

Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

ΗΡΥ 411

Σύστημα αυτόματου εντοπισμού ουρανίων σωμάτων με τηλεσκόπιο

Milestone 2

Μοντελοποίηση με χρήση Matlab

Έλεγχος περιστροφικής βάσης μέσω του AVR

Callibration βάσης

Γιακουμάκης Πάρις-Παύλος

Γκιώνης Νίκος

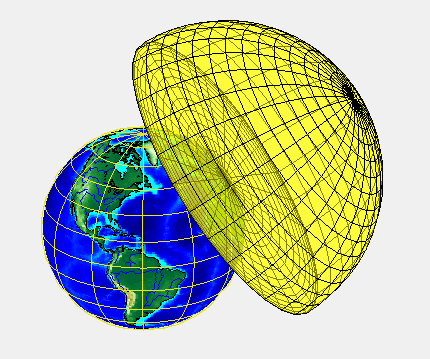
Πορτοκαλάκης Πέτρος

27 Οκτωβρίου 2017

# Εισαγωγή

# Για το milestone 2 η μελέτη και η εργασία κινήθηκε γύρω από τρία βασικά θέματα-άξονες. Σε πρώτη φάση υλοποιήθηκε η μοντελοποίηση του συστήματος με χρήση Matlab. Στη συνέχεια υλοποιήθηκε το βασικό κύκλωμα του συστήματος που συνδέει τον μικροελεγκτή με την περιστροφική βάση κα τέλος μελετήθηκε η ταχύτητα περιστροφής της βάσης, προκειμένου να γίνει μια διαδικασία calibration, που θα επιτρέψει τη δημιουργία κατάλληλων ρουτινών ελεγχόμενης κίνης της βάσης.

# Μοντελοποίηση σε Matlab

Η μοντελοποίηση του συστήματος που υλοποιήθηκε περιλαμβάνει 2 μέρη. Αρχικά, στο πρώτο σχήμα (δεξιά) φαίνεται η γη μαζί με ένα ημισφαίριο κεντραρισμένο σε μία περιοχή. Με αυτό το σχήμα μοντελοποιείται το σύστημα observer – celestial sphere, ή πιο συγκεκριμένα, το πως αντιλαμβάνεται ένας παρατηρητής τον ουρανό.

Χρησιμοποιώντας το HORIZONS System του Jet Propulsion Laboratory της NASA, και την δυνατότητά του αποστολής δεδομένων σε e-mail, μοντελοπήθηκε σε MATLAB η κίνηση ενός ουράνιου σώματος.

Τα δεδομένα από το e-mail μεταφέρθηκαν σε ένα αρχείο .txt, και έπειτα το διαβάζουμε.

Το αρχείο .txt έχει το ακόλουθο format:

2017-Dec-02 20:14 m 30.9625 -60.1750

2017-Dec-02 20:16 m 31.8575 -59.9630

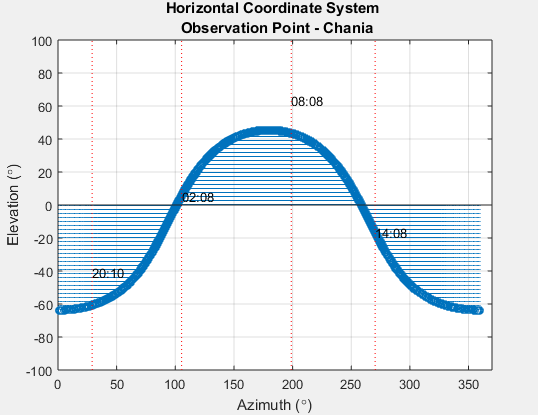
2017-Dec-02 20:18 m 32.7415 -59.7456

2017-Dec-02 20:20 m 33.6146 -59.5229

Από αριστερά προς τα δεξιά, οι στήλες του αρχείου περιλαμβάνουν ημερομηνία, ώρα, ένα σύμβολο που δεν μας αφορά, την συντεταγμένη azimuth, και την συντεταγμένη elevation.

Έπειτα, τοποθετούνται σε ξεχωριστά arrays η ώρα, η συντεταγμένη azimuth και η συντεταγμένη elevation. Για την εφαρμογή που υλοποιήθηκε η ημερομηνία είναι παράμετρος που δεν λαμβάνεται υπόψην, μιας και το σύστημα θα λειτουργήσει για λίγη ώρα σχετικά με την αλλαγή της ημέρας.

Ακολουθεί παράδειγμα με δεδομένα από την 02/12/2017 20:08 έως την 3/12/2017 20:08, με ανανέωση συντεταγμένων κάθε 2 λεπτά.



Πιο συγκεκριμένα, το παραπάνω γράφημα περιγράφει την κίνηση ενός ουρανίου σώματος, κατα την διάρκεια ενός 24-ώρου. Έχουν τοποθετηθεί 4 διαφορετικές στιγμές της ημέρας, για την καλύτερη αντίληψη της κίνησης του σώματος, αλλά και την δυνατότητα επαλήθευσης του συστήματος, σε περίπτωση ενός real time testing.

Λίγα λόγια για το Horizontal Coordinate System, το οποίο χρησιμοποιεί όπως αναφέρθηκαν πριν, azimuth και elevation. Το azimuth πρακτικά, υποδηλώνει τις μοίρες που πρέπει να περιστραφεί η βάση μας γύρω από τον ορίζοντα του παρατηρητή, με σημείο 0° τον βορρά, και φορά προς την ανατολή. Το elevation, υποδηλώνει τις μοίρες μεταξύ του ορίζοντα του παρατηρητή και του ουράνιου σώματος που μελετάμε.

