Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολόγων Εφαρμογών

Τμήμα Μηχανολογίας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και Ανάπτυξη οχήματος τύπου Segway
Design and Development of Segway Vehicle

Σπουδαστές : **Μπαλής Αγαπητός** A.M. tm 6071

Πατεράκης Μιχάλης A.M. tm 6004

Επιβλέπων καθηγητής: Δρ. **Καβουσανός Εμμανουήλ**

Ηράκλειο Κρήτης, Μάρτιος 2018

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και ανάπτυξη οχήματος τύπου Segway

Μπαλής Αγαπητός – Πατεράκης Μιχάλης

Επιβλέπων: Δρ. Καβουσσανός Εμμανουήλ

Ευχαριστίες

Η εργασία καθώς και η υλοποίηση της κατασκευής του οχήματος τύπου **SEGWAY** δεν ήταν μια απλή τυπική διαδικασία, αλλά μια πραγματική προσπάθεια για να θυμόμαστε όλους αυτούς τους ανθρώπους, χωρίς τη συνεργασία των οποίων δεν θα ήταν σε θέση να ολοκληρωθεί το έργο μας.
Θέλουμε να ευχαριστήσουμε το Τμήμα Μηχανολογίας που μας έδωσε μια τέτοια ευκαιρία για να αρχίσει αυτό το έργο σε πρώτη φάση. Εκφράζουμε την ειλικρινή ευγνωμοσύνη μας στον Δρ. Καβουσσανό Εμμανουήλ, που μας βοήθησε να μετατρέψουμε αυτή την ευκαιρία σε πραγματικό αποτέλεσμα. Χωρίς τη δυνατότητα καθοδήγηση και τις συμβουλές του θα ήταν αδύνατο για εμάς να ολοκληρωθεί αυτό το έργο.

Τέλος, θα θέλαμε να εκφράσουμε την ειλικρινή εκτίμησή μας προς τους γονείς μας για την υπομονή και την ενθάρρυνση τους κατά τη διάρκεια αυτής της εργασίας.

Περίληψη

Η παρούσα εργασία περιγράφει την ανάπτυξη και κατασκευή ενός δικύκλου αυτόματου οχήματος εξισορρόπησης, τύπου SEGWAY, που διατηρεί τη δική του ισορροπία και αυτή του επιβάτη. Είναι εξοπλισμένο με έναν σταθερό άξονα και τιμόνι σχήματος Τ τοποθετημένα σε μια πλατφόρμα η οποία συγκρατεί επίσης δύο παράλληλους τροχούς.
Η ρύθμιση της ταχύτητας της περιστροφής των τροχών πραγματοποιείται μέσω ενός συστήματος κλειστού βρόχου με αντιστάθμιση PID.
Το όχημα είναι εξοπλισμένο με διάφορους αισθητήρες, γυροσκόπιο, επιταχυνσιόμετρο, διακόπτες, ποτενσιόμετρα κτλ. , ώστε να αντιλαμβάνεται τους χειρισμούς του οδηγού.
Ένας μικροελεγκτής ARDUINO είναι υπεύθυνος για την επεξεργασία όλων των εξωτερικών πληροφοριών, έτσι ώστε ελέγχοντας κατάλληλα την κατεύθυνση και την ταχύτητα κάθε ενός από τους δύο κινητήρες ξεχωριστά, να κρατήσει το όχημα αλλά και τον οδηγό σε όρθια θέση.
Η κατεύθυνση καθώς και η ταχύτητα του οχήματος εξαρτώνται αποκλειστικά από την κλίση του σώματος του οδηγού μπρός πίσω, αλλά και του τιμονιού δεξιά ή αριστερά.
Τα οχήματα Segway μετακινούνται, στέκονται και χειρίζονται σύμφωνα με τη δυναμική (κλίση) του ανθρώπινου σώματος.

Abstract

The present project presents and describes the development and construction of a SEGWAY type vehicle, which maintains its own balance and that of the passenger. It is equipped with a fixed shaft and T-shaped steering wheel mounted on a platform that also holds two parallel wheels. Wheel speed adjustment is performed via a PID closed loop.
The vehicle is equipped with various sensors, gyroscope, accelerometer, switches, potentiometers etc. in order to understand the driver's handling.
An ARDUINO microcontroller is responsible for processing all external information so that by properly checking the direction and speed of each of the two motors separately, keep the vehicle and the driver in an upright position.
The direction and speed of the vehicle depend exclusively on the inclination of the driver's body towards the rear and the steering wheel to the left or right.
Segway vehicles move, stand and operate according to the dynamic (gradient) of the human body.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1

Εισαγωγή

Συστήματα αυτό-εξισορρόπησης μπορούμε να συναντήσουμε σε πολλά μέρη και είναι απαραίτητη για την ομαλή λειτουργία πολλών τύπων μηχανών. Μερικά από αυτά είναι τα Segways, τα δίποδα ρομπότ και οι διαστημικοί πύραυλοι (πολλοί διαστημικοί πύραυλοι έχουν χαθεί εξαιτίας ενός ελαττωματικού συστήματος εξισορρόπησης).

Αυτό που πολλοί άνθρωποι δεν συνειδητοποιούν είναι ότι τα αυτό-εξισορρόπησης χρησιμοποιούν τους ίδιους ελεγκτές που μπορεί να δει κανείς σε σέρβο-κινητήρες, πυραύλους, μονάδες κλιματισμού, ακόμα και θερμοστάτες. Φυσικά οι πύραυλοι χρησιμοποιούν σημαντικά πιο περίπλοκους ελεγκτές από ό, τι τα κλιματιστικά, αλλά η βασική αρχή παραμένει η ίδια: πώς να ρυθμίσουμε το σύστημα έτσι ώστε να λειτουργεί όσο το δυνατόν πιο κοντά στο επιθυμητό αποτέλεσμα.

Αυτός είναι ο λόγος για τον οποίο η κατασκευή ενός οχήματος αυτό-ισορρόπησης είναι ενδιαφέρον , δεδομένου ότι περιέχει πολλά σχετικά με την τεχνολογία που αφορά τις φιλικές προς το περιβάλλον και ενεργειακά αποδοτικές μεταφορές.

Αυτή η εργασία περιγράφει την ανάπτυξη ενός παρόμοιου οχήματος από το μηδέν, λαμβάνοντας υπόψη σε κάθε φάση τη βιβλιογραφία που υπάρχει για τον προγραμματισμό, το σχεδιασμό, την κατασκευή και την επαλήθευση του οχήματος.

Σκοπός εργασίας

Ο κύριος σκοπός της εργασίας είναι η μελέτη, ο σχεδιασμός και η κατασκευή ενός πλήρως λειτουργικού δίτροχου αυτό-εξισορροπούμενου οχήματος τύπου SEGWAY, το οποίο μπορεί να χρησιμοποιηθεί ως μέσο μεταφοράς για ένα μόνο άτομο. Θα πρέπει να καθοδηγείται εμπρός και πίσω με φυσικές κινήσεις, με αντίστοιχη κλίση του οχήματος προς τα εμπρός και προς τα πίσω. Η στροφή του οχήματος πρέπει να επιτευχθεί με την κλίση του τιμονιού προς τα πλάγια. Η πηγή ενέργειας για τη λειτουργία του θα να είναι μια μπαταρία. Ένας από τους στόχους είναι η εύκολη επαναφόρτιση αυτής της μπαταρίας.

Ο κύριος στόχος είναι να κατασκευαστεί ένα όχημα ικανό να μεταφέρει ένα άτομο με βάρος έως 100 kg για 30 λεπτά ή μια απόσταση 5 χλμ, ή όποιο από τα δύο συμβεί πρώτο.

Πιο συγκεκριμένα, οι στόχοι της εργασίας είναι οι εξής:

- Η ταχύτητα του οχήματος πρέπει να ελέγχεται από τον αναβάτη με κλίση του σώματος προς τα εμπρός και προς τα πίσω.
- Η στροφή του οχήματος πρέπει να ελέγχεται από την κλίση του τιμονιού.
- Να επιτρέπονται αλλαγές των παραμέτρων ελέγχου PID χωρίς επαναπρογραμματισμό.
- Να είναι επαναφορτιζόμενο από μια τυπική πρίζα 220V 50Hz.

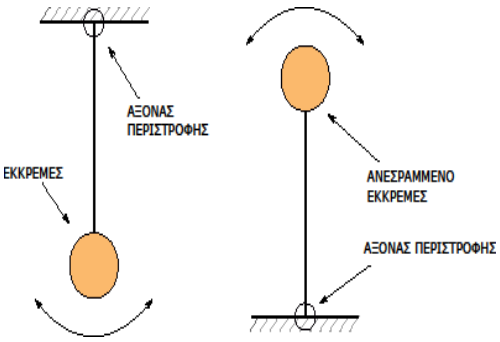
Γενικοί στόχοι:

- Ο ελεγκτής θα πρέπει να είναι ένας γραμμικός ελεγκτής.
- Ο ελεγκτής θα πρέπει να εφαρμοστεί στο λειτουργικό σύστημα σε πραγματικό χρόνο. Η απόκριση του ελεγκτή πρέπει να είναι αρκετά καλή ώστε να παρέχει την ασφαλή λειτουργία του οχήματος για τον αναβάτη.

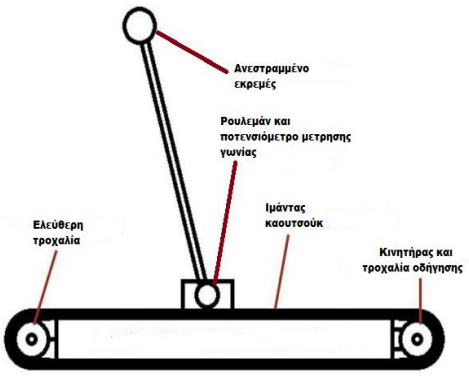
Περιγραφή εργασίας

Το αντικείμενο της παρούσας εργασίας είναι η μελέτη και κατασκευή ηλεκτρονικού οχήματος με δύο τροχούς, αυτόματης εξισορρόπησης, με μπαταρία που διατηρεί τη δική του ισορροπία και αυτή του επιβάτη. Είναι εξοπλισμένο με έναν σταθερό άξονα ελέγχου σχήματος **T** τοποθετημένο σε μια πλατφόρμα η οποία συγκρατεί επίσης δύο παράλληλους τροχούς. Τα οχήματα αυτόματης εξισορρόπησης (Segways) κινούνται, στέκονται και χειρίζονται σύμφωνα με τη δυναμική του ανθρώπινου σώματος. Μια κλίση του ανθρώπινου σώματος προς τα εμπρός προκαλεί μια κίνηση του οχήματος προς τα εμπρός, μια στάση σε κάθετη θέση ένα σταμάτημα και μια κλίση προς τα πίσω μια κίνηση πίσω. Η συσκευή δεν έχει φρένα ή επιταχυντή, αλλά έχει χειρολαβή για την πραγματοποίηση στροφών. Είναι το μόνο όχημα που μπορεί να γυρίσει στη θέση του, ακριβώς όπως ένα άτομο, επειδή οι τροχοί του έχουν τη δυνατότητα να γυρίσουν σε αντίθετες κατευθύνσεις. Για τα ρομπότ αυτόματης εξισορρόπησης με δύο τροχούς, η σταθερότητα είναι ζωτικής σημασίας, καθώς δεν μπορούν να παραμείνουν όρθια (ισορροπημένα) χωρίς προσπάθεια. Ο σχεδιαστικές τους έννοιες προέρχονται κατά κύριο λόγο από την κλασσική ρομποτική έρευνα που ονομάζεται ανεστραμμένο εκκρεμές. Ένα ανεστραμμένο εκκρεμές, όπως υποδηλώνει και το όνομά του, είναι ένα εκκρεμές που έχει τη μάζα του πάνω από το σημείο περιστροφής του και όχι κάτω από τα παραδοσιακά εκκρεμή. Ένα ρομπότ αυτόματης εξισορρόπησης, όπως ένα Segway, είναι μια εκτεταμένη έκδοση ενός ανεστραμμένου εκκρεμούς.

Ένα από τα πιο σημαντικά καθήκοντα αυτόνομων συστημάτων οποιουδήποτε είδους είναι να μπορούν να επικοινωνήσουν με το περιβάλλον τους. Αυτό μπορεί να γίνει με τη χρήση αισθητήρων (Ronald Siegwart et.al 2004). Η χρήση αισθητήρων στη ρομποτική παρέχει στα ρομπότ εξωτερικές φυσικές πληροφορίες που βοηθούν τα συστήματα αυτά να ανταποκρίνονται ανάλογα. Το όχημά μας χρησιμοποιεί δύο τύπους αισθητήρων, ένα επιταχυνσιόμετρο, είναι μια ηλεκτρομηχανική συσκευή που μετρά δυνάμεις αδρανείας επιτάχυνσης, και ένα γυροσκόπιο, είναι μια συσκευή που χρησιμοποιείται για τη μέτρηση της ταχύτητας περιστροφής (γωνιακή ταχύτητα) ενός σώματος. Χρησιμοποιώντας μικροελεγκτή, το Segway αναλύει συνεχώς την απόδοσή της. Συνδυασμός γυροσκοπίων και άλλων αισθητήρων καθορίζουν τη θέση του σε σχέση με το κέντρο βάρους του. Οι επεξεργαστές αναλύουν τις μετρήσεις τους και αντισταθμίζουν σε πραγματικό χρόνο τις επιφανειακές ανωμαλίες προκειμένου να ελέγχουν την κίνηση της συσκευής και να διασφαλίζουν τη σταθερότητα του αναβάτη.



Κανονικό και ανεστραμμένο εκκρεμές



Πειραματική προσέγγιση ανεστραμμένο εκκρ

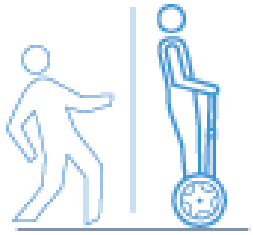
Κεφάλαιο 2

2.1 Οχήματα τύπου SEGWAY – Ρομπότ αυτό-ισορρόπησης.

2.1.1 Αρχή λειτουργίας των οχημάτων αυτό-ισορρόπησης

Το πιο συναρπαστικό χαρακτηριστικό του Segway και το κλειδί για το χειρισμό του είναι η ικανότητά του να διατηρεί την ισορροπία του. Για να κατανοήσουμε πώς λειτουργεί το σύστημα, είναι χρήσιμο να δούμε το μοντέλο του Kamen – για το ανθρώπινο σώμα και το όχημα.

Κίνηση εμπρός



Κίνηση πίσω

Εάν στεκόμαστε και γέρνουμε προς τα εμπρός ώστε να υπερβάλλουμε της ισορροπίας, δεν θα πέσουμε. Ο εγκέφαλος γνωρίζει ότι δεν βρίσκεστε σε ισορροπία επειδή το υγρό στο εσωτερικό αυτιού μας έχει μετακινηθεί. Αυτό αναγκάζει τα πόδια να προχωρήσουν προς τα εμπρός για να αποφευχθεί μια πτώση. Όσο σκύβουμε προς τα εμπρός, ο εγκέφαλος μας θα ξεκινήσει την κίνηση του ποδιού προς τα εμπρός. Αντί να πέσουμε, προχωράμε ένα βήμα.

Το Segway κάνει σχεδόν το ίδιο, με την διαφορά ότι έχει τροχούς αντί για πόδια, κινητήρα αντί για μυς, μικροεπεξεργαστή αντί για εγκέφαλο και εξελιγμένους αισθητήρες εξισορρόπησης αντί για αίσθηση ισορροπίας στο εσωτερικό του αυτιού. Όπως και ο εγκέφαλος, το Segway ξέρει πότε ο οδηγός κλίνει προς τα εμπρός ή προς τα πίσω. Για να διατηρηθεί η ισορροπία, το όχημα κινεί τους τροχούς ακριβώς με τη σωστή ταχύτητα. Αυτή η συμπεριφορά που επιτρέπει στο Segway να διατηρεί την ισορροπία μόνο σε δύο τροχούς ονομάζεται "*δυναμική σταθεροποίηση*" και έχει κατοχυρωθεί με δίπλωμα ευρεσιτεχνίας.
Το Segway αποτελείται από ένα ευφυές δίκτυο αισθητήρων, μηχανικών εξαρτημάτων, συστήματος μετάδοσης κίνησης και συστήματος διεύθυνσης. Από τη στιγμή που ο οδηγός στέκεται στο όχημα, γυροσκοπικοί αισθητήρες και αισθητήρες επιτάχυνσης αρχίζουν να αναλύουν το έδαφος και τη θέση του σώματος εκατό φορές ανά δευτερόλεπτο.

Τα αισθητήρια παρακολουθούν σε πραγματικό χρόνο την κατάσταση ανύψωσης και τον ρυθμό αλλαγής κατάστασης του αμαξώματος τα οποία αποστέλλει σε έναν υψηλής ταχύτητας μικροελεγκτή που υπολογίζει κατάλληλα τα δεδομένα αυτά. Οι κινητήρες οδήγησης παράγουν επιτάχυνση προς τα εμπρός ή προς τα πίσω για να επιτευχθεί η ισορροπία.
Ας υποθέσουμε ότι το συνολικό κέντρο βάρους της στάσης του οδηγού είναι κάθετο στο όχημα, και τον άξονα του οχήματος λαμβάνεται ως γραμμή αναφοράς. Όταν ο άξονας αυτός είναι κεκλιμένος προς τα εμπρός, οι αισθητήρες διαπιστώνουν ότι το σημείο βαρύτητας του οδηγού κλίνει εμπρός, στέλνουν τις κατάλληλες πληροφορίες στον μικροελεγκτή και αυτός με την σειρά του αφού τις επεξεργαστεί ρυθμίζει ταχύτητα και κατεύθυνση στους κινητήρες οι οποίοι παράγουν δύναμη τέτοια που να κινήσουν το όχημα προς τα εμπρός ώστε να επιτευχθεί το αποτέλεσμα της ισορροπίας.

Επομένως, όσο ο οδηγός αλλάζει τη γωνία του σώματός του με κλίση προς τα εμπρός ή προς τα πίσω, το όχημα θα κινείται προς τα εμπρός ή προς τα πίσω σύμφωνα με την κλίση και η ταχύτητα του θα είναι ανάλογη με τον βαθμό κλίσης του σώματος του οδηγού.

2.1.2 Ρομπότ αυτό-ισορρόπησης

Το πρόβλημα του ανεστραμμένου εκκρεμούς δεν είναι ασυνήθιστο στον τομέα της μηχανικής ελέγχου. Η μοναδικότητα και η ευρεία εφαρμογή της τεχνολογίας που προέρχεται από αυτό το ασταθές σύστημα έχει προκαλέσει ενδιαφέρον για πολλές έρευνες σε όλο τον κόσμο. Τα τελευταία χρόνια, οι ερευνητές έχουν εφαρμόσει την ιδέα ενός κινητού μοντέλου ανεστραμμένου εκκρεμούς σε διάφορα προβλήματα όπως το σχεδιασμό ποδιών για ανθρωποειδή ρομπότ, ρομποτικές αναπηρικές καρέκλες και συστήματα προσωπικής μεταφοράς.

Ερευνητές στο Εργαστήριο Βιομηχανικών Ηλεκτρονικών στο Ελβετικό Ομοσπονδιακό Ινστιτούτο Τεχνολογία έχουν δημιουργήσει ένα πρωτότυπο, ελεγχόμενο από επεξεργαστή ψηφιακού σήματος, δίτροχο όχημα σε σμίκρυνση στηριζόμενο στο ανεστραμμένο εκκρεμές. Το πρωτότυπο διαθέτει βάρη που συνδέονται με το σύστημα για την προσομοίωση ενός ανθρώπινου οδηγού. Ένας γραμμικός ελεγκτής, αισθητήρια και πληροφορίες από γυροσκόπιο και κωδικοποιητές κινητήρα χρησιμοποιούνται για τη σταθεροποίηση αυτού του συστήματος (Grasser κ.ά., 2002).

Πρωτότυπο δίτροχο όχημα αυτό-ισορρόπησης στηριζόμενο στο ανεστραμμένο εκκρεμές.

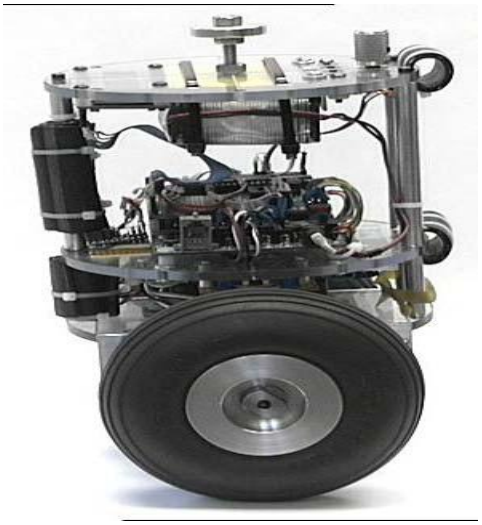
Ένα παρόμοιο και εμπορικά διαθέσιμο σύστημα, ‘’SEGWAY HT’’ έχει εφευρεθεί από τον *Dean Kamen*, ο οποίος κατέχει περισσότερες από 150 αμερικανικές και ξένες ευρεσιτεχνίες που σχετίζονται με ιατρικές συσκευές, συστήματα ελέγχου κλίματος και σχεδιασμός ελικοπτέρων.

Το Segway HT ξεκίνησε τη ζωή του όταν ο επιχειρηματίας και μηχανικός, *Dean Kamen*, γλίστρησε κατά την έξοδο του από τη ντουζιέρα, και το σώμα του προσπαθώντας να εξουδετερώσει την ολίσθηση, πετάχτηκε προς τα πίσω. Παρόλο που συντρίφθηκε στο πάτωμα, άρχισε να σκέφτεται αργότερα ότι εάν το ανθρώπινο σώμα μπορεί να ανταποκριθεί τόσο γρήγορα σε μια τέτοια κατάσταση, θα μπορούσε και μια μηχανή να είναι ικανή να το κάνει. Το SEGWAY HT είναι σε θέση να ισορροπήσει και να μεταφέρει τον χειριστή (άνθρωπο) ο οποίος στέκεται στην πλατφόρμα του.

Αυτή η καινοτομία χρησιμοποιεί πέντε γυροσκόπια και μια συλλογή άλλων αισθητήρων κλίσεως για να κρατηθεί σε όρθια θέση. Μόνο τρία γυροσκόπια χρειάζονται για όλο το σύστημα, είναι οι πρόσθετοι αισθητήρες περιλαμβάνονται για λόγους ασφαλείας.

Όχημα SEGWAY

Η μοναδικότητα αυτών των συστημάτων έχει προκαλέσει ενδιαφέρον από τους λάτρεις ρομπότ. Για παράδειγμα, το Nbot είναι ένα ρομπότ ισορρόπησης δύο τροχών παρόμοιο με το JOE που κατασκευάστηκε από David .P Anderson. Αυτό το ρομπότ χρησιμοποιεί ένα εμπορικά διαθέσιμο αδρανειακό αισθητήρα και πληροφορίες θέσης από έναν κωδικοποιητή κινητήρα για την εξισορρόπηση του συστήματος.
Ο Steven Hassenplug έχει δημιούργησε με επιτυχία ένα ρομπότ ισορρόπησης που ονομάζεται Legway χρησιμοποιώντας το LEGO Mindstorms robotics kit. Δύο ηλεκτρο-οπτικοί αισθητήρες προσέγγισης (EOPD) χρησιμοποιούνται για την παροχή πληροφοριών γωνίας κλίσης του ρομπότ στον ελεγκτή ο ποίος είναι προγραμματισμένος σε μια γλώσσα προγραμματισμού τύπου C / C ++ BrickOS, ειδικά για το LEGO Mindstorms.



Nbot

Legway

Η ταχεία αύξηση του ηλικιωμένου πληθυσμού σε χώρες όπως η Ιαπωνία προκάλεσε τους ερευνητές να αναπτύξουν ρομποτικές αναπηρικές καρέκλες για να βοηθήσουν τους ασθενείς να μετακινηθούν, (Takahashi κ.ά., 2000). Το σύστημα ελέγχου για ένα ανεστραμμένο εκκρεμές εφαρμόζεται όταν η αναπηρική καρέκλα χειρίζεται μικρό-ελιγμούς ή εμπόδια στο οδοστρώματα.

Δεν είναι ευρέως γνωστό ότι το πρώτο εγχείρημα του Kamen σε οχήματα αυτό-ισορρόπησης οδήγησε στο iBot, μια μηχανοκίνητη αναπηρική καρέκλα που είναι σε θέση να ανεβαίνει σκάλες περιστρέφοντας δύο σειρές ομοαξονικών τροχών και να χρησιμοποιεί μόνο μία σειρά τροχών κατά την αυτό-ισορρόπηση του.

Δεν είναι ένα Segway αλλά ένας μεγαλύτερος αδερφός του Segway HT είναι το DEKA iBOT (σειρά iBOT Mobility) (λογισμικό / OS 10-14.1) Ανακοινώθηκε 6/99, Κυκλοφόρησε 12/03.

Κατά τη διάρκεια της ανάπτυξης του iBot, ο Kamen είδε τη δυνατότητα χρήσης της αρχής της αυτό-ισορρόπησης για ένα καθημερινό όχημα μεταφοράς. Το Segway HT αναπτύχθηκε από την εταιρεία έρευνας και ανάπτυξης Kamen's, DEKA. Το 2001, το πρώτο Segway παρουσιάστηκε στην δημόσια τηλεόραση και τέθηκε σε διάθεσή για αγορά το 2002.

Σε υψηλότερο επίπεδο, η Sugihara et al. (2002) μοντελοποίησε την κίνηση του ανθρώπου σαν ένα ανεστραμμένο εκκρεμές κατά το σχεδιασμό σε πραγματικό χρόνο ενός ανθρωποειδούς ρομπότ που ελέγχει το κέντρο βάρους του με έμμεσο χειρισμό του Zero Moment Point (ZMP). Η απόκριση σε πραγματικό χρόνο της μεθόδου παρέχει ένα ανθρωποειδείς ρομπότ με υψηλή κινητικότητα.

*Το μοντέλο ανθρωποειδούς ρομπότ **H5***

Στις αρχές της τρέχουσας χιλιετίας ο κόσμος περίμενε αγωνία το "Ginger", μια νέα εφεύρεση από τον κύριο εφευρέτη Dean Kamen. Ο Kamen υποσχέθηκε ότι το "Ginger " (το κωδικό όνομα του) θα έκανε επανάσταση στον κόσμο. Τον Δεκέμβριο του 2001 αποκαλύφθηκε τελικά η νέα του εφεύρεση. Το "Ginger" αποδείχθηκε ότι ήταν το Segway HT (Human Transport - Ανθρώπινες Μεταφορές). Το όνομα Segway επελέγη ως παιχνίδι για τη λέξη "segue" - ομαλή μετάβαση - δεδομένου ότι το Segway HT έπρεπε να ‘’μεταφέρει’’ την ανθρωπότητα στην επόμενη εποχή της ανθρώπινης μεταφοράς. Προφανώς ο Kamen πίστευε ότι το περπάτημα είναι ένα ξεπερασμένο μέσο μεταφοράς. Ήθελε να το αντικαταστήσει με το νέο του θαύμα της σύγχρονης τεχνολογίας, το Segway HT. Ωστόσο, η υπόλοιπη ανθρωπότητα δεν φάνηκε να μοιράζεται τη γνώμη του, καθώς λιγότερα από 24.000 Segways είχαν πωληθεί μέχρι τον Αύγουστο του 2006.

Καθώς το Segway δεν μπορεί ποτέ να αντικαταστήσει το περπάτημα και την ποδηλασία ως το προτιμώμενο μέσο, δεν υπάρχει αμφιβολία ότι η οδήγηση ενός Segway είναι μια εξαιρετικά διασκεδαστική εμπειρία. Επιπλέον, η οδήγηση ενός Segway είναι πολύ εύκολη για τον χειριστή. Χρειάζεται λιγότερο από μισή ώρα για άνετη οδήγηση ενός Segway, και η άριστη ικανότητα μπορεί να επιτευχθεί μόνο σε λίγες ώρες. Μια μελέτη που έγινε πριν από μερικά χρόνια κατέληξε στο συμπέρασμα ότι το 93% όλων των ανθρώπων που είδαν ένα Segway θέλουν να το δοκιμάσουν, ενώ μόνο το 1% θα θέλανε να το αποκτήσουν. Με κόστος άνω των 6000 δολαρίων ανά μονάδα, δεν είναι περίεργο!

2.1.3 Το Segway i2 (Μια δραστική βελτίωση της τεχνολογίας)

Μεταξύ του Δεκεμβρίου 2001, και του Αύγουστου του 2006, όλες οι βελτιώσεις στο Segway ήταν αρκετά μικρές. Οι αλλαγές μπορεί να ήταν διακοσμητικές ή έγιναν μικρές βελτιώσεις στο λογισμικό, αλλά βασικά η λειτουργικότητα ήταν η ίδια για τα πρώτα πέντε χρόνια. Τον Αύγουστο του 2006 κυκλοφόρησε το νέο Segway i2, με πλήρη τεχνολογία LeanSteer ™. Επικρατούσε η εντύπωση ότι το Segway PT (Personal Transporter) διευθυνόταν με κλίση, αλλά αυτό δεν ισχύε στην πραγματικότητα. Το νέο Segway i2 PT

είναι η πρώτη νέα έκδοση που επέτρεψε στον αναβάτη να γυρίσει προς οποιαδήποτε κατεύθυνση, απλώς γέρνοντας το σώμα του.

Όλα τα παλαιότερα μοντέλα προχωρούσαν εμπρός με κλίση προς τα εμπρός και προς τα πίσω κάνοντας κλίση προς τα πίσω. Για να στρίψουν αριστερά ή δεξιά, ο αναβάτης υποχρεωνόταν να γυρίσει με το αριστερό χέρι ένα χειριστήριο τύπου γκαζιού. Το Segway έστριβε προς τα αριστερά αν ο αναβάτης γύρισε (στραγγαλίσει) τον έλεγχο μακριά από αυτόν, ή έστριβε προς τα δεξιά όταν ο μοχλός γύρισε στραμμένος προς τον αναβάτη. Ο βαθμός στον οποίο γινόταν ο έλεγχος αυτός, ο οποίος βρίσκεται στην αριστερή λαβή του τιμονιού, καθορίζει την ένταση της στροφής.

Η τελειοποιημένη τεχνολογία LeanSteer των νέων μοντέλων Segway έχει κάνει πολύ πιο εύκολο τον έλεγχο. Τώρα το Segway γυρίζει προς οποιαδήποτε κατεύθυνση ο μοτοσικλετιστής κλίνει, όχι μόνο προς τα εμπρός και προς τα πίσω. Για να στρίψει αριστερά ή δεξιά στο νέο Segway i2, ο αναβάτης απλά κλίνει το σώμα του αριστερά ή δεξιά, και η ένταση της στροφής ελέγχεται από το μέγεθος της κλίσης.

Το Segway είναι πραγματικά ένα θαύμα της σύγχρονης τεχνολογίας. Είναι ο πρώτος παγκοσμίως αυτο-ισορροπημένος μεταφορέας ανθρώπων. Χρησιμοποιεί μια διαδικασία που ονομάζεται δυναμική σταθεροποίηση για να σας κρατά σε όρθια θέση ανά πάσα στιγμή.

Διάφορες εκδόσεις του SEGWAY i2 και X2

Η σειρά mini SEGWAY

Βίντεο ασφαλούς οδήγησης

Ασφαλής οδήγηση Segway

https://www.youtube.com/watch?time_continue=178&v=0NRbGyHQY9I

Κεφάλαιο 3

Το σύστημα που αναπτύχθηκε

Στόχος αυτής της εργασίας είναι να κατασκευαστεί ένα όχημα Segway με χαμηλό κόστος, χωρίς όμως να υστερεί σε αξιοπιστία. Η απόκτηση τέτοιων οχημάτων σήμερα είναι υψηλού κόστους και ένα δύσκολο να αποκτηθούν. Το Segway στην αγορά χρησιμοποιεί τόσο αισθητήρα γυροσκόπιο και επιταχυνσιόμετρο για τη λειτουργία του καθώς επίσης και αισθητήρες κλίσης για να διαβάζετε η τιμή της κλίσης της διεύθυνσης του. Χρησιμοποιεί υψηλής ποιότητας επεξεργαστές και μικρο-ελεγκτές για τη λειτουργία του.

Τα οχήματα αυτά έχουν ένα μειονέκτημα λόγω έλλειψης σταθερότητας που συχνά προκαλούν τραυματισμούς σε πολλούς ανθρώπους, πρόβλημα που λάβαμε σοβαρά υπόψη στο σχεδιασμό και στην ανάπτυξη του.

Σχεδιάστηκε και κατασκευάστηκε ώστε έχει αρκετά καλή σταθερότητα και λειτουργία για την ασφάλεια του χρήστη. Αυτό το όχημα είναι εύκολο στη χρήση, καθώς έχουμε εισαγάγει σύστημα διακόπτη για τον έλεγχο της εκκίνησης του.

Περιγραφή συστήματος

Το όχημα μας αποτελείται από ένα πλαίσιο στο οποίο πέραν του ότι στέκεται ο αναβάτης, στερεώνονται και όλα τα επιμέρους μηχανικά και μεγάλο τμήμα των ηλεκτρικών συστημάτων. Όλα αυτά είναι τοποθετημένα στο κάτω μέρος του πλαισίου ώστε η πάνω πλευρά να παραμείνει ελεύθερη στην πρόσβαση του οδηγού.

Το μηχανικό τμήμα αποτελείται από:

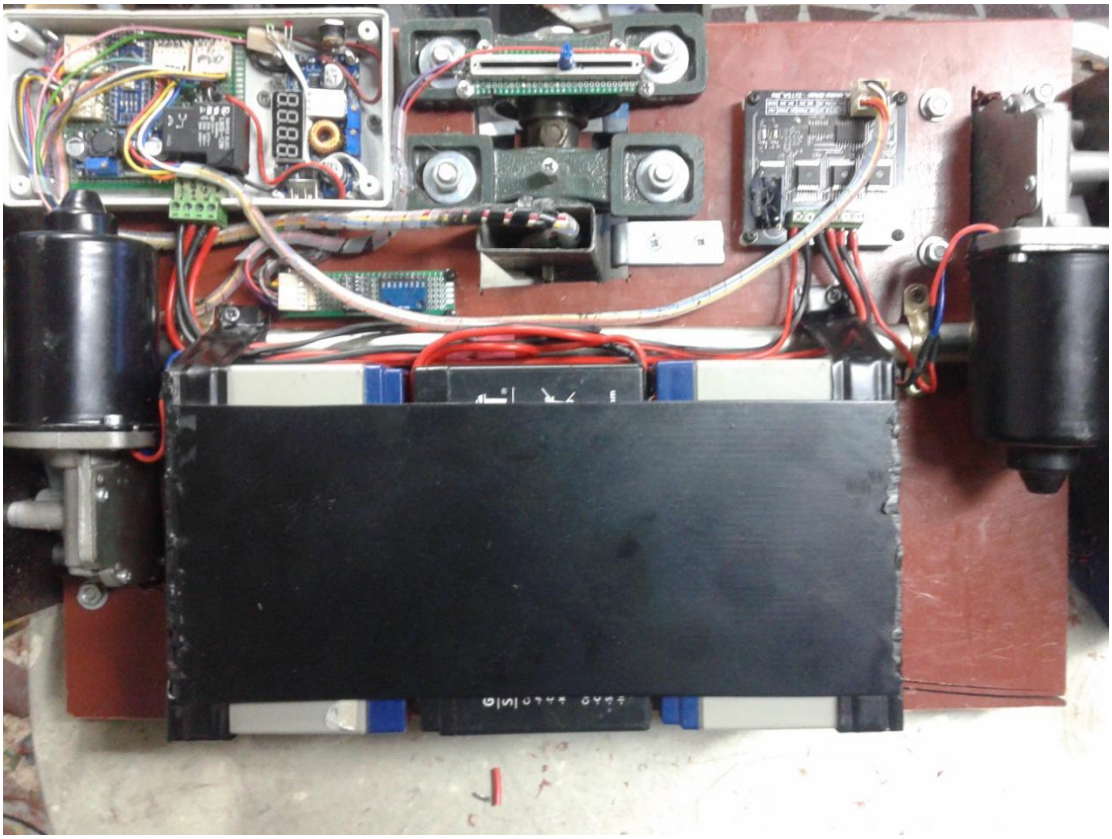
- Τιμόνι με το σύστημα κίνησής και επαναφοράς του σε κάθετη θέση
- Ρόδες με το σύστημα μετάδοσης κίνησης και τον άξονα στερέωσης

Το ηλεκτρικό τμήμα

- 2 κινητήρες
- 3 μπαταρίες 12V – 7Ah (συνολικά 12V – 21Ah)
- Ρυθμιστής Τάσης
- Φορτιστής μπαταριών.

Το τμήμα ελέγχου τώρα είναι πιο σύνθετο. Εκεί εντοπίζουμε τα ακόλουθα:

- Μικροεπεξεργαστής Arduino nano
- Γυροσκόπιο – επιταχυνσιόμετρο MPU6050
- Ελεγκτής κινητήρων Dual motor driver
- Step down converter (για τη τροφοδοσία 5V από τα 12V)
- Logic level converter
- Οθόνη LCD 20x4 - I2C επικοινωνίας
- Ποτενσιόμετρα

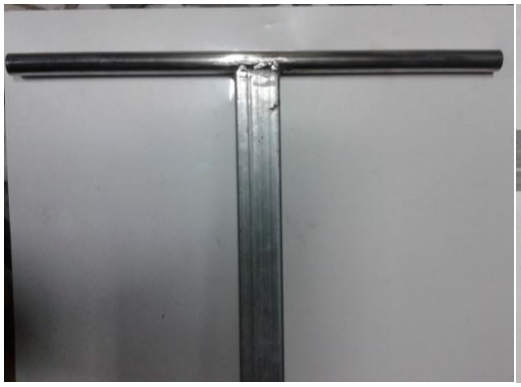


Η κάτω πλευρά του πλαισίου με τα περισσότερα μέρη του οχήματος

3.1 Το μηχανικό μέρος

3.1.1 Τιμόνι - μηχανισμός κίνησής - σύστημα επαναφοράς τιμονιού.

Το τιμόνι είναι συνδεδεμένο με το πλαίσιο και χρησιμεύει στην υποστήριξη του αναβάτη αλλά και στην οδήγηση του οχήματος.



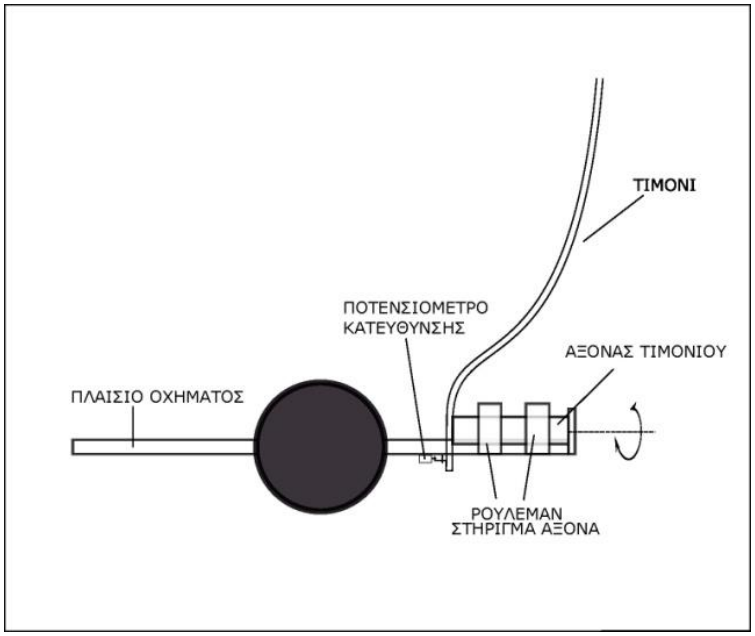
Πάνω τμήμα τιμονιού.

Κάτω τμήμα τιμονιού.

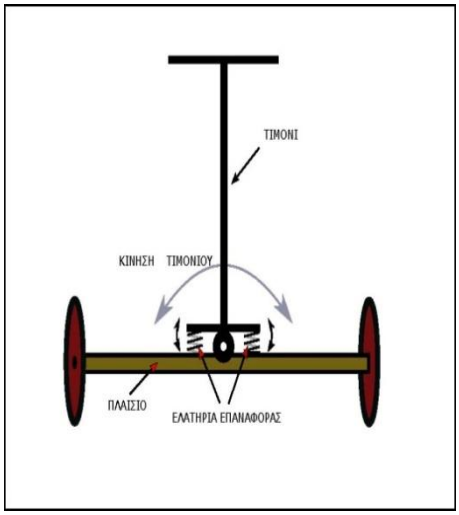
Παρέχει την απαιτούμενη υποστήριξη του αναβάτη και την καθοδήγηση του οχήματος με κλίση. Ο αναβάτης πρέπει να αισθάνεται κάποια αντίσταση κατά την κλίση του τιμονιού, και γι αυτό το λόγο τοποθετήθηκαν ελατήρια τα οποία το επαναφέρουν σε κατακόρυφη

θέση όταν δεν του ασκείται κάποια δύναμη.

