Τ.Ε.Ι. ΠΕΙΡΑΙΑ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

«ΑΥΤΟΜΑΤΟ ΣΥΣΤΗΜΑ ΕΝΑΛΛΑΓΗΣ ΤΑΧΥΤΗΤΩΝ ΣΕ ΠΟΔΗΛΑΤΟ»

ΟΝΟΜΑΤΑ ΦΟΙΤΗΤΩΝ:
ΒΟΥΡΔΕΡΗΣ ΑΝΤΩΝΙΟΣ Α.Μ: 30086
ΙΩΑΝΝΟΥ ΙΩΑΝΝΗΣ Α.Μ: 33359
ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΗΣ

ΑΘΗΝΑ,
ΣΕΠΤΡΕΜΒΡΙΟΣ 2013

Αφιερωμένο

Στους Γονείς μου στην αδερφή μου Βασιλεία και στον γαμπρό μου Βαγγέλη

Βουρδέρης Αντώνιος

Αφιερωμένο

Στην μητέρα μου Αγγελική που την έχασα τον Μάιο του 2013, στον πατέρα μου Ηλία , στην αδερφή μου Ειρήνη και στον γαμπρό μου Δημήτρη.

Ιωάννου Ιωάννης

Ευχαριστίες

Πριν αρχίσει η παρουσίαση της πτυχιακής μας εργασίας, θα θέλαμε να ευχαριστήσουμε τον κ.ο. Νικολάου Γρηγόρη, καθηγητής στο τμήμα αυτοματισμού του ΤΕΙ Πειραιά, που μας επέτρεψε να ασχοληθούμε με το συγκεκριμένο σύγχρονο και άκρως ενδιαφέρον θέμα, για τη βοήθεια που μας προσέφερε για την ολοκλήρωση της εργασίας και την άψογη συνεργασίας μας .

Αισθανόμαστε την υποχρέωση να ευχαριστήσουμε το ΤΕΙ Πειραιά για τη βοήθεια που μας παρείχε στη συλλογή στοιχείων και το δανεισμό σημαντικών άρθρων και βιβλίων από την πλούσια βιβλιοθήκη του για την εκπόνηση της πτυχιακής μας εργασίας.

Ιδιαίτερες ευχαριστίες οφείλουμε και στην κ.α. Βουρδέρη Βασιλεία για τις πολύτιμες συμβουλές της.

Αθήνα Σεπτέμβριος 2013

Βουρδέρης Αντώνιος

Ιωάννου Ιωάννης

Περίληψη

Τα τελευταία χρόνια και κυρίως από τα μέσα της δεκαετίας του 2000 και μετά παρατηρείται παγκοσμίως μια μεγάλη αύξηση τόσο των οδηγών ποδηλάτων όσο και των ατυχημάτων στα οποία αυτά εμπλέκονται. Κατά συνέπεια είναι πολύ σημαντικό να σχεδιαστούν μέτρα, τα οποία θα προστατεύουν τους οδηγούς των ποδηλάτων, από πιθανά ατυχήματα και θα μειώνουν τις επιπτώσεις των ατυχημάτων στους αναβάτες. Για το σκοπό αυτό θα πρέπει να προσδιοριστεί ένα πρόσθετο πλεονέκτημα το οποίο θα βοηθά τον αναβάτη ώστε να μην αποσπάται η προσοχή του. Ο προσδιορισμός του πλεονεκτήματος τροφοδοτεί τα ευφυή συστήματα με τα απαραίτητα δεδομένα ώστε να ρυθμίζουν τις μηχανικές παραμέτρους του ποδηλάτου και να προστατεύουν τον αναβάτη από πιθανό ατύχημα. Η παρούσα πτυχιακή εστιάζει στον εντοπισμό των μεταβλητών που υποδεικνύουν την ομαλή αλλαγή των ταχυτήτων στο ποδήλατο.

Ο εντοπισμός των μεταβλητών γίνεται αναλύοντας δεδομένα που προέρχονται από πείραμα φυσικής οδήγησης. Σκοπός της ανάλυσης αυτής είναι να ανιχνευθούν διαφοροποιήσεις στο προφίλ του οδηγού ανάλογα με το αν κινείται με εντατική ή ομαλή οδήγηση και ανάλογα της χρονικής στιγμής που πραγματοποιείται η διαδρομή.

Με την ανάλυση των κύριων συνιστωσών επιτυγχάνεται συμπύκνωση της πληροφορίας που παρέχουν οι μεταβλητές και στη συνέχεια με τον μικροελεγκτή Atmega έχει εντοπισθεί το όριο των ταχυτήτων. Γνωρίζοντας το όριο των ταχυτήτων εύκολα προσδιορίζεται η ομαλή αλλαγή τους. Στη συνέχεια μελετάται η ψηφιακή απεικόνιση βασικών πληροφοριών (ταχύτητα, χιλιόμετρα ανά ώρα, προφίλ, στάθμη μπαταρίας) ώστε ο αναβάτης να έχει καλύτερη αντίληψη της οδηγικής συμπεριφοράς. Τέλος ο αναβάτης (μέσο της κάρτας SD) μπορεί να δει πόσα χιλιόμετρα διένυσε σε κάθε πορεία του και επίσης του δίνεται η δυνατότητα αλλαγής των μεταβλητών, ανάλογα με την δυσκολία που επιθυμεί.

Τα βασικά κομμάτια της πτυχιακής αυτής είναι:

- Περιγραφή της λειτουργίας του ποδηλάτου, ώστε να κατανοήσουμε αρχικά πως γίνεται χειροκίνητα η αλλαγή ταχυτήτων και ποιες είναι οι συνθήκες που χρειάζονται.
- Ανάλυση των μηχανικών εξαρτημάτων του ποδηλάτου και επεξήγηση του τρόπου λειτουργίας τους.
- Λεπτομερής ανάλυση των πρόσθετων εξαρτημάτων που τοποθετήσαμε και επεξήγηση της κατασκευής τους.
- Επεξήγηση των κυκλωμάτων που υλοποιήσαμε για την επίτευξη της πτυχιακής μας.
- Αναφορά στα προβλήματα που αντιμετωπίσαμε και στις λύσεις που βρήκαμε για την αντιμετώπισή τους.
- Πιθανές βελτιώσεις για την εξέλιξη της υπάρχουσας πτυχιακής.
- Λεπτομερής περιγραφή και επεξήγηση του προγραμματισμού που χρησιμοποιήσαμε για την λειτουργία της πτυχιακής.

Abstract

In recent years and especially since the mid- 2000s and after, globally a strong increase in bicycle accidents in which they are involved. Consequently, it is crucial to design measures that will protect their drivers from possible accidents and reduce their impact on riders. For this purpose should be determined an additional advantage which will help the rider to not distracted. Determining the advantage powering intelligent systems with the necessary data to regulate the mechanical parameters of the bike and protect the rider from possible accident. This project focuses on identifying the variables that indicate the smooth gear change on the bike.

Identifying the variables is analyzing data from physics experiment driving. The purpose of the analysis is to detect variations in the profile of the driver depending on whether moving or intensive smooth ride and depending the time made the trip.

With the analysis of the main components is achieved condensation of information provided variables and with the microcontroller Atmega we have identified the limits of the gears. Knowing the limits of the gears, it's easy to determine smooth changes. Then we examine the basic information of the digital display (speed, miles per hour, profiles, battery level) so that the rider can have a better idea of driving behavior. Finally the rider (with the SD card) can see how many kilometers have gone through each path and also gives the possibility of changing the variables, depending on the difficulty he wishes.

The main parts of this dissertation are:

- Functional description of the bike in order to understand how initially done manually gear changes and what are the conditions they need.
- Analysis of the mechanical parts of the bike and an explanation of how they work.
- Detailed analysis of the accessories we put and an explanation of their construction.
- Explanation of circuits implemented to achieve our graduation.
- Report to the problems faced and the solutions found to deal with them.
- Possible improvements for the evolution of existing dissertation.
- Detailed description and explanation of programming that we used for the operation of the graduation.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΕΔΙΑΓΡΑΜΜΑΤΩΝ

ΕΥΡΕΤΗΡΙΟ ΚΥΚΛΩΜΑΤΩΝ

Σκοπός

Το θέμα της παρούσας πτυχιακής είναι να αυτοματοποιηθεί η χειροκίνητη αλλαγή ταχυτήτων, ώστε να μην αποσπάται η προσοχή του αναβάτη, προσπαθώντας να βρει την κατάλληλη ταχύτητα για την εκάστοτε περίπτωση.

Στην εργασία περιγράφουμε όλη την διαδικασία, την οποία ακολουθήσαμε για να πετύχουμε το σκοπό μας. Επίσης περιγράφουμε τα υλικά που χρησιμοποιήσαμε, πως κατασκευάσαμε τα κυκλώματά μας, πως λειτουργεί η συσκευή μας, καθώς και τον κώδικα του προγραμματισμού σε υπορουτίνες, λόγω μεγάλου μεγέθους.

Κεφάλαιο 1

1.1 Ιστορική Αναδρομή

Ποδήλατο ονομάζεται το δίτροχο (μερικές φορές τρίτροχο) όχημα, που κινείται καθώς ο αναβάτης του χρησιμοποιεί τη μυϊκή δύναμη των ποδιών του. Στην κλασική του μορφή, το ποδήλατο αποτελείται από δύο τροχούς, οι οποίοι βρίσκονται ο ένας πίσω από τον άλλο και συνδέονται μεταξύ τους με μεταλλικό σκελετό. Βασικά επίσης μέρη ενός τυπικού ποδήλατου αποτελούν το τιμόνι, η σέλα, το σύστημα μετάδοσης της κίνησης και τα φρένα. Ως συμπληρωματικός εξοπλισμός, όχι δηλαδή απαραίτητος για τη λειτουργικότητα του ποδηλάτου, χρησιμοποιείται ένα πλήθος από εξαρτήματα.

Η πρώτη απτή εμφάνιση του ποδηλάτου, με τελείως διαφορετική κατασκευή από τα σημερινά, ήταν γύρω στο 1750 στη Νυρεμβέργη. Αυτός ο πρώτος παππούς του ποδηλάτου ήταν τόσο απλός που δεν είχε ούτε πετάλια, ούτε τιμόνι, και ήταν εξολοκλήρου κατασκευασμένος από ξύλο!!

Το 1817 στη Γερμανία, ο βαρώνος Karlvon Drais, θέλοντας ένα μεταφορικό μέσο για τη βόλτα του στους κήπους του, κατασκεύασε από ξύλο και αυτός την ντρεζίνα (draisienne) που πήρε το όνομα της από το επώνυμο του. Η διαφορά με το προηγούμενο ήταν πως είχε τιμόνι, αλλά η κίνηση εξακολουθούσε να γίνεται ουσιαστικά περπατώντας και τσουλώντας αυτό το όχημα, γι' αυτό και πήρε το όνομα «μηχανή περπατήματος».

Μερικά χρόνια αργότερα, το 1839 στη Σκωτία, ο Kirkpatrick Macmillan ήταν ο πρώτος που έβαλε πετάλια και τα συνέδεσε με την πίσω ρόδα, αλλά χωρίς αλυσίδα.

Το 1960 στη Γαλλία, ο Pierre Michaux τοποθέτησε τα πετάλια στην μπροστινή ρόδα, δημιουργώντας το Velocipede, που σημαίνει «γρήγορα πόδια». Επίσης αύξησε το μέγεθος της εν λόγω ρόδας και έβαλε λάστιχα από σκληρό καουτσούκ.

Η πρώτη εμφάνιση του διπλού ποδήλατου έγινε το 1886 και είχε 4 ρόδες: δύο μεγάλες και δύο μικρές. Οι αναβάτες ουσιαστικά κάθονταν ανάμεσα στις δύο μεγάλες ρόδες και είχαν τις μικρές για να μην πέφτουν!!

Στη συνέχεια το 1870, οι James Starley και William Hillman στη Βρετανία, κατασκεύασαν την Ariel, με πολύ μεγαλύτερη την μπροστινή ρόδα με τα πετάλια. Η

φιλοσοφία τους ήταν πως, όσο μεγαλύτερη η ρόδα (η οποία πολλές φορές κατασκευαζόταν με βάση το μήκος του ποδιού του αναβάτη!), τόσο μεγαλύτερη απόσταση θα διένυε σε κάθε περιστροφή των πεταλιών.

Το 1885 έγινε η κυριότερη μετατροπή και από τότε το ποδήλατο πήρε την κλασική του εμφάνιση με τις δύο ίδιες ρόδες, την αλυσίδα που δίνει κίνηση στην πίσω ρόδα και τις μεταλλικές ακτίνες. Σε αυτό βοήθησε πολύ και η εξέλιξη της μεταλλουργίας. Υπεύθυνος για όλα αυτά καθώς και για τη σαμπρέλα και τις ταχύτητες ήταν ο ανηψιός του James, ο John Kemp Starley. Επίσης άλλαξε τον σκελετό με κούφιο μεταλλικό μειώνοντας το βάρος του ποδηλάτου. Το μοντέλο αυτό το ονόμασε rover.

Αξίζει να αναφέρουμε επίσης πως το 1885, το ποδήλατο έκανε την πρώτη του εμφάνιση στην Ελλάδα.

Τρία χρόνια μετά, το 1888, ο γιατρός Dr. John Boyd Dunlop, θέλοντας να κάνει τις βόλτες του γιού του με το ποδηλατάκι του πιο άνετες, άλλαξε τα υπάρχοντα λάστιχα από καουτσούκ με λάστιχα πεπιεσμένου αέρα.

Το 1947 αντικαταστάθηκε ο μεταλλικός σκελετός με αλουμίνιο, ίδιο με αυτό που χρησιμοποιούσαν στα αεροπλάνα.

Το 1965 βγήκε στην αγορά ένα μίνι ποδήλατο, και φτάνοντας στο 1970 βλέπουμε την εμφάνιση του πολύ δημοφιλούς στις μέρες μας mountainbike!

Κεφάλαιο 2

2.1 Λειτουργία Ποδηλάτου

Το ποδήλατο ξεκίνησε σαν ένα μέσο μεταφοράς των ανθρώπων. Στα πρώτα βήματά του η λειτουργία του ποδηλάτου ήταν πολύ απλή.

