ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Υλοποίηση ρομποτικού οχήματος με Arduino

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τσαρίδης Γεώργιος (3987)

Επιβλέπων: Ιωάννης Καλόμοιρος, Καθηγητής

ΘΕΣΣΑΛΟΝΙΚΗ, ΙΟΥΝΙΟΣ 2019

<u>Υπεύθυνη Δήλωση</u>: *Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδος.*

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία αφορά τον σχεδιασμό και την ανάπτυξη ενός μικρού αυτόνομου οχήματος, το οποίο θα μπορεί να κινείται στο χώρο, να αναγνωρίζει εμπόδια και να παίρνει τις κατάλληλες αποφάσεις ώστε να τα αποφεύγει. Το όχημα κινείται χάρη σε έναν ηλεκτρικό κινητήρα, του οποίου η ταχύτητα και η φορά ελέγχονται μέσω κατάλληλου ολοκληρωμένου κυκλώματος (ελεγκτής ταχύτητας), που καθορίζει το εύρος της παρεχόμενης τάσης. Επίσης το όχημα διαθέτει ένα σερβοκινητήρα ο οποίος είναι υπεύθυνος για τη στροφή των μπροστινών τροχών, ώστε να μπορεί να αλλάζει πορεία όποτε χρειάζεται. Για την ανίχνευση των εμποδίων χρησιμοποιείται ένας αισθητήρας υπερήχων, που μπορεί να μετρά αποστάσεις από πιθανά εμπόδια.

Τα παραπάνω εξαρτήματα συνδέονται σε μία προγραμματιζόμενη πλακέτα Arduino Uno όπου και εκτελείται το πρόγραμμα ελέγχου του οχήματος. Ο αλγόριθμος είναι σχεδιασμένος ώστε να εκτελούνται οι διεργασίες που αναφέρθηκαν.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στον Τομέα Αρχιτεκτονικής Υπολογιστών και Βιομηχανικών Εφαρμογών, του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδος.

Η εργασία είχε ως σκοπό τη μετατροπή ενός τηλεκατευθυνόμενου οχήματος σε αυτόνομο, με τη δυνατότητα να κινείται στο χώρο, εντοπίζοντας και αποφεύγοντας πιθανά εμπόδια που βρίσκονται στην πορεία του. Το όχημα διαθέτει έναν κινητήρα συνεχούς ρεύματος, ο οποίος δέχεται την κατάλληλη τάση μέσω ενός ελεγκτή ταχύτητας (H-bridge), και έτσι καθορίζεται η ταχύτητα κίνησης του οχήματος και εάν αυτό θα κινηθεί προς τα εμπρός ή προς τα πίσω. Ένας σερβοκινητήρας είναι υπεύθυνος για τη γωνία στροφής των εμπρός τροχών, ενώ για τον εντοπισμό των εμποδίων χρησιμοποιήθηκε ένας υπερηχητικός αισθητήρας HC-SR04, που μπορεί να μετρά την απόσταση από αντικείμενα που βρίσκονται μπροστά από το όχημα.

Η οδήγηση των παραπάνω μερών του οχήματος γίνεται μέσω μίας προγραμματιζόμενης πλακέτας Arduino Uno Rev3, με την οποία συνδέονται κατάλληλα. Ο μικροελεγκτής εκτελεί το πρόγραμμα που έχει αναπτυχθεί και παράγει παλμούς για την οδήγησή τους. Ο προγραμματισμός του Arduino γίνεται μέσω ηλεκτρονικού υπολογιστή, με τη βοήθεια του περιβάλλοντος ανάπτυξης εφαρμογών Arduino IDE.

Η εργασία χωρίσθηκε σε πέντε κεφάλαια και το περιεχόμενο καθενός κεφαλαίου είναι το εξής:

Στο πρώτο κεφάλαιο γίνεται μία γενική αναφορά στα αυτόνομα οχήματα. Αναφέρεται τι εννοούμε με τον όρο αυτόνομο όχημα, ποια τα είδη και οι κατηγορίες τους και σε τι πλεονεκτούν σε σχέση με τα συμβατικά οχήματα.

Στο δεύτερο κεφάλαιο αναλύονται τα βασικά ηλεκτρονικά και μηχανικά μέρη του οχήματος. Γίνεται ανάλυση των χαρακτηριστικών και του τρόπου

λειτουργείας κάθε εξαρτήματος ξεχωριστά, καθώς και ο ρόλος που επιτελεί σε ένα κύκλωμα.

Στο επόμενο κεφάλαιο γίνεται η κυκλωματική ανάλυση του οχήματος και περιγράφεται η διαδικασία της μετατροπής του σε αυτόνομο. Παρουσιάζονται όλα τα στάδια αλλαγών και δοκιμών που χρειάστηκαν για τη μετατροπή του οχήματος, όπως επίσης και οι λόγοι που οδήγησαν στις αποφάσεις αυτές, μέχρι το όχημα να φτάσει στην τελική του μορφή.

Στο τέταρτο κεφάλαιο αναλύεται το πρόγραμμα που αναπτύχθηκε για τις ανάγκες της εργασίας. Γίνεται ανάλυση του αλγορίθμου που αναπτύχθηκε και παρουσιάζεται ο τρόπος λειτουργείας του.

Το τελευταίο κεφάλαιο της εργασίας περιλαμβάνει μερικά χρήσιμα συμπεράσματα που προέκυψαν μετά την ολοκλήρωσή της. Επίσης, αναφέρονται ορισμένοι λόγοι για τους οποίους περιορίστηκε η έρευνα και γίνονται μερικές προτάσεις για βελτίωση και περαιτέρω ανάπτυξη της εργασίας στο μέλλον.

Μετά το τέλος του πέμπτου κεφαλαίου, υπάρχουν τα παραρτήματα τα οποία περιλαμβάνουν τον κώδικα του βασικού προγράμματος που αναπτύχθηκε στα πλαίσια της εργασίας, καθώς και κάποιων βοηθητικών προγραμμάτων, που χρησιμοποιήθηκαν κατά τη διάρκεια της έρευνας και επίσης παρατίθενται οι πηγές πληροφοριών που χρησιμοποιήθηκαν.

Ολοκληρώνοντας θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της εργασίας κ. Ιωάννη Καλόμοιρο, Καθηγητή του Διεθνούς Πανεπιστημίου της Ελλάδος, για την υπόδειξη του θέματος και την πολύτιμη βοήθειά του κατά τη διάρκεια της εκπόνησης της πτυχιακής εργασίας.

Γεώργιος Τσαρίδης

Θεσσαλονίκη, Ιούνιος 2019

ΚΕΦΑΛΑΙΟ 1: Τα αυτόνομα οχήματα και οι εφαρμογές τους

Αυτή η πτυχιακή εργασία αφορά την ανάπτυξη ενός αυτόνομου ρομποτικού οχήματος. Έτσι, κρίθηκε απαραίτητο να γίνει μία γενική αναφορά στα είδη και τις εφαρμογές αυτού του είδους οχημάτων, καθώς και στα πλεονεκτήματά τους σε σχέση με τα συμβατικά οχήματα. Ουσιαστικά υπάρχει ένας ολόκληρος κλάδος της ρομποτικής που ασχολείται με τη μελέτη και την ανάπτυξη τέτοιου είδους οχημάτων.

Με τον όρο αυτόνομο όχημα ορίζεται ένα όχημα το οποίο λαμβάνει πληροφορίες από το περιβάλλον του, τις κατανοεί και κινείται σε αυτό, με ελάχιστη ή καθόλου ανθρώπινη παρεμβολή. Πρόκειται για επανδρωμένα ή μη οχήματα, προγραμματιζόμενα μέσω ηλεκτρονικού υπολογιστή. Συνήθως η λειτουργία τους εποπτεύεται, ώστε να επαληθεύεται πως εκτελούν σωστά τις εργασίες που τους έχουν ανατεθεί. Επίσης, σε ορισμένες περιπτώσεις δίνεται η δυνατότητα άμεσης παρεμβολής του ανθρώπου, την ώρα που λειτουργεί το όχημα, ώστε να υπάρχει καλύτερος έλεγχος.

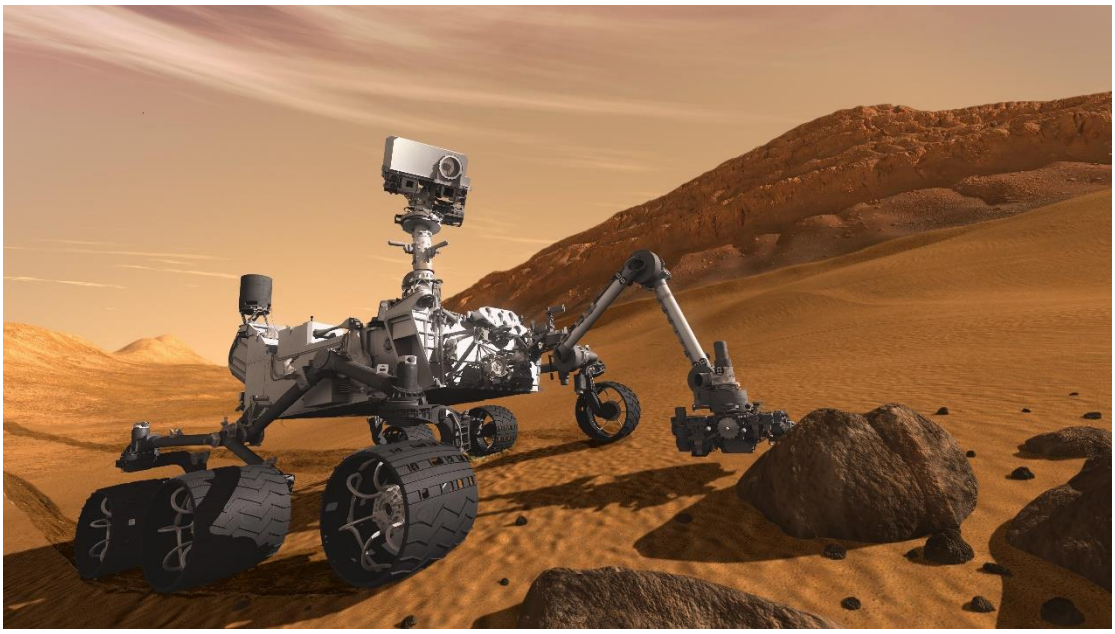
Υπάρχουν αυτόνομα οχήματα που μπορούν να κινηθούν στη στεριά, στον αέρα ή στο νερό. Κάθε όχημα, ανάλογα με το σκοπό και την πολυπλοκότητα της κατασκευής του, αλλά και το χώρο για τον οποίο είναι σχεδιασμένο να κινείται, διαθέτει κατάλληλους αισθητήρες ώστε να κατανοεί το περιβάλλον του. Οι αισθητήρες αυτοί μπορεί να είναι: ραντάρ, σόναρ, GPS, συστήματα οδομετρίας,κάμερες κ.τ.λ. Στη συνέχεια, ειδικά συστήματα ελέγχου αναλαμβάνουν να μεταφράσουν τις πληροφορίες που έλαβαν από τους αισθητήρες και να οδηγήσουν το όχημα προς τον επιθυμιτό προορισμό. Ανάλογα με την πολυπλοκότητα και τον σκοπό του οχήματος, μπορεί να αναγνωρίζει εμπόδια και να τα αποφεύγει, να βρίσκει τη βέλτιστη διαδρομή, να αναγνωρίζει την οδική σήμανση και να αντιδρά κατάλληλα κ.τ.λ., ενώ πράλληλα μπορεί να εκτελεί και άλλες εργασίες.

Ανάλογα με το σκοπό που επιτελούν, τα ρομποτικά οχήματα μπορούν να χωριστούν σε διάφορες κατηγορίες. Για παράδειγμα, υπάρχουν μικρά αυτόνομα οχήματα με απλό τρόπο λειτουργείας, όπως αυτό που αναπτύχθηκε στα πλάισια της εργασίας, που χρησιμοποιούνται για εκπαιδευτικούς σκοπούς ή ως χόμπι. Όμως υπάρχουν και πολύ πιο εξελιγμένα συστήματα, τα οποία χρησιμοποιούνται για ερευνητικούς σκοπούς, για να εκτελούν διάφορες εργασίες, για στρατιωτικούς σκοπούς, για τη βοήθεια ατόμων με αναπηρίες κ.τ.λ.

Το μεγαλύτερο πλεονέκτημα των αυτοκινούμενων οχημάτων σε σχέση με τα συμβατικά είναι ότι μπορούν να κινηθούν κάτω από συνθήκες που κάνουν επικίνδυνη ή και αδύνατη την ανθρώπινη παρουσία. Έτσι, έχουμε αυτόνομα οχήματα σχεδιασμένα για:

- Εξερεύνηση του εδάφους άλλων ουράνιων σωμάτων. Χαρακτηριστικό τέτοιο παράδειγμα είναι τα οχήματα τύπου Mars rover. Πρόκειται για τροχήλατα οχήματα, τα οποία αφού προσεδαφιστούν στο έδαφος του πλανήτη Άρη μπορούν να κινούνται σε αυτό και να μας παρέχουν χρήσιμες πληροφορίες χάρη σε εξελιγμένα συστήματα αισθητήρων που διαθέτουν.
- Συλλογή πληροφοριών για περιοχές της Γης στις οποίες είναι επικίνδυνη ή και αδύνατη η πρόσβαση του ανθρώπου. Υπάρχουν περιοχές που είναι δυσπρόσιτες εξαιτίας των γεωγραφικών και γεωλογικών τους χαρακτηριστικών. Για παράδειγμα για τη μελέτη και χαρτογράφηση των ωκεανών, σε βάθη μεγαλύτερα από αυτά που μπορούν να καταδυθούν οι άνθρωποι, έχουν κατασκευαστεί αυτόνομα υποβρύχια οχήματα (AUV). Τέτοιου είδους οχήματα έχουν βοηθήσει επίσης σε έρευνες που αφορούσαν αεροσκάφη που κατέληξαν στη θάλασσα, αλλά και για στρατιωτικούς σκοπούς. Υπάρχουν όμως και περιοχές που έχουν γίνει δυσπρόσιτες εξαιτίας της ανθρώπινης παρέμβασης. Έτσι, σε περιοχές με υψηλά ποσοστά ραδιενέργειας, σε περιοχές που ήταν ή είναι σε εμπόλεμη κατάσταση, σε υπερβολικά μολυσμένες περιοχές και γενικά σε περιοχές όπου οι συνθήκες που επικρατούν είναι επικίνδυνες για τους ανθρώπους, στέλνονται αυτόνομα οχήματα. Τα οχήματα αυτά κινούνται στο

έδαφος, στο νερό ή τον αέρα, ανάλογα με την περίπτωση και βοηθούν στη μελέτη των περιοχών αυτών, ώστε να καθοριστεί το είδος και η έκταση του προβλήματος.



Εικόνα 1.1 Το όχημα Curiosity rover στην επιφάνεια του πλανήτη Άρη

Υπάρχουν όμως και περιπτώσεις όπου τα ρομποτικά οχήματα χρησιμοποιούνται, ώστε να βοηθήσουν τον άνθρωπο να εκτελέσει κάποιες εργασίες ή να διευκολύνουν την καθημερινή του ζωή. Μερικά παραδείγματα τέτοιων εφαρμογών είναι τα εξής:

- Εξόρυξη πετρωμάτων και μετάλλων με τη χρήση ειδικών αυτόνομων οχημάτων.
- Εκτέλεση γεωργικών εργασιών, όπως πότισμα, ράντισμα, σπορά και θερισμός μεγάλων εκτάσεων. Οι παραπάνω εργασίες εκτελούνται κυρίως από οχήματα εδάφους, ενώ ορισμένες από αυτές μπορούν να εκτελεστούν και από ειδικά σχεδιασμένα drones.
- Συσκευές καθημερινής χρήσης, όπως αυτόνομες ηλεκτρικές σκούπες, ρομποτικές καρέκλες σχεδιασμένες για ανθρώπους με κινητικά προβλήματα κ.α.

