Ο αναλογικός και ο ολοκληρωτικός ελεγκτής θα ανταποκριθούν στο μέγεθος του σφάλματος, αλλά θα δυσκολευτούν να φτάσουν μέχρι το πόσο γρήγορα συνέβη το σφάλμα. Ο διαφορικός ελεγκτής μπορεί να το αντιμετωπίσει αυτό καθώς εξετάζει τον ρυθμό του σφάλματος από την αρχή.



Εικόνα 23: Σύστημα PID Controller - Τύποι

Στο παραπάνω σύστημα βλέπουμε πως η μεταβλητή εισόδου ή αλλιώς Set Point είναι η r(t), μεταβλητή εξόδου η y(t), η ελεγχόμενη μεταβλητή είναι η u(t) και το σφάλμα είναι e(t). Συνεχίζοντας με το παράδειγμα του φούρνου, η r(t) θα είναι η επιθυμητή θερμοκρασία και y(t) θα είναι η πραγματική θερμοκρασία, e(t) είναι η διαφορά μεταξύ της επιθυμητής και της πραγματικής θερμοκρασίας και u(t) είναι το άθροισμα των διορθώσεων των P, I, D ελεγκτών.

Προσέχουμε ότι ο PID Controller δεν μπορεί να χρησιμοποιηθεί έξω από το κουτί. Πρέπει να γίνει ο συντονισμός του συστήματος ώστε να διασφαλιστεί ότι επιτυγχάνεται η επιθυμητή απόδοση. Αυτό γίνεται με προσεκτική αλλαγή των σταθερών K_p, K_i, K_d όπως φαίνεται στο παραπάνω διάγραμμα. Αυτές οι σταθερές πρέπει να προσδιορίζονται εκ των προτέρων και να τροποποιούνται σύμφωνα με την πραγματική απόκριση του συστήματος έως ότου επιτευχθούν οι βέλτιστες τιμές που θα προσφέρουν την καλύτερη απόδοση του συστήματος.

3.4 MATLAB

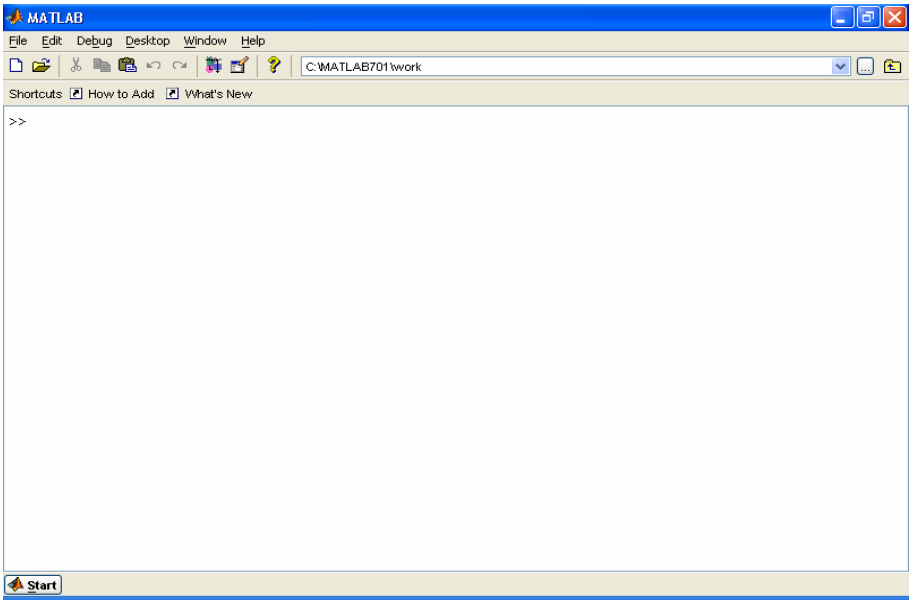
3.4.1 Το περιβάλλον εργασίας MATLAB

Το Matlab (Matrix Laboratory) είναι ένα πολυδιάστατο αριθμητικό περιβάλλον πληροφορικής και γλώσσα προγραμματισμού τέταρτης γενιάς. Το πρώτο version του Matlab αναπτύχθηκε στη γλώσσα C και δημιουργήθηκε το 1984. Αυτό το λογισμικό αναπτύχθηκε από την εταιρεία MathWorks. Οι χρήστες του Matlab προέρχονται από διάφορα περιβάλλοντα μηχανικής, επιστήμης και οικονομίας. Χρησιμοποιείται ευρέως σε ακαδημαϊκά και ερευνητικά ιδρύματα

καθώς και σε βιομηχανικές επιχειρήσεις. Το MATLAB είναι ένα διαδραστικό περιβάλλον το οποίο διαθέτει μια γλώσσα προγραμματισμού υψηλού επιπέδου και χρησιμοποιείται για αριθμητικούς υπολογισμούς, προγραμματισμό και οπτικοποίηση των αποτελεσμάτων. Η γλώσσα προγραμματισμού MATLAB λειτουργεί ως διερμηνέας εντολών (command interpreter), οι οποίες δίνονται μέσω του παραθύρου εντολών της (MATLAB command window). Οι εντολές αυτές μπορεί να είναι:

1. Ορισμοί μεταβλητών και πράξεις.
2. Κλήση ενσωματωμένων συναρτήσεων της MATLAB και των εγκατεστημένων εργαλειοθηκών της (toolboxes).
3. Κλήση συναρτήσεων (functions) ή αρχείων εντολών MATLAB (scripts) που κατασκευάζονται από τους χρήστες με τη μορφή M-file.

Το περιβάλλον εργασίας της MATLAB απεικονίζεται παρακάτω.

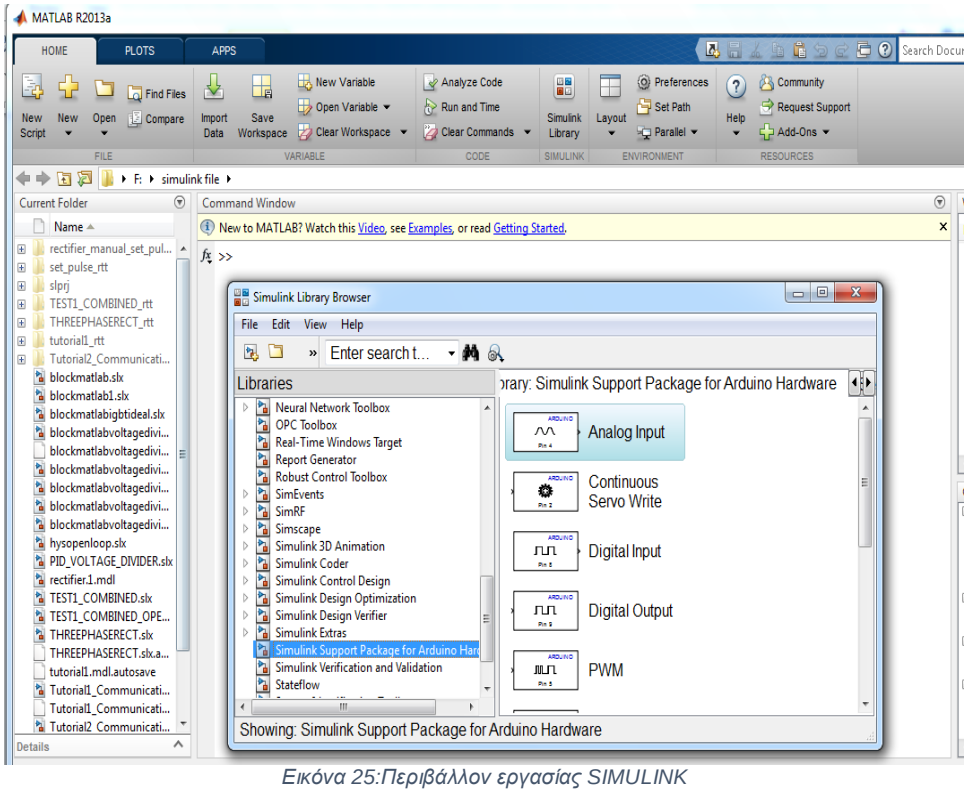


Εικόνα 24:Περιβάλλον εργασίας MATLAB

Η ευρεία χρήση της MATLAB οφείλεται σε μεγάλο βαθμό στην επεκτασιμότητα της μεσώ των διαφόρων εργαλειοθηκών, κάθε μια από τις οποίες περιέχει ένα αριθμό συναρτήσεων για ένα συγκεκριμένο αντικείμενο. Η δομή των υπαρχόντων στοιχειών σε μια εγκατάσταση MATLAB παρουσιάζεται εκτελώντας την εντολή help (Faugel, Bobkov, 2013).

Κάθε φορά που εκκινείται η MATLAB δημιουργείται στη μνήμη του υπολογιστή ο χώρος εργασίας (workspace) εντός του οποίου αποθηκεύονται οι οριζόμενες στο παράθυρο εντολών μεταβλητές. Οι μεταβλητές αυτές είναι διαθέσιμες μέχρι την έξοδο από τη MATLAB, ενώ είναι δυνατή η αποθήκευση τους στο δίσκο και η ανάκτηση τους σε επόμενη εκκίνηση της MATLAB. Ο εξ' ορισμού τύπος μεταβλητής είναι πραγματικός διπλής ακρίβειας (double). Άλλοι τύποι μεταβλητών που μπορούν να χρησιμοποιηθούν είναι ο μιγαδικός αριθμός και η μεταβλητή-αντικείμενο συνάρτησης μεταφοράς. Η δήλωση μιας μεταβλητής στο παράθυρο εντολών γίνεται ταυτόχρονα με την απόδοση τιμής σε αυτήν (Aras, et al., 2009).

Αυτό το λογισμικό είναι φιλικό προς το χρήστη και επιπλέον περιέχει μια βιβλιοθήκη, ιδιαίτερα χρήσιμη για την εργασία μας, τη Simulink Browser. Βασικά χαρακτηριστικά του Matlab Simulink, είναι η γλώσσα υψηλού επιπέδου για αριθμητικό υπολογισμό, οπτικοποίηση και ανάπτυξη εφαρμογών. Επίσης, το διαδραστικό περιβάλλον για την εξερεύνηση, τον σχεδιασμό και την επίλυση προβλημάτων. Οι μαθηματικές λειτουργίες για γραμμική άλγεβρα, στατιστικές, ανάλυση Fourier, φιλτράρισμα, βελτιστοποίηση, αριθμητική ολοκλήρωση και επίλυση συνήθων διαφορικών εξισώσεων και δεδομένων. Διαθέτει ενσωματωμένα γραφικά για την απεικόνιση δεδομένων όπως και εργαλεία ανάπτυξης για τη βελτίωση της ποιότητας και της συντηρησιμότητας του κώδικα για την βέλτιστη εκτέλεση του. Ακόμα, περιλαμβάνει λειτουργίες για την ενσωμάτωση αλγορίθμων βασισμένων σε MATLAB συνδυασμένων με εξωτερικές εφαρμογές και γλώσσες όπως C, Java, .NET και Microsoft Excel. (www.mathworks.com). Αυτό το λογισμικό επιλέγεται και συνιστάται για αυτή την εργασία λόγω του φιλικού περιβάλλοντος ανάπτυξης προς τον χρήστη και του ότι μας επιτρέπει να δημιουργούμε και να εκτελούμε μοντέλα στον Arduino. Το νέο μοντέλο μπορεί να σχεδιαστεί από τη Βιβλιοθήκη Simulink και στη συνέχεια το να τρέξει σε πραγματικό χρόνο στο Arduino. Εικονίδια από το πρόγραμμα περιήγησης βιβλιοθήκης μπορούν να μεταφέρονται και να μετακινούνται σε αυτό το μοντέλο, ώστε να σχεδιασθεί το επιθυμητό τελικό αποτέλεσμα.