Στην πιο συνηθισμένη του μορφή έχει δύο τροχούς, οι οποίοι είναι κατασκευασμένοι από ένα μεταλλικό πλαίσιο και είναι εφοδιασμένοι με πλαστικούς αεροθαλάμους (ελαστικά) για πιο καλή και άνετη κίνηση. Η κίνηση δίνεται στον πισινό τροχό με τη βοήθεια μιας αλυσίδας (καδένα), που τυλίγεται γύρω από δύο οδοντωτούς τροχούς με μικρή, σχετικά, διάμετρο. Η κινητήρια δύναμη δίνεται από τα πόδια του ανθρώπου στα πεντάλ που περιστρέφουν τον έναν οδοντωτό τροχό. Με τη βοήθεια τη αλυσίδας η κίνηση μεταφέρεται στον άλλο οδοντωτό τροχό, που είναι προσαρμοσμένος στον πισινό τροχό του ποδηλάτου και έτσι δημιουργείται η κίνηση. Οι δύο οδοντωτοί τροχοί διαφέρουν στη διάμετρο και, συνεπώς, στον αριθμό των δοντιών. Μικρότερος είναι αυτός που προσαρμόζεται στον πίσω τροχό του ποδηλάτου και συνήθως έχουνε σχέση 2:1.

Οι δύο τροχοί του ποδηλάτου συνδέονται μεταξύ τους με το "σκελετό" του. Αυτός είναι ένα τριγωνικό πλαίσιο από τρεις κυρίως χαλυβδοσωλήνες με διάμετρο που ποικίλλει. Η διεύθυνση της κίνησης δίνεται με το τιμόνι, που συνδέεται με τον μπροστινό τροχό με τη βοήθεια της περόνης. Ο ποδηλάτης κάθεται πάνω σε μια σέλα από πλαστικό συνήθως, που βρίσκεται τοποθετημένη πάνω στο σκελετό, κάπου ανάμεσα στους δύο τροχούς. Το σύστημα πέδησης (φρεναρίσματος) του ποδηλάτου αποτελείται κυρίως από ένα ισχυρό λεπτό καλώδιο, που με τη βοήθεια ενός μοχλού που είναι τοποθετημένος στο τιμόνι θέτει σε λειτουργία ένα ζεύγος από δύο "μαξιλάρια", συνήθως από καουτσούκ. Τα μαξιλάρια αυτά δρουν πάνω στη μεταλλική στεφάνη κάθε τροχού και προκαλώντας μεγάλες τριβές ακινητοποιούν το ποδήλατο.

Εκτός από τα κύρια αυτά μέρη του ποδηλάτου που είναι απαραίτητα για την ομαλή και ακίνδυνη κίνησή του, υπάρχουν και μερικά άλλα στοιχεία σ' ένα συνηθισμένο ποδήλατο. Όπως προφυλακτήρες, σχάρα πάνω από τον πισινό τροχό, φώτα μπροστά και πίσω, κ.λ.π.

2.2 Σύστημα αλλαγής ταχυτήτων Ποδηλάτου

Αρκετά ποδήλατα διαθέτουν και κιβώτιο ταχυτήτων, όμοιο αναλογικά με το κιβώτιο ταχυτήτων που έχουν τα αυτοκίνητα, τοποθετημένο στον πισινό τροχό.

Σε αυτό το κεφάλαιο θα εξηγήσουμε ποιά είναι τα εξαρτήματα του κιβώτιου ταχυτήτων και ποιος ο μηχανισμός αλλαγής τους. Αυτό θα δώσει μια ιδέα για το τι συμβαίνει όταν αλλάζουμε ταχύτητα.

Η πεταλιέρα, η οποία είναι βιδωμένη στο κάτω μέρος του σκελετού, αποτελείται από τρία γρανάζια. Τα γρανάζια αυτά είναι βιδωμένα πάνω στην πεταλιέρα. Το κάθε ένα από αυτά έχει διαφορετικό μέγεθος. Το μεγάλο γρανάζι έχει 42-48 δόντια, το μεσαίο έχει 32 έως 36 δόντια και το μικρό 20 έως 26 δόντια.

Στον πίσω άξονα υπάρχει ακόμα ένα σετ γραναζιών που όλο μαζί λέγεται "κασέτα". Οι σύγχρονες "κασέτες" αποτελούνται από 7 έως 9 γρανάζια και έχουν από 11 έως 36 δόντια.

Με την πάροδο του χρόνου υπήρξε μεγάλη εξέλιξη στην μετάδοση κίνησης ενός ποδηλάτου. Προστέθηκαν πολλαπλές ταχύτητες για πιο αποδοτική κίνηση. Υπάρχουν διάφοροι τύποι συστημάτων αλλαγής ταχυτήτων ποδηλάτου. Οι 2 πιο συχνοί τύποι είναι:

1. Οι εσωτερικού τύπου, που είναι ενσωματωμένοι στο κέντρο του τροχού και περιέχουν εσωτερικά τα γρανάζια για την αλλαγή της ταχύτητας και
2. Οι εξωτερικού τύπου που όλα τα γρανάζια και τα ντεραγιέρ (ο μηχανισμός που αλλάζει την ταχύτητα) είναι σε εμφανή σημείο πάνω στο ποδήλατο.

Και οι 2 τύποι χρησιμοποιούν συρματόσχοινα για τη αλλαγή ταχυτήτων. Τα συρματόσχοινα καταλήγουν σε μοχλούς επάνω στο τιμόνι , που μας επιτρέπουν να αλλάξουμε ταχύτητα.

Οι λεβιέδες/επιλογείς ταχυτήτων βρίσκονται δίπλα στα χερούλια, ένας δεξιά και ένας αριστερά. Ο δεξιός επιλογέας είναι για τα πίσω γρανάζια (μετακινεί την αλυσίδα στα πίσω γρανάζια). Ο αριστερός επιλογέας/λεβιές είναι για τα μπροστινά γρανάζια και μετακινεί την αλυσίδα μεταξύ των τριών εμπρός γραναζιών.

Όταν η αλυσίδα είναι πάνω στο μπροστινό μεγάλο γρανάζι σημαίνει ότι έχει επιλεχθεί μεγάλη ταχύτητα στο σασμάν, που συνήθως χρησιμοποιείται για επίπεδους δρόμους ή

κατηφόρες. Τα μπροστινά μικρότερα γρανάζια είναι για αναβάσεις, για όταν μεταφέρονται βάρη με το ποδήλατο, όταν υπάρχουν δυνατοί μετωπικοί άνεμοι ή για όταν οδηγούμε στην πόλη και θέλουμε καλύτερο έλεγχο στα συνεχή σταμάτα-ξεκίνα των φαναριών και στην κίνηση.

Όταν η αλυσίδα είναι στο μικρότερο πίσω γρανάζι σημαίνει έχει επιλεχθεί μεγάλη ταχύτητα στο σασμάν και θα κάνουμε μεγαλύτερη απόσταση με κάθε περιστροφή του πεταλιού. Με μεγαλύτερο πίσω γρανάζι έχει επιλεχθεί μικρότερη ταχύτητα στο σασμάν και βοηθάει στις ανηφόρες και στο ξεκίνημα από τα φανάρια. Αφού έχετε επιλεχθεί με το λεβιέ ταχύτητα στο σασμάν, θα πρέπει να κάνουμε πετάλι για να γίνει αυτή η αλλαγή.

Είτε οδηγούμε σε δρόμους είτε σε μονοπάτια, θα πρέπει να κάνουμε τις αλλαγές ταχυτήτων ενστικτωδώς για να αποφύγουμε διάφορα προβλήματα με την κίνηση των δρόμων της πόλης ή προβλήματα που μπορεί να προκληθούν στα μονοπάτια της εξοχής.

Παρακάτω φαίνεται ο μηχανισμός που αλλάζει τις ταχύτητες χειροκίνητα, ο οποίος ονομάζεται ντεραγιέρ. Επειδή η κατασκευή μας εξαρτάται , σε πολύ μεγάλο βαθμό από αυτό τον μηχανισμό , θα κάνουμε μια σύντομη περιγραφή του ντεραγιέρ.

Ντεραγιέρ είναι ο μηχανισμός που μεταβάλλει τις σχέσεις των ταχυτήτων και χρησιμοποιείται συνήθως για ποδήλατα. Αποτελείται από μια αλυσίδα και από πολλαπλά γρανάζια διαφόρων μεγεθών, καθώς και ένα μηχανισμό για να μετακινεί την αλυσίδα από το ένα γρανάζι στο άλλο. Στα σύγχρονα ποδήλατα τα εμπρός και πίσω ντεραγιέρ συνήθως αποτελούνται από μια κινητή αλυσίδα (οδηγό) που λειτουργεί από απόσταση με ένα καλώδιο BOWDEN , το οποίο συνδέεται με ένα λεβιέ ταχυτήτων τοποθετημένο στο τιμόνι. Όταν ένας αναβάτης αλλάζει ταχύτητα με τον λεβιέ, ενώ κάνει πεντάλ, η αλλαγή στην τάση του καλωδίου μετακινεί την αλυσίδα-οδηγό από τη μία πλευρά στην άλλη, προκαλώντας «εκτροχιασμό» της αλυσίδας σε διαφορετικούς οδοντωτούς τροχούς.

Όπως αναφέραμε προηγουμένως έχουμε το πίσω και το μπροστά ντεραγιέρ.

Το πίσω ντεραγιέρ εξυπηρετεί διπλό καθήκον: τη μετακίνηση της αλυσίδας μεταξύ των πίσω γραναζιών και την σύσφιξη της χαλαρής αλυσίδας που προκαλείται από τη μετάβαση σε ένα μικρότερο γρανάζι από το πίσω ή από το μπροστινό ντεραγιέρ.

Εικόνα 1 Πίσω εξωτερικό ντεραγιερ

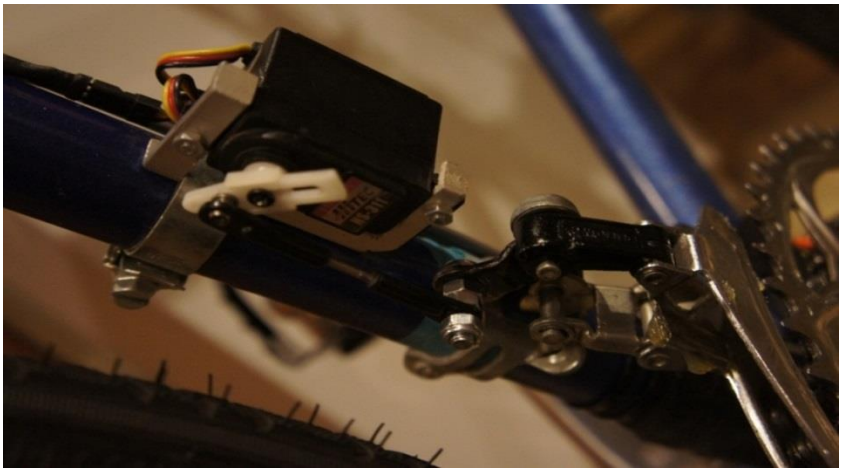
Το μπροστινό ντεραγιέρ έχει να μετακινήσει μόνο την πλευρά της αλυσίδας στην άλλη μεταξύ των μπροστινών δακτυλίων της, η οποία είναι η κορυφή της αλυσίδας, άρα, το τεντωμένο τμήμα της. Πρέπει επίσης να φιλοξενήσει μεγάλες διαφορές στο μέγεθος του δακτυλίου της αλυσίδας , δηλαδή από τα 53 δόντια στα 20 δόντια.

Εικόνα 2 Μπροστά εξωτερικό ντεραγιέρ

2.3 Αυτόματη Εναλλαγή Ταχυτήτων

Σε αυτό το κεφάλαιο περιγράφουμε τον τρόπο υπολογισμού της αυτόματης εναλλαγής ταχυτήτων καθώς και τα υλικά από τα οποία αποτελείται ο μηχανισμός.

Για την αυτόματη εναλλαγή ταχυτήτων έπρεπε να υπολογίσουμε τον αριθμό των στροφών του πεταλιού σε κάθε δευτερόλεπτο. Όλο αυτό το πετύχαμε με την χρήση ενός μικροελεγκτή AVRATMEL 328. Για την αυτόματη εναλλαγή ταχυτήτων χρησιμοποιήσαμε 2 σερβοκινητήρες , έναν αναλογικό και ένα ψηφιακό όπως φαίνεται στις παρακάτω εικόνες



Εικόνα 3 Αναλογικός σερβοκινητήρας

Εικόνα 4 Ψηφιακός σερβοκινητήρας

2.3.1 Περιγραφή servo και ταχυτήτων

Ο μικροελεγκτής γράφει μια τιμή σε μικροδευτερόλεπτα στο σέρβο, ελέγχοντας τον άξονα αναλόγως. Σε ένα τυπικό servo, αυτό θα καθορίσει τη γωνία του άξονα. Στο πρότυπο servo μια τιμή παραμέτρου του 1000 είναι πλήρως αριστερόστροφα, 2000 δεξιόστροφα, και 1500 είναι στη μέση.

Σημειώστε ότι μερικοί κατασκευαστές δεν ακολουθούν αυτό το πρότυπο, κάποια servo συχνά ανταποκρίνονται σε τιμές μεταξύ 700 και 2300.

Παράδειγμα προγραμματισμού

```
#include<Servo.h>

Servo myservo;

void setup()
{
myservo.attach(9);
myservo.writeMicroseconds(1500);  // set servo to mid-point
}

void loop() {}
```

Στην δική μας περίπτωση έχουμε χρησιμοποιήσει τις παρακάτω τιμές:

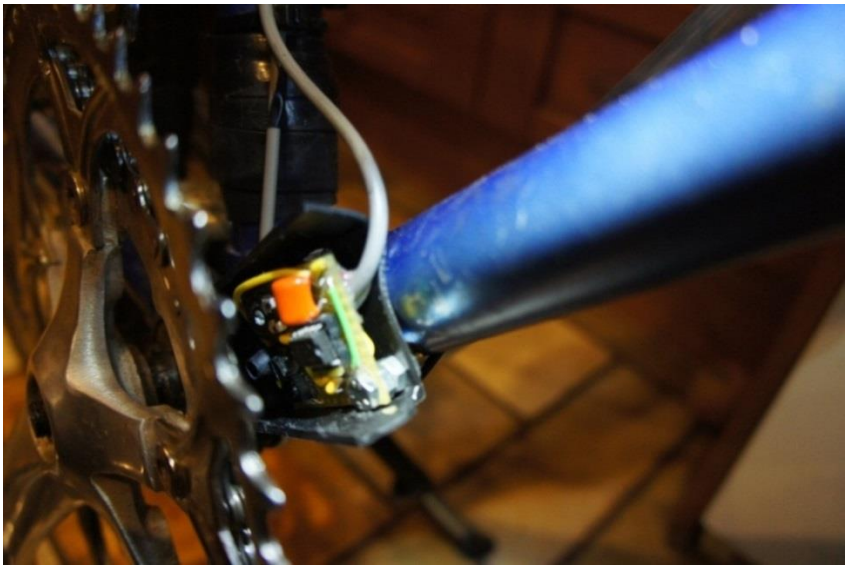
Πίσω servo:

1. Γρανάζι=850
2. Γρανάζι=1000
3. Γρανάζι=1200
4. Γρανάζι=1400
5. Γρανάζι=1750

Μπροστά servo:

1. Γρανάζι=2000
2. Γρανάζι=1350

Επίσης κατασκευάσαμε ένα οπτικό αισθητήριο, το οποίο κάνει καταμέτρηση των στροφών και λειτουργεί με υπέρυθρες.