Η κατασκευή του έγινε με συνδυασμό ανοξείδωτου σωλήνα διαμέτρου 2.5 εκατοστών και παραλληλόγραμμης στράνζας 2x5 εκατοστών. Το τιμόνι στηρίζεται σε έναν κάθετο σε αυτό άξονα ο οποίος περιστρέφεται στηριζόμενο σε ρουλεμάν και κρατείται σε κατακόρυφη θέση με την βοήθεια δυο ελατηρίων. Τα ελατήρια παρέχουν αντίσταση στην κίνηση του τιμονιού όταν ο αναβάτης το γέρνει δεξιά ή αριστερά δίνοντάς ένα φυσικό τρόπο για την περιστροφή του οχήματος. Όταν η τιμόνι γέρνει η γωνία γίνεται αισθητή από ένα ποτενσιόμετρο το οποίο εξάγει τη γωνία κλίσης ως τάση σε μια αναλογική είσοδο στον κεντρικό επεξεργαστή.

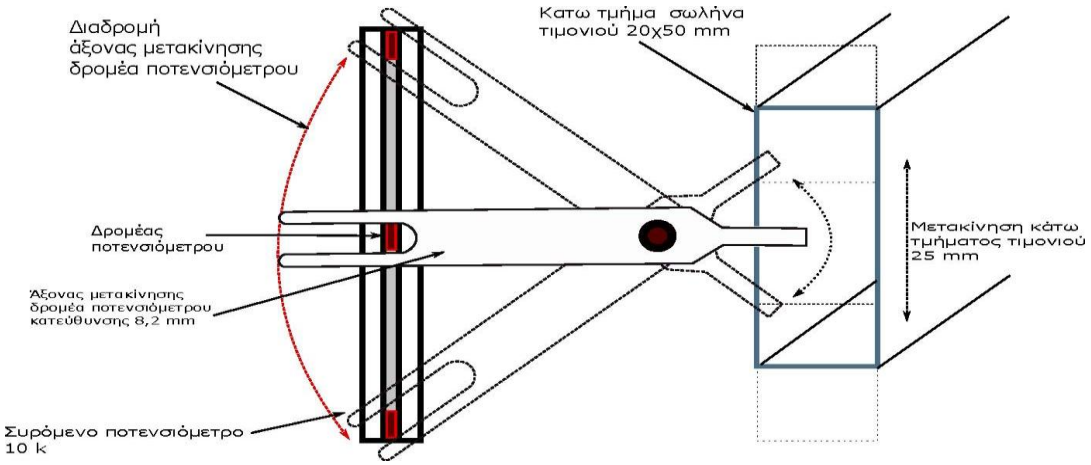


Διάγραμμα μηχανισμού στήριξης τιμονιού και το προσαρμοσμένο ποτενσιόμετρο (Inkscape)

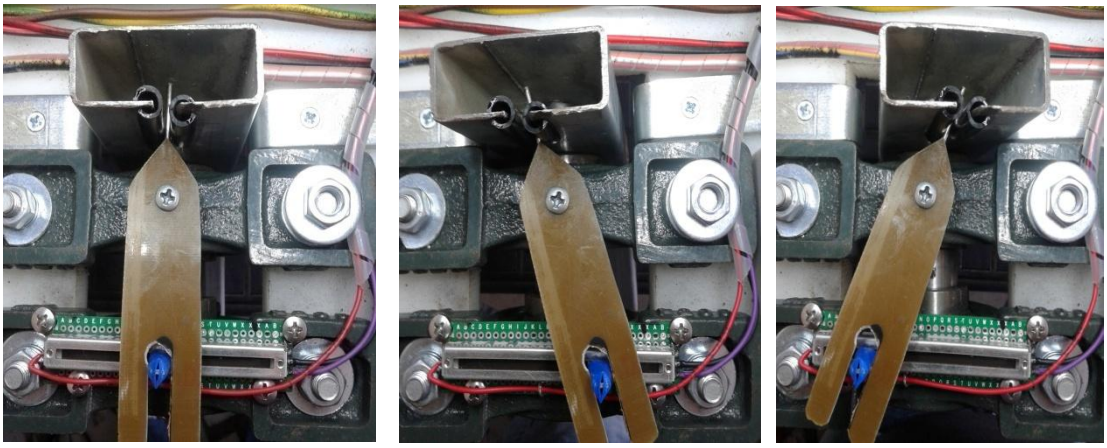


Ο μηχανισμός τιμονιού με τα ελατήρια επαναφοράς (Inkscape)

Δοκιμή μηχανισμού ελατηρίων σε κίνηση αριστερά –δεξιά.



Διάγραμμα προσαρμογής συρόμενου ποτενσιόμετρου κατεύθυνσης στο τιμόνι.



Το ποτενσιόμετρο κατεύθυνσης προσαρμοσμένο με μοχλό στον άξονα τιμονιού.

Ρόδες.

Είναι το μέσο ώθησης και επαφής με το έδαφος του οχήματος. Είναι κατανοητό πως μια μεγάλη διάμετρος τροχού διανύει μεγαλύτερη απόσταση από ό, τι μια μικρότερη διάμετρος. Λογικά, αυτό έχει νόημα, γιατί κάθε φορά που ο κινητήρας κάνει μία πλήρη περιστροφή, το όχημα πρέπει να διανύει την ίδια απόσταση με την εξωτερική περιφέρεια του τροχού. Έτσι, ένα μεγαλύτερος τροχός διανύει μεγαλύτερη απόσταση από ένα μικρότερο, χρησιμοποιώντας τις ίδιες στροφές κινητήρα.
Για την κατασκευή του οχήματός μας χρησιμοποιήθηκαν ελαστικά τύπου 4.10 / 3.50 - 4 με περιφέρεια = 80 cm
Η ρόδες τοποθετήθηκαν και στερεώθηκαν σε ένα ατσάλινο άξονα διαμέτρου 2.2cm ο οποίος φαίνεται στην εικόνα παρακάτω

Οι ρόδες με τα ενσωματωμένα γρανάζια, οι αλυσίδες, τα γρανάζια κινητήρων, και ο άξονας στήριξης

Μετάδοση κίνησης.

Για τη μετάδοση κίνησης από τους κινητήρες στις ρόδες χρησιμοποιούνται αλυσίδες και οδοντωτοί τροχοί. Οι αλυσίδες συνήθως πωλούνται με καθολική σύνδεση, και κόβονται – επανασυνδέονται με ειδικό σύνδεσμο, ώστε το μέγεθος της να ταιριάζει στις ανάγκες μας. Αυτό το είδος της αλυσίδας καθώς και οι ανάλογοι οδοντωτοί τροχοί είναι τα συνηθέστερα για κατασκευές, και τα συναντάμε συνήθως σε ηλεκτρικά σκούτερ. Μπορούμε εύκολα λοιπόν να επιλέξουμε τον λόγο μεταξύ των ζευγών των οδοντωτών τροχών επιλέγοντας τον αριθμό των δοντιών για κάθε γρανάζι. Χρησιμοποιούμε ένα μικρό οδοντωτό τροχό στον κινητήρα και ένα μεγαλύτερο οδοντωτό τροχό στις ρόδες, παρέχοντας έτσι μείωση ταχύτητας και αυξημένη ροπή.

Γρανάζια – αλυσίδα - κινητήρας – άξονας - ρόδα όλα στηρίζονται στο πλαίσιο.

Κατά την τοποθέτηση της αλυσίδας είναι σημαντικό να διατηρηθεί η κατάλληλη τάση (τέντωμα) της αλυσίδας. Πάρα πολύ τάση μπορεί να οδηγήσει σε μια σπασμένη αλυσίδα, ενώ μια χαμηλή τάση δημιουργεί μια χαλαρή μετάδοση κίνησης που μπορεί να οδηγήσει σε φθορά του οδοντωτού τροχού, αλλά και σημαντική αστάθεια στο σύστημα.

Τα τελικά τμήματα του συστήματος μετάδοσης κίνησης είναι οι τροχοί και τα ελαστικά, τα οποία μεταφέρουν τη δύναμη που παράγεται από τους κινητήρες στο έδαφος.

Υπολογισμός μέγιστης ταχύτητας του οχήματος

Γνωρίζουμε ότι ο λόγος των διαμέτρων δύο συνεργαζόμενων γραναζιών καλείται σχέση μετάδοσης και ότι επίσης η σχέση μετάδοσης μπορεί να εκφραστεί και με τον αριθμό των δοντιών κάθε γραναζιού. Στην εργασία μας προσαρμόσαμε γρανάζι 15 δοντιών στον κάθε κινητήρα και γρανάζι 22 δοντιών στον κάθε τροχό.

Ο λόγος των γραναζιών είναι 15 /22 = 0,68 και αυτό μας δείχνει πως οι τροχοί θα κινούνται με μέγιστη ταχύτητα **95 x 0.68 = 64.5** στροφές το λεπτό. Γνωρίζοντας την διάμετρο του τροχού και τις στροφές ανά λεπτό μπορούμε να υπολογίσουμε την μέγιστη ταχύτητα του οχήματός μας. Έτσι έχουμε:

64.5 στροφές το λεπτό x 60 = 3876 στροφές την ώρα

Η περιφέρεια του κάθε τροχού είναι 80 εκατοστά = 0.8 μέτρα.

Επομένως 3876 x 0.8 = 3100 μέτρα την ώρα (3.1Km/h)

** Η μέγιστη ταχύτητα περιστροφής του κινητήρα μας είναι 95 rpm βάση των τεχνικών χαρακτηριστικών του (θα αναφερθούμε παρακάτω)*

Η σχέση μετάδοσης φανερώνει πόσο μεταβάλλεται ο αριθμός των στροφών που περιστρέφεται το δεύτερο γρανάζι. Η αντίστροφη τιμή της (1/0,68 = 1,47) δείχνει πόσες φορές πολλαπλασιάζεται η ροπή που φτάνει στο δεύτερο γρανάζι.

Έτσι η μέγιστη ροπή στους τροχούς θα είναι 19,207 N m * 1,47 = 28,22 Nm

3.2 Το ηλεκτρικό μέρος

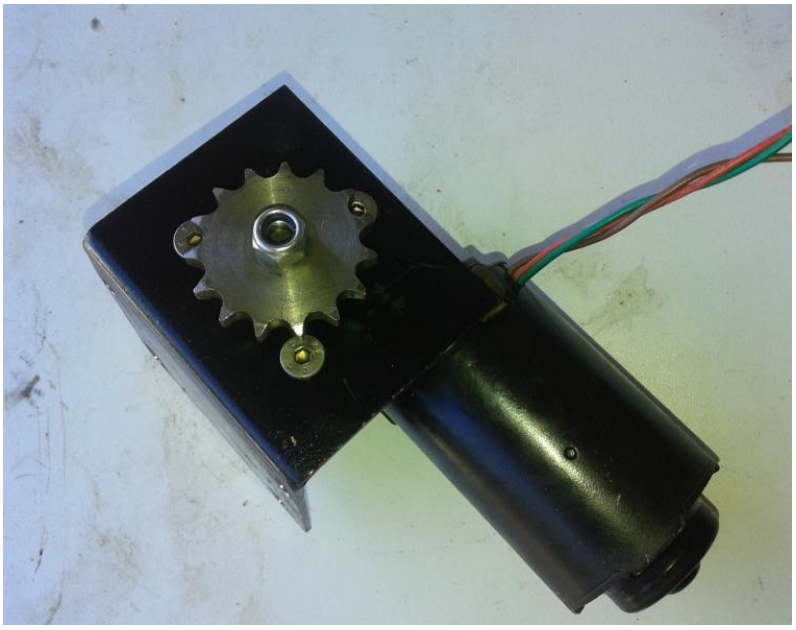
3.2.1 Κινητήρες.

Η επιλογή κινητήρων ήταν ένα από τα πρώτα πράγματα που έγιναν. Μετά από αρκετή έρευνα επιλέχτηκαν, λόγω χαμηλού κόστους, κινητήρες 12V όπως αυτοί που χρησιμοποιούνται στου υαλοκαθαριστήρες αυτοκινήτων οι οποίοι διαθέτουν ενσωματωμένο σύστημα μείωσης στροφών (αύξησης ροπής). Με μέγιστη ισχύ εξόδου 50 watt (1/15 HP) και μέγιστη ροπή 19,2 N m το καθένα, ήταν μια συμβιβαστική λύση που πιστέψαμε ότι θα λειτουργήσει αξιόπιστα με τον ελαφρύ σχεδιασμό μας. Το σύστημα μείωσης στροφών συνδυάζεται με τους κινητήρες για τη μετατροπή της ενέργειας των κινητήρων από υψηλής ταχύτητας - χαμηλής ροπής σε χαμηλής ταχύτητας / υψηλής ροπής.

Τεχνικά χαρακτηριστικά κινητήρα:

- Ονομαστική ροπή: 53 in-lb = 5,988 N
- Μέγιστη ροπή: 177 in-lb = 19,207 N m
- Υψηλή ταχύτητα χωρίς φορτίο: **95** rpm, 1.5A (12VDC)
- Χαμηλή ταχύτητα χωρίς φορτίο: **75** rpm, 1.0A (12VDC)
- Μέγιστη ισχύς: 50W/12VDC
- Θόρυβος κινητήρα: < 45dB
- Μέγεθος σπειροειδούς / μεταλλικού σπειρώματος: M-6
- Περιστροφή άξονα 360 μοίρες
- Βάρος κινητήρα: 1,22Kg
- Διαστάσεις: 18.4cm x 10.16cm x 8.9cm

Οι δύο κινητήρες με ενσωματωμένο σύστημα μείωσης στροφών.



Ο κινητήρας με το γωνιακό στήριγμα και το γρανάζι έτοιμος να τοποθετηθεί στο πλαίσιο.

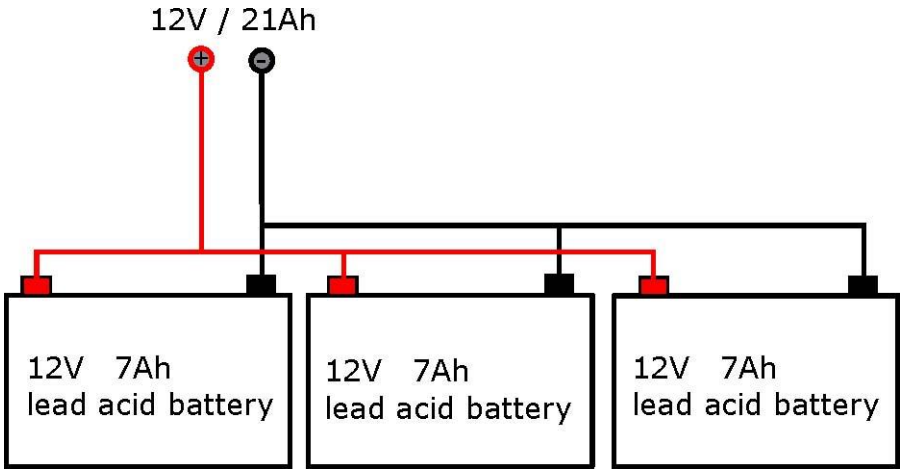
3.2.2 Τροφοδοσία - μπαταρίες

Για να πάρουμε τη βέλτιστη δύναμη από τους κινητήρες, θα πρέπει αυτοί να τροφοδοτούνται με την συνιστώμενη τάση λειτουργίας τους, η οποία είναι 12V DC.
Το συνολικό βάρος του οχήματος επηρεάζει την κατανάλωση ρεύματος από τους κινητήρες, και κατά συνέπεια το χρόνο λειτουργίας του. Η χρήση μπαταριών οξέων-μολύβδου Lead-Acid είναι η φθηνότερη λύση και παρέχει μεγάλη χωρητικότητα Amp/ώρες, αλλά οι μπαταρίες αυτές έχουν αρκετό βάρος. Η χρήση μπαταριών LiPo (πολυμερών λιθίου) είναι ακριβότερη, αλλά ζυγίζει περίπου το ¼ των μπαταριών Lead-Acid.

Μπαταρία οξέων μολύβδου.

Οι μπαταρίες οξέων μολύβδου (Lead – Acid).

Οι μπαταρίες Lead - Acid (οξέων μολύβδου) είναι η πιο αποτελεσματική λύση στην εργασία μας καθώς έχουν χαμηλή τιμή και το βάρος δεν αποτελεί το πρωταρχικό μέλημα. Χρησιμοποιούμε τρείς μπαταρίες 12V /7Ah συνδεδεμένες παράλληλα ώστε να μας παρέχουν συνολικά 12V / 21Ah.



Παράλληλη σύνδεση τριών μπαταριών.

Χωρητικότητα των μπαταριών.

Η χωρητικότητα των μπαταριών μετριέται σε αμπερώρια (Ah). Μια μπαταρία για παράδειγμα των 100Ah θεωρητικά μπορεί να παρέχει ρεύμα έντασης 1 αμπέρ (ampere) για 100 ώρες πριν να εκφορτιστεί ολοκληρωτικά. Αυτό όμως ποτέ δεν πρέπει να συμβαίνει, μια μπαταρία δεν πρέπει να εκφορτίζεται 100% γιατί χάνει από την διάρκεια ζωής της. Πρακτικά λοιπόν το 100% δεν μπορεί να είναι χρήσιμο πάντα, κάτω από το 30% δεν πρέπει να πέφτει για κανένα λόγο, ενώ αν η χρήση της (φόρτιση – εκφόρτιση) είναι καθημερινή καλό είναι να μην πέφτει κάτω από το 60% της ικανότητας αποθήκευσης.

Αυτονομία οχήματος.

Η συνολική μέγιστη ισχύς των κινητήρων μας είναι 2 x 50 watt = 100 watt που σημαίνει ότι για μέγιστη απόδοση χρειάζονται 100watt / 12 V = 8,3 A Από αυτά προκύπτει πως το όχημα θα ήταν ικανό να λειτουργήσει με μέγιστη ταχύτητα για 28Ah / 8,3A = **2,5 ώρες**. Για τις μπαταρίες που χρησιμοποιούμε βάση των παραπάνω μπορούμε να αξιοποιήσουμε το 40% της χωρητικότητας τους δηλαδή 21Ah x 0.4 = 8,4Ah

Άρα έχουμε 8,4Ah /8.3A = **1 ώρα λειτουργίας με μέγιστη απόδοση κινητήρων.**

Φυσικά αυτοί οι υπολογισμοί δεν λαμβάνουν υπόψη το βάρος του χειριστή καθώς το επίπεδο του εδάφους που και τα δύο επηρεάζουν αρνητικά τον χρόνο λειτουργίας.

3.2.3 Ο Ρυθμιστής Τάσης

Στον σχεδιασμό μας καλούμαστε να παρέχουμε μια τάση 5V για τον μικροελεγκτή και τα πρωτοβάθμια περιφερειακά, μια τάση 3.3V για τη μονάδα επιταχυνσιόμετρου - γυροσκόπιου, καθώς και μια ενισχυμένη τάση 12V τροφοδοσίας των κινητήρων (προκειμένου να αυξηθεί η ταχύτητα τους χωρίς φορτίο σε 50 RPM),.

Όλες αυτές οι τάσεις θα προέλθουν από μία και μόνο πηγή δύο μπαταριών 12V συνδεδεμένων παράλληλα. Είναι σαφές ότι απαιτείται ένας step-down ρυθμιστής μεταγωγής τάσης για τη γραμμή των 12V των κινητήρων. Για να επιτευχθεί αυτό γίνεται χρήση ενός DC-DC Buck Converter Step Down LM2596 module, καθώς αυτό είναι φθηνό, πλήρως ολοκληρωμένο πακέτο ρύθμισης που ταιριάζει απόλυτα στις προδιαγραφές του σχεδιασμού μας.

Το module αυτό χρησιμοποιήθηκε για τη ρύθμιση των 5V με είσοδο 12V της μπαταρίας. Ωστόσο, επειδή η πτώση από 5V σε 3.3V είναι σχετικά ελάχιστη, και η κατανάλωση ρεύματος από το MPU-6050 (3.3V) είναι επίσης αρκετά μικρή, ήμαστε σε θέση να χρησιμοποιήσουμε απλά τη γραμμή 3.3V του ARDUINO.

Ρυθμιστής τάσης 12V σε 5V

3.2.4 Φόρτιση μπαταριών.

Η φόρτιση των μπαταριών γίνεται με ένα *DC-DC Power Step-down Charge Module* με τάση εισόδου 19V DC από τυπικό τροφοδοτικό όπως αυτά των laptop.

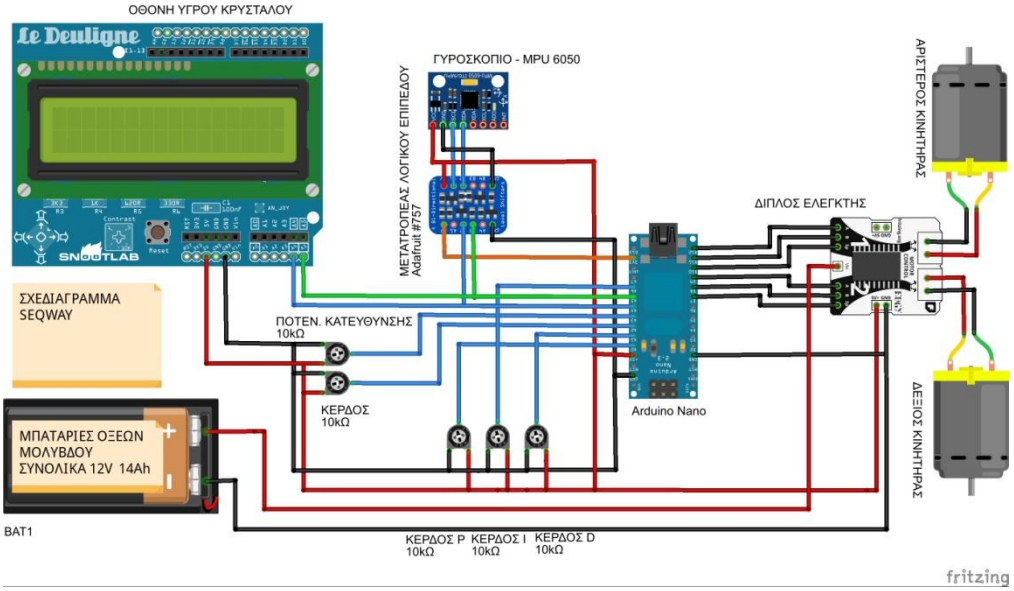
Μονάδα φόρτισης μπαταριών μολύβδου

.

Είναι ρυθμισμένος για φόρτιση μπαταριών Lead-Acid 12v. Χρησιμοποιούμε το χαμηλότερο επίπεδο φόρτισης βάση προδιαγραφών των μπαταριών C/6 = 7A/6 = 1.15A όπου C είναι η χωρητικότητα της κάθε μπαταρίας. Για την φόρτιση και των τρειών μπαταριών συγχρόνως σε παράλληλη σύνδεση η συνολική χωρητικότητα γίνεται 3 x 7 = 21Ah και έτσι το επίπεδο φόρτισης τους θα πρέπει να είναι C/6 = 3,5A.
Η μονάδα φόρτισης είναι τοποθετημένη στο κάτω μέρος του οχήματος.

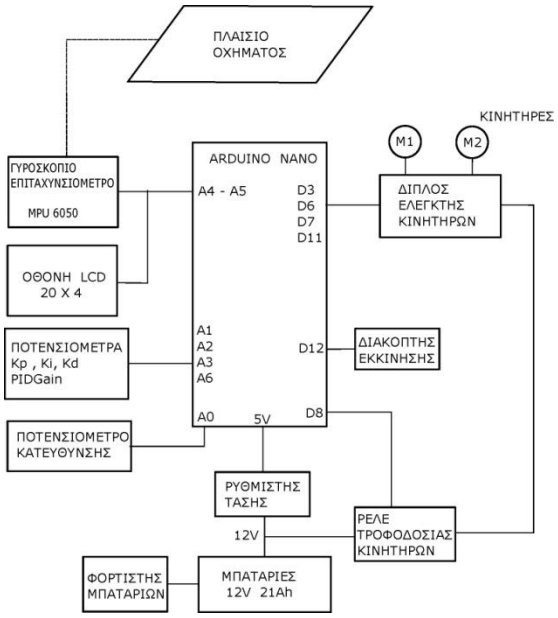
3.3 Το σύστημα ελέγχου

Σε αυτή την ενότητα θα γίνει μια λεπτομερείς περιγραφή των μονάδων που είναι απαραίτητες για την λειτουργία και τον έλεγχο του οχήματός μας.
Χρειάζεται ένας αξιοσημείωτος αριθμός μονάδων και φυσικά η συνδεσμολογία μεταξύ τους είναι αρκετά σύνθετη. Κάποιες μονάδες προσπαθήσαμε να τις ενσωματώσουμε σε κάποιο κουτί κατασκευών για να αποφύγουμε τις πολλές καλωδιώσεις αλλά και συγχρόνως να τις προστατέψουμε από υγρασία και άλλες καταπονήσεις. Ένα αναλυτικό σχεδιάγραμμα αυτών των διασυνδέσεων σχεδιασμένο στο γνωστό *FRITZING* φαίνεται παρακάτω.



Το πλήρες διάγραμμα συνδέσεων των μονάδων του οχήματος

Για να γίνουν πιο κατανοητές οι δρομολογήσεις των συνδέσεων σχεδιάστηκε ένα απλό *block διάγραμμα* αυτών.



Διάγραμμα συνδέσεων των μονάδων του οχήματος

Πίνακες συνδεσμολογιών του Arduino nano

ARDUINO UNO PIN	ΣΥΝΔΕΣΗ
0 – RX	
1 – TX	
D2 – INTERRUPT	LA_IS & RA_IS MOTOR M1 / ΕΛΕΓΧΟΣ ΡΕΥΜΑΤΟΣ ΚΙΝΗΤΗΡΑ M1
D3 – PWM	PWM_M1
D4 -	LB_IS & RB_IS MOTOR M2 / ΕΛΕΓΧΟΣ ΡΕΥΜΑΤΟΣ ΚΙΝΗΤΗΡΑ M2
D5 – PWM	
D6 – PWM	DIR_M2
D7	DIR_M1
D8	RELAY CONTROL / MOTOR POWER
D9 – PWM	
D10 – PWM	BUZZER
D11 – PWM	PWM_M2
D12	START SWITCH
D13	LED
A0	ΠΟΤΕΝΣΙΟΜΕΤΡΟ ΚΑΤΕΥΘΥΝΣΗΣ (
A1	ΠΟΤΕΝΣΙΟΜΕΤΡΟ Kp
A2	ΠΟΤΕΝΣΙΟΜΕΤΡΟ Ki
A3	ΠΟΤΕΝΣΙΟΜΕΤΡΟ Kd
A4 – SDA	MPU 6050 / LCD SDA
A5 - SCL	MPU 6050 / LCD SCL
A6	ΠΟΤΕΝΣΙΟΜΕΤΡΟ PIDGain
A7	

Πίνακας συνδεσμολογίας ARDUINO

Πίνακας συνδεσμολογιών επιμέρους τμημάτων – καλωδιώσεις

MPU Connector		
1	ΜΩΒ	STEERING POTENSIOMETER → A0
2	ΚΟΚΚΙΝΟ	+5V
3	ΓΑΛΑΖΙΟ	+3.3V
4	ΜΑΥΡΟ	GND
5	ΚΙΤΡΙΝΟ	A5 SCL
6	ΠΟΡΤΟΚΑΛΙ	A4 SDA

LCD Connector		
1	ΠΟΖ	START SWITCH → D12
2	ΛΕΥΚΟ	+12V BATT μέσω διόδου
3	ΚΟΚΚΙΝΟ	+5V
4	ΜΑΥΡΟ	GND
5	ΓΑΛΑΖΙΟ	A5 SCL
6	ΚΙΤΡΙΝΟ	A4 SDA

PID Connector		
1	ΠΟΡΤΟΚΑΛΙ	Potensiometer Kp → A1
2	ΠΡΑΣΙΝΟ	Potensiometer Ki → A2
3	ΓΑΛΑΖΙΟ	Potensiometer Kd → A3
4	ΛΕΥΚΟ	SYSTEM POWER SWITCH +12V τροφοδοσία για step down converter
5	ΜΩΒ	LED D8
6	ΚΙΤΡΙΝΟ	Balance Led D5

MOTOR Connector		
1	ΚΟΚΚΙΝΟ	+5V
2	ΜΑΥΡΟ	GND
3	ΠΟΡΤΟΚΑΛΙ	M1 PWM → D3
4	ΛΕΥΚΟ	M1 DIR → D7
5	ΚΙΤΡΙΝΟ	M2 PWM → D11
6	ΓΑΛΑΖΙΟ	M2 DIR → D6

3.3.1 Ο μικροεπεξεργαστής Arduino nano

Το Arduino είναι μια ηλεκτρονική πλατφόρμα ανοικτού κώδικα και σχεδιασμού, που βασίζεται σε ευέλικτο και εύκολο στη χρήση υλικό και λογισμικό. Προορίζεται για καλλιτέχνες, σχεδιαστές, υλοποίηση χόμπι και δραστηριοτήτων, και γενικότερα για οποιονδήποτε ενδιαφέρεται να δημιουργήσει αλληλεπιδραστικά αντικείμενα ή περιβάλλοντα.

Για να μιλήσουμε λίγο πιο τεχνικά, υπάρχει ένα κύκλωμα που χρησιμοποιεί μικροελεγκτή, το οποίο μας δίνει ένα αριθμό πυλών οι οποίες μπορεί να λειτουργήσουν είτε ως είσοδοι είτε ως έξοδοι στα κυκλώματά μας. Αυτές τις εισόδους ή εξόδους μπορούμε να τις διαχειριστούμε γράφοντας κώδικα στο περιβάλλον προγραμματισμού Arduino IDE που έχει βασιστεί στη γλώσσα C/C++.

Στην επίσημη σελίδα του Arduino (http://arduino.cc/) μπορείτε να βρείτε πολλές πληροφορίες για αυτό, και να κατεβάσετε το περιβάλλον προγραμματισμού από την αντίστοιχη σελίδα (http://arduino.cc/en/Main/Software).

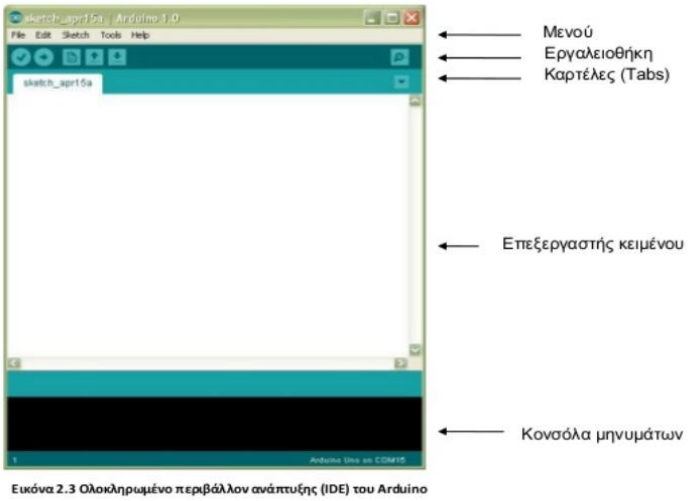
Εκτός από τη βασική έκδοση του περιβάλλοντος Arduino IDE, υπάρχει και μια παραλλαγμένη έκδοση του *Scratch* , η οποία μπορεί να χρησιμοποιηθεί για να γράψουμε προγράμματα για το Arduino, η *S4A - Scratch For Arduino*, η οποία επίσης είναι ανοικτού κώδικα και δωρεάν. Το πλεονέκτημα της έκδοσης αυτής είναι ο οπτικός προγραμματισμός (blocks όπως στο Scratch) σε σχέση με το γράψιμο εντολών στο κλασσικό περιβάλλον.

Παρόμοιας λογικής είναι και το *ArduBlock*, το οποίο επίσης χρησιμοποιεί οπτικό προγραμματισμό μέσω έτοιμων blocks για τον προγραμματισμό του. Ακόμα, υπάρχουν οπτικές εκδόσεις στο διαδίκτυο (web περιβάλλοντα), όπως το *BlocklyDuino* ή το *ArduinoMio*.

Στις επόμενες ενότητες θα δούμε αναλυτικά τη λειτουργία του μικροελεγκτή και τον βασικό προγραμματισμό του, μέσα από την κλασσική πλατφόρμα του Arduino IDE, ώστε να εισαχθούμε στο βασικό περιβάλλον και αφού αποκτήσει μια πρώτη ευχέρεια και κατανόηση των βασικών αρχών στον προγραμματισμό, να δοκιμάσει μόνος ή με βοήθεια και τις υπόλοιπες προσφερόμενες λύσεις.

Εγκατάσταση περιβάλλοντος Arduino IDE

Για να προγραμματίσουμε τη μονάδα μας θα χρειαστούμε το περιβάλλον προγραμματισμού Arduino IDE (εικόνα 1). Στο περιβάλλον αυτό γράφουμε κώδικα (βασίζεται στη γλώσσα C/C++) τον οποίο μετά μεταγλωττίζουμε και μεταφορτώνουμε στη μονάδα μας. Το Arduino IDE υπάρχει σε εκδόσεις για Windows, Mac και Linux και μπορούμε να το κατεβάσουμε εντελώς δωρεάν από την επίσημη ιστοσελίδα (http://arduino.cc/en/Main/Software).



Εικόνα 2.3 Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino

Περιβάλλον προγραμματισμού Arduino IDE

Το περιβάλλον αυτό έχει εξελληνισμένο μενού, καθώς και αρκετά έτοιμα παραδείγματα χρήσης βασικών λειτουργιών (Αρχείο => Παραδείγματα)

Τάση λειτουργίας

Το Arduino Nano μπορεί να δουλέψει με ρεύμα από τη USB θύρα του υπολογιστή μας ή με αυτόνομη παροχή ρεύματος από μπαταρία. Η μονάδα παρέχει σταθερά τάση 5V στις εξόδους της.

Για παροχή ρεύματος στη μονάδα από εξωτερική πηγή συνδέουμε απευθείας στα pins που προορίζονται για αυτό το σκοπό: (+) στο Pin VCC IN και (-) στο Gnd δίπλα του.

Στην περίπτωση που είναι συνδεδεμένη η μονάδα μας μόνιμα με θύρα USB τότε δουλεύει χωρίς πρόβλημα με τα 5V που παρέχει η USB θύρα.

Θύρες εισόδου/εξόδου (Pins)

Το Arduino nano έχει 14 ψηφιακές θύρες εισόδου ή εξόδου (digital input/outpit pins) και 8 αναλογικές εισόδους (analog input pins). Οι 14 ψηφιακές θύρες ονομάζονται με νούμερα από το 0 έως το 13, ενώ οι 8 αναλογικές με το γράμμα Α ακολουθούμενο από ένα νούμερο από 0 μέχρι το 7 (π.χ. A3). Στην έξοδο τα pins μπορούν να δώσουν 0 έως και 5V τάση. Από τις 14 ψηφιακές θύρες οι έξι, και ειδικότερα οι 3, 5, 6, 9, 10, 11, είναι και PWM θύρες (Pulse Width Modulation), δηλαδή μπορούν να προσομοιώσουν αναλογικές εξόδους.

Έτσι, συνοπτικά για την είσοδο και έξοδο έχουμε:

- Για ψηφιακή είσοδο, χρησιμοποιούμε τις 14 ψηφιακές 0..13. Όταν δουλεύουν ψηφιακά, η είσοδος μπορεί να είναι ή 0 ή 5V, με τον χαρακτηρισμό LOW ή HIGH όπως θα δούμε παρακάτω.
- Για αναλογική είσοδο, δηλαδή να διαβάσουμε τιμές ρεύματος στο διάστημα 0 έως 5V, χρησιμοποιούμε τις οκτώ αναλογικές θύρες A0..A7.
- Για αναλογική έξοδο, μπορούμε να χρησιμοποιήσουμε τις έξι PWM ψηφιακές θύρες (3, 5, 6, 9, 10, 11), οι οποίες θα μας δώσουν ρεύμα εξόδου όποιας τιμή θέλουμε στο διάστημα από 0 έως 5V.

Γράφοντας κώδικα θα πρέπει να αρχικοποιήσουμε τις θύρες που χρησιμοποιούμε με τη συνάρτηση pinMode(), δηλαδή να δίνουμε την πληροφορία για όποιες χρησιμοποιήσουμε αν θα είναι για είσοδο ή για έξοδο. Η συνάρτηση αυτή αναλύεται στην επόμενη ενότητα.

Όταν χρησιμοποιείται η σειριακή οθόνη παρακολούθησης της επικοινωνίας με τον υπολογιστή, χρησιμοποιούνται τα pins 0 και 1 για αυτό, οπότε προτείνεται να μην τα χρησιμοποιούμε στις εφαρμογές μας, εκτός αν αυτό είναι απαραίτητο (π.χ. δεν μας φτάνουν τα υπόλοιπα 12 pins για την εφαρμογή μας).
Επίσης, στη θύρα 13 υπάρχει συνήθως συνδεδεμένο ήδη ένα Led πάνω στην πλακέτα Arduino nano, κι έτσι μπορούμε να το χρησιμοποιούμε για σχετικές λειτουργίες.

Προγραμματισμός - Βασικές λειτουργίες

Μετά την εγκατάσταση του Arduino IDE μπορούμε να γράψουμε τα πρώτα μας τμήματα κώδικα. Η λογική του Arduino είναι πολύ απλή - στην ουσία υπάρχουν δύο βασικές συναρτήσεις, η setup() και η loop() οι οποίες δουλεύουν ως εξής:

• **setup()** - εδώ βάζουμε όλες τις εντολές που πρέπει να τρέξουν μία φορά, όταν ενεργοποιείται η μονάδα μας (όταν δηλαδή δίνουμε ρεύμα ή όταν πατηθεί το πλήκτρο reset που υπάρχει). Συνήθως μπαίνουν αρχικοποιήσεις τιμών μεταβλητών και οπωσδήποτε ο χαρακτηρισμός των εισόδων/εξόδων που θα χρησιμοποιήσουμε (αν δηλαδή ένα συγκεκριμένο Pin θα είναι είσοδος ή έξοδος).

• **loop()** - εδώ γράφουμε το πρόγραμμά μας. Οι εντολές που υπάρχουν θα τρέξουν κι όταν φτάσει στο τέλος θα ενεργοποιηθεί ξανά η loop(), συνεχίζοντας από την αρχή της, και ξανά. Αυτό θα συμβαίνει συνεχώς, όσο έχει ρεύμα το Arduino ή μέχρι να πατηθεί το πλήκτρο reset.

Έτσι, η βασική λειτουργία του Arduino είναι ότι τρέχει η συνάρτηση setup() μία φορά στην αρχή και ακολούθως η loop() ξανά και ξανά μέχρι να το κλείσουμε (να μην τροφοδοτείται με ρεύμα) ή να πατήσουμε το πλήκτρο reset. Στην περίπτωση του Reset ξανατρέχει η συνάρτηση setup() μία φορά και ακολούθως η loop() ξανά και ξανά, όπως δηλαδή ακριβώς και όταν αρχικά ενεργοποιείται με ρεύμα ο μικροελεγκτής.

Στην περίπτωση που έχουμε κάνει αλλαγές στο πρόγραμμά μας και το φορτώσουμε στον μικροελεγκτή (θα δούμε παρακάτω τη διαδικασία αυτή) αρκεί να πατήσουμε το πλήκτρο Reset ώστε να φορτώσει το πρόγραμμά μας από την αρχή με τον τρόπο που περιγράφηκε. Ένα τυπικό πρόγραμμα έχει την παρακάτω δομή:

```
void setup() {
    /* οι εντολές εδώ θα τρέξουν μόνο στην ενεργοποίηση ή μετά από Reset */
}
void loop() {
    /* οι εντολές εδώ θα τρέχουν ξανά και ξανά, μέχρι να απενεργοποιηθεί ή να
    πατηθεί το Reset */
}
```

Δηλώσεις μεταβλητών

Όπως σε όλες τις γλώσσες προγραμματισμού, μπορώ να δηλώσω ονόματα μεταβλητών. Οι τύποι μεταβλητών που υποστηρίζονται στο Arduino είναι αρκετοί. Για έναν αρχάριο χρήστη οι παρακάτω τύποι θα είναι αρκετοί:

- boolean, με τιμές το 0 και 1 (ή True – False)
- byte, με τιμές από 0 έως και 255
- int, ακέραιος με δυνατές τιμές από -32768 έως και 32767
- long, ακέραιος με δυνατές τιμές από -2147483648 έως και 2147483647
- float, δεκαδικοί αριθμοί
- char, ένας χαρακτήρας (μέγεθος ένα Byte)
- string, πίνακας χαρακτήρων

Ένα παράδειγμα δήλωσης μεταβλητών δίνεται παρακάτω:

int ledPin = 13; // ορίζω ακέραια μεταβλητή ledPin και αρχικοποιώ την τιμή της σε 13
float SinVal; // ορίζω πραγματική μεταβλητή SinVal

Σχόλια

Όπως σε όλες τις γλώσσες προγραμματισμού, μπορώ να έχω σχόλια για την ευκολότερη κατανόηση και συντήρηση του κώδικα που γράφω. Μπορώ να χρησιμοποιήσω τις δύο κάθετες // για σχόλιο σε μία γραμμή (ότι ακολουθεί τις // αγνοείται), ή τα /* */ που περικλείουν τα σχόλια που γράφονται σε περισσότερες γραμμές (ότι υπάρχει ανάμεσα στο /* και στο */ αγνοείται).