- Χρήση αυτόνομων αυτοκινήτων και μέσων μαζικής μεταφοράς. Ήδη σε ορισμένες πόλεις ανά τον κόσμο υπάρχουν αυτοκινούμενα τρένα και λεωφορεία. Ο αριθμός των αυτοκινούμενων αυτοκινήτων παραμένει χαμηλός, για διάφορους λόγους, όπως ο περιορισμένος αριθμός αυτοκινήτων που παράγονται, το υψηλό κόστος, θέματα ασφάλειας και νομιμότητας. Αναμένεται πάντως να αυξηθεί τα επόμενα χρόνια, καθώς όλο και περισσότερες εταιρείες επενδύουν στον τομέα αυτό.

Εικόνα 1.2 Το Tesla Model X υποστηρίζει αυτόνομη οδήγηση χάρη στο σύστημα Autopilot

ΚΕΦΑΛΑΙΟ 2: Τα βασικά κομμάτια του οχήματος

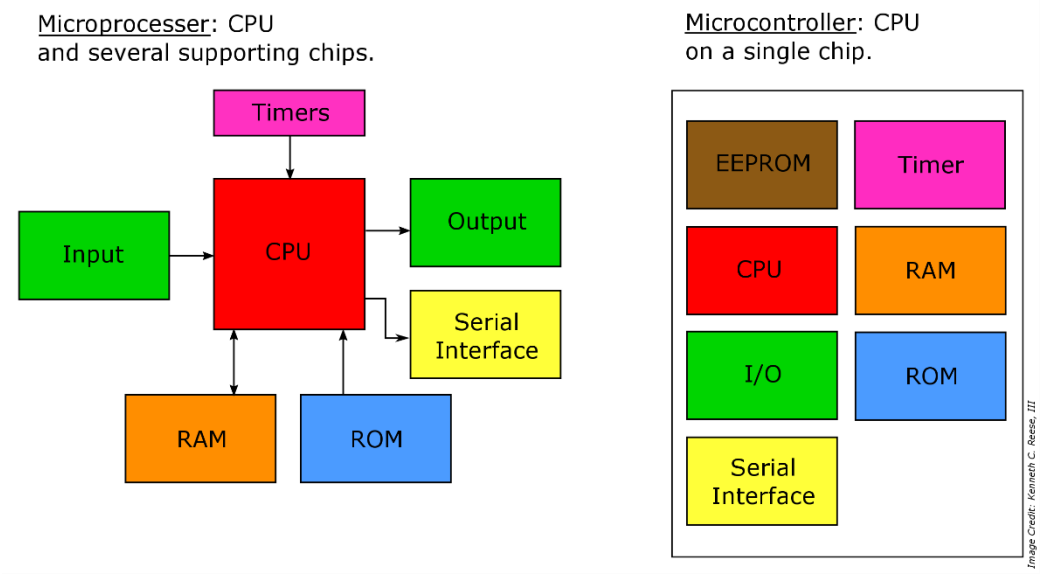
Στο κεφάλαιο αυτό περιγράφονται τα βασικά εξαρτήματα που είναι υπεύθυνα για τον έλεγχο του οχήματος, καθώς και ο ρόλος καθενός από αυτά.

2.1 Οι μικροελεγκτές και ο Arduino

Ένα από τα πιο σημαντικά κομμάτια του συγκεκριμένου οχήματος είναι μία προγραμματιζόμενη πλακέτα Arduino Uno Rev3, που στηρίζεται στον μικροελεγκτή AVR ATmega328P. Έτσι είναι αναγκαίο να γίνει μία γενική αναφορά στο τι είναι οι μικροελεγκτές και ποιες οι εφαρμογές τους. Στη συνέχεια θα γίνει μία ανάλυση της συγκεκριμένης πλακέτας και των δυνατοτήτων της, καθώς και του ρόλου της στη συγκεκριμένη εργασία.

2.1.1 Οι μικροελεγκτές και οι εφαρμογές τους

Με τον όρο μικροελεγκτές αναφερόμαστε σε υπολογιστικά συστήματα ειδικού σκοπού, τα οποία πέρα από την CPU (Central Processing Unit) που τους δίνει τη δυνατότητα εκτέλεσης εντολών, διαθέτουν επίσης μνήμη RAM (Random Access Memory), μνήμη προγράμματος, κυκλώματα εισόδου-εξόδου κλπ. Όλα τα παραπάνω είναι ενσωματωμένα σε ένα ολοκληρωμένο κύκλωμα (chip) και αυτή είναι και η βασική διαφορά τους από τους μικροεπεξεργαστές, οι οποίοι δεν μπορούν να λειτουργήσουν αυτόνομα, αλλά χρειάζονται ένα πλήθος υποσυστημάτων και περιφερειακών συσκευών, ώστε να μπορέσουν να αξιοποιηθούν από το χρήστη (π.χ. προσωπικός ηλεκτρονικός υπολογιστής).



Εικόνα 2.1:Μικροεπεξεργαστής (αριστερά) και μικροελεγκτής (δεξιά).

Ένα από τα πιο βασικά χαρακτηριστικά των μικροελεγκτών, είναι το γεγονός ότι μπορούν να αλληλοεπιδρούν με το φυσικό κόσμο με τρόπο που ένας τυπικός προσωπικός υπολογιστής δεν μπορεί. Για παράδειγμα, μπορούν να λαμβάνουν μετρήσεις από το περιβάλλον μέσω ειδικών αισθητήρων και να αποφασίζουν για τη συμπεριφορά άλλων εξαρτημάτων όπως για παράδειγμα μοτέρ.

Παρακάτω αναφέρονται μερικές πρακτικές εφαρμογές που βασίζουν τη λειτουργεία τους σε μικροελεγκτές:

- Προστασία και παρακολούθηση του περιβάλλοντος με δίκτυα αισθητήρων.
- Βιομηχανικός έλεγχος
- Ρομποτικά συστήματα
- Απομακρυσμένος έλεγχος και παρακολούθηση χώρου
- Συστήματα αυτόματων συναλλαγών (ATM, έκδοση εισιτηρίων κλπ)
- Συστήματα μετρήσεων
- Έξυπνο σπίτι
- Παιχνιδομηχανές
- Συστήματα επεξεργασίας ήχου

2.1.2 Η πλατφόρμα του Arduino

Το Arduino αναπτύχθηκε αρχικά το 2003 στην Ιβρέα της Ιταλίας, από μία ομάδα φοιτητών. Πρόκειται ουσιαστικά για μία επέκταση του Wiring project που είχαν αναπτύξει στα πλαίσια μιας εργασίας. Η εργασία αφορούσε την ανάπτυξη ενός απλού συστήματος βασισμένου σε μικροελεγκτή, το οποίο θα μπορεί να αλληλοεπιδρά με το περιβάλλον. Σκοπός ήταν να απευθύνεται τόσο σε αρχάριους όσο και σε πιο έμπειρους χρήστες, διατηρώντας παράλληλα το κόστος όσο πιο χαμηλά γίνεται.

Ουσιαστικά το Arduino είναι μία πλατφόρμα ανάπτυξης έργων, ανοιχτού κώδικα. Όπως προαναφέρθηκε, πρόκειται για μία πλακέτα μικροελεγκτή, η οποία συνοδεύεται από το αντίστοιχο περιβάλλον ανάπτυξης εφαρμογών (Arduino IDE), για τον προγραμματισμό του μικροελεγκτή μέσω ηλεκτρονικού υπολογιστή. Τα σχέδια της πλακέτας και το λογισμικό διατίθενται δωρεάν μέσω της επίσημης ιστοσελίδας του Arduino (https://www.arduino.cc).

Για τον προγραμματισμό του μικροελεγκτή χρησιμοποιείται η γλώσσα προγραμματισμού Wiring C. Η γλώσσα αυτή αποτελεί μία παραλλαγή της γλώσσας C++, ενώ έχουν προστεθεί διάφορες βιβλιοθήκες και έτσι υποστηρίζονται περισσότερες λειτουργείες και δυνατότητες, όπως για παράδειγμα η οδήγηση κινητήρων και αισθητήρων. Το περιβάλλον ανάπτυξης εφαρμογών, που προαναφέρθηκε, είναι δανεισμένο από τη γλώσσα Processing.

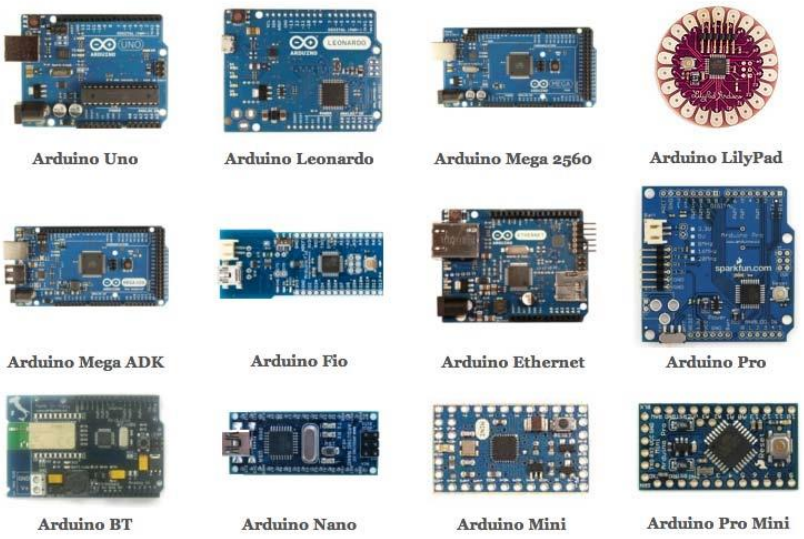
Υπάρχουν πολλές και διαφορετικές πλακέτες και εκδόσεις του Arduino, όπως για παράδειγμα τα: Arduino UNO, Arduino Mega, Arduino Nano, Arduino Robot κλπ. Οι παραπάνω πλακέτες διαφέρουν μεταξύ τους όσων αφορά τον τύπο μικροελεγκτή που διαθέτουν, το σύνολο των ακροδεκτών εισόδου-εξόδου, το μέγεθος κλπ. Με βάση την εφαρμογή που θέλουμε να αναπτύξουμε, επιλέγουμε και την κατάλληλη πλακέτα Arduino, ώστε να ικανοποιούνται καλύτερα οι απαιτήσεις της συγκεκριμένης εφαρμογής. Για παράδειγμα για την ανάπτυξη ενός ολοκληρωμένου robot, κατάλληλο θα ήταν το Arduino Robot, καθώς διαθέτει τροχούς κίνησης, δύο μικροελεγκτές σε

διαφορετικές πλακέτες για τον έλεγχο της κίνησης και την ανάγνωση των αισθητήρων, οθόνη LCD, μεγάφωνο κλπ, ενώ εάν μας ενδιέφερε περισσότερο το μέγεθος και η δυνατότητα μεταφοράς της εφαρμογής, τότε ίσως η κατάλληλη επιλογή να ήταν το Arduino Nano λόγω του μικρού του μεγέθους. Για τη συγκεκριμένη εργασία επιλέχθηκε η πλακέτα Arduino UNO, για την οποία ειδική αναφορά θα γίνει σε επόμενη παράγραφο.

Τέλος μερικά από τα κυριότερα πλεονεκτήματα του Arduino είναι τα εξής:

- Συμβατότητα: Ο προγραμματισμός του Arduino μπορεί να γίνει μέσω του Arduino IDE που είναι διαθέσιμο για περιβάλλοντα Windows, Linux και Macintosh.
- Απλότητα: Πρόκειται για ένα απλό σχετικά περιβάλλον, που είναι εύκολο στην εκμάθηση ακόμα και από αρχάριους χρήστες.
- Χαμηλό κόστος: Τόσο η πλακέτα όσο και εξαρτήματα που χρειάζονται για την ανάπτυξη απλών project, όπως καλώδια, απλοί αισθητήρες, led, οθόνες lcd κλπ έχουν μικρό κόστος αγοράς. Χαρακτηριστικό είναι πως υπάρχουν πακέτα που περιέχουν ότι είναι απαραίτητο για να ξεκινήσει κάποιος (starter kit) με τιμές κάτω των 50 ευρώ, ενώ το αντίστοιχο λογισμικό διατίθεται δωρεάν.
- Επεκτάσιμο λογισμικό ανοιχτού κώδικα: Τα προγράμματα του Arduino είναι ανοιχτού κώδικα, οπότε ο οποιοσδήποτε μπορεί να βρει πηγαίο κώδικα για ένα θέμα που τον αφορά και αφού τον μελετήσει να τον προσαρμόσει στις ανάγκες του έργου του και να τον χρησιμοποιήσει. Επίσης, υπάρχει ένα μεγάλο πλήθος έτοιμων βιβλιοθηκών κάτι που βοηθάει τους αρχάριους χρήστες να χρησιμοποιούν διάφορες δυνατότητες χωρίς να πρέπει να γνωρίζουν τις χαμηλού επιπέδου λεπτομέρειες του προγραμματισμού των διατάξεων που χρησιμοποιούν.

Για τους παραπάνω λόγους το Arduino επιλέγεται συχνά για την εκπαίδευση ανθρώπων με μικρή ή και καθόλου εμπειρία στην ανάπτυξη τέτοιου είδους έργων, όπως φοιτητών και μηχανικών και εκεί οφείλεται η μεγάλη επιτυχία του.

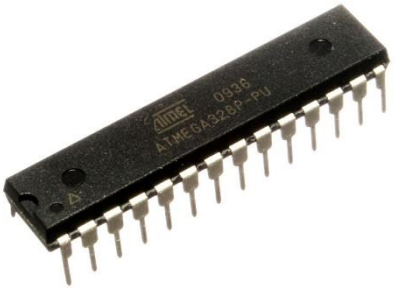


Εικόνα 2.2 Διάφορες πλακέτες Arduino

2.1.3 Τα χαρακτηριστικά και οι δυνατότητες του Arduino UNO Rev3 και ο ρόλος του στη συγκεκριμένη εργασία

Ο Arduino UNO είναι μία από τις πιο διαδεδομένες πλακέτες Arduino, καθώς μπορεί να υποστηρίξει ένα μεγάλο πλήθος διαφορετικών εφαρμογών και παράλληλα έχει χαμηλό κόστος αγοράς (γύρω στα 20 ευρώ).