Εικόνα 5 Οπτικό αισθητήριο

Μία LCD οθόνη με σειριακή επικοινωνία που μας δείχνει βασικές πληροφορίες.

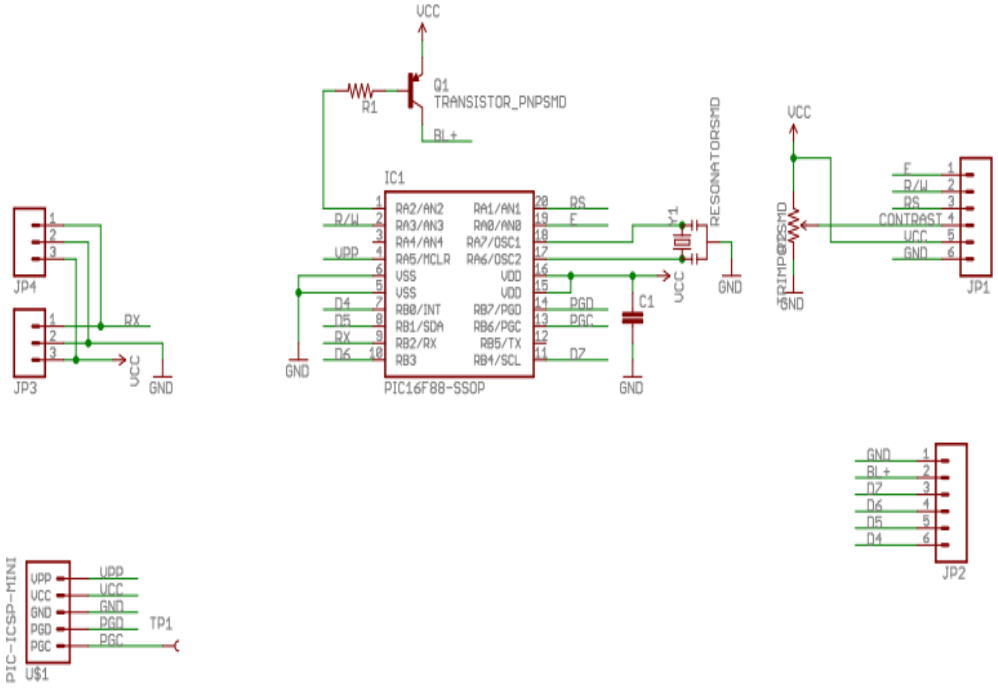
Εικόνα 6 Οθόνη LCD

Είναι μια 16x2 LCD οθόνη. Το ενσωματωμένο κύκλωμα βασίζεται γύρω από ένα PIC 16F88 , το οποίο PIC παίρνει μία TTL σειριακή είσοδο και εκτυπώνει τους χαρακτήρες που λαμβάνει πάνω στην οθόνη LCD. Η εγκατεστημένη firmware επιτρέπει επίσης για μια σειρά

από ειδικές εντολές ώστε να μπορεί να καθαρίσει την οθόνη, να ρυθμίσει τη φωτεινότητα του οπίσθιου φωτισμού, να ενεργοποιήσει την οθόνη on / off, και πολλά άλλα.

Η επικοινωνία με το SerLCD απαιτεί 5V TTL σειράς σε μια προεπιλεγμένη ταχύτητα baud του 9600bps (8-N-1). Μπορεί να ρυθμιστεί το baud σε οποιοδήποτε κανονικό συντελεστή μεταξύ του 2400 και 38400bps

Το SerLCD έχει την ικανότητα να ελαττώσει το φωτισμό για εξοικονόμηση ενέργειας, αν χρειαστεί. Υπάρχει επίσης ένα ποτενσιόμετρο στο πίσω μέρος της οθόνης για να ρυθμιστεί η αντίθεση.



Σχεδιάγραμμα 1 Οθόνη LCD

Χαρακτηριστικά:

- Ενσωματωμένο PIC 16F88 για μεγαλύτερη ακρίβεια επικοινωνίας
- Ρυθμιζόμενοι ρυθμοί baud 2400, 4800, 9600 (προεπιλογή), 14400, 19200 και 38400
- Επιχειρησιακό Backspace

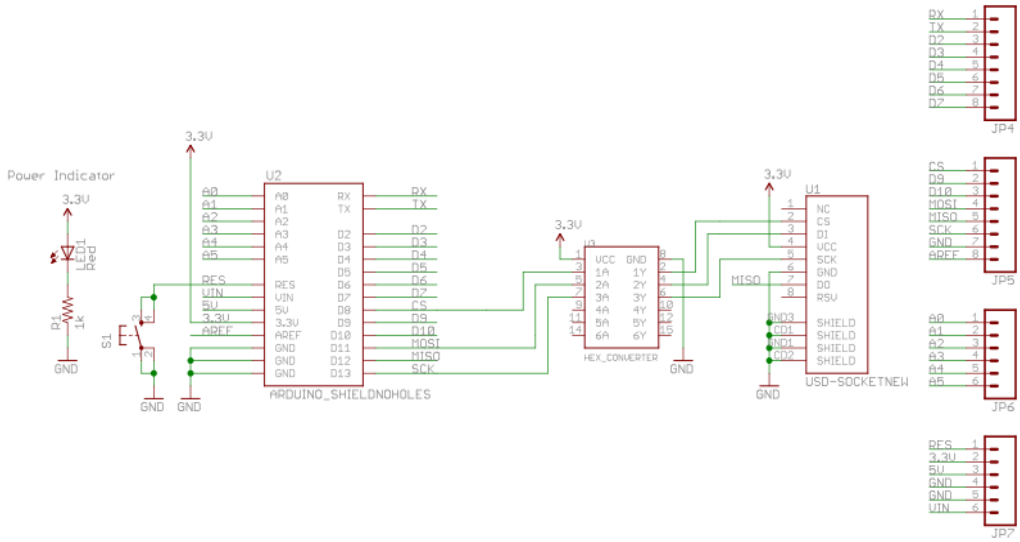
- Μεγαλύτερη ταχύτητα επεξεργασίας σε 10MHz
- Εισερχόμενη buffer αποθηκεύει έως 80 χαρακτήρες
- Τρανζίστορ οπίσθιου φωτισμού το οποίο μπορεί να χειριστεί μέχρι 1A
- Πλάτος παλμού διαφοροποίησης του οπίσθιου φωτισμού το οποίο επιτρέπει τον άμεσο έλεγχο της φωτεινότητα του οπίσθιου φωτισμού και ελέγχει την κατανάλωση ρεύματος
- Ταχύτερο χρόνο εκκίνησης
- Το Boot-up της οθόνης μπορεί να ενεργοποιηθεί / απενεργοποιηθεί μέσω firmware

Ένα απλό παράδειγμα προγραμματισμού της οθόνης εμφανίζεται παρακάτω:

```
/*
  serLCD - Example
*/
#include<SoftwareSerial.h>
#include<serLCD.h>

// Set pin to the LCD's rxPin
int pin = 2;

serLCDlcd(pin);

void setup()
{
  lcd.print("Hello, World!");
```

```
15.     }

16.

17.     void loop() {}
```

Επίσης τοποθετήσαμε και μία κάρτα SD που επικοινωνεί με SPI και αναγράφονται πληροφορίες που έχουμε ορίσει εμείς.



Σχεδιάγραμμα 2 Κάρτα SD

Για την σωστή λειτουργία του παραπάνω κυκλώματος ο κατασκευαστής χρησιμοποιεί το τσιπάκι τύπου M54/74HC4049.Οπότε καλό είναι να εξετάσουμε την λειτουργία του.

Τα κυριότερα χαρακτηριστικά του είναι:

. Υψηλή ταχύτητα TPD = 9 ns (typ) μεVCC = 5 V. Χαμηλής ισχύς διασπράςICC = 1 mA (max.) μεTA = 25 ° C. Υψηλή ανοσία στο θόρυβο VNIHNOISE = VNIL = 28% VCC

. Συμμετρική αντίσταση εξόδου | IOH | = IOL = 6 mA

. Μικρή καθυστέρηση διάδοσης

tPLH = tPHL

. Ευρύ φάσμα τάσης λειτουργίας VCC (OPR) = 2 V έως 6V

2.3.2 Περιγραφή

Το M54/74HC4049 και το M54/74HC4050 είναι υψηλής BUFFER HEX ταχύτητας CMOS και κατασκευάζονται σε sili ¬ con πύλη C2MOS τεχνολογίας.

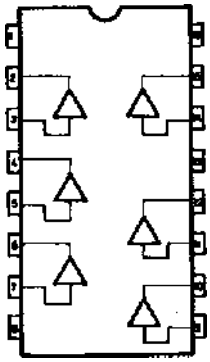
Έχουν την ίδια υψηλή ταχύτητα των LSTTL σε συνδυασμό με την αληθινή CMOS χαμηλής ισχύος con ¬ κατανάλωση.

Το M54/75HC4049 είναι μια αναστρέφοντας ρυθμιστικό, ενώ η M54/74HC4050 είναι ένας μη αναστρέφουσα ρυθμιστικό.

Το εσωτερικό κύκλωμα αποτελείται από 3 στάδια ή 2 - μετατροπείς στάδιο, το οποίο παρέχει υψηλή ανοσία θορύβου και μια σταθερή έξοδο.

Κυκλώματα προστασίας εισόδου είναι διαφορετικά από εκείνα της υψηλής ταχύτητας CMOS SD.

Οι δίοδοι VCC έχουν σχεδιαστεί για να επιτρέπει τη λογική επιπέδου μετατροπή από την υψηλού επιπέδου τάσεις (έως 15 V) προς χαμηλού επιπέδου τάσεις.

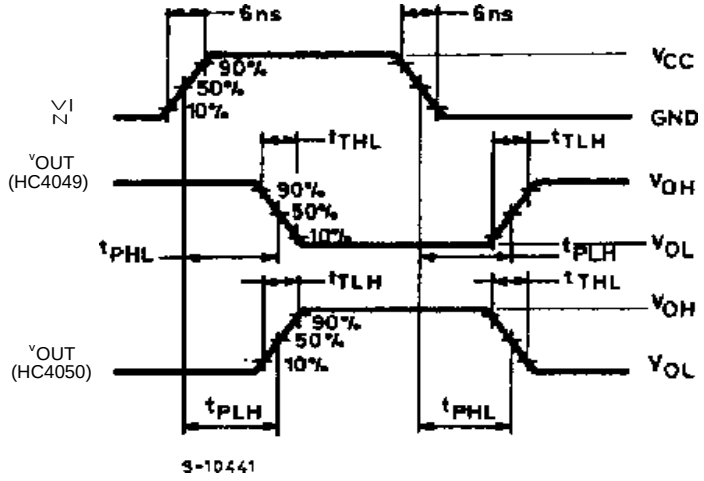


Σχεδιάγραμμα 3 Δίοδος VCC

Για την καλύτερη κατανόηση της λειτουργίας παρατηρούμε τον πίνακα λειτουργίας του:

INPUT	OUTPUT
nA	**nY**
L	H
H	L

Καθώς και την μεταβολή των χαρακτηριστικών:



2.3.3 Ανάλυση Κώδικα

Για να διαβαστεί ή να γραφτεί ένα αρχείο, το αρχείο πρέπει πρώτα να ανοίξει. Για να ανοίξουμε ένα αρχείο το όνομα του αρχείου πρέπει να είναι γνωστό. Υπάρχουν αρκετοί διαφορετικοί τρόποι για να ανοίξουμε ένα αρχείο: ένα αρχείο μπορεί να δημιουργηθεί και να ανοίξει , μπορεί να ανοίξει σε κατάσταση ανάγνωσης, ή μπορεί να είναι ανοιχτό στη λειτουργία εγγραφής. Μπορούμε επίσης να καθορίσουμε το δείκτη του αρχείου που θέλουμε να ξεκινήσουμε την ανάγνωση ή την γραφή. Για παράδειγμα, αν θέλουμε να διαβάσουμε από την αρχή του αρχείου θα ανοίξει το αρχείο όπως αυτό:

```
file.open(root, name, O_READ);    //Open the file in read mode.
```

Ωστόσο, αν θέλουμε να γράψουμε στο τέλος ενός αρχείου θα αλλάξει την εντολή και θα μοιάζει σαν αυτήν:

```
file.open(root, name, O_WRITE | O_APPEND);  //Open the file in

write mode and append the data to the end of the file.
```

Είναι επίσης σημαντικό να θυμόμαστε ότι, όταν θα τελειώσουμε την ανάγνωση ή την εγγραφή δεδομένων στο αρχείο, θα πρέπει να κλείσει. Εάν το αρχείο δεν έχει κλείσει μετά την εγγραφή δεδομένων σε αυτό , τα δεδομένα ενδέχεται να μην καταλήξουν να σωθούν. Είναι επίσης πολύ καλή πρακτική να κλείσουμε αυτό το αρχείο, δεδομένου ότι βοηθά να διατηρηθεί η ακεραιότητα του συστήματος αρχείων. Για να κλείσουμε το αρχείο απλά χρησιμοποιούμε αυτή την εντολή:

```
file.close();  //Close the file
```

Η πιο χρήσιμη πτυχή μιας κάρτας microSD είναι το να γράψουμε πληροφορίες σε ένα αρχείο που μπορεί αργότερα να χρησιμοποιηθεί σε έναν υπολογιστή για να δούμε τις πληροφορίες. Ανεξάρτητα από το τι θέλουμε να καταγράψουμε , θα πρέπει να ξέρουμε πώς να γράψουμε τις πληροφορίες που προέρχονται από τον πραγματικό κόσμο και να το αποθηκεύσουμε σε ένα αρχείο. Μόλις έχουμε δημιουργήσει και ανοίξει ένα αρχείο, η αποθήκευση των δεδομένων είναι πολύ εύκολη. Ας ρίξουμε μια ματιά σε μερικές γραμμές κώδικα που θα αντιγράψει απλά το string "Millis:" ακολουθούμενο από έναν ακέραιο αριθμό σε μια σειρά και στη συνέχεια να αποθηκεύσουμε το string σε ένα αρχείο.

```
file.open(root, name, O_CREAT | O_APPEND | O_WRITE);
```

```
   //Open or create the file 'name' in 'root' for
writing to the end of the file.
sprintf(contents, "Millis: %d\n", millis());
   //Copy the letters 'Millis: ' followed by the
integer value of the millis() function into the
    'contents' array.
file.print(contents);    //Write the 'contents' array
to the end of the file.
file.close();           //Close the file.
```

Έχουμε ανοίξει ένα αρχείο σε κατάσταση εγγραφής, και θα πάμε να γράψουμε στο τέλος του αρχείου. Η λειτουργία του «sprintf ()« είναι μία λειτουργία που μας επιτρέπει να αντιγράψουμε ένα μορφοποιημένο string σε ένα buffer. Στην περίπτωση αυτή, η λειτουργία αντιγράφει το »Millis:« Το κείμενο, τότε θα αντικαταστήσει το '% d' με τον αριθμό που επιστρέφεται από τη συνάρτηση Millis (). Τέλος, έχουμε προσθέσει μια νέα γραμμή χαρακτήρων στο τέλος του string. Όλα αυτά αντιγράφονται για την άμβλυνση των «περιεχόμενων». Η λειτουργία του «Millis ()" είναι μια λειτουργία του Arduino που επιστρέφει απλά τον αριθμό των χιλιοστών του δευτερολέπτου που έχουν περάσει από την στιγμή που το πρόγραμμα άρχισε να τρέχει. Έτσι, για παράδειγμα, αν το πρόγραμμα έχει τρέξει για 200 χιλιοστά του δευτερολέπτου και εκτελέστηκε το »Millis: 200 ', στο κείμενο θα πρέπει να αντιγραφεί ο αριθμός 200. Τώρα έχουμε συλλάβει τα δεδομένα που πρέπει να αποθηκευτούν στο αρχείο. Το μόνο που έχουμε να κάνουμε είναι να εκτυπώσουμε το buffer στο αρχείο. Στο τέλος θα πρέπει να κλείσουμε το αρχείο, αφού ολοκληρώσουμε τη σύνταξη πληροφοριών σε αυτό.