Για παράδειγμα:

```
int ledPin = 13; // ορίζω τον αριθμό του Pin για το LED
                 /* Στον κώδικα που ακολουθεί θα προγραμματίσουμε ένα LED να
                    Αναβοσβήνει ανά 1 sec */
```

Συναρτήσεις διαχείρισης θυρών εισόδου – εξόδου (Pins)

Όπως αναφέρθηκε, η κύρια λειτουργία του μικροελεγκτή βασίζεται στο να ελέγχει τις θύρες που διαθέτει και είτε να δίνει ρεύμα είτε να παίρνει ρεύμα από αυτές. Στην αρχικοποίηση κάθε προγράμματος (μέσα στη συνάρτηση setup) θα χρειαστεί να χαρακτηρίσουμε τα Pins που χρησιμοποιούμε ως είσοδο ή ως έξοδο.

Η συνάρτηση pinMode(Pin, Mode) χρησιμοποιείται με το όνομά της και ορίσματα α) τον αριθμό Pin και β) την κατάσταση λειτουργίας που χαρακτηρίζεται με τη λέξη INPUT (είσοδος) ή OUTPUT(έξοδος).

Όπως έχουμε αναφέρει έχουμε 14 ψηφιακά Pins, 6 εκ των οποίων είναι PWM, με ονόματα 0..13 και οκτώ αναλογικά με ονόματα A0..A7.

Για παράδειγμα:

```
pinMode(12, OUTPUT);
pinMode(ledPin, OUTPUT);
pinMode(A2, INPUT);
```

Συναρτήσεις εισόδου - εξόδου ρεύματος

Για να μπορέσουμε να δώσουμε ρεύμα προς τα έξω μέσω μιας θύρας (pin) θα πρέπει πρώτα να έχει αυτή οριστεί ως εξόδου, όπως είδαμε στην προηγούμενη παράγραφο. Ακολούθως, με χρήση της κατάλληλης εντολής μπορούμε να δώσουμε κάθε φορά την επιθυμητή τάση προς τα έξω. Αντίστοιχα, για να "διαβάσουμε" από μια είσοδο, θα πρέπει αρχικά να την ορίσουμε ως είσοδο και με χρήση της κατάλληλης κάθε φοράς συνάρτησης να διαβάζουμε την αντίστοιχη τιμή.

Ψηφιακή έξοδος

Και τα 14 pins του Arduino μπορούν δουλεύουν ως ψηφιακές έξοδοι, δηλαδή δίνουν έξοδο 0 ή 5V. Αυτό γίνεται με χρήση της συνάρτησης digitalWrite(Pin, Value), όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα δώσουμε τάση εξόδου, ενώ η τάση εξόδου μπορεί να είναι 0 V ή 5 V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην παράμετρο value

- LOW : θα δώσει 0 V στην έξοδο (pin)
- HIGH : θα δώσει 5 V στην έξοδο (pin)

Για παράδειγμα:

```
digitalWrite(ledPin, HIGH);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εξόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(10, OUTPUT);
```

Ψηφιακή είσοδος

Και τα 14 ψηφιακά pins του Arduino μπορούν δουλεύουν ως ψηφιακές είσοδοι, δηλαδή να "διαβάσουν" ως είσοδο τάση με τιμή είτε 0 είτε 5V. Αυτό γίνεται με χρήση της συνάρτησης digitalRead(Pin), όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει με το όνομά της την τιμή εισόδου. Η τάση εισόδου μπορεί να είναι 0V ή 5V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην τιμή που διαβάζουμε:

- LOW : όταν λάβει τάση 0 V στην είσοδο (pin)
- HIGH : όταν λάβει τάση 5 V στην είσοδο (pin)

Για παράδειγμα:

```
Val = digitalRead(ledPin);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εισόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(10, INPUT);
```

Αναλογική έξοδος (PWM pins)

Κάποια από τα 14 Pins του Arduino έχουν την ένδειξη PWM, δηλαδή μπορούν να προσομοιώσουν την αναλογική έξοδο μέσω παλμοκωδικής διαμόρφωσης.
Έτσι, με τιμές από το 0 μέχρι το 255 προσομοιώνουμε (αναλογικά) το διάστημα από 0 έως 5V. Αυτό γίνεται με χρήση της συνάρτησης analogWrite(Pin, Value), όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα δώσουμε ρεύμα εξόδου, ενώ η τάση εξόδου κυμαίνεται από 0 V μέχρι και 5 V, οι οποίες τιμές της τάσης αναλογικά αναπαρίστανται με τιμές στη μεταβλητή value. Τιμή 0 δίνει 0V στην έξοδο (pin), τιμή 255 δίνει τάση 5V στην έξοδο (pin), ενώ αναλογικά μπορούμε να δώσουμε ενδιάμεσες τάσεις (π.χ. 122 για τάση 2,5V).

Για παράδειγμα:

```
analogWrite(ledPin, 122);
```

Υπενθύμιση: Τη λειτουργία αυτή μπορούν να υποστηρίξουν μόνο τα PWM pins κι όχι όλα τα ψηφιακά. Τα PWM pins είναι τα 3, 5, 6, 9, 10, 11.

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εξόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(10, OUTPUT);
```

Αναλογική είσοδος

Το Arduino nano έχει 8 αναλογικές εισόδους, οι οποίες χαρακτηρίζονται με τα σύμβολα A0, A1, A2, A3, A4, A5, A6, A7. Μπορούμε να συνδέσουμε κάποιο αναλογικό εξάρτημα (π.χ. ένα ποτενσιόμετρο) και να το διαβάσουμε ως είσοδο. Αυτό γίνεται με χρήση της συνάρτησης analogRead(Pin), όπου το όρισμα Pin αναφέρεται στο νούμερο της θύρας για την οποία θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει με το όνομά της την τιμή εισόδου. Η τιμή εισόδου κυμαίνεται από 0 μέχρι και 1023. Συνήθως χρησιμοποιούμε μια μεταβλητή για να καταχωρήσουμε την τιμή.

Για παράδειγμα:

```
int r = analogRead(A1);
```

Προσοχή: Η αντίστοιχη θύρα θα πρέπει να έχει οριστεί ως εισόδου στη διαδικασία setup(), με χρήση της συνάρτησης pinMode.

Για παράδειγμα:

```
pinMode(A1, INPUT);
```

Συνάρτηση καθυστέρησης – delay()

Στο πρόγραμμά μας μπορούμε να ορίσουμε μια καθυστέρηση ώστε να διαρκέσει για το χρόνο που εμείς ορίζουμε ένα γεγονός. Αυτό το επιτυγχάνουμε με χρήση της συνάρτησης delay(time) όπου στη θέση time δίνουμε το χρόνο σε ms (1/1000 sec). Η εντολή delay(time) σημαίνει ότι σταματά στο σημείο αυτό η εκτέλεση του προγράμματός μας για το χρόνο time.

Για παράδειγμα:

```
delay(1000); //σταματά την εκτέλεση του προγράμματος για 1000 ms = 1 sec
delay(500); //σταματά την εκτέλεση στο σημείο αυτό για 500 ms = 0.5 sec
```

Συνάρτηση καθυστέρησης - delayMicroseconds()

Ακριβώς αντίστοιχα με την παραπάνω συνάρτηση delay, μπορούμε να χρησιμοποιήσουμε την delayMicroseconds(), όπου ο χρόνος καθυστέρησης δίνεται πλέον σε microseconds (1/106 sec).

Για παράδειγμα:

```
delayMicroseconds(10); // σταματά την εκτέλεση για 10μsec = 1/100000 sec
```

Συνάρτηση καταγραφής χρόνου - millis()

Το Arduino έχει ενσωματωμένο ρολόι, το οποίο μετράει το χρόνο από τη στιγμή που ενεργοποιείται (ή του γίνεται reset). Η πληροφορία αυτή μας είναι διαθέσιμη σε κάθε σημείο με κλήση της συνάρτησης millis(), η οποία μας επιστρέφει το χρόνο σε milliseconds (1/1000 sec) που έχει περάσει από την ενεργοποίηση της μονάδας μας. Αυτό μας βοηθά να μετράμε το χρόνο στα προγράμματά μας, ειδικά στις περιπτώσεις που θέλουμε να "θυμόμαστε" πράγματα. Για παράδειγμα, αν θέλουμε να ελέγξουμε αν έχει περάσει συγκεκριμένο διάστημα από τότε που έγινε κάτι (π.χ. πατήθηκε τελευταία φορά ένα πλήκτρο), μπορούμε να καταγράφουμε το γεγονός σε μια μεταβλητή χρόνου και τον χρόνο αυτό να τον αφαιρούμε από τον επόμενο κτλ.

Για παράδειγμα:

```
lastPress = millis();
if (lastPress – millis() > 1000) {...}
```

Συνάρτηση αντιστοίχισης τιμών - map()

Πολλές φορές θα χρειαστεί να αντιστοιχίσουμε μια τιμή που ανήκει σε ένα πεδίο τιμών σε μια άλλη τιμή που ανήκει σε ένα άλλο πεδίο τιμών. Η μαθηματική πράξη αυτή είναι σχετικά απλή, αλλά το Arduino μας παρέχει μια συνάρτηση για να το κάνει αυτό, την

map(<τιμή>,<κάτω όριοΑ>,<πάνω όριοΑ>,<κατω όριοΒ>,<πάνω όριοΒ>)

όπου

<τιμή> είναι η τιμή που θέλουμε να μετατρέψουμε
<κάτω όριοΑ> είναι το κάτω όριο του διαστήματος της αρχικής τιμής
<πάνω όριοΑ> είναι το πάνω όριο του διαστήματος της αρχικής τιμής
<κατω όριοΒ> είναι το κάτω όριο του διαστήματος της τελικής τιμής (που θέλω να μετατραπεί)

<πάνω όριοΒ> είναι το πάνω όριο του διαστήματος της τελικής τιμής (που θέλω να μετατραπεί)

Για παράδειγμα:

```
val = map(val, 0, 1023, 0, 179); /* μετατρέπω μια αναλογική τιμή που διάβασα από ένα
                                    ποτενσιόμετρο (0 έως 1023) σε μια τιμή για ένα σέρβο (0 έως 179) */
```

Η σειριακή θύρα επικοινωνίας (Serial)

Το Arduino παρέχει μια σειριακή θύρα επικοινωνίας μεταξύ της πλακέτας και του υπολογιστή ή κάποιας συσκευής που θέλουμε. Για το σκοπό αυτό χρησιμοποιείται η σύνδεση με καλώδιο USB (όταν πρόκειται για τον υπολογιστή) ή τα pins 0 και 1 όταν θέλουμε κάποια πιο εξειδικευμένη σύνδεση (π.χ. με κάποια άλλη συσκευή). Για το λόγο αυτό προτείνεται, αν δεν είναι απαραίτητο στις εφαρμογές μας, να μην χρησιμοποιούνται τα pins αυτά.

Για να ενεργοποιήσουμε τη σειριακή θύρα επικοινωνίας αρκεί να δώσουμε στη διαδικασία setup() την εντολή Serial.begin(BaudRate), όπου το BaudRate εκφράζει το ρυθμό με τον οποίο θα μεταδίδονται τα bits (μια τιμή στα 9600 είναι συνήθως αρκετή).
Για παράδειγμα:

```
Serial.begin(9600);
```

Μπορούμε να χρησιμοποιήσουμε τη σειριακή θύρα στις εφαρμογές για αμφίδρομη επικοινωνία, δηλαδή να στείλουμε και να λάβουμε δεδομένα. Μία απλή περίπτωση χρήση της επικοινωνίας αυτής είναι για εκσφαλμάτωση (debugging) των προγραμμάτων μας, να μπορούμε δηλαδή να δούμε τί τιμές μας δίνουν μετρητές και τί τιμές έχουν οι μεταβλητές μας μέσω της οθόνης σειριακής επικοινωνίας. Μια εντολή που μας βοηθάει σε αυτό είναι η print(), που εκτυπώνει ένα μήνυμα ή τιμές ή η println() που λειτουργεί ακριβώς το ίδιο αλλά εκτυπώνοντας με αλλαγή γραμμής κάθε φορά.

Για παράδειγμα:

```
Serial.print("H epikoinwnia ksekinhse"); /* Θα εμφανίσει το μήνυμα αυτό στην οθόνη
                                            χωρίς να αλλάξει γραμμή μετά */
Serial.println(distance); /* Θα εμφανίσει την τιμή της μεταβλητής distance σε μια
                             γραμμή */
```

Όταν έχετε συνδέσει το Arduino σας με τη θύρα USB στον υπολογιστή, η σειριακή οθόνη ενεργοποιείται από το εικονίδιο πάνω δεξιά "Σειριακή Οθόνη" (εικόνα 2), και στο παράθυρο που ανοίγει μπορείτε να βλέπετε όλα τα μηνύματα που στέλνονται από τον κώδικα που έχει φορτωθεί ήδη και τρέχει στην πλακέτα

Εικόνα 2 – Σειριακή οθόνη

Δομή επιλογής

Στον προγραμματισμό πολλές φορές θα χρειαστεί να ελέγξουμε κάποια συνθήκη για να αποφασίσουμε αν θα εκτελεστεί ένα τμήμα κώδικα ή αν θα εκτελεστεί κάποιο άλλο αντί για αυτό στη θέση του. Αυτό το επιτυγχάνουμε με τη χρήση της δομής επιλογής, η οποία συντάσσεται

If<συνθήκη> {<εντολές 1> }
else {<εντολές 2> }

όπου, στη <συνθήκη> έχουμε τον έλεγχο που θέλουμε να γίνει, συνήθως χρησιμοποιώντας τους τελεστές σύγκρισης (>, <>, >=, <=), π.χ. potVal > 500. Η συνθήκη μπορεί να είναι και πιο σύνθετη, χρησιμοποιώντας τους λογικούς τελεστές (|| για το Η', && για το ΚΑΙ), π.χ. (potVal > 500) && (timePass >= 1000) .

Στα μπλοκ {<εντολές> } εκτελούνται αντίστοιχα οι εντολές που θέλουμε σε κάθε περίπτωση. Αν ισχύει η <συνθήκη> θα εκτελεστούν οι <εντολές 1>, αν δεν ισχύει οι <εντολές 2> . Σε κάθε περίπτωση, το τελευταίο κομμάτι else {<εντολές 2> } δεν είναι απαραίτητο να υπάρχει.

Τέλος, υπάρχουν πιο σύνθετες μορφές της εντολής επιλογής, οι οποίες ξεφεύγουν από το σκοπό του παρόντος εγχειριδίου – μπορείτε να τις συζητήσετε με τον καθηγητή σας όταν αυτές χρειαστούν.

Δομή επανάληψης (For)

Πολλές φορές θα χρειαστεί να επαναλάβουμε κάποια διαδικασία αρκετές φορές. Στην περίπτωση αυτή έχουμε εντολές οι οποίες επαναλαμβάνουν ένα σύνολο σύνολο εντολών όσες φορές θέλουμε, είτε μετρώντας τις επαναλήψεις είτε ελέγχοντας κάθε φορά μία συνθήκη. Η συχνότερη μορφή που συναντάμε σε μια επανάληψη είναι αυτή με τον προκαθορισμένο αριθμό βημάτων. Η σύνταξη της εντολής αυτής είναι η εξής:

for (<αρχική τιμή>; <συνθήκη τερματισμού>; <βήμα>)
{<εντολές> }

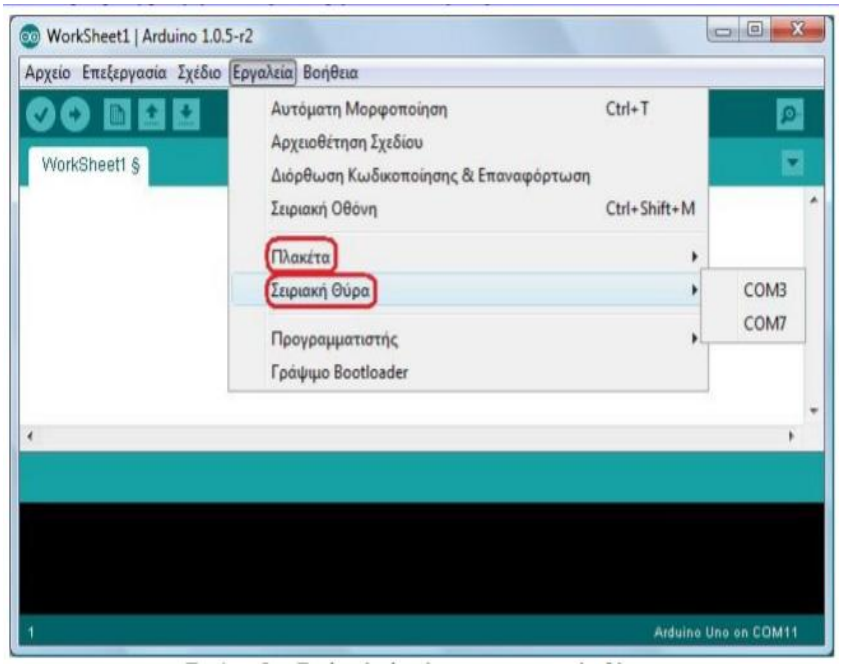
όπου χρησιμοποιείται μια μεταβλητή ελέγχου ως εξής
<αρχική τιμή>; δίνουμε την αρχική τιμή , π.χ. i = 0
<βήμα> δίνουμε την αλλαγή κάθε επανάληψης, π.χ. i+5 (το i++ που θα δείτε σημαίνει i+1)
<συνθήκη τερματισμού> η συνθήκη για να τελειώσει η επανάληψη, π.χ. i < 10 (όσο ισχύει αυτή θα τρέχει) π.χ.

```
for (i=1;i<10;i=i+1){
        brightnes= brightnes+5
                anologWrite(ledPin, brightnes);
 };
```

Υπάρχουν εντολές επανάληψης που δεν έχουν προκαθορισμένο αριθμό βημάτων, αλλά συνεχίζουν επ' αόριστο ελέγχοντας μια συνθήκη. Μια τέτοια εντολή είναι η

- While<συνθήκη> {<εντολές> } //όσο ισχύει η <συνθήκη> τρέχουν οι εντολές
- repeat {<εντολές>} until<συνθήκη> // οι <εντολές> τρέχουν όσο δεν ισχύει η συνθήκη

Μεταφόρτωση προγράμματος στη μονάδα μας



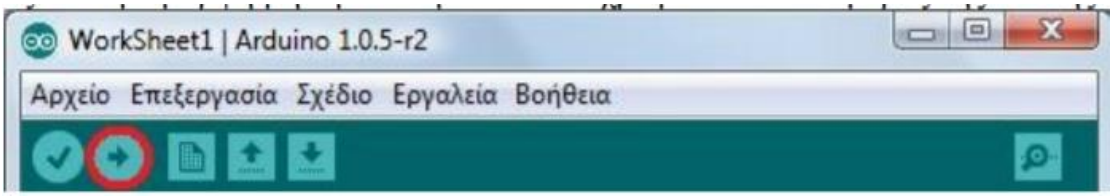
Επιλογή πλακέτας και σειριακής θύρας

Για να μεταφορτώσουμε το πρόγραμμά μας στη μονάδα θα πρέπει να τη συνδέσουμε με ένα USB καλώδιο στον υπολογιστή. Ο υπολογιστής μας θα αναγνωρίσει τη μονάδα Arduino μας ως σειριακή θύρα, κάτι που μπορείτε να επιβεβαιώσετε και από τον πίνακα ελέγχου του Η/Υ σας. Από το μενού Εργαλεία του Arduino επιλέγουμε δύο πράγματα:

- Πλακέτα - διαλέγουμε τον τύπο της μονάδας μας. π.χ. Arduino nano
- Σειριακή θύρα - είναι η σειριακή θύρα που έχει αντιστοιχίσει το λειτουργικό σας στην πλακέτα Arduino που συνδέεται μέσω του USB καλωδίου. Αν χρησιμοποιείτε Windows αυτή θα είναι της μορφής COMX (π.χ. COM3, COM11), ενώ στο Linux η θύρα θα εμφανιστεί ως /dev/ttyXXX.

Μεταγλώττιση

Ακολούθως, πατάμε το πλήκτρο της μεταγλώττισης (εικόνα 6) το οποίο θα ελέγξει το πρόγραμμά μας για λάθη και θα το προετοιμάσει για τη μεταφόρτωση στην πλακέτα. Αν τυχόν υπάρξουν λάθη, αυτά εμφανίζονται με μορφή μηνυμάτων με κόκκινο χρώμα στο κάτω μέρος της οθόνης.



Μεταφόρτωση

Τέλος, εφόσον έχουμε επιτυχώς εκτελέσει όλα τα παραπάνω, δηλαδή έχουμε συνδέσει τη μονάδα μας, έχουμε επιλέξει τον τύπο της και τη θύρα που είναι συνδεδεμένη, έχουμε γράψει κάποιο πρόγραμμα και το έχουμε μεταγλωττίσει χωρίς λάθη, μπορούμε πατώντας το πλήκτρο της φόρτωσης (εικόνα 7) να μεταφορτώσουμε το πρόγραμμα πλέον στη μονάδα και αυτό να αρχίσει να τρέχει πλέον σε πραγματικό περιβάλλον. Η διαδικασία της μεταγλώττισης επαναλαμβάνεται αυτόματα στο βήμα αυτό.

Παρακάτω ακολουθούν μερικές από τις πλατφόρμες Arduino που έχουν αναπτυχθεί και όπου η κάθε μία είτε αποτελεί εξέλιξη κάποιας άλλης, είτε έχει αναπτυχθεί για κάποιο συγκεκριμένο σκοπό :

- Arduino Uno
- Arduino Diecimila
- Arduino Duemilanove
- Arduino Mega1280
- Arduino Mega2560
- Arduino Mini
- Arduino Nano
- Arduino USB
- Arduino Stamp
- Arduino Fio
- Arduino NG
- Arduino NG+
- Arduino Extreme
- Arduino Bluetooth
- LilyPad Arduino
- Serial Arduino

Ακολουθεί ένας πίνακας που περιέχει τα βασικά χαρακτηριστικά του Arduino όσο αφορά το υλικό τους μέρος.

Πλατφόρμα Arduino	Μικροελεγκτής Atmel AVR	Flash KiB	EEPROM KiB	SRAM KiB	Ψηφιακές Επαφές	PWM	Αναλογικές Επαφές εισόδου
Diecimila	ATmega168	16	0.5	1	E/E 14	6	6
Duemilanove	ATmega168/328	16	0.5	1	14	6	6
Uno	ATmega328	32	1	2	14	6	6
Mega	ATmega1280	128	4	8	54	14	16
Nano	ATmega328P	32	1	2	14	6	8
Mega 2560	ATmega2560	256	4	8	54	14	16

Βασικά χαρακτηριστικά από τις κυριότερες πλατφόρμες Arduino

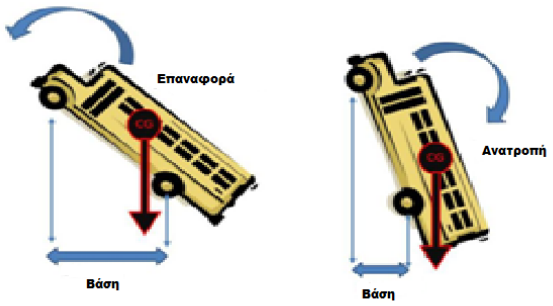
3.3.2 Γυροσκόπιο – επιταχυνσιόμετρο MPU6050

Γενικά.

Όπως αναφέραμε και παραπάνω η δυναμική του οχήματος είναι παρόμοια με αυτή του κλασσικού προβλήματος ελέγχου ανεστραμμένου εκκρεμούς, το οποίο σημαίνει ότι είναι ασταθής και επιρρεπής σε πτώσεις. Για να καταφέρουμε να σταθεί σε όρθια θέση το όχημά μας, και να μην πέσει, πρέπει να καταλάβουμε πώς λειτουργεί πραγματικά η ισορροπία. Η ισορροπία είναι κάτι που εμείς θεωρούμε δεδομένο στην καθημερινή μας ζωή. Για να γίνει αυτό…

α. Χρειαζόμαστε έναν αισθητήρα που να μας λέει πόσο «όρθια» στεκόμαστε. Στους ανθρώπους, αυτός ο αισθητήρας βρίσκεται στο εσωτερικό του αυτιού. Σε ένα ρομπότ, θα μπορούσαμε να χρησιμοποιήσουμε έναν αισθητήρα, όπως το MPU6050 που μπορεί να χρησιμοποιηθεί για την παροχή δεδομένων κλίσης σε δύο άξονες απλά με απευθείας ανάγνωση από τα μητρώα του.

β. Να γίνει μέτρηση της απόκλισης του αντικειμένου από την «όρθια» θέση. Προφανώς, καθώς η απόκλιση αυτή αυξάνεται, θα προκαλέσει μετατόπιση του κέντρου βάρους έξω από τα όρια της βάσης του αντικειμένου, προκαλώντας την ανατροπή του .

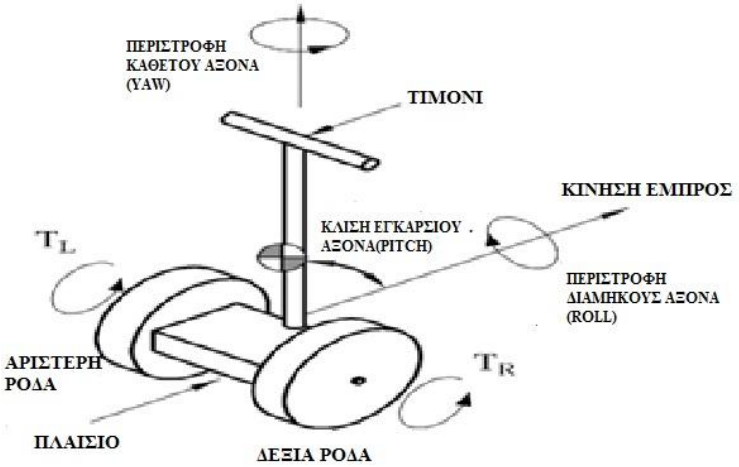


Εάν το κέντρο βάρους του σώματος βρεθεί έξω από τη γραμμή βάσης, το σώμα θα ανατραπεί.

γ. Για να αποφευχθεί η ανατροπή, ανάλογα με την απόκλιση που παρατηρείται, μπορούμε να μετακινήσουμε τη γραμμή βάσης γρήγορα ακριβώς κάτω από τη νέα θέση του κέντρου βάρους. Στους ανθρώπους αυτό γίνεται καθώς περπατάμε με την κίνηση των ποδιών μας προς τα εμπρός για να αποφύγουμε την πτώση προς τα εμπρός. Σε ένα τροχοφόρο όχημα

εξισορρόπησης, θα μπορούσαμε μετρώντας την απόκλιση από την «όρθια» θέση και με τη χρήση των κινητήρων να φέρουμε τους τροχούς άμεσα κάτω από αυτή τη θέση.

Έτσι, για την επίτευξη ισορροπίας, πρώτο βήμα είναι η εξαγωγή της γωνιακής απόκλισης από την «όρθια» θέση. Η γωνιακή απόκλιση για κάθε ένα από τους τρεις άξονες x,y,z ονομάζεται PITCH(κλίση), YAW (εκτροπή), και ROLL(κύλιση), αντίστοιχα.



Η γωνιακή απόκλιση για κάθε ένα από τους τρεις άξονες x,y,z

Το γυροσκόπιο – επιταχυνσιόμετρο MPU6050

IMU (Μονάδα αδρανειακής μέτρησης). Πρόκειται για μια συσκευή κατάλληλη για τη μέτρηση της δύναμης (επιτάχυνση) και της ταχύτητας. Γενικά αποτελείται από ένα επιταχυνσιόμετρο και ένα γυροσκόπιο. Ως εκ τούτου μια IMU δεν μετρά γωνίες. Τουλάχιστον όχι άμεσα, απαιτεί ορισμένους υπολογισμούς προκειμένου να λάβουμε τις γωνίες. Το MPU-6050 είναι ένα 6DOF IMU (διαβάζει "6 βαθμοί ελευθερίας»). Αυτό σημαίνει ότι διαθέτει ένα επιταχυνσιόμετρο και ένα γυροσκόπιο, σε κάθε ένα από τους 3 άξονες (2 x 3 = 6DOF). Υπάρχουν επίσης IMU των 9DOF, που διαθέτουν επιπλέον ένα μαγνητόμετρο και στους δύο άξονες, αλλά και 5DOF, που σε αυτήν την περίπτωση το γυροσκόπιο μετρά μόνο δύο άξονες.

Γυροσκόπιο – Επιταχυνσιόμετρο 6 αξόνων 6 MPU6050.

-Το MPU-6050 λειτουργεί με 3,3 V, αν και μερικές εκδόσεις έχουν ένα ρυθμιστή που επιτρέπει την σύνδεσή της με 5V.
-Το MPU-6050 χρησιμοποιεί το πρωτόκολλο επικοινωνίας I2C, θα αναφερθούμε παρακάτω για το πώς λειτουργεί αυτό το πρωτόκολλο, απλά συνδέοντας το SDA (δεδομένα) και SCL (ρολόι) μεταξύ της μονάδας και του μικροελεγκτή.

Επιταχυνσιόμετρο.

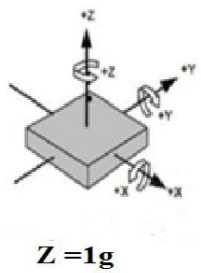
Το επιταχυνσιόμετρο μετρά την επιτάχυνση. Μπορούμε δηλαδή να λάβουμε την επιτάχυνση των τριών διαστάσεων του 3D χώρου μας: X, Y και Z,. Αν κινήσουμε για παράδειγμα, την IMU προς τα πάνω, θα λάβουμε την επιτάχυνση στον άξονα Z, προς τα εμπρός και προς τα πίσω στον άξονα Y και από την μία πλευρά στην άλλη στον άξονα X.

Η επιτάχυνση της Γης είναι περίπου 9,8 m / s², κάθετα προς το έδαφος. Ως εκ τούτου, η IMU ανιχνεύει επίσης την επιτάχυνση της επίγειας βαρύτητας. Οι αξίες που παίρνουμε από το IMU δεν θα είναι σε μονάδες "g".

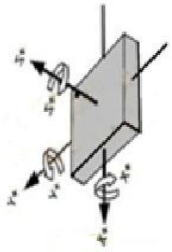
Αυτή η συσκευή λειτουργεί με μητρώα 8 bits. Κάθε τιμή επιτάχυνσης αποθηκεύεται σε δύο μητρώα, χαμηλών και υψηλών bits, και ως εκ' τούτου παρέχει 16 bits δεδομένων. Με τα δεδομένα αυτά των 16 bits, η MPU-6050 παρέχει 2^{16} = 65536 διακριτές τιμές επιτάχυνσης θετικού ή αρνητικού πρόσημου. Βάση του εγχειρίδιου της συσκευής «1g» ισούται με 16384. Έτσι, αν διαβάσουμε την τιμή της επιτάχυνσης και διαιρέσουμε αυτή την τιμή με 16384 παίρνουμε την επιτάχυνση σε μονάδες "g".

Χάρη στην επίγεια βαρύτητα μπορούμε να χρησιμοποιήσετε τις αναγνώσεις του επιταχυνσιόμετρου ώστε να υπολογίσουμε τη γωνία κλίσης σε σχέση με τον άξονα X ή τον Y.

Ας υποθέσουμε ότι το IMU είναι απόλυτα ευθυγραμμισμένη με το δάπεδο. Όπως μπορούμε να δούμε στην παρακάτω εικόνα, ο άξονας Z θα είναι 1g = 9.8, και οι άλλοι δύο άξονες θα είναι 0. Αν υποθέσουμε ότι περιστρέφουμε την IMU κατά 90 μοίρες, τότε ο άξονας X θα είναι κάθετος προς το έδαφος, και ως εκ τούτου, θα σηματοδοτήσει την επιτάχυνση 1g της βαρύτητας.



Z =1g



X = -1g

Βάση του ότι η βαρύτητα είναι 9,8 m / s², και γνωρίζοντας τις μετρήσεις στους τις τρεις άξονες του επιταχυνσιόμετρου, υπολογίζουμε τη γωνία κλίσης της IMU. Οι τύποι για τον υπολογισμό των γωνιών είναι:

$$\boldsymbol{AngleY = atan\left(\frac{X}{\sqrt{Y^2 + Z^2}}\right)}$$

$$\boldsymbol{AngleX = atan\left(\frac{Y}{\sqrt{X^2 + Z^2}}\right)}$$

Δεδομένου ότι η γωνία υπολογίζεται από τη βαρύτητα, δεν είναι δυνατόν να υπολογιστεί η γωνία Z με αυτόν τον τύπο ή με οποιοδήποτε άλλο. Για να γίνει αυτό θα πρέπει να έχουμε ένα άλλο στοιχείο: το μαγνητόμετρο, το οποίο είναι ένα είδος ψηφιακή πυξίδα. Το MPU-6050 δεν διαθέτει μαγνητόμετρο, και ως εκ τούτου δεν μπορεί να υπολογίσει με ακρίβεια τη γωνία Z. Ωστόσο, για τη συντριπτική πλειοψηφία των εφαρμογών απαιτούνται μόνο οι άξονες X και Y.

Γυροσκόπιο

Τα ηλεκτρικά γυροσκόπια στην πρώτη τους εμφάνιση ήταν ογκώδη αντικείμενα που η αξία τους ήταν μεγαλύτερη και από τον στρατιωτικό προϋπολογισμό ενός κράτους. Αργότερα, κατά τη διάρκεια του Δευτέρου Παγκοσμίου Πολέμου χρησιμοποιήθηκαν για τον έλεγχο της κατεύθυνσης πυραύλων και τορπιλών. Ευτυχώς, χάρη στην ψηφιακή επανάσταση και τη σμίκρυνση των κυκλωμάτων, στις μέρες μας αυτά είναι εύκολα να βρεθούν και φυσικά η τιμή τους είναι μηδαμινή.

Σημαντικό χαρακτηριστικό ενός γυροσκοπίου και κριτήριο επιλογής του για την εφαρμογή για την οποία προορίζεται, είναι οι επιλογές που προσφέρει όσων αφορά την ανάλυση του. Αυτή εκφράζεται σε μοίρες ή ακτίνια ανά δευτερόλεπτο, που αντιστοιχεί στο πόσο γρήγορη μεταβολή μπορεί το αισθητήριο να αντιληφθεί.

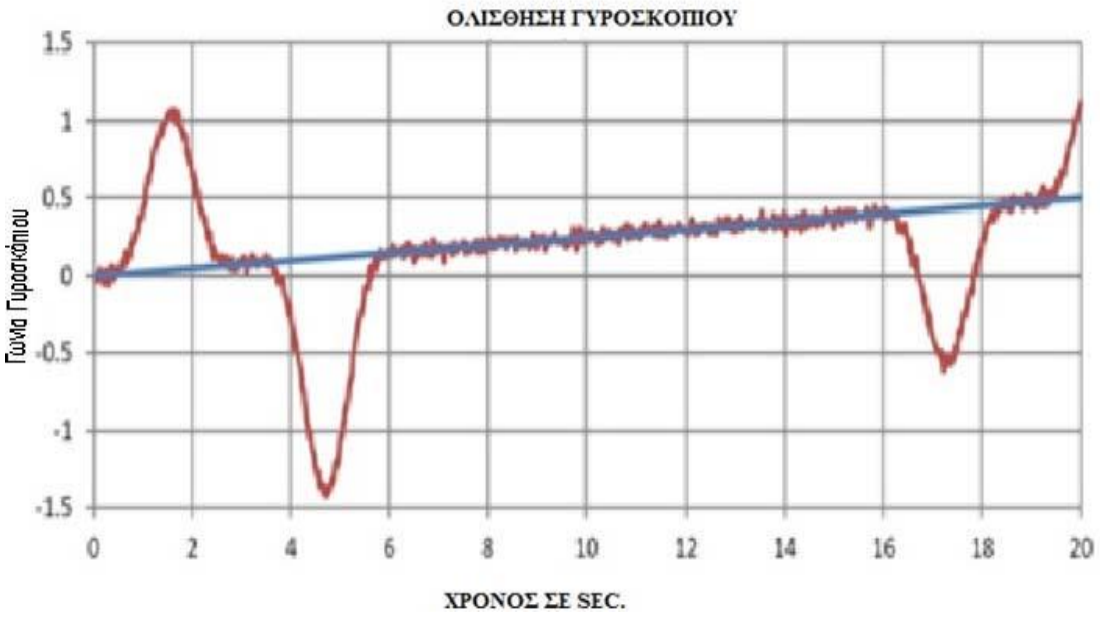
Συνήθως, οι προτιμώμενες τιμές είναι μεταξύ ±100◦/s ως ±300◦/s. Το MPU-6050 που χρησιμοποιείται στο όχημα έχει τέσσερις ρυθμίσεις ευαισθησίας που μπορεί να επιλέξει ο χρήστης: ±250◦/s, ±500◦/s,±1000◦/s και ±2000◦/s. Το όχημα της παρούσας εργασίας χρησιμοποιεί την πρώτη επιλογή ευαισθησίας του αισθητηρίου.

Το πλεονέκτημα των γυροσκοπίων

Το μεγαλύτερο πλεονέκτημα της IMU είναι ότι εξαλείφει την ανάγκη για την εκτέλεση πολύπλοκων και έντασης πόρων υπολογισμούς από την πλευρά του Arduino.

Τα μειονεκτήματα των γυροσκοπίων

Το κύριο μειονέκτημα των γυροσκοπίων, είναι το φαινόμενο της ολίσθησης ή drift. Το φαινόμενο αυτό, αποτρέπει την αποκλειστική χρήση γυροσκοπίων για την ευστάθεια του οχήματος, καθώς μετά από κάποιο χρονικό διάστημα, η απόκλιση των μετρήσεων θα μεταβάλλει το αρχικά θεωρημένο σημείο μηδέν, με αποτέλεσμα να μην υπάρχει το απαραίτητο set point για τη διατήρηση της ευστάθειας. Αυτό φαίνεται σύμφωνα με εργαστηριακές μετρήσεις που πραγματοποιούνται και ενώ η τιμή εξόδου του αισθητηρίου θα έπρεπε να παραμένει μηδενική κατά την κατάσταση ακινησίας, παρατηρείται αργή, σταδιακή μεταβολή στο χρόνο.



Γράφημα ολίσθησης γυροσκόπιου

Συμπληρώνοντας τις μετρήσεις με αυτές ενός επιταχυνσιόμετρου και τα κατάλληλα φίλτρα, μπορούμε να μειώσουμε κατά πολύ αυτή την ολίσθηση.

Ένα ακόμη μειονέκτημα που παρουσιάζουν τα γυροσκόπια είναι η ευαισθησία τους στις δονήσεις. Κάτι τέτοιο μπορεί να αποτελέσει πρόβλημα στο όχημά μας, καθώς η παραμικρή ανωμαλία του οδοστρώματος που δημιουργεί δονήσεις στο σώμα του οχήματος μπορεί να επηρεάσει ανεπιθύμητα τα δεδομένα εξόδου του γυροσκόπιου.

Γνωρίζοντας την αρχική γωνία της IMU, μπορούμε να προσθέσουμε σε αυτήν την αξία που στέλνει το γυροσκόπιο και έτσι να υπολογίσουμε τη νέα γωνία σε κάθε στιγμή.
Ας υποθέσουμε ότι έχουμε ξεκινήσει την IMU στις 0 μοίρες. Αν μας δίνεται μια μέτρηση ανά δευτερόλεπτο, και έχουμε σήματα στον άξονα X, θα μπορούμε να υπολογίσουμε τη γωνία με αυτό το απλό τύπο:

$$\boldsymbol{AngleY = PreviousAngleY + GyroDataY \times elapseTime}$$

$$\boldsymbol{AngleX = PreviousAngleX + GyroDataX \times elepseTime}$$

ElapsedTime είναι ο χρόνος που μεσολαβεί κάθε φορά που ο τύπος αυτός υπολογίζεται στο εσωτερικό του βρόχου, PreviousAngle είναι η γωνία που υπολογίστηκε την τελευταία φορά που ο τύπος αυτός κλήθηκε και GyroData είναι η ανάγνωση της γωνίας Y ή X του γυροσκοπίου . Το ίδιο συμβαίνει και με όλους τους άξονες. Μόνο ο άξονας Z συνήθως αγνοείται, αφού καθώς δεν μπορούμε να υπολογίσουμε τη γωνία Z με το επιταχυνσιόμετρο, δεν μπορούμε να εφαρμόσουμε τον παραπάνω τύπο για τον Z-άξονα.

Οι μετρήσεις επιταχυνσιόμετρου και γυροσκόπιου εξηγούνται στο δελτίο MPU-6050 στις περιγραφές των μητρώων GYRO_CONFIG και ACCEL_CONFIG. Η κλίμακα του καθενός εξαρτάται από τις ρυθμίσεις ευαισθησίας που επιλέγονται , οι οποίες μπορεί να είναι από +/- 2, 4, 8, ή 16 g για το επιταχυνσιόμετρο και από +/- 250, 500, 1000, ή 2000 deg /sec για το γυροσκόπιο. Το επιταχυνσιόμετρο παράγει δεδομένα σε μονάδες επιτάχυνσης (m/s^2), και το γυροσκόπιο παράγει δεδομένα σε μονάδες γωνιακής ταχύτητας.

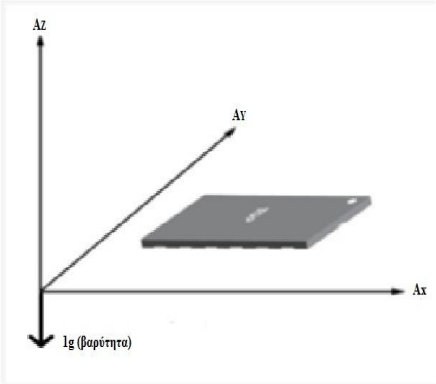
Η κλίμακα εξόδου για κάθε ρύθμιση είναι (-32768, 32767) για κάθε έναν από τους έξι άξονες. Η προεπιλεγμένη ρύθμιση στην βιβλιοθήκη I2Cdev.h είναι +/- 2g για την accel και +/- 250 deg / sec για το γυροσκόπιο. Αν η συσκευή είναι σε απόλυτο επίπεδο και δεν κινείται, τότε:

- Στους άξονες X / Y accel πρέπει να διαβάσουμε 0.
- Στον άξονα Z accel πρέπει να διαβάσουμε 1g, η οποία είναι 16.384 σε ευαισθησία των 2g.
- Στους X / Y / Z άξονες του γυροσκόπιου πρέπει να διαβάσουμε 0.

Στην πραγματικότητα, οι τιμές για τους άξονες X/Y accel δεν θα είναι ακριβώς 0, δεδομένου ότι είναι δύσκολο να είναι απόλυτα επίπεδο και υπάρχει κάποιος θόρυβος / λάθος. Το ίδιο θα συμβεί και για τους άξονες X/Y του γυροσκόπιου, για τον ίδιο λόγο (θόρυβος / λάθος).

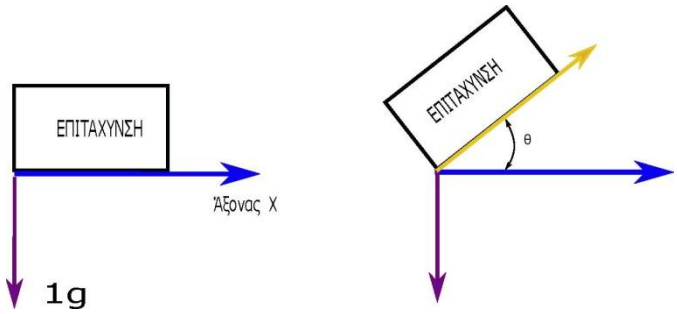
Υπολογισμός γωνίας κλίσης.

Μία από τις πιο κοινές χρήσεις του επιταχυνσιόμετρου είναι η μέτρηση της κλίσης σε έναν συγκεκριμένο άξονα. Το επιταχυνσιόμετρο σε μια επίπεδη επιφάνεια θα παράγει 1g σε έναν από τους άξονες του (πιθανότατα Z). Η έξοδος του επιταχυνσιόμετρου δεν είναι γραμμική, αλλά μάλλον ημιτονοειδές, οπότε πρέπει να μετατρέψετε τις δυνάμεις g ανάλογα με την κλίση σε μοίρες.



Μέτρηση κλίσης με έναν άξονα

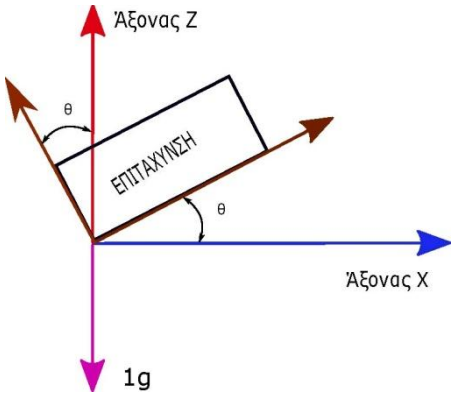
Ο απλούστερος τρόπος μέτρησης της κλίσης είναι να χρησιμοποιήσουμε μόνο έναν άξονα. Βασικά το αντίστροφο της συνάρτησης sine θα σας δώσει τη γωνία.



Έτσι έχουμε, **θ = sin-1 (x)**. Ωστόσο, λόγω της φύσης του ημιτονοειδούς κύματος μπορούμε να μετρήσουμε αξιόπιστα κλίσεις από 45 ° έως -45 °. Πέρα από αυτά τα σημεία η ευαισθησία των μετρήσεων μειώνεται σημαντικά.

Μέτρηση της κλίσης με δύο άξονες

Μια ελαφρώς πιο αξιόπιστη μέθοδος για τον υπολογισμό της κλίσης είναι η χρήση δύο αξόνων. Μπορούμε να μετρήσουμε από 90 ° έως -90 °, χωρίς απώλεια ευαισθησίας.



έτσι θα έχουμε

$$\tan\theta = \frac{x}{z} \quad , \quad \theta = arctan\left(\frac{x}{z}\right)$$

Η χρήση δύο αξόνων βελτιώνει σημαντικά την ακρίβεια της μέτρησης της γωνίας. Ωστόσο, εάν το επιταχυνσιόμετρό μας είναι ελαφρώς στρεφόμενο προς την κατεύθυνση του άξονα Y, οι μετρήσεις μας θα είναι και πάλι ανακριβείς, καθώς μερικές από τις συνιστώσες του διανύσματος από τον άξονα Z θα χαθούν στον άξονα Y.

Μέτρηση της κλίσης με τρεις άξονες

Για να έχουμε την καλύτερη ακρίβεια κατά τη μέτρηση κλίσης, πρέπει να χρησιμοποιήσουμε και τους τρεις άξονες για να καθορίσουμε τη γωνία.

$$\theta = \arctan\left(\frac{x}{\sqrt{z^2 + y^2}}\right)$$

Με την παραπάνω εξίσωση υπολογίζουμε τη γωνία μεταξύ του διανύσματος βαρύτητας και του άξονα X. Ανάλογα με τον τρόπο που τοποθετούμε το επιταχυνσιόμετρο στο όχημά μας μπορεί να είναι είτε pitch είτε roll.

$$Pitch = arctan\left(\frac{x}{\sqrt{z^2+y^2}}\right) \qquad // \ \textit{γωνία επιτάχυνσης στον Y}$$

$$Roll = arctan\left(\frac{y}{\sqrt{z^2+x^2}}\right) \qquad // \ \textit{γωνία επιτάχυνσης στον X}$$

Ο κώδικας θα γραφόταν ως εξής:

```
double pitch = (atan(AcX,sqrt(AcY * AcY + AcZ * AcZ));
 double roll = (atan(AcY,sqrt(AcX * AcX + AcZ * AcZ));
```

Είναι προφανές πόσο σύνθετος και περίπλοκος είναι αυτός τρόπος εξαγωγής της γωνίας πράγμα που θα επιβαρύνει αρκετά τον χρόνο επεξεργασίας σε κάθε κύκλο του προγράμματος ιδιαίτερα σημαντικό για την κατασκευή μας. Το ίδιο αποτέλεσμα θα λάβουμε χρησιμοποιώντας τη λειτουργία *atan2,* Η συνάρτηση atan2 επιστρέφει τη γωνία σε ακτίνια, οπότε βάση μαθηματικών 1 ακτίνο = 180 / π.

```
#define degconvert 57.2957786          //  180 / 3.141592654  μετατροπή rad σε μοίρες
   double roll = atan2(AcY, AcZ)*degconvert;          //accel angle Y
   double pitch = atan2(-AcX, AcZ)*degconvert;          //accel angle X
```

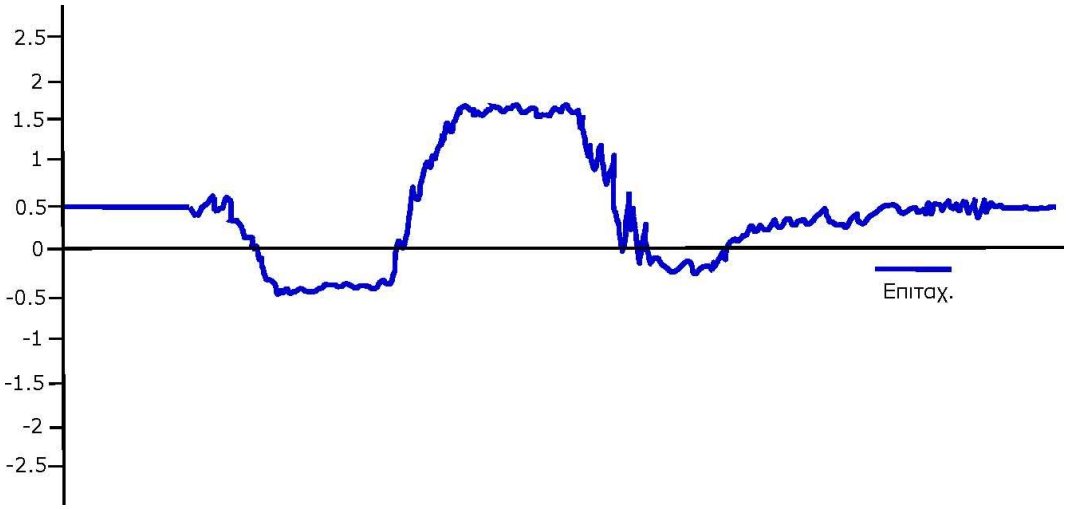
Θα χρησιμοποιήσουμε αυτά τα δεδομένα για να διορθώσουμε τυχόν αθροιστικά σφάλματα στον προσανατολισμό που αναπτύσσεται το γυροσκόπιο.
Οι τιμές που εξάγει το γυροσκόπιο είμαι μονάδες γωνιακής ταχύτητας. Για να μετατρέψουμε αυτά τα πρωτογενή δεδομένα σε deg / δευτερόλεπτο, διαιρούμε με 131 σύμφωνα με τις οδηγίες του κατασκευαστή. Διαιρούμε με ένα *double* "131.0" αντί για το int 131.

```
double gyroXrate = GyX/131.0;
double gyroYrate = GyY/131.0;
```

Τα επιταχυνσιόμετρα είναι πολύ ευαίσθητα και έτσι είναι αρκετά επιρρεπής σε κραδασμούς. Εάν τα χρησιμοποιήσουμε, σε ένα σύστημα με μοτέρ ή άλλες πηγές κραδασμών είναι απαραίτητη η προσθήκη κάποιας μορφής απόσβεσης (αφρός).
Το Arduino, λαμβάνει πολλαπλές αναγνώσεις από το επιταχυνσιόμετρο και ο μέσος όρος τους φιλτράρεται από κάθε θόρυβο.

Θόρυβος – Σφάλματα – Φίλτρα.

Οι συσκευές IMU είναι μαγικά αντικείμενα καθώς με λίγη της τριγωνομετρίας μπορούν να μας δώσουν την γωνία με απόλυτη ακρίβεια. Υπάρχουν όμως δύο πολύ σημαντικά προβλήματα: ο θόρυβος και τα λάθη. Ο θόρυβος προέρχεται από τις ίδιες τις ηλεκτρονικές συσκευές. Το επιταχυνσιόμετρο είναι σε θέση να μετρήσει οποιαδήποτε γωνία, ωστόσο οι αναγνώσεις του είναι θορυβώδεις και να έχουν ένα ορισμένο περιθώριο λάθους. Αν θελήσουμε να σχεδιάσουμε μια γραφική παράσταση των μετρήσεων ενός επιταχυνσιόμετρου, ως συνάρτηση του χρόνου, τότε το αποτέλεσμα θα είναι κάπως έτσι:



Γραφική παράσταση μετρήσεων ενός επιταχυνσιόμετρου

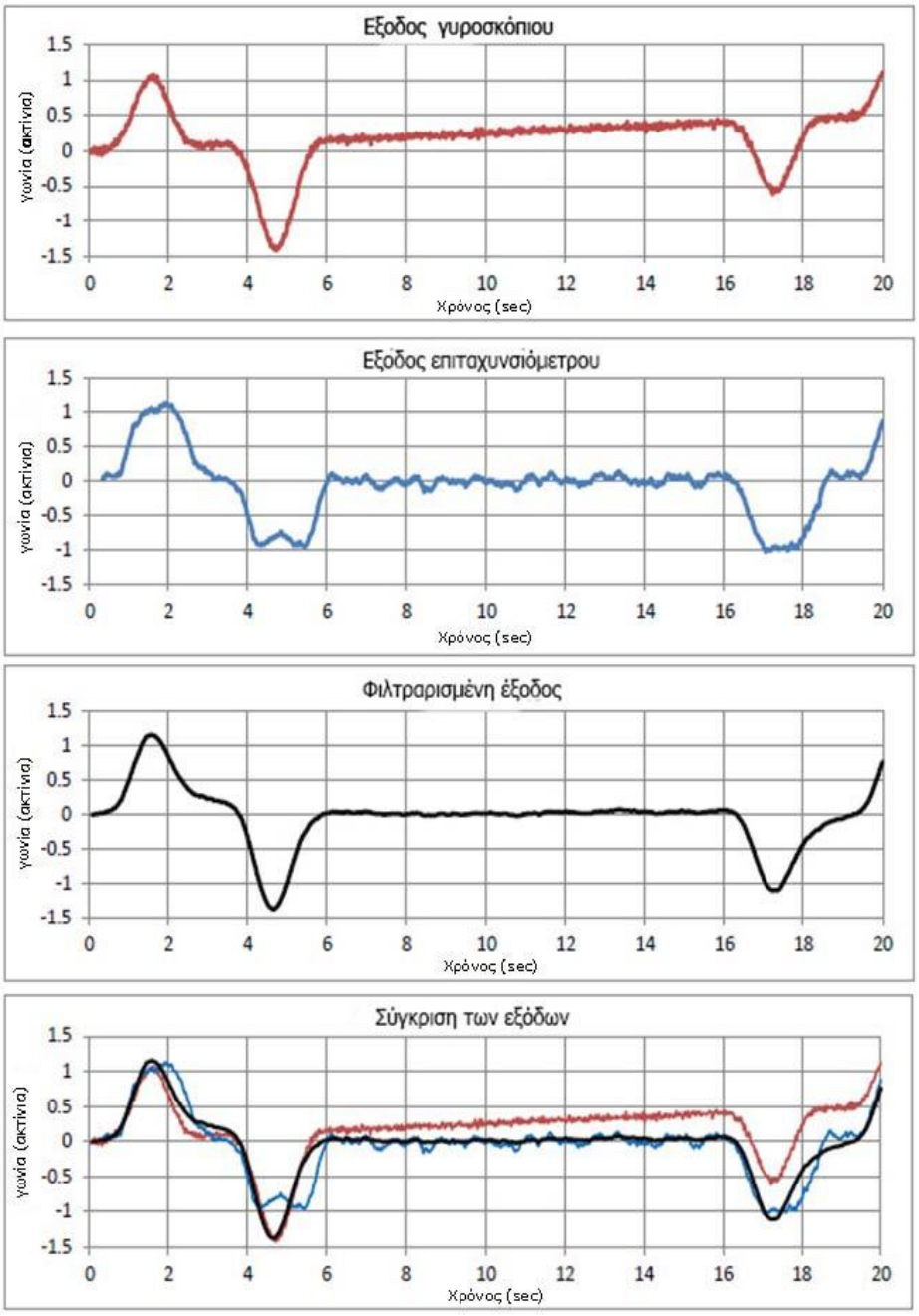
Φαίνεται ξεκάθαρα πως τα δεδομένα από το επιταχυνσιόμετρο είναι θορυβώδεις . Ακόμη και με ένα φίλτρο χαμηλής διέλευσης θα εξακολουθεί να υπάρχουν αιχμές και θόρυβος. Αν προσθέσουμε και τη γωνία που λαμβάνουμε από το γυροσκόπιο θα μια ακόμη πιο θορυβώδη κυματομορφή. Αν παίρνουμε τη γωνία χρησιμοποιώντας τύπους Euler με αυτά τα δεδομένα επιτάχυνσης η γωνία δεν θα είναι ακριβής σε κάθε στιγμή. Για αυτό το λόγο θα πρέπει να χρησιμοποιήσουμε φίλτρα και να κάνουμε ένα συνδυασμό μεταξύ των δεδομένων του γυροσκόπιο και του επιταχυνσιόμετρου .:

Η ιδέα είναι πολύ απλή: θα πρέπει να εξαλείψουμε το θόρυβο, την ολίσθηση (drift) και να αναγκάσουμε το επιταχυνσιόμετρο να μην αλλάξει τη γωνία του όταν ανιχνεύει οποιαδήποτε δύναμη εκτός από τη βαρύτητα. Υπάρχουν διάφοροι αλγόριθμοι, που ονομάζονται φίλτρα, και εκτελούν αυτή τη διαδικασία. Ένα από τα καλύτερα είναι το περίφημο ***Kalman Filter***. Χρησιμοποιείται στα αεροσκάφη, πυραύλους και σε γεωστατικούς δορυφόρους. Θεωρείται ένα από τα μεγαλύτερα ευρήματα του περασμένου αιώνα, και δικαίως. Είναι σε θέση να υπολογίσει το σφάλμα κάθε μέτρησης από τις προηγούμενες μετρήσεις, και αφαιρώντας το να δώσει την πραγματική τιμή της γωνίας. Με λίγα λόγια είναι ένας αλγόριθμος που ‘’μαθαίνει’’ σε κάθε επανάληψη.

Ωστόσο, υπάρχουν δύο βασικά προβλήματα

- Έχει ένα κάπως υψηλό κόστος επεξεργασίας (διαδικασία)
- Είναι πολύ περίπλοκο στην κατανόησή του.

Έτσι, έχουμε άλλα φίλτρα στη διάθεσή μας. Το ένα που πρόκειται να χρησιμοποιήσουμε είναι γνωστή ως Complementary Filter (συμπληρωματικό φίλτρο). Είναι ιδανικό για να εφαρμοστεί με Arduino, καθώς είναι εύκολο στη χρήση, με χαμηλό κόστος επεξεργασίας και με πολύ καλή ακρίβεια.



Οι έξοδοι γυροσκόπιου – επιταχυνσιόμετρου και εφαρμογή φίλτρου complementary

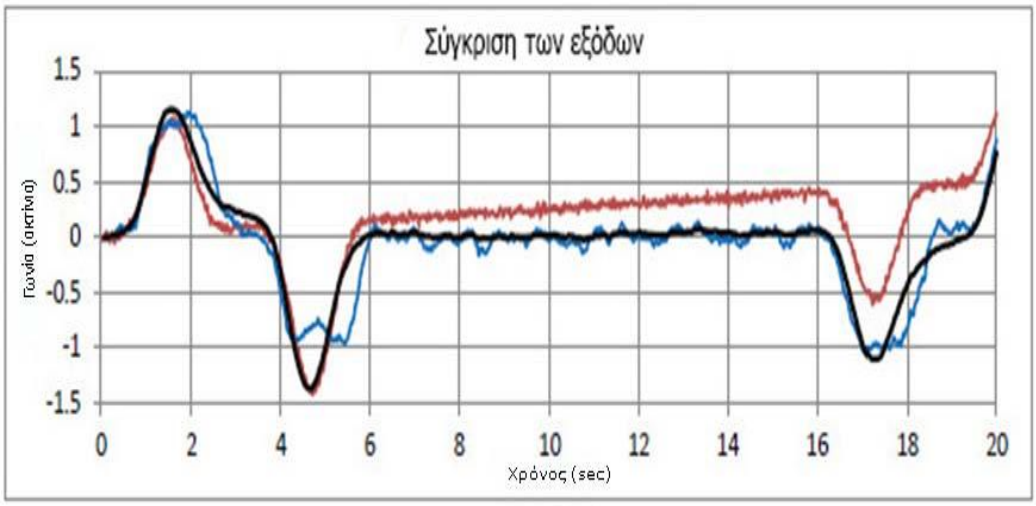
Ένα φίλτρο ***Complementary*** είναι στην πραγματικότητα μια ένωση δύο διαφορετικών φίλτρων: ενός High-pass υψιπερατού φίλτρου για το γυροσκόπιο και ενός Low-pass διέλευσης χαμηλών συχνοτήτων φίλτρου για το επιταχυνσιόμετρο. Το πρώτο αφήνει να περάσουν μόνο τιμές πάνω από ένα ορισμένο όριο, σε αντίθεση με το φίλτρο Low-pass, το οποίο επιτρέπει μόνο τιμές κάτω από ένα ορισμένο όριο. Ο τύπος που προκύπτει από το συνδυασμό (συμπληρώνοντας, εξ ου και το όνομα), των δύο φίλτρων είναι:

$$\boldsymbol{TotalAngleX = 0.98 \times (TotalAngleX + GyroDataY \times elapseTime) + 0.02 \times atan\left(\frac{Y}{\sqrt{X^2 + Z^2}}\right)}$$

$$\boldsymbol{TotalAngleY = 0.98 \times (TotalAngleY + GyroDataX \times elapseTime) + 0.02 \times atan\left(\frac{X}{\sqrt{Y^2 + X^2}}\right)}$$

Το ***Complementary Filter*** προσθέτει το 98% της γωνίας που λαμβάνεται με τα δεδομένα του γυροσκόπιου και 2% της γωνίας που λαμβάνεται με τα δεδομένα της επιτάχυνσης. Βλέπουμε ότι στη συνολική γωνία X προστίθενται τα δεδομένων gyro Y. Αυτό οφείλεται στο γεγονός ότι η γωνία κλίσης του άξονα X δίνεται από τον άξονα περιστροφής Y.

Μετά την εφαρμογή των φίλτρων θα πρέπει να δούμε κάτι σαν το παρακάτω γράφημα το οποίο είναι αρκετά κοντά στην πραγματικότητα.



έξοδος γυροσκόπιου

έξοδος επιταχυνσιόμετρου

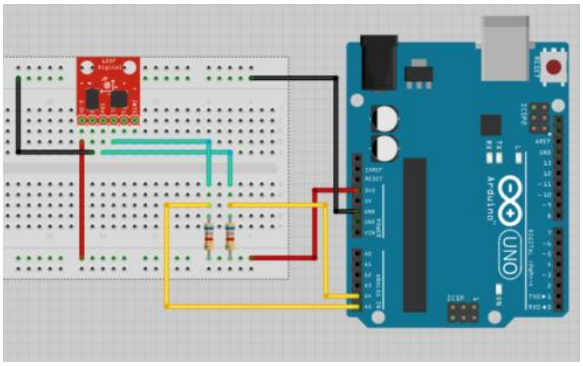
έξοδος με Complementary φίλτρο

Είναι ξεκάθαρο πως η μαύρη κυματομορφή (comp) αντιπροσωπεύει την πραγματική αναπαράσταση της γωνίας, και είναι πολύ ''καθαρότερη'' από τα ανεπεξέργαστα δεδομένα.

Το μόνο που χρειάζεται να κάνουμε είναι να χρησιμοποιήσουμε αυτούς τους τύπους στο κώδικα Arduino για να αποκτήσουμε τη γωνία.

Η σύνδεση της MPU6050 με το ARDUINO είναι πολύ απλή.

MPU Vcc -----> Arduino 5V
MPU Gnd ----> Arduino Gnd
MPU SCL -----> Arduino A5
MPU SDA ----> Arduino A4



Συνδεσμολογία της MPU6050 στην ARDUINO με χρήση Fritzing.

Αυτές οι συνδέσεις ισχύουν για Arduino Uno και Arduino Nano, των οποίων τα pins SDA και SCL είναι A4 και A5 αντίστοιχα. Εάν χρησιμοποιήσουμε μια άλλη έκδοση Arduino μπορεί τα pins να είναι σε διαφορετική θέση.

Το MPU-6050 θα δώσει τις αξίες ανεπεξέργαστες (raw value) που θα πρέπει στη συνέχεια να επεξεργαστούν προκειμένου να ληφθούν χρησιμοποιήσιμες τιμές.

Ο κώδικας επικοινωνίας MPU6050 - Arduino.

Ας δούμε πως εφαρμόζονται τα παραπάνω στον κώδικά μας.
Πρώτα απ 'όλα θα περιλαμβάνουν τις βιβλιοθήκες MPU6050_6Axis_MotionApps20.h και Wire.h.

```
#include <MPU6050_6Axis_MotionApps20.h>
#include <Wire.h>
```

Χρειαζόμαστε την βιβλιοθήκη MPU6050_6Axis_MotionApps20.h για να λάβουμε τις τιμές από το MPU6050. Η βιβλιοθήκη Wire.h θα δημιουργήσει μια επικοινωνία I2C με τη μονάδα MPU6050.

Ζητάμε από την MPU-6050 να μας δώσει 14 bits δεδομένων τα οποία θα διαβαστούν από τα αντίστοιχα μητρώα (registers) και θα εξισωθούν με τις ανάλογες μεταβλητές *AcX,AcY* … κτλ. Τα δεδομένα διαιρούνται σε 2 * 8bits μητρώα.

```
double AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers

AcX=Wire.read()<<8|Wire.read();  //  0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read();  //  0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read();  //  0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
Tmp=Wire.read()<<8|Wire.read();  //  0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
GyX=Wire.read()<<8|Wire.read();  //  0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
```

```
GyY=Wire.read()<<8|Wire.read();  //  0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
GyZ=Wire.read()<<8|Wire.read();  //  0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
```

Οι δύο παρακάτω γραμμές υπολογίζουν τον προσανατολισμό του επιταχυνσιόμετρου σε σχέση με την γη και μετατρέπουν την εξόδου του από ακτίνια σε μοίρες.
Θα χρησιμοποιήσουμε αυτά τα δεδομένα για τη διόρθωση τυχόν σωρευτικών λαθών στον προσανατολισμό που αναπτύσσει το γυροσκόπιο.

```
#define degconvert 57.2957786   //57.2957786 = 180 / 3.141592654  Η λειτουργία  atan

                                // είναι σε ακτίνια

double roll = atan2(AcY, AcZ)*degconvert;
double pitch = atan2(-AcX, AcZ)*degconvert;
```

Για την εξαγωγή αποτελέσματος φιλτραρισμένων γωνιών με την χρήση του Complementary Filter ο κώδικας διαμορφώνεται ως εξής:

```
compAngleX = 0.98 * (compAngleX + gyroXrate * dt) + 0.02 * roll;
compAngleY = 0.98 * (compAngleY + gyroYrate * dt) + 0.02 * pitch;
```

όπου:

- *compAngleX* = αποτέλεσμα φιλτραρισμένης γωνίας X
- *compAngleY* = αποτέλεσμα φιλτραρισμένης γωνίας Y
  0.98 = ποσοστό συμμετοχής γυροσκόπιου.
  0.02 = ποσοστό συμμετοχής επιταχυνσιόμετρου.
- *gyroXrate = GyX/131.0* Για την γωνία του άξονα X του γυροσκόπιου σε μοίρες /sec διαιρούμε την ανεπεξέργαστη τιμή GyX με 131.0 σύμφωνα με τον κατασκευαστή.
- *gyroYrate* = GyY/131.0 Για την γωνία του άξονα Y του γυροσκόπιου σε μοίρες/sec διαιρούμε την ανεπεξέργαστη τιμή GyY με 131.0 σύμφωνα με τον κατασκευαστή.
- *dt = (double)(micros() - timer) / 1000000 =* τρέχων χρόνος – διάρκεια σε sec

3.3.3 Ελεγκτής κινητήρων.

Εισαγωγή

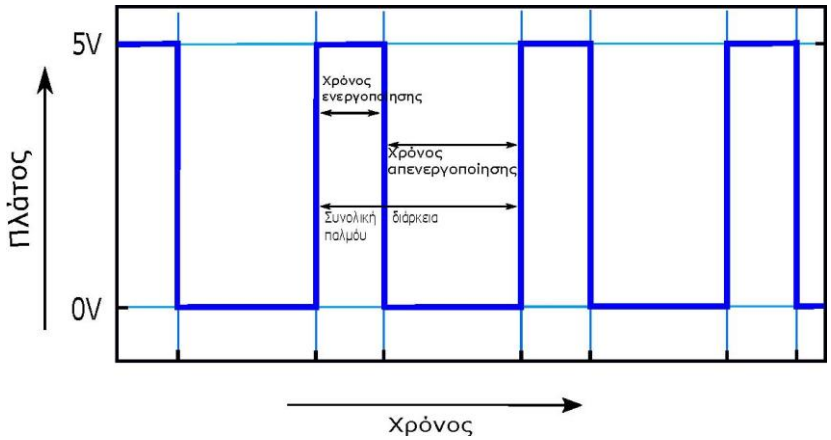
Οι κινητήρες συνεχούς ρεύματος χρησιμοποιούνται όλο και περισσότερο για ένα ευρύ φάσμα εφαρμογών, συμπεριλαμβανομένων της ρομποτικής, φορητά ηλεκτρονικά είδη, αθλητικό εξοπλισμό, συσκευές, ιατρικό συσκευές, εφαρμογές στην αυτοκινητοβιομηχανία, ηλεκτρικά εργαλεία και πολλών άλλων. Επιλέγονται λόγο απλής και αξιόπιστης λειτουργίας καθώς και του χαμηλού κόστους τους.
Ανάλογα απλά και χαμηλού κόστους είναι και κυκλώματα ελέγχου ταχύτητας, κατεύθυνσης και φρένου.

Παλμός διαμορφούμενου πλάτους (Pulse Width Modulation) PWM

Ο παλμός με διαμόρφωση εύρους, ή PWM, είναι μια τεχνική για να πάρουμε ένα αναλογικό αποτελέσματα από ψηφιακά μέσα. Ο ψηφιακός έλεγχος χρησιμοποιείται για να δημιουργήσει μία τετραγωνική κυματομορφή, ένα σήμα μεταγωγής δηλαδή μεταξύ on και off. Αυτό το πρότυπο on-off μπορεί να προσομοιώσει τάσεις μεταξύ πλήρους on (5 Volts)

και off (0 Volts) αλλάζοντας το τμήμα του χρόνου που το σήμα είναι on (high) σε συνάρτηση του χρόνου που το σήμα είναι off (low). Η διάρκεια του χρόνου on ονομάζεται εύρος παλμού. Για να πάρουμε διαφορετικές αναλογικές τιμές, μπορούμε να αλλάξουμε, ή να διαμορφώσουμε, αυτό το εύρος παλμού.
Αν επαναλάβουμε αυτό το μοτίβο on-off αρκετά γρήγορα σε ένα LED για παράδειγμα, ελέγχουμε τη φωτεινότητα του LED και έχουμε το ίδιο αποτέλεσμα με αυτό που το σήμα είναι μια σταθερή τάση μεταξύ 0 και 5V.

Η μαθηματική εξήγηση δίνεται παρακάτω:



Στο παραπάνω σχήμα φαίνεται ένα τετραγωνικό σήμα.

$\boldsymbol{T_{ενεργοπ.}}$ είναι ο χρόνος κατά τον οποίο η έξοδος είναι HIGH και $\boldsymbol{T_{απενεργ.}}$. είναι ο χρόνος κατά τον οποίο η έξοδος είναι LOW. Κατά συνέπεια η χρονική διάρκεια του σήματος $\boldsymbol{T_{συνολ.διάρκεια}}$ είναι,

$$\boldsymbol{T_{συνολ.διάρκεια} = T_{ενεργοπ.} + T_{απενεργ.}}$$

Η διάρκεια παλμού (**D**uty cycle) του τετραγωνικού σήματος είναι:

$$\boldsymbol{D = \frac{T_{ενεργοπ.}}{T_{ενεργοπ.} + T_{απενεργ.}} = \frac{T_{ενεργοπ.}}{T_{συνολ.διάρκεια}}}$$

Η τάση εξόδου μεταβάλλεται ανάλογα με την διάρκεια παλμού (Duty cycle) και δίνετε από τον τύπο:

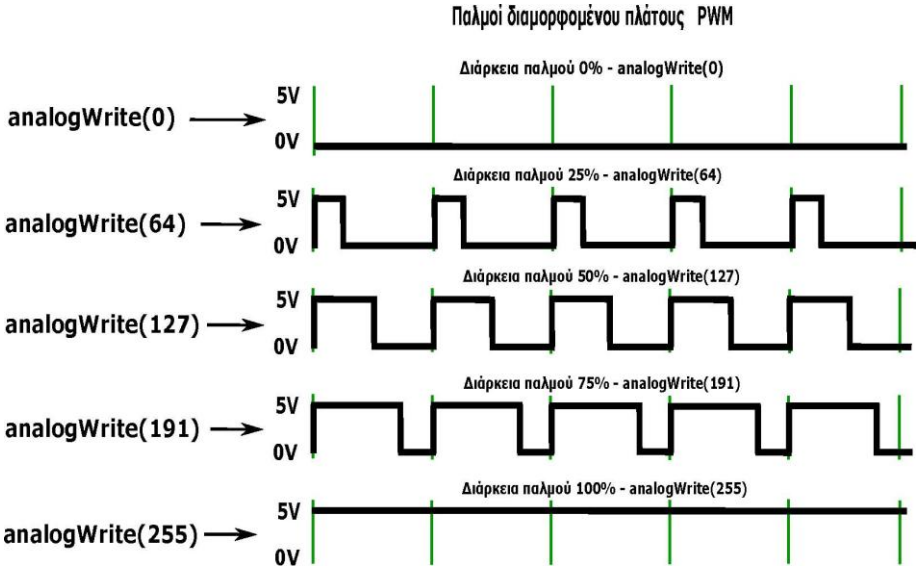
$$\boldsymbol{V_{εξοδ.} = D \times V_{εισοδ.} = \frac{T_{ενεργοπ.}}{T_{συνολ.διάρκεια}} \times V_{εισοδ.}}$$

Έτσι από την τελική εξίσωση της τάσης εξόδου βλέπουμε την άμεση μεταβολή της με τη μεταβολή της τιμής $\boldsymbol{T_{ενεργοπ.}}$
Αν ο $\boldsymbol{T_{ενεργοπ.}}$είναι 0, η $\boldsymbol{V_{εξοδ.}}$ είναι επίσης 0.
Αν ο $\boldsymbol{T_{ενεργοπ.}}$ είναι ίσος με τον $\boldsymbol{T_{συνολ.διάρκεια}}$, τότε και η $\boldsymbol{V_{εξοδ.}}$ θα είναι ίση με την $\boldsymbol{V_{εισοδ.}}$ ή αλλιώς μέγιστη.

Με τον τρόπο αυτό μπορούμε να ελέγχουμε την ταχύτητα του κινητήρα οδηγώντας τον με σύντομο τετραγωνικό παλμό. Ο τετραγωνικός παλμός on-off κυμαίνεται από 0 έως 100

τοις εκατό. Όσο μεγαλύτερη η διάρκειά του τόσο ο κινητήρας περιστρέφεται γρηγορότερα, και το αντίστροφο.

Στο παρακάτω γράφημα, οι πράσινες γραμμές αντιπροσωπεύουν ένα τακτικό χρόνο περιόδου. Αυτή η διάρκεια ή η περίοδος είναι το αντίστροφο της συχνότητας PWM. Με άλλα λόγια, με τη συχνότητα PWM του Arduino σε περίπου 500Hz, οι πράσινες γραμμές είναι 2 χιλιοστά του δευτερολέπτου κάθε μία.
Η εντολή *analogWrite ()* είναι σε μια κλίμακα από 0 έως 255, οπότε για παράδειγμα καλόντας *analogWrite (255)* θα πάρουμε κύκλο εργασίας 100% (πάντα on), και με analogWrite (127) ένας κύκλο 50%



Παλμοί διαμόρφωσης πλάτους (PWM) όπως εμφανίζονται σε μια ψηφιακή έξοδο.

Ελεγκτής κινητήρων συνεχούς ρεύματος.

Με τον όρο ελεγκτής κινητήρων αναφερόμαστε σε έναν ενισχυτή που έχει σχεδιαστεί για να ελέγχει την ταχύτητα και την κατεύθυνση ενός κινητήρα με ένα συγκεκριμένο σύνολο εντολών. Είναι ένα κύκλωμα που έχει σχεδιαστεί για να χρησιμοποιεί ένα σήμα εισόδου χαμηλής ισχύος και να παρέχει ένα σήμα εξόδου υψηλής ισχύος ώστε να εφαρμόζεται η κατάλληλη ταχύτητα και κατεύθυνση προς τον κινητήρα με τη χρήση PWM. Αυτές είναι συνήθως προκατασκευασμένες μονάδες που κοστίζουν περισσότερο αλλά απαιτούν λιγότερη δουλειά.

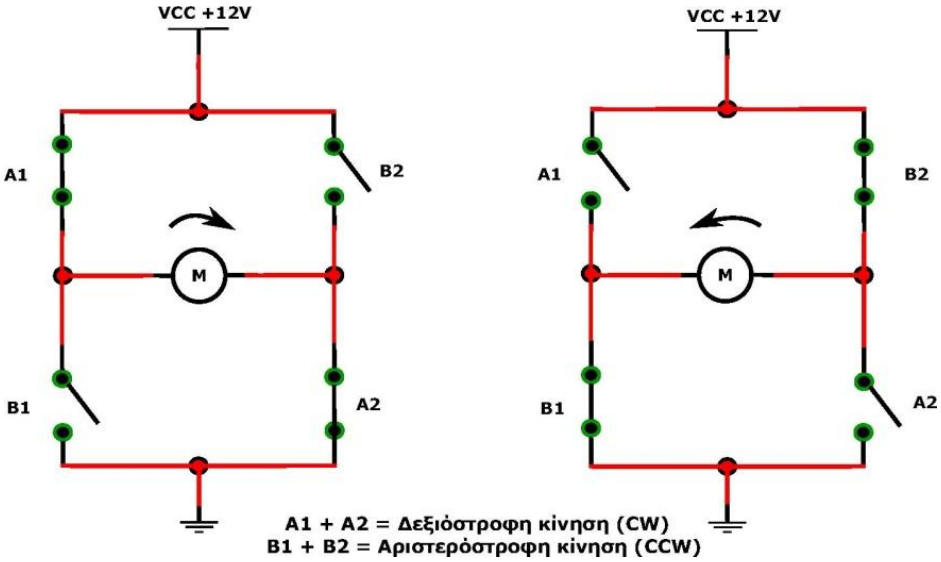
Η πλειοψηφία των ελεγκτών κινητήρων είναι κατασκευασμένοι για χρήση σε χόμπι αεροπλάνο, αυτοκίνητο και τον εξοπλισμό σκαφών, και χειρίζονται ένα σήμα εισόδου παλμού Servo. Υπάρχουν επίσης ελεγκτές κινητήρων που είναι προσανατολισμένοι προς χρήση σε ρομποτική, με μια ποικιλία από διαφορετικές επιλογές διασύνδεσης.

Ελεγκτής κινητήρων τύπου H-Bridge.

Η H-Bridge είναι η κυριότερη και πιο διαδεδομένη διάταξη ελέγχου ενός κινητήρα συνεχούς ρεύματος και με αυτή θα ελέγξουμε τους κινητήρες της παρούσας εργασίας.

Η H-Bridge είναι κυρίως μια διάταξη τεσσάρων διακοπτών, που ενεργοποιούνται με ειδικό τρόπο για να ελέγχουν την κατεύθυνση της ροής του ρεύματος μέσω του κινητήρα.

(Στους κινητήρες συνεχούς ρεύματος, η κατεύθυνση περιστροφής του οπλισμού του κινητήρα αλλάζει με την αλλαγή της κατεύθυνσης του ρεύματος που ρέει).
Αναφερόμενοι στους DC κινητήρες, είναι χρήσιμο να γνωρίζουμε ότι «*Η τρέχουσα ροή ρεύματος σε ένα κινητήρας είναι ανάλογη με τη ροπή εξόδου, ενώ η γωνιακή ταχύτητα (rpm) του άξονα είναι ανάλογη με την τάση σε όλες τις περιελίξεις του κινητήρα* ».



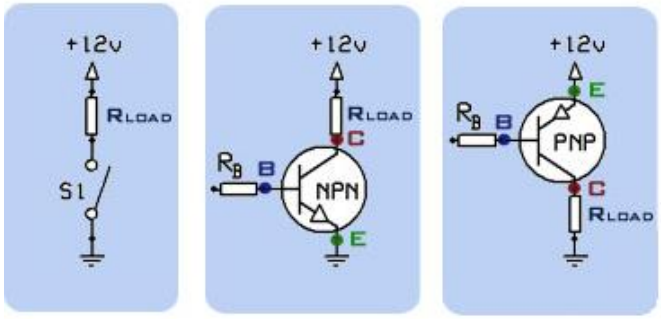
Απλοποιημένο σχεδιάγραμμα H-Bridge

Στο παραπάνω σχήμα, φαίνονται τα απλοποιημένα διαγράμματα που δείχνουν τη λειτουργία της διάταξης H-Bridge (παρατηρούμε ότι το σχήμα του διαγράμματος, μοιάζει με το γράμμα «Η», αυτό είναι το πώς αυτό το περίφημο κύκλωμα πήρε αυτό το όνομα!). Υπάρχουν δύο πιθανές διαδρομές για την τρέχουσα διάταξη.
Αυτή του πρώτου διαγράμματος , όπου το ρεύμα κατευθύνεται προς τον κινητήρα μέσω των διακοπτών A1 και A2, προκαλώντας τον κινητήρα για να περιστραφεί δεξιόστροφα, και αυτή του δεύτερου διαγράμματος, όπου το ρεύμα κατευθύνεται προς τον κινητήρα μέσω των διακοπτών B1και B2, προκαλώντας τον κινητήρα για να περιστραφεί αριστερόστροφα.

Η μόνη διαφορά μεταξύ αυτού του απλοποιημένου διαγράμματος H-Bridge και του πραγματικού H-Bridge είναι ότι οι διακόπτες αντικαθίστανται από τρανζίστορ, με σκοπό τον έλεγχο ηλεκτρονικά της ροής του ρεύματος στον κινητήρα, ως εκ τούτου, τον έλεγχο της ταχύτητα και την κατεύθυνση του κινητήρα από έναν μικροελεγκτή.

Το σχήμα που ακολουθεί, απλά δείχνει την έννοια της χρήσης ενός τρανζίστορ ως διακόπτη. Η μόνη διαφορά ανάμεσα σε ένα μηχανικό διακόπτη και έναν διακόπτη τρανζίστορ είναι ότι η ενεργοποίηση και η απενεργοποίηση ενός κανονικού διακόπτη γίνεται μηχανικά, ενώ ενός διακόπτη τρανζίστορ με τη χρήση μικρών ηλεκτρικών ρευμάτων που εφαρμόζεται στη βάση του (B), συνήθως μικρότερη από 20 mA. Για ένα τρανζίστορ NPN, όταν ένα μικρό ρεύμα ρέει στη βάση του τρανζίστορ, ένα μεγάλο ρεύμα θα ρέει από τον συλλέκτη προς τον εκπομπό.

Από την άλλη πλευρά, για ένα τρανζίστορ PNP, όταν ένα μικρό ρεύμα ρέει στη βάση του, ένα μεγάλο ρεύμα ρέει από τον εκπομπό στον συλλέκτη.



Τρανζίστορ ως διακόπτης

Για να χρησιμοποιήσουμε το τρανζίστορ ως διακόπτη, η τάση βάσης πρέπει να είναι υψηλότερη από την τάση συλλέκτη (σε περίπτωση NPN τρανζίστορ), ή κάτω από την τάση συλλέκτη (σε περίπτωση PNP τρανζίστορ). Επίσης, για να εξασφαλιστεί το τρανζίστορ είναι σε κορεσμό, θα πρέπει να υπολογίζει την κατάλληλη τιμή της αντίστασης Rb που φαίνεται στο σχηματικό διάγραμμα.

Οι δύο διαφορετικές υλοποιήσεις του διακόπτη τρανζίστορ, η μία με NPN τρανζίστορ,και η άλλη με PNP είναι για να εξασφαλιστεί ότι η τάση βάσης είναι σε ένα κατάλληλο επίπεδο ώστε το κάθε τρανζίστορ είναι σε κορεσμό αν είναι συνδεδεμένο με τη γείωση (GND), ή με τα 12V. (Στη H-Bridge, τα δύο τρανζίστορ είναι συνδεδεμένα με 12V, ενώ τα άλλα δύο είναι συνδεδεμένα με τη γείωση.)

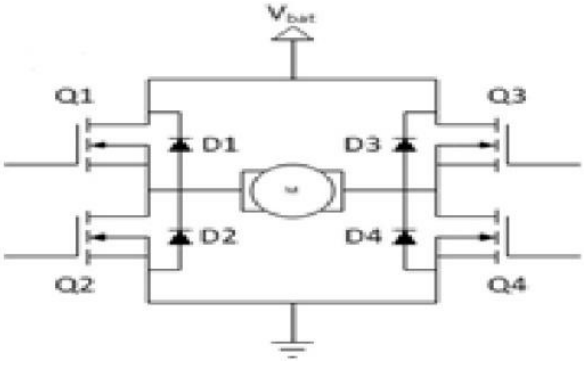
Ένα μειονέκτημα των τρανζίστορ είναι η αντίσταση συλλέκτη εκπομπού όταν αυτό είναι σε κορεσμό που έχει ως συνέπεια απώλειες ισχύος, (θερμικές).

Για να εξασφαλιστεί όσο το δυνατόν μικρότερη πτώση τάσης μεταξύ συλλέκτη και εκπομπού και κατά συνέπεια να έχουμε όσο το δυνατόν μικρότερες απώλειες ισχύος στην εργασία θα χρησιμοποιηθούν τρανζίστορ τύπου N-FET και P-FET.

Μία από τις πιο σημαντικές ιδιότητες ενός MOSFET είναι η εσωτερική αντίσταση μεταξύ Drain της και Source, όταν είναι ενεργοποιημένο. Αυτό είναι σημαντικό επειδή αυτή η αντίσταση καθορίζει το ποσό της θερμότητας που θα δημιουργηθεί με ένα δεδομένο επίπεδο ισχύος (IRF9530 = 200mΩ, IRF530 = 60mΩ).

Έτσι εξασφαλίζουμε τον έλεγχο μεγάλων ρευμάτων με αμελητέες απώλειες.

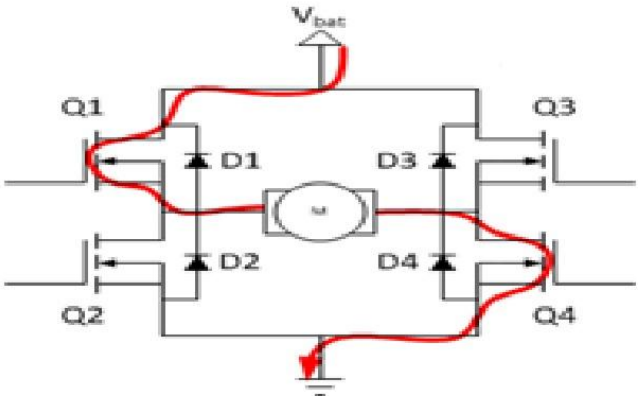
Στο παρακάτω σχήμα φαίνεται το βασικό κύκλωμα H-Bridge με τα τέσσερα MOSFETs ισχύος και αυτό το κύκλωμα λειτουργεί με τον ίδιο τρόπο όπως και οι τέσσερεις διακόπτες.



H-Bridge με τα τέσσερα MOSFETs

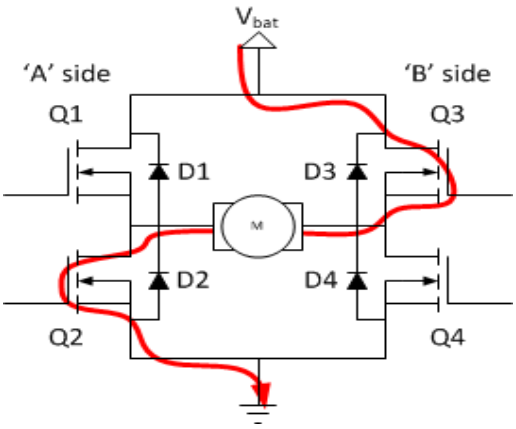
Αν το Q1 και Q4 είναι ενεργοποιημένα, ο αριστερός πόλος του κινητήρα θα είναι συνδεδεμένος με την παροχή ηλεκτρικού ρεύματος, ενώ ο δεξιός πόλος είναι συνδεδεμένος

στη γείωση. Ρεύμα αρχίζει να ρέει μέσω του κινητήρα που ενεργοποιεί τον κινητήρα στην κατεύθυνση προς τα εμπρός και ο άξονας του κινητήρα αρχίζει να γυρίζει.



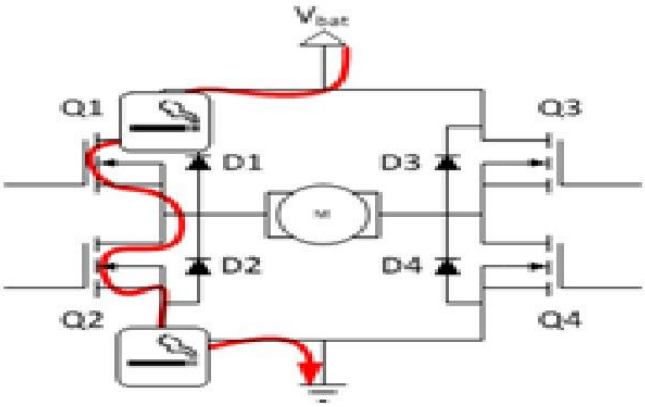
Q1 και Q4 είναι ενεργοποιημένα κίνηση εμπρός

Αν τα Q2 και Q3 είναι ενεργοποιημένα, θα συμβεί το αντίθετο, ο κινητήρας ενεργοποιείται προς την αντίθετη κατεύθυνση, και ο άξονας θα αρχίσει να γυρίζει προς τα πίσω.



Q2 και Q3 είναι ενεργοποιημένα κίνηση πίσω

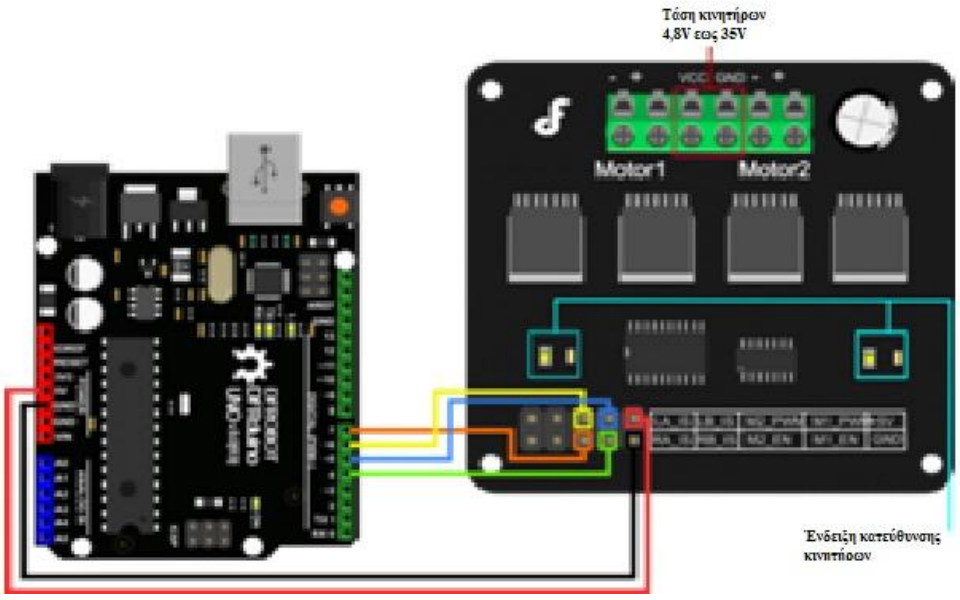
Σε μια H-Bridge, δεν πρέπει ποτέ να κλείσει το ζεύγος Q1 και Q2 (ή Q3 και Q4) ταυτόχρονα. Αν το κάναμε αυτό, θα δημιουργήσουμε μια διαδρομή πολύ χαμηλής αντίστασης μεταξύ τροφοδοσίας και GND, βραχυκυκλώνοντας την τροφοδοσία μας. Η κατάσταση αυτή ονομάζεται «shoot-through» και αποτελεί σχεδόν ένα εγγυημένο τρόπο για να καταστρέψουμε γρήγορα τη γέφυρα μας.



Q1 και Q2 (ή Q3 και Q4) ενεργοποιημένα ➔ shoot-through

Ο διπλός ελεγκτής 2×15A BTS7960

Για την κατασκευή μας επιλέχθηκε η συγκεκριμένη πλακέτα οδήγησης η οποία χρησιμοποιεί τέσσερα διπλά υψηλής απόδοσης και υψηλού ρεύματος (BTS7960) MOSFETs με τις ακόλουθες προστατευτικές λειτουργίες: βραχυκύκλωμα, υπέρβαση θερμοκρασίας και υπέρταση. Μπορούμε να ελέγξουμε ταυτόχρονα 2 κινητήρες με μόνο 4 ψηφιακές I/O. Διαχειρίζεται 15A στα 13.8V μέγιστο ρεύμα εξόδου, έχει καλή απόκριση και απόδοση φρεναρίσματος. Παρέχονται τέσσερις ενδεικτικές λυχνίες για εύκολο έλεγχο χωρίς κινητήρες. Υπάρχουν δύο ψύκτες αλουμινίου στο πίσω μέρος της μονάδας, για την παροχή ψύξης για παρατεταμένη κατάσταση υψηλής ισχύος. Αυτή η μονάδα DC Motor Driver είναι άμεσα συμβατή με το Arduino.



Συνδεσμολογία motor driver με το arduino

Ένας απλός κώδικας για τον έλεγχο της πλακέτας Motor driver είναι ο ακόλουθος

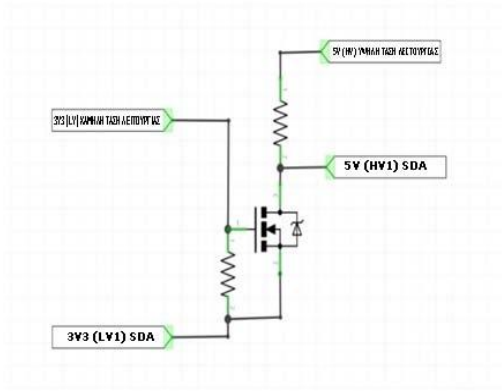
```
/*  BTS7960 Motor Driver Test  */