# Έλεγχος περιστροφικής βάσης μέσω AVR

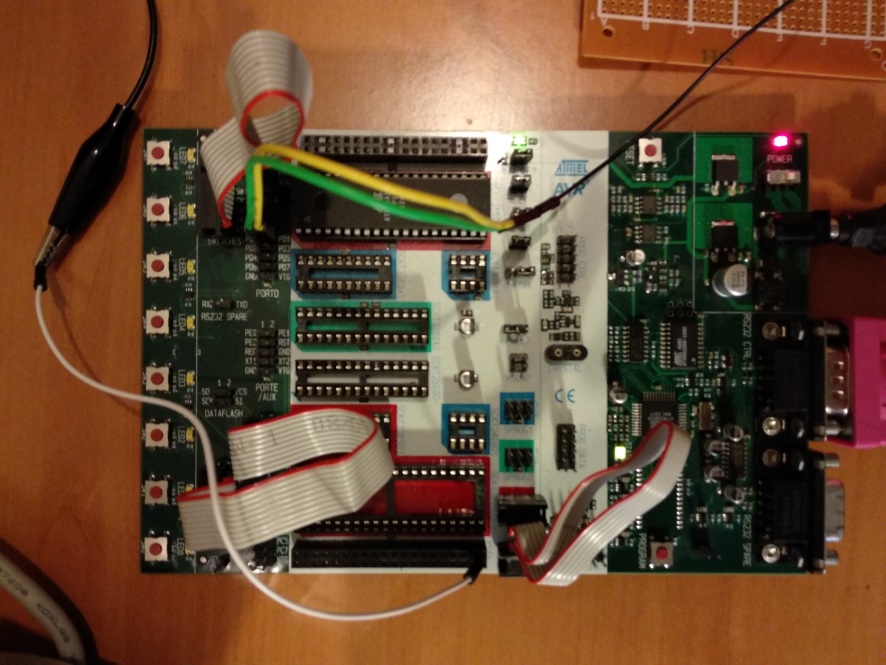
Το σύστημα που έχει υλοποιηθεί έως αυτό το σημείο, το οποίο αποτελεί σημαντικό μέρος και του τελικού συστήματος, αποτελείται από τρία βασικά υποσυστήματα. Το πρώτο είναι η περιστροφική βάση, το δεύτερο ο μικροελεγκτής ATmega16L, μαζί με την αναπτυξιακή πλακέτα STK500, και τέλος το κύκλωμα που εξασφαλίζει να γίνεται σωστά η διεπαφή και η επικοινωνία των δύο προαναφερθέντων συστημάτων.

## Περιστροφική Βάση

Η περιστροφική βάση AS20RS485 του εργαστηρίου που χρησιμοποιείται, έχει τα εξής χαρακτηριστικά: Η διεπαφή της γίνεται μέσω RS485, ενώ η τροφοδοσία της είναι AC 24V. Για την τροφοδοσία χρησιμοποιήθηκε κατάλληλος μετασχηματιστής που μας δόθηκε από τον κ. Κιμιωνή. Επιπλέον, η βάση επιτρέπει μέσω dip-switches τη ρύθμιση του πρωτοκόλλου επικοινωνίας καθώς και του baud rate της επικοινωνίας. Το πρωτόκολλο που επιλέχθηκε είναι το PELCO-P με baud rate 9600bps. Τα χαρακτηριστικά του πρωτοκόλλου, καθώς και ο κώδικας χειρισμού θα αναλυθούν παρακάτω.

## AVR/STK500

Ο μικροελεγκτής AVR έχει τη δυνατότητα να στέλνει και να δέχεται δεδομένα μέσω UART. Συγκεκριμένα, από το pin PD0 δέχεται δεδομένα ενώ από το PD1 στέλνει δεδομένα. Τα δεδομένα που στέλνει ο AVR είναι στάθμης TTL, το οποίο σημαίνει ότι το λογικό ‘0’ αντιστοιχίζεται στα 0 Volts ενώ το λογικό ‘1’ στα +5V.

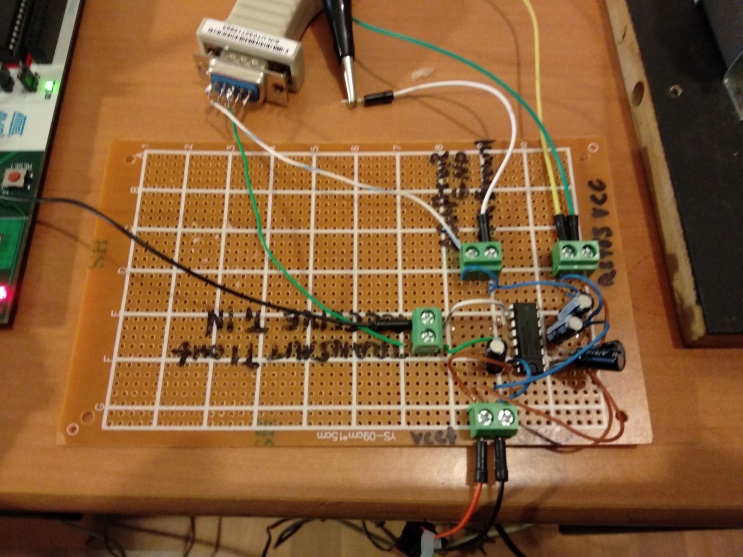


Στην παραπάνω εικόνα παρουσιάζετα η σύνδεση του STK500 και του AVR με το υπόλοιπο κύκλωμα. Στο δεξί άκρο φαίνονται η τροφοδοσία και η σύνδεση με τον υπολογιστή για τον προγραμματισμό του AVR, τα καλώδια κίτρινο-πράσινο προέρχονται από τα PD0 και PD1, από τα οποία μέσω του μαύρου καλωδίου λαμβάνεται στη συνέχεια το PD1 και κατευθύνεται προς το κύκλωμα. Το σήμα αυτό είναι το transmit σήμα του UART του AVR. Τέλος φαίνεται η γείωση, μέσω του λευκού καλωδίου, που εξασφαλίζει κοινή τάση αναφοράς με όλα τα υπόλοιπα υποσυστήματα.