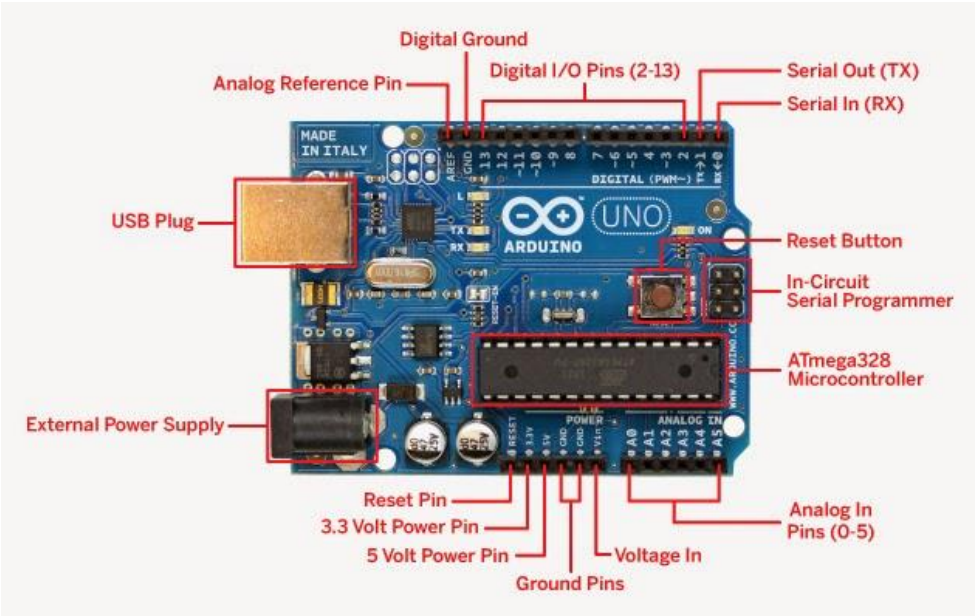
Η λειτουργεία της τρίτης έκδοσης της πλακέτας (Rev3) βασίζεται, όπως προαναφέρθηκε, στον μικροελεγκτή ATmega 328P. Πρόκειται για έναν μικροελεγκτή οκτώ bit, τύπου RISC (Reduced Instruction Set Computer) της εταιρείας AVR. Πάνω στο chip είναι διαθέσιμα 32 Kilobytes μνήμης προγράμματος, τύπου Flash, με κάθε θέση να έχει εύρος 16 bits. Από τα 32 Kbytes τα 0.5 Kbytes χρησιμοποιούνται από τον bootloader, ενώ τα υπόλοιπα είναι διαθέσιμα για την αποθήκευση των προγραμμάτων του χρήστη. Έτσι, τα προγράμματα παραμένουν αποθηκευμένα ακόμη και όταν δεν τροφοδοτείται η πλακέτα. Ο μικροελεγκτής διαθέτει επίσης 32 βασικούς καταχωρητές και μνήμη δεδομένων τύπου SRAM (Static Random Access Memory), μεγέθους 2Kbytes, ενώ χρονίζεται από κρυσταλλικό ταλαντωτή 16MHz.



Εικόνα 2.3 Το chip του μικροελεγκτή AVR ATmega 328P

Η πλακέτα διαθέτει δεκατέσσερεις ψηφιακούς ακροδέκτες εισόδου-εξόδου από τους οποίους οι έξι (3,5,6,9,10,11) μπορούν να παράγουν παλμούς τύπου PWM, κάτι που είναι χρήσιμο σε εφαρμογές οδήγησης φορτίων, όπως για παράδειγμα η οδήγηση μοτέρ. Επίσης, δύο από αυτούς (ακροδέκτες 0 και 1) αποτελούν τους ακροδέκτες RX και LX της σειριακής θύρας (UART). Ο μικροελεγκτής δέχεται και αναλογική είσοδο μέσω έξι ακροδεκτών (A0-A5).

Επειδή, όπως αναφέρθηκε, υπάρχει εγκατεστημένος bootloader, ο Arduino συνδέεται κατευθείαν στον υπολογιστή μέσω θύρας USB, χωρίς να χρειάζεται η χρήση εξειδικευμένων εργαλείων προγραμματισμού. Ο προγραμματισμός μέσω της σειριακής θύρας (UART) επιτυγχάνεται χάρη σε chip USB to Serial που διαθέτει η πλακέτα.



Εικόνα 2.4 Τα βασικά κομμάτια του Arduino UNO Rev3

Για την τροφοδοσία του μικροελεγκτή χρειάζεται τάση 5 Volt .Όταν ο Arduino είναι συνδεμένος με τον υπολογιστή τροφοδοτείται μέσω της θύρας USB, ενώ μπορεί να τροφοδοτηθεί και από εξωτερική πηγή τάσης (μπαταρία), είτε μέσω της θύρας τροφοδοσίας (συνήθως συνδέεται με μπαταρία 9 Volt), είτε μέσω του pin Voltage In. Σε περίπτωση τροφοδοσίας από τη θύρα τροφοδοσίας, θα πρέπει να παρέχεται τάση 7-12V.

Στη συγκεκριμένη εργασία, ο Arduino επιτελεί ένα πολύ σημαντικό ρόλο, καθώς εκεί εκτελείται το πρόγραμμα που είναι υπεύθυνο για τον έλεγχο του οχήματος και με βάση αυτό λαμβάνονται οι αποφάσεις για τη συμπεριφορά του.

2.2 Οι ηλεκτρικοί κινητήρες και οι ελεγκτές ταχύτητας

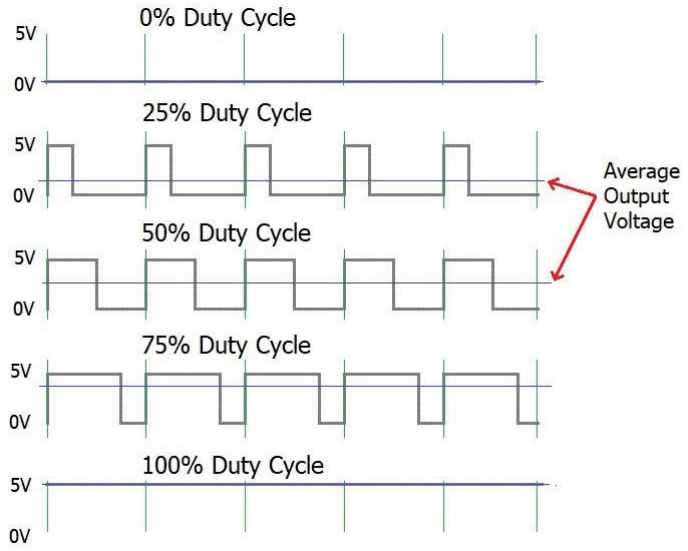
Για να μπορέσει να κινηθεί ένα όχημα είναι απαραίτητο να διαθέτει κάποιου είδους κινητήρες. Στα μικρά αυτόνομα οχήματα ή στα τηλεκατευθυνόμενα οχήματα χρησιμοποιούνται συνήθως δύο ειδών ηλεκτρικοί κινητήρες: οι κινητήρες συνεχούς ρεύματος και οι κινητήρες τύπου σέρβο. Οι πρώτοι χρησιμοποιούνται για να ορίσουν την ταχύτητα και την φορά (εμπρός ή πίσω) κίνησης του οχήματος, ενώ οι σέρβο για τη στροφή των μπροστινών τροχών. Ένας ελεγκτής ταχύτητας είναι ένα ηλεκτρονικό κύκλωμα υπεύθυνο για τον έλεγχο των παραπάνω κινητήρων.

2.2.1 Το σήμα ελέγχου PWM

Όπως αναφέρθηκε και παραπάνω, οι έξοδοι του Arduino είναι ψηφιακές. Αυτό σημαίνει πως μπορούν να βρίσκονται σε δύο μόνο καταστάσεις: HIGH (5V) και LOW (0V). Υπάρχουν όμως περιπτώσεις, όπως για παράδειγμα ο έλεγχος της ταχύτητας περιστροφής ενός μοτέρ, όπου δεν μας ενδιαφέρουν μόνο δύο καταστάσεις, αν για παράδειγμα το μοτέρ περιστρέφεται με συγκεκριμένη ταχύτητα (HIGH) ή δεν θα περιστρέφεται καθόλου (LOW), αλλά μας ενδιαφέρουν και οι τιμές μεταξύ των 0V και 5V, ώστε να μπορούμε να ελέγξουμε πόσο γρήγορα ή πόσο αργά θα περιστρέφεται. Τη λύση στο παραπάνω πρόβλημα δίνει η δυνατότητα παραγωγής παλμών τύπου PWM (Pulse Width Modulation) και έτσι είναι σκόπιμη μία αναφορά στον τρόπο λειτουργείας τους, προτού αναλυθούν οι τύποι κινητήρων.

Τα ψηφιακά σήματα έχουν τη μορφή περιοδικού τετραγωνικού παλμού. Κάθε περιοδικός παλμός ολοκληρώνει έναν πλήρες κύκλο του σήματος, μέσα σε μία περίοδο T. Το ποσοστό της περιόδου που αντιπροσωπεύει το θετικό μέτωπο του παλμού ονομάζεται κύκλος εργασίας (duty cycle). Μία συσκευή (πχ μοτέρ) αντιλαμβάνεται τόσο ποσοστό της μέγιστης τάσης, όσος και ο κύκλος εργασίας. Αν για παράδειγμα σε μία έξοδο του Arduino παράγονται

παλμοί με κύκλο εργασίας 25% και σε αυτή την έξοδο έχουμε συνδεμένο ένα μοτέρ συνεχούς ρεύματος τότε αυτό θα αντιλαμβάνεται τάση 1.25V (5V *0.25=1.25V), οπότε θα περιστρέφεται αργά.



Εικόνα 2.5 Παλμοί PWM με διαφορετικούς κύκλους εργασίας και οι αντίστοιχες τάσεις εξόδου

Έξι από τους ψηφιακούς ακροδέκτες του Arduino Uno (3,5,6,9,10,11) μπορούν να χρησιμοποιηθούν ως έξοδοι παλμών PWM, ενώ για ευκολία έχει τοποθετηθεί το σύμβολο ~ δίπλα από τον αντίστοιχο αριθμό.

2.2.2 Οι κινητήρες συνεχούς ρεύματος (dc motors)

Τα οχήματα όπως αυτό που χρησιμοποιήθηκε για την παρούσα εργασία, συνήθως φέρουν έναν ή δύο κινητήρες συνεχούς ρεύματος (dc motors) που είναι υπεύθυνοι για την περιστροφή των τροχών. Τις περισσότερες φορές, όπως και στην περίπτωσή μας, δεν διαθέτουν κάποιο σύστημα ψύξης (brushless dc motors). Ο συγκεκριμένος τύπος μοτέρ επιλέγεται αντί των μοτέρ που φέρουν κάποιο σύστημα ψύξης (brushed dc motors), λόγω των καλύτερων αποδόσεών τους, του μικρότερου βάρους τους

και της δυνατότητας ηλεκτρονικού ελέγχου μέσω ηλεκτρονικών ελεγκτών ταχύτητας.

Ανάλογα με την ένταση του ρεύματος που δέχεται το μοτέρ αλλάζει και η ταχύτητα περιστροφής του άξονα του και κατ' επέκταση του οχήματος, ενώ η πολικότητα ορίζει τη φορά περιστροφής και έτσι το όχημα κινείται προς τα εμπρός ή προς τα πίσω. Επειδή οι απαιτήσεις ρεύματος αυτών των μοτέρ είναι μεγάλες και ο Arduino δεν μπορεί να τις καλύψει, συνήθως τροφοδοτούνται μέσω κάποιου ελεγκτή ταχύτητας. Ειδική αναφορά στα κυκλώματα τέτοιου τύπου, γίνεται σε επόμενη παράγραφο.

Εικόνα 2.6 Το brushless motor GT 550 της εταιρείας HPI Racing

Ο Arduino μπορεί να οδηγήσει τέτοιου είδους κινητήρες με την παραγωγή παλμών PWM από τους αντίστοιχους ακροδέκτες.

2.2.3 Οι κινητήρες τύπου σέρβο (servomotors)

Εκτός όμως από τον έλεγχο της ταχύτητας περιστροφής και της φοράς των τροχών του οχήματος θα πρέπει με κάποιον τρόπο να ελέγχουμε και τη γωνία στροφής των μπροστινών τροχών, ώστε το όχημα να μπορεί να στρίβει κατάλληλα για να αποφεύγει τυχόν εμπόδια. Επειδή κάτι τέτοιο δεν μπορεί να γίνει με τη χρήση κινητήρων συνεχούς ρεύματος, το όχημα είναι εξοπλισμένο

με μία μικρή συσκευή που είναι σχεδιασμένη για να εκτελεί την παραπάνω εργασία και ονομάζεται σερβοκινητήρας (servo motor).

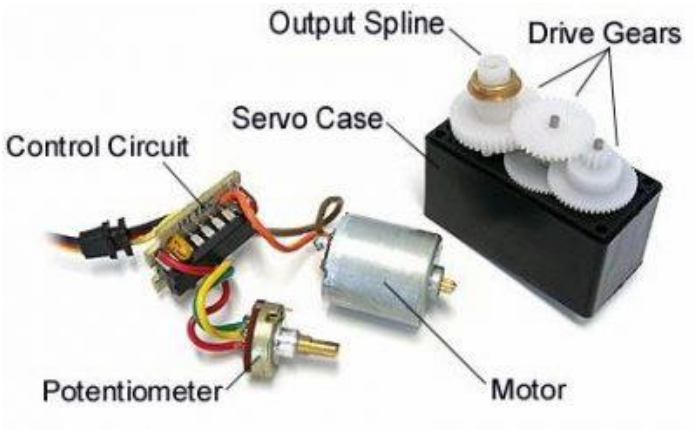
Ένας τυπικός σερβοκινητήρας αποτελείται από έναν μικρό κινητήρα συνεχούς ρεύματος, ένα σύστημα γραναζιών και ένα κύκλωμα ελέγχου. Η σύνδεση στο υπόλοιπο κύκλωμα γίνεται μέσω τριών καλωδίων.

Εικόνα 2.7 Το SG90, ένα από τα πιο διαδεδομένα σέρβο σε εφαρμογές Arduino

Το παραπάνω σύστημα μπορεί να στρέφει την κεφαλή του σέρβο (head), δηλαδή έναν εξωτερικό άξονα, μέσα σε ένα συγκεκριμένο εύρος τιμών, ανάλογα με την διάρκεια των παλμών PWM που δέχεται. Ένα ποτενσιόμετρο που είναι συνδεμένο στον εξωτερικό άξονα βοηθάει το κύκλωμα ελέγχου να παρακολουθεί την τρέχουσα γωνιακή θέση του άξονα. Σε περίπτωση που αυτή είναι διαφορετική από την επιθυμητή γωνία, ο κινητήρας συνεχούς ρεύματος ξεκινά να περιστρέφει τον άξονα προς τη σωστή κατεύθυνση μέχρι να ευθυγραμμιστεί με την επιθυμητή γωνία. Όταν ο άξονας βρίσκεται στην σωστή θέση, ο κινητήρας σταματά να περιστρέφεται. Το πόσο θα περιστραφεί ο άξονας, εξαρτάται από την απόσταση της τρέχουσας γωνίας από την επιθυμητή και έτσι η τάση που θα εφαρμοσθεί στο μοτέρ είναι ανάλογη αυτής της απόστασης. Αν η απόσταση είναι μεγάλη θα εφαρμοσθεί μεγαλύτερη τάση, απ’ ότι αν είναι μικρή.