int RPWM=11; //M2PWM
int LPWM=3;   //M1PWM
// timer 0
int L_EN=7;    //M1EN (DIR M1)
int R_EN=6;   //M2EN (DIR M2)
int Motor_power=8;

void setup() {
    digitalWrite(Motor_power, HIGH);     // put your setup code here, to run once:
   for(int i=4;i<8;i++){
         pinMode(i,OUTPUT);
   }
  for(int i=4;i<8;i++){
       digitalWrite(i,LOW);
 }
  delay(1000);
   Serial.begin(9600);
 }

void loop() {   // put your main code here, to run repeatedly:
 Serial.println("EN High");
 digitalWrite(R_EN,HIGH);
 digitalWrite(L_EN,HIGH);
delay(1000);
    for(int i=0;i<256;i++){
      analogWrite(RPWM,i);
      analogWrite(LPWM,i);
     Serial.print("I=");
     Serial.println(i);
     delay(100);
}
delay(500);

for(int i=255;i>0;i--){
      analogWrite(RPWM,i);
      analogWrite(LPWM,i);
      Serial.print("I=");
      Serial.println(i);
     delay(100);
}
delay(500);

Serial.println("EN LOW");
 digitalWrite(R_EN,LOW);
  digitalWrite(L_EN,LOW);
    delay(1000);
for(int i=0;i<256;i++){
```

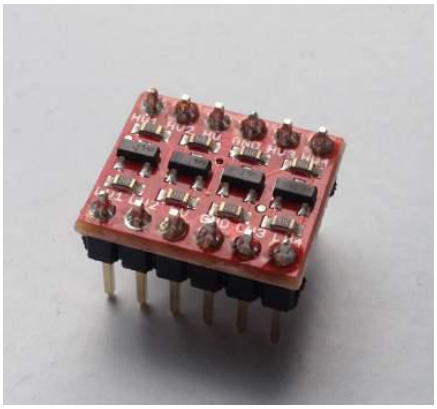
```
    analogWrite(RPWM,i);
    analogWrite(LPWM,i);
    Serial.print("I=");
   Serial.println(i);
  delay(100);
}
delay(500);

for(int i=255;i>0;i--){
    analogWrite(RPWM,i);
    analogWrite(LPWM,i);
   Serial.print("I=");
   Serial.println(i);
 delay(100);
}
delay(500);
}
```

3.3.4 Μετατροπέας λογικού επιπέδου (logic level converter)

Ο αισθητήρας MPU6050 λειτουργεί σε ένα εύρος τάσης μεταξύ 2.375V-3.46V, σύμφωνα με το datasheet. Έχει ενσωματωμένο ρυθμιστή χαμηλής τάσης, έτσι ώστε να είναι ασφαλές με τροφοδοσία 5V μέσω του Arduino. Δεν γίνεται χρήση της γραμμής Arduino 3.3V γιατί λόγω αυτού του ρυθμιστή τάσης δεν παρέχεται αρκετή τάση για τη σωστή λειτουργία του αισθητήρα.

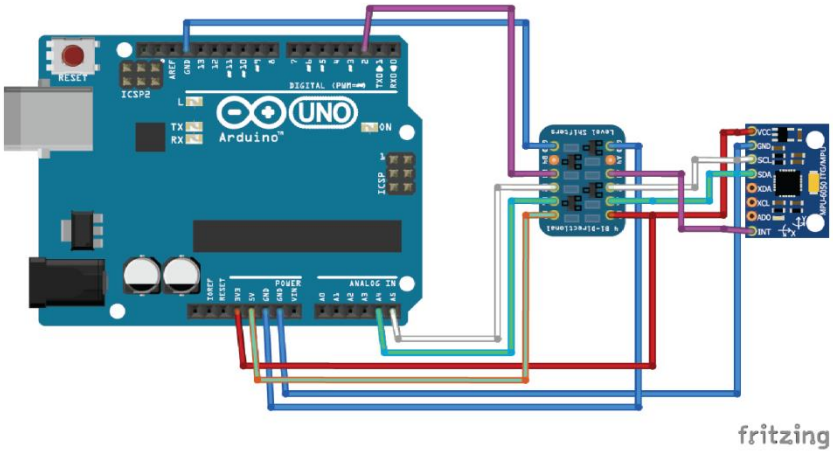
Το Arduino λειτουργεί σε επίπεδο U = 5V έτσι ένα HIGH στο δίκτυο I2C πρέπει να είναι σε αυτό το επίπεδο για να είναι σε θέση να επικοινωνεί με όλες συσκευές. Όλες οι συσκευές λοιπόν πρέπει να λειτουργούν σε επίπεδο 5V. Δεδομένου ότι αυτό δεν ισχύει στην περίπτωση του MP6050 το οποίο λειτουργεί σε επίπεδο 3V υπάρχει ανάγκη για ένα μετατροπέα επίπεδου (logic level converter σχήμα 5.2) μεταξύ του MPU6050 και του υπόλοιπου δικτύου (σχήμα 5.1) . Ο μετατροπέας επίπεδου λειτουργεί ως διερμηνέας για το MPU6050 και καθιστά δυνατή την επικοινωνία με το Arduino.



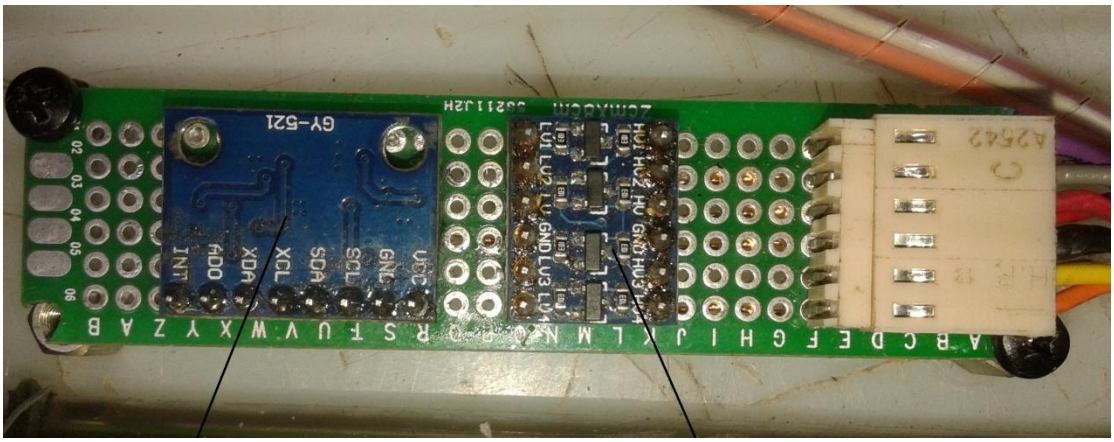


Μετατροπέας λογικού επιπέδου τεσσάρων εισόδων - εξόδων

MPU6050	ARDUINO UNO
VCC (Υπάρχει ρυθμιστής τάσης 3.3 V)	5V
GND	GND
SDA 0-3.3V	A4 (I2C SDA) 0 – 5V
SCL 0-3.3V	A5 (I2C SCL) 0 – 5 V
INT 0 – 3.3V	D2 (INTERUPT 0) 0 – 5 V



Συνδεσμολογία μετατροπέα λογικού επιπέδου με MPU6050 και ARDUINO

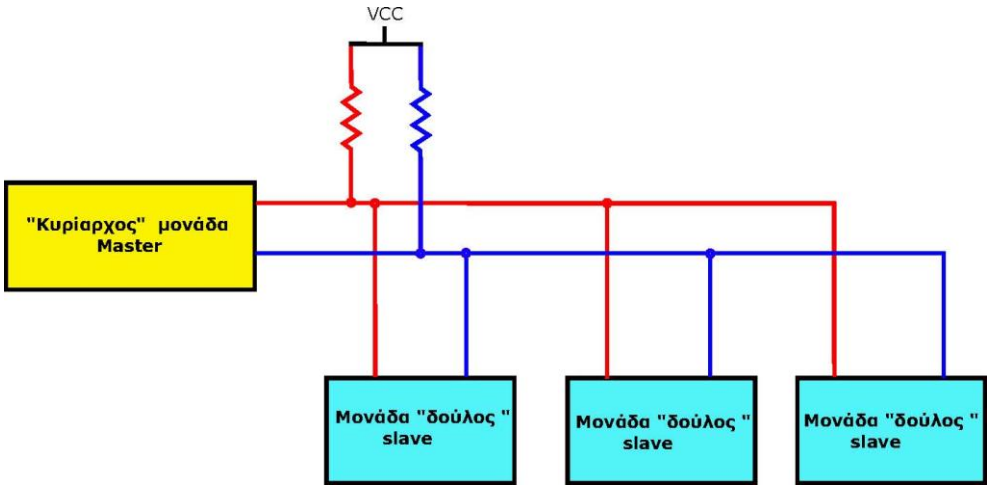


Η MPU6050 και ο μετατροπέας λογικού επιπέδου Logic συνδεδεμένα σε μια πλακέτα.

3.3.5 Πρωτόκολλο επικοινωνίας I2C

I2C σημαίνει «inter-integrated circuit» και επίσης μερικές φορές αναφέρεται ως «two wire interface». Είναι ένα πρωτόκολλο επικοινωνίας που χρησιμοποιεί μόνο δύο καλώδια για την επικοινωνία πολλαπλών κύριων και βοηθητικών συσκευών (master και slave). Ένα τέτοιο I2C bus δίκτυο φαίνεται παρακάτω. Τα δύο καλώδια είναι αμφίδρομης επικοινωνίας,

και ονομάζονται γραμμή σειριακών δεδομένων SDA, και σειριακή γραμμή χρονισμού SCL, αντίστοιχα.

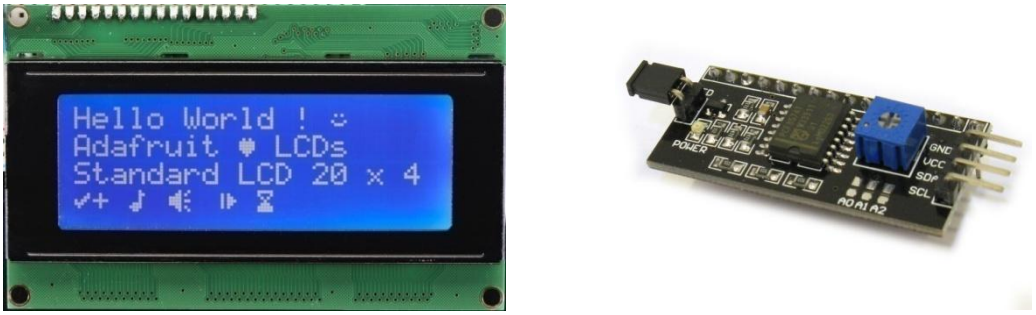


Παράδειγμα τυπικού δικτύου I2C.

Το I2C bus έχει δύο τύπους μονάδων: τον “κυρίαρχο” (master) και τον “δούλο” (slave). Η “κυρίαρχη” μονάδα παρέχει το χρονισμό και την διεύθυνση και η μονάδα “δούλος” δέχεται το χρονισμό και τη διεύθυνση. Για να λάβει νέα δεδομένα από έναν “δούλο” ο master θα στείλει μια διεύθυνση στο δίκτυο και οι “δούλοι” θα ανταποκριθούν. Αν υπάρχει ένας “δούλος” με την αποσταλμένη διεύθυνση στο δίκτυο αυτός θα απαντήσει με "Βεβαίωση λήψης" και ο “κυρίαρχος” τότε θα στείλει ποιο μητρώο (register) του και πόσα bytes του θέλει να διαβάσει.
Η μετάδοση ολοκληρώνεται όταν ο master στέλνει ένα stop-bit ή, αν θέλει να διατηρήσει το ελέγχου του δικτύου, ένα άλλο start bit.

3.3.6 Οθόνη LCD 20x4 με επικοινωνία I2C.

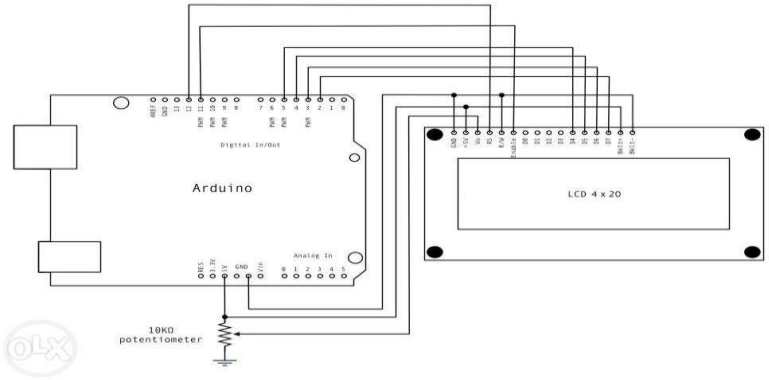
Για την αδιάλειπτη πληροφόρηση μας σχετικά με την κατάσταση του οχήματος αλλά και την ένδειξη των τιμών διάφορων μεταβλητών που εμείς ανά πάσα στιγμή θέτουμε, εισάγαμε στο σύστημα μας μια οθόνη LCD 20 x 4.



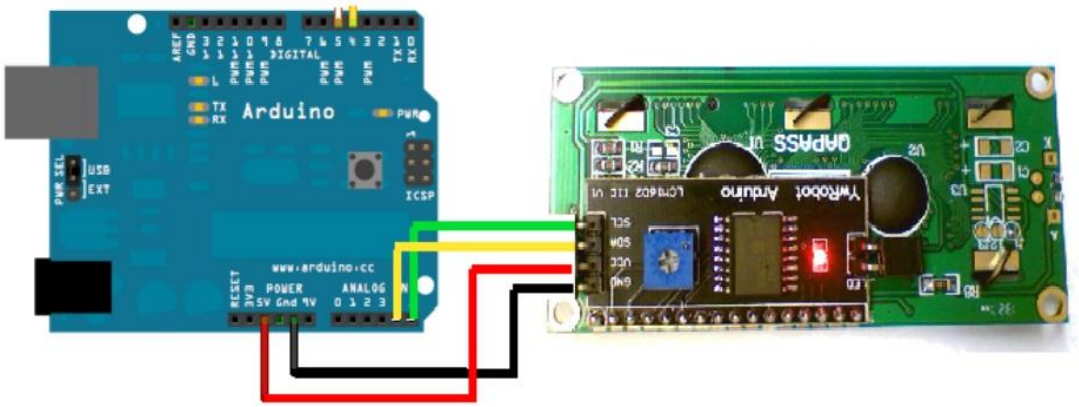
Οθόνη LCD 20X4 χαρακτήρων και το I2C interface

Επειδή οι συνδέσεις μιας τέτοιας οθόνης με το arduino δεσμεύουν αρκετές εξόδους επιλέχθηκε οθόνη με ενσωματωμένο *I2C interface LCD module*.

Με αυτού του είδους την οθόνη καταφέρνουμε να χρησιμοποιούμε μόνο τις ακίδες A4, A5, +5V και GND, οι οποίες όμως όπως είδαμε και παραπάνω χρησιμοποιούνται και από την MPU6050 χωρίς κανένα πρόβλημα.



Σύνδεση οθόνης LCD χωρίς I2C interface (9 καλώδια)



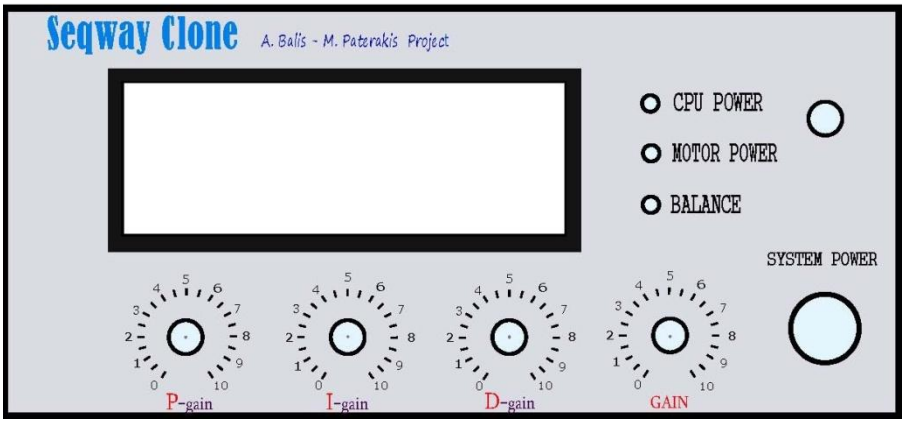
Σύνδεση οθόνης LCD μέσω διαύλου I2C (4 καλώδια)

Για την λειτουργία της πρέπει να ενσωματώσουμε στο κώδικα μας την βιβλιοθήκη *LiquidCrystal_I2C.h* καθώς και την εντολή

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
Η διεύθυνση επικοινωνίας της οθόνης είναι *0x27*

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
void setup() {
  lcd.begin(20,4);
lcd.clear();
  lcd.setCursor(0,0); //εκκίνηση LCD στη 1η γραμμή και 1η στήλη.
  lcd.print("Hello World !!!!); //εμφάνιση μηνύματος Hello World !!!!
}
void loop() {
}
```

Η οθόνη μαζί με τα ποτενσιόμετρα Kp,Ki,Kd και Gain καθώς και τα ενδεικτικά LED τοποθετήθηκαν σε κουτί (μονάδα ενδείξεων) στη μέση του τιμονιού για εύκολη πρόσβαση μιας και εδώ γίνονται αλλά και εμφανίζονται όλα σχεδόν τα δεδομένα του οχήματος.

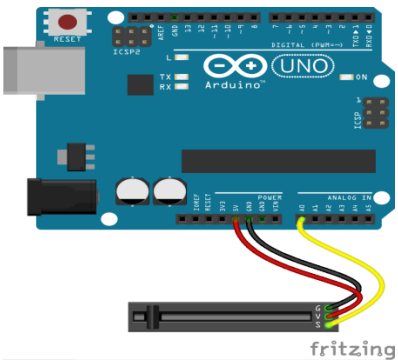


Η πρόσοψη της μονάδας ενδείξεων σχεδιασμένη στο Inkscape.

Η ένδειξη CPU POWER μας πληροφορεί κατά κύριο λόγο για την τροφοδοσία του Arduino καθώς και των υπολοίπων μονάδων που χειρίζονται 5V, ενώ η MOTOR POWER για το πότε οι κινητήρες τροφοδοτούνται με 12V. Η ένδειξη αυτή δεν θα είναι ενεργή όταν για παράδειγμα το όχημα βρίσκεται σε κλίση μεγαλύτερη των +/- 10 μοιρών, όπου οι κινητήρες είναι ανενεργοί. Τέλος η ένδειξη BALANCE μας πληροφορεί για το πότε ο όχημα είναι σε θέση +/- 2 μοιρών από τη κάθετη έτσι ώστε το όχημα να ενεργοποιείτε ομαλά χωρίς απότομες εκκινήσεις.

3.3.7 Ποτενσιόμετρα

Συνολικά ο χειριστής του οχήματος έχει την δυνατότητα ρύθμισης πέντε ποτενσιόμετρων. Τέσσερα από αυτά βρίσκονται στην μονάδα ενδείξεων, Kp – Ki – Kd – Gain, και ένα στην βάση του τιμονιού ώστε να ελέγξει την κατεύθυνση του οχήματος. Όλα τα ποτενσιόμετρα συνδέονται σε αναλογικές εισόδους στο arduino A0, A1,A2,A3 και A6. Ένα ποτενσιόμετρο είναι ένας απλός επιλογέας που παρέχει μια μεταβλητή αντίσταση, την οποία μπορούμε να διαβάσουμε στην πλακέτα Arduino ως αναλογική τιμή.



Τυπική σύνδεση συρόμενου ποτενσιόμετρου στο Arduino

Περιστρέφοντας τον άξονα του ποτενσιόμετρου, αλλάζουμε την ποσότητα αντίστασης δίνοντάς μας μια διαφορετική αναλογική είσοδο. Όταν ο άξονας είναι στραμμένος σε όλη τη διαδρομή προς μία κατεύθυνση, υπάρχουν 0 βολτ που πηγαίνουν στην ακίδα και διαβάζουμε 0. Όταν ο άξονας στρέφεται προς την άλλη κατεύθυνση, υπάρχουν 5 βολτ που πηγαίνουν στον πείρο και διαβάζουμε 1023. Στο μεταξύ, το analogRead () επιστρέφει έναν αριθμό μεταξύ 0 και 1023 που είναι ανάλογος προς την ποσότητα τάσης που εφαρμόζεται στην ακίδα.

Στο παράδειγμα που ακολουθεί, η τιμή *analogRead(potPin)* ελέγχει την ταχύτητα με την οποία αναβοσβήνει ένα LED.

```
 /* Analog Read to LED
 * turns on and off a light emitting diode(LED) connected to digital
 * pin 13. The amount of time the LED will be on and off depends on
 * the value obtained by analogRead(). In the easiest case we connect
 * a potentiometer to analog pin 2.
 * Created 1 December 2005
 * copyleft 2005 DojoDave <http://www.0j0.org>
 * http://arduino.berlios.de
 */

int potPin = 2;    // select the input pin for the potentiometer
int ledPin = 13;   // select the pin for the LED
int val = 0;       // variable to store the value coming from the sensor

void setup() {
  pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}

void loop() {
  val = analogRead(potPin);    // read the value from the sensor
  digitalWrite(ledPin, HIGH);  // turn the ledPin on
  delay(val);                  // stop the program for some time
  digitalWrite(ledPin, LOW);   // turn the ledPin off
  delay(val);                  // stop the program for some time
}
```

3.4 Η μέθοδος ελέγχου (PID)

Ας υποθέσουμε ότι πρέπει να καθοδηγήσουμε ένα φίλο μας με δεμένα τα μάτια, ώστε να προσεγγίσει μια καρέκλα και να καθίσει σε αυτήν. Μπορεί να δει την καρέκλα; , προφανώς όχι. Υπάρχουν αρκετοί τρόποι για να τον καθοδηγήσουμε. Ένας τρόπος είναι λέγοντάς του να πάει προς τα εμπρός μέχρι να αισθάνεται την καρέκλα και στη συνέχεια, καθίσει σε αυτήν. Σε αυτήν την περίπτωση, δεν παίρνουμε οποιαδήποτε ανατροφοδότηση από αυτόν. Έχουμε δηλαδή ένα σύστημα ελέγχου ανοικτού βρόχου, διότι δεν υπάρχει καμία σχέση μεταξύ της εισόδου και της εξόδου. Αυτό λειτουργεί σε ορισμένες απλές περιπτώσεις, αλλά σε άλλες περιπτώσεις, μπορεί προφανώς να αποτύχει.
Τι θα συμβεί αν η θέση στόχου κινείται; Σε αυτή την περίπτωση θα πρέπει να του δώσουμε στοιχεία (ανατροφοδότηση για το πού κινείται ο στόχος. Υπάρχει δηλαδή μια

σύνδεση μεταξύ της εισόδου (θέση του στόχου) και την έξοδο (που θα του πούμε να πάει). Ως εκ τούτου, αυτό είναι ένα παράδειγμα ενός συστήματος ελέγχου κλειστού βρόχου.

Τα συστήματα ελέγχου κλειστού βρόχου είναι τα περισσότερο διαδεδομένα στον έλεγχο συστημάτων. Κοινές εφαρμογές των συστημάτων ελέγχου στα πρώτα ρομπότ είναι σκόπευσης της κάμερας, η καθοδήγηση των servos για την αυτόνομη οδήγηση του ρομπότ, αλλά και των βραχιόνων κίνησης ή περιστροφής του. Ο πιο απλός τρόπος για την καθοδήγηση είναι απλά να δούμε τη διαφορά μεταξύ του πού βρισκόμαστε τώρα, και του σημείου που θέλουμε να είμαστε. Με άλλα λόγια, ρυθμίζουμε την τιμή εξόδου προς τον κινητήρα (-ες) που ελέγχει το μηχάνημα να είναι ανάλογη με τη διαφορά μεταξύ της τρέχουσας θέσης και της θέσης στόχου, που είναι γνωστή ως το σφάλμα. Αυτός ονομάζεται **αναλογικός έλεγχος**, και είναι το «P» του PID. Σε κάθε επανάληψη του βρόχου PID, ο ελεγκτής υπολογίζει το P χρησιμοποιώντας έναν τύπο:

P = Kp * (θέση στόχος - την τρέχουσα θέση) [P = Kp * τρέχον σφάλμα]

float error = (targetAngle - compAngleY);
*float pTerm = Kp * error*

Όπου **Kp** είναι μια σταθερά που καθορίζεται με πειραματισμό, και οι δύο θέσεις, θέση στόχου και τρέχουσα θέση, μας δίνονται από ένα διακόπτη, μια συσκευή αισθητήρα, ή ένα ποτενσιόμετρο στο βραχίονα. Αυτός ο τύπος του συστήματος ελέγχου λειτουργεί, και θα πάρει σχεδόν πάντα στη σωστή θέση αλλά μετά από αρκετό χρόνο.
Για να βελτιώσουμε τα πράγματα ας υποθέσουμε ότι θέλουμε να μάθουμε *τι συνέβη στο παρελθόν, ώστε να ελέγξουμε καλύτερα το μέλλον.* Για να το κάνουμε αυτό, ενσωματώνουμε όλα τα λάθη του παρελθόντος, και τα πολλαπλασιάζουμε με μία σταθερά, δημιουργώντας ένα είδος σταθμισμένου μέσου όρου του σφάλματος. Αυτό είναι το **ολοκληρωτικό τμήμα του ελέγχου**, το «I» στο PID. Σε κάθε επανάληψη του βρόχου, η μονάδα ελέγχου προσθέτει το τρέχον σφάλμα (θέση στόχου - την τρέχουσα θέση) με τα προηγούμενα:

Sum += τρέχον σφάλμα

και στη συνέχεια πολλαπλασιάζεται με μια άλλη πειραματική σταθερά Ki, για να δημιουργήσει την έξοδο από το ολοκληρωτικό τμήμα:

I = Ki * Sum

*iTerm += Ki * error * timeChange*

Αυτό μας βοηθά να φθάσουμε με αυστηρότερο βαθμό ελέγχου στον στόχο.

Ένα σύστημα που αποτελείται μόνο από αναλογικό τμήμα θα μπορούσε συνεχώς να οδηγείται σε υπέρβαση με συνέπεια να ταλαντεύεται. Το ολοκληρωτικό τμήμα βοηθά στην σταθεροποίηση αυτού του είδους λάθους, και στην επίτευξη ενός πιο ακριβέστερου βαθμού προσέγγισης του στόχου.

Το τελευταίο κομμάτι του βρόγχου PID, παράγωγο το «D» στο PID, *προσπαθεί να προβλέψει το μέλλον*, εξετάζοντας την αλλαγή. Ενώ στο ολοκληρωτικό τμήμα χρησιμοποιήσαμε το άθροισμα των σφαλμάτων του παρελθόντος, στο παράγωγο τμήμα του βρόχου θα εξετάσουμε την πρόσφατη αλλαγή στο σφάλμα. Το πρώτο παράγωγο σφάλματος διαχρονικά υπολογίζεται και πολλαπλασιάζεται με μια άλλη σταθερά, Kd, προκειμένου να δημιουργηθεί το τελικό κομμάτι του PID βρόχου μας:

Διαφορά = τρέχον σφάλμα - προηγούμενο σφάλμα

D = Kd * Διαφορά

*float dTerm = Kd * (compAngleY - lastpitch) / timeChange*

Αυτό το μέρος μας επιτρέπει να χρησιμοποιήσουμε την τρέχουσα αλλαγή σφάλματος, να συμπεράνουμε για το παρελθόν και να ρυθμίσουμε την έξοδο κατάλληλα. Παρατηρούμε ότι εάν το τρέχων σφάλμα είναι μικρότερο από το προηγούμενο σφάλμα, το D θα είναι αρνητικό, με συνέπεια την επιβράδυνση του συστήματος, καθώς είμαστε στη σωστή κατεύθυνση, βοηθώντας έτσι στην εξάλειψη υπέρβασης.
Για να ανακεφαλαιώσουμε, ένας ελεγκτής PID είναι ο συνδυασμός αναλογικού, ολοκληρωτικού, και παράγωγου ελεγκτή. Ο ελεγκτής PID ρυθμίζει την έξοδο με βάση όχι μόνο το τρέχων σφάλμα, αλλά και από το άθροισμα των προηγούμενων λαθών και την τρέχουσα μεταβολή σφάλματος.