## Κύκλωμα διεπαφής

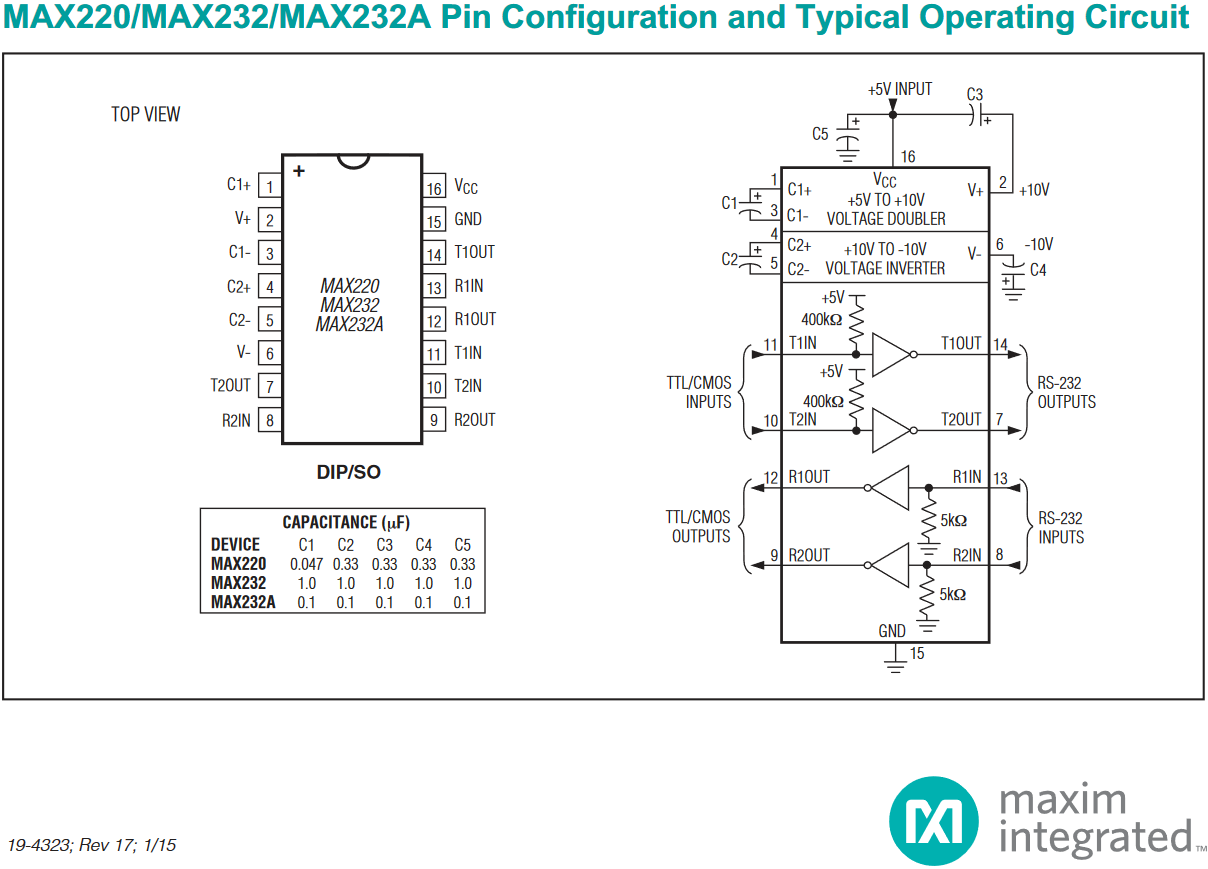
Όπως γίνεται κατανοητό, υπάρχει η ανάγκη μετατροπής του σήματος στάθμης TTL του AVR σε σήμα RS485, προκειμένου να ληφθεί σωστά από την βάση. Ο ρόλος του ενδιάμεσου κυκλώματος είναι να εξασφαλίσει την επικοινωνία αυτών των υποσυστημάτων. Αυτό επιτυγχάνεται μέσω δύο υποκυκλωμάτων. Το πρώτο custom PCB, το οποίο βασίζεται στο ολοκληρωμένο MAX232, μετατρέπει το σήμα από TTL σε RS232. Στη συνέχεια η έξοδος του PCB συνδέεται στο δεύτερο υποκύκλωμα, έναν RS232to485 adaptor. Μέσω της διαδικασίας αυτής, το σήμα TTL μεταφράστηκε σε σήμα RS485.

Στην παρακάτω εικόνα φαίνεται το custom PCB που υλοποιήθηκε. Διακρίνονται το ολοκληρωμένο MAX232 μαζί με τους απαραίτητους πυκνωτές για τη λειτουργία του, καθώς και τέσσερα screw connectors για τη σύνδεση των καλωδίων.



Στον κάτω screw connector συνδέεται η τάση τροφοδοσίας του MAX232, ένα τροφοδοτικό 5V DC. Στον αριστερό connector συνδέεται το σήμα TTL που στέλνει ο AVR, και εξέρχεται το σήμα RS232 που οδηγείται στον adaptor. Στον πάνω αριστερά connector, συνδέονται οι δύο γειώσεις του STK500 και του RS232to485 adaptor, έτσι ώστε να έχουν όλα τα υποκυκλώματα κοινή γείωση. Τέλος στον πάνω δεξιά connector έχει μεταφερθεί η DC 5V τροφοδοσία, ώστε να συνδεθούν τα σήματα VCC και Gnd του RS232to485 adaptor.

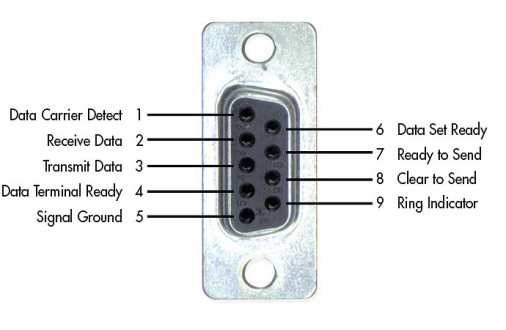
Η συνδεσμολογία έγινε με βάση το datasheet του MAXIM Max232:



## Στη συνέχεια παρουσιάζεται ο RS232to485 adaptor. Όπως φαίνεται και στην εικόνα, από τη μία μεριά δέχεται είσοδο το σήμα RS232 που προέρχεται από το custom PCB, και από την άλλη μεριά έχει ως έξοδο τα δύο σήματα RXD (receive) και ΤXD (transmit) της RS485, καθώς και τα δύο pins, GND και VCC που εξασφαλίζουν την απαραίτητη τροφοδοσία που απαιτεί η RS485:



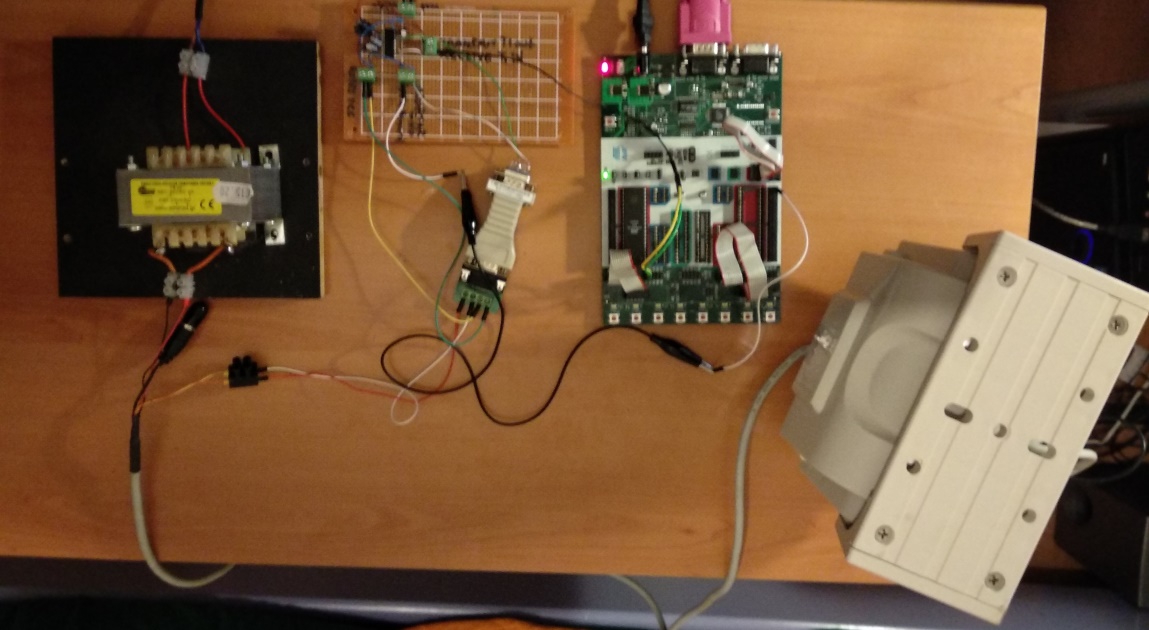
Η συνδεσμολογία από τη μεριά της RS232 αποσαφηνίζεται από την παρακάτω εικόνα, σε συνδυασμό με τις εικόνες του Custom PCB:



Τα pins που αξιοποιήθηκαν στην πράξη ήταν η γείωση (λευκό καλώδιο) και το Transmit data (πράσινο καλώδιο). Τα data προέρχονται από την έξοδο του MAX232.

Για την RS485 χρησιμοποιούνται δύο σήματα, για την μετάδοση των δεδομένων. Για τη μετάδοση του λογικού ‘0’ το ένα σήμα, Τ/R+, γίνεται +5Volts και το άλλο, Τ/R-, 0Volts, ενώ για τη μετάδοση του λογικού ‘1’ γίνεται το αντίθετο. Έτσι επιτυγχάνεται half-duplex επικοινωνία, όπου μόνο στέλνονται δεδομένα προς τη βάση. Επομένως, τα τέσσερα σήματα που συνδέονται στη μεριά του RS485 του adaptor είναι: Δύο για την τροφοδοσία των 5Volts DC (VCC και GND) και δύο για τη μεταφορά δεδομένων (Τ/R+, Τ/R-) που συνδέονται στη βάση.

Στη συνέχεια παρουσιάζεται η πλήρης συνδεσμολογία. Συνοψίζοντας, η ακολουθία των υποσυστημάτων έχει ως εξής: Ο AVR στέλνει, μέσω UART τις εντολές σε στάθμη TTL. Αυτές λαμβάνονται από το Custom PCB και μετατρέπονται μέσω του MAX232 σε σήμα RS232. Το σήμα αυτό ακολούθως εισέρχεται στον RS232to485 adaptor, και από την έξοδο του adaptor λαμβάνεται η εντολή σε RS485 στάθμη πλέον, οπότε και συνδέεται με τη διεπαφή της βάσης. Τέλος δεξιά φαίνεται ο μετασχηματιστής που τροφοδοτεί την περιστροφική βάση.



## Κώδικας χειρισμού της βάσης

Για την κατανόηση του κώδικα και της λειτουργίας του είναι απαραίτητη η μελέτη του πρωτοκόλλου PELCO-P που χρησιμοποιεί η περιστροφική βάση.

Η δομή του πρωτοκόλλου συνοψίζεται στον ακόλουθο πίνακα:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte 1** | **Byte 2** | **Byte 3** | **Byte 4** | **Byte 5** | **Byte 6** | **Byte 7** | **Byte 8** |
| STX | Camera Address | Data 1 | Data 2 | Data 3 | Data 4 | ETX | Checksum |

Όπως φαίνεται, μια εντολή PELCO-P αποτελείται από 8 bytes. Το πρώτο byte (πάντα 0xΑ0) λειτουργεί ως start byte (start of text). Το δεύτερο byte αφορά στη διεύθυνση της κάμερας, που έχει ρυθμιστεί μέσω dip switches. Τα data bytes παρουσιαζονται αναλυτικά παρακάτω. Το byte 7 (πάντα 0xAF) είναι το end byte (end of text). Το byte 8 χρησιμοποιείται για έλεγχο σφαλμάτων στη μετάδοση και υπολογίζεται ως το xOR των προηγούμενων bytes.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| Data 1 | Fixed to 0 | Camera On | Auto Scan On | Camera On/Off | Iris Close | Iris Open | Focus Near | Focus Far |
| Data 2 | Fixed to 0 | Zoom Wide | Zoom Tele | Tilt Down | Tilt Up | Pan Left | Pan Right | 0 (for pan / tilt) |
| Data 3 | Pan speed 00 (stop) to 3F (high speed) and 40 for Turbo | | | | | | | |
| Data 4 | Tilt speed 00 (stop) to 3F (high speed) | | | | | | | |

Από τα παραπάνω bits, αυτά που χρησιμοποιήθηκαν είναι τα bits 4 έως bit1 του byte Data 2 που καθορίζουν τον τύπο της κίνησης που θα κάνει η βάση. Τα bytes Data 3 και Data 4 τέθηκαν στην προκαθορισμένη τιμή 0x20, καθώς η συγκεκριμένη βάση δεν υποστηρίζει διαφορετικές ταχύτητες περιστροφής. Το Data 1 byte τέθηκε σε 0x00 καθώς αφορά σε χειρισμό κάμερας, όπου δεν χρησιμοποιείται στην εφαρμογή που υποποιείται.

Με βάση τις παραπάνω πληροφορίες, οι βασικές συναρτήσεις left( ), right( ), up( ), down( ), γίνονται ξεκάθαρες.