Υπάρχουν πολλοί λόγοι που μπορεί να θέλουμε να διαβάσουμε τις πληροφορίες από ένα αρχείο σε μια κάρτα microSD. Για να διαβάσουμε τα δεδομένα, θα χρειαστούμε ένα buffer για την αποθήκευση των πληροφοριών. Τότε θα χρησιμοποιήσουμε μόνο το «read ()» σε λειτουργία SdFat με την βιβλιοθήκη, για να αποσπάσουμε πληροφορίες από ένα ανοιχτό αρχείο. Σε αυτό το παράδειγμα κώδικα έχει δημιουργηθεί ένας φάκελος με όνομα «file_contents.»

```
charfile_contents[256];          //This is a data

buffer that holds data read from a file
```

Τώρα το μόνο που χρειάζεται είναι να ανοίξουμε ένα αρχείο και να ξεκινήσουμε την ανάγνωση των δεδομένων από αυτό. Υπάρχουν δύο τρόποι για να διαβάσουμε δεδομένα από ένα αρχείο χρησιμοποιώντας τη βιβλιοθήκη SdFat. Μπορούμε να διαβάσουμε ένα χαρακτήρα κάθε φορά ή να διαβάσουμε ένα συγκεκριμένο ποσό των δεδομένων από ένα αρχείο. Θα εξετάσουμε τις δύο μεθόδους. Αν διαβάζουμε ένα χαρακτήρα κάθε φορά που θα πρέπει να κρατήσει το μέγεθος του buffer στη μνήμη , το μέγεθος τουbuffermας θα πρέπει να είναι 256 χαρακτήρες. Εδώ είναι ένα παράδειγμα του πώς να διαβάσει δεδομένα από ένα αρχείο:

```
int index=0;  //Create a variable to keep track of

our position in the data buffer.

file.open(root, name, O_READ);    //Open the file

in read mode.

file_contents[index]=file.read();

                //Get the first byte in the file.

//Keep reading characters from the file until we

get an error or reach the end of the file.

(This will output the entire contents of the file).

while(file_contents[index] >=0 && index < 256){

   //If the value of the character is less than 0 we've reached

the end of the file. If index is 256 than our buffer is full.
```

```
index+=1;                //Move to the next position in the

data buffer.

file_contents[index]=file.read(); //Get the next character

}

file.close();    //Close the file
```

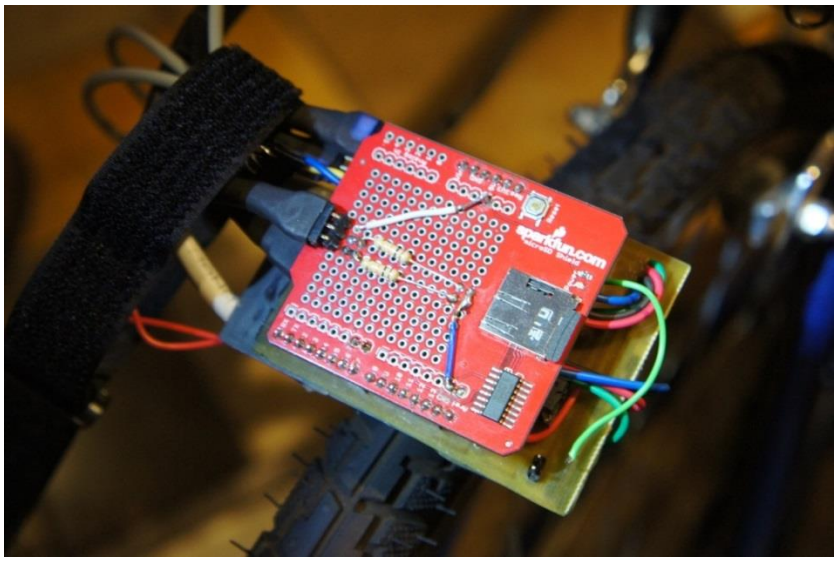
```
for(int i=0; i
```

Σε αυτό το παράδειγμα, διαβάζουμε ένα χαρακτήρα κάθε φορά από το αρχείο και το αποθηκεύουμε σε buffer δεδομένων μας. Ο βρόχος παρακολουθεί δύο προϋποθέσεις. Εάν ο χαρακτήρας που διαβάζεται από το αρχείο είναι 0 τότε έχουμε φτάσει στο τέλος του αρχείου, έτσι ώστε να βγούμε από το βρόχο και να κλείσουμε το αρχείο. Από την άλλη πλευρά, αν η αξία του «δείκτη» φτάσει 256 από οπότε έχουμε γεμίσει πλήρως το buffer δεδομένων, άρα δεν μπορούμε να συνεχίσουμε να προσθέτουμε στοιχεία σε αυτό. Αν συμβεί αυτό, βγαίνουμε από το loop και κλείνουμε το αρχείο. Μετά το κλείσιμο του αρχείου το πρόγραμμα εκτυπώνει τα περιεχόμενα του buffer στην οθόνη.

Αυτή η μέθοδος ανάγνωσης δεδομένων από ένα αρχείο (ένα χαρακτήρα κάθε φορά), μπορεί να είναι λίγο αργή και δυσκίνητη για τον κώδικα. Αν γνωρίζουμε πόσα δεδομένα θα πρέπει να διαβάσουμε από ένα αρχείο και να είμαστε σίγουροι ότι τα δεδομένα του buffer έχουν αρκετό χώρο για να αποθηκευτούν, τότε μπορεί να προτιμήσουμε αυτή τη μέθοδο ανάγνωσης δεδομένων από ένα αρχείο:

```
int index=0;

intdata_size=10;

//This variable sets the number of bytes to read from the file.

index=file.read(file_contents, data_size);

//file_contents is the data buffer for storing data. data_size is a

variable set to the amount of data to read.

file.close();    //Close the file

for(int i=0; i
```

Αυτό το παράδειγμα θα διαβάσει το «data_size» με bytes (10) από το αρχείο και θα αποθηκεύσει τα δεδομένα στο buffer των «file_contents». Η μεταβλητή του «δείκτη» θα περιέχει τον αριθμό των bytes από το αρχείο σε περίπτωση που είναι διαφορετικό από τον αριθμό που καθορίζεται. Αυτό θα μπορούσε να συμβεί αν έχουμε φτάσει στο τέλος του αρχείου πριν από την ανάγνωση των «data_size» bytes.



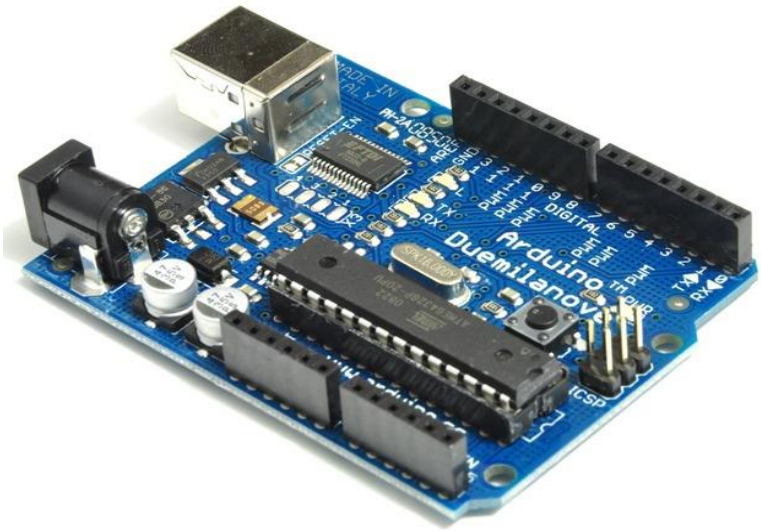
Εικόνα 7 Κάρτα SD

Όπως παρατηρούμε στην πιο πάνω φωτογραφία ξεχωρίζουν οι 2 αντιστάσεις (pulldown resistor) που αντιστοιχούν στα κουμπιά αύξησης και μείωσης (αναφέρεται παρακάτω λεπτομερής περιγραφή).

Ο μικροελεγκτής προγραμματίστηκε σε περιβάλλον Arduino, το οποίο χρησιμοποιεί την λογική της γλώσσας προγραμματισμού C++ .

Για να κατανοήσουμε την λειτουργία του μικροελεγκτή θα αναφέρουμε κάποιες βασικές πληροφορίες.

Όπως το περιγράφει ο δημιουργός του , τα Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «πρωτοτυποποίησης» ηλεκτρονικών βασισμένη σε ευέλικτο και εύκολο στη χρήση hardware και software που προορίζεται για οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία ,στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα.



Εικόνα 8 Arduino

Στην ουσία , πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στον μικροελεγκτή ATmega της Atmel και του οποίου όλα τα σχέδια , καθώς και το software που χρειάζεται για την λειτουργία του , διανέμονται ελεύθερα και δωρεάν ώστε να μπορεί να κατασκευαστεί από τον καθένα. Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, αφού ο χρήστης μπορεί να συνδέσει επάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει τον μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου. Το κύριο πλεονέκτημά του είναι η τεράστια κοινότητα που του υποστηρίζει και η οποία έχει δημιουργήσει, συντηρεί και επεκτείνει μια ανάλογου μεγέθους online γνωσιακή βάση.

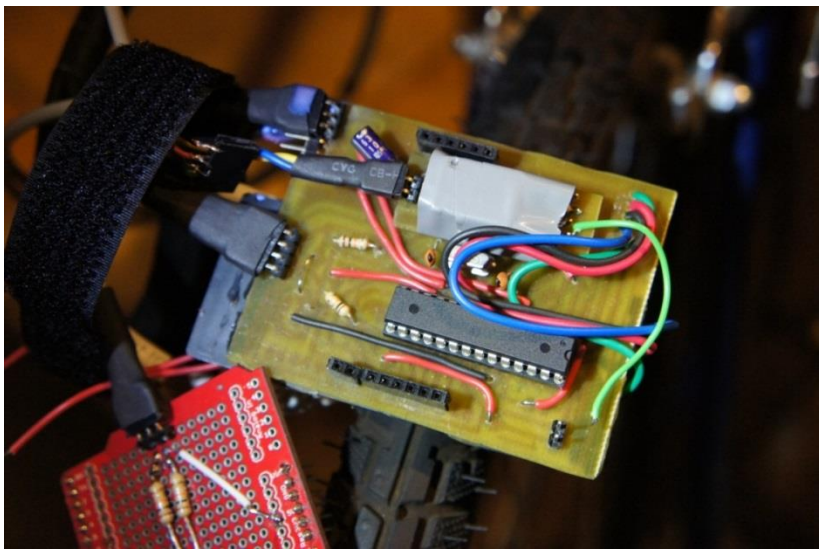
Το Arduino βασίζεται στον ATmega 328, έναν 8-bit RISC μικροελεγκτή, τον οποίο χρονίζει στα 16MHz. Ο ATmega328 διαθέτει ενσωματωμένη μνήμη τριών τύπων:

- 2Kb μνήμης SRAM που είναι η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματά σας για να αποθηκεύουν μεταβλητές, πίνακες κ.λπ. κατά το runtime. Όπως και σε έναν υπολογιστή, αυτή η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή αν γίνει reset.

- 1Kb μνήμης EEPROM η οποία μπορεί να χρησιμοποιηθεί για «ωμή» εγγραφή/ανάγνωση δεδομένων (χωρίς datatype) ανά byte από τα προγράμματά σας κατά

το runtime. Σε αντίθεση με την SRAM, η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset οπότε είναι το ανάλογο του σκληρού δίσκου.

- 32Kb μνήμης Flash, από τα οποία τα 2Kb χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware αυτό που στην ορολογία του Arduino ονομάζεται bootloader είναι αναγκαίο για την εγκατάσταση των δικών σας προγραμμάτων στον μικροελεγκτή μέσω της θύρας USB, χωρίς δηλαδή να χρειάζεται εξωτερικός hardware programmer. Τα υπόλοιπα 30Kb της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή σας. Η μνήμη Flash, όπως και η EEPROM δεν χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή reset. Επίσης, ενώ η μνήμη Flash υπό κανονικές συνθήκες δεν προορίζεται για χρήση runtime μέσα από τα προγράμματά σας, λόγω της μικρής συνολικής μνήμης που είναι διαθέσιμη σε αυτά (2Kb SRAM + 1Kb EEPROM), έχει σχεδιαστεί μια βιβλιοθήκη που επιτρέπει την χρήση όσου χώρου περισσεύει (30Kb μείον το μέγεθος του προγράμματός σας σε μεταγλωττισμένη μορφή).