Εικόνα 25:Περιβάλλον εργασίας SIMULINK

3.4.2 Σύνδεση του MATLAB με τον ARDUINO UNO

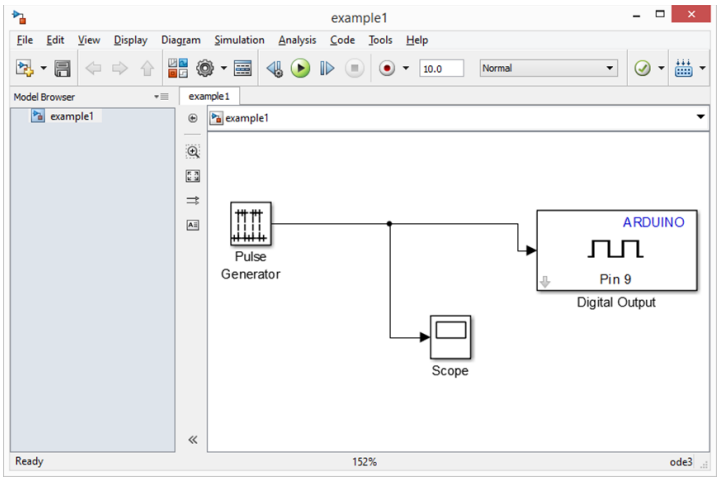
ΧΡΗΣΗ SIMULINK

Για την εκπόνηση αυτής της εργασίας χρησιμοποιούμε το Matlab 2017a και είναι απαραίτητο να δημιουργήσουμε έναν λογαριασμό στην MathWorks ώστε να μπορούμε να «κατεβάσουμε» οποιοδήποτε πακέτο υποστήριξης χρειαστούμε. Το πιο σημαντικό είναι το *Support Package for Arduino Hardware* το οποίο μας επιτρέπει να δημιουργούμε και να εκτελούμε μοντέλα Simulink στον Arduino. Για να το προσθέσουμε, μέσα από το περιβάλλον του Matlab επιλέγουμε Add-Ons → Get Hardware Support Packages → Arduino Uno. Το package αυτό μας επιτρέπει να δημιουργήσουμε ένα μοντέλο Simulink το οποίο έπειτα συντάσσεται, φορτώνεται και εκτελείται στον Arduino.

Παρακάτω, αναφέρουμε ορισμένα παραδείγματα που ήταν απαραίτητα για να καταλάβουμε πως λειτουργεί ο συνδυασμός Simulink-Arduino.

Οδήγηση ενός LED με γεννήτρια παλμών

- Επιλέγουμε SIMULINK LIBRARY (ή στο command prompt του Matlab την δίνουμε την εντολή Simulink). Επιλέγουμε Simulink Support Package for Arduino Hardware και έπειτα Examples for Arduino Uno. Εμφανίζονται στα Tutorials: «Getting Started with Arduino Uno Hardware» και «Servo Control» και στα Examples: «Drive with PID Control».
- Επιλέγουμε Getting Started with Arduino Uno Hardware → «Task 1 - Connect a LED to an Arduino Output Pin». Ακολουθούμε τα παρακάτω βήματα ώστε να δημιουργήσουμε στο SIMULINK μια γεννήτρια παλμών η οποία να αναβοσβήνει ένα LED στον Arduino:
  - HOME → NEW → SIMULINK Model και ανοίγει το παράθυρο ανάπτυξης του μοντέλου, στο οποίο δίνουμε το όνομα example1.slx.
  - Σύρουμε από το SIMULINK Library Browser και από τη βιβλιοθήκη “Simulink Support Package for Arduino Hardware” το μπλοκ Digital Output και από τη βιβλιοθήκη “Sources” το Pulse Generator. Ενώνουμε τη έξοδο του δεύτερου με την είσοδο του πρώτου.
  - Μπορούμε να τοποθετήσουμε οθόνη που θα δείχνει την έξοδο της γεννήτριας παλμών επιλέγοντας τη βιβλιοθήκη “Sinks” → Scope.



Εικόνα 26: Simulink Model 1

- Για την εξομοίωση του Simulink model επιλέγουμε Simulation → Run και βλέπουμε την έξοδο της γεννήτριας στον παλμογράφο.

- Για να φορτώσουμε το πρόγραμμα υλοποίησης στον Arduino επιλέγουμε Tools → Run On Target Hardware → Prepare to Run και έπειτα Target Hardware=Arduino Uno. Στη συνέχεια επιλέγουμε Tools → Run On Target Hardware → Run και το πρόγραμμα φορτώνεται στον Arduino και εκτελείται.
- Για να φορτωθεί το πρόγραμμα πρέπει το Matlab να δείχνει στον φάκελο που υπάρχει το example3.

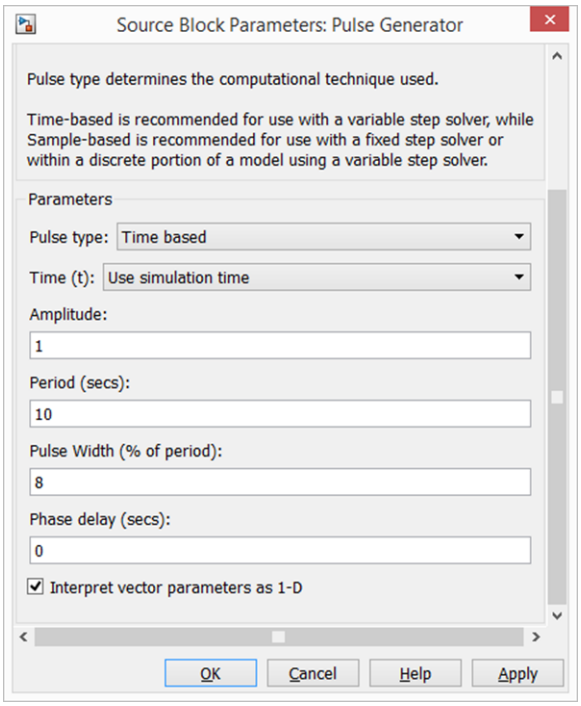
Πριν φορτώσουμε το πρόγραμμα στον Arduino, καλό είναι να ελέγξουμε τα εξής:

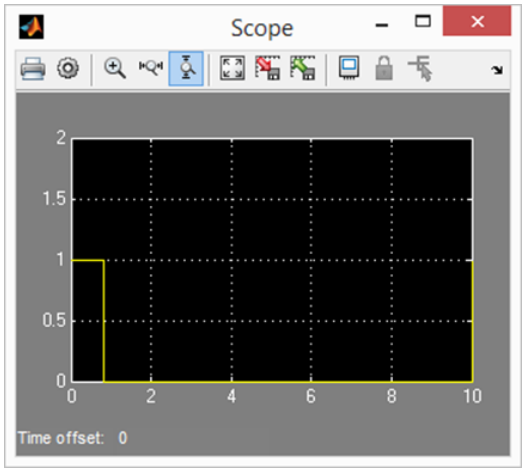
- Tools → Run On Target Hardware → Options. Πρέπει στο “COM Port Number” να υπάρχει η πόρτα την οποία έχει αναθέσει ο υπολογιστής στον Arduino (π.χ. COM5). Αν δεν είναι η σωστή τότε θα έχουμε σφάλμα. Μπορούμε να δούμε την ορισμένη πόρτα από τον Διαχειριστή Συσκευών (Device Manager).
- Διπλοπατώντας στο μπλοκ “Digital Output” μπορούμε να αλλάξουμε τον ακροδέκτη εξόδου.
- Διπλοπατώντας στο μπλοκ “Pulse Generator” βλέπουμε τις παραμέτρους της γεννήτριας και μπορούμε να κάνουμε αλλαγές ανάλογα με το αποτέλεσμα που θέλουμε να λάβουμε. Οι παράμετροι που χρησιμοποιούνται είναι οι εξής:

    - **Time (t)**: Χρήση εσωτερικού ή εξωτερικού χρονισμού εξομοίωσης.
    - **Amplitude**: Πλάτος παλμών.
    - **Period** (Secs-Number of samples): Σε seconds ή σε αριθμό δειγμάτων.
    - **Pulse Width** (% of period-Number of samples): Ορίζει το Duty Cycle, δηλαδή για πόσο χρονικό διάστημα ο παλμός είναι HIGH σε μια περίοδο. Εκφράζεται με (%) ή με αριθμό δειγμάτων αντίστοιχα.
    - **Phase delay** (secs-Number of samples): Σε seconds ή σε αριθμό δειγμάτων.
    - **Sample Time**: Εκφράζει την χρονική απόσταση των δειγμάτων σε seconds και υπάρχει μόνο στην επιλογή Sample based.