Όσον αφορά στις συναρτήσεις που κάνουν τις αντίστοιχες κινήσεις αλλά παίρνουν ως όρισμα και τις μοίρες της εκάστοτε περιστροφικής κίνησης, να αναφερθεί ότι η λειτουργία τους βασίζεται στην εισαγωγή μιας καθυστέρησης μέσω της delay\_ms( ), όπου αφήνει την αντίστοιχη απλή εντολής περιστροφής ενεργή για κατάλληλο χρόνο, πρωτού κληθεί η συνάρτηση stop( ). Για να καθοριστούν οι παραπάνω χρόνοι καθυστέρησης, έγινε μια διαδικασία calibration.

## Callibration Βάσης

Αρχικά υλοποιήθηκε η συνάρτηση gimbal\_init( ) η οποία φέρνει την περιστροφική βάση σε μια προκαθορισμένη θέση. Η θέση αυτή είναι η θέση τερματισμού κατά τη left κίνηση όσον αφορά στην περιστροφή pan, και η θέση τερματισμού κατά την down κίνηση όσον αφορά στην περιστροφή tilt. Η διαδικασία calibration έγινε ως εξής: Για την κίνηση pan, η βάση ξεκινώντας από την αρχική της θέση περιστρεφόταν με σκοπό να κάνει περιστροφή 270 μοιρών. Μέσω πολλών δοκιμών, καθορίστηκε ο χρόνος εκείνος που εξασφάλιζε την περιστροφή ακριβώς 270 μοιρών. Οι συναρτήσεις τύπου “\_degree( )” λαμβάνουν σαν όρισμα τις μοίρες της κίνησης, και εκτελούν ένα while-loop για την τιμή του ορίσματος. Ως εκ τούτου, μέσω της διαδικασίας που προαναφέρθηκε, στην πραγματικότητα προσδιορίστηκε ο χρόνος περιστροφής κατά μία μοίρα, ο οποίος εκτελείται 270 φορές. Ο λόγος που επιλέχθηκε η περιστροφή 270 μοιρών είναι αρχικά διότι μας παρείχε καλή οπτική επιβεβαίωση της γωνίας περιστροφής, αλλά κυρίως διότι, εφόσον εξασφαλίσαμε ότι το σφάλμα της μίας μοίρας ήταν το ελάχιστο δυνατό για μια τόσο μεγάλη γωνία, τότε σε όλες τις γωνίες μικρότερες των 270 μοιρών το σφάλμα θα είναι επίσης πολύ μικρό, ενώ για μεγαλύτερες γωνίες, το σφάλμα θα προσθεθεί το πολύ 90 φορές επιπλέον (μέχρι τη στροφή των 360 μοιρών). Ο λόγος που δεν επιλέχθηκε ως γωνία περιστροφής για το calibration οι 360 μοίρες είναι το γεγονός ότι η βάση δεν κάνει πλήρη περιστροφή 360 μοιρών, αλλά κάτι λιγότερο. Ως εκ τούτου, δεν θα μπορούσαμε να έχουμε καλή οπτική επιβεβαίωση της περιστροφής.

Με παρόμοιο τρόπο έγινε το calibration και για την κίνηση tilt, έχοντας ως αρχική θέση τις -49 μοίρες, και ελέγχοντας τον χρόνο που απαιτείται για να φτάσει η βάση σε κλίση 0 μοιρών. Η κίνηση σε αυτή την περίπτωση είναι σαφώς μικρότερη (το μέγιστο είναι 98 μοίρες αλλαγή) και συνεπώς η διαδικασία calibration σαφώς ευκολότερη.

# Παρατηρήσεις

Να σημειωθεί ότι δεν ελέγχθηκε ακόμη η είσοδος δεδομένων από planetary ephimerides και η αξιοποίησή τους στον AVR, ωστόσο έχει μελετηθεί η δομή τους και είναι το τελικό βήμα που απομένει, με τη σύνδεση ενδεχομένως ενός laser pointer (για τον εντοπισμό του στόχου στο νυχτερινό ουρανό) προκειμένου να ολοκληρωθεί το project. Ήδη η αξιοποίηση των πινάκων αυτών με τις θέσεις των ουρανίων σωμάτων χρησιμοποιήθηκε για την εξαγωγή του μοντέλου που εμφανίζεται στην αρχή της αναφοράς. Επίσης να σημειωθεί ότι το σύστημα συντεταγμένων που θα αξιοποιηθεί εν τέλει δεν είναι το right ascension και declination, το οποίο είναι μια εμπειρική-πρακτική λύση για τον εντοπισμό ουράνιων σωμάτων, αλλά, όπως φαίνεται και στα μοντέλα, το σύστημα azimuth-elevation.

Σημαντική βοήθεια στην κατανόηση της λειτουργίας των υποσυστημάτων και στη μεταφορά των εντολών μας παρείχε η σύνδεση των υποσυστημάτων της βάσης ή του STK500 απευθείας με τον υπολογιστή. Με χρήση του προγράμματος 232Analyzer ή ενός σειριακού τερματικού αντίστοιχα, ελέγχαμε στην πρώτη περίπτωση την αποστολή εντολών προς τη βάση, και τη λειτουργία αυτής, ενώ στη δεύτερη περίπτωση τα δεδομένα που έστελνε ο AVR μέσω UART, σύμφωνα με το τι είχαμε προγραμματίσει να κάνει.

Όσον αφορά στα ζητήματα που έπρεπε να αντιμετωπιστούν σε αυτή τη χρονική περίοδο του δεύτερου milestone, να αναφερθούν τα εξής: Υπήρχε αστοχία υλικού, καθώς ο πρώτος AC24V μετασχηματιστής που χρησιμοποιήθηκε, κάηκε. Επιπλέον, ύστερα από ενδελεχή έλεγχο προέκυψε ότι τα pins Τ2IN και Τ2OUT του MAX232 δεν λειτουργούν όπως πρέπει, και για τον λόγο αυτό χρησιμοποιήθηκε το δεύτερο διαθέσιμο ζευγάρι εισόδου/εξόδου, T1IN, T1OUT.

Σημαντικό πρόβλημα εντοπίστηκε στη σωστή αποστολή δεδομένων από τον AVR μέσω UART. Χρησιμοποιώντας τερματικό πρόγραμμα όπως προαναφέρθηκε, τα δεδομένα που είχε προγραμματιστεί να στέλνει ο μικροελεγκτής διέφεραν από αυτά που λαμβάνονταν στο τερματικό. Το πρόβλημα λύθηκε όταν μεταβλήθηκε λίγο η τιμή στον καταχωρητή UBRR, σε σχέση με την τιμή που θεωρητικά έπρεπε να τεθεί. Υποψία μας είναι ότι το ρολόι του επεξεργαστή δεν ήταν ακριβώς 8MHz, όπως είχε τεθεί, αλλά κάτι λίγο διαφορετικό, επηρεάζοντας έτσι και την τιμή που έπρεπε να καταχωρηθεί στον καταχωρητή UBRR, ώστε να επιτευχθεί το κατάλληλο baud rate αλλά και η επιτυχής επικοινωνία.