Όπως προαναφέρθηκε, ο σερβοκινητήρας συνδέεται με το υπόλοιπο κύκλωμα μέσω τριών καλωδίων. Το ένα (κόκκινο) συνδέεται στην τάση

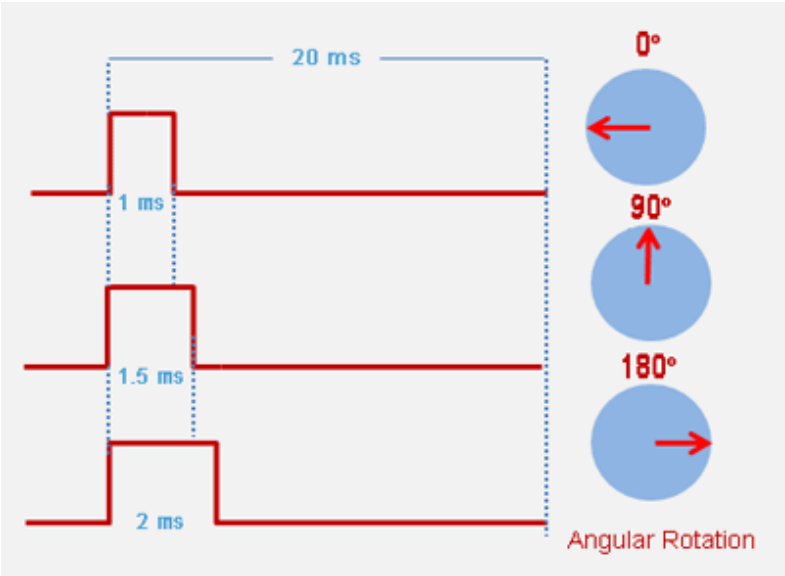
τροφοδοσίας (συνήθως 5V), το δεύτερο (μαύρο ή καφέ) συνδέεται στον αρνητικό πόλο τροφοδοσίας και το τρίτο (λευκό ή κίτρινο) είναι το καλώδιο ελέγχου των παλμών PWM και στην περίπτωσή μας συνδέεται στον Arduino.



Εικόνα 2.8 Τα κομμάτια από τα οποία αποτελείται ένας σερβοκινητήρας.

Ο σερβοκινητήρας δέχεται παλμούς PWM με περίοδο 20 millisecond. Ανάλογα με τη διάρκεια του παλμού καθορίζεται και η γωνιακή θέση της κεφαλής. Συνήθως η διάρκεια των παλμών είναι από 1ms έως και 2ms. Κάθε σέρβο μπορεί να περιστρέφει τον άξονα μέσα σε ένα συγκεκριμένο διάστημα τιμών, ανάλογα με τα κατασκευαστικά του χαρακτηριστικά. Για παράδειγμα το SG90 μπορεί να στρέφει τον άξονά του από 0 έως 180 μοίρες. Επομένως εάν ο παλμός που δέχεται είναι διάρκειας 1ms τότε ο άξονας θα βρίσκεται στην θέση με τη μικρότερη γωνία (0 μοίρες), ενώ όταν ο παλμός έχει τη μέγιστη διάρκεια (2ms) θα βρίσκεται στην θέση με τη μεγαλύτερη γωνία (180 μοίρες). Για παλμούς με εύρος 1.5ms, ο άξονας θα βρίσκεται στην ουδέτερη θέση (90 μοίρες).

Ο Arduino μπορεί να ελέγξει σερβοκινήτηρες μέσω των pin που μπορούν να παράγουν παλμούς PWM, ενώ ο μεταγλωττιστής υποστηρίζει εξειδικευμένη βιβλιοθήκη για την οδήγηση σέρβο (Servo.h).

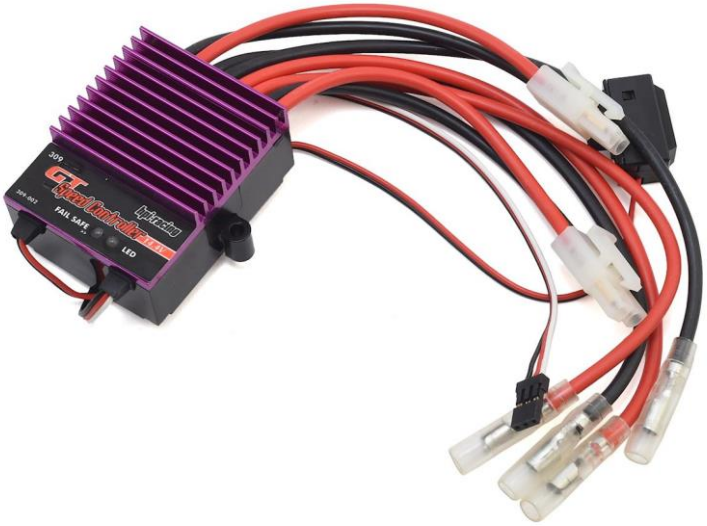


Εικόνα 2.9 Η συμπεριφορά του σερβοκινητήρα ανάλογα με τη διάρκεια των παλμών που δέχεται

2.2.4 Οι ηλεκτρονικοί ελεγκτές ταχύτητας (ESC)

Λόγω των υψηλών απαιτήσεων ρεύματος των μοτέρ δεν είναι δυνατή η απευθείας τροφοδοσία τους μέσω του Arduino. Για το λόγο αυτό είναι απαραίτητη η χρήση ενός κυκλώματος οδήγησης που ονομάζεται ηλεκτρονικός ελεγκτής ταχύτητας (ESC). Πρόκειται για ηλεκτρονικά κυκλώματα τα οποία μπορούν να τροφοδοτούν και να οδηγούν τους κινητήρες του οχήματος, με βάση τους παλμούς PWM που δέχονται από το Arduino ή το σύστημα τηλεκατεύθυνσης στα τηλεκατευθυνόμενα οχήματα. Τα παραπάνω σήματα μεταβάλλουν ένα δίκτυο από τρανζίστορ τύπου FET,και έτσι αλλάζει η ταχύτητα περιστροφής των κινητήρων. Λόγω της γρήγορης εναλλαγής των τρανζίστορ, το μοτέρ παράγει έναν ήχο υψηλής συχνότητας, ιδιαίτερα όταν περιστρέφεται αργά. Να σημειωθεί εδώ πως διαφορετικοί ελεγκτές χρησιμοποιούνται για τους κινητήρες που διαθέτουν σύστημα ψύξης (brushed dc motors) και διαφορετικοί για αυτούς που δεν διαθέτουν (brushless dc motors). Στην πρώτη περίπτωση ο έλεγχος του μοτέρ επιτυγχάνεται με τη μεταβολή της τάσης του οπλισμού του, ενώ στη δεύτερη, που είναι και αυτή που μας ενδιαφέρει, μεταβάλλεται η διάρκεια των παλμών

ρεύματος που δέχεται το μοτέρ. Αυτό πρακτικά γίνεται με την μετατροπή του συνεχούς ρεύματος της πηγής σε εναλλασσόμενο τριφασικό ρεύμα.

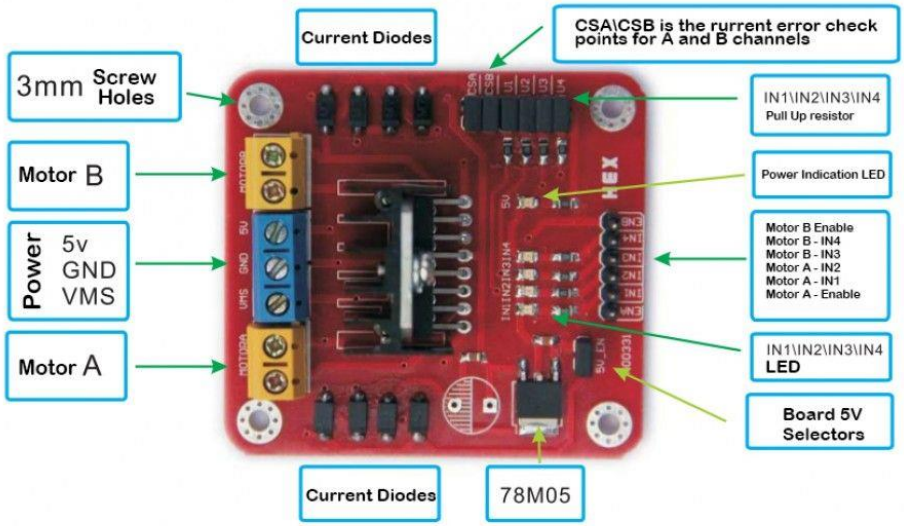


Εικόνα 2.10 Ο ελεγκτής ταχύτητας GT speed controller της εταιρείας HPI

Για την οδήγηση μοτέρ μέσω Arduino συχνά χρησιμοποιείται ο ελεγκτής ταχύτητας L298N ή αλλιώς H-bridge, λόγω του εύκολου προγραμματισμού του και του ακριβή ελέγχου που προσφέρει. Ο παραπάνω ελεγκτής βασίζεται στο ολοκληρωμένο L298 και μπορεί να ελέγξει μέχρι δύο κινητήρες συνεχούς ρεύματος ή ένα διπολικό βηματικό κινητήρα (stepper motor). Μπορεί να παρέχει στα μοτέρ τάση από 5 έως και 35 Volt, με μέγιστη ένταση ρεύματος έως και 2 Ampere. Διαθέτει επίσης έναν ακροδέκτη που έχει τη δυνατότητα να παρέχει σταθεροποιημένη τάση 5V και μπορεί να χρησιμοποιηθεί για την τροφοδοσία του Arduino ή άλλων συσκευών, όταν η τάση τροφοδοσίας της γέφυρας είναι έως και 12V.

Ο έλεγχος του κάθε μοτέρ γίνεται μέσω τριών ακροδεκτών οι οποίοι συνδέονται κατάλληλα με τον Arduino. Οι ακροδέκτες ENA και ENB, καθορίζουν την ταχύτητα περιστροφής των αντίστοιχων μοτέρ. Ο έλεγχος της ταχύτητας γίνεται με παλμούς PWM και για αυτό θα πρέπει τα αντίστοιχα pin του Arduino να υποστηρίζουν την παραγωγή τέτοιων παλμών. Οι ακροδέκτες IN1, IN2 για το πρώτο μοτέρ και IN3, IN4 για το δεύτερο, ορίζουν τη φορά

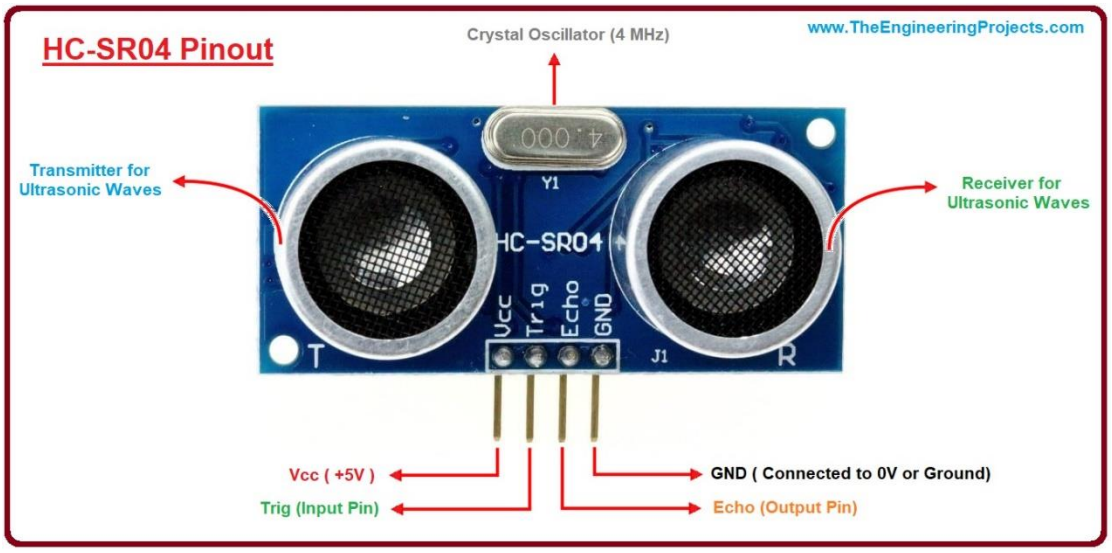
περιστροφής των κινητήρων και στην περίπτωσή μας ορίζουν εάν το όχημα θα κινείται προς τα εμπρός ή προς τα πίσω. Κάθε ακροδέκτης δέχεται από το Arduino μία ψηφιακή είσοδο, δηλαδή την τιμή HIGH ή LOW. Για παράδειγμα εάν στο πρώτο μοτέρ ο ακροδέκτης IN1 πάρει την τιμή HIGH και ο IN2 πάρει την τιμή LOW, τότε ο κινητήρας θα περιστραφεί προς τη μία κατεύθυνση (στην περίπτωσή μας το όχημα θα κινηθεί προς τα πίσω), ενώ αν οι ακροδέκτες πάρουν τις αντίστροφες τιμές, ο κινητήρας θα περιστραφεί προς την άλλη κατεύθυνση (το όχημα θα κινηθεί προς τα μπροστά). Σε περίπτωση που και οι δύο ακροδέκτες δεχτούν την τιμή LOW, ο κινητήρας σταματά να περιστρέφεται. Όλα τα παραπάνω ισχύουν και για το δεύτερο μοτέρ και τους ακροδέκτες IN3 και IN4 αντίστοιχα.



Εικόνα 2.11 Το κύκλωμα οδήγησης μοτέρ L298N (H-Bridge)

2.3 Ο αισθητήρας απόστασης HC-SR04

Κάθε αυτόνομο όχημα θα πρέπει με κάποιον τρόπο να μπορεί να ανιχνεύει πιθανά εμπόδια που βρίσκονται στην πορεία του και στη συνέχεια να λαμβάνει τις κατάλληλες αποφάσεις ώστε να τα αποφεύγει. Ο εντοπισμός των εμποδίων συνήθως γίνεται από έναν ή και περισσότερους αισθητήρες οι οποίοι με κάποιο τρόπο μπορούν να υπολογίζουν την απόσταση από πιθανά εμπόδια, ενώ πιο προηγμένα συστήματα χρησιμοποιούν πιο εξελιγμένες και ακριβής μεθόδους, όπως κάμερες, συστήματα χαρτογράφησης χώρου,GPS κλπ. Για τις ανάγκες αυτής της εργασίας χρησιμοποιήθηκε ο υπερηχητικός αισθητήρας απόστασης HC-SR04. Πρόκειται για έναν οικονομικό και απλό στη χρήση αισθητήρα, ενώ παράλληλα είναι αρκετά ακριβής για απλές εφαρμογές.



Εικόνα 2.12 Ο αισθητήρας HC-SR04 και τα βασικά κομμάτια του

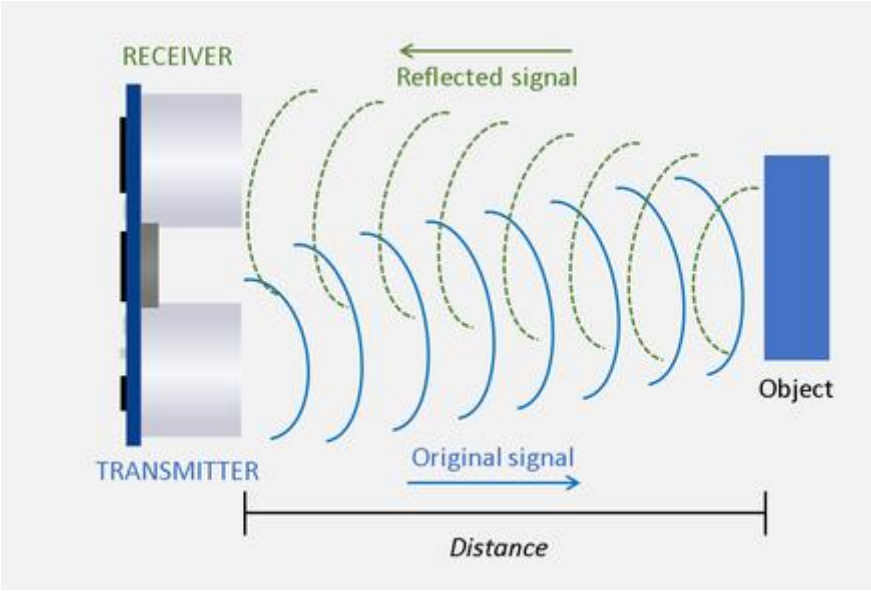
Ο αισθητήρας μπορεί να ανιχνεύει αντικείμενα σε απόσταση από δύο εκατοστά έως τέσσερα μέτρα και σε γωνία δεκαπέντε μοιρών. Η λειτουργεία του βασίζεται στην τεχνική “time of flight”. Σύμφωνα με αυτή την τεχνική, ο αισθητήρας εκπέμπει μία δέσμη ηλεκτρομαγνητικού ή άλλου κύματος και την αποστέλλει προς ένα αντικείμενο. Στη συνέχεια η δέσμη αυτή ανακλάται από

το αντικείμενο πίσω στον αισθητήρα, ο οποίος την λαμβάνει και υπολογίζει την απόσταση από το αντικείμενο, με βάση το χρόνο που χρειάστηκε να ταξιδέψει έως το αντικείμενο και πίσω στον αισθητήρα. Ο HC-SR04 παράγει σήματα με συχνότητα 4MHz με τη βοήθεια ενός κρυσταλλικού ταλαντωτή που διαθέτει. Τα σήματα εκπέμπονται από τον πομπό (συμβολίζεται με T από το transmitter), ενώ το ανακλώμενο σήμα επιστρέφει στον δέκτη (συμβολίζεται με R από το receiver).