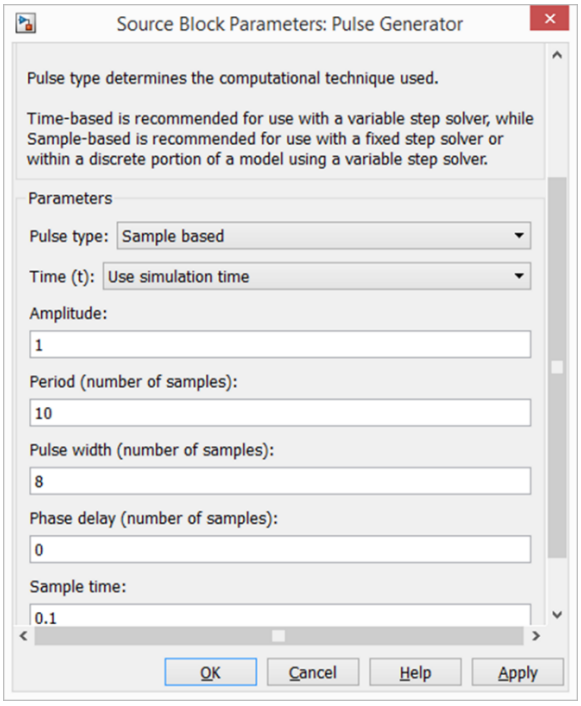
Παρακάτω απεικονίζονται δύο διαφορετικοί παλμοί:

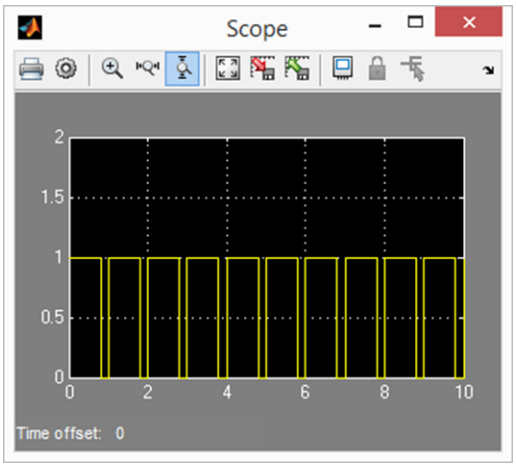
- Time based:



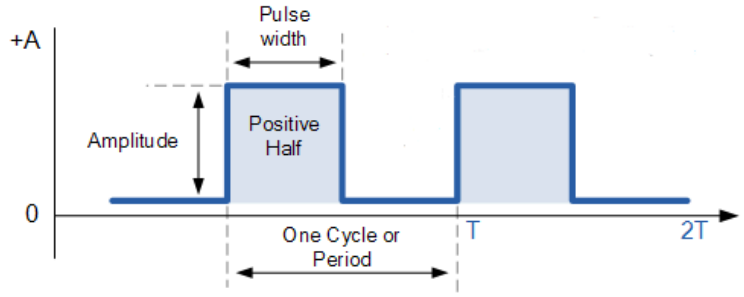


- Sample based:





Η διαφορά ένδειξης μεταξύ Time based και Sample based οφείλεται στη διαφορά της περιόδου. Στην πρώτη περίπτωση είναι 10 s ενώ στη δεύτερη είναι 10 samples * 0.1s=1s.

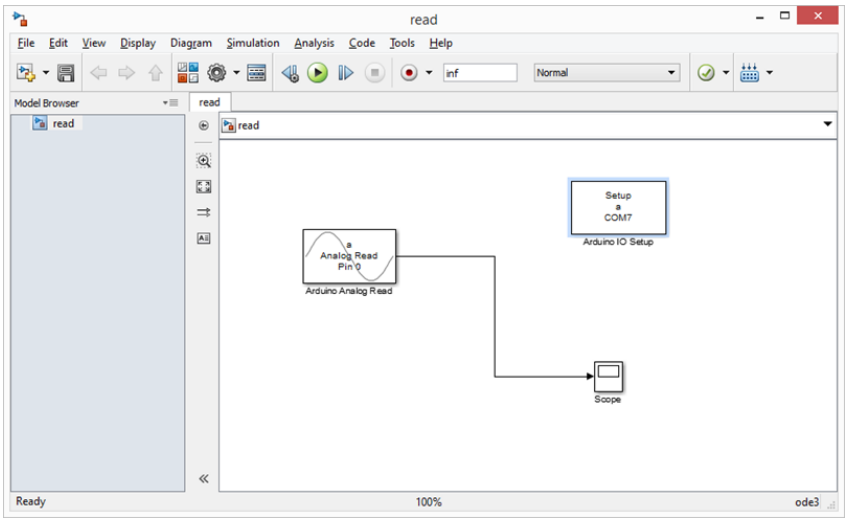


Pulse Type: Time based – Sample based

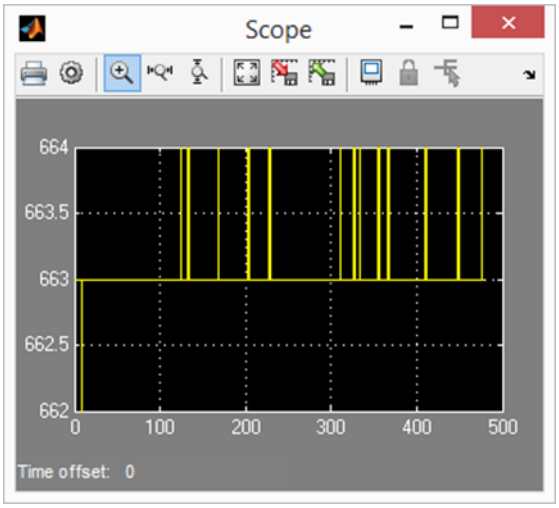
Στο *Time* based, το Simulink υπολογίζει τις εξόδους της γεννήτριας μόνο στους χρόνους όπου υπάρχει μεταβολή επιπέδου.
Στο *Sample* based, υπολογίζει τις εξόδους της γεννήτριας στους χρόνους με βάση τον καθορισμό των δειγμάτων.
Η πρώτη περίπτωση κάνει πιο γρήγορη την εξομοίωση.

Διάβασμα αναλογικής εισόδου Arduino

Για να διαβάσουμε την αναλογική είσοδο του Arduino μέσω του Simulink φορτώνουμε στον Arduino το “adioes.pde”. Εκτελούμε την εντολή a=arduino(‘COM5’) και δημιουργούμε το μπλοκ (ArduinoIO Library):



Η έξοδος του παλμογράφου είναι:



ΧΡΗΣΗ ΤΟΥ MATLAB ΓΙΑ ΣΥΝΔΕΣΗ ΜΕ ΤΟΝ ARDUINO UNO

Χρησιμοποιώντας το account που χρησιμοποιήσαμε στη Mathwork κατεβάζουμε το ArduinoIO package και το αποσυμπιέζουμε (http://www.mathworks.com/matlabcentral/fileexchange/27843-arduino-io-package--slides-and-examples). Έπειτα, ανοίγουμε το Matlab ως Administrators και επιλέγουμε Set Path → Add with Subfolders τον φάκελο ArduinoIO. Εκτελούμε στο Matlab την εντολή install_arduino και οι φάκελοι του πακέτου τοποθετούνται στο path του Matlab. Ανοίγουμε το Arduino IDE και κάνουμε upload ένα από τα sketches *.pde που υπάρχουν στον φάκελο pde του ArduinoIO package. Κλείνουμε το Arduino IDE και στη συνέχεια μπορούμε να δώσουμε όποια εντολή θέλουμε αφού όμως πρώτα δημιουργήσουμε το αντικείμενο a που αφορά τον Arduino στην κατάλληλη πόρτα. Τοποθετούμε την εντολή a=arduino('port') όπου 'port' είναι η COM port στην οποία είναι συνδεδεμένος ο Arduino (π.χ. 'COM5' ή 'COM8'). Τώρα είμαστε σε θέση να χρησιμοποιούμε τις εντολές: a.pinMode, a.digitalRead, a.digitalWrite, a.analogRead, a.analogWrite. Μπορούμε ακόμα να χρησιμοποιήσουμε τις εξής εντολές:

- a.serial: Επιστρέφει το όνομα της σειριακής πόρτας.
- a.flush: Καθαρίζει τον buffer της σειριακής πόρτας.
- a.roundTrip: Στέλνει μια τιμή στον Arduino και επιστρέφει πίσω εφόσον λειτουργεί σωστά.

- a.delete: Διαγράφει το αντικείμενο του Arduino και ελευθερώνει τη σειριακή πόρτα.

Παρακάτω παρατίθενται ορισμένα παραδείγματα με την χρήση των εντολών αυτών:

```
a=arduino('COM5'); //connect the board
pinMode(a,4,'input'); //specify pin mode for pin 4
pinMode(a,13,'input'); //specify pin mode for pin 13
dv=digitalRead(a,4); // read digital input from pin 4
digitalWrite(a,13,dv); // output the digital value (0 or 1) to pin 13
av=analogRead(a,5); // read analog input from analog pin 5
av=(av/1023)*254; // normalize av from 0:1023 to 0:254
serial(a); //gets the name of the serial port to which the arduino is connected to
flush(a); // flushes the PC's serial input buffer
```

Εάν για οποιοδήποτε λόγο η σειριακή πόρτα δεν έχει απελευθερωθεί μετά το κλείσιμο του session, τότε εκτελούμε μια από τις παρακάτω εντολές:

```
// delete MATLAB serial connection on COM5
delete(instrfind({'Port'},{'COM5'}));

//delete all MATLAB serial connections
delete(instrfind('Type', 'serial'));
```

ΚΕΦΑΛΑΙΟ 4

Ανάπτυξη του PID Controller με την χρήση του Arduino και Matlab

4.1 Υλικά Κατασκευής

Σε αυτή την ενότητα θα παρουσιάσουμε τα εξαρτήματα που χρειαστήκαμε για να υλοποιήσουμε την κατασκευή που θα χρησιμοποιήσουμε για την καλύτερη ρύθμιση της θερμοκρασίας σε συνδυασμό με το MATLAB. Για το ηλεκτρονικό κομμάτι της κατασκευής χρειαστήκαμε τα εξής:

- ✓ ***1x RS232 - USB***
- ✓ ***1x LM35D Sensor***
- ✓ ***1x MPSA13-D***
- ✓ ***1x ATE - RESISTOR RB25 22R***
- ✓ ***1x LCD Display***
- ✓ ***1x Arduino Uno***
- ✓ ***1x Battery 7.5V***

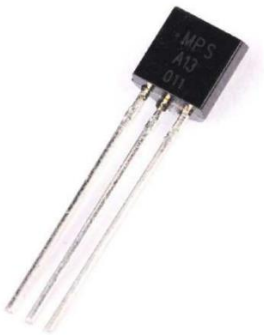
4.1.1 LM35D Sensor

Ο LM35D είναι αισθητήρας θερμοκρασίας ολοκληρωμένου κυκλώματος ακριβείας, του οποίου η τάση εξόδου είναι γραμμικώς ανάλογη με τη θερμοκρασία Κελσίου. Έχει διαβαθμιστεί να λειτουργεί σε μια περιοχή θερμοκρασίας από -55 °C έως 150 °C, λειτουργεί από 4 έως 30 Volt και έχει κατανάλωση ρεύματος μικρότερη από 60 μA. Το αισθητήριο θερμοκρασίας LM35D δημιουργεί την ανάδραση για να επιτευχθεί ο έλεγχος θερμοκρασίας.