Μεγάλο πρόβλημα ήταν επίσης η έλλειψη documentation για τη βάση αλλά και η έλλειψη εμπειρίας στον τρόπο σύνδεσης και μετατροπής των επιπέδων τάσεων των διαφορετικών πρωτοκόλλων που χρησιμοποιήθηκαν. Στο σημείο αυτό οφείλουμε τις ευχαριστίες μας στον Αποστολάκη Στάυρο, απόφοιτο της σχολής ΗΜΜΥ του πολυτεχνείου Κρήτης, για την πολύτιμη καθοδήγησή του στον τομέα αυτό.

# Βιβλιογραφικές Αναφορές

## Manuals και Datasheets από το επίσημο site της Atmel:

* User Guide για το STK500
* Manual για τον ATmega16L

<http://www.atmel.com/Images/DOC0841.PDF>

## Διαδικτυακές πηγές για AVR:

* Σειρά tutorials για τη χρήση του AVR και τον προγραμματισμό του σε C

<https://www.youtube.com/watch?v=BKg2hoD89jY&list=PLRuRKN7_FVgvhdj6JmelCihl6jmKHc_Xt>

* Σειρά tutorials για τη χρήση του AVR και τον προγραμματισμό του σε C

<https://www.youtube.com/watch?v=9ADxPRjZI4Q&list=PLA6BB228B08B03EDD>

* Online Calculator για AVR timer

<http://eleccelerator.com/avr-timer-calculator/>

## Πηγές για την περιστροφική βάση AS20-RS485

* Κώδικες χειρισμού από εργαστήριο
* Manual με τις βασικές συνδέσεις από εργαστήριο
* Χρήσιμα links για το RS-485

<http://www.windmill.co.uk/rs485.html>

<https://en.wikipedia.org/wiki/RS-485>

* PELCO-P πρωτόκολλο

<https://www.commfront.com/pages/pelco-p-protocol-tutorial>

## Μοντελοποίηση σε MATLAB

* Teach Yourself Astronomy, David Evans

<https://www.amazon.co.uk/TEACH-YOURSELF-ASTRONOMY-DAVID-EVANS/dp/B005TSUXN2>

* Plotting a 3-D Dome as a Mesh Over a Globe

<https://www.mathworks.com/help/map/examples/plotting-a-3-d-dome-as-a-mesh-over-a-globe.html>

* Horizontal coordinate system

<https://en.wikipedia.org/wiki/Horizontal_coordinate_system>

* Matlab Documentation

<https://www.mathworks.com/help/matlab/>

* JPL HORIZONS System

<https://ssd.jpl.nasa.gov/?horizons>

* Συναρτήσεις για κάθετες γραμμές σε συγκεκριμένα σημεία του MATLAB plot

<https://www.mathworks.com/matlabcentral/fileexchange/1039-hline-and-vline>

* Ιστοσελίδα για τον προσδιορισμό συντεταγμένων οποιουδήποτε σημείου της γης

<https://www.latlong.net/>

## Επικοινωνία AVR-Βάσης

* MAXIM 232 Datasheet

<https://datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf>

* RS485 (εμφανίζονται στη βάση)
* RS232

<https://en.wikipedia.org/wiki/RS-232>

<https://www.lammertbies.nl/comm/info/RS-232_specs.html>

## Γενικές Πηγές

Μεταπτυχιακή εργασία Αποστολάκη Στάυρου με τίτλο «Σχεδίαση και Υλοποίηση Ενσωματωμένου Συστήματος Πραγματικού Χρόνου για Ιχνηλάτηση Κατευθυντικών Κεραιών από Κινούμενο Όχημα» στη σχολή ΗΜΜΥ του πολυτεχνείου Κρήτης.

# Παράρτημα

## Κώδικας που έχει φορτωθεί στον ΑΤmega16L για την παρουσίαση του δεύτερου milestone

## /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## University: Technical University of Crete

## School: School of Electrical & Computer Engineering

## Course: Embedded Systems - HRY 411

## Professor: A. Dollas

## Author: S. Apostolakis

## Modified by: P. Giakoumakis, N. Ghionis, P.Portokalakis

## Create Date: 24-11-2017

## Project Name: Automatic Detection of Celestial Bodies with Telescope

## Target Device: ATMEL AVR ATmega16L

## Gimbal: AS20-RS485

## Development Platform: AtmelStudio 7

## Version: 0.2

## Additional Comments: The purpose of this file is to show the basic functions of a gimbal. This version implements all the functions needed to tilt and pan the gimbal. Pan/tilt for specific ammount of degrees supported. A test program is shown at the main function, which initializes the gimbal to a specific position, and then makes a custom move.