Ο υπολογισμός της απόστασης γίνεται με βάση τον παρακάτω τύπο:

Απόσταση = (χρόνος αποστολής-λήψης σήματος * ταχύτητα ήχου) / 2

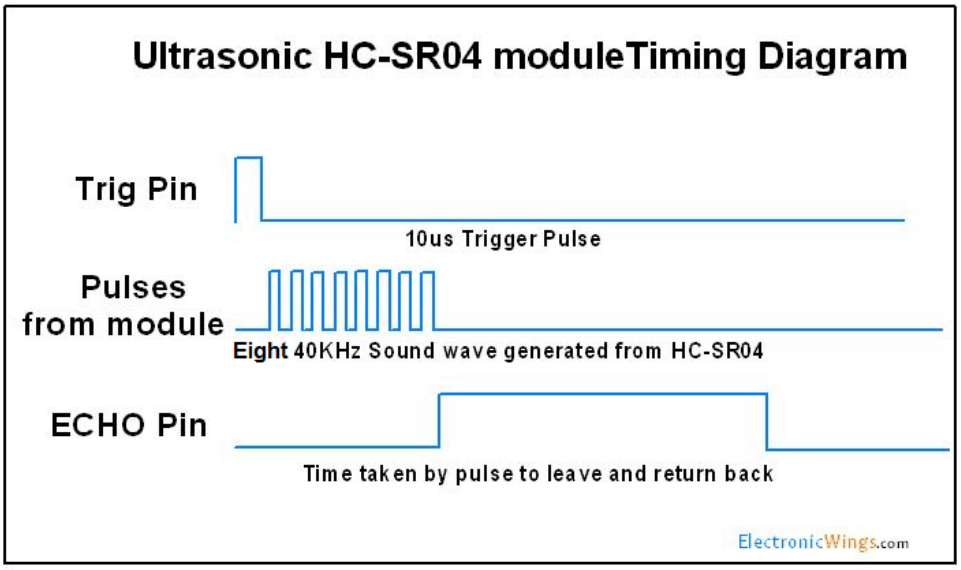
Είναι γνωστό πως η ταχύτητα του ήχου στον αέρα είναι 340 m/sec ή 0.034 cm/sec, οπότε αν γνωρίζουμε το χρόνο που ταξίδεψε το σήμα είναι εύκολο να υπολογιστεί οποιαδήποτε απόσταση. Αν για παράδειγμα ο αισθητήρας δεχθεί παλμό διάρκειας 500 μsec, τότε η απόστασή του από το αντικείμενο θα είναι: (500 μsec * 0.034 cm/sec)/2= 8.5 cm.



Εικόνα 2.13 Η διαδικασία παραγωγής και ανάκλασης του σήματος

Η επικοινωνία του αισθητήρα με τον Arduino γίνεται μέσω δύο ακροδεκτών, του Trig και του Echo, που συνδέονται σε δύο ψηφιακούς ακροδέκτες εισόδου/εξόδου του Arduino. Στον ακροδέκτη Trig παρέχουμε ένα θετικό παλμό (HIGH), μικρής διάρκειας (συνήθως 10 μsec) από τον Arduino, ώστε να παραχθεί το υπερηχητικό σήμα και να ξεκινήσει η μέτρηση του χρόνου επιστροφής. Ο αισθητήρας παράγει ένα θετικό παλμό (HIGH) στο pin Echo, από τη στιγμή αποστολής του υπερηχητικού παλμού έως και τη λήψη του ανακλώμενου παλμού, οπότε και αλλάζει η τιμή του pin σε LOW.

Οι ακροδέκτες VCC και GND χρησιμοποιούνται για την τροφοδοσία και για τη γείωση του αισθητήρα αντίστοιχα. Η τάση τροφοδοσίας του είναι 5V και μπορεί να τροφοδοτηθεί από τον Arduino μέσω του pin που παρέχει 5V, καθώς δεν έχει υψηλές απαιτήσεις ρεύματος (15 mA).



Εικόνα 2.14 Οι παλμοί που παράγει και δέχεται ο αισθητήρας HC-SR04

Στα αρνητικά της χρήσης του συγκεκριμένου αισθητήρα θα μπορούσαμε να συμπεριλάβουμε διάφορους εξωτερικούς παράγοντες που μπορεί να επηρεάσουν τη φυσιολογική του λειτουργεία. Για παράδειγμα η λειτουργεία του μπορεί να επηρεαστεί από την ύπαρξη θορύβων στο χώρο χρήσης του ή από την ύπαρξη άλλων υπερηχητικών αισθητήρων σε κοντινή απόσταση. Επίσης προβλήματα στις μετρήσεις μπορεί να προκληθούν από την ύπαρξη αντικειμένων με κεκλιμένη επιφάνεια ή ηχομονωτικών υλικών.

ΚΕΦΑΛΑΙΟ 3: Κυκλωματική ανάλυση του οχήματος και η διαδικασία της μετατροπής

Στο κεφάλαιο αυτό περιγράφεται η διαδικασία μετατροπής του οχήματος από τηλεκατευθυνόμενο σε αυτόνομο όχημα οδηγούμενο μέσω του Arduino.Επίσης, αναφέρονται τα κυκλώματα που αναπτύχθηκαν για τις ανάγκες της εργασίας. Γίνεται αναφορά τόσο στην αρχική μορφή του οχήματος, όσο και στα διάφορα στάδια αλλαγών που υποβλήθηκε μέχρι να φτάσει στην τελική του μορφή.

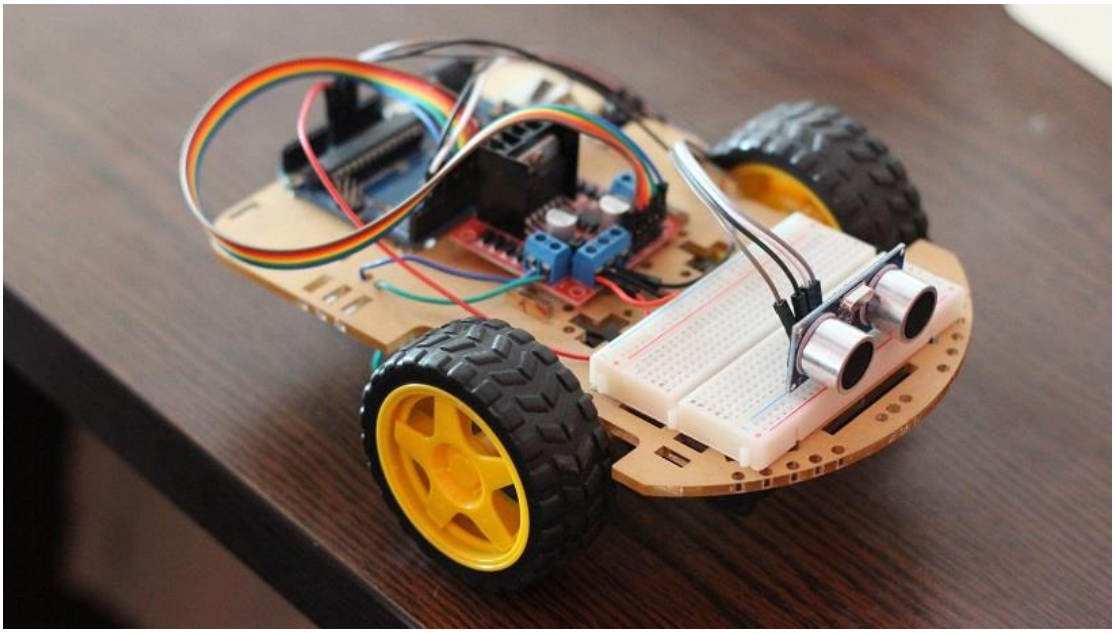
Εικόνα 3.1 Το όχημα πριν από τις αλλαγές

Σε κάθε στάδιο της μετατροπής αναλύονται τα εξαρτήματα που χρησιμοποιήθηκαν και οι κυκλωματικές αλλαγές που χρειάστηκαν, ώστε να υπάρξει όσο το δυνατόν καλύτερο αποτέλεσμα. Για το σχεδιασμό των κυκλωμάτων χρησιμοποιήθηκε το πρόγραμμα Fritzing.

3.1 Εξοικείωση με τα βασικά εξαρτήματα της εφαρμογής με τη βοήθεια οχήματος DIY

Για την καλύτερη κατανόηση του τρόπου λειτουργείας και προγραμματισμού των βασικών εξαρτημάτων ενός απλού αυτόνομου οχήματος, κρίθηκε απαραίτητη η εκπαίδευση με τη βοήθεια ενός μικρού DIY (Do It Yourself) οχήματος. Πρόκειται για ένα όχημα που λειτουργεί με πιο απλό τρόπο σε σχέση με αυτό που χρησιμοποιήθηκε στη συνέχεια και έτσι συχνά οχήματα σαν αυτό επιλέγονται για την εκπαίδευση αρχαρίων.

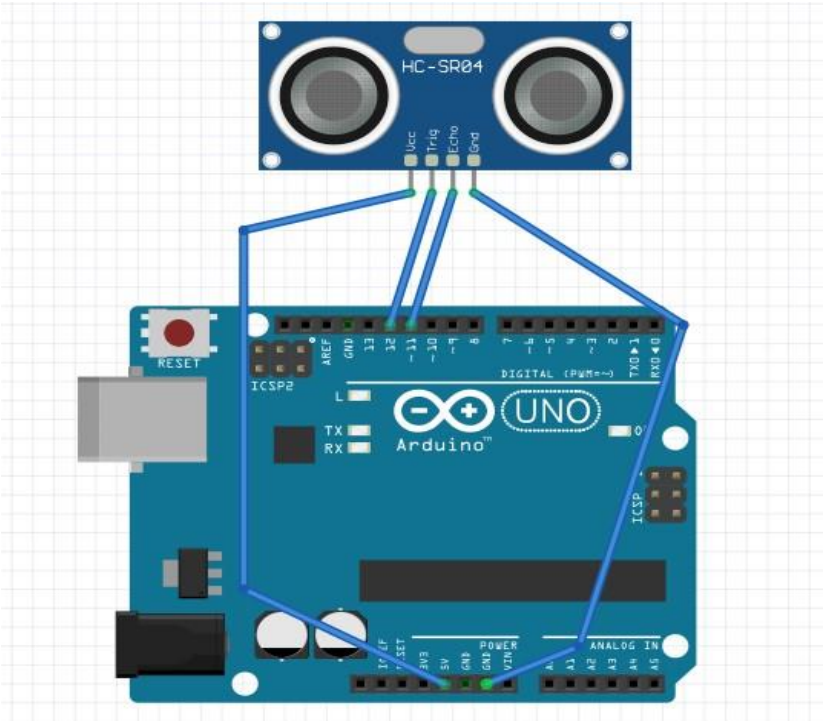
Το όχημα διαθέτει δύο τροχούς οι οποίοι περιστρέφονται ανάλογα με την ταχύτητα περιστροφής δύο κινητήρων συνεχούς ρεύματος (dc motors). Επίσης, ανάλογα με την ταχύτητα περιστροφής, αλλάζει και η γωνία των τροχών σε σχέση με τον άξονα περιστροφής, κάτι που δίνει τη δυνατότητα στο όχημα να στρίβει. Για την οδήγηση των μοτέρ χρησιμοποιείται ο ελεγκτής ταχύτητας L298N (H-Bridge), που είναι συνδεμένος με έναν μικροελεγκτή Arduino Uno. Το πρόγραμμα οδήγησης του οχήματος εκτελείται από τον Arduino και παράγονται οι κατάλληλοι παλμοί τύπου PWM. Οι κινητήρες δέχονται τους παλμούς μέσω της γέφυρας και περιστρέφονται αναλόγως.



Εικόνα 3.2 Όχημα DIY, παρόμοιο με αυτό που χρησιμοποιήθηκε

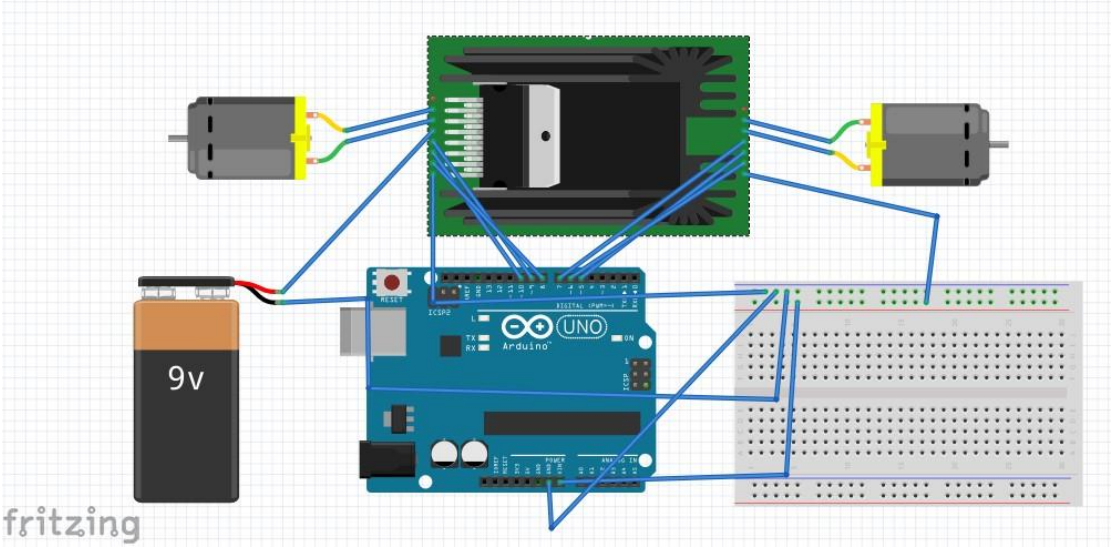
Υπάρχει επίσης ειδική βάση όπου μπορεί να στερεωθεί ένας υπερηχητικός αισθητήρας HC-SR04 κάτι που δίνει στο όχημα τη δυνατότητα να ανιχνεύει εμπόδια και να τα αποφεύγει με τη χρήση κατάλληλου αλγορίθμου.