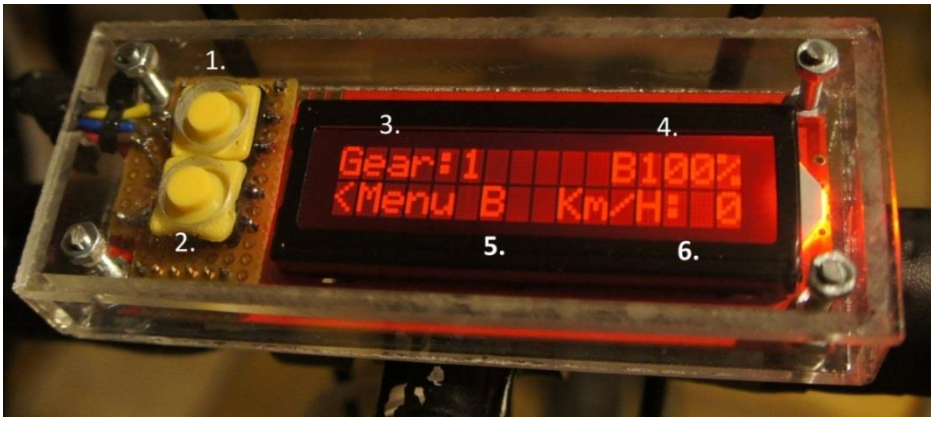


Εικόνα 9 Κύκλωμα (Βασική πλακέτα)

Κεφάλαιο 3

3.1 Manual

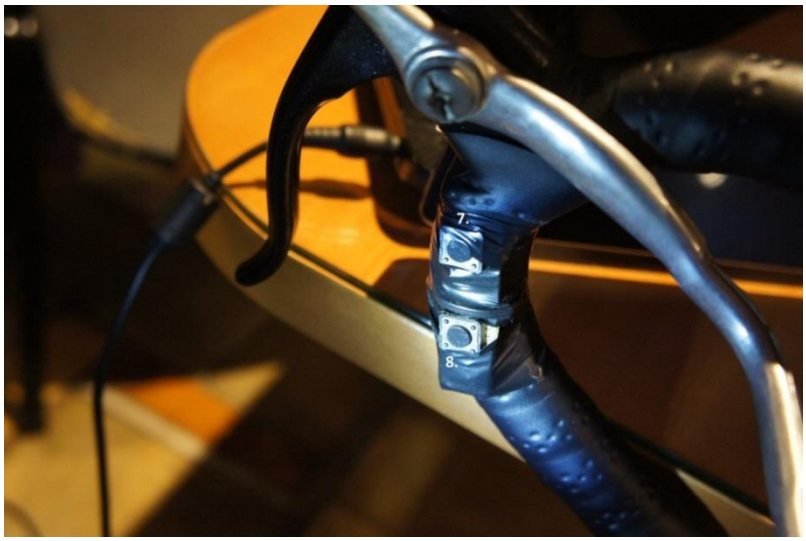
Αρχικά θα κάνουμε μια σύντομη περιγραφή λειτουργίας της οθόνης.



Εικόνα 10 Ανάλυση οθόνης

1. Κουμπί ON
2. Κουμπί OFF(παρατεταμένο για 3 δευτερόλεπτα)/Κουμπί MENU
3. Ένδειξη ταχυτήτων
4. Ένδειξη στάθμης μπαταρίας
5. Ένδειξη προφίλ
6. Ένδειξη χιλιομέτρων

7. Κουμπί αύξησης
8. Κουμπί μείωσης



Εικόνα 11 Κουμπιά επιλογής ταχύτητας

Όταν πατηθεί το κουμπί 2 μπαίνουμε στο MENU όπως βλέπουμε παρακάτω και με τα κουμπιά 7 και 8 επιλέγουμε πιο προφίλ θέλουμε και για να επιστρέψουμε στην αρχική οθόνη πατάμε το κουμπί 2

Έχουμε 4 προφίλ:

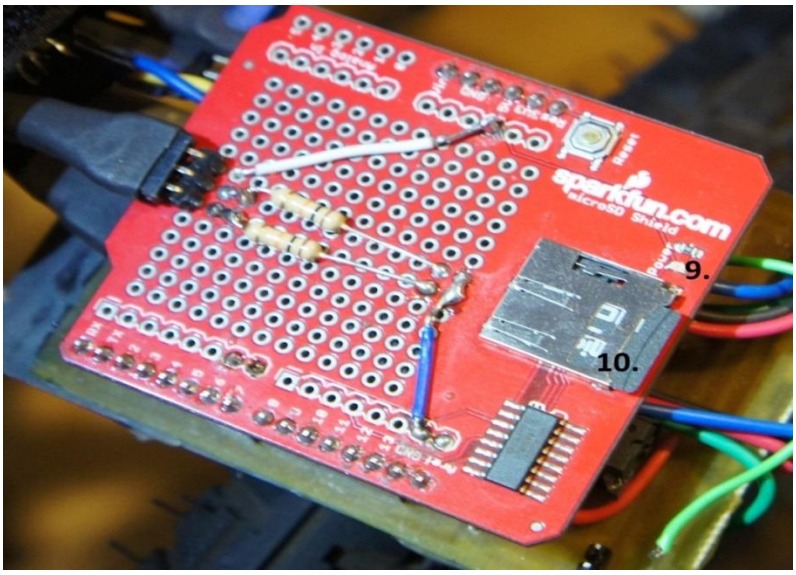
- Beginner (προφίλ για αρχάριους χρήστες το οποίο αλλάζει ψηλά τις στροφές για πιο εύκολη άσκηση)
- Amateur (προφίλ για μέτριους χρήστες το οποίο αλλάζει ταχύτητα σε μεσαίες στροφές για μέτρια άσκηση)
- Custom (προφίλ που ο χρήστης καθορίζει το επίπεδο δυσκολίας που επιθυμεί)
- Manual (προφίλ χειροκίνητης αλλαγής ταχύτητας με τα κουμπιά 7 και 8)

Σε περίπτωση που το ποδήλατο παραμείνει σε ακινησία ή ο χρήστης δεν επεμβαίνει στον αυτοματισμό έως και 5 λεπτά το κύκλωμα απενεργοποιείται και γίνεται αυτόματο σώσιμο των δεδομένων.

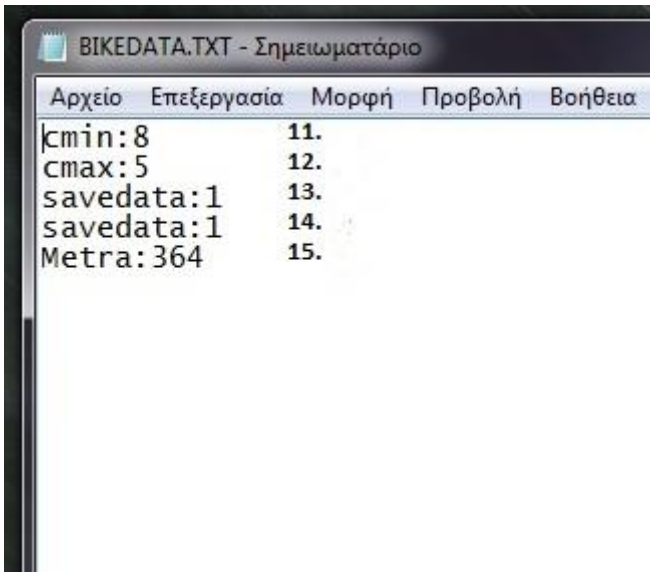
Υπάρχει δυνατότητα αποσύνδεσης της οθόνης και των χειριστηρίων για οποιοδήποτε λόγο επιθυμεί ο χρήστης

Εικόνα 12 Καλωδίωση

Όπως αναφέραμε πιο πάνω ο χρήστης μέσω του προφίλ Custom έχει την δυνατότητα να καθορίζει το επίπεδο δυσκολίας που επιθυμεί. Αυτό το καταφέρνει με την κάρτα αποθήκευσης SD στην οποία μέσου οποιουδήποτε κινητού που έχει εφαρμογή επεξεργασίας κειμένου ή μέσου του υπολογιστή μπορεί να καθορίσει τα επίπεδα δυσκολίας. Επίσης στην κάρτα καταγράφεται μετά από κάθε άσκηση του χρήστη με το ποδήλατο πόσα μέτρα διένυσε.



9. Λαμπάκι ON
10. Κάρτα SD



11. Κατώτερο όριο στροφών (μέγιστη τιμή 9)
12. Μέγιστο όριο στροφών
13. και 14 . Δεν τα πειράζουμε διότι είναι δεδομένα του μικροελεγκτή
15. Απόσταση που διανύσαμε (μηδενίζεται σε κάθε έναρξη του κυκλώματος)

Αν για οποιοδήποτε λόγο το αρχείο BIKEDATA.TXT αλλοιωθεί απλώς το διαγράφουμε και ο μικροελεγκτής θα το δημιουργήσει με τις default τιμές.

Αν για οποιοδήποτε λόγο δεν υπάρχει η κάρτα SD στην θύρα τότε στην οθόνη κατά την έναρξη θα εμφανιστεί το μήνυμα ‘’SDCARDNOTFOUND’’ και θα ανοίξει στις default καταστάσεις (προφίλ beginner ταχύτητα 1).

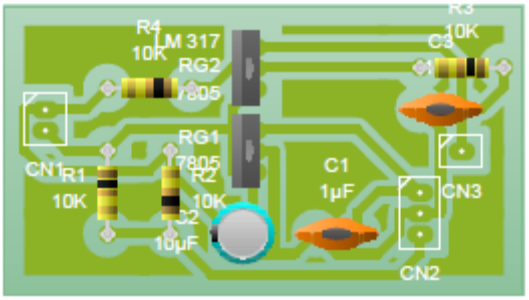
ΠΡΟΣΟΧΗ!!! Η κάρτα SD δεν πρέπει να αφαιρεθεί κατά την εκκίνηση και τον τερματισμό της συσκευής.

\

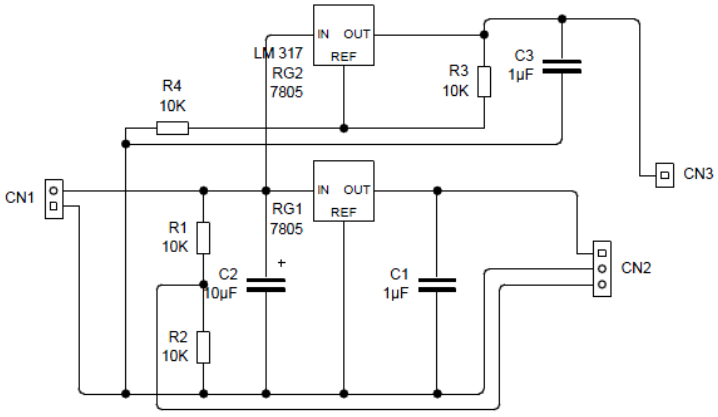
3.2 Κυκλώματα

Για την υλοποίηση του αυτοματισμού έπρεπε να κατασκευάσουμε διάφορα κυκλώματα και πλακέτες. Για την κατασκευή τους χρησιμοποιήθηκε το πρόγραμμα circuit wizard καθώς και φωτοευαίσθητες πλακέτες. Παρακάτω θα αναφέρουμε λεπτομερώς την χρήση κάθε κυκλώματος:

3.2.1 Κύκλωμα τροφοδοσίας



Κύκλωμα 1 Τροφοδοσίας



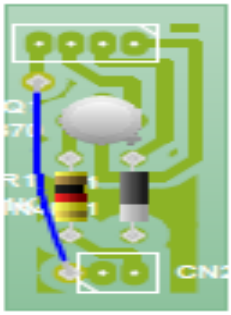
Σχεδιαγραμμα 4 Κύκλωμα τροφοδοσίας

Με το συγκεκριμένο κύκλωμα παρέχουμε τάση 3.3V, 5V και τον έλεγχο της στάθμης της μπαταρίας.

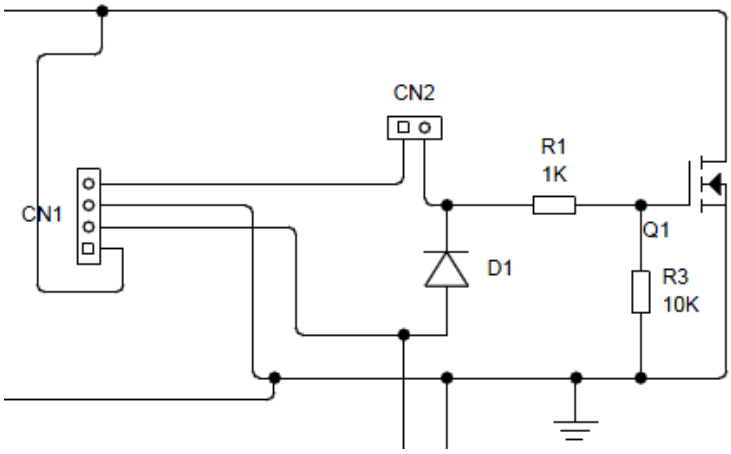
Στο CN1 παρέχουμε τάση από 6.5-10V, στο CN3 παίρνουμε τάση 3.3V που τροφοδοτούμε την κάρτα SD, στο CN2 παίρνουμε τάση 5V, τον έλεγχο της στάθμης της μπαταρίας και την γείωση.

Αναλυτικά για το παραπάνω κύκλωμα χρησιμοποιούμαι 2 σταθεροποιητές τον LM7805 και τον LM317. Ο LM7805 μας παρέχει 5V στο κύκλωμα ενώ ο LM317 μας παρέχει 3.3v. Οι αντιστάσεις R3 καιR4 χρησιμοποιήθηκαν σε διάταξη διαιρέτη τάσης ώστε το LM317 να παρέχει την επιθυμητή τάση ενώ στο LM7805 η συγκεκριμένη διάταξη είναι ενσωματωμένη. Οι πυκνωτές στο παραπάνω κύκλωμα χρησιμοποιούνται για την σταθεροποίηση όλου του κυκλώματος και συγκεκριμένα ο ηλεκτρολυτικός πυκνωτής C2 χρησιμοποιείται για τις απότομες βυθίσεις τάσεις ενώ ο C1 καιC3 χρησιμοποιούνται για τυχόν ανωμαλίες στο ρεύμα. Τέλος οι αντιστάσεις R1 και R2 έχουν τοποθετηθεί σε διάταξη διαιρέτη τάσης για να μας κόβουν την τάση στο μισό ώστε ο μικροελεγκτής να μπορεί να κάνει τον έλεγχο της στάθμης της μπαταρίας.

3.2.2 Κύκλωμα αυτόματης απενεργοποίησης



Κύκλωμα 2 Αυτόματη απενεργοποίηση



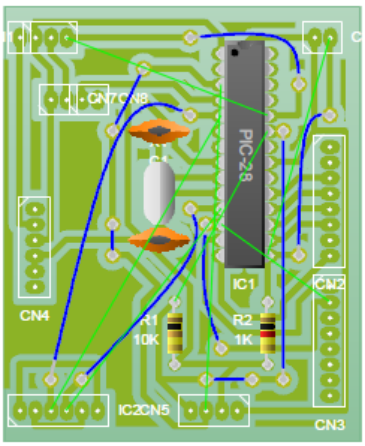
Σχεδιαγραμμα 5 Κύκλωμα αυτόματης απενεργοποίησης

Αυτό το κύκλωμα ενεργοποιεί-απενεργοποιεί την συσκευή.