Συστήματα ελέγχου PID

Το πιο σημαντικό στοιχείο κάθε ρομπότ είναι το σύστημα ελέγχου.
Ειδικά για το όχημα αυτο-εξισορρόπησης (SEGWAY), το σύστημα ελέγχου είναι ζωτικής σημασίας, δεδομένου ότι λαμβάνει – επεξεργάζεται τα δεδομένα του αισθητήρα (Gyro – Accel MPU6050) και αποφασίζει πόσο πρέπει να περιστραφούν οι κινητήρες προκειμένου το όχημα να παραμείνει όρθιο. Η πιο κοινός ελεγκτής που χρησιμοποιείται για τα συστήματα σταθεροποίησης είναι ο ελεγκτής PID.

Κινητήρας ως ελεγχόμενο σύστημα.

Το ελεγχόμενο σύστημα της εργασίας μας (SEGWAY) είναι ένας κινητήρας που θα να παρακολουθείται έμμεσα από μια συσκευή ανάγνωσης θέση (γυροσκόπιο – επιταχυνσιόμετρο). Παρόμοιο σύστημα ελέγχου θα μπορούσε φυσικά να εφαρμοστεί (με διάφορους αισθητήρες θέσης) και σε άλλα συστήματα όπως στο μηχανισμό οδήγησης ενός μεταφορέα σε έναν εκτυπωτή, ή στο μηχανισμό γκαζιού σε ένα σύστημα ελέγχου πορείας του αυτοκινήτου, ή σχεδόν οποιαδήποτε άλλο ελεγκτή θέσης.

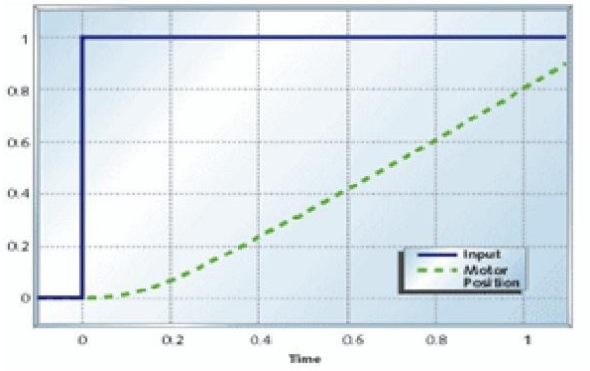
Το σχήμα 3α δείχνει ένα διάγραμμα ενός τέτοιου συστήματος διαμορφωμένο για το όχημα SEGWAY. Ο ελεγκτής κινητήρα οδηγείται από μία ψηφιακή τιμή κατεύθυνσης και ταχύτητας από το λογισμικό και τροφοδοτεί κατάλληλα τους κινητήρες. Η έξοδος των κινητήρων είναι προσαρμοσμένη για να οδηγήσει τις ρόδες του μηχανισμού. Η θέση πλαισίου του οχήματος μετράται με ένα γυροσκόπιο – επιταχυνσιόμετρο.

Σχήμα 3α: Σύστημα ελέγχου κλειστού βρόχου.

Σε ένα κινητήρα συνεχούς ρεύματος τον οποίο οδηγούμε με μία τάση θέλουμε να στρέφεται με μια ελεγχόμενη ταχύτητα και κατεύθυνση ανάλογες με την εφαρμοζόμενη τάση στον κινητήρα.

Συνήθως ο οπλισμός του κινητήρα έχει κάποια αντίσταση που περιορίζει την ικανότητά του να επιταχύνει, και έτσι ο κινητήρας έχει κάποια καθυστέρηση μεταξύ της μεταβολής της τάσης εισόδου και της προκύπτουσας αλλαγής ταχύτητας.

Το σχήμα 3β φαίνεται η απόκριση βήματος του συνδυασμού κινητήρα και μετάδοσης. Χρησιμοποιούμε μια σταθερά χρόνου t_0 = 0.2s. Η απόκριση βήματος ενός συστήματος είναι μόνο η συμπεριφορά της εξόδου σε απόκριση ενός σήματος εισόδου που πηγαίνει από το μηδέν σε κάποιο σταθερή αξία τη χρονική στιγμή t = 0. Η απόκριση του κινητήρα ξεκινά αργά λόγω του σταθερού χρόνου, και μετά από κάποιο χρόνο (που είναι έξω από το διάγραμμα) αποκτά μία σταθερή ταχύτητα.



Σχήμα 3β: Απόκριση κινητήρα σε συνάρτηση του χρόνου.

3.4.1 Ο ελεγκτής PID

Ο ελεγκτής PID βασίζεται στις μαθηματικές εξισώσεις καθενός από τους τρεις ελέγχους του, του αναλογικού (P), του ολοκληρωτικού (I) και του παράγωγου (D), που χρησιμοποιούνται για τον υπολογισμό της εξόδου. Κάθε ένα από αυτά τα στοιχεία εκτελεί μια διαφορετική εργασία και έχει μια διαφορετική επίδραση στη λειτουργία του συστήματος.

Ελεγκτής τριών όρων

Η συνάρτηση μεταφοράς του PID ελεγκτή είναι η ακόλουθη:

$$\boldsymbol{K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{S}}$$

Οι όροι Kp, Ki και Kd είναι αντίστοιχα:

- Kp = Αναλογικό κέρδος
- K_I = Ολοκληρωτικό κέρδος
- Kd = Διαφορικό κέρδος

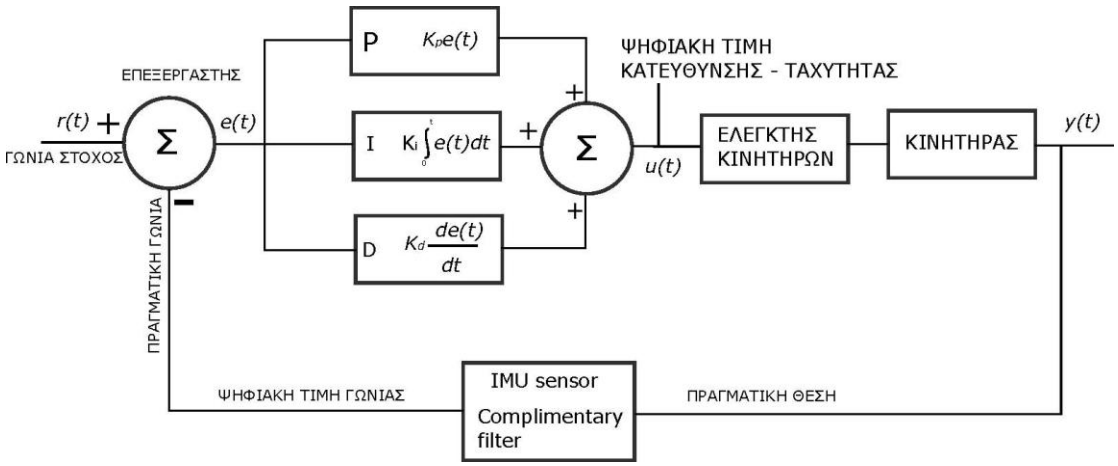
Ένα σύστημα ελέγχου κλειστού βρόχου είναι γνωστό και ως αρνητικό σύστημα ανάδρασης. Η βασική ιδέα ενός αρνητικού συστήματος ανάδρασης είναι ότι μετρά την έξοδο διεργασίας, πραγματική θέση, από έναν αισθητήρα (για την εργασία είναι ο MPU6050). Η μετρούμενη έξοδος διεργασίας αφαιρείται από την τιμή που έχει οριστεί ως σημείο αναφοράς, στόχος γωνία, για την παραγωγή ενός σφάλματος (e(t)). Το σφάλμα στη συνέχεια τροφοδοτεί τον ελεγκτή PID, ο οποίος το διαχειρίζεται με τρεις τρόπους. Ο ελεγκτής PID θα εκτελέσει τον αναλογικό P, τον ολοκληρωτικό I, και τον διαφορικό D έλεγχο, και με τον κατάλληλο αλγόριθμο PID επεξεργάζεται το σφάλμα, και παράγει ένα σήμα ελέγχου u.
Το σήμα (u) αμέσως μετά τον ελεγκτή είναι πλέον ίσο με το αναλογικό κέρδος (Kp) επί την τιμή του σφάλματος, συν το ολοκληρωτικό κέρδος (Ki) επί το ολοκλήρωμα του σφάλματος, συν το διαφορικό κέρδος (Kd) επί την παράγωγο του σφάλματος.

$$\boldsymbol{u = K_p e + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}}$$

*u = PID = (Kp * error) + (Ki * error * timeChange) + (Kd * (compAngleY - lastpitch) / timeChange)*

$$floatPIDValue = (pTerm + iTerm + dTerm)$$

Το σήμα αυτό θα σταλεί στο σύστημα ελέγχου κινητήρων και στη συνέχεια θα λάβουμε ένα νέο σήμα εξόδου *y(t)*. Το αποτέλεσμα από την επίδραση του σήματος εξόδου *y(t)* στους κινητήρες και κατά συνέπεια στο όχημα θα ανιχνευθεί από το αισθητήριο. Τα δεδομένα του αισθητηρίου, το οποίο θα εξάγει την πραγματική γωνία, θα αφαιρεθούν από την τιμή του επιθυμητού σημείου στόχου, γωνία στόχος (target angle – *r(t)*), και έτσι θα παραχθεί ένα νέο σήμα σφάλματος *e(t)*.
Για παράδειγμα, εάν το όχημά μας κινείται με 15 χιλ./ώρα αλλά θέλουμε να κινείται με 10 χιλ./ώρα, η τιμή του σφάλματος θα είναι 15 χιλ./ώρα - 10 χιλ./ώρα = 5 χιλ./ώρα. Ένα σύστημα ελέγχου δεν μπορεί να κάνει τίποτα αν δεν υπάρχει σφάλμα.
Ο ελεγκτής θα πάρει αυτό το νέο σήμα σφάλματος και θα υπολογίσει ξανά την παράγωγο και το ολοκλήρωμα και η ίδια διαδικασία θα επαναλαμβάνεται συνέχεια.
Το σήμα ελέγχου PID τροφοδοτεί τη διαδικασία επεξεργασίας (ARDUINO), και στη συνέχεια τροφοδοτούνται ανάλογα οι δύο κινητήρες του SEGWAY. Το σήμα ελέγχου PID προσπαθεί συνεχώς να προωθεί τη διαδικασία επεξεργασίας προς την τιμή του επιθυμητού σημείου στόχου (target angle). Στην περίπτωση του οχήματος SEGWAY, η επιθυμητή τιμή σημείου θέσης είναι η κατακόρυφη θέση μηδέν μοιρών.



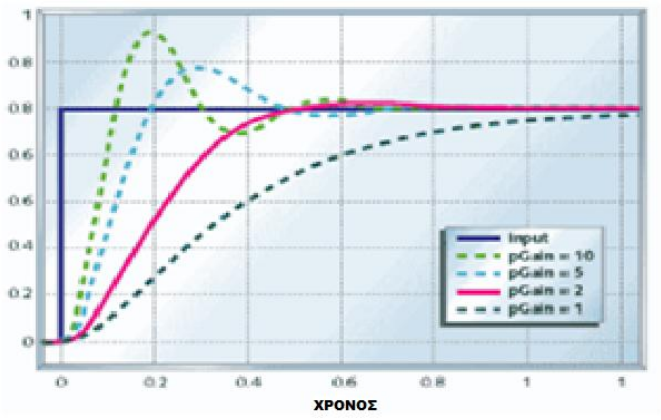
Βασικός ελεγκτής PID που εφαρμόζεται στην κατασκευή μας.

Αναλογικός Ελεγκτής (Proportional Controller)

Ο αναλογικός έλεγχος (P) είναι ο ευκολότερος και ίσως ο κοινός τρόπος ελέγχου για μία εφαρμογή. Ένας αναλογικός ελεγκτής είναι το σήμα σφάλματος πολλαπλασιαζόμενο με μία σταθερά (k) και το αποτέλεσμα τροφοδοτεί το ελεγχόμενο σύστημα (κινητήρες). Η χρήση ενός αναλογικού ελεγκτή (P), έχει ως αποτέλεσμα την ελάττωση του χρόνου ανύψωσης και την μείωση, αλλά ποτέ την εξάλειψη, του μόνιμου σφάλματος.

Ο ελεγκτής P λαμβάνει στην τρέχουσα γωνία του οχήματος (κλήση) και κάνει τους κινητήρες να κινούνται στην ίδια κατεύθυνση με το όχημα καθώς αυτό τείνει να πέσει. Ως εκ τούτου, όσο μεγαλύτερη είναι η κλήση του οχήματος, τόσο πιο γρήγορα κινούνται οι κινητήρες. Αν ο ελεγκτής P χρησιμοποιείται μόνος του, το όχημα θα μπορούσε να σταθεροποιηθεί για κάποιο χρονικό διάστημα, αλλά το σύστημα θα τείνει να ταλαντώνεται και τελικά θα πέσει.

Στο σχήμα 3δ φαίνεται τι συμβαίνει όταν χρησιμοποιήσουμε μόνο αναλογικό έλεγχο στον κινητήρα. Για μικρό κέρδος (kp = 1) ο κινητήρας πηγαίνει στο σωστό στόχο, αλλά αρκετά αργά. Η αύξηση του κέρδους (kp = 2) επιταχύνει την απόκριση σε ένα σημείο. Πέρα από αυτό το σημείο (kp = 5, kp = 10), ο κινητήρας ξεκινά πιο γρήγορα, αλλά με υπερβάσεις του στόχου. Αν συνεχίσουμε να αυξάνουμε το κέρδος θα φτάσει τελικά σε ένα σημείο όπου το σύστημα θα ταλαντεύεται γύρω από τον στόχο χωρίς να τον πετυχαίνει και το σύστημα θα είναι ασταθές.

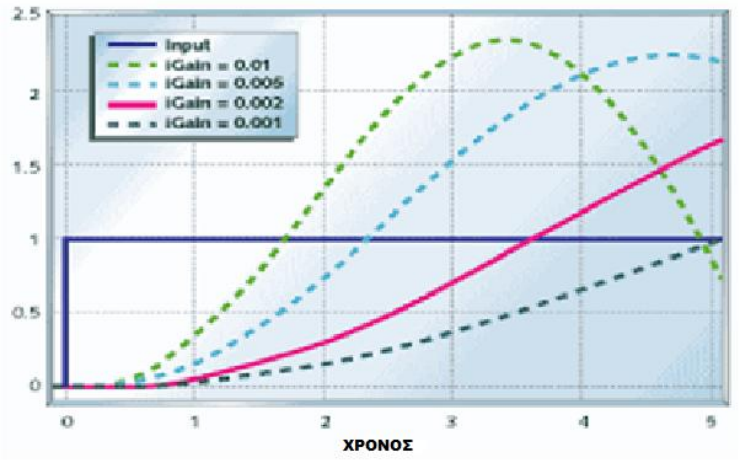


Σχήμα 3δ: Απόκριση κινητήρα με αναλογικό έλεγχο.

Ολοκληρωτικός Ελεγκτής (I - Controller)

Ο ολοκληρωτικός έλεγχος χρησιμοποιείται για να εξαλειφθούν τυχόν σφάλματα. Για παράδειγμα, εάν το όχημα τείνει να πέσει σε μια πλευρά, το στοιχείο I γνωρίζει ότι το όχημα πρέπει να κινηθεί προς την αντίθετη κατεύθυνση, προκειμένου να κρατηθεί όρθιο και έτσι προλαμβάνει την πτώση του μπρός ή πίσω. Ο ολοκληρωτικός έλεγχος (Ki) θα εξαλείψει το μόνιμο σφάλμα, αλλά θα χειροτερέψει την μεταβατική απόκριση (αριθμός των ταλαντώσεων μέχρι την τελική ισορροπία του συστήματος). Χρησιμοποιείται πάντοτε σε συνδυασμό με αναλογικό έλεγχο. Ο ολοκληρωτικός έλεγχος από μόνος του συνήθως μειώνει τη σταθερότητα, ή την καταστρέφει τελείως.

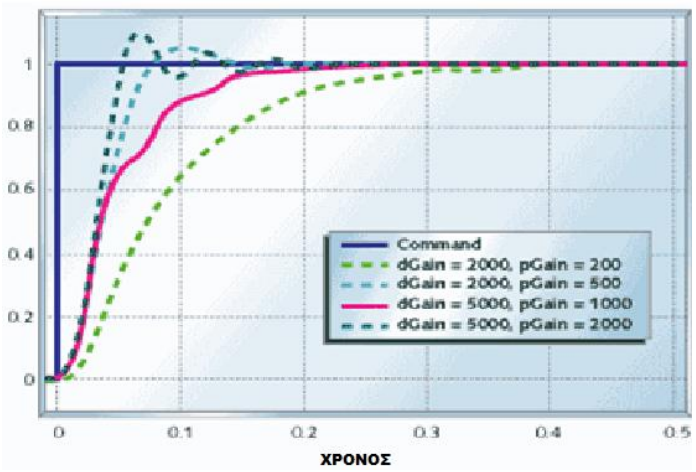
Το σχήμα 3ε δείχνει την απόκριση του κινητήρα με χαμηλό ολοκληρωτικό έλεγχο και μηδενικό αναλογικό(kp = 0). Οι κινητήρες με μόνο ολοκληρωτικό έλεγχο θα ταλαντώνονταν με όλο και μεγαλύτερες διακυμάνσεις.



*Σχήμα 3ε: Απόκριση κινητήρα με **ολοκληρωτικό** έλεγχο*

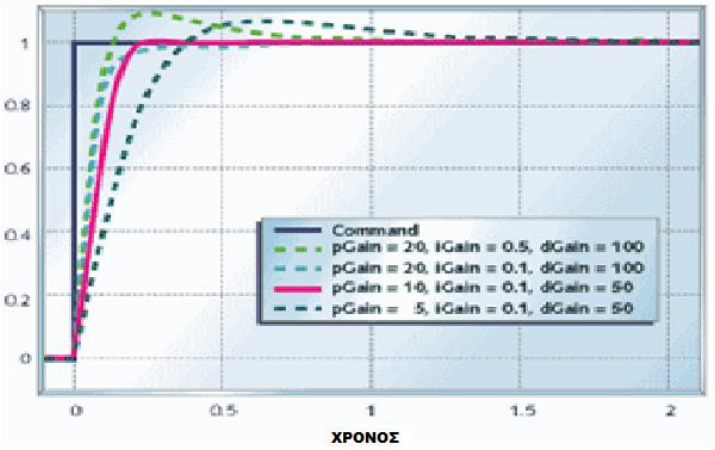
Διαφορικός ελεγκτής (D - Controller)

Σε γενικές γραμμές, αν δεν μπορούμε να σταθεροποιήσουμε ένα ελεγχόμενο σύστημα με αναλογικό έλεγχο, δεν μπορούμε να ο σταθεροποιήσουμε με τον έλεγχο PI.
Από τα παραπάνω γνωρίζουμε ότι ο αναλογικός έλεγχος επεμβαίνει στην παρούσα συμπεριφορά του ελεγχόμενου συστήματος, και ότι ο ολοκληρωτικός έλεγχος ασχολείται με την παρελθούσα συμπεριφορά του. Αν είχαμε κάποιο στοιχείο που προβλέπει την συμπεριφορά του ελεγχόμενου συστήματος, τότε αυτό θα μπορούσε να χρησιμοποιηθεί για να σταθεροποιήσει το ελεγχόμενο σύστημα. Ένας διαφορικός ελεγκτής θα μπορούσε να το κάνει αυτό. Ο διαφορικός έλεγχος (Kd) επιδρά στην αύξηση της σταθερότητας του συστήματος, μειώνοντας την υπερύψωση και βελτιώνοντας την μεταβατική απόκριση. Είναι υπεύθυνος για την απόσβεση τυχόν ταλαντώσεων και εξασφαλίζει ότι το όχημα δεν θα δονείται πάρα πολύ. Απλά δρα ενάντια σε κάθε κίνηση.
Το σχήμα 3η δείχνει την απόκριση του συστήματος με αναλογικό και διαφορικό (PD)έλεγχο. Το σύστημα αυτό σταθεροποιείται σε λιγότερο από το 1/2 του δευτερολέπτου, σε σύγκριση με τα άλλα συστήματα(2-3 δευτ.).



Σχήμα 3η: Κινητήρας με έλεγχο PD.

Το σχήμα 3θ δείχνει το σύστημα με έλεγχο PID. Μπορούμε να δούμε την βελτίωση των επιδόσεων που έχουμε με τη χρήση πλήρη ελέγχου PID σε αυτό το ελεγχόμενο σύστημα.

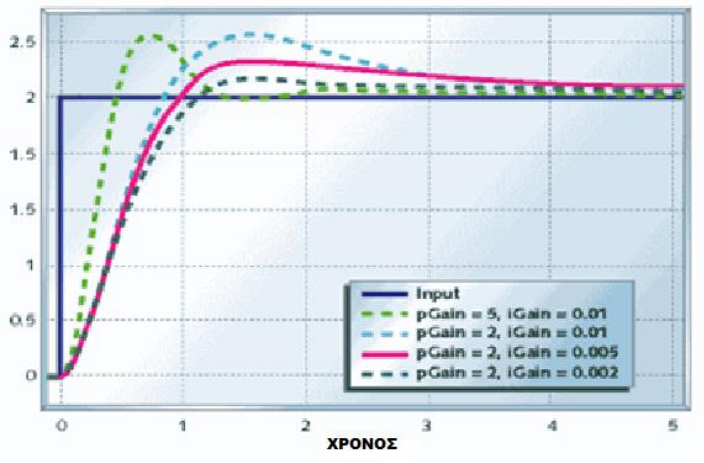


Σχήμα 3θ: Κινητήρας με έλεγχο PID.

Επιλεκτική χρήση όρων ελέγχου

Παρόλο που ένας ελεγκτής PID έχει τρεις όρους ελέγχου, ορισμένες εφαρμογές χρησιμοποιούν μόνο έναν ή δύο όρους για να παρέχουν τον κατάλληλο έλεγχο. Αυτό επιτυγχάνεται με τον μηδενισμό των μη χρησιμοποιούμενων παραμέτρων και ονομάζεται ελεγκτής PI, PD, P ή I απουσία των άλλων όρων ελέγχου. Οι ελεγκτές PI είναι αρκετά συνηθισμένοι, δεδομένου ότι η διαφορική δράση είναι ευαίσθητη στον θόρυβο μέτρησης, ενώ η απουσία ενός ολοκληρωτικού όρου μπορεί να εμποδίσει το σύστημα να φτάσει την τιμή στόχο του.

Το σχήμα 3ζ δείχνει την απόκριση του κινητήρα με συνδυασμό αναλογικού και ολοκληρωτικού (PI) έλεγχου. Συγκρίνοντάς το με τα σχήματα 3.5 και 3.6 φαίνεται ότι ο αναλογικός - ολοκληρωτικός ελεγκτής, μειώνει το χρόνο ανόδου και μηδενίζει το μόνιμο σφάλμα .



*Σχήμα 3ζ: Απόκριση κινητήρα με αναλογικό **και ολοκληρωτικό** έλεγχο*

Συνοψίζοντας όλα τα παραπάνω έχουμε τον ακόλουθο πίνακα.

Αντίδραση Ελεγκτή	**Χρόνος Ανύψωσης**	**Υπερύψωση**	**Χρόνος Αποκατάστασης**	**Μόνιμο Σφάλμα**
Kp	**Μείωση**	**Αύξηση**	**Μικρή αλλαγή**	**Μείωση**
Ki	**Μείωση**	**Αύξηση**	**Αύξηση**	**Εξάλειψη**
Kd	**Μικρή αλλαγή**	**Μείωση**	**Μείωση**	**Μικρή αλλαγή**

3.4.2 Ο αλγόριθμος ελέγχου PID.

Ο αλγόριθμος ελέγχου PID μπορεί να απεικονιστεί σε μία μαθηματική αναπαράσταση, για τον υπολογισμό του όρου «correction διόρθωση»:

*Διόρθωση = Kp*σφάλμα + Ki* ∫ σφάλματος + Kd* d/dt(σφάλμα);*

Kp, Ki και Kd είναι σταθερές που καθορίζονται πειραματικά.

Ο ολοκληρωτικός όρος I είναι απλώς το άθροισμα όλων των προηγούμενων αποκλίσεων και καλείται integral- «totalerror». Ο διαφορικός όρος D είναι η διαφορά μεταξύ της τρέχουσας απόκλισης και της προηγούμενης απόκλιση. Ο κώδικας που ακολουθεί εκτελείται κατ' επανάληψη και υπολογίζει τη διόρθωση *correction*. Αυτές οι γραμμές πρέπει να τρέχουν σε κάθε επανάληψη (loop):

Διόρθωση = Kp* απόκλιση + Ki * συνολικό σφάλμα + Kd * (απόκλιση – προηγούμενη απόκλιση)

u =PID = (Kp * error) + (Ki * error * timeChange) + (Kd * (compAngleY - lastpitch) / timeChange)

Έτσι ο κώδικάς θα γραφτεί ως εξής :

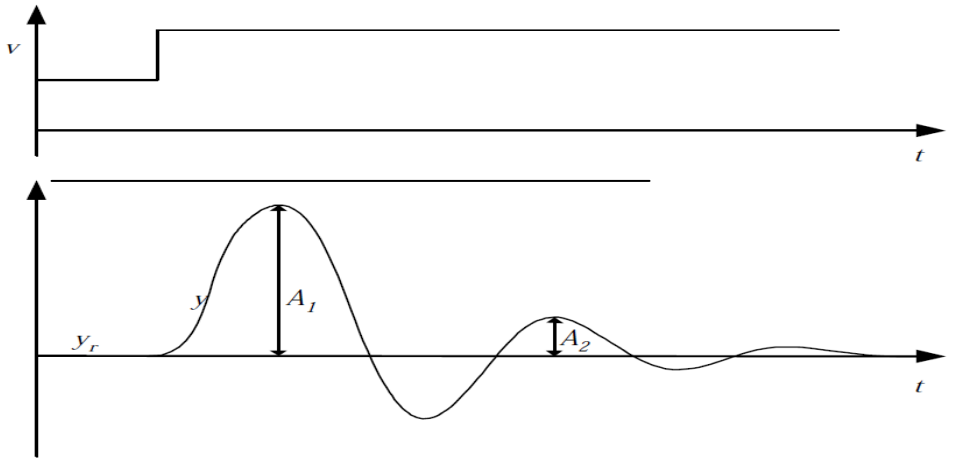
```
float pTerm = Kp * error;
iTerm += Ki * error * timeChange ;
float dTerm = Kd * (compAngleY - lastpitch) / timeChange ;
float PIDValue = pTerm + iTerm + dTerm;
```

3.4.3 Ρύθμιση των σταθερών PID

Η μέθοδος συντονισμού των Ziegler-Nichols.

Μια δημοφιλής μέθοδος για τον συντονισμό των ελεγκτών PID είναι αυτή που ανέπτυξαν και πρότειναν οι Ziegler- Nichols (1940) οι οποίοι παρουσίασαν δύο κλασσικές μεθόδους για τον προσδιορισμό τιμών αναλογικού κέρδους, του χρόνου ολοκλήρωσης και του χρόνου παραγώγισης με βάση τα χαρακτηριστικά μεταβατικής απόκρισης μιας δεδομένης εγκατάστασης ή συστήματος.

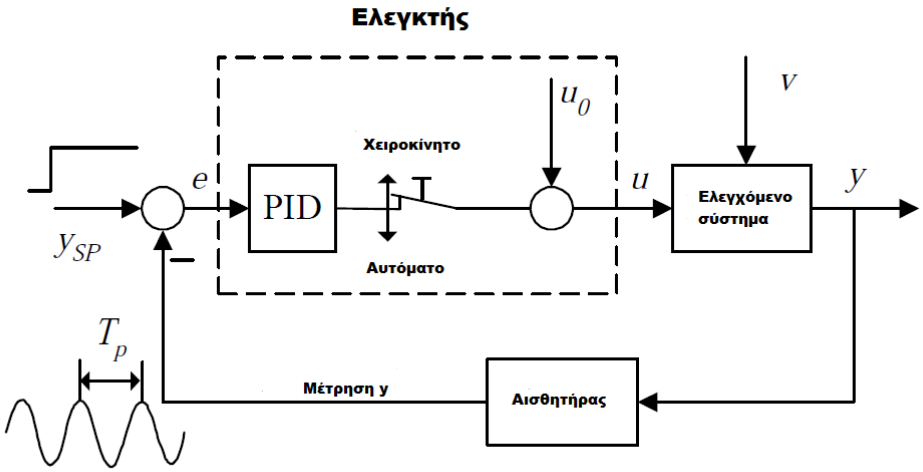
Οι Ziegler και Nichols χρησιμοποίησαν ένα ορισμό της αποδεκτής σταθερότητας ως βάση για τους κανόνες ρύθμισης του ελεγκτή, σύμφωνα με το οποίο η αναλογία του εύρους δύο συνεχόμενων κορυφών της ίδιας κατεύθυνσης (λόγω μιας βαθμιαίας αλλαγής της διαταραχής ή βαθμιαία αλλαγή της επιθυμητής τιμής στον βρόγχο ελέγχου) είναι περίπου $\frac{A_2}{A_1} = \frac{1}{4}$.



Εάν $\frac{A_2}{A_1} = \frac{1}{4}$ *η σταθερότητα του συστήματος είναι εντάξει, σύμφωνα με τον Ziegler και Nichols*

Η διαδικασία ρύθμισης PID των Ziegler-Nichols

Η μέθοδος κλειστού βρόχου του Ziegler-Nichols βασίζεται σε πειράματα που εκτελούνται σε έναν καθιερωμένο βρόχο ελέγχου (ένα πραγματικό σύστημα ή ένα προσομοιωμένο σύστημα).



Η διαδικασία συντονισμού έχει ως εξής:

1. Θέτουμε το ελεγχόμενο σύστημα στο καθορισμένο σημείο λειτουργίας του συστήματος ελέγχου για να διασφαλιστεί ότι ο ελεγκτής κατά τη διάρκεια του συντονισμού θα έχει αίσθηση της αντιπροσωπευτικής δυναμικής διαδικασίας και ελαχιστοποιούμε την πιθανότητα οι μεταβλητές κατά τη διάρκεια της ρύθμισης να φτάσουν στα όρια(μέγιστη ή ελάχιστη τιμή). Η μεταβλητή του συστήματος ελέγχου πρέπει να είναι περίπου ίση με την επιθυμητή τιμή.

2. Στον ελεγκτή PID ρυθμίζουμε τους όρους (κέρδη**) Kp=0, Ki=0**, και **Kd=0**, και θέτουμε τον βρόχο ελέγχου σε λειτουργία ρυθμίζοντας τον ελεγκτή στην αυτόματη λειτουργία.

3. Αυξάνουμε το $\boldsymbol{K_p}$ μέχρι να υπάρξουν παρατεταμένες ταλαντώσεις στο σύστημα ελέγχου. Αυτή η τιμή $\boldsymbol{K_p}$ δηλώνει το **τελικό ή κρίσιμο** κέρδος, $\boldsymbol{K_{pu}}$.

4. Μετρούμε την τελική ή κρίσιμη περίοδο **Pu** της διατηρούμενης ταλάντωσης.

5. Υπολογίζουμε τις τιμές των παραμέτρων του ελεγκτή σύμφωνα με τον πίνακα που ακολουθεί, και χρησιμοποιούμε αυτές τις τιμές παραμέτρων στον ελεγκτή.
Εάν η σταθερότητα του βρόχου ελέγχου είναι κακή, προσπαθούμε να βελτιώσετε τη σταθερότητα μειώνοντας το Kp, για παράδειγμα κατά 20%.

ΕΛΕΓΚΤΗΣ	K_p	K_i	K_d
P	0.5 K_{pu}	0	0
PI	0.45 K_{pu}	$\frac{P_u}{1.2}$	0
PID	0.6 K_{pu}	$Kp\frac{P_u}{2}$	$Kp\frac{P_u}{8}$

Υπάρχουν πολλοί άλλοι μέθοδοι συντονισμού όπως αυτή των **Chien, Hrones και Reswick** οι οποίοι πρότειναν μια τροποποίηση της πρώτης μεθόδου των Ziegler και Nichols ώστε το τελικό σύστημα να έχει μεγαλύτερο βαθμό απόσβεσης, και αυτή των **COHEN και COON** με

βασικό πλεονέκτημα της την απλότητα υλοποίησης, αλλά με μειονέκτημα τις ταλαντώσεις των αποκρίσεων.

Για την ρύθμιση του ελεγκτή PID του οχήματός μας ακολουθήσαμε τα παρακάτω βήματα

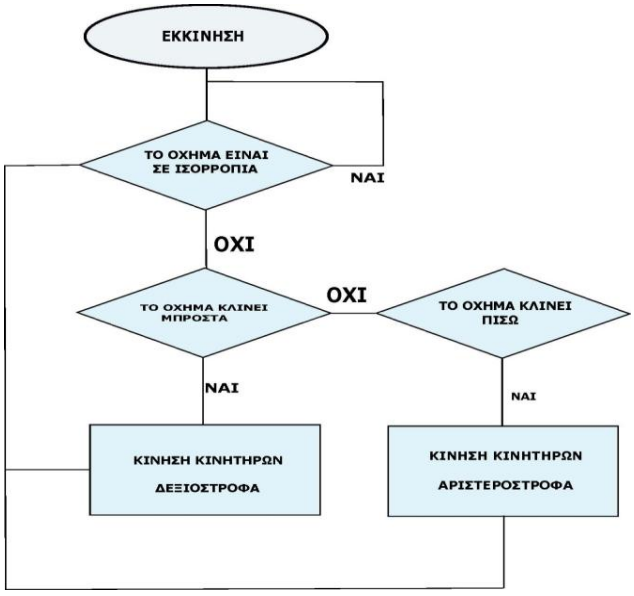
Χρειάζονται μερικά απλά βήματα για να ρυθμίσουμε τα στοιχεία Kp, Ki και Kd.

- Κρατάμε το όχημα σε κάθετη θέση. Θέση ισορροπίας
- Ρυθμίζουμε όλες τις σταθερές PID στο μηδέν.
- Σιγά-σιγά αυξάνουμε την σταθερά Kp. Ενώ κάνουμε αυτό, κρατάμε το όχημα σταθερά σε όρθια θέση και προσέχουμε να μην πέσει. Θα πρέπει να αυξήσουμε την σταθερά Kp μέχρι το όχημα να ανταποκρίνεται γρήγορα σε οποιαδήποτε κλίση, και να αρχίσει να ταλαντώνεται (κινείτε μπρος-πίσω) γύρω από τη θέση ισορροπίας. Το στοιχείο Kp θα πρέπει να είναι αρκετό ώστε το όχημα να κινηθεί, αλλά όχι πολύ μεγάλο γιατί η κίνηση δεν θα είναι ομαλή.
- Με ρυθμισμένο το στοιχείο Kp, αυξάνουμε το στοιχείο Ki. Αυτό το στοιχείο θα πρέπει να είναι σχετικά χαμηλά ρυθμισμένο, δεδομένου ότι διαφορετικά μπορεί να συσσωρεύονται τα λάθη πολύ γρήγορα. Στη θεωρία, το όχημα θα πρέπει να είναι σε θέση να σταθεροποιηθεί μόνο με αύξηση του στοιχείου Kp, αλλά θα ταλαντευόταν πολύ και τελικά θα έπεφτε.
- Αυξάνουμε την σταθερά Kd. Τα παράγωγα συστατικά λειτουργούν ενάντια σε κάθε κίνηση, οπότε αυτό βοηθάει στο να αμβλυνθούν οι ταλαντώσεις και να μειωθεί η υπέρβαση.

Δεν πρέπει να θέσουμε τα Kp,Ki,Kd πάρα πολύ υψηλά, καθώς θα μειωθεί η ικανότητα του οχήματος να αντιδρά σε εξωτερικές δυνάμεις. Φυσικά χρειάζονται επανειλημμένες προσπάθειες μικρορυθμίσεων για την επίτευξη του βέλτιστου αποτελέσματος.

3.5 Ο κώδικας του οχήματος

Μετά από αρκετή έρευνα και μελέτη σε δεκάδες κώδικες που σχετίζονται με οχήματα αυτό – ισορρόπησης και υπάρχουν διαθέσιμοι στο διαδίκτυο, καταλήξαμε να ενσωματώσουμε τμήματα διαφορετικών κωδικών ώστε να δημιουργήσουμε ένα κώδικα που να είναι εύκολο να τροποποιηθεί και να προσαρμοστεί στις ανάγκες και τα προβλήματα που παρουσιαζόταν κατά την υλοποίηση της κατασκευής του οχήματος.



Απλοποιημένο διάγραμμα ροής λειτουργίας του SEGWAY

ΑΛΓΟΡΙΘΜΟΣ SEGWAY

1 . Εκκίνηση λειτουργίας όταν ο διακόπτης εκκίνησης ενεργοποιηθεί.
2. Συλλογή των φυσικών παραμέτρων χρησιμοποιώντας τη Μονάδα Αδρανειακής Κίνησης IMU.
3. Χρήση της τεχνικής συμπληρωματικού φίλτρου για τον υπολογισμό αλλαγής γωνίας κλίσης.
4. Εάν το όχημα κλίνει, οδήγηση των τροχών προς την κατεύθυνση της κλίσης.
5. Αν η κλίση μεγαλώνει, αύξηση ταχύτητας.
6. Εάν η κλίσης είναι απότομη, αύξηση ταχύτητας.
7. Στο συστήματα ελέγχου:
α. Η μεταβλητή εισόδου είναι η γωνία κλίσης της πλατφόρμας.
β. Η μεταβλητή εξόδου είναι η ταχύτητα του κινητήρα.
8. Η μεταβλητή εισόδου να κρατηθεί μηδενική, ελέγχοντας την μεταβλητή εξόδου.
9. Όταν η κλίση είναι μηδέν σε σχέση με τις αρχικές κατακόρυφες θέσεις, παύση οδήγησης των κινητήρων.
10. Τερματισμός λειτουργίας όταν ο διακόπτης εκκίνησης απενεργοποιηθεί.

Για να γίνει πιο εύκολη ολόκληρη η διαδικασία, έχουμε δημιουργήσει μια ξεχωριστή υπορουτίνα για κάθε βήμα στο βρόχο loop(). Ο κώδικας που περιέχεται σε κάθε υπορουτίνα θα μπορούσε να εισαχθεί μέσα στον κύριο βρόχο, το όχημα θα λειτουργούσε με τον ίδιο τρόπο, αλλά ο χειρισμός και οι ρυθμίσεις του προγράμματος θα ήταν σίγουρα πιο περίπλοκος.

3.5.1 Επεξήγηση κώδικα

Κύριος βρόγχος

```
void loop() {
setup_mpu_6050_registers();   //  Υπορουτίνα ρύθμισης των καταχωρητών της MPU-6050
read_mpu_6050_data();         //  Υπορουτίνα συλλογής δεδομένων από την MPU6050
Gyro_Acc_angle_calculations (); //  Υπορουτίνα υπολογισμού γωνιών.
read_pots();                  //  Υπορουτίνα ανάγνωσης τιμών ποτενσιομέτρων.
auto_level();                 //  Υπορουτίνα ελέγχου ορθής εκκίνησης του οχήματος.
PID_and_Movement();           //  Υπορουτίνα υπολογισμού τιμών PID και κινήσεων.
timekeeper();                 //  Υπορουτίνα συμπλήρωσης χρόνου βρόγχου αν απαιτείται.
}
```

Οι ακόλουθες ενότητες αναφέρονται σε κάθε υπορουτίνα του κύριου βρόχου και περιγράφουν τι κάνει κάθε γραμμή του κώδικα. Στη συνέχεια, στο τέλος του κεφαλαίου, θα δούμε τον κώδικα στην πλήρη του μορφή, έτοιμο να προγραμματίσει το Arduino.

Η υπορουτίνα setup_mpu_6050_registers().

Ο έλεγχος επικοινωνίας καθώς και το «ξύπνημα» της MPU6050 είναι το πρώτο πράγμα που πρέπει γίνει. Αυτό επιτυγχάνεται μέσω του διαύλου I2C και έχει ως εξής:

```
Wire.beginTransmission(MPU_addr); // Ξεκίνησε επικοινωνία στην διεύθυνση 0x68
```

```
Wire.write(0x6B);
Wire.write(0);    // Θέτοντας ''0'' επαναφέρουμε την MPU από την κατάσταση sleep
Wire.endTransmission(true);     // Τέλος επικοινωνίας
```

Η υπορουτίνα read_mpu_6050_data().