Για να κατανοηθεί καλύτερα ο τρόπος λειτουργείας του παραπάνω αισθητήρα χρησιμοποιήθηκε το πρόγραμμα “Distance_Sensor” (Παράρτημα Β). Με το πρόγραμμα αυτό μπορεί να μετρηθεί η απόσταση από ένα αντικείμενο και να εμφανιστεί στην σειριακή κονσόλα η απόσταση αυτή σε εκατοστά, καθώς και η διάρκεια του παλμού που δέχτηκε ο αισθητήρας μετά από την ανάκλαση στο αντικείμενο. Ο ακροδέκτης Trig του αισθητήρα συνδέθηκε με τον ακροδέκτη 12 του Arduino και ακροδέκτης Echo με τον ακροδέκτη 11. Η τροφοδοσία του αισθητήρα μπορεί να γίνει είτε από κάποια εξωτερική πηγή τροφοδοσίας, όπως για παράδειγμα από την έξοδο 5V που παρέχει η H-Bridge, είτε απευθείας από την έξοδο 5V του Arduino, καθώς δεν έχει μεγάλες απαιτήσεις ρεύματος.



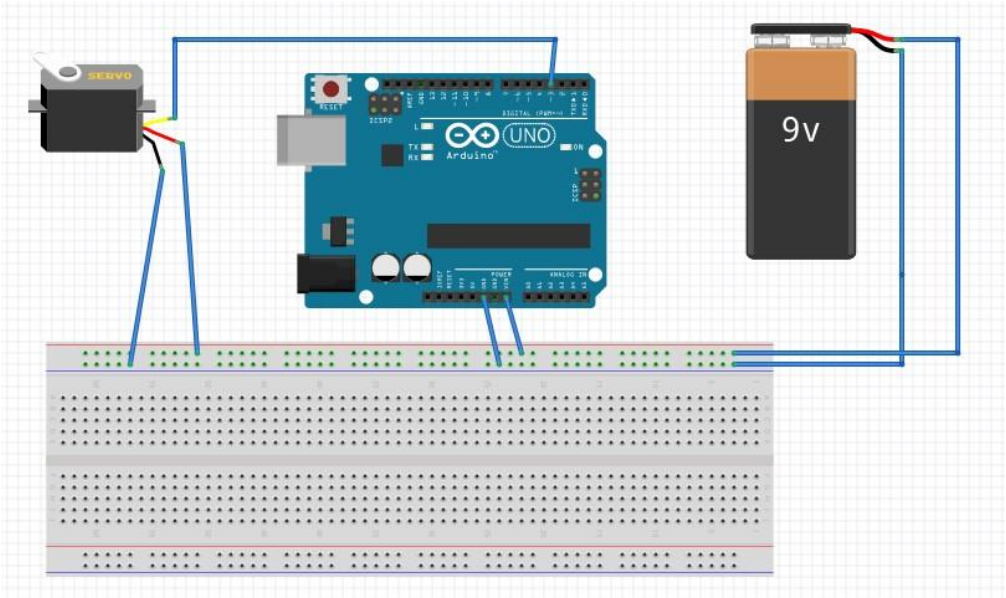
Εικόνα 3.3 Παράδειγμα σύνδεσης του Arduino με τον αισθητήρα HC-SR04

Επίσης, έγινε χρήση προγραμμάτων για την εξοικείωση με την οδήγηση κινητήρων συνεχούς ρεύματος μέσω της πλακέτας L298N. Συγκεκριμένα χρησιμοποιήθηκαν τα προγράμματα “dcmotor_control” και “dcmotor_control_2” (Παράρτημα Γ).Κατά την εκτέλεση του πρώτου προγράμματος το όχημα αρχικά κινείται προς τα εμπρός με σταθερή ταχύτητα και μετά από μια σύντομη στάση, αλλάζει η φορά περιστροφής των κινητήρων και κινείται προς τα πίσω και πάλι με σταθερή ταχύτητα. Όταν εκτελείται το δεύτερο πρόγραμμα το όχημα αρχικά κινείται προς τα εμπρός, ενώ η ταχύτητά του μεταβάλλεται από την ελάχιστη έως και τη μέγιστη τιμή. Κατόπιν, η ταχύτητα αρχίζει να μειώνεται μέχρι το όχημα να σταματήσει να κινείται και στη συνέχεια επαναλαμβάνεται η ίδια κίνηση με την διαφορά ότι έχει αλλάξει η φορά περιστροφής των μοτέρ. Τελικά, το όχημα ακινητοποιείται για μερικά δευτερόλεπτα, πριν επαναλάβει τις παραπάνω κινήσεις. Για τον έλεγχο της ταχύτητας περιστροφής των κινητήρων, οι ακροδέκτες ENA και ENB συνδέθηκαν αντίστοιχα στους ψηφιακούς ακροδέκτες 10 και 5 του Arduino. Για τον έλεγχο της φοράς περιστροφής του πρώτου μοτέρ (MOTOR A), τα pin IN1 και IN2 της H-bridge συνδέθηκαν στους ακροδέκτες 9 και 8 του Arduino, ενώ για το δεύτερο μοτέρ (MOTOR B), οι ακροδέκτες IN3 και IN4 συνδέθηκαν στους ακροδέκτες 6 και 7.Η τροφοδοσία της γέφυρας έγινε με μπαταρία 9V, ενώ ο Arduino τροφοδοτείται από την έξοδο 5V που παρέχει η H-bridge, μέσω του ακροδέκτη Vin.



Εικόνα 3.4 Τυπική σύνδεση δύο κινητήρων DC στη γέφυρα L298 που είναι συνδεμένη με τον Arduino

Τέλος θεωρήθηκε αναγκαία μία εισαγωγή στον τρόπο λειτουργείας και προγραμματισμού των κινητήρων τύπου σέρβο (servo motors). Για το σκοπό αυτό χρησιμοποιήθηκε ένας σερβοκινητήρας τύπου SG90, ο οποίος συνδέθηκε κατάλληλα με τον Arduino και έγιναν δοκιμές με το πρόγραμμα “servo_control” (Παράρτημα Δ). Κατά την εκτέλεση του προγράμματος, ο άξονας του σέρβο κινείται με σταθερή ταχύτητα λαμβάνοντας όλες τις διαθέσιμες γωνίες (από 0 έως 180 μοίρες), ενώ στη συνέχεια επαναλαμβάνεται η ίδια κίνηση, με αντίθετη φορά περιστροφής. Ο έλεγχος του σέρβο έγινε μέσω του ψηφιακού ακροδέκτη 3 του Arduino και η τροφοδοσία του έγινε από μπαταρία 9V, ώστε να μην επιβαρύνεται ο Arduino από τις υψηλές απαιτήσεις ρεύματος του μοτέρ.



Εικόνα 3.5 Παράδειγμα σύνδεσης του Arduino με ένα σερβοκινητήρα

Να σημειωθεί εδώ, πως τόσο τα κυκλώματα που αναφέρθηκαν όσο και αυτά που θα παρουσιαστούν στη συνέχεια, είναι ενδεικτικά και θα μπορούσαν να έχουν άλλη μορφή. Για παράδειγμα θα μπορούσαμε να συνδέσουμε τα διάφορα εξαρτήματα σε άλλους ακροδέκτες του Arduino ή να χρησιμοποιήσουμε διαφορετικές πηγές τροφοδοσίας. Επίσης, δεν γίνεται μεγάλη ανάλυση του κώδικα των προγραμμάτων που αναφέρθηκαν παραπάνω, καθώς αποτέλεσαν τη βάση πιο σύνθετων προγραμμάτων, τα οποία αναλύονται σε επόμενο κεφάλαιο.

3.2 Το τηλεκατευθυνόμενο όχημα και αρχική ιδέα της μετατροπής

Το όχημα που χρησιμοποιήθηκε για την παρούσα εργασία είναι το τηλεκατευθυνόμενο όχημα E-Savage Sport RTR της εταιρείας HPI Racing. Πρόκειται για ένα αγωνιστικό τετρακίνητο όχημα, τύπου monster truck, σχεδιασμένο να κινείται με υψηλές ταχύτητες, ανεξάρτητα από τον τύπο του εδάφους.

Πριν γίνει οποιαδήποτε αλλαγή, το όχημα είχε τα χαρακτηριστικά που παρουσιάζονται παρακάτω. Η περιστροφή των τροχών πραγματοποιούνταν χάρη σε δύο κινητήρες συνεχούς ρεύματος (HPI Racing GT550 14.4V), ο έλεγχος των οποίων γινόταν μέσω του ελεγκτή ταχύτητας GT Speed Controller. Για να μπορούν να στρίβουν οι μπροστινοί τροχοί, το όχημα διαθέτει ένα σερβοκινητήρα (HPI SF-10 Servo), ενώ για να καλυφθούν οι υψηλές ενεργειακές απαιτήσεις, κυρίως λόγω των δύο κινητήρων συνεχούς ρεύματος, ήταν αναγκαία η ύπαρξη δύο μπαταριών (PLAZMA 7.2V Stick Pack Rechargeable Battery) με χωρητικότητα 3300 mAh η καθεμία. Ο ελεγκτής ταχύτητας και ο σερβοκινητήρας συνδέονται στο σύστημα τηλεκατεύθυνσης, το οποίο δέχεται το σήμα που παράγεται από το ασύρματο τηλεχειριστήριο μέσω κεραίας. Το όχημα κινείται με βάση τις εντολές που δέχεται από το τηλεχειριστήριο.

Η αρχική ιδέα ήταν να αντικατασταθεί το σύστημα τηλεκατεύθυνσης από έναν μικροελεγκτή Arduino Uno και αφού γίνουν οι κατάλληλες αλλαγές στο κύκλωμα, η συμπεριφορά του οχήματος να ελέγχεται από το πρόγραμμα που εκτελείται από τον Arduino. Όμως στην πορεία, διαπιστώθηκε πως πρέπει να γίνουν και άλλες αλλαγές, ώστε να έχουμε το καλύτερο δυνατό αποτέλεσμα.

Πριν γίνει οποιαδήποτε αλλαγή, το όχημα δοκιμάστηκε ως τηλεκατευθυνόμενο, για να βεβαιωθούμε πως λειτουργεί κανονικά και να εντοπιστούν τυχόν προβλήματα. Το σημαντικότερο πρόβλημα που παρατηρήθηκε ήταν η εξάντληση των μπαταριών μέσα σε πολύ σύντομο χρονικό διάστημα και γι’ αυτό αποφασίστηκε η αντικατάστασή τους. Στη

συνέχεια έγινε μέτρηση της διάρκειας των παλμών που στέλνονται από το σύστημα τηλεκατεύθυνσης στον ελεγκτή ταχύτητας με παλμογράφο και παρατηρήθηκε η συμπεριφορά των μοτέρ συνεχούς ρεύματος. Παρατηρήθηκε ότι η περίοδος αποστολής των παλμών είναι 18ms και ότι ανάλογα με τη διάρκεια του ενεργού παλμού, τα μοτέρ συμπεριφέρονται ως εξής:

Πίνακας 3.1 Συμπεριφορά των dc κινητήρων, ανάλογα με τη διάρκεια των παλμών που δέχονται από το σύστημα τηλεκατεύθυνσης σύμφωνα με παλμογράφο.

Διάρκεια ενεργού παλμού	**Συμπεριφορά των dc κινητήρων**
Από 0ms έως 1ms	Περιστροφή προς τη μία κατεύθυνση (Το όχημα κινείται προς τα πίσω).
Από 1ms έως 1.5ms	Σημείο ισορροπίας (Το όχημα παραμένει ακίνητο).
Από 1.5ms έως 2ms	Αντίθετη φορά περιστροφής (Το όχημα κινείται προς τα εμπρός).

Τα παραπάνω όρια όπως θα δούμε και σε επόμενες δοκιμές που έγιναν δεν είναι απόλυτα, αλλά μεταβάλλονται εξαιτίας διάφορων παραγόντων όπως για παράδειγμα το επίπεδο φόρτισης των μπαταριών. Παρόλα αυτά οι μετρήσεις αυτές είναι αρκετά χρήσιμες, καθώς μας βοηθούν να καταλάβουμε πως συμπεριφέρονται τα μοτέρ όταν δέχονται διαφορετικής διάρκειας παλμούς, έστω και αν δεν ισχύουν πάντα ακριβώς οι ίδιες τιμές.

Σε επόμενο στάδιο έγινε η αντικατάσταση του συστήματος τηλεκατεύθυνσης από τον Arduino. Από το σύστημα τηλεκατεύθυνσης αφαιρέθηκαν τα καλώδια με τα οποία συνδεόταν με τον ελεγκτή ταχύτητας και με τον σερβοκινητήρα και βγήκε εκτός λειτουργείας. Έπειτα τα παραπάνω μέρη συνδέθηκαν με τον Arduino, όπως περιγράφεται στη συνέχεια.

Ο ελεγκτής ταχύτητας δέχεται παλμούς PWM μέσω του λευκού καλωδίου. Έτσι το εξής καλώδιο θα πρέπει να συνδεθεί σε έναν ψηφιακό

ακροδέκτη του Arduino που να έχει τη δυνατότητα παραγωγής παλμών τέτοιου είδους (επιλέχθηκε ο ακροδέκτης 5). Το κόκκινο καλώδιο μπορεί να παρέχει τροφοδοσία τάσης 5V σε άλλες συσκευές, ενώ το μαύρο είναι η γείωση. Μέσω αυτών των δύο καλωδίων τροφοδοτείται ο Arduino (μέσω του pin Vin και GND αντίστοιχα), καθώς και ο σερβοκινητήρας (συνδέοντας τα καλώδια ίδιου χρώματος). Το λευκό καλώδιο του σερβοκινητήρα είναι επίσης αυτό από το οποίο στέλνεται το σήμα ελέγχου. Συνδέεται και αυτό σε έναν ψηφιακό ακροδέκτη του Arduino με δυνατότητα παραγωγής παλμών τύπου PWM (επιλέχθηκε το pin 3).

Για να μπορέσει να κατασκευαστεί ένα πρόγραμμα οδήγησης των κινητήρων συνεχούς ρεύματος και του σερβοκινητήρα θα πρέπει πρώτα να γίνει η κατάλληλη βαθμονόμηση, ώστε να γνωρίζουμε τι τιμές εισόδου θα πρέπει να δέχονται τα μοτέρ σε κάθε περίπτωση, για να έχουν την επιθυμητή συμπεριφορά. Για το σκοπό αυτό χρησιμοποιήθηκε το πρόγραμμα “my_motor_test” (Παράρτημα Ε). Ο Arduino λαμβάνει τιμές από το χρήστη μέσω της σειριακής κονσόλας και αν οι τιμές αυτές είναι έγκυρες, τότε στέλνει το ανάλογο σήμα είτε στον ελεγκτή ταχύτητας και από εκεί στα dc motors, είτε στον κινητήρα τύπου σέρβο.

Για την οδήγηση του σερβοκινητήρα χρησιμοποιήθηκε η συνάρτηση της βιβλιοθήκης <Servo.h>, servo.write(n), όπου servo βάζουμε το όνομα που έχουμε επιλέξει για τον σερβοκινητήρα που θέλουμε να οδηγήσουμε και όπου n τον αριθμό των μοιρών που θέλουμε να στρίψει. Κατόπιν δοκιμών βρέθηκαν τα παρακάτω αποτελέσματα:

Πίνακας 3.2 Θέση των τροχών ανάλογα με τη γωνία στροφής του servo.