Εικόνα 27:LM35D Sensor

4.1.2 MPSA13-D

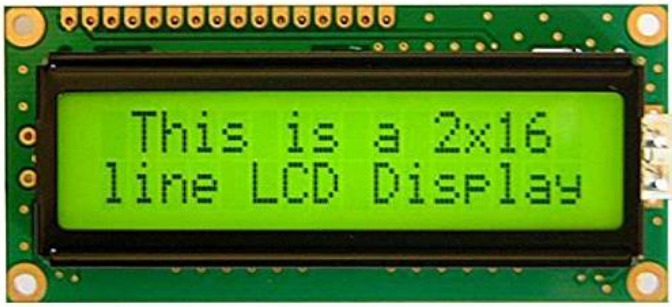
Το MPSA13 είναι ένα Transistor NPN ισχύος. Αποτελείται από τον εκπομπό, που κανονικά είναι συνδεδεμένος με την γείωση, την βάση η οποία χρησιμοποιείται για την ενεργοποίηση/απενεργοποίηση του τρανζίστορ και τον συλλέκτη.



Εικόνα 28:MPSA13-D

4.1.3 LCD DISPLAY

Η LCD (Liquid Crystal Display) είναι ένας τύπος επίπεδης οθόνης και μπορεί να εμφανίσει 16x2 χαρακτήρες σε 2 γραμμές. Θα μας φανεί ιδιαίτερα χρήσιμη στην απεικόνιση των τιμών που θέλουμε να υπολογίσουμε (temperature, Kp,Ki,Kd).



Εικόνα 29:LCD Display

4.1.4 ATE - RESISTOR RB25 22R

Η αντίσταση που θα χρησιμοποιήσουμε στην κατασκευή μας, η οποία κυρίως θα θερμαίνεται και την θερμοκρασία της θα ελέγχει το αισθητήριο.

Εικόνα 30: Resistor RB25 22R

4.2 Παρουσίαση της κατασκευής

Σε αυτή την ενότητα θα παρουσιάσουμε την εφαρμογή Arduino που προγραμματίστηκε για τον έλεγχο των τιμών P, I, D για την δημιουργία ενός PID ελεγκτή, ο οποίος θα είναι υπεύθυνος για την βέλτιστη σταθεροποίηση του συστήματος.

Τα βήματα που ακολουθήθηκαν είναι τα εξής:

- Ανοίγουμε το πρόγραμμα “Arduino.exe”.
- Από την πάνω μπάρα επιλέγουμε File→New.
- File → Save As → PID_exam.ino.

Οι βιβλιοθήκες που χρειάστηκαν για την συγγραφή του κώδικα είναι οι παρακάτω:

PID_v1.h: Μας βοηθάει να υπολογίσουμε τις τιμές P, I, D με σκοπό την σταθεροποίηση του συστήματος κατά την διακύμανση της θερμοκρασίας.

LiquidCrystal.h: Αυτή η βιβλιοθήκη επιτρέπει στον Arduino να ελέγχει την LCD οθόνη της κατασκευής μας. Η βιβλιοθήκη αυτή δουλεύει χρησιμοποιώντας είτε 4-bit είτε 8-bit mode.

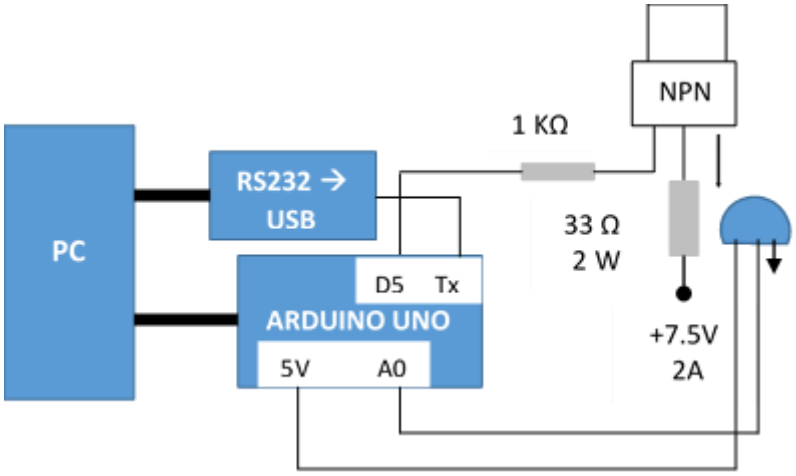
SoftwareSerial.h: Ο Arduino διαθέτει ενσωματωμένη υποστήριξη για σειριακή επικοινωνία στις ακίδες 0 , 1. Αυτή η λειτουργικότητα υποστηρίζεται μέσω ενός ενσωματωμένου υλικού στο τσιπ, του UART. Η βιβλιοθήκη SoftwareSerial έχει αναπτυχθεί για να επιτρέπει την σειριακή επικοινωνία και σε άλλες ακίδες του Arduino, χρησιμοποιώντας software για να αναπαράξει την παραπάνω λειτουργικότητα.

```
#include <PID_v1.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>

#define PIN_INPUT 0
#define PIN_OUTPUT 9

int t1 = 0;
int t2 = 0;
int temp1 =0;
int temp2=0;
int sensorValue = 0;  // variable to store the value coming from the sensor
int sensorPin = A0;    // select the input pin for the potentiometer


//Define Variables we'll be connecting to
double Setpoint, Input, Output;

//Specify the links and initial tuning parameters
double Kp=0.8, Ki=3, Kd=0.5;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);


const int rs = 3, en = 2, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup()
{
  //initialize the variables we're linked to
  Input = analogRead(PIN_INPUT);
  Setpoint = 70;
  Serial.begin(9600);
```

```
void setup()
{
  //initialize the variables we're linked to
  Input = analogRead(PIN_INPUT);
  Setpoint = 70;
  Serial.begin(9600);

  //turn the PID on
  myPID.SetMode(AUTOMATIC);
  t1=0;
  t2=0;
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.display();
    lcd.print("Temp: "+String(sensorValue));
   Serial.print("Kp:");
   Serial.print(Kp);
   Serial.print(" - ");
   Serial.print("Ki:");
   Serial.print(Ki);
   Serial.print(" - ");
   Serial.print("Kd:");
   Serial.print(Kd);
   Serial.println();
}


void loop()
{
  Input = analogRead(PIN_INPUT);
  myPID.Compute();
  analogWrite(PIN_OUTPUT, Output);

  Serial.print("Temp:");
  Serial.print(Input);
  Serial.print(" - ");
  Serial.print("PID:");
  Serial.print(Output);
  Serial.println();

  sensorValue = analogRead(sensorPin)/2;
   sensorValue = Input;
  temp2=sensorValue;
  if (t2>0){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Temp: "+String(sensorValue));
    lcd.setCursor(1,1);
    lcd.print("PID: "+String( Output));
    t2=0;
  } else {
    t2=t2+1;
  }
  delay(1000);
}
```

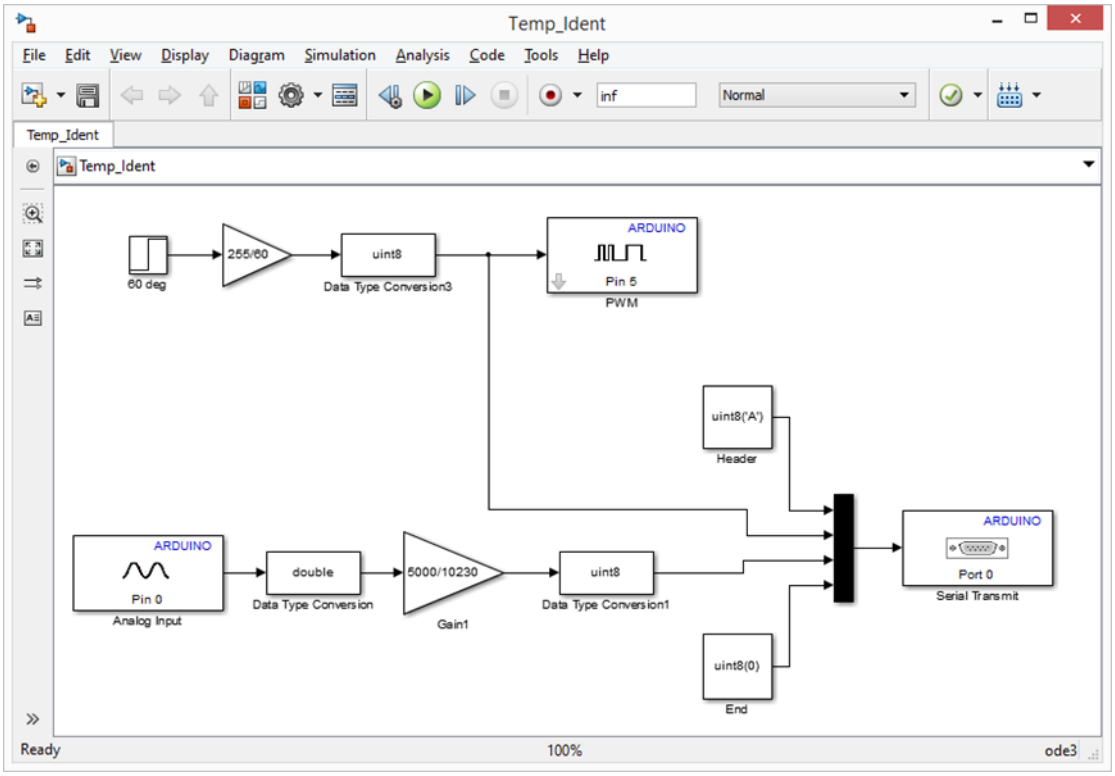
4.3 PID - Έλεγχος της Θερμοκρασίας

Το κυκλωματικό διάγραμμα που χρησιμοποιήθηκε για τον συντονισμό του PID με σκοπό τον έλεγχο της θερμοκρασίας μέσω του MATLAB και του Arduino UNO είναι το ακόλουθο:



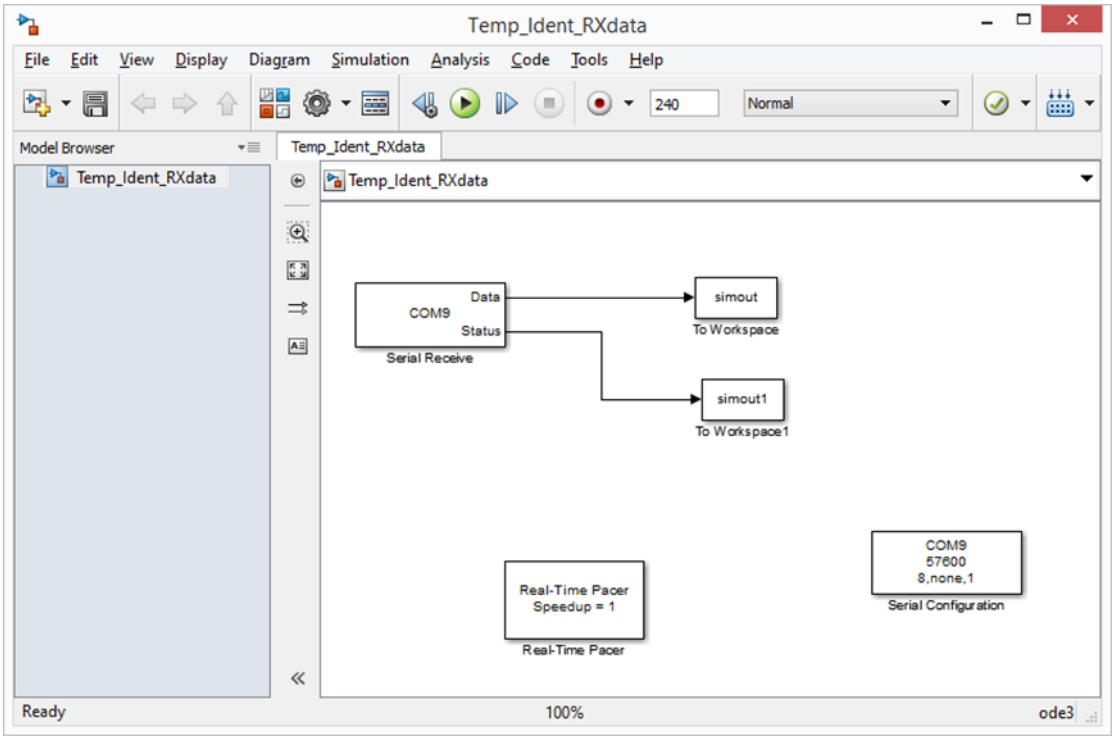
Πρώτα από όλα συνδέουμε τον Arduino σε μια θύρα USB στο PC (π.χ. COM7 port) και τον μετατροπέα RS232→USB σε μια άλλη θύρα (π.χ. COM9). Η αντίσταση θερμαίνεται με το σήμα PWM και πρέπει να διατηρεί σταθερή την θερμοκρασία της και ίση με το Set Point χρησιμοποιώντας έναν PID ελεγκτή, η υλοποίηση του οποίου γίνεται με έναν Arduino Uno, το Matlab καθώς και το Simulink του Matlab. Ο αισθητήρας θερμοκρασίας LM35D δημιουργεί την ανάδραση (feedback) για να πραγματοποιηθεί ο έλεγχος της θερμοκρασίας. Οι τιμές του αισθητηρίου θερμοκρασίας μεταφέρονται στο Matlab μέσω του Tx του Arduino.

Ανοίγουμε το περιβάλλον εργασίας του MATLAB και δημιουργούμε τον φάκελο myPID όπου και θα αποθηκεύουμε όλα τα αρχεία του Project. Έπειτα, ανοίγουμε το Simulink και δημιουργούμε το παρακάτω Simulink Model:

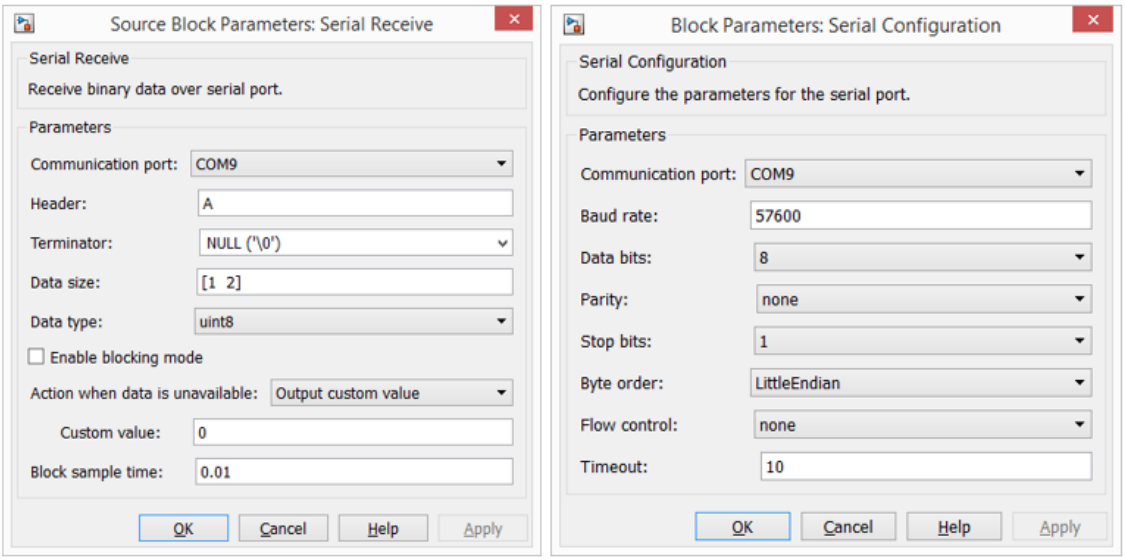


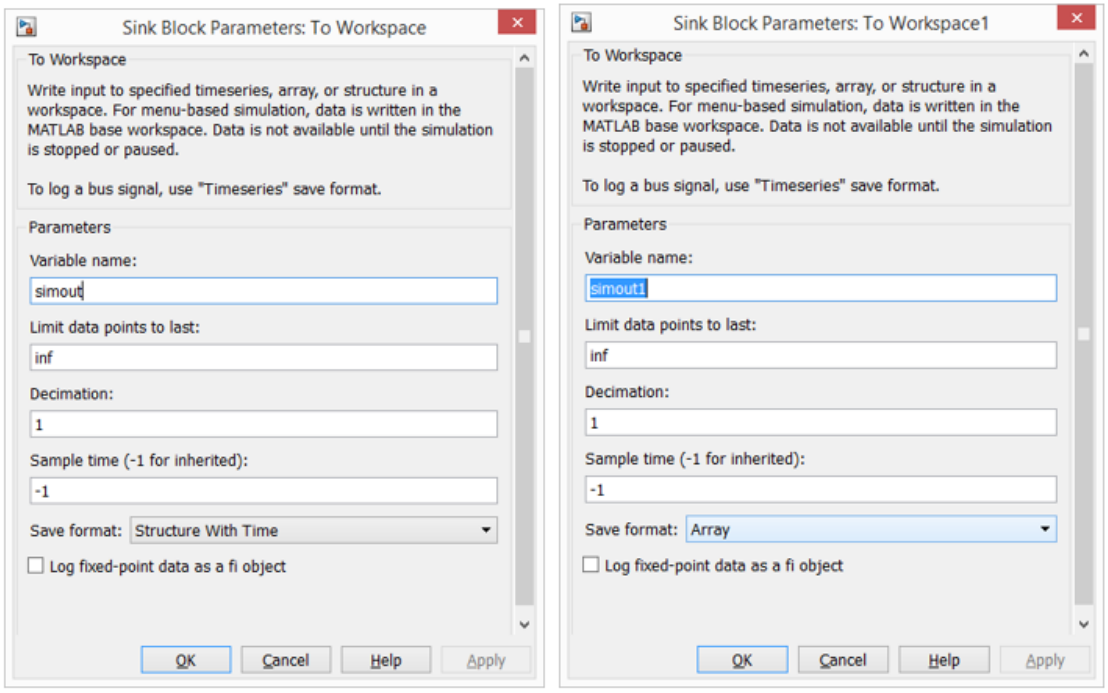
Δίνουμε ένα step=255 , θεωρούμε ότι είναι ένα Set Point μας 60 βαθμούς αυθαίρετα, το οποίο δημιουργεί συνεχή τάση +5V στη βάση του Τρανζίστορ. Έπειτα, η αντίσταση θερμαίνεται και ο αισθητήρας μεταφέρει σειριακά τη θερμοκρασία. Χρησιμοποιούμε την ενίσχυση 5000/10230 ώστε να μετατρέψουμε την αναλογική ένδειξη του μετατροπέα σε βαθμούς Κελσίου. Στην συνέχεια ορίζουμε από το menu τα μας επιλογές που επιθυμούμε για το μοντέλο μας. Ακολουθούμε Tools→ Run on Target Hardware → Options και επιλέγουμε Arduino Uno, COM7, τάση αναφοράς +5V και Baud Rate=57600. Επιλέγουμε Build ώστε να φορτώσουμε το πρόγραμμα στον Arduino.

Για να διαπιστώσουμε πως ανταποκρίνεται το σύστημα στην βηματική απόκριση, του τοποθετούμε εξωτερική τροφοδοσία +7.5V και κάνουμε Simulate για 120 λεπτά το παρακάτω μοντέλο:



Τα blocks Serial Receive, Serial Configuration και τα blocks δεδομένων simout, simout1 που συμπεριλαμβάνονται στο μοντέλο έχουν τις ακόλουθες αντίστοιχες μορφές:

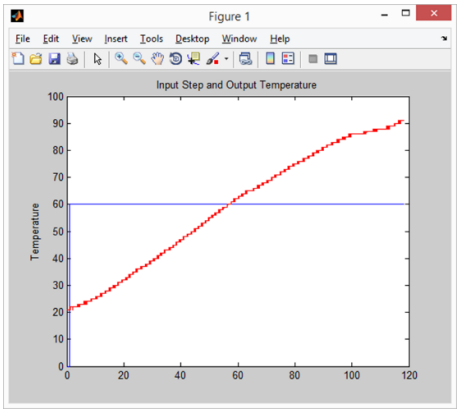




Το πρόγραμμα με το οποίο διαβάζουμε και απεικονίζουμε τις τιμές simout και simout1 είναι το εξής:

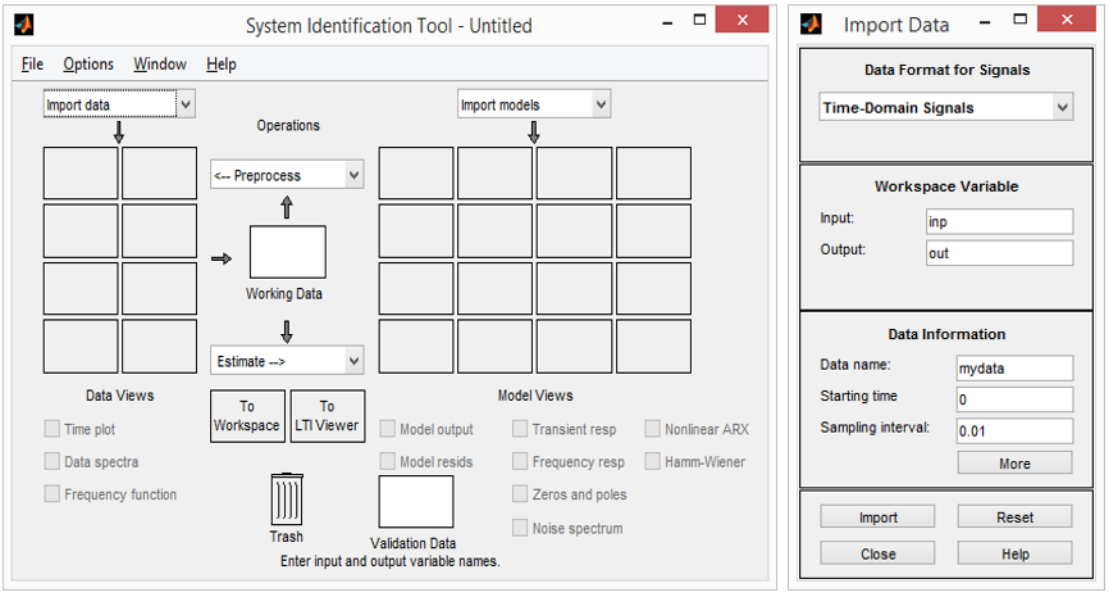
```
%%Without Controller
T=simout.signals.values;
clear t inp out stat t1;
t=simout.time;
inp=double(reshape(T(1,1,:),length(t),1));
inp=inp*60/255;      %in degrees Celsius
out=double(reshape(T(1,2,:),length(t),1)); %in degrees Celsius
stat=find(simout1==1);
inp=inp(stat);
out=out(stat);
t1=0.01*[0:size(stat)-1]';


figure,plot(t1,inp),hold on;
plot(t1,out,'r'),title('Input Step and Output
Temperature'),ylabel('Temperature'),hold off;
```