Μέσω αυτής της υπορουτίνας γίνεται η συλλογή όλων των απαραίτητων πληροφοριών του επιταχυνσιόμετρου καθώς και του γυροσκόπιου. Οι τιμές που θα λάβουμε αφορούν αυτές του επιταχυνσιόμετρου (AcX, AcY, AcZ) και στους τρεις άξονες X,Y,Z και δίνονται σε ακτίνια, καθώς και αυτές του γυροσκόπιου (GyX,GyY,GyZ) πάλι και στους τρεις άξονες σε μονάδες γωνιακής ταχύτητας. Η τιμή Tmp αφορά θερμοκρασία της πλακέτας MPU6050. Στο σύνολό της η υπορουτίνα έχει ως εξής:

```
        Wire.beginTransmission(MPU_addr);
        Wire.write(0x3B);  // Εκκίνηση από καταχωριτή 0x3B (Accel_Xout_H)
        Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // Αίτηση εξαγωγής δεδομένων από 14
                                       // καταχωρητές (7 High + 7 Low)
       AcX=Wire.read()<<8|Wire.read();   // 0x3B (Accel_Xout_H) & 0x3C (Accel_Xout_L)
       AcY=Wire.read()<<8|Wire.read();   // 0x3D (Accel_Yout_H) & 0x3E (Accel_Yout_L)
       AcZ=Wire.read()<<8|Wire.read();   // 0x3F (Accel_Zout_H) & 0x40 (Accel_Zout_L)
       Tmp=Wire.read()<<8|Wire.read();   // 0x41 (Temp_out_H) & 0x42 (Temp_out_L)
      GyX=Wire.read()<<8|Wire.read();   // 0x41 (Gyro_Xout_H) & 0x44 (Gyro_Xout_L)
      GyY=Wire.read()<<8|Wire.read();   // 0x45 (Gyro_Yout_H) & 0x46 (Gyro_Yout_L)
      GyZ=Wire.read()<<8|Wire.read();   // 0x47 (Gyro_Zout_H) & 0x48 (Gyro_Zout_L)
```

Με την εντολή *Wire.beginTransmission(MPU_addr);* ξεκινάει η επικοινωνία μέσω του διαύλου I2C. Στη συνέχεια με την *Wire.write(0x3B);* προσδιορίζεται ο πρώτος καταχωρητής από τον οποίο θα ξεκινήσει η ανάγνωση δεδομένων. Αμέσως μετά με την

Wire.endTransmission(false); δηλώνεται πως η επικοινωνία δεν έχει τελειώσει, και με την

Wire.requestFrom(MPU_addr,14,true); γίνεται αίτημα απόδοσης των τιμών από 14 καταχωρητές. Κάθε τιμή του επιταχυνσιόμετρου, γυροσκόπιου και θερμόμετρου προέρχεται από 2 καταχωρητές, ενός χαμηλών τιμών και ενός υψηλών τιμών, των 8bits.

Για να αποδοθούν οι 7 τιμές ζητάμε τις τιμές των 14 καταχωρητών και απλά κάνουμε το άθροισμα κάθε ζεύγους. Αυτός είναι ο λόγος που μετατοπίζουμε προς τα αριστερά τις υψηλές τιμές (<<) και εκτελούμε λογική πράξη OR (|) για να προσθέσουμε τις χαμηλές τιμές (*Wire.read()<<8|Wire.read())*.

Η υπορουτίνα Gyro_Acc_angle_calculations ().

Όπως υποδηλώνεται από τον τίτλο της εδώ γίνονται οι υπολογισμοί – μετατροπές των λαμβανόμενων πληροφοριών από το επιταχυνσιόμετρο – γυροσκόπιο. Όπως προαναφέρθηκε οι τιμές αυτές θα πρέπει να μετατραπούν από ακτίνια για το επιταχυνσιόμετρο σε μοίρες / δευτερόλεπτο, και από γωνιακή ταχύτητα για το γυροσκόπιο σε μοίρες / δευτερόλεπτο επίσης. Η σταθερά *degconvert* (57.2957786) που έχει δηλωθεί, αντιστοιχεί στο πηλίκο 180/π και δηλώνεται έτσι για να ελαφρύνει το κυρίως πρόγραμμα από παραπανίσιους υπολογισμούς. Τα αποτελέσματα για τον Y άξονα λαμβάνονται στην

```
double roll = atan2(AcY, AcZ)*degconvert;
```

και για τον X στην

*double pitch = atan2(-AcX, AcZ)*degconvert;*

Για τις τιμές του γυροσκοπίου βάση του MPU6050 datasheet μια απλή διαίρεση με το 131.0 των τιμών που λαμβάνονται θα μας δώσει το επιθυμητό αποτέλεσμα.

double gyroXrate = GyX/131.0; και *double gyroYrate = GyY/131.0*

```
void Gyro_Acc_angle_calculations () {
double dt = (double)(micros() - timer)  /  1000000;  // microsecond to second
timer = micros();
double roll = atan2(AcY, AcZ)*degconvert;
double pitch = atan2(-AcX, AcZ)*degconvert;
double gyroXrate = GyX/131.0;
double gyroYrate = GyY/131.0;
```

Στη συνέχεια θα εφαρμοστεί το συμπληρωματικό φίλτρο (complimentary filter).

Οι τιμές του γυροσκόπιου φιλτράρουν τις ανεπιθύμητες αιχμές από το επιταχυνσιόμετρο που προκαλούνται από προσκρούσεις, δονήσεις, ή ξαφνικές αλλαγές στη βαρύτητα που δεν έχουν επηρεάσει τη γωνία.

Το επιταχυνσιόμετρο επηρεάζεται σε μεγάλο βαθμό από δονήσεις και χτυπήματα, έτσι το ποσοστό της στο μέσο όρο θα πρέπει να είναι χαμηλό ίσο με 2% (0,02). Αυτό σημαίνει ότι η compAngleX + gyroXrate, η οποία περιέχει ενημέρωση βασισμένη σε προηγούμενη φιλτραρισμένη γωνία, σταθμίζεται με μεγαλύτερο ποσοστό κατά 98% (0,98). Αυτό συμβαίνει επειδή το γυροσκόπιο είναι πολύ πιο σταθερό δεδομένης της τρέχουσας γωνίας από το επιταχυνσιόμετρο. Το μικρό βάρος 2% από το επιταχυνσιόμετρο ωστόσο, είναι αρκετό για να κρατήσει το γυροσκόπιο από εκτροπή (drift).

```
compAngleX = 0.98 * (compAngleX + gyroXrate * dt) + 0.02 * roll;
compAngleY = 0.98 * (compAngleY + gyroYrate * dt) + 0.02 * pitch;
```

Για να πάρουμε την τρέχουσα γωνία, ορίζουμε τη γωνία ίση με το άθροισμα των δύο σταθμισμένων αισθητήρων ((0.98) γυροσκόπιο + (0.02) επιταχυνσιόμετρο). Μπορούμε να αλλάξουμε τις στάθμες των δύο αυτών μεταβλητών από 98% γυροσκόπιο και 2% επιταχυνσιόμετρο, πηγαίνοντας στο 95% και 5% αντίστοιχα όμως κατά τις δοκιμές παραγόταν ασταθές αποτελέσματα.

Η υπορουτίνα *read_pots().*

Εδώ διαβάζονται και προσαρμόζονται ανάλογα με τη συνάρτηση map() οι τιμές από τα ποτενσιόμετρα κατεύθυνσης (Steering) , και των συντελεστών kp,ki,kd του ελεγχτή PID.

Για την επίτευξη αλλαγής κατεύθυνσης του οχήματος, θα πρέπει να δημιουργηθεί μια ανισορροπία μεταξύ των εξόδων του κάθε κινητήρα. Το ποτενσιόμετρο steering δεν κάνει τίποτα περισσότερο από το να επιβάλει στον ένα κινητήρα να κινείται ταχύτερα από ό, τι ο άλλος, γεγονός που συμβάλει με τη σειρά του στην αλλαγή κατεύθυνσης του οχήματος.

Η τιμή του ποτενσιόμετρου διευθύνσεως διαβάζεται από το Steer_Pot στην αναλογική είσοδο (ακροδέκτης A2) και αποθηκεύεται στη μεταβλητή Steering. Στη συνέχεια η τιμή

χαρτογραφείται (*map*) ώστε η τιμή *Steer* να αντιπροσωπεύει την γωνία του τιμονιού. Αν το τιμόνι είναι απόλυτα κατακόρυφο το ποτενσιόμετρο Steer_Pot θα είναι στη μέση της διαδρομής του και έτσι η τιμή *Steering* θα είναι 512 και η τιμή *Steer =0* άρα το όχημά μας θα κινείται ευθεία. Αν η κλίση του τιμονιού ποτενσιόμετρου είναι είτε αριστερά είτε δεξιά, το Arduino επιβάλει στους κινητήρες διαφορετική ταχύτητά, (διαφορετική τιμή PWM σε κάθε κινητήρα) δημιουργώντας έτσι αλλαγή κατεύθυνσης.

```
Steering = analogRead(Steer_Pot);
Steer = map (Steering, 0,1023,-255,255); //  -255 έως -1 αριστερά / 1 έως 255 δεξιά.
```

Η τιμή Steer θα επηρεάσει την έξοδο PWM προς τους κινητήρες στην υπορουτίνα PID_and_Movement() . Μια περίπτωση φαίνεται παρακάτω.

```
if (Steer < 0) {                          // αν το τιμόνι γέρνει αριστερά
left_PWM = left_PWM - abs(Steer);       // μείωση ταχύτητας αριστερού κινητήρα
right_PWM = right_PWM + abs(Steer);  //αύξηση ταχύτητας δεξιού κινητήρα
 }
```

Η υπορουτίνα *auto_level().*

Ένα μικρό κομμάτι του κώδικα ελέγχει όχι μόνο την κατάσταση του διακόπτη ενεργοποίησης αλλά και τη γωνία της IMU. Για όσο χρόνο ο διακόπτης ενεργοποίησης είναι ανοιχτός (όχι πατημένος), το όχημα δεν μπορεί να κινηθεί. Όταν ο διακόπτης αυτός είναι κλειστός (πατημένος), ο κώδικας αρχίζει να ελέγχει τη γωνία από την IMU για να βεβαιωθεί ότι είναι σε επίπεδο 0^o μοίρες πριν από τη ενεργοποίηση των κινητήρων. Όταν το όχημα είναι σε επίπεδο 0^o μοιρών, γίνεται μια ομαλή ενεργοποίηση των κινητήρων και αρχίζει πάλι η επεξεργασία των τιμών ταχύτητας των κινητήρων. Αυτό απομακρύνει τις σπασμωδικές κινήσεις του οχήματος αν κατά λάθος πατηθεί ο διακόπτης ενεργοποίησης ενώ δεν είμαστε στο σωστό επίπεδο. .... Πρώτα απ όλα η ασφάλεια!

Οι μεταβλητές που χρησιμοποιούνται για τη λειτουργία auto_level () είναι:

```
int Start_switch = 12;
int Start = false;
int Start_state = 1;
```

Κατ 'αρχάς, διαβάζεται η κατάσταση του *Start_switch* αποθηκεύεται στη μεταβλητή *Start_state*:

```
Start_state = digitalRead(Start_switch)
```

Τώρα ελέγχετε εάν ο διακόπτης είναι ανοικτός ή κλειστός. Για να αποφευχθεί η χρήση αντιστάσεων pull-up για τον *Start_switch*, χρησιμοποιούνται οι ενσωματωμένες στο Arduino αντιστάσεις pull-up. Αυτό σημαίνει ότι όταν το κουμπί δεν είναι πατημένο, ο διακόπτης διαβάζει HIGH ή 1 και όταν πατηθεί το κουμπί, ο διακόπτης θα διαβάσει LOW ή 0.

```
pinMode(Start_switch, INPUT);
digitalWrite(Start_switch, HIGH);  //  ενεργοποίηση pull-up αντίστασης
```

Αυτό είναι αντιφατικό, αλλά κάνει την καλωδίωση του διακόπτη πολύ πιο απλή, συνδέοντας τον ένα ακροδέκτη του διακόπτη στην είσοδο Arduino (D12) και τον άλλο στο GND.

Χρησιμοποιείτε μια εντολή if για να δούμε αν η μεταβλητή *Start_state* είναι υψηλή. (Αυτό αν ο διακόπτης είναι ανοικτός.) Αν ναι, η μεταβλητή *Start* τίθεται ως ψευδείς. (Δηλαδή, γίνεται απεμπλοκή των κινητήρων.)

```
if (Start_state == 1){
  Start = false;
```

Σε αντίθετη περίπτωση (else), αν η *Start_state* είναι LOW (δηλαδή, το κουμπί είναι πατημένο), χρησιμοποιήστε μια άλλη εντολή if για να ελεγχθεί αν η μεταβλητή *Start* έχει οριστεί σε false. Εάν πατηθεί το πλήκτρο, αλλά η μεταβλητή *Start* είναι ψευδής, αυτό σημαίνει ότι ο διακόπτης ήταν ανοιχτός, αλλά κάποιος τον έχει ενεργοποιήσει.

Σε αυτό το σημείο, το Arduino είναι έτοιμο να επιβληθεί στους κινητήρες, αλλά θα πρέπει πρώτα να βεβαιωθεί ότι το όχημα είναι σε επίπεδο 0^{o}. Για να γίνει αυτό, χρησιμοποιείτε ένα ακόμη εντολή if που ελέγχει την τρέχουσα τιμή της γωνίας του οχήματος. Αν η γωνία είναι κάτω από 0,4 και πάνω από -0.4 (δηλαδή, είναι σχεδόν σε τέλειο επίπεδο), επανεμπλέκονται εκ νέου οι κινητήρες θέτοντας τη μεταβλητή *Start* ίση με true. Σε αντίθετη περίπτωση, εάν η γωνία δεν είναι κοντά στις 0^{o}, οι κινητήρες απεμπλένονται μέχρι να διορθωθεί η γωνία. Αν η μεταβλητή *Start* είναι ήδη true, όταν εισέρχεται σε αυτό το *else* {}, εξακολουθεί να ισχύει true έως ότου απελευθερωθεί ο *Start _switch*.

```
  Start_state = digitalRead(Start_switch);
  if (Start_state == 1){
  Start = false;
  }
   else {
     if (Start == false){
       if (compAngleY < 0.4 &&  compAngleY > -0.4)
       Start = true;
     else {
    Start = false;
     }
  }
  else {
  Start = true;
  }
  }
}
```

Η υπορουτίνα PID_and_Movement().

Εδώ εκτελούνται οι κυριότερες λειτουργίες του σχεδίου πράγμα που δικαιολογεί και το μέγεθός της. Πέρα από το σημαντικό υπολογισμό της τιμής PID, έχουμε αναφερθεί σχετικά, εδώ υπολογίζονται και οι εντολές κατεύθυνσης, ταχύτητας, που θα αποσταλούν στους ελεγκτές κινητήρων (h-bridge) για κάθε κινητήρα ξεχωριστά.
Ένα τμήμα της υπορουτίνας αυτής είναι αρκετό για την κατανόησή της δημιουργίας σημάτων προς τους ελεγκτές.

```
if (left_PWM >= 0){
digitalWrite(DIR_M1, LOW);                //κίνηση αριστερού κινητήρα εμπρός
analogWrite(PWM_M1, left_PWM);      //  με ταχύτητα left_PWM
}
 else if (left_PWM < 0){
  digitalWrite(DIR_M1, HIGH);               //κίνηση αριστερού κινητήρα πίσω
  analogWrite(PWM_M1, -left_PWM);     // με ταχύτητα  - left_PWM
 }
```

Όπως φαίνεται παραπάνω το σήμα *left_PWM* είναι αυτό που θα καθορήσει πέρα από την ταχύτητα και την κατεύθυνση του κινητήρα ανάλογα αν το σήμα αυτό έχει θετική ή αρνητική τιμή. Αν η τιμή είναι θετική η ψηφιακή έξοδος *DIR_M1* γίνεται *LOW ενώ στην* αναλογική έξοδο *PWM_M1* εγγράφεται η τιμή *left_PWM.* Στην περίπτωση που η τιμή *left_PWM* είναι αρνητική η ψηφιακή έξοδος *DIR_M1* γίνεται *HIGH,* ενώ στην αναλογική έξοδο μια αρνητική τιμή *PWM* θα προκαλούσε μη κατανοητό από τον ελεγκτή σήμα. Αυτό είναι και ο λόγος που εγγράφεται τιμή *- left_PWM η οποία μετατρέπει την αρνητική τιμή σε θετική. (-- = +).*

Η υπορουτίνα ***timekeeper()***

Αυτή η υπορουτίνα ελέγχει την χρόνο του κάθε κύκλου βρόχου, και προσθέτει μια καθυστέρηση delay() μέχρι να ισούται με το χρονικό διάστημα
STD_LOOP_TIME = 10 milliseconds.

Θα πρέπει κάθε κύκλος βρόχου να ισούται με 10 χιλιοστά του δευτερολέπτου (ή κάποιο άλλο γνωστό χρόνο), ώστε να έχουμε τον σωστό υπολογισμό τιμής του γυροσκόπιο σε μοίρες ανά δευτερόλεπτο, αντί μοίρες ανά κύκλο βρόχου (η οποία δεν είναι χρήσιμη).
Θα χρησιμοποιηθούν σαν χρόνος ανανέωσης τα 10 milliseconds για να παρέχεται μια ανάγνωση γωνίας που ενημερώνεται από το Arduino 100 φορές κάθε δευτερόλεπτο! Αυτό παρέχει απρόσκοπτη διόρθωση γωνίας από το όχημά μας.

```
lastLoopUsefulTime = millis() - loopStartTime;
```

Αφαιρώντας την loopStartTime μεταβλητή από την τρέχουσα τιμή millis (), μπορούμε να πάρουμε τον ακριβή χρόνο *lastLoopUsefulTime* (σε χιλιοστά του δευτερολέπτου) από την τελευταία ανανέωση.

```
 if (lastLoopUsefulTime < STD_LOOP_TIME) {
delay(STD_LOOP_TIME - lastLoopUsefulTime);
```

Εάν η τρέχουσα τιμή *lastLoopUsefulTime* είναι μικρότερη από 10ms προστίθεται μια καθυστέρηση ίση με 10ms - *lastLoopUsefulTime* . Αυτό αναγκάζει αποτελεσματικά, κάθε φορά τον κύκλο βρόχου να ισούται με 10 ms, ώστε να γνωρίζουμε ότι κάθε κύκλος έχει την ίδια διάρκεια.
Στη συνέχεια επανατοποθετούνται ανάλογα οι τιμές των μεταβλητών μας μέχρι ξανακληθεί η υπορουτίνα.

```
lastLoopTime = millis() - loopStartTime;

 loopStartTime = millis();
```

Κεφάλαιο 4

4.1 Δοκιμές.

Κατά τις δοκιμές διαπιστώθηκαν μερικά σχεδιαστικά και λειτουργικά προβλήματα όπως το ότι η κατανομή του βάρους των τμημάτων του οχήματος δεν είχε κατανεμηθεί σωστά με αποτέλεσμα το όχημα να μην σταθεροποιείτε καθώς τα τμήματα προκαλούσαν συνεχή κλίση στο πλαίσιο. Η μετακίνηση των μπαταριών σε πιο ακραία θέση διόρθωσε αισθητά το πρόβλημα αυτό.
Η ταχύτητα των κινητήρων ήταν χαμηλή σε σχέση με τις απαιτήσεις του συστήματος με αποτέλεσμα σε έντονες ωθήσεις το όχημα να μην μπορεί να ανταπεξέλθει και να ισορροπήσει.
Η διάμετρος των τροχών έπρεπε να ήταν μεγαλύτερη καθώς το πλαίσιο σε κλίση μεγαλύτερη των 15 μοιρών ακουμπούσε στο έδαφος. Αυτό όμως, αν και όχι τόσο χρηστικό, καθώς και το πρόβλημα των κινητήρων θα πραγματοποιηθεί σε μελλοντική αναβάθμιση του οχήματος.

4.2 Συμπεράσματα

Το έργο ήταν επιτυχές στην επίτευξη των στόχων του για την εξισορρόπηση ενός αυτόνομου δικύκλου Segway με βάση το μοντέλο ανεστραμμένου εκκρεμούς.

Το κέρδος των συντελεστών του ελεγκτή PID (Kp, Ki, Kd) κατά τις δοκιμές σε πραγματικό χρόνο έδειξαν πολλά υποσχόμενα αποτελέσματα στην εξισορρόπηση του οχήματος καθώς είναι το όχημα είναι ικανό να διατηρήσει την κατακόρυφη θέση του ρυθμίζοντας ελαφρώς τους τροχούς του.

Το Complementary φίλτρο εφαρμόστηκε με επιτυχία. Η ολίσθηση του γυροσκόπιου εξουδετερώθηκε αποτελεσματικά καθώς το όχημα παρέμεινε σταθερά στην ίδια θέση σε επίπεδο έδαφος για μεγάλο χρονικό διάστημα.

Οι κανόνες ρύθμισης που χρησιμοποιούνται για τον ελεγκτή PID είναι απλοί, απαιτούν λίγες γνώσεις των διαδικασιών και μπορούν να εφαρμοστούν με μέτρια προσπάθεια. Ωστόσο, για μια ακριβή και στιβαρή λειτουργία, μπορεί να μοντελοποιηθεί η δυναμική του ελέγχου για να είναι δυνατή η σχεδίαση ενός πιο κατάλληλου ελεγκτή.

Η ταχύτητα των κινητήρων, αν και χαμηλή, είναι ικανή να κρατήσει το όχημα σε ισορροπία ακόμα και με την επιβολή εξωτερικών μικρών ωθήσεων του πλαισίου του.

4.3 Μελλοντική δουλειά

Αφού υλοποιήθηκε η κατασκευή του οχήματος και έγιναν οι απαραίτητοι έλεγχοι διαπιστώθηκε πως είναι απαραίτητο να γίνουν κάποιες βελτιώσεις στο σύστημα.

Ακολουθούν μερικές μελλοντικές ιδέες βελτίωσης για το Segway της εργασίας μας.

- Κύκλωμα παρακολούθησης τάσης. Αυτό γίνεται με ένα ζεύγος αντιστάσεων που δημιουργούν ένα διαιρέτη τάσης κατεβάζοντας τα 12V της μπαταρίας σε 5V και θα παρακολουθούνταν από ένα ADC στο Arduino (η είσοδος A7 είναι διαθέσιμη) . Εάν η τάση πέσει κάτω από ένα προκαθορισμένο όριο για μια ορισμένη χρονική περίοδο, μια λυχνία LED θα αναβοσβήνει με ρυθμό 2Hz για να μας ενημερώσει για την ανάγκη επαναφόρτισης της μπαταρίας..
- Αντικατάσταση των μπαταριών μολύβδου οξέος με φτηνού τύπου LIPO ή LiFePO4.

- Αντικατάσταση των κινητήρων με ταχύτερους και μεγαλύτερης ροπής για μια πιο άνετη και ευχάριστη οδήγηση.
- Προσθήκη κωδικοποιητών στους τροχούς για υπολογισμό ταχύτητας και διανυθείσας απόστασης.
- Απαιτείται περισσότερη έρευνα της δυναμικής λειτουργίας του συστήματος για τη βελτίωση της σταθερότητας του, έτσι ώστε να μπορούν να εξαλειφθούν οι πιθανές μικρές ταλαντευτικές κινήσεις του οχήματος

Βιβλιογραφία

SEGWAY
https://en.wikipedia.org/wiki/Segway_PT

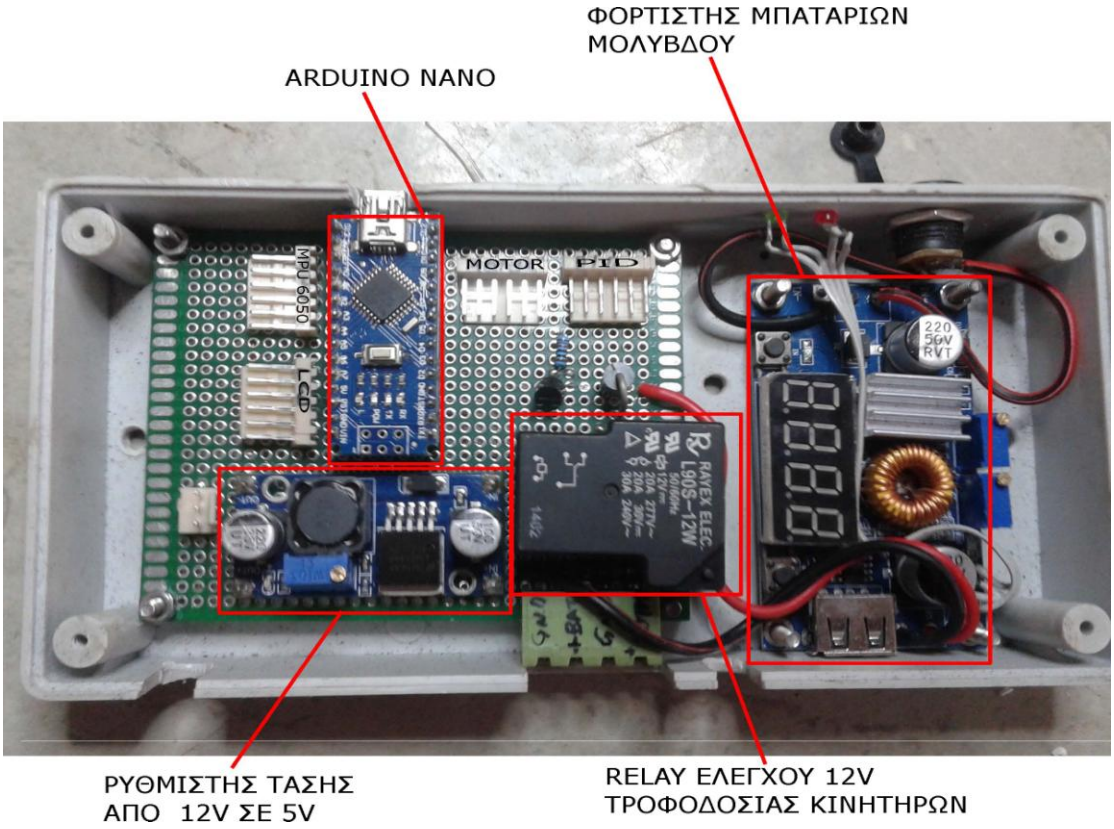
Arduino
http://arduino.cc/en/Guide/Environment?from=Tutorial.Bootloader
http://arduino.cc/en/Main/Software.

BTS7960 dual motor controller
http://www.robotpower.com/downloads/BTS7960_v1.1_2004-12-07.pdf

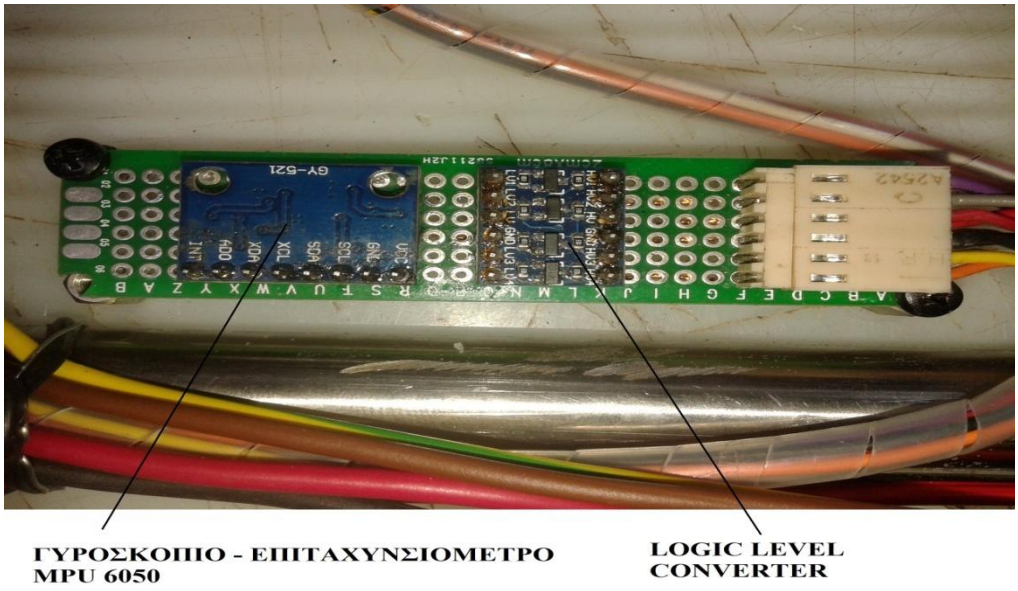
Γυροσκόπιο-επιταχυνσιόμετρο MPU6050
http://www.instructables.com/id/How-to-Measure-Angle-With-MPU-6050GY-521/
http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/

Ελεγκτής PID
https://en.wikipedia.org/wiki/PID_controller
https://www.csimn.com/CSI_pages/PIDforDummies.html
http://www.ni.com/white-paper/3782/en/

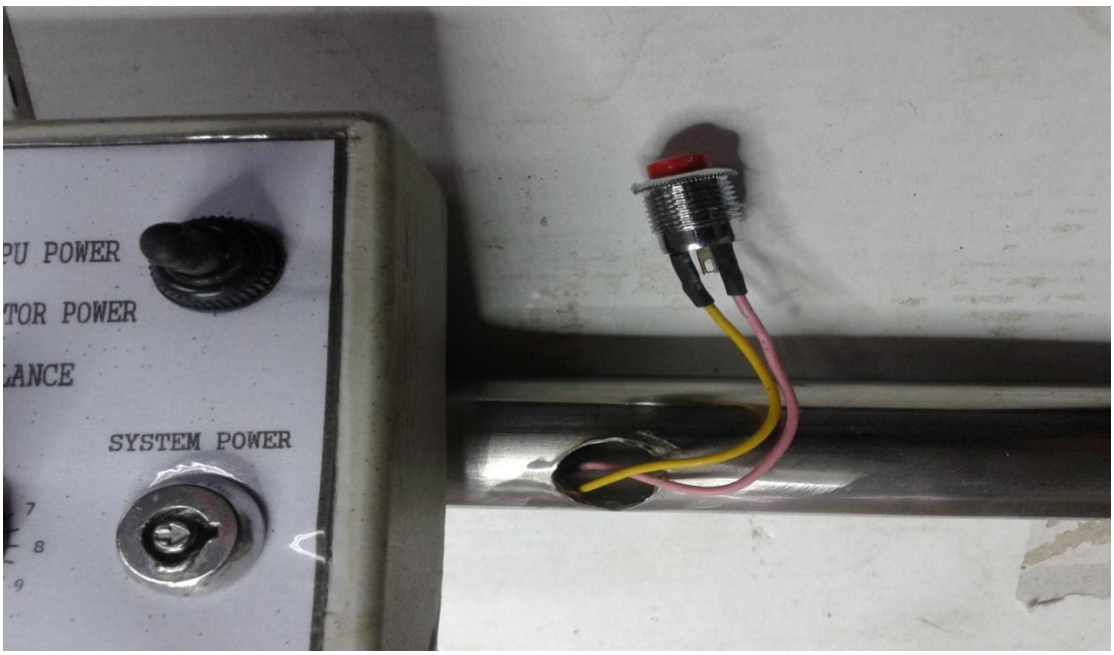
Παράρτημα Α. Φωτογραφίες κατασκευής



Η μονάδα κεντρικού ελέγχου του οχήματος



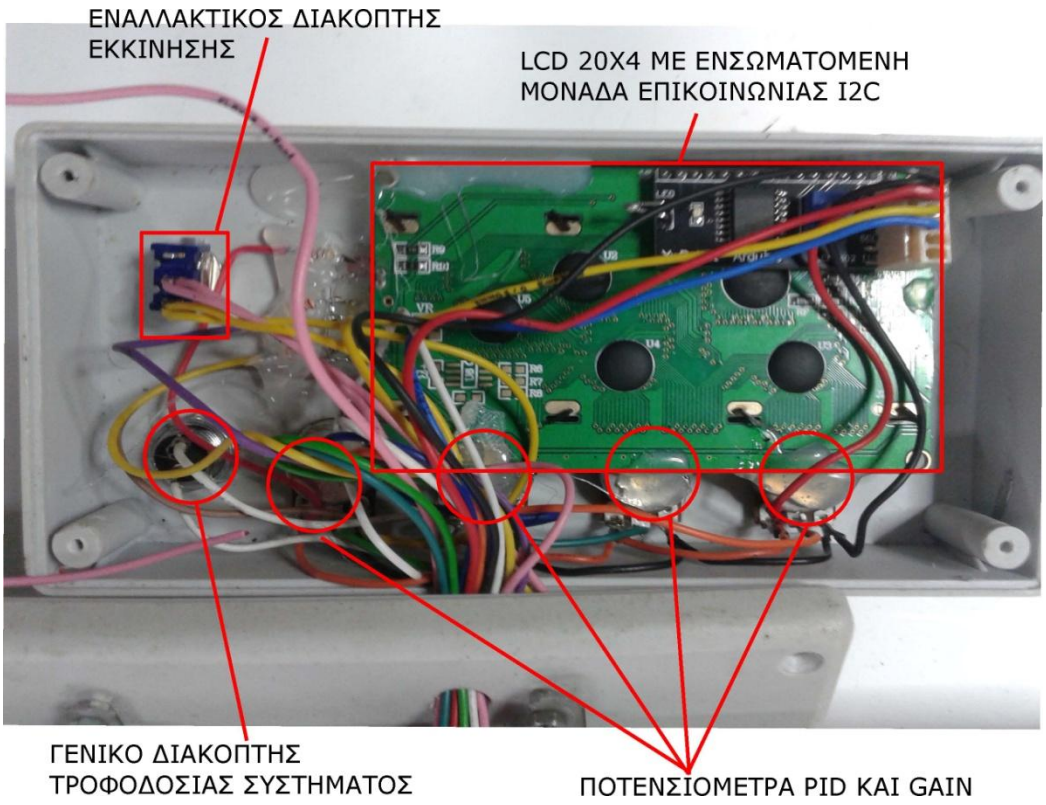
Η μονάδα γυροσκόπιου - επιταχυνσιόμετρου



Τοποθέτηση διακόπτη εκκίνησης στο τιμόνι

Τα ελατήρια επαναφοράς τιμονιού σε όρθια θέση

Η μονάδα ενδείξεων –ρυθμίσεων



Το εσωτερικό της μονάδας ενδείξεων –ρυθμίσεων

Το πάνω τμήμα του τιμονιού με την μονάδα ενδείξεων -ρυθμίσεων

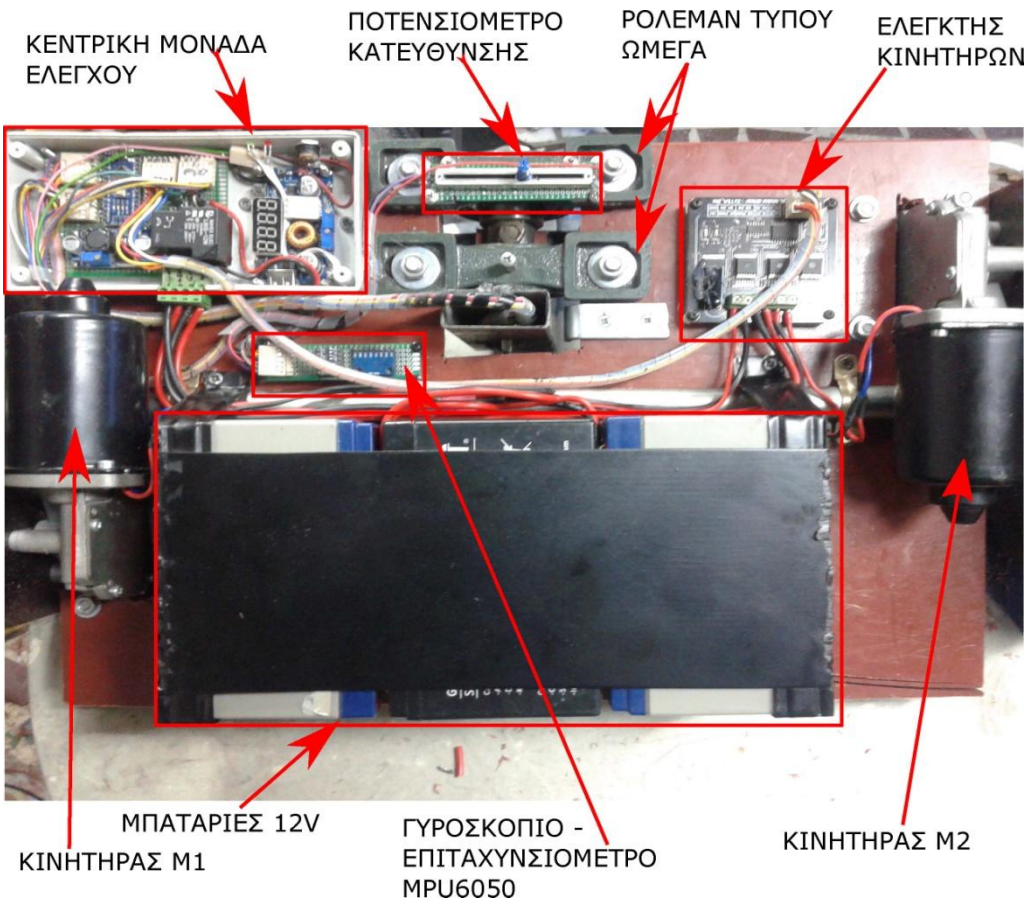
Το κάτω τμήμα του τιμονιού με τον άξονα περιστροφής

Μηχανισμός στήριξης – περιστροφής τιμονιού

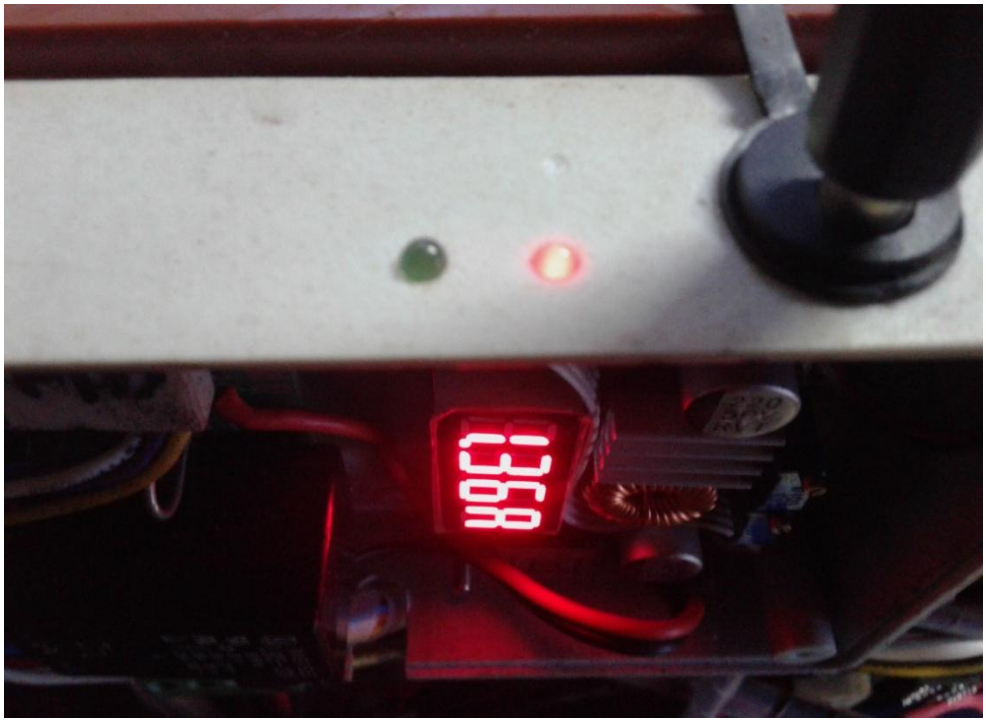
Οι ρόδες, τα γρανάζια, οι αλυσίδες και ο άξονας στήριξης του οχήματος