Γωνία σε μοίρες	Θέση μπροστά τροχών
62	Ακραία αριστερά θέση
95	Σε ευθεία σε σχέση με τους πίσω τροχούς
135	Ακραία δεξιά θέση

Στην περίπτωση των κινητήρων συνεχούς ρεύματος χρησιμοποιήθηκε και πάλι μία συνάρτηση της βιβλιοθήκης <Servo.h>, και συγκεκριμένα η servo.writeMicroseconds(), η οποία λαμβάνει ως παράμετρο τη διάρκεια του παλμού που θέλουμε να σταλεί στο μοτέρ σε microseconds (χιλιοστά του δευτερολέπτου). Παρόλο που δεν πρόκειται για τέτοιου είδους κινητήρες, χρησιμοποιήθηκε αυτή η συνάρτηση για λόγους ευκολίας. Μετά από δοκιμές παρατηρήθηκε η εξής συμπεριφορά:

Πίνακας 3.3 Συμπεριφορά των dc κινητήρων ανάλογα με τη διάρκεια των παλμών που δέχονται από τον Arduino.

Διάρκεια ενεργού παλμού	**Συμπεριφορά των dc κινητήρων**
Από 1550μs έως 1600μs	Περιστροφή προς τη μία κατεύθυνση (Το όχημα κινείται προς τα εμπρός).
Από 1500μs έως 1550μs	Σημείο ισορροπίας (Το όχημα παραμένει ακίνητο).
Από 1460μs έως 1420μs	Αντίθετη φορά περιστροφής (Το όχημα κινείται προς τα πίσω).

Παρατηρούμε πως οι τιμές αυτές διαφέρουν αρκετά από αυτές που μετρήθηκαν από τον παλμογράφο και αφορούσαν τους παλμούς που παρήγαγε το σύστημα τηλεκατεύθυνσης. Επίσης, οι παραπάνω τιμές είναι ενδεικτικές και συχνά μεταβαλλόταν εξαιτίας διάφορων παραγόντων όπως για παράδειγμα το επίπεδο φόρτισης των μπαταριών.

Για να μπορέσει να λυθεί το παραπάνω πρόβλημα αλλά και για να υπάρξει καλύτερος έλεγχος της ταχύτητας κίνησης του οχήματος δοκιμάστηκε να γίνει αντικατάσταση του ελεγκτή GT speed controller με μια πλακέτα L298 (H-bridge). Πράγματι, η ταχύτητα μπορούσε να ελεγχθεί με μεγαλύτερη ακρίβεια από πριν. Όμως, οι μεγάλες απαιτήσεις ρεύματος των δύο μοτέρ δημιουργούσαν προβλήματα στη γέφυρα, καθώς υπερθερμαινόταν και σταματούσε να λειτουργεί σωστά. Για το λόγο αυτό κρίθηκε πως αυτός ο τρόπος οδήγησης των μοτέρ δεν είναι ο κατάλληλος και έγινε προσπάθεια να βρεθεί κάποια άλλη λύση.

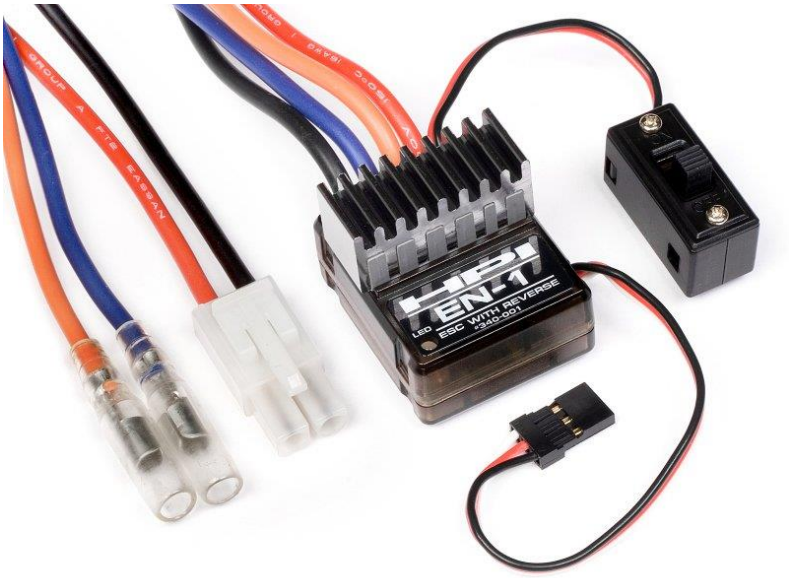
3.3 Η αντικατάσταση του συστήματος οδήγησης του οχήματος

Όπως αναφέρθηκε και σε προηγούμενη παράγραφο οι δύο κινητήρες συνεχούς ρεύματος που διέθετε το όχημα για να μπορεί να κινείται, είχαν μεγάλες απαιτήσεις ρεύματος με αποτέλεσμα να εξαντλούνται σε πολύ μικρό χρονικό διάστημα οι μπαταρίες. Επίσης, οι ταχύτητες που αναπτυσσόταν χάρη σε αυτά τα μοτέρ, ήταν υπερβολικά υψηλές για τις ανάγκες της συγκεκριμένης εφαρμογής. Έτσι αποφασίστηκε πως θα ήταν καλύτερα να γίνει αντικατάσταση των δύο μοτέρ από ένα, ώστε το όχημα να κινείται με μικρότερες ταχύτητες, να υπάρχει καλύτερος έλεγχος και μικρότερες ενεργειακές απαιτήσεις.

Εικόνα 3.6 Ο κινητήρας Crawler 55T της εταιρείας HPI Racing

Για την αντικατάσταση των δύο μοτέρ επιλέχθηκε ο κινητήρας Crawler 55T, της εταιρείας HPI Racing. Ο κινητήρας αυτός περιστρέφεται με λιγότερες στροφές από τους GT550 που έφερε αρχικά το όχημα, με αποτέλεσμα να μην αναπτύσσονται τόσο υψηλές ταχύτητες και η οδήγηση να είναι πιο ομαλή. Κατά συνέπεια απαιτείται μικρότερης έντασης ρεύμα για την περιστροφή αυτού του κινητήρα και έτσι εξαντλούνται πιο αργά οι μπαταρίες. Για να μπορέσει να προσαρμοστεί το καινούργιο μοτέρ, χρειάστηκε να γίνουν

αλλαγές στο σύστημα γραναζιών που κινούν τους άξονες μετάδοσης της κίνησης στους τροχούς. Επίσης, χρειάστηκε αλλαγή του ηλεκτρονικού ελεγκτή ταχύτητας. Ο ελεγκτής που χρησιμοποιήθηκε είναι ο EN-1 της εταιρείας HPI Racing, που είναι κατάλληλος για την οδήγηση του συγκεκριμένου κινητήρα και συνδέθηκε με τον Arduino όπως και ο προηγούμενος. Τέλος έγινε αντικατάσταση των μπαταριών καθώς δεν βρισκόταν σε καλή κατάσταση και αποφορτιζόταν γρήγορα. Για την τροφοδοσία του νέου κυκλώματος χρησιμοποιήθηκε μία μπαταρία λιθίου-πολυμερών (LiPo), που παρέχει τάση 7.4V και έχει χωρητικότητα 1000 mAh, ενώ σε επόμενο στάδιο αντικαταστάθηκε από μπαταρία ίδιου τύπου με μεγαλύτερη χωρητικότητα (5000 mAh).



Εικόνα 3.7 Ο ελεγκτής ταχύτητας HPI EN-1

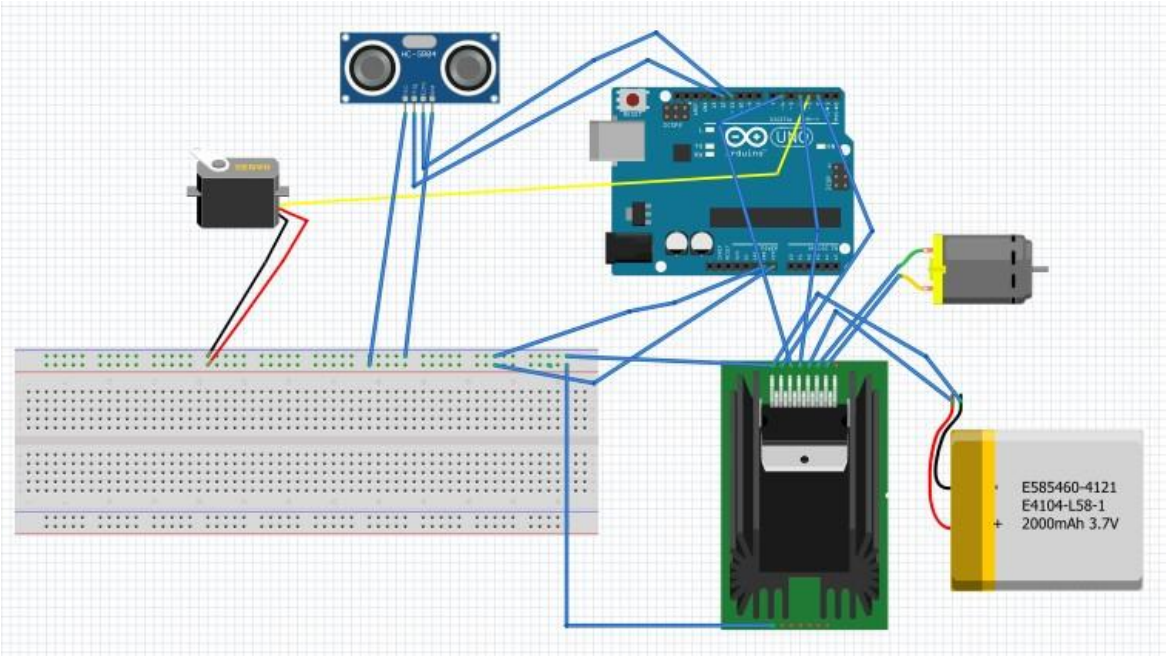
Αφού έγιναν δοκιμές με το πρόγραμμα “my_motor_test” (Παράρτημα Ε), διαπιστώθηκε πως ο κινητήρας έχει περίπου την ίδια συμπεριφορά, με τους δύο κινητήρες που έφερε αρχικά το όχημα (Βλέπε πίνακας 3.3). Στο κύκλωμα προστέθηκε ένας υπερηχητικός αισθητήρας HC-SR04 και ξεκίνησε η ανάπτυξη αλγορίθμου, ώστε να μπορεί να μετράτε η απόσταση από αντικείμενα και να λαμβάνονται οι κατάλληλες αποφάσεις για να αποφεύγονται. Οι ακροδέκτες Trig και Echo συνδέθηκαν αντίστοιχα στους ακροδέκτες 12 και 11 του Arduino, ενώ για την τροφοδοσία του αισθητήρα

χρησιμοποιήθηκε η έξοδος 5V που παρέχει ο ελεγκτής ταχύτητας. Ο αλγόριθμος που αναπτύχθηκε δούλεψε ικανοποιητικά ως ένα βαθμό, όμως παρουσιάστηκε και πάλι αστάθεια στη συμπεριφορά του κινητήρα συνεχούς ρεύματος, με αποτέλεσμα να μην μπορεί να ολοκληρωθεί η βελτιστοποίηση του αλγορίθμου.

3.4 Η τελική μορφή του οχήματος

Σε μία προσπάθεια να αντιμετωπιστούν τα προβλήματα που αναφέρθηκαν παραπάνω και να γίνει ομαλότερη η λειτουργεία του οχήματος, αντικαταστάθηκε εκ νέου ο ελεγκτής ταχύτητας από μία γέφυρα L298N. Το κύκλωμα που δημιουργήθηκε, έχει τη μορφή που περιγράφεται στη συνέχεια και είναι αυτό που χρησιμοποιήθηκε έως το τέλος της εργασίας. Ο Arduino συνδέεται με την H-bridge μέσω του pin 6 που συνδέεται με το pin ENA και των pin 4 και 2 που συνδέονται με τα pin IN1 και IN2 αντίστοιχα. Στον ψηφιακό ακροδέκτη 3 του Arduino συνδέεται το λευκό καλώδιο του σερβοκινητήρα, ενώ στους ακροδέκτες 12 και 11 συνδέονται αντίστοιχα τα pin Trig και Echo του αισθητήρα απόστασης. Η τροφοδοσία της γέφυρας γίνεται από τη μπαταρία μέσω των υποδοχών VMS (συνδέεται το κόκκινο καλώδιο) και GND (συνδέεται το μαύρο καλώδιο). Ο αισθητήρας απόστασης και ο σερβοκινητήρας τροφοδοτούνται από την έξοδο 5V που διαθέτει η πλακέτα L298N, ενώ όλα τα εξαρτήματα έχουν ως κοινή γη τον ακροδέκτη GND. Για την τροφοδοσία του Arduino χρησιμοποιήθηκε μία δεύτερη μπαταρία 5000 mAh.

Στην παρακάτω εικόνα παρουσιάζεται το κύκλωμα που περιγράφηκε. Η διαφορά σε σχέση με το πραγματικό κύκλωμα είναι πως χρησιμοποιήθηκαν μπαταρίες με διαφορετικά χαρακτηριστικά καθώς, δεν ήταν διαθέσιμες από το πρόγραμμα σχεδιασμού κυκλωμάτων. Επίσης, η τροφοδοσία του Arduino έγινε μέσω της θύρας τροφοδοσίας που διαθέτει, κάτι που δεν ήταν δυνατό να γίνει στο πρόγραμμα σχεδιασμού κυκλωμάτων. Για αυτό το λόγο έγινε μέσω του ακροδέκτη Vin.