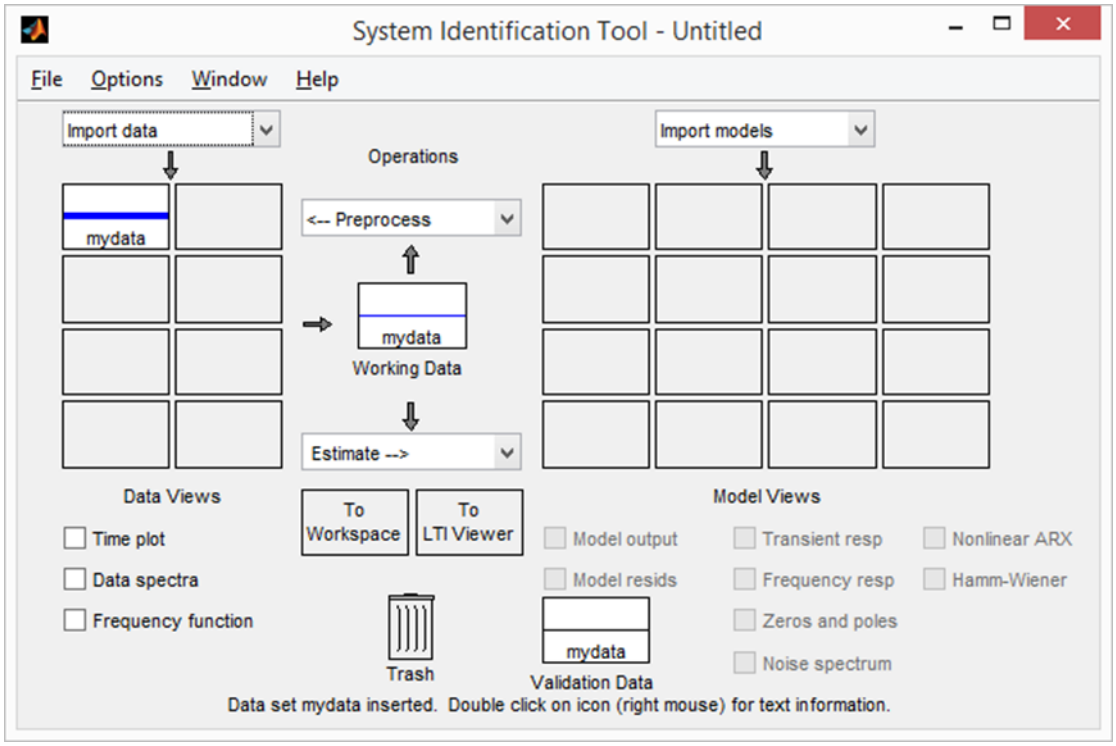
Προκύπτει η ακόλουθη γραφική παράσταση:



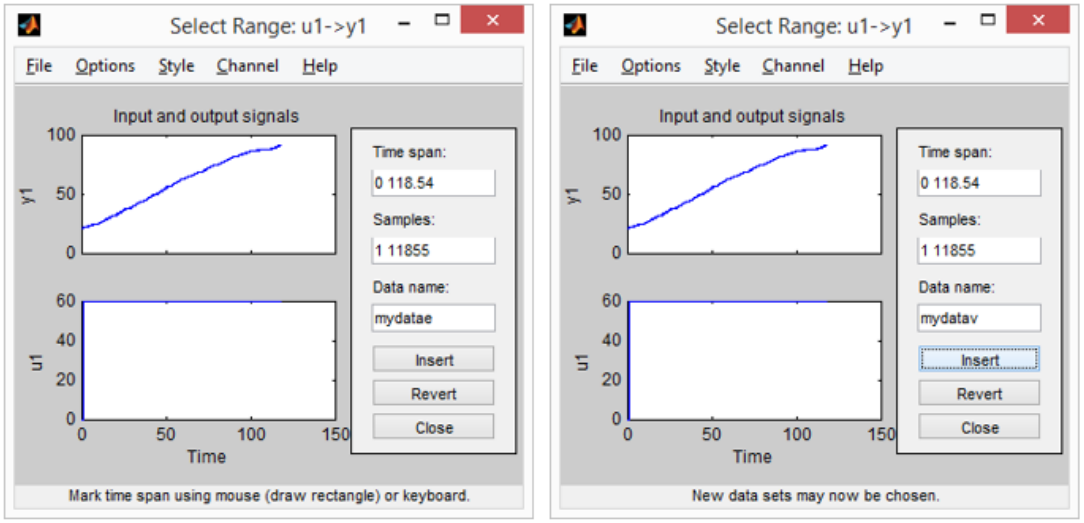
Με βάση αυτές τις τιμές προχωρούμε στην αναγνώριση του συστήματος (Identification). Ακολουθούμε τα παρακάτω βήματα:

- Εκτελούμε την εντολή “ident” και επιλέγοντας “Import Data” ορίζουμε τις μεταβλητές inp και out ως input και output αντίστοιχα, Starting Point το 0 και sampling interval 0.01s.

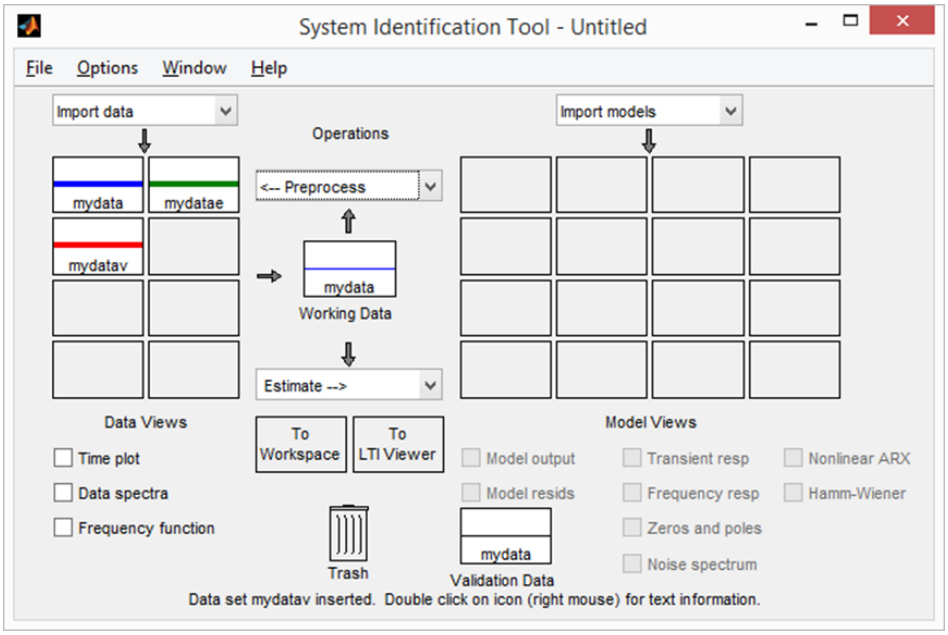




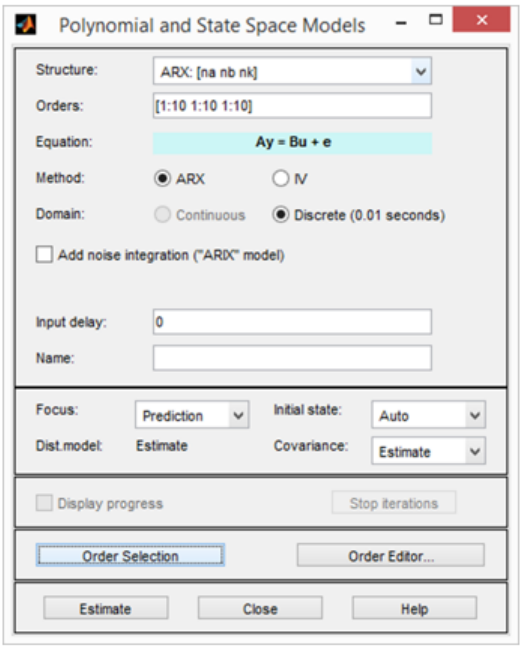
- Από το Operations επιλέγουμε “Select Range” και τοποθετούμε τα “Estimation data-mydatae” και τα “Validation data-mydatav” όπου στην δική μας περίπτωση είναι τα ίδια και πατάμε Insert.

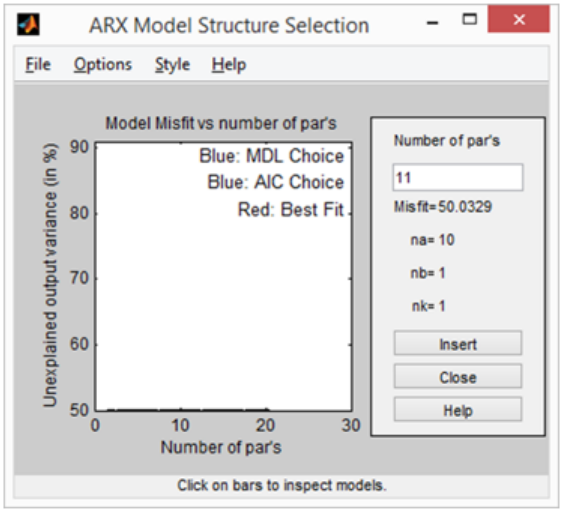


- Το παράθυρο αναγνώρισης προτύπων έχει πλέον την παρακάτω μορφή. Σύρουμε το “mydatae” στο “Working Data” και το “mydatav” στο “Validation Data”.