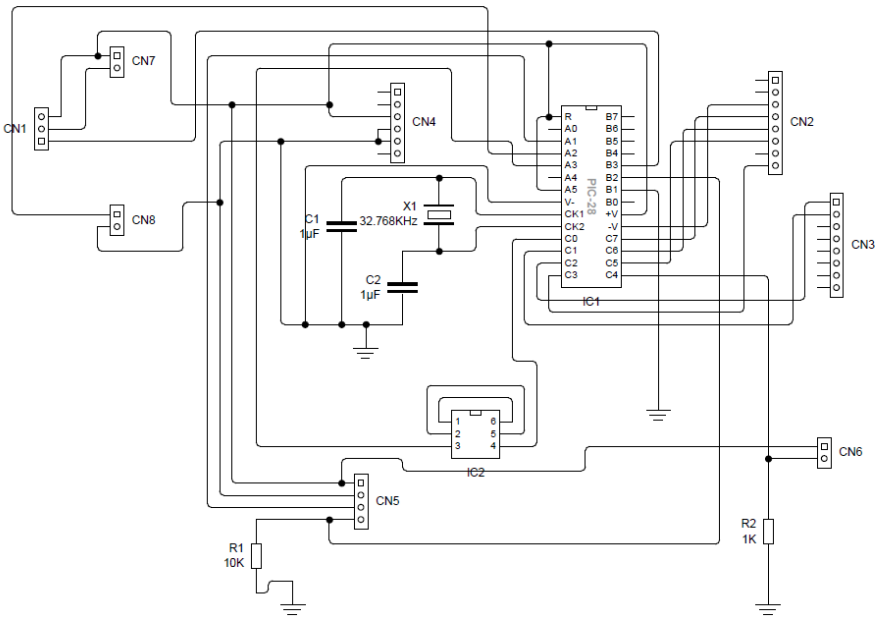
Η επαφή CN1 ενώνεται με την πλακέτα του μικροελεγκτή και η επαφή CN2 συνδέεται με το κουμπί ON της οθόνης. Όλο το κύκλωμα ανοιγοκλείνει με το MOSFETQ1.

Αναλυτικά στο παραπάνω κύκλωμα όταν πατηθεί το κουμπί ON μας στέλνει το αριστερό άκρο της R1 +5v , οπότε το MOSFET αρχίζει και άγει και μας δίνει γείωση στον μικροελεγκτή. Μόλις ο μικροελεγκτής πάρει τάση κάνει εκκίνηση και εν συνεχεία στέλνει 5νστο ένα άκρο της διόδου D1 έτσι ώστε το MOSFETνα συνεχίζει να άγει ακόμη και όταν αφήσουμε το κουμπί ON .

3.2.3 Κύκλωμα μικροελεγκτή (Atmega 328)



Κύκλωμα 3 Μικροελεγκτής



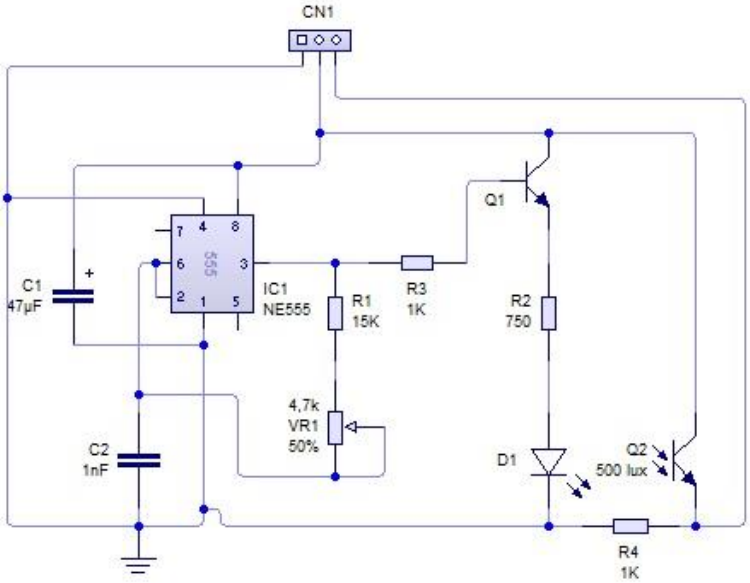
Σχεδιαγραμμα 6 Κύκλωμα μικροελεγκτή

Σε αυτό το κύκλωμα ο μικροελεγκτής επικοινωνεί με την οθόνη, με τα servo, τον αισθητήρα στροφών και την κάρτα SD.Επίσης δέχεται και την τροφοδοσία του.

Στην επαφή IC2 ενώνονται με τα σέρβο. Στην επαφή CN5 ενώνεται η οθόνη. Στις επαφές CN2 , CN3 και CN4 συνδέεται η κάρτα SD. Στις επαφές CN7 και CN8 συνδέεται η πλακέτα αυτόματης απενεργοποίησης. Στην επαφή CN6 συνδέεται ο αισθητήρας στροφών.

Στην επαφή CN1 ενώνεται η τροφοδοσία.

3.2.4 Κύκλωμα οπτικού αισθητηρίου (έλεγχος στροφών)

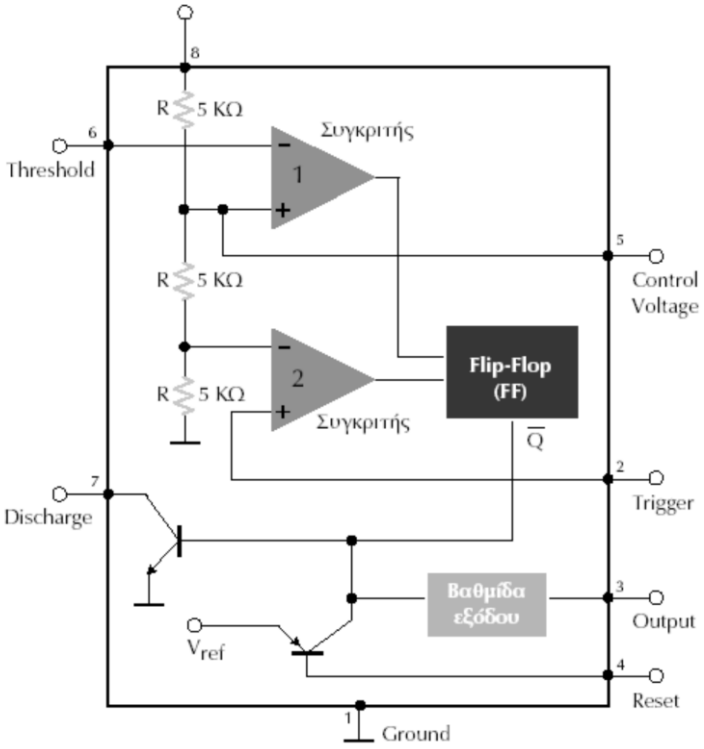


Σχεδιαγραμμα 7 Κύκλωμα οπτικού αισθητηρίου

Για την υλοποίηση αυτού του κυκλώματος χρησιμοποιήσαμε ένα timer 555 που μας παράγει έναν παλμό στα 38KHz που το τροφοδοτούμε σε ένα LED υπερύθρων , που το φως του LEDαντανακλάται στον δέκτη υπερύθρων. Όλο αυτό το κάναμε για να μην επηρεάζεται από εξωτερικό φωτισμό.

Αναλυτικά στο παραπάνω κύκλωμα οι πυκνωτές C1 C2 όπως και οι αντιστάσεις R1 και VR1 χρησιμοποιούνται για να συγχρονίσουν τον timer 555 να βγάλει τον κατάλληλο παλμό (38 khz). Τον συγκεκριμένο παλμό τον στέλνουμε σε ένα τρανζίστορ που ελέγχει το Led. Για δέκτη υπερύθρων χρησιμοποιούμαι το ολοκληρωμένο TSOP 4838 ο οποίος δέχεται τον παλμό 38 khz , το φιλτράρει και μας εξάγει bitsαπό τα πινάκια του.

3.3 Περιγραφή του Timer 555



Σχεδιαγραμμα 8 Timer 555

Όπως παρατηρούμε οTimer 555αποτελείται από ένα διαιρέτη τάσης που έχει τρεις ίσες αντιστάσεις και από2συγκριτές. Ο πρώτος συγκριτής συγκρίνει την τάση threshold με την τάση VCC και ο δεύτερος συγκρίνει την τάση διέγερσης (trigger) με την τάση VCC, όπου VCC= τροφοδοσία του 555.

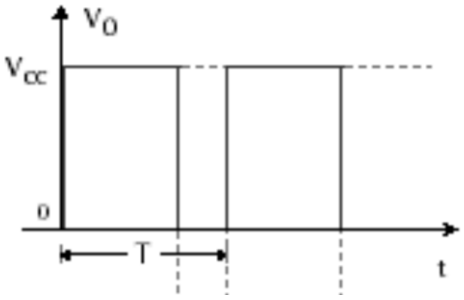
Επίσης αποτελείται από ένα κύκλωμα flip-flop, με τις δύο εισόδους του συνδεδεμένες στις εξόδους των δύο παραπάνω συγκριτών, από μία βαθμίδα εξόδου που λειτουργεί σαν απομονωτής-αντιστροφέας και από ένα τρανζίστορ ενωμένο με τον ακροδέκτη εκφόρτισης.

Η λειτουργία του Timer 555 είναι η εξής:

Όταν ο ακροδέκτης trigger έχει τάση μικρότερη από το VCC τότε ο ακροδέκτης output έχει τάση VCC. Όταν ο ακροδέκτης threshold αποκτήσει τάση μεγαλύτερη του VCC τότε ο ακροδέκτης output έχει τάση0 V. Όταν ο ακροδέκτης reset έχειτάση0 V τότε ο ακροδέκτης output είναι στα 0V. Όταν ο ακροδέκτης αυτός δεν χρησιμοποιείται, είναι συνδεδεμένος στην τάση 0 V. Ο ακροδέκτης control συνήθως χρησιμοποιείται για να οδηγήσει διαφόρους θορύβους στη γη μέσω ενός πυκνωτή των 10 nF.

3.4 Παραγόμενοι παλμοί

Παράγεται μια συνεχή ροή από ορθογώνιους OFFON παλμούς (βλέπε παρακάτω σχήμα). Η συχνότητα των παλμών και οι χρόνοι t1και t2εξαρτώνταιαπότιςτιμέςτωνR1,R2και C.



Υπολογισμός t1, t2:

t1 = 0,693(R1 + R2) × C

t2 = 0,693 × R2 × C

Υπολογισμός συχνότητας λειτουργίας:

0,693 C (R 2R)

f 1

× × 1 + 2

=

Για παραπάνω σχέσεις: t1, t2 σε sec (s), f σε Hz,

R1, R2 σε Ohm (Ω) και C σε Farad (F)

Κύκλος εργασίας:

100%

R 2R

100% R R

t t

D t

1 2

1 2

1 2

1

+

+

=

+

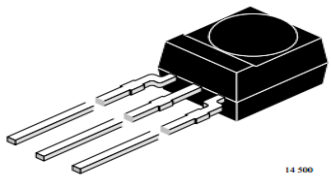
=

Αν R1>>R2 ο κύκλος εργασίας είναι ο μέγιστος

(100%) ενώ αν R1<<R2 ο κύκλος εργασίας είναι ο

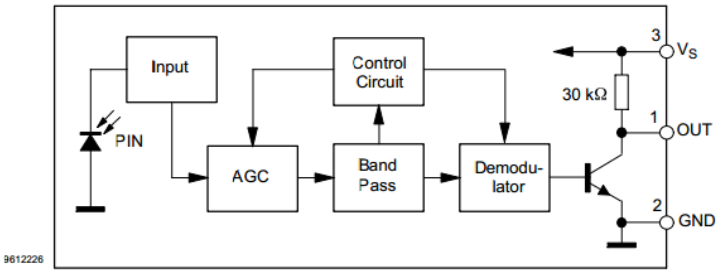
ελάχιστος και ίσος με 50%.

3.5 Περιγραφή του TSOP4838

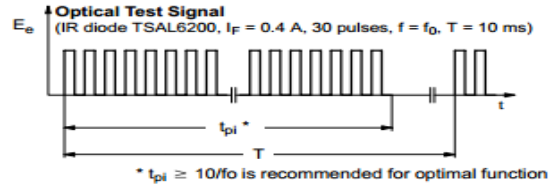


Το TSOP4838 είναι μια μικρογραφία ενός δέκτη υπερύθρων για συστήματα τηλεχειρισμού. Το αποδιαμορφωμένο σήμα εξόδου μπορεί άμεσα να αποκωδικοποιηθεί από έναν μικροεπεξεργαστή. Το TSOP4838 είναι το πρότυπο IR, υποστηρίζοντας όλους τους τύπους κωδικών μετάδοσης.

Block Diagram



Παρακάτω παρατηρούμε το σήμα μηνύματος με το φέρον , όπως και την αποδιαμόρφωση , που έχει κάνει το TSOP4838 , έτσι ώστε να πάρουμε τον καθαρό παλμό.



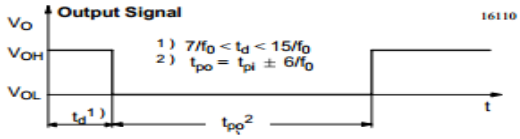
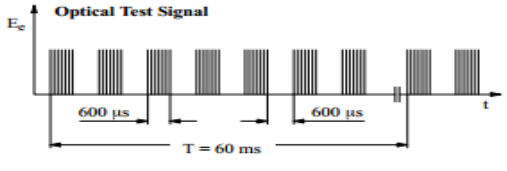
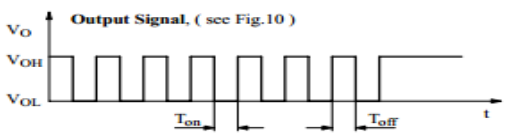


Figure 7.





Κεφάλαιο 4

4.1 Προβλήματα και Λύσεις

Προσπαθώντας να κατασκευάσουμε τον μηχανισμό αυτόματης εναλλαγής ταχυτήτων βρεθήκαμε αντιμέτωποι με διάφορα προβλήματα που έπρεπε να λύσουμε.

Το βασικότερο πρόβλημα που αντιμετωπίσαμε ήταν το θέμα χώρου των κυκλωμάτων. Χρειάστηκε να κατασκευάσουμε δικά μας κυκλώματα σε όσο το δυνατόν μικρότερη επιφάνεια.