Μηχανισμός μετάδοσης κίνησης

Η βάση στήριξης – ασφάλισης μπαταριών



Η κάτω πλευρά του οχήματος



Ο φόρτισης μπαταριών την στιγμή της φόρτισης

Το Segway της εργασίας

Παράρτημα Β. Κώδικας αντιστάθμισης (offset) του MPU6050

```
// Εξαγωγή αντισταθμιστικών τιμών βαθμονόμησης για την MPU6050
// Πρέπει να εγκατασταθούν οι βιβλιοθήκες I2Cdev και MPU6050

#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

//////////////////////////////////  CONFIGURATION  //////////////////////////////
//Αλλάξτε αυτές τις 3 μεταβλητές αν θέλετε να συντονίσετε το σχέδιο στις ανάγκες σας.

int buffersize=1000;    //Πλήθος μετρήσεων για υπολογισμό του μέσου όρου.
int acel_deadzone=8;    //Επιτρεπόμενο σφάλμα επιταχυνσιόμετρου, ¨Όσο μικρότερο τόσο μεγαλύτερη ακρίβεια.
int giro_deadzone=1;    //Επιτρεπόμενο σφάλμα γυροσκόπιου. Όσο μικρότερο τόσο μεγαλύτερη ακρίβεια.

// Προεπιλεγμένη διεύθυνση I2C είναι 0x68
// Ειδική διεύθυνση I2C μπορεί να εισαχθεί εδώ ως παράμετρος.
// AD0 low = 0x68 (προεπιλογή της πλακέτας InvenSense)
// AD0 high = 0x69
//MPU6050 accelgyro;
MPU6050 accelgyro(0x68);
 int16_t ax, ay, az,gx, gy, gz;
 int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz,state=0;
int ax_offset,ay_offset,az_offset,gx_offset,gy_offset,gz_offset;

/////////////////////////////////////  SETUP  ////////////////////////////////////
void setup() {

        // σύνδεση στο I2C bus (η βιβλιοθήκη I2Cdev δεν το κάνει αυτόματα
 Wire.begin();
  // COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE
  TWBR = 24; // Χρονισμός I2C 400kHz καθώς η CPU είναι 16MHz).

  // Εναρξη σειριακής επικοινωνίας
  Serial.begin(115200);

  // Εναρξη επικοινωνίας της συσκευής
  accelgyro.initialize();

  // Αναμονή για αποστολή οποιουδήποτε χαρακτήρα για έναρξη
  while (Serial.available() && Serial.read()); //Άδειασμα μνήμης buffer
  while (!Serial.available()){
    Serial.println(F("Send any character to start sketch.\n"));
    delay(1500);
  }
  while (Serial.available() && Serial.read()); // Αδειασμα ξανά της buffer

  // Εναρξη εμφάνισης πληροφοριών στη σειριακή οθόνη.
  Serial.println("\nMPU6050 Calibration Sketch");
  delay(2000);
  Serial.println("\nYour MPU6050 should be placed in horizontal position, with package letters facing up. \nDon't touch it until
you see a finish message.\n");
  delay(3000);
  // Επιβεβαίωση επικοινωνίας συσκευής
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");
  delay(1000);
  // Μηδενισμός offsets
  accelgyro.setXAccelOffset(0);
  accelgyro.setYAccelOffset(0);
  accelgyro.setZAccelOffset(0);
  accelgyro.setXGyroOffset(0);
  accelgyro.setYGyroOffset(0);
  accelgyro.setZGyroOffset(0);
}

/////////////////////////////////////  LOOP  ////////////////////////////////////
void loop() {
  if (state==0){
```

```
  Serial.println("\nReading sensors for first time...");
  meansensors();
  state++;
  delay(1000);
 }

 if (state==1) {
  Serial.println("\nCalculating offsets...");
  calibration();
  state++;
  delay(1000);
 }

 if (state==2) {
  meansensors();
  Serial.println("\nFINISHED!");
  Serial.print("\nSensor readings with offsets:\t");
  Serial.print(mean_ax);
  Serial.print("\t");
  Serial.print(mean_ay);
  Serial.print("\t");
  Serial.print(mean_az);
  Serial.print("\t");
  Serial.print(mean_gx);
  Serial.print("\t");
  Serial.print(mean_gy);
  Serial.print("\t");
  Serial.println(mean_gz);
  Serial.print("Your offsets:\t");
  Serial.print(ax_offset);
  Serial.print("\t");
  Serial.print(ay_offset);
  Serial.print("\t");
  Serial.print(az_offset);
  Serial.print("\t");
  Serial.print(gx_offset);
  Serial.print("\t");
  Serial.print(gy_offset);
  Serial.print("\t");
  Serial.println(gz_offset);
  Serial.println("\nData is printed as: acelX acelY acelZ giroX giroY giroZ");
  Serial.println("Check that your sensor readings are close to 0 0 16384 0 0 0");
  Serial.println("If calibration was succesful write down your offsets so you can set them in your projects using something
similar to mpu.setXAccelOffset(youroffset)");
  while (1);
 }
}

////////////////////////////////////   FUNCTIONS  ////////////////////////////////////
void meansensors(){
 long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;

 while (i<(buffersize+101)){
  //Ανάγνωση ακατέργαστων τιμών επιτάχυνσιόμετρου / γυροσκόπιου
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  if (i>100 && i<=(buffersize+100)){   //Οι 100 πρώτες μετρήσεις απορήπτονται
    buff_ax=buff_ax+ax;
    buff_ay=buff_ay+ay;
    buff_az=buff_az+az;
    buff_gx=buff_gx+gx;
    buff_gy=buff_gy+gy;
    buff_gz=buff_gz+gz;
  }
  if (i==(buffersize+100)){
    mean_ax=buff_ax/buffersize;
    mean_ay=buff_ay/buffersize;
    mean_az=buff_az/buffersize;
    mean_gx=buff_gx/buffersize;
    mean_gy=buff_gy/buffersize;
    mean_gz=buff_gz/buffersize;
```

```
   }
   i++;
   delay(2);    //Η καθυστέρηση χρειάζεται, ώστε να μην λαμβάνουμε επαναλαμβανόμενες μετρήσεις
  }
}

void calibration(){
  ax_offset=-mean_ax/8;
  ay_offset=-mean_ay/8;
  az_offset=(16384-mean_az)/8;

  gx_offset=-mean_gx/4;
  gy_offset=-mean_gy/4;
  gz_offset=-mean_gz/4;
  while (1){
    int ready=0;
    accelgyro.setXAccelOffset(ax_offset);
    accelgyro.setYAccelOffset(ay_offset);
    accelgyro.setZAccelOffset(az_offset);

    accelgyro.setXGyroOffset(gx_offset);
    accelgyro.setYGyroOffset(gy_offset);
    accelgyro.setZGyroOffset(gz_offset);

    meansensors();
    Serial.println("...");

    if (abs(mean_ax)<=acel_deadzone) ready++;
    else ax_offset=ax_offset-mean_ax/acel_deadzone;

    if (abs(mean_ay)<=acel_deadzone) ready++;
    else ay_offset=ay_offset-mean_ay/acel_deadzone;

    if (abs(16384-mean_az)<=acel_deadzone) ready++;
    else az_offset=az_offset+(16384-mean_az)/acel_deadzone;

    if (abs(mean_gx)<=giro_deadzone) ready++;
    else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);

    if (abs(mean_gy)<=giro_deadzone) ready++;
    else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);

    if (abs(mean_gz)<=giro_deadzone) ready++;
    else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);

    if (ready==6) break;
  }
}
```

Παράρτημα Γ. Ο πηγαίος κώδικας της εργασίας.

```
/*
// ================================================
// ===         SEGWAY CLONE  2018            ===
// ================================================


 ΣΥΝΔΕΣΕΙΣ Arduino UNO ή NANO
 ----------------------------------------
 *| A0   --> STEER POT
 *| A1   --> P POT  Kp
 *| A2   --> I POT  Ki
 *| A3   --> D POT  Kd
 *| A4   --> SDA MPU6050 & SDA LCD
 *| A5   --> SCL MPU6050 & SCL LCD
 *| A6   -->
 *| A7   -->

 *| D2  --> LA_IS & RA_IS MOTOR M1  /  ΕΛΕΓΧΟΣ ΡΕΥΜΑΤΟΣ ΚΙΝΗΤΗΡΑ M1
 *| D3  --> PWM_M1
 *| D4  --> LB_IS & RB_IS MOTOR M2  / Ε ΛΕΓΧΟΣ ΡΕΥΜΑΤΟΣ ΚΙΝΗΤΗΡΑ M2
 *| D5  -->
 *| D6  --> DIR_M2
 *| D7  --> DIR_M1
 *| D8  --> RELAY CONTROL
 *| D9
 *| D10 --> BUZZER
 *| D11 --> PWM_M2
 *| D12 --> START SWITCH
 *| D13 --> LED
 ----------------------------------------
*/

#include<Wire.h>
#include <I2Cdev.h>
#include <MPU6050_6Axis_MotionApps20.h>
#include <LiquidCrystal_I2C.h>

MPU6050 mpu;       // ΠΡΟΡΥΘΜΙΣΜΕΝΟ ΜΕ AD0 low = ΔΙΕΥΘΥΝΣΗ 0x68
const int MPU_addr=0x68;
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // Set the LCD I2C address


double AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ; // ΑΝΕΠΕΞΕΡΓΑΣΤΑ ΔΕΔΟΜΕΝΑ ΑΠΟ ΤΗΝ MPU6050.
uint32_t timer; //ΧΡΟΝΙΣΤΗΣ. ΧΡΗΣΗΜΟΠΟΙΗΤΕ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΔΙΑΡΚΕΙΑΣ ΒΡΟΓΧΟΥ ΚΤΛ.
double compAngleX, compAngleY; //ΓΩΝΙΕΣ X,Y ΜΕΤΑ ΤΗΝ ΕΦΑΡΜΟΓΗ ΤΟΥ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟΥ ΦΙΛΤΡΟΥ (COMPLEMENTARY
FILTER)

#define degconvert 57.2957786  //57.2957786 = 1 / (3.142 / 180)    ΜΕΤΑΤΡΟΠΗ RAD ΣΕ ΜΟΙΡΕΣ

#define DIR_M1 7     // ΑΚΙΔΑ ΚΑΤΕΥΘΥΝΣΗΣ ΚΙΝΗΤΗΡΑ Α (ΑΡΙΣΤΕΡΟΣ ΚΙΝΗΤΗΡΑΣ)
#define DIR_M2 6     // ΑΚΙΔΑ ΚΑΤΕΥΘΥΝΣΗΣ ΚΙΝΗΤΗΡΑ Β (ΔΕΞΙΟΣ ΚΙΝΗΤΗΡΑΣ)
#define PWM_M1 3     // ΑΚΙΔΑ PWM, ΤΑΧΥΤΗΤΑ ΚΙΝΗΤΗΡΑ Α
#define PWM_M2 11     // ΑΚΙΔΑ PWM, ΤΑΧΥΤΗΤΑ ΚΙΝΗΤΗΡΑ Β

//#define BRK_A 9   // ΦΡΕΝΟ ΚΙΝΗΤΗΡΑ Α
//#define BRK_B 8   // ΦΡΕΝΟ ΚΙΝΗΤΗΡΑ Β

#define Buzz_Pin 10  //ΑΚΙΔΑ ΒΟΜΒΙΤΗ


// ΑΚΙΔΕΣ  I/O
int ledPin = 13;
int counter=0;
int Steer_Pot = 0; // ΣΥΝΔΕΣΗ ΠΟΤΕΝΣΙΟΜΕΤΡΟΥ ΚΑΤΕΥΘΥΝΣΗΣ ΣΤΗΝ ΑΝΑΛΟΓΙΚΗ ΑΚΙΔΑ 2
int Start_switch = 12; // ΔΙΑΚΟΠΤΗΣ ΕΚΚΙΝΗΣΗΣ ΣΤΗ ΑΚΙΔΑ 12
int Motor_power = 8;  // ΕΛΕΓΧΟΣ ΤΑΣΗΣ ΚΙΝΗΤΗΡΩΝ ΜΕ ΡΕΛΕ
```

```
// ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΕΛΕΓΧΟ ΚΙΝΗΤΗΡΩΝ
int left_PWM, right_PWM;   // ΜΕΤΑΒΛΗΤΕΣ ΤΑΧΥΤΗΤΑΣ ΚΙΝΗΤΗΡΩΝ
//int direction_A = 0;      // ΚΑΤΕΥΘΥΝΣΗ 0 - ΕΜΠΡΟΣ,  1 - ΠΙΣΩ (ΑΡΙΣΤΕΡΟΣ ΚΙΝΗΤΗΡΑΣ)
//int direction_B = 0;      // ΚΑΤΕΥΘΥΝΣΗ 0 - ΕΜΠΡΟΣ,  1 - ΠΙΣΩ (ΔΕΞΙΟΣ ΚΙΝΗΤΗΡΑΣ)
//int brake_A = 1;          // ΦΡΕΝΟ   1 - ΕΝΕΡΓΟ,  0 - ΑΝΕΝΕΡΓΟ  (ΑΡΙΣΤΕΡΟΣ ΚΙΝΗΤΗΡΑΣ)
//int brake_B = 1;          // ΦΡΕΝΟ   1 - ΕΝΕΡΓΟ,  0 - ΑΝΕΝΕΡΓΟ  (ΔΕΞΙΟΣ ΚΙΝΗΤΗΡΑΣ)

// ΣΥΝΔΕΣΗ ΠΑΡΑΛΛΗΛΑ ΤΩΝ LA_IS ΚΑΙ RA_IS ΣΕ UNO digital 2
// ΣΥΝΔΕΣΗ ΠΑΡΑΛΛΗΛΑ ΤΩΝ LB_IS ΚΑΙ RB_IS ΣΕ UNO digital 4
int val1=digitalRead(2);   // ΕΛΕΓΧΟΣ ΡΕΥΜΑΤΟΣ ΚΙΝΗΤΗΡΑ Μ1
int val2=digitalRead(4);   // ΕΛΕΓΧΟΣ ΡΕΥΜΑΤΟΣ ΚΙΝΗΤΗΡΑ Μ2

// ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΟΝ ΕΛΕΓΧΤΗ PID

   float Kp = 0;        //ΠΑΡΑΜΕΤΡΟΣ (P)roportional
   float Ki = 0;        //ΠΑΡΑΜΕΤΡΟΣ (I)ntegral
   float Kd = 0;        //ΠΑΡΑΜΕΤΡΟΣ (D)erivative

float lastpitch;      // ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΣΦΑΛΜΑΤΟΣ ΜΕ ΤΗΝ ΠΑΡΟΔΟ ΤΟΥ ΧΡΟΝΟΥ.

float iTerm;                        // ΓΙΑ ΤΗΝ ΣΥΣΣΩΡΕΥΣΗ ΣΦΑΛΜΑΤΟΣ (Intergral)
float targetAngle = 0;              // ΓΩΝΙΑ ΣΤΟΧΟΣ (ΜΠΟΡΕΙ ΝΑ ΡΥΘΜΙΣΤΕΙ ΣΥΜΦΩΝΑ ΜΕ ΤΟ ΚΕΝΤΡΟ ΒΑΡΥΤΗΤΑΣ)
float PIDGain = 0.4;                // ΧΡΗΣΗΜΟΠΟΙΗΤΕ ΓΙΑ ΤΗΝ ΟΜΑΛΗ ΕΚΚΙΝΗΣΗ ΤΟΥ ΟΧΗΜΑΤΟΣ
#define degconvert 57.2957786  //ΓΙΑ ΤΗΝ ΜΕΤΑΤΡΟΠΗ ΑΚΤΙΝΙΩΝ ΣΕ ΜΟΙΡΕΣ (ΜΟΙΡΕΣ = ΑΚΤΙΝΙΑ *180/π = 180/3.14159)

// ΜΕΤΑΒΛΗΤΕΣ ΠΟΤΕΝΣΙΟΜΕΤΡΩΝ
int Steering;
int Steer;


// ΜΕΤΑΒΛΗΤΕΣ ΓΙΑ ΤΟΝ ΕΛΕΓΧΟ ΧΡΟΝΩΝ
#define STD_LOOP_TIME 9  //ΣΤΟΧΟΣ ΤΑ 10ms ΣΕ ΚΑΘΕ ΒΡΟΓΧΟ (100Hz)
#define STD_CALIBRATE_TIME 40500   // ΧΡΟΝΟΣ ΒΑΘΜΟΝΟΜΗΣΗΣ ΕΠΙΤΑΧΥΝΣΙΟΜΕΤΡΟΥ - ΓΥΡΟΣΚΟΠΟΙΟΥ
int lastLoopTime = STD_LOOP_TIME;
int lastLoopUsefulTime = STD_LOOP_TIME;
unsigned long loopStartTime = 0;
unsigned long thisTime;             // ΕΛΕΓΧΟΣ ΟΛΟΚΛΗΡΩΣΗΣ ΧΡΟΝΟΥ ΒΑΘΜΟΝΟΜΗΣΗΣ
unsigned long lastTime;             // ΕΛΕΓΧΟΣ ΔΙΑΡΚΕΙΑΣ ΤΕΛΕΥΤΑΙΑΣ ΚΛΗΣΗΣ ΤΟΥ PID (ΜΕΓΙΣΤΟ 10ms)
int looptimeslow = 0;               // ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΛΚΛΙΣΗΣ ΧΡΟΝΟΥ ΒΡΟΓΧΟΥ
int looptimefast = 0;               // ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΚΛΙΣΗΣ ΚΑΘΥΣΤΕΡΗΣΗΣ ΒΡΟΓΧΟΥ
int looptimeaverage = 0;            // ΣΥΝΟΛΙΚΟΣ ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΚΛΙΣΗΣ

// ΜΕΤΑΒΛΗΤΕΣ ΔΙΑΚΟΠΤΗ ΕΚΚΙΝΗΣΗΣ
int Start = false;
int Start_state;


// ================================================
// ===              SETUP               ===
// ================================================

void setup() {

  pinMode(Buzz_Pin, OUTPUT);
  digitalWrite(Buzz_Pin, HIGH);
    pinMode(Motor_power, OUTPUT);
    digitalWrite(Motor_power, LOW);

  pinMode(DIR_M1, OUTPUT);
  pinMode(DIR_M2, OUTPUT);
  pinMode(PWM_M1, OUTPUT);
    digitalWrite(PWM_M1, 0);
  pinMode(PWM_M2, OUTPUT);
    digitalWrite(PWM_M2, 0);


  Serial.begin(115200);  // ΕΚΙΝΗΣΗ ΣΕΙΡΙΑΚΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ

  lcd.begin(20,4);
  lcd.clear();
  lcd.setCursor(0,0); //ΕΚΚΙΝΗΣΗΣ LCD ΣΤΗ ΓΡΑΜΜΗ 0 ΣΤΗΛΗ 0
```

```
 lcd.print("SEGWAY CLONE PROJECT"); //ΕΜΦΑΝΙΣΗ "SEGWAY CLONE PROJECT" ΣΤΗΝ LCD ΟΘΟΝΗ
 delay(1000);  //ΚΑΘΥΣΤΕΡΗΣΗ 1000ms
 lcd.setCursor(3,1);
 lcd.print("BALIS AGAPITOS");
 delay(500);
 lcd.setCursor(1,2);
 lcd.print("PATERAKIS MICHALIS");
 delay(500);
 lcd.setCursor(8,3);
 lcd.print("2017");
 delay(2000);
 lcd.clear();

 // Set up MPU 6050:
 Wire.begin();  // ΕΚΚΙΝΗΣΗ ΔΙΑΥΛΟΥ ΕΠΙΚΟΙΝΩΝΙΑΣ I2C

 #if ARDUINO >= 157
 Wire.setClock(400000UL); // ΡΥΘΜΙΣΗ ΣΥΧΝΟΤΗΤΑΣ ΔΙΑΥΛΟΥ I2C ΣΤΑ 400kHz
 #else
  TWBR = ((F_CPU / 400000UL) - 16) / 2; // ΡΥΘΜΙΣΗ ΣΥΧΝΟΤΗΤΑΣ ΔΙΑΥΛΟΥ I2C ΣΤΑ 400kHz
 #endif

setup_mpu_6050_registers();

   // ΕΠΙΒΕΒΑΙΩΣΗ ΣΩΣΤΗΣ ΣΥΝΔΕΣΗΣ MPU6050 ΜΕ ARDUINO "mpu.testConnection()"
   lcd.setCursor(0,0);
   lcd.print("MPU6050 connections.");
   delay(500);
  lcd.setCursor(5,1);
  lcd.print(mpu.testConnection() ? "SUCCESS !!!" : " failed !!!");


// ΓΙΑ ΚΑΘΕ ΔΙΑΦΟΡΕΤΙΚΗ MPU6050 ΠΡΕΠΕΙ ΝΑ ΑΛΛΑΧΘΟΥΝ ΤΑ OFFSETS
// ΑΦΟΥ ΤΟΠΟΘΕΤΗΣΟΥΜΕ ΤΗΝ MPU6050 ΣΤΗΝ ΤΕΛΙΚΗ ΘΕΣΗ ΤΗΣ ΚΑΙ ΤΟ ΟΧΗΜΑ ΕΙΝΑΙ
// ΣΕ ΘΕΣΗ ΙΣΟΡΟΠΟΙΑΣ ΦΟΡΤΩΝΟΥΜΕ  ΤΟ ΣΧΕΔΙΟ OFFSETS ΚΑΙ ΠΕΡΝΟΥΜΕ ΤΙΣ ΤΙΜΕΣ
// ΠΟΥ ΘΑ ΕΙΣΑΓΟΥΜΕ ΠΑΡΑΚΑΤΩ

   mpu.setXGyroOffset(78);
   mpu.setYGyroOffset(-3);
   mpu.setZGyroOffset(-22);
   mpu.setXAccelOffset(-1452);
   mpu.setYAccelOffset(471);
   mpu.setZAccelOffset(1842);

 delay(500);
 lcd.setCursor(0,3);
 lcd.print("MPU Offsets -->  OK");
 delay(2000);

pinMode(Start_switch, INPUT);
digitalWrite(Start_switch, HIGH);  // ΘΕΤΟΥΜΕ ΤΗΝ ΕΣΩΤΕΡΙΚΉ pull-up ΑΝΤΙΣΤΑΣΗ ΤΟΥ Arduino ΣΤΗΝ ΑΚΙΔΑ Start_switch.

 lcd.clear();
 lcd.setCursor(1,0);
 lcd.print("SEGWAY CLONE READY ");
 lcd.setCursor(3,1);
 lcd.print("TO BALANCE !!!");
 lcd.setCursor(0,2);
 lcd.print("HOLD START BUTTON ");
 lcd.setCursor(0,3);
 lcd.print("TO RUN...  ENJOY!!!");

 timer = micros(); //ΕΚΚΙΝΗΣΗ ΧΡΟΝΟΜΕΤΡΟΥ
}


// ======================================================
// ===              MAIN PROGRAM                ===
// ======================================================

void loop() {
```

```
setup_mpu_6050_registers();

read_mpu_6050_data(); //ΚΛΙΣΗ ΥΠΟΡΟΥΤΙΝΑΣ ΣΥΛΟΓΗΣ ΔΕΔΩΜΕΝΩΝ ΑΠΟ ΤΗΝ MPU6050

Gyro_Acc_angle_calculations (); //ΚΛΙΣΗ ΥΠΟΡΟΥΤΙΝΑΣ ΥΠΟΛΟΓΙΣΜΟΥ ΓΩΝΙΩΝ ΕΠΙΤΑΧΥΝΣΙΟΜΕΤΡΟΥ - ΓΥΡΟΣΚΟΠΙΟΥ

read_pots(); //ΚΛΙΣΗ ΥΠΟΡΟΥΤΙΝΑΣ

//SOFT START();

auto_level(); //ΚΛΙΣΗ ΥΠΟΡΟΥΤΙΝΑΣ ΕΛΕΓΧΟΥ ΟΡΘΗΣ ΕΚΚΙΝΗΣΗΣ ΟΧΗΜΧΤΟΣ

PID_and_Movenent(); // ΚΛΙΣΗ ΥΠΟΡΟΥΤΙΝΑΣ ΥΠΟΛΟΓΙΣΜΟΥ PID

 //  Serial.print("direction_A = ");
 //  Serial.print(direction_A);Serial.print("\t");
 //  Serial.print("direction_B = ");
 //  Serial.print(direction_B);Serial.print("\t");

//current_sense(); // ΕΛΕΓΧΟΣ ΥΠΕΡ-ΡΕΥΜΑΤΟΣ ΣΤΟΥΣ ΚΙΝΗΤΗΡΕΣ

timekeeper();  //ΚΛΙΣΗ ΥΠΟΡΟΥΤΙΝΑΣ ΧΡΟΝΟΜΕΤΡΗΣΗΣ

} // ΤΕΛΟΣ ΚΥΡΙΟΥ ΒΡΟΓΧΟΥ


// ============================================================
// ===          MPU6050 REGISTERS SETUP              ===
// ============================================================

void setup_mpu_6050_registers(){
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);    // ΘΕΤΟΝΤΑΣ 0 ΞΥΠΝΗΜΑ MPU-6050
  Wire.endTransmission(true);
}


// ============================================================
// ===             MPU DATA READING                 ===
// ============================================================

void read_mpu_6050_data(){
  //setup starting angle
  //1) collect the data
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // ΕΚΚΙΝΗΣΗ ΜΕ ΚΑΤΑΧΩΡΙΤΗ 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);  // ΑΙΤΗΣΗ ΕΞΑΓΩΓΗΣ ΔΕΔΟΜΕΝΩΝ ΑΠΟ 14 ΚΑΤΑΧΩΡΗΤΕΣ (7 HIGH + 7 LOW)
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
}

// ============================================================
// ===             GYRO - ACCEL  CALCULATIONS           ===
// ============================================================

void Gyro_Acc_angle_calculations (){

  double dt = (double)(micros() - timer) / 1000000; //ΕΔΩ ΕΚΤΕΛΟΥΝΤΑΙ 3 ΛΕΙΤΟΥΡΓΙΕΣ
                                    //1)ΣΤΑΜΑΤΗΜΑ ΧΡΟΝΟΜΕΤΡΟΥ
                                    //2)ΜΕΤΑΤΡΟΠΗ ΔΕΔΩΜΕΝΩΝ ΧΡΟΝΟΜΕΤΡΟΥ ΑΠΟ microseconds ΣΕ seconds
                                    //3)ΑΠΟΘΗΚΕΥΣΗ ΤΗΣ ΤΙΜΗΣ ΩΣ double "dt".

  timer = micros(); //ΕΠΑΡΥΘΜΙΣΗ ΤΟΥ ΧΡΟΝΟΜΕΤΡΟΥ ΓΙΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΟΥ ΕΠΟΜΕΝΟΥ dt.
```

```
 //ΣΤΙΣ ΕΠΟΜΕΝΕΣ ΔΥΟ ΓΡΑΜΜΕΣ 1)ΥΠΟΛΟΓΙΖΕΤΑΙ Ο ΠΡΟΣΑΝΑΤΟΛΙΣΜΟΣ ΕΠΙΤΑΧΥΝΣΙΟΜΕΤΡΟΥ ΣΕ ΣΧΕΣΗ ΜΕ ΤΗΝ ΓΗ και
 //                  2)ΜΕΤΑΤΡΕΠΕΤΑΙ Η ΕΞΟΔΟΣ ΑΠΟ ΑΚΤΙΝΙΑ ΣΕ ΜΟΙΡΕΣ

 //ΤΑ ΔΕΔΟΜΕΝΑ ΘΑ ΧΡΗΣΗΜΟΠΟΙΗΘΟΥ ΓΙΑ ΤΗΝ ΔΙΟΡΘΩΣΗ ΤΥΧΩΝ ΣΥΣΩΡΕΥΤΙΚΩΝ ΣΦΑΛΜΑΤΩΝ ΣΤΟΝ ΠΡΟΣΑΝΑΤΟΛΙΣΜΟ
ΤΟΥ ΓΥΡΟΣΚΟΠΙΟΥ
 double roll = atan2(AcY, AcZ)*degconvert;  //accel angle Y
 double pitch = atan2(-AcX, AcZ)*degconvert;  //accel angle X
  // double roll = (atan2(-AcY, -AcZ)+PI)*degconvert;
  // double pitch = (atan2(-AcX, -AcZ)+PI)*degconvert;
//Serial.print("pitch = ");Serial.println(pitch);

 // ΤΟ ΓΥΡΟΣΚΟΠΙΟ ΠΑΡΕΧΕΙ ΕΞΟΔΟ ΓΩΝΙΑΚΗΣ ΤΑΧΥΤΗΤΑΣ. ΓΙΑ ΤΗΝ ΜΕΤΑΤΡΟΠΗ ΣΕ ΜΟΙΡΕΣ /ΔΕΥΤΕΡΟΛΕΠΤΟ  ΔΙΑΙΡΟΥΜΕ ΜΕ
131.0 ΣΥΜΦΩΝΑ ΜΕ ΤΙΣ ΟΔΗΓΙΕΣ ΤΟΥ ΚΑΤΑΣΚΕΥΑΣΤΗ.
 double gyroXrate = GyX/131.0;
 double gyroYrate = GyY/131.0;

 //ΤΟ ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ ΦΙΛΤΡΟ (COMPLEMENTARY FILTER)
 //ΑΥΤΟ ΤΟ ΦΙΛΤΡΟ ΥΠΟΛΟΓΙΖΕΙ ΤΗΝ ΓΩΝΙΑ ΒΑΣΙΖΟΜΕΝΟ ΣΤΗΝ ΕΝΣΩΜΑΤΩΣΗ ΤΗΣ ΓΩΝΙΑΚΗΣ ΤΑΧΥΤΗΤΑΣ ΣΕ ΚΑΘΕ ΓΩΝΙΑΚΗ
ΜΕΤΑΤΟΠΗΣΗ
 compAngleX = 0.98 * (compAngleX + gyroXrate * dt) + 0.02 * roll; // ΥΠΟΛΟΓΙΣΜΟΣ ΓΩΝΙΑΣ ΜΕ ΤΗ ΧΡΗΣΗ
ΣΥΜΠΛΗΡΩΜΑΤΙΚΟΥ ΦΙΛΤΡΟΥ.
 compAngleY = 0.98 * (compAngleY + gyroYrate * dt) + 0.02 * pitch;
}


// ============================================================
// ===               READ POTENSIOMETERS               ===
// ============================================================

void read_pots(){

   // ΓΙΑ ΤΟΝ ΕΛΕΓΧΤΗ PID
   Kp = map (analogRead(A1),0,1023,0,1000)/50.0; // ΤΙΜΗ ΑΠΟ 0 - 20 // 2.5; //ΠΑΡΑΜΕΤΡΟΣ (P)roportional    μετατροπη και
σε float
   Ki = map (analogRead(A2),0,1023,0,1000)/100.0; //ΤΙΜΗ ΑΠΟ 0 - 20  //0.5;  //ΠΑΡΑΜΕΤΡΟΣ (I)ntegral
   Kd = map (analogRead(A3),0,1023,0,1000)/100.0; //ΤΙΜΗ ΑΠΟ 0-20   //1.5;  //ΠΑΡΑΜΕΤΡΟΣ (D)erivative


   Steering = analogRead(Steer_Pot);
  Serial.print("Steering = ");Serial.print(Steering);  Serial.print("   ");
 //   if (Steering > 500 || Steering <600)
 //   Steer=0;

   //else
   if (Steering < 500)
    Steer = map (Steering, 0,500, -180,0);

   else if (Steering >600)
   Steer = map(Steering, 600,1023,0,180);

   //Steer = map (Steering, 0,1023,-180,180); //ΓΩΝΙΑ 180
Serial.print("Steer = ");Serial.print(Steer); Serial.print("   ");
}


// ======================================================
// ===               AUTO LEVEL               ===
// ======================================================
void auto_level(){
// enable auto-level turn On
Start_state = digitalRead(Start_switch);

//   Serial.print("Start_state = ");
//    Serial.print(Start_state); Serial.print("   ");
//    Serial.print("Start = ");
//   Serial.print(Start); Serial.print("   ");
   Serial.print("compAngleY = ");
   Serial.println(compAngleY);

   if (Start_state == 1){
```

```
      Start = false;
      digitalWrite(Motor_power,LOW);
     }

   else {
    if (Start == false){
     //if (compAngleY < 0.4 && compAngleY > -0.4){
     if (compAngleY < 2 && compAngleY > -2){
    Start = true;
    digitalWrite(Motor_power, HIGH);
    }
   else {
       Start = false;
      digitalWrite(Motor_power,LOW);
      }
     }
   else {
    Start = true;
    digitalWrite(Motor_power,HIGH);
      }
     }
}

// ========================================================
// ===           PID CONTROLLER                 ===
// ========================================================

void PID_and_Movenent() {

   thisTime = millis();
   double timeChange = (double)(thisTime - lastTime); // ΥΠΟΛΟΓΙΣΜΟΣ ΧΡΟΝΟΥ ΑΠΟ ΤΗΝ ΤΕΛΕΥΤΑΙΑ ΚΛΙΣΗ ΤΗΣ PID (~10ms)

   float error = (targetAngle - compAngleY); // ΥΠΟΛΟΓΙΣΜΟΣ ΣΦΑΛΜΑΤΟΣ

   // ΥΠΟΛΟΓΙΣΜΟΣ ΟΡΩΝ PID
   // ΟΙ ΤΙΜΕΣ PID ΠΟΛΑΠΛΑΣΙΑΖΟΝΤΑΙ / ΔΙΑΙΡΟΥΝΤΑΙ ΜΕ 10 ΩΣΤΕ ΝΑ ΠΡΟΚΥΨΟΥΝ ΣΤΑΘΕΡΕΣ ΜΕΤΑΞΥ 0-10
   float pTerm = Kp * error * 10;
   iTerm += Ki * error * timeChange / 100;
   if (Ki == 0) iTerm = 0;

   else if(iTerm > 255) iTerm= 255;  // ΠΕΡΙΟΡΙΣΜΟΣ ΤΙΜΩΝ ΣΤΑ ΟΡΙΑ PWM max
   else if(iTerm < -255) iTerm= -255;                    // να δω αν επιρεαζουν οι αρνητικες τιμες την κινηση
   float dTerm = Kd * (compAngleY - lastpitch) / timeChange * 1000;
   lastpitch = compAngleY;
   lastTime = thisTime;


// Serial.print("Kp  =");Serial.print(Kp);Serial.print("   ");
// Serial.print("Ki  =");Serial.print(Ki);Serial.print("   ");
// Serial.print("Kd  =");Serial.print(Kd);Serial.print("   ");
//  Serial.print("pTerm  =");Serial.print(pTerm);Serial.print("   ");
// Serial.print("iTerm  =");Serial.print(iTerm);Serial.print("   ");
//  Serial.print("dTerm  =");Serial.print(dTerm);Serial.print("   ");
   // Set PWM Value
   float PIDValue = (pTerm + iTerm + dTerm) * PIDGain;

   //ΟΡΙΣΜΟΣ ΕΛΑΧΙΣΤΗΣ ΤΑΧΥΤΗΤΑΣ ΠΟΥ ΟΙ ΚΙΝΗΤΗΡΕΣ ΘΑ ΕΙΝΑΙ ΚΑΤΑΣΤΑΣΗ ΑΝΑΜΟΝΗΣ (ΣΕ ΑΥΤΑ ΤΑ ΟΡΙΑ ΔΕΝ ΥΠΑΡΧΕΙ
ΑΡΚΕΤΗ ΡΟΠΗ ΝΑ ΚΙΝΗΘΕΙ ΤΟ ΟΧΗΜΑ)
   if(PIDValue > 0) PIDValue = PIDValue + 20;
   if(PIDValue < 0) PIDValue = PIDValue - 20;

  left_PWM = right_PWM = PIDValue;// ΟΙ ΔΥΟ ΚΙΝΗΤΗΡΕΣ ΤΗΘΕΝΤΑΙ ΣΤΗΝ ΙΔΙΑ ΤΑΧΥΤΗΤΑ
//Serial.print("PIDValue  =");Serial.print(PIDValue);Serial.print("   ");


  if (Start == true) // ΕΛΕΧΟΣ ΑΝ Ο ΔΙΑΚΟΠΤΗΣ ΕΚΚΙΝΗΣΗΣ ΕΙΝΑΙ ΕΝΕΡΓΟΣ
  {
  //  Serial.print("compAngleY  =");Serial.print(compAngleY);Serial.print("   ");

   if (compAngleY < -10 || compAngleY > 10){ // ΟΡΙΑ ΚΛΙΣΗΣ ΟΧΗΜΑΤΟΣ. ΑΝ ΓΙΝΕΙ ΥΠΕΡΒΑΣΗ ΟΙ ΚΙΝΗΤΗΡΕΣ
ΑΠΕΝΕΡΓΟΠΟΙΟΥΝΤΑΙ.
```

```
   digitalWrite(Motor_power,LOW);
   lcd.clear();
   lcd.setCursor(1,0);
   lcd.print("BALANCE SYSTEM STOP");
   Serial.print("BALANCE SYSTEM STOPED !!!");Serial.print("   ");
   //SOFT START();

   return ;
  }

  if (Steer == 0){
     // ΚΑΜΙΑ ΕΝΕΡΓΕΙΑ
      }
   else if (Steer > 0){
      left_PWM = left_PWM + abs(Steer);
      right_PWM = right_PWM - abs(Steer);
      }
   else if (Steer < 0){
      left_PWM = left_PWM - abs(Steer);
      right_PWM = right_PWM + abs(Steer);
      }

   // ΠΕΡΙΟΡΙΣΜΟΣ ΤΙΜΩΝ PID ΣΤΗΝ ΜΕΓΙΣΤΗ ΤΑΧΥΤΗΤΑ ΚΙΝΗΤΗΡΩΝ
   if (left_PWM > 255) left_PWM = 255;
   else if (left_PWM < -255) left_PWM = -255;
   if (right_PWM > 255) right_PWM = 255;
   else if (right_PWM < -255) right_PWM = -255;


// Serial.print("left_PWM  =");Serial.print(left_PWM);Serial.print("   ");
// Serial.print("right_PWM  =");Serial.println(right_PWM);

if (left_PWM >= 0){
  digitalWrite(DIR_M1, HIGH);   //κινηση αριστερου κινητηρα εμπρος
  analogWrite(PWM_M1, left_PWM); // με ταχυτητα left_PWM
}
else if (left_PWM < 0){
  digitalWrite(DIR_M1, LOW);   //κινηση αριστερου κινητηρα πισω
  analogWrite(PWM_M1, -left_PWM); // με ταχυτητα left_PWM  **************************allagi se -
}
    if (right_PWM >= 0){
    digitalWrite(DIR_M2,HIGH);   //κινηση δεξιου κινητηρα εμπρος
  analogWrite(PWM_M2, right_PWM); // με ταχυτητα right_PWM
}
else if (right_PWM < 0){
  digitalWrite(DIR_M2, LOW);   //κινηση δεξιου κινητηρα πισω
  analogWrite(PWM_M2, -right_PWM); // με ταχυτητα right_PWM ****************************** al.agi se -
}
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("BALANCE SYSTEM RUN !!!");
    Serial.print("BALANCE SYSTEM RUN !!!");Serial.println("");
  }
  else{
    analogWrite(PWM_M1, 0);
    analogWrite(PWM_M2, 0);
    lcd.clear();
    lcd.setCursor(1,0);
    lcd.print("balance false... ");
   Serial.print("balance false... ");Serial.println("");
  }
}
/*
// ================================================================
// ===                CURRENT SENSE                   ===
// ================================================================

void current_sense() // current sense and diagnosis
{
  int i;
  int j;
```

```
 if(val1==HIGH || val2==HIGH)
 {
  counter++;
  if(counter==3)
  {
   counter=0;
   lcd.clear();
   lcd.setCursor(1,0);
   lcd.print("Warning!!! ");
   lcd.setCursor(2,0);
   lcd.print(" Motor Over current");
   Serial.println("Warning Motor Over current");

   for (i=0; i<3; i++)
   {
     for (j=0; j<3; j++)
      {
       digitalWrite(Buzz_Pin, LOW);
       delay(100);
       digitalWrite(Buzz_Pin, HIGH);
       delay(100);
      }
   delay(1000);
   }
  }
 }
}
*/

// ================================================================
// ===               TIME KEEPER  ~100Hz                ===
// ================================================================

void timekeeper() {

  // ΥΠΟΛΟΓΙΣΜΟΣ ΔΙΑΡΚΕΙΑΣ ΧΡΟΝΟΥ ΑΠΟ ΤΗΝ ΑΡΧΗ ΤΟΥ ΒΡΟΓΧΟΥ
  lastLoopUsefulTime = millis() - loopStartTime;

  // ΑΝ ΔΕΝ ΕΧΕΙ ΕΠΙΤΕΥΧΘΕΙ Ο ΑΠΑΙΤΟΥΜΕΝΟΣ ΧΡΟΝΟΣ ΠΡΟΣΘΕΣΕ ΑΝΑΛΟΓΗ ΚΑΘΥΣΤΕΡΗΣΗ
  if (lastLoopUsefulTime < STD_LOOP_TIME) {
    delay(STD_LOOP_TIME - lastLoopUsefulTime);

    // ΚΑΤΑΓΡΑΦΗ ΧΡΟΝΟΥ ΑΠΟΚΛΙΣΗΣ ΑΠΟ ΤΟΝ ΣΤΟΧΟ (10ms)
    looptimefast += STD_LOOP_TIME - lastLoopUsefulTime;
    looptimefast = looptimefast / 2;

  // ΕΛΕΓΧΟΣ ΑΠΑΙΤΟΥΜΕΝΟΥ ΧΡΟΝΟΥ ΒΡΟΓΧΟΥ
  } else {
    // ΚΑΤΑΓΡΑΦΗ ΧΡΟΝΟΥ ΑΠΟΚΛΙΣΗΣ ΑΠΟ ΤΟΝ ΣΤΟΧΟ (10ms)
    looptimeslow += lastLoopUsefulTime - STD_LOOP_TIME;
    looptimeslow = looptimeslow / 2;
  }
  // ΜΕΣΟΣ ΟΡΟΣ ΑΠΟΚΛΙΣΗΣ ΑΠΟ ΤΑ 100Hz (ΑΡΝΗΤΙΚΟ. = ΚΑΘΥΣΤΕΡΗΣΗ,   ΘΕΤΙΚΟ. = ΝΩΡΙΤΕΡΑ)
  looptimeaverage = looptimefast - looptimeslow;

  // ΕΝΗΜΕΡΩΣΗ ΜΕΤΑΒΛΗΤΩΝ ΒΡΟΓΧΟΥ ΧΡΟΝΟΜΕΤΡΟΥ
  lastLoopTime = millis() - loopStartTime;
  loopStartTime = millis();
}
```