## 

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## //Declare functions

## void stop(void);

## void USART\_Transmit(unsigned char data);

## void gimbal\_transmit(int byte1,int byte2,int byte3,int byte4,int byte5,int byte6,int byte7,int byte8);

## void down(void);

## void up(void);

## void left(void);

## void left\_deg(int deg);

## void right(void);

## void right\_deg(int deg);

## void down\_deg(int deg);

## void USART\_Init(int baud);

## void gimbal\_init(void);

## void USART\_Transmit(unsigned char data)

## {

## // Wait for empty transmit buffer

## while (!( UCSRA & (1<<UDRE)));

## // Put data into buffer, sends the data

## UDR = data;

## }

## void USART\_Init(int baud) {

## // Set baud register

## UBRRH=(baud>>8);

## UBRRL=baud;

## // Enable UART transmit

## UCSRB=(1<<TXEN)| (0<<UCSZ2);

## // Enable transmit: data 8 bytes, 1 stop bit and parity: none

## UCSRC=(1<<URSEL) |(0<<UMSEL) | (0<<UPM1) | (0<<UPM0) | (0<<USBS) | (1<<UCSZ0) | (1<<UCSZ1);

## }

## // Initialize gimbal on starting position (pan left, tilt down)

## void gimbal\_init(void) {

## left();

## \_delay\_ms(40000);

## down();

## \_delay\_ms(19000);

## }

## // Transmits each byte separately

## void gimbal\_transmit(int byte1,int byte2,int byte3,int byte4,int byte5,int byte6,int byte7,int byte8)

## {

## USART\_Transmit(byte1);

## USART\_Transmit(byte2);

## USART\_Transmit(byte3);

## USART\_Transmit(byte4);

## USART\_Transmit(byte5);

## USART\_Transmit(byte6);

## USART\_Transmit(byte7);

## USART\_Transmit(byte8);

## }

## // Tilt down

## void down(void)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x10,0x00,0x20,0xAF,0x3F);

## }

## // Tilt down on specific degrees

## void down\_deg(int deg)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x10,0x00,0x20,0xAF,0x3F);

## while(deg != 0)

## {

## \_delay\_us(190000); // Tilt down one degree

## deg--;

## }

## stop();

## }

## // Tilt up

## void up(void)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x08,0x00,0x20,0xAF,0x27);

## }

## // Tilt up on specific degrees

## void up\_deg(int deg)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x08,0x00,0x20,0xAF,0x27);

## 

## while(deg != 0)

## {

## \_delay\_us(190000); // Tilt up one degree

## deg--;

## }

## stop();

## }

## // Pan left

## void left(void)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x04,0x20,0x00,0xAF,0x2B);

## }

## // Pan left on specific degrees

## void left\_deg(int deg)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x04,0x20,0x00,0xAF,0x2B);

## 

## while(deg != 0)

## {

## \_delay\_us(114000); // Pan left one degree

## deg--;

## }

## stop();

## }

## // Pan right

## void right(void)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x02,0x20,0x00,0xAF,0x2D);

## }

## // Pan right on specific degrees

## void right\_deg(int deg)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x02,0x20,0x00,0xAF,0x2D);

## while(deg != 0)

## {

## \_delay\_us(114000); // Pan right one degree

## deg--;

## }

## stop();

## }

## // Stop movement

## void stop(void)

## {

## gimbal\_transmit(0xA0,0x00,0x00,0x00,0x00,0x00,0xAF,0x0F);

## }

## void test(void) {

## right\_deg(90);

## up\_deg(49);

## }

## int main(void)

## {

## USART\_Init(55); //set at 55 for a baud rate of 9600

## gimbal\_init();

## test();

## while (1)

## {

## }

## }

## Κώδικας που υλοποιήθηκε σε Matlab για τη μοντελοποίηση

### Πρόγραμμα σχεδιασμού μοντέλων

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

% University: Technical University of Crete

% School: School of Electrical & Computer Engineering

% Course: Embedded Systems - HRY 411

% Professor: A. Dollas

% Authors: P. Giakoumakis, N. Ghionis, P.Portokalakis

% Create Date: 29-11-2017

% Project Name: Automatic Detection of Celestial Bodies with % Telescope

% Target Device: ATMEL AVR ATmega16L

% Gimbal: AS20-RS485

% Development Platform: ATMEL STUDIO 7

% Version: 0.1

%

% Additional Comments: This MATLAB code is the accompanying file of the main project for better

% understanding of the coordinate system used to

% find and/or track a celestial body.

% For more details, see the reports

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% First Part

% Source: https://www.mathworks.com/help/map/examples/plotting-a-3-d-dome-as-a-mesh-over-a-globe.html

grs80 = referenceEllipsoid('grs80','km');

domeRadius = 10000; % km

domeLat = 35.52; % degrees

domeLon = 24.06; % degrees

domeAlt = 0; % km

[x,y,z] = sphere(30);

xEast = domeRadius \* x;

yNorth = domeRadius \* y;

zUp = domeRadius \* z;

zUp(zUp < 0) = 0;

% figure('Renderer','opengl')

% surf(xEast, yNorth, zUp,'FaceColor','yellow','FaceAlpha',0.5)

% axis equal

%

[xECEF, yECEF, zECEF] ...

= enu2ecef(xEast, yNorth, zUp, domeLat, domeLon, domeAlt, grs80);

% surf(xECEF, yECEF, zECEF,'FaceColor','yellow','FaceAlpha',0.5)

% axis equal

%figure('Renderer','opengl')

ax = axesm('globe','Geoid',grs80,'Grid','on', ...

'GLineWidth',1,'GLineStyle','-',...

'Gcolor',[0.9 0.9 0.1],'Galtitude',100);

ax.Position = [0 0 1 1];

axis equal off

view(3)

load topo

geoshow(topo,topolegend,'DisplayType','texturemap')

demcmap(topo)

land = shaperead('landareas','UseGeoCoords',true);

plotm([land.Lat],[land.Lon],'Color','black')

rivers = shaperead('worldrivers','UseGeoCoords',true);

plotm([rivers.Lat],[rivers.Lon],'Color','blue')

surf(xECEF, yECEF, zECEF,'FaceColor','yellow','FaceAlpha',0.5)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Second Part: Visualization of the data to be used

% from the avr to track/find a celestial body

%

% Data taken from the JPL HORIZONS on-line solar system data

% and ephimeris computation service.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Import data from a .txt file

data = readtable('data2.txt','Delimiter',' ');

%Organize data into corresponding arrays

azimuth = table2array(data(1:end,4));

elevation = table2array(data(1:end,5));

times = table2array(data(1:end,2));

%Select key timestamps to draw to the plot

middle = times(length(times)/2);

middle2 = middle{1};

azimuth\_1st = azimuth(1);

elevation\_1st = elevation(1);

azimuth\_between\_1\_and\_middle = azimuth(floor((1 + length(azimuth)/2)/2));

elevation\_between\_1\_and\_middle = elevation(floor((1 + length(azimuth)/2)/2));

time\_between\_1\_and\_middle = times(floor((1 + length(azimuth)/2)/2));

azimuth\_middle = azimuth(length(azimuth)/2);

elevation\_middle = elevation(length(elevation)/2);

azimuth\_between\_middle\_and\_end = azimuth(floor(length(azimuth)/2 + length(azimuth)/2/2));

elevation\_between\_middle\_and\_end = elevation(floor(length(azimuth)/2 + length(azimuth)/2/2));

time\_between\_middle\_and\_end = times(floor(length(azimuth)/2 + length(azimuth)/2/2));

azimuth\_last = azimuth(end)