Κατά την διάρκεια των δοκιμών μας στην τοποθέτηση των υλικών , όταν συνδέσαμε την οθόνη στο υπόλοιπο κύκλωμα, παρατηρήσαμε πρόβλημα στην ταχύτητα επικοινωνίας του μικροελεγκτή με την οθόνη, με αποτέλεσμα να επηρεάζει τις υπόλοιπες εντολές. Έπρεπε, τελικά να αυξήσουμε το baudrate στα 38.300 kbps και να αλλαχτούν οι εντολές επικοινωνίας. Επίσης υπήρχε πρόβλημα και με την τροφοδοσία της οθόνης, δηλαδή κατά την εκκίνηση του κυκλώματος οι σερβοκινητήρες τραβούσαν απότομα ρεύμα με αποτέλεσμα να ξεσυγχρονίζεται η οθόνη. Το πρόβλημα λύθηκε τοποθετώντας δύο πυκνωτές για να μην υπάρχουν βυθίσεις τάσης. Ένα επιπρόσθετο πρόβλημα ήταν ότι η EEP Rom της οθόνης δεν αποθήκευε δεδομένα, οπότε έπρεπε κάθε φορά να στέλνουμε εντολές συγχρονισμού στην οθόνη.

Στο μηχανικό σκέλος, αρχικά στο πίσω ντεραγιέρ είχαμε τοποθετήσει έναν σερβοκινητήρα με μικρότερη ροπή από τον υπάρχον, με αποτέλεσμα να γίνονται λανθασμένα οι αλλαγές στην ταχύτητα (κάποιες φορές αντί για μία ταχύτητα άλλαζε τρεις). Αργότερα σκεφτήκαμε να τοποθετήσουμε ένα έμβολο με περισσότερη δύναμη το οποίο θα τραβούσε το συρματόσχοινο, που αλλάζει την ταχύτητα του ντεραγιέρ, αλλά κατά την διάρκεια των δοκιμών χάλασε το κιβώτιο ταχυτήτων του εμβόλου. Επίσης είχε υπερβολική κατανάλωση και το κύκλωμα έβγαινε υπερβολικά μεγάλο (περιείχε Η-γέφυρα, τελεστικούς ενισχυτές, σταθεροποιητές κλπ.) Οπότε καταλήξαμε σε έναν σερβοκινητήρα με μεγαλύτερη ροπή και οι αλλαγές γινόντουσαν ομαλά όπως επιθυμούσαμε.

Επίσης αντιμετωπίσαμε ένα πρόβλημα στον ελεγκτή στροφών πεταλιού. Ο πρώτος ελεγκτής που χρησιμοποιήσαμε ήταν μαγνητικός, ο οποίος αποτελείτο από έναν μαγνήτη που ήταν στερεωμένος πάνω στο πετάλι και μια επαφή που ανοιγόκλεινε με τον μαγνήτη σε κάθε

περιστροφή. Αυτό είχε σαν αποτέλεσμα να μην μας δίνει σωστά αποτελέσματα λόγω της ταχύτητας στον μικροελεγκτή. Οπότε καταλήξαμε να χρησιμοποιήσουνε έναν οπτικό ελεγκτή, που δουλεύει με υπέρυθρες, ο οποίος αποτελείτο από ένα Led υπερύθρων που αναβοσβήνει με συχνότητα 38Khz το οποίο αντανακλούσαν πάνω σε έναν δίσκο που είχαμε τοποθετήσει και το αντιλαμβανόταν ένας αισθητήρας υπερύθρων. Ο δίσκος δεν είχε το επιθυμητό αποτέλεσμα διότι είχαμε πρόβλημα με τις αντανακλάσεις του Led. Άρα το αφαιρέσαμε και η αντανάκλαση γίνεται πάνω στα μπράτσα στήριξης του πεταλιού.

4.2 Πιθανές βελτιώσεις

Στην παρούσα πτυχιακή εργασία έγινε ένα πρώτο βήμα έτσι ώστε να αυτοματοποιηθεί ο τρόπος αλλαγής ταχυτήτων.

Φυσικά υπάρχει χώρος για μεγαλύτερες βελτιώσεις οι οποίες ενδεικτικά μπορεί να είναι οι εξής.

Μια πιθανή βελτίωση είναι η αφαίρεση της οθόνης και στην θέση της θα τοποθετείται συσκευή ANDROID,IOS στην οποία θα έχουμε εγκατεστημένη εφαρμογή όπου θα αναγράφονται όλες οι υπάρχουσες λειτουργίες που έχουμε στην οθόνη και με επιπλέον δυνατότητα GPS. Με αυτή την βελτίωση πετυχαίνουμε την αφαίρεση της οθόνης και του card reader, οπότε έχουμε εξοικονόμηση χώρου, οικονομία στην μπαταρία και οικονομικότερη κατασκευή.

Βασικό προτέρημα στην υπάρχουσα κατασκευή μας θα είναι η τοποθέτηση δυναμό το οποίο θα μας παρέχει ηλεκτρικό ρεύμα για να φορτίζουμε την μπαταρία του μηχανισμού. Έτσι θα έχουμε εξοικονόμηση ενέργειας.

Στο μηχανικό μας σκέλος μια σημαντική παρέμβαση θα είναι η τοποθέτηση ηλεκτρικών εμβόλων (αντί των σερβοκινητήρων που έχουμε τοποθετήσει ήδη), διότι μπορούν να τοποθετηθούν χωρίς να υπάρχει καμία αλλαγή στα ντεραγιέρ.

Για ένα ολοκληρωμένο αυτοματοποιημένο ποδήλατο, το οποίο το χρησιμοποιούν παραπάνω από ένας χρήστες, είναι να τοποθετηθούν μηχανισμοί στην σέλα και στο τιμόνι, ώστε ανάλογα τον χρήστη να υπάρχει αυτόματη αυξομείωση στο ύψος όπως έχουν καθοριστεί.

Τέλος θα μπορούσαμε να χρησιμοποιήσουμε κυκλώματα SMD για μείωση όγκου και οικονομίας.

Βιβλιογραφία

http://kapogiannis-bikes.gr/

http://arduino.cc/en/reference/servo

http://coolweb.gr/istoria-tou-podilatou/

http://arduino.cc/en/Tutorial/AnalogInput

users.sch.gr/giannouleask/BICYCLE.doc

http://www.northbike.gr/

http://www.cyclist.gr/article.php?id=1495

http://grobotronics.com/serial-enabled-16x2-lcd-red-on-black-5v.html#.Uits1sZSi5h

http://robotstore.gr/?main_page=product_info&cPath=1476_1521&products_id=5520&zenid=a2c4a7dbbc00c62da0dd28d8a8c463c7

http://www.why.gr/#/state/itemCard/ID/709520/language/el_GR

http://arduino.cc/en/tutorial/button

http://arduino.cc/en/Hacking/ParallelProgrammer?from=Main.ParallelProgrammer

http://robotstore.gr/hitec-spline-metal-servo-horn-tapped-9635.html

http://www.servocity.com/html/hs-645mg_ultra_torque.html

http://www.atomik-rc.com/Power-HD-HD-3689MG-High-SpeedDigital-Servo.html

http://arduino.cc/en/Reference/SD

http://playground.arduino.cc/Code/LCD

http://hildstrom.com/projects/sparkfunserlcd/index.html

http://www.ohmslawcalculator.com/555_astable.php

http://www.vishay.com/docs/82459/tsop48.pdf

http://www.ladyada.net/learn/arduino/lesson4.html

http://www.tested.com/tech/robots/456466-know-your-arduino-guide-most-common-boards/

http://en.wikipedia.org/wiki/Derailleur_gears

http://en.wikipedia.org/wiki/Bicycle

Παράρτημα Α

Προγραμματισμός

Ο προγραμματισμός έγινε σε περιβάλλον Arduino. Επειδή οι γραμμές εντολών ήταν πολλές , χρησιμοποιήσαμε υπορουτίνες για διευκόλυνσή μας.

a. Υπορουτίνα μέτρησης στροφών

```
1.  voidmetrisistrofon(){
2.  SensorState = digitalRead(Sensor);//Διαβάζει την ψηφιακή είσοδο και τη περνάει μέσα σε μια σταθερά
3.  if (SensorState == HIGH && previous == LOW ) {//Ανιχνεύει αισθητήρα
4.  x++;//Αυξάνει τον μετρητή
5.  w = 0;
6.  ctax++;
7.  if (ctax == 5){
8.  kph = 0;//(Από 8-17) Υπολογίζει την ταχύτητα του ποδηλάτου σε σχέση με τις στροφές του πεταλιού
9.  sec = t * 0.007;
10. mps = roda / sec;
11. kph = mps * 3.600;
12. sec = 0.000;
13. mps = 0.000;
14. t = 0;
15. ctax = 0;
16. metra = metra + roda
17. apost = metra;
```

Αρχικά έχουμε μια μεταβλητή X που αυξάνεται κάθε φορά που το οπτικό αισθητήριο ανιχνεύσει πέρασμα του πεταλιού και έτσι υπολογίζουμε τις στροφές, όπως και με το ctax υπολογίζουμε την ταχύτητα του ποδηλάτου.

b. Υπορουτίνα αλλαγής ταχύτητας

```
1.  voidalagitaxititas(){
2.  if (x <minstrofes){//Συγκρίνει τις στροφές του πεταλιού που έχουμε μετρήσει σε σχέση με μία σταθερά
3.  if ( x > 5){
4.  gear--;//Μειώνει ταχύτητα


5.  if (gear< 1){//(Από 5-6)Μας κρατάει το όριο της ταχύτητας
6.  gear = 1;
7.  if (x >maxstrofes) {
8.  gear++;//Αυξάνει ταχύτητα
9.  if (gear> 10){//(Από 9-10)Μας κρατάει το όριο της ταχύτητας
10. gear = 10;
11. }
```

Εφόσον έχουμε υπολογίσει τις στροφές, τις συγκρίνουμε με κάποιες σταθερές και υπολογίζουμε αν θα αυξηθεί , θα μειωθεί , ή θα μείνει η ίδια ταχύτητα. Το gear είναι η μεταβλητή της κάθε ταχύτητας.

c. Υπορουτίνα επιλογής θέσης

```
1.  voidepilogithesis(){
2.  switch (gear) {//Ανάλογα με την ταχύτητα επιλέγει την κατάλληλη συνθήκη
3.  case 1:
4.  var = emv1;//(Από 4-6) Μας βάζει στην μεταβλητή την θέση που πρέπει να έχει ο σερβοκινητήρας
5.  var3 = emv1;
6.  var2 = thesi1;
7.  roda = 2.400;//Υπολογίζει τα χλμ του ποδηλάτου
8.  if (au == 1){//(Από 8-9) Με αυτή την συνθήκη όταν γίνεται κατέβασμα ταχυτήτων , αυξάνει λίγο παραπάνω την θέση του servo για σωστή αλλαγή
9.  var = emv1 + 80;
10. }
11. break;
12. case 2:
13. var = emv2;
14. var3 = emv2;
15. var2 = thesi1;
16. roda = 3.050;
17. if (au == 1){
18. var = emv2 + 180;
19. }
20. break;
21. case 3:
22. var = emv3;
23. var3 = emv3;
24. var2 = thesi1;
25. roda = 3.900;
26. if (au == 1){
27. var = emv3 + 230;
28. }
29. break;
30. case 4:
31. var = emv4;
```

```
32.     var3 = emv4;
33.     var2 = thesi1;
34.     roda = 4.600;
35.     if (au == 1){
36.     var = emv4 + 260;
37.     }
38.     break;
39.     case 5:
40.     var = emv5;
41.     var3 = emv5;
42.     var2 = thesi1;
43.     roda = 5.400;
44.     break;
45.     case 6:
46.     var = emv1;
47.     var3 = emv1;
48.     var2 = thesi2;
49.     roda = 2.920;
50.     if (au == 1){
51.     var = emv1 + 80;
52.     }
53.     break;
54.     case 7:
55.     var = emv2;
56.     var3 = emv2;
57.     var2 = thesi2;
58.     roda = 3.500;
59.     if (au == 1){
60.     var = emv2 + 180;
61.     }
62.     break;
63.     case 8:
64.     var = emv3;
65.     var3 = emv3;
```

```
66.     var2 = thesi2;
67.     roda = 4.290;
68.     if (au == 1){
69.     var = emv3 + 230;
70.     }
71.     break;
72.     case 9:
73.     var = emv4;
74.     var3 = emv4;
75.     var2 = thesi2;
76.     roda = 4.930;
77.     if (au == 1){
78.     var = emv4 + 260;
79.     }
80.     break;
81.     case 10:
82.     var = emv5;
83.     var3 = emv5;
84.     var2 = thesi2;
85.     roda = 5.900;
86.     break;
87.     }
88.     }
```

Σε αυτή την υπορουτίνα , ανάλογα με την ταχύτητα που έχουμε εκείνη την χρονική στιγμή , επιλέγουμε τις θέσεις των σερβοκινητήρων , με τις μεταβλητές var, var2 και var3.

d. Υπορουτίνα κίνησης servo

```
1. voidkinisiservo(){
2. myservo.attach(bservo);//Δεσμεύει στο servo το συγκεκριμένο πινάκι
3. myservo2.attach(fservo);
4. myservo.writeMicroseconds(var);//Γράφει ταMicrosecondsστοservo
5. myservo2.writeMicroseconds(var2);
```

Σε αυτή την υπορουτίνα στέλνουμε τις εντολές στα servo για να κινηθούν.

e. Υπορουτίνα επιλογής προγράμματος

```
1.  voidprogselect(){
2.  if (menuon == 0){//(Από 2-4) Λειτουργία εισόδου-εξόδου από tomenu κατά την εκκίνηση του μικροελεγκτή
3.  toggle = 1;
4.  menuon = 1;
5.  }
6.  if (menu == 1 &&menupre == 0){  //(Από 6-16) Λειτουργία εισόδου-εξόδου από tomenu χωρίς εκκίνηση του μικροελεγκτή
7.
8.  if (toggle == 0){
9.  toggle = 1;
10. }
11. else {
12. toggle = 0;
13.
14. }
15. }
16. menupre = menu;
17. if (toggle == 1){//(Από 17-32) Τύπωμα παραμέτρων στη οθόνη ανάλογα με το προφίλ
18. w = 0;
19. Serial.print("Menu");
20. Serial.print("          ");
21. Serial.print("Profil:");
22. if (profil == 1){
23. Serial.print(" Beginner");
24. }
25. if (profil == 2){
26. Serial.print(" Amateur ");
27. }
28. if (profil == 3){
29. Serial.print(" Custom  ");
30. }
31. if (profil == 4){
```

```
32.  Serial.print(" Manual  ");
33.  if (buttonstate1 == HIGH && buttonpre1 == LOW){//(Από 33-40) Επιλογή προφίλ
34.  profil++;
35.  if (profil> 4){
36.  profil =  4;   }
37.  if (buttonstate2 == HIGH && buttonpre2 == LOW){
38.  profil--;
39.  if (profil< 1){
40.  profil =  1;
41.  }
42.  }
43.  switch (profil) {//(Από 43-61) Ορίζει τις μεταβλητές του κάθε προφίλ
44.  case 1:
45.  minstrofes = arxariosmin;
46.  maxstrofes = arxariosmax;
47.  manual = 0;
48.  break;
49.  case 2:
50.  minstrofes = metriosmin;
51.  maxstrofes = metriosmax;
52.  manual = 0;
53.  break;
54.  case 3:
55.  minstrofes = custommin;
56.  maxstrofes = custommax;
57.  manual = 0;
58.  break;
59.  case 4:
60.  manual = 1;
61.  break;
62.  }
63.  }
64.  if (menuoff == 0){
65.  toggle = 0;
```

```
66.     menuoff = 1;
67.     }
```