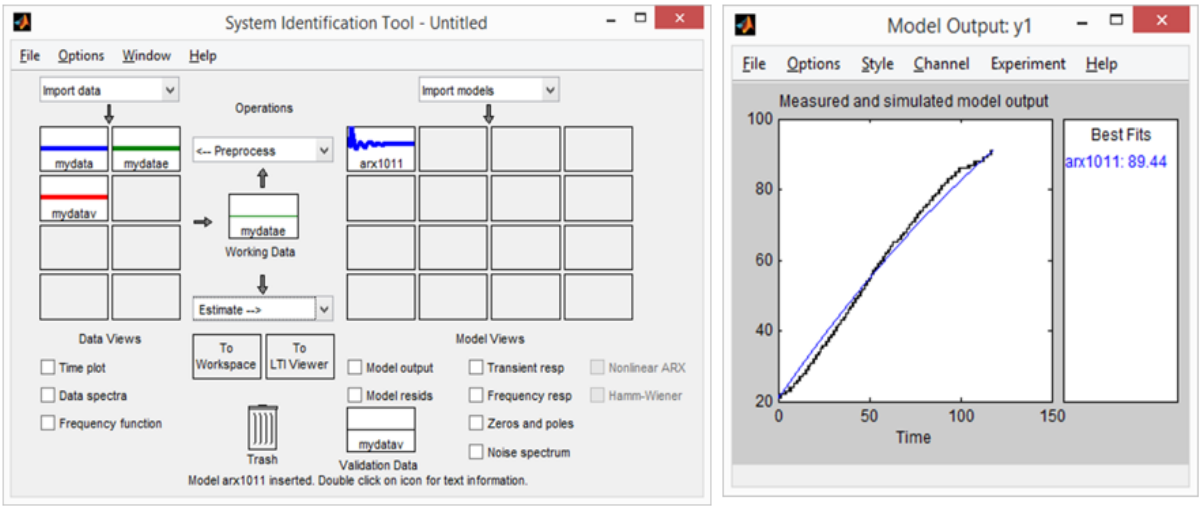


- Από το “Estimate” διαλέγουμε το “Polynomials Model” οπότε εμφανίζεται η οθόνη εκτίμησης και προσαρμογής στην καμπύλη των δεδομένων. Επιλέγοντας “Order Selection” εμφανίζεται η παρακάτω οθόνη όπου θα επιλέξουμε “Estimate” για να εμφανιστεί η οθόνη εκτίμησης και να κάνουμε Insert.





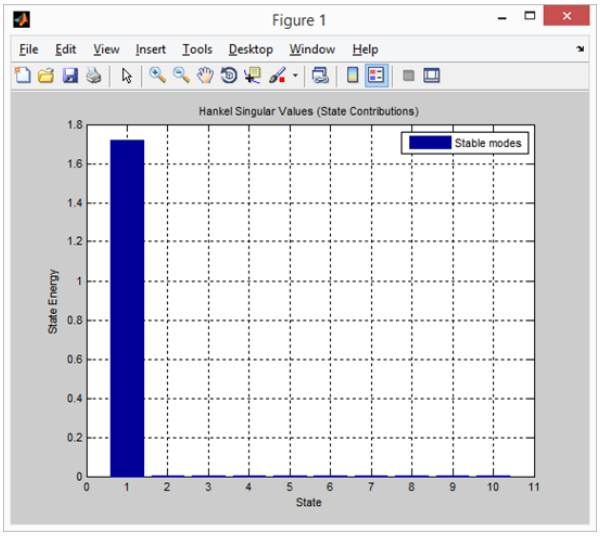
- Το μοντέλο “arx1011” εμφανίζεται στην οθόνη αναγνώρισης. Επιλέγοντας το “Model Output” παρατηρούμε πως εμφανίζεται η προσαρμογή στην καμπύλη των δεδομένων με επιτυχία 89% περίπου.



Στηριζόμενοι στην παραπάνω προσέγγιση προχωρούμε στην εξαγωγή της συνάρτησης μεταφοράς του “plant” ακολουθώντας τις εξής εντολές:

- tf1=tf(arx1011,’m’);
- tff=d2c(tf1,’Ts’,0.01);
- hsvd(tff)

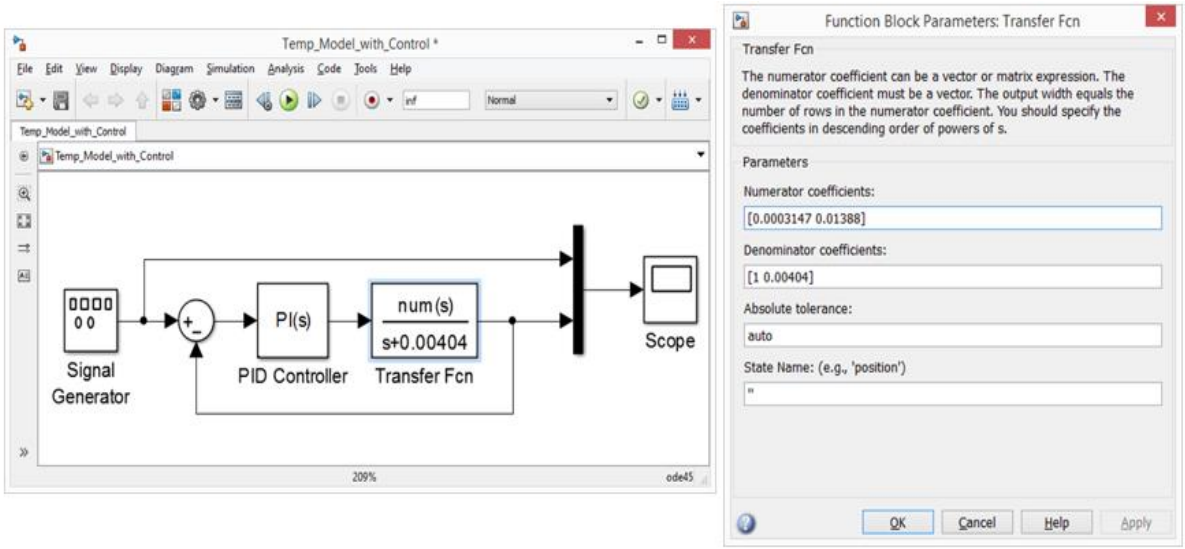
Από το διάγραμμα που προκύπτει φαίνεται πως ένα σύστημα πρώτου βαθμού συγκεντρώνει την μεγαλύτερη ενέργεια του συστήματος.



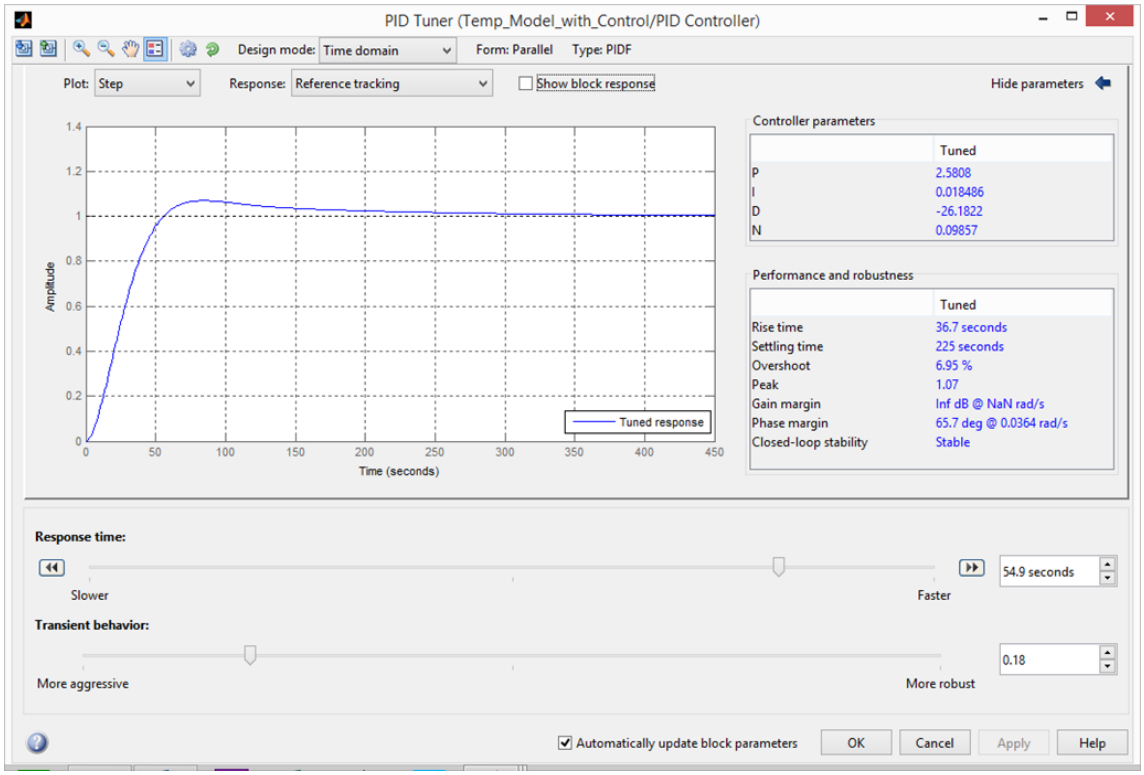
Έτσι με την εντολή “tfs=balred(tff,1)” από τη συνάρτηση μεταφοράς tff δημιουργούμε την tfs η οποία είναι πρώτου βαθμού.

$$tfs = \frac{0.0003147s + 0.01388}{s + 0.00404}$$

Έπειτα, ανοίγουμε το μοντέλο Temp_Model_with_Control.slx στο Simulink, επιλέγουμε την συνάρτηση μεταφοράς του συστήματος και τοποθετούμε τις παραπάνω τιμές.