%Constructing the plot

figure;

stem(azimuth,elevation,':');

text(azimuth(length(azimuth)/2),elevation(length(elevation)/2)+20,times(length(times)/2))

text(azimuth\_1st,elevation\_1st+20,times(1))

text(azimuth\_between\_1\_and\_middle,elevation\_between\_1\_and\_middle,time\_between\_1\_and\_middle);

text(azimuth\_between\_middle\_and\_end,elevation\_between\_middle\_and\_end,time\_between\_middle\_and\_end);

azim\_elev\_text = sprintf('Horizontal Coordinate System \n Observation Point - Chania');

title(azim\_elev\_text);

xlabel('Azimuth (\circ)')

ylabel('Elevation (\circ)')

grid on;

axis([0 370 -100 100])

hold on;

%Plot vertical lines to clearly see selected hours of the day

vline(azimuth\_1st)

vline(azimuth\_middle)

vline(azimuth\_last)

vline(azimuth\_between\_1\_and\_middle)

vline(azimuth\_between\_middle\_and\_end)

%figure;

%polar(azimuth,elevation);

### Συνάρτηση σχεδιασμού οριζόντιας γραμμής

function hhh=hline(y,in1,in2)

% function h=hline(y, linetype, label)

%

% Draws a horizontal line on the current axes at the location specified by 'y'. Optional arguments are

% 'linetype' (default is 'r:') and 'label', which applies a text label to the graph near the line. The

% label appears in the same color as the line.

%

% The line is held on the current axes, and after plotting the line, the function returns the axes to

% its prior hold state.

%

% The HandleVisibility property of the line object is set to "off", so not only does it not appear on

% legends, but it is not findable by using findobj. Specifying an output argument causes the function to

% return a handle to the line, so it can be manipulated or deleted. Also, the HandleVisibility can be

% overridden by setting the root's ShowHiddenHandles property to on.

%

% h = hline(42,'g','The Answer')

%

% returns a handle to a green horizontal line on the current axes at y=42, and creates a text object on

% the current axes, close to the line, which reads "The Answer".

%

% hline also supports vector inputs to draw multiple lines at once. For example,

%

% hline([4 8 12],{'g','r','b'},{'l1','lab2','LABELC'})

%

% draws three lines with the appropriate labels and colors.

%

% By Brandon Kuczenski for Kensington Labs.

% brandon\_kuczenski@kensingtonlabs.com

% 8 November 2001

if length(y)>1 % vector input

for I=1:length(y)

switch nargin

case 1

linetype='r:';

label='';

case 2

if ~iscell(in1)

in1={in1};

end

if I>length(in1)

linetype=in1{end};

else

linetype=in1{I};

end

label='';

case 3

if ~iscell(in1)

in1={in1};

end

if ~iscell(in2)

in2={in2};

end

if I>length(in1)

linetype=in1{end};

else

linetype=in1{I};

end

if I>length(in2)

label=in2{end};

else

label=in2{I};

end

end

h(I)=hline(y(I),linetype,label);

end

else

switch nargin

case 1

linetype='r:';

label='';

case 2

linetype=in1;

label='';

case 3

linetype=in1;

label=in2;

end

g=ishold(gca);

hold on

x=get(gca,'xlim');

h=plot(x,[y y],linetype);

if ~isempty(label)

yy=get(gca,'ylim');

yrange=yy(2)-yy(1);

yunit=(y-yy(1))/yrange;

if yunit<0.2

text(x(1)+0.02\*(x(2)-x(1)),y+0.02\*yrange,label,'color',get(h,'color'))

else

text(x(1)+0.02\*(x(2)-x(1)),y-0.02\*yrange,label,'color',get(h,'color'))

end

end

if g==0

hold off

end

set(h,'tag','hline','handlevisibility','off') % this last part is so that it doesn't show up on legends

end % else

if nargout

hhh=h;

end

### Συνάρτηση σχεδιασμού κάθετης γραμμής

function hhh=vline(x,in1,in2)

% function h=vline(x, linetype, label)

%

% Draws a vertical line on the current axes at the location specified by 'x'. Optional arguments are

% 'linetype' (default is 'r:') and 'label', which applies a text label to the graph near the line. The

% label appears in the same color as the line.

%

% The line is held on the current axes, and after plotting the line, the function returns the axes to

% its prior hold state.

%

% The HandleVisibility property of the line object is set to "off", so not only does it not appear on

% legends, but it is not findable by using findobj. Specifying an output argument causes the function to

% return a handle to the line, so it can be manipulated or deleted. Also, the HandleVisibility can be

% overridden by setting the root's ShowHiddenHandles property to on.

%

% h = vline(42,'g','The Answer')

%

% returns a handle to a green vertical line on the current axes at x=42, and creates a text object on

% the current axes, close to the line, which reads "The Answer".

%

% vline also supports vector inputs to draw multiple lines at once. For example,

%

% vline([4 8 12],{'g','r','b'},{'l1','lab2','LABELC'})

%

% draws three lines with the appropriate labels and colors.

%

% By Brandon Kuczenski for Kensington Labs.

% brandon\_kuczenski@kensingtonlabs.com

% 8 November 2001

if length(x)>1 % vector input

for I=1:length(x)

switch nargin

case 1

linetype='r:';

label='';

case 2

if ~iscell(in1)

in1={in1};

end

if I>length(in1)

linetype=in1{end};

else

linetype=in1{I};

end

label='';

case 3

if ~iscell(in1)

in1={in1};

end

if ~iscell(in2)

in2={in2};

end

if I>length(in1)

linetype=in1{end};

else

linetype=in1{I};

end

if I>length(in2)

label=in2{end};

else

label=in2{I};

end

end

h(I)=vline(x(I),linetype,label);

end

else

switch nargin

case 1

linetype='r:';

label='';

case 2

linetype=in1;

label='';

case 3

linetype=in1;

label=in2;

end

g=ishold(gca);

hold on

y=get(gca,'ylim');

h=plot([x x],y,linetype);

if length(label)

xx=get(gca,'xlim');

xrange=xx(2)-xx(1);

xunit=(x-xx(1))/xrange;

if xunit<0.8

text(x+0.01\*xrange,y(1)+0.1\*(y(2)-y(1)),label,'color',get(h,'color'))

else

text(x-.05\*xrange,y(1)+0.1\*(y(2)-y(1)),label,'color',get(h,'color'))

end

end

if g==0

hold off

end

set(h,'tag','vline','handlevisibility','off')

end % else

if nargout

hhh=h;

end