Σε αυτή την υπορουτίνα όταν πατηθεί το κουμπί menu , η μεταβλητή menu γίνεται 1 και εισέρχεται μέσα στο menu και κάνει και τις ανάλογες εκτυπώσεις στην οθόνη. Η μεταβλητή profil επιλέγει σε ποιο προφίλ θα είμαστε.

f. Υπορουτίνα χειροκίνητης αλλαγής ταχυτήτων

```
1. voidmanualvoid(){
2. if (manual == 1){//Επιλογή προφίλ ‘’Manual’’
3. if (buttonstate1 == HIGH&&buttonpre1 == LOW){//(Από 3-8) Αυξομείωση ταχυτήτων σε σχέση με τα πλήκτρα
4. gear++;
5. au = 0;
6. if (gear > 10){
7. gear = 10;
8. }
```

Σε αυτή την υπορουτίνα , όταν η μεταβλητή manual γίνεται 1 μπορούμε με το κουμπί αύξησης να ανεβάσουμε ταχύτητα.

g. Υπορουτίνα στάθμης μπαταρίας

```
1.  voidbatterylvl(){//(Από 1-6) Υπολογισμός της στάθμης της μπαταρίας. Αν είναι κάτω από 635 (20%) κλείνει ο μικροελεγκτής
2.  if (battery>= 635){
3.  batteryvol = battery - 620;
4.  batteryvol = batteryvol/3;
5.  if (batteryvol> 100){
6.  batteryvol = 100;
7.  }
8.  }
9.  else{
10. if (nosd == 1){//(Από 10-30) Όταν η στάθμη της μπαταρίας είναι κάτω από το 20% και υπάρχει και κάρτα μνήμης , σώζει τα υπάρχον δεδομένα , τυπώνει το μήνυμα στην οθόνη και σβήνει τον μικροελεγκτή
11. custommin = custommin - 10;
12. custommax = custommax - 20;
13. Serial.write(0xFE);
14. Serial.write(0x01);
15. delay(100);
16. Serial.print("Low Battery     Saving Data ");
17. SD.remove("Bikedata.txt");
18. myFile = SD.open("Bikedata.txt", FILE_WRITE);
19. // if the file opened okay, write to it:
20. if (myFile) {
21. myFile.print("cmin:");
22. myFile.println(custommin);
23. myFile.print("cmax:");
24. myFile.println(custommax);
25. myFile.print("savedata:");
26. myFile.println(profil);
27. myFile.print("savedata:");
28. myFile.println(gear);
a.  // close the file:
29. myFile.close();
```

```
30.     }
31.     delay(1000);     ///(Από 31-46) Όταν η στάθμη της μπαταρίας είναι κάτω από το 20% και δεν υπάρχει και κάρτα μνήμης ,δεν  σώζει τα υπάρχον δεδομένα , τυπώνει το μήνυμα στην οθόνη και σβήνει τον μικροελεγκτή
32.     Serial.write(0xFE);
33.     Serial.write(0x01);
34.     delay(100);
35.     digitalWrite(pinon, LOW);
36.     }else{
37.     Serial.write(0xFE);
38.     Serial.write(0x01);
39.     delay(100);
40.     Serial.print("Low Battery");
41.     delay(1000);
42.     Serial.write(0xFE);
43.     Serial.write(0x01);
44.     delay(100);
45.     digitalWrite(pinon, LOW);
46.     }
```

Σε αυτή την υπορουτίνα μετράμε την στάθμη της μπαταρίας και την αναγράφουμε στην οθόνη , με την μεταβλητή batteryvol. Αν η στάθμη της μπαταρίας πέσει κάτω από 20% και η μεταβλητή nosd είναι ίση με 1 (περιέχεται κάρτα sd) , τότε γίνεται αυτόματο σώσιμο των μεταβλητών που χρειαζόμαστε και το κύκλωμα απενεργοποιείται.

h. Υπορουτίναvoid setup

```
1.  void setup(){
2.  pinMode(10, OUTPUT);//(Από 2-7) Ορίζει εισόδους-εξόδους μικροελεγκτή
3.  pinMode(button1, INPUT);
4.  pinMode(button2, INPUT);
5.  pinMode(pinon, OUTPUT);
6.  pinMode(Sensor, INPUT);
7.  digitalWrite(pinon, HIGH);
8.  delay(2000);
9.  Serial.begin(9600);//(Από 9-19) Συγχρονίζεται ο μικροελεγκτής με την οθόνη και μας ορίζει το baud-rate
10. delay(1000);
11. Serial.write(0x7c);
a.  Serial.write(185);
12. delay(100);
13. Serial.write(0x7c);
14. Serial.write(0xef);
15. Serial.end();
16. delay(100);
17. Serial.begin(38400);
18. Serial.write(0xFE);
a.  Serial.write(0x01);
19. delay(100);
20. battery = analogRead(batterypin);//(Από 20-27) Αν κατά την εκκίνηση του μικροελεγκτή ανιχνεύσει ότι η στάθμη της μπαταρίας είναι κάτω από το 20% , τότε τερματίζεται η λειτουργία του
21. if (battery <= 630){
22. Serial.print("Low Battery");
23. delay(1000);
24. Serial.write(0xFE);
25. Serial.write(0x01);
26. delay(100);
27. digitalWrite(pinon, LOW);
```

```
28.     }
29.     if (!SD.begin(8)) {//Ανιχνεύει αν έχουμε κάρτα SD
30.     Serial.print("SDCardNotFound");//(Από 30-35) Τυπώνει το αντίστοιχο μήνυμα ότι δεν έχουμε κάρτα SD
31.     nosd = 0;
32.     delay(1000);
33.     Serial.write(0xFE);
34.     Serial.write(0x01);
35.     delay(100);
36.     }else{
37.     nosd = 1;//(Από 37-46) Αν ανιχνεύσει ότι υπάρχει κάρτα SD αλλά δεν υπάρχει το αρχείο που αποθηκεύουμε τα δεδομένα , τότε το δημιουργεί με default τιμές
38.     // Create file if not exist!
39.     if (!SD.exists("Bikedata.txt")){
40.     myFile = SD.open("Bikedata.txt", FILE_WRITE);
41.     if (myFile) {
42.     myFile.println("cmin:3");
43.     myFile.println("cmax:5");
44.     myFile.println("savedata:1");
45.     myFile.println("savedata:1");
46.     myFile.close();
47.     }
48.     myFile = SD.open("Bikedata.txt");//(Από 48-72) Ανοίγει το φάκελο της κάρτας SD και ανακτά τα δεδομένα μας (κάνει κάποιες μαθηματικές πράξεις για να αποκόψει τους ASCII χαρακτήρες)
49.     if (myFile) {
50.     for (p = 0; p<39; p++){
51.     l = myFile.read();
52.     // Serial.print(l);
53.     if (p==5){
54.     custommin = o & l;
55.     }
56.     if (p==13){
57.     custommax = o & l;
58.     }
```

```
59.  if (p==25){
60.  profil = o & l;
61.  if (profil == 4){
62.  manual =1;
63.  }
64.  }
65.  if (p==37){
66.  gear = o & l;
67.  }
68.  if (p==38){
69.  if (l == 48){
70.  gear = 10;
71.  }else{
72.  myFile.close();
73.  } else {
74.  Serial.print("ErrorOpeningBikedata");//(Από 74-78) Μας εμφανίζει μήνυμα λάθους στην περίπτωση που δεν μπορεί να ανοίξει το αρχείο
75.  delay(1000);
76.  Serial.write(0xFE);
77.  Serial.write(0x01);
78.  delay(100);
79.  }
80.  custommin = custommin + 10;
81.  custommax = custommax + 20;
82.  }
```

Σε αυτή την υπορουτίνα αρχικά αν η μεταβλητή battery είναι μικρότερη του 630 και δεν υπάρχει κάρτα sd , τότε αυτόματα απενεργοποιείται ο μικροελεγκτής. Επίσης κάνουμε αναγνώριση αν υπάρχει κάρτα sd , αν υπάρχει ο φάκελος της κάρτας sd και αν δεν υπάρχει τον δημιουργούμε με default τιμές και ύστερα διαβάζουμε τις μεταβλητές από την κάρτα sd.

i. Υπορουτίναloop

```
1.  void loop(){
2.  for ( inti = 0; i< 280 ; i++){
3.  t++;
4.  if (t > 240){
5.  kph = 0;
6.  }
7.  w++;
8.  op++;
9.  buttonstate1 = digitalRead(button1);//(Από 9-12) Διαβάζει τις εισόδους από τα πινάκια και τις περνάει στις μεταβλητές
10. buttonstate2 = digitalRead(button2);
11. amenu = analogRead(menupin);
12. battery = analogRead(batterypin);
13. if (amenu> 717){//(Από 13-22) Απενεργοποιεί τον μικροελεγκτή υπό κατάλληλες συνθήκες
14. menu = 1;
15. off++;
16. }else{
17. menu = 0;
18. off = 0;
19. }
20. if (off == 420){
21. off3 = 1;
22. // digitalWrite(pinon, LOW);
23. }
24. batterylvl();//(Από 24-26) Καλεί τις αντίστοιχες συνθήκες
25. metrisistrofon();
26. epilogithesis();
27. if (var != provar){
28. provar = var;
29. op = 0;
30. }
31. progselect();
```

```
32.  if (toggle == 0){
33.  manualvoid();
34.  if (buttonstate2 == HIGH&&buttonpre2 == LOW){//(Από 34-41) Μας μειώνει τις ταχύτητες σε οποιοδήποτε προφίλ , όταν πατηθεί το button κατεβάσματος ταχύτητας
35.  gear--;
36.  au = 1;
37.  if (gear < 1){
38.  gear = 1;
39.  }
40.  buttonpre1 = buttonstate1;
41.  buttonpre2 = buttonstate2;
42.  if (toggle == 0){    //(Από 42-75) Τυπώνει στην οθόνη τα δεδομένα του κεντρικού μενού
43.  Serial.print("Gear:");
44.  Serial.print(gear);
45.  if (gear < 10){
46.  Serial.print("     B");
47.  }
48.  else{
49.  Serial.print("    B");
50.  }
51.  if (batteryvol>=100){
52.  Serial.print(batteryvol);
53.  }else{
54.  Serial.print(" ");
55.  Serial.print(batteryvol);
56.  }
57.  Serial.print("%<Menu ");
58.  if (profil == 1){
59.  Serial.print("B");
60.  }
61.  if (profil == 2){
62.  Serial.print("A");
63.  }
64.  if (profil == 3){
```

```
65.     Serial.print("C");
66.     }
67.     if (profil == 4){
68.     Serial.print("M");
69.     }
70.     Serial.print("  Km/H:");
71.     if (kph<10){
72.     Serial.print(" 0");
73.     }else{
74.     Serial.print(kph);
75.     }
76.     if (w == 42000 || off3 == 1){//(Από 76-98) Μετά από w χρονικό απενεργοποιείται ο μικροελεγκτής , αποθηκεύοντας τα δεδομένα στην  κάρτα SD
77.     custommin = custommin - 10;
78.     custommax = custommax - 20;
79.     w = 0;
80.     Serial.write(0xFE);
81.     Serial.write(0x01);
82.     delay(100);
83.     if (nosd == 1){
84.     Serial.print("Saving Data");
85.     SD.remove("Bikedata.txt");
86.     myFile = SD.open("Bikedata.txt", FILE_WRITE);
87.     if (myFile) {
88.     myFile.print("cmin:");
89.     myFile.println(custommin);
90.     myFile.print("cmax:");
91.     myFile.println(custommax);
92.     myFile.print("savedata:");
93.     myFile.println(profil);
94.     myFile.print("savedata:");
95.     myFile.println(gear);
96.     myFile.print("Metra:");
97.     myFile.print(apost);
```

```
98.     myFile.close();
99.     }
100.    delay(1000);//(Από 100-104) Κλείνει ο μικροελεγκτής
101.    Serial.write(0xFE);
102.    Serial.write(0x01);
103.    delay(100);
104.    digitalWrite(pinon, LOW);
105.    }
106.    else{
107.    digitalWrite(pinon, LOW);//(Από 107-110) Επαναφέρει το servo στην αρχική κατάσταση μετά από την αύξηση που έχει γίνει από  το κατέβασμα
108.    if (op == 280){
109.    myservo.attach(bservo);
110.    myservo.writeMicroseconds(var3);
111.    }
112.    if (op == 600){//(Από 112-117) Μας αποδεσμεύει το servo μετά από χρονικό 4sec
113.    myservo.detach();
114.    myservo2.detach();
115.    if (manual == 0){
116.    alagitaxititas();
117.    x=0;
```

Σε αυτή την υπορουτίνα καλούμε όλες τις υπόλοιπες υπορουτίνες του προγράμματος και κάνουμε εκτύπωση όλων των πληροφοριών που χρειαζόμαστε. Η μεταβλητή w είναι ένα χρονικό που απενεργοποιεί τον μικροελεγκτή αν δεν χρησιμοποιείται για 5 λεπτά και πριν από την απενεργοποίηση αποθηκεύει όλες τις απαραίτητες μεταβλητές. Επίσης μπορούμε να κάνουμε κατέβασμα των ταχυτήτων , όπως επίσης απενεργοποιεί τα servo μετά από 5 δευτερόλεπτα για οικονομία της μπαταρίας.

Παράρτημα Β

Τεχνικά Χαρακτηριστικά

ΟΝΟΜΑΣΙΑ	**ΤΑΣΗ ΛΕΙΤΟΥΡΓΙΑΣ (VOLT)**	**ΚΑΤΑΝΑΛΩΣΗ (AMPERE)**
Atmega 328	5v	80mA
Servo back	5v	250mA
Servo front	5v	160mA
LCD monitor	5v	45mA
3xButton	5v	0.0015mA
Οπτικός αισθητήρας	5v	25mA

Παράρτημα Γ

Τεχνικά Χαρακτηριστικά Αυτοματισμού

Τάση τροφοδοσίας : 10 voltmax , 6.5 voltmin

Ampere τροφοδοσίας: Standby 10mA , Normal 150mA , Max 560mA

Αυτοδυναμία κυκλώματος (6x2300mA Ni-MH):Standby 240hrs, Normal 12hrs