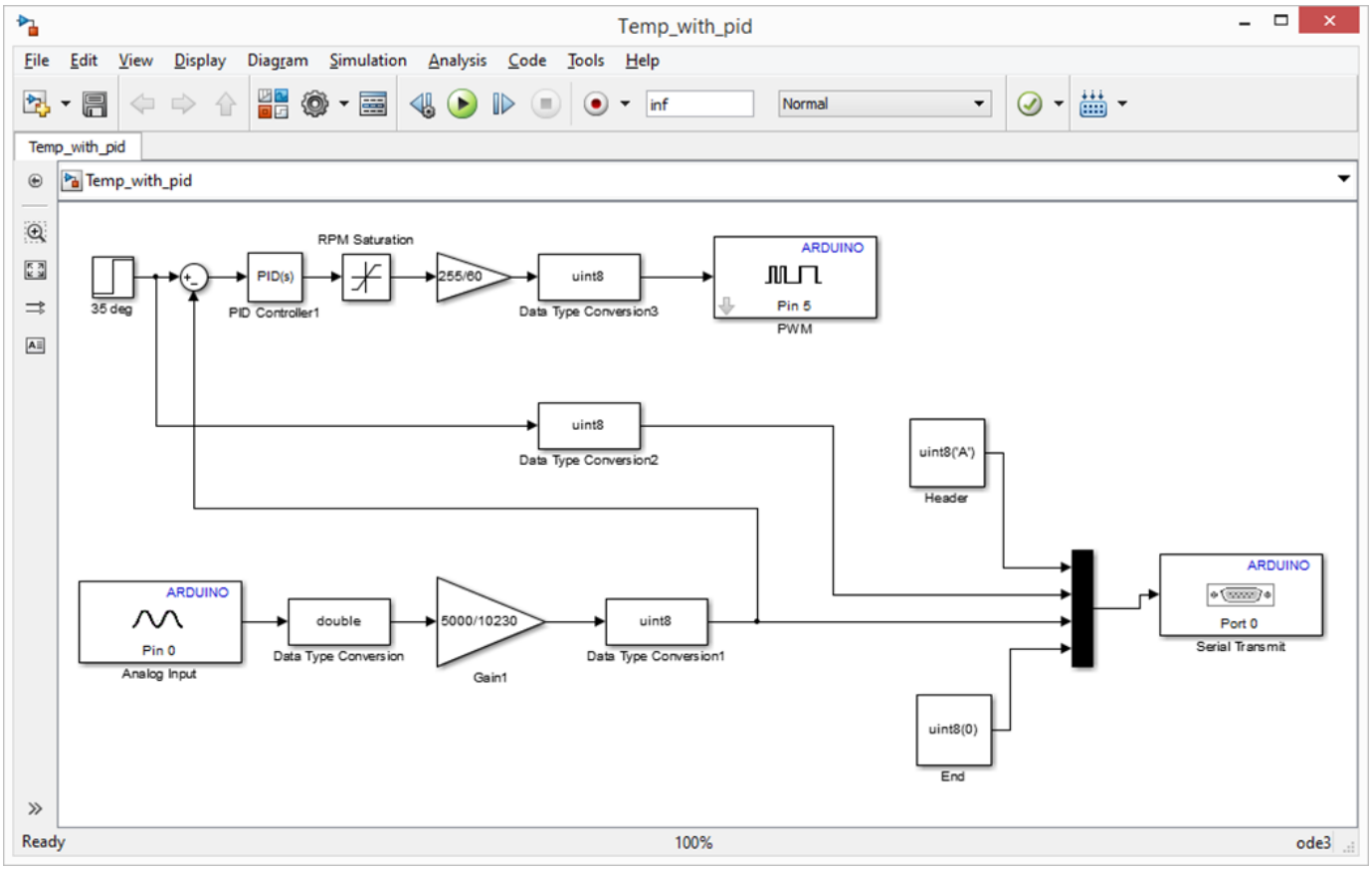


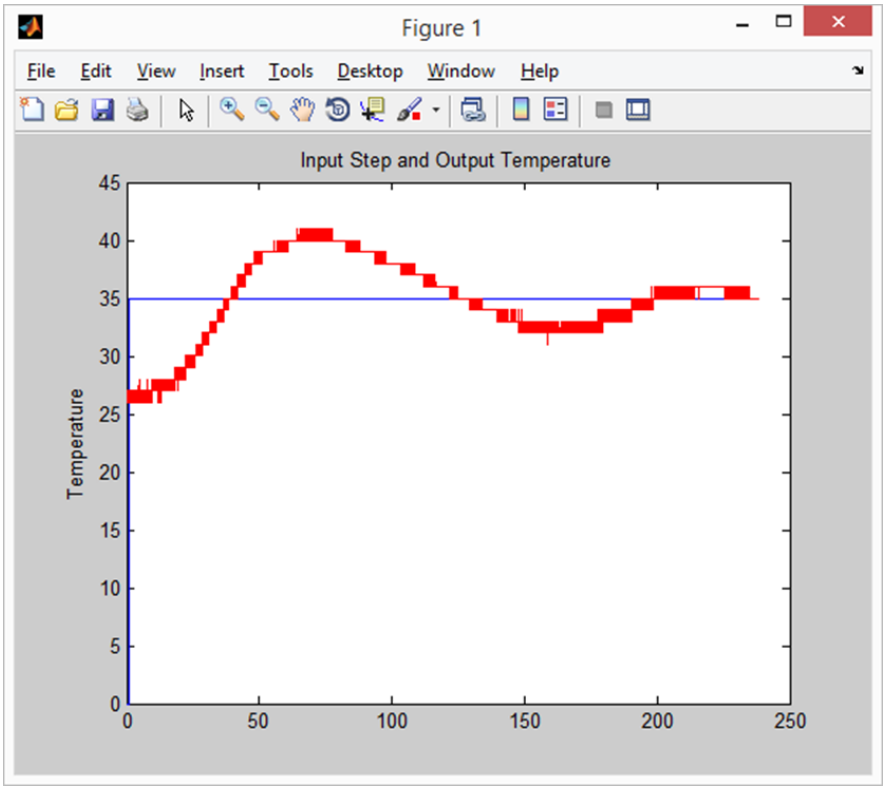
Επιλέγουμε τον ελεγκτή PID ώστε να τον συντονίσουμε βάσει της συνάρτησης μεταφοράς του συστήματος. Ρυθμίζουμε τους όρους για μια λογική απόκριση. Έστω ότι προκύπτουν οι τιμές P=2.58, I=0.015 και D=-26.18.



Δημιουργούμε το κλειστό κύκλωμα ελέγχου θερμοκρασίας με δεδομένες τις τιμές του PID και την συνάρτηση μεταφοράς του συστήματος.

- Έστω ότι τοποθετείται ένα Set Point=35 βαθμοί Κελσίου στο οποίο πρέπει να σταθεροποιηθεί το σύστημα.
- Δημιουργούμε έναν περιοριστή έτσι ώστε να μην ξεπερνιέται το όριο των 60 βαθμών.
- Επαναλαμβάνεται η αρχική διαδικασία. Το μπλοκ γίνεται Build και φορτώνεται στον Arduino. Στη συνέχεια τοποθετείται η εξωτερική τάση των +7.5V και γίνεται simulation του μοντέλου Temp_Ident_RXdata.slx. Τα δεδομένα που συλλέγονται simout και simout1 δείχνουν την παρακάτω καμπύλη αν εκτελεστεί το m-file Extract_data_with.m.
- Γίνεται φανερή η λειτουργία του ελεγκτή PID.





Μπορούμε να παίξουμε με τις τιμές P, I, και D ελεγκτή και να εξετάσουμε τη συμπεριφορά του συστήματος μέχρι να βρούμε τις πιο κατάλληλες τιμές ώστε το σύστημα να λειτουργεί σύμφωνα με τις επιθυμητές προδιαγραφές.

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ

Με την ολοκλήρωση της πτυχιακής μας εργασίας συμπεραίνουμε πως μια κατασκευή μπορεί εύκολα να υλοποιηθεί. Συγκεκριμένα, στην εργασία μας η χρήση του Arduino Uno μας βοήθησε να δημιουργήσουμε την δική μας ηλεκτρονική συσκευή και να την προγραμματίσουμε σύμφωνα με τις ανάγκες μας. Δεν είναι απαραίτητο να έχεις άψογες γνώσεις προγραμματισμού καθώς το Arduino προγραμματίζεται εύκολα.

Ακόμα, εξοικειωθήκαμε με το περιβάλλον εργασίας του «MATLAB». Το σημαντικότερο πλεονέκτημα του MATLAB ήταν το ότι μας επέτρεπε να δημιουργούμε μοντέλα SIMULINK στον Arduino. Έτσι, συντάσσαμε ένα μοντέλο Simulink, εκείνο εκτελείτο στον Arduino και έπειτα με την σειρά του επέστρεφε τις επιθυμητές τιμές στο Matlab.

Με αυτό τον τρόπο, θέλαμε να ελέγξουμε την συμπεριφορά του συστήματος μας αναπτύσσοντας έναν PID Controller για τον έλεγχο της θερμοκρασίας. Στόχος ήταν η θερμοκρασία μέσα στο «κουτί» που δημιουργήσαμε να διατηρείται σταθερή σύμφωνα με το Set Point που ορίζουμε κάθε φόρα και ταυτόχρονα η απόκριση του συστήματος να είναι όσο το δυνατόν πιο βέλτιστη/ομαλή γίνεται.

Όπως ήταν αναμενόμενο, η απόκριση του κλειστού συστήματος στις διαφορετικές τιμές των όρων του PID Controller διέφερε. Η αλλαγή των όρων αυτών είναι μια αργή διαδικασία στον Arduino. Για αυτό τον λόγο δημιουργήθηκε στο Simulink και μεταφέρθηκε σαν πίνακας στον Arduino. Καταλήγοντας λοιπόν, μπορεί η δημιουργία της κατασκευής να είναι μια εύκολη διαδικασία, όμως ο συντονισμός του συστήματος για την διασφάλιση της επιθυμητής απόδοσης είναι μια διαδικασία που θέλει πολλές δοκιμές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]A. P. Singh, —Speed Control of DC Motor using PID Controller Based on Matlab, vol. 4.

[2]V. K. Mehta, ELEMENTS OF ELECTRONIC AND INSTRUMENTATION. S. CHAND & COMPANY LIMITED, 1996.

[3]Juan W. Dixon, — Three-Phase Controlled Rectifier, in Power Electronics Handbook, Muhammad H. Rashid, Ed. Academic Press, 2001

[4]M. S. Microcontroller, — Low-cost embedded solution for PID controllers of DC motors, 2009.

[5]Gardner, J.W 2000 Μικροαισθητήρες – Αρχές και Εφαρμογές. Θεσσαλονίκη : Εκδόσεις Τζιόλα

[6]Σήματα και Συστήματα με Matlab, Παλαμίδης – Βελώνη – Σύγχρονη Εκδοτική - 2008.

[7]Σήματα και Συστήματα για Τεχνολόγους, Φωτόπουλος – Βελώνη – Σύγχρονη Εκδοτική

[8] Συστήματα Αυτομάτου Ελέγχου, Βελώνη Αναστασία – Εκδόσεις Τζιόλα

[9]Modern Signals and Systems, Kwakernaak - Sivan, Prentice - Hall Intl., 1991.

[10]Ziegler-Nichols Tuning Rules for PID, Microstar Laboratories.

[11] Schmidt, M. ["Arduino: A Quick Start Guide"], *Pragmatic Bookshelf*, January 22 2011

[12] Atmel ATmega328: http://www.atmel.com/devices/atmega328.aspx

[13] https://en.wikipedia.org/wiki/PID_controller#History

[14] https://el.wikipedia.org/wiki/Arduino

[15] Igoe, Tom (April 4, 2006). "Arduino Shields for Prototyping". tigoe.net

[16] https://www.mathworks.com/discovery/pid-control.html

[17]https://www.motioncontroltips.com/how-to-address-overshoot-in-servo-control/