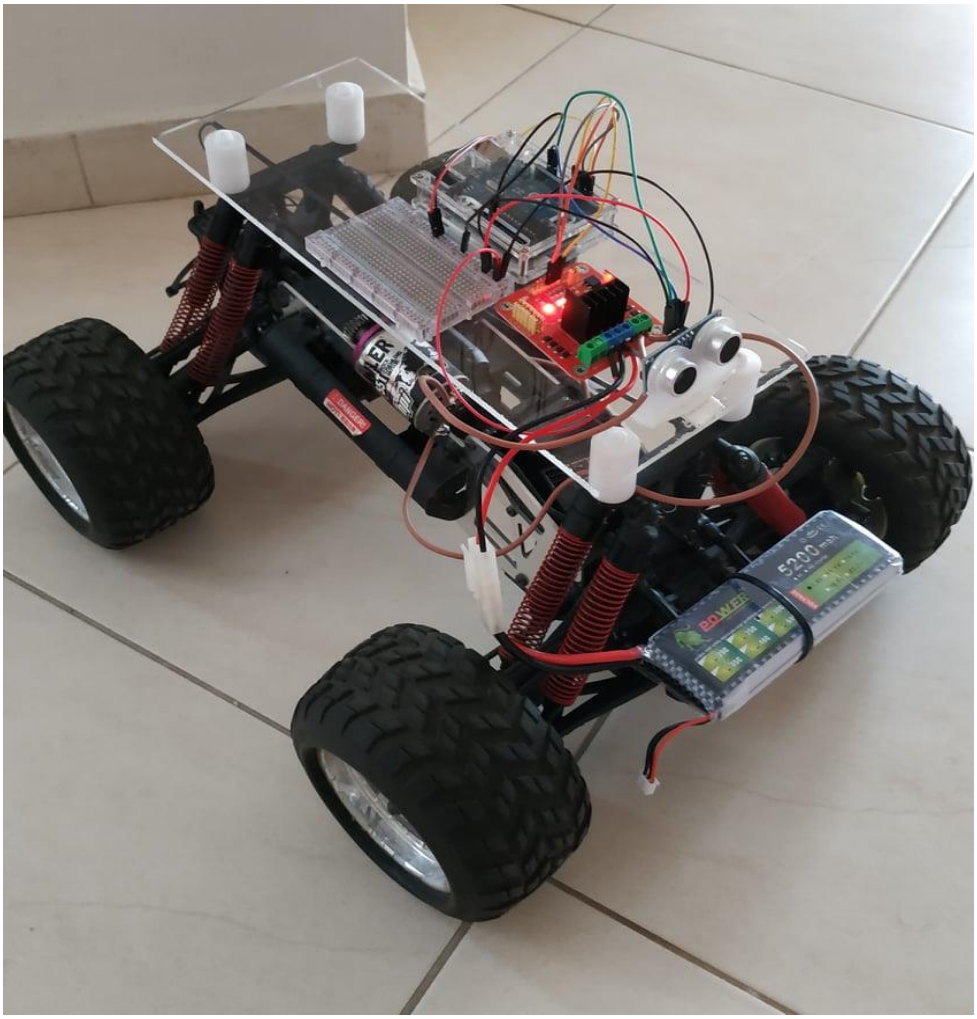


Εικόνα 3.8 Το τελικό κύκλωμα που χρησιμοποιήθηκε

Αφού κατασκευάστηκε το παραπάνω κύκλωμα έπρεπε να γίνει εκ νέου βαθμονόμηση του κινητήρα καθώς η οδήγηση με H-bridge γίνεται με διαφορετικό τρόπο απ' ότι με τους ελεγκτές ταχύτητας που χρησιμοποιήθηκαν αρχικά. Για το σκοπό αυτό χρησιμοποιήθηκε το πρόγραμμα "my_motor_test_h_bridge"(Παράρτημα Ε), το οποίο αποτελεί έκδοση του προγράμματος "my_motor_test" προσαρμοσμένη στον τρόπο οδήγησης μοτέρ με το συγκεκριμένο ηλεκτρονικό (L298). Να υπενθυμίσουμε εδώ πως η γέφυρα αυτή χρησιμοποιεί για την οδήγηση του κάθε μοτέρ τρία pin. Στην περίπτωσή μας επειδή διαθέτουμε ένα μοτέρ τα pin IN1και IN2 ορίζουν τη φορά περιστροφής και το pin ENA την ταχύτητα. Για να οριστεί η φορά περιστροφής, κάθε pin λαμβάνει την τιμή HIGH ή LOW μέσω της συνάρτησης digitalWrite() και ανάλογα με το συνδυασμό το όχημα κινείται προς τα εμπρός, προς τα πίσω ή δεν κινείται (Βλέπε παράγραφος 2.2.4). Το pin ENA δέχεται μέσω της συνάρτησης analogWrite() έναν αριθμό που αντιστοιχεί σε παλμό PWM συγκεκριμένης διάρκειας και το μοτέρ περιστρέφεται με την αντίστοιχη ταχύτητα. Η ελάχιστη τιμή είναι το μηδέν,

όπου το μοτέρ δεν περιστρέφεται και η μέγιστη είναι το 255, όπου το μοτέρ περιστρέφεται με τη μέγιστη δυνατή ταχύτητα.

Μετά από δοκιμές που πραγματοποιήθηκαν με αυτό το πρόγραμμα παρατηρήθηκε πως ο κινητήρας ξεκινά να περιστρέφεται για τιμές μεγαλύτερες του 100. Εξαιτίας του βάρους του οχήματος και για να έχουμε μία ικανοποιητική ταχύτητα, θα πρέπει να θέτουμε τιμές μεγαλύτερες του 180. Επίσης, παρατηρήθηκε ότι η συμπεριφορά του οχήματος είναι πιο σταθερή σε σχέση με τη συμπεριφορά που είχε με τους ελεγκτές ταχύτητας που χρησιμοποιήθηκαν σε προηγούμενα στάδια. Έτσι, αποφασίστηκε πως αυτός είναι ο πιο κατάλληλος τρόπος οδήγησης του οχήματος και ξεκίνησε η ανάπτυξη του προγράμματος “Room_Navigation_L298”(Παράρτημα Α) που αποτελεί και τον βασικό αλγόριθμο της εφαρμογής.



Εικόνα 3.9 Το όχημα μετά την ολοκλήρωση των αλλαγών

ΚΕΦΑΛΑΙΟ 4: Αλγοριθμική ανάλυση της εφαρμογής

Στο κεφάλαιο αυτό, γίνεται ανάλυση της δομής και της λειτουργείας του βασικού αλγορίθμου που αναπτύχθηκε για τις ανάγκες της εργασίας. Αρχικά γίνεται μία γενική επισκόπηση του τρόπου λειτουργείας του προγράμματος και στη συνέχεια αναλύονται οι συγκεκριμένες εντολές που χρησιμοποιήθηκαν.

4.1 Η δομή του βασικού αλγορίθμου

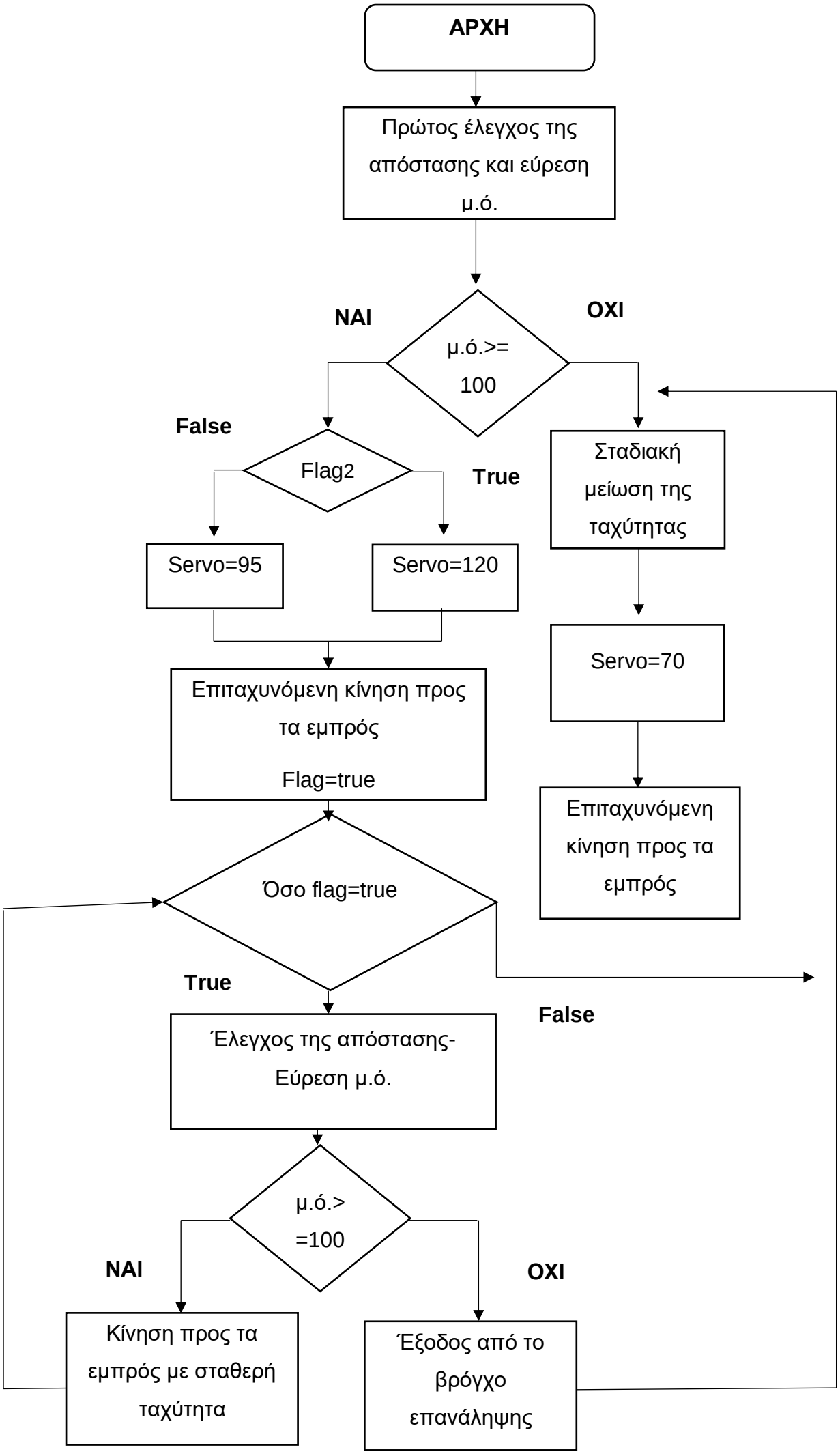
Η κεντρική ιδέα ήταν να αναπτυχθεί ένα πρόγραμμα για Arduino με το οποίο το όχημα θα μπορεί να μετρά αποστάσεις από πιθανά εμπόδια και να λαμβάνει τις κατάλληλες αποφάσεις, ώστε να τα αποφεύγει. Για να μπορέσουν να εκτελεστούν οι παραπάνω λειτουργείες, θα πρέπει ο αισθητήρας απόστασης να μετρά με ακρίβεια αποστάσεις και με βάση αυτές να οδηγούνται κατάλληλα ο σερβοκινητήρας και ο κινητήρας συνεχούς ρεύματος . Έτσι το όχημα θα μπορεί να κινείται σε ένα χώρο, μεταβάλλοντας την κατεύθυνση και ταχύτητα κίνησής του, ανάλογα με τα εμπόδια που εντοπίζει. Ο αλγόριθμος που αναλύεται στη συνέχεια αφορά το κύκλωμα που περιεγράφηκε παραπάνω (Εικόνα 3.5), όπου το ρόλο του ελεγκτή ταχύτητας έχει η γέφυρα οδήγησης μοτέρ που βασίζεται στο ηλεκτρονικό L298 (H-bridge).

Ο αλγόριθμος έχει τη βασική μορφή που περιγράφεται στη συνέχεια. Το όχημα αρχικά είναι ακίνητο και γίνεται ένας πρώτος έλεγχος από τον αισθητήρα για πιθανά εμπόδια. Αν κριθεί πως η απόσταση που μετράτε, είναι ασφαλής, το όχημα ξεκινά να κινείται σε ευθεία προς τα εμπρός. Η ταχύτητά του αυξάνεται σταδιακά μέχρι τη μέγιστη τιμή και στη συνέχεια ορίζεται μία σταθερή ταχύτητα. Το όχημα συνεχίζει να κινείται με τη σταθερή ταχύτητα που ορίστηκε, ενώ παράλληλα συνεχίζονται οι έλεγχοι της απόστασης από τα αντικείμενα που βρίσκονται μπροστά του. Σε περίπτωση που μετρηθεί απόσταση μικρότερη από την ελάχιστη ασφαλή απόσταση, το όχημα

επιβραδύνει και σταματά στιγμιαία. Στη συνέχεια αλλάζει η φορά περιστροφής των κινητήρων και το όχημα ξεκινά να εκτελεί επιταχυνόμενη κίνηση προς τα πίσω, υπό γωνία καθώς έχουν στραφεί οι μπροστινοί τροχοί. Η παραπάνω κίνηση συνεχίζεται, μέχρι να μετρηθεί απόσταση που θεωρείται πως είναι αρκετά μεγάλη, ώστε να κινηθεί προς τα εκεί το όχημα. Σε αυτή την περίπτωση οι μπροστά τροχοί στρίβουν κατάλληλα και έτσι το όχημα μπορεί να κινηθεί προς την επιθυμητή κατεύθυνση, με επιταχυνόμενη κίνηση προς τα εμπρός. Έπειτα, ξεκινά η επανάληψη της διαδικασίας που περιεγράφηκε, καθώς οι εντολές που είναι υπεύθυνες για τις παραπάνω ενέργειες περιλαμβάνονται στη συνάρτηση loop (). Η ανάλυση του κώδικα και η περιγραφή του τρόπου λειτουργείας των κυριότερων εντολών, γίνεται σε επόμενη παράγραφο.

4.2 Ανάλυση του κώδικα του βασικού προγράμματος οδήγησης του οχήματος

Στον παραπάνω αλγόριθμο στηρίχθηκε η ανάπτυξη ενός προγράμματος, το οποίο όταν εκτελείται από τον Arduino, οδηγεί το όχημα, όπως περιεγράφηκε στην προηγούμενη παράγραφο. Η ανάπτυξη του προγράμματος έγινε με χρήση του περιβάλλοντος προγραμματισμού Arduino IDE και ο μικροελεγκτής προγραμματίστηκε, αφού συνδέθηκε με τον υπολογιστή μέσω θύρας USB. Το πρόγραμμα ονομάστηκε “Room_Navigation_L298” και όπως υποδηλώνει και το όνομα του, σκοπό έχει την περιήγηση του οχήματος μέσα σε ένα δωμάτιο. Η συμβολική ονομασία χρησιμοποιήθηκε εκτός από το όνομα του προγράμματος και σε ονόματα μεταβλητών, σταθερών και συναρτήσεων, έτσι ώστε να είναι περισσότερο κατανοητός ο ρόλος τους. Η ανάλυση που γίνεται στη συνέχεια αφορά το συγκεκριμένο πρόγραμμα, ενώ αναφέρονται και ορισμένες αρχές του προγραμματισμού Arduino μέσω της Wiring C, που ισχύουν γενικότερα. Πριν από την έναρξη της ανάλυσης του προγράμματος, παρουσιάζεται ένα διάγραμμα ροής που περιγράφει τον τρόπο λειτουργείας της συνάρτησης loop().


ΑΡΧΗ
Πρώτος έλεγχος της απόστασης και εύρεση μ.ό.
μ.ό.>= 100
ΝΑΙ
ΟΧΙ
Flag2
False
True
Servo=95
Servo=120
Σταδιακή μείωση της ταχύτητας
Servo=70
Επιταχυνόμενη κίνηση προς τα εμπρός
Flag=true
Επιταχυνόμενη κίνηση προς τα εμπρός
Όσο flag=true
True
False
Έλεγχος της απόστασης- Εύρεση μ.ό.
μ.ό.> =100
ΝΑΙ
ΟΧΙ
Κίνηση προς τα εμπρός με σταθερή ταχύτητα
Έξοδος από το βρόγχο επανάληψης

Σχήμα 4.1: Το διάγραμμα ροής της συνάρτησης loop()

Αρχικά συμπεριλαμβάνεται στο πρόγραμμα η βιβλιοθήκη <Servo.h> και έτσι μπορούν να χρησιμοποιηθούν εντολές που αφορούν την οδήγηση κινητήρων τύπου σέρβο. Ορίζονται κάποιες καθολικές μεταβλητές που έχουν ισχύ σε όλη τη διάρκεια του προγράμματος. Ορίζονται και γίνεται αντιστοίχιση με τους ακροδέκτες που αφορούν, οι ακέραιες μεταβλητές echo, trigger, enA, in1 και in2. Οι δύο πρώτες αφορούν τον έλεγχο του αισθητήρα απόστασης και λαμβάνουν τις τιμές 11 και 12 αντίστοιχα. Οι τιμές αυτές αντιστοιχούν στους ακροδέκτες του Arduino με τους οποίους συνδέεται ο αισθητήρας απόστασης. Οι μεταβλητές enA,in1 και in2 αντιστοιχίζονται στους ακροδέκτες 6,4 και 2, οι οποίοι συνδέονται με τα αντίστοιχα pin της H-bridge και μέσω αυτών γίνεται ο έλεγχος του μοτέρ συνεχούς ρεύματος. Επίσης ορίζεται μία μεταβλητή τύπου Servo με όνομα steering_servo, όπου και θα αποθηκεύεται η γωνία στροφής του σερβοκινητήρα που είναι υπεύθυνος για τη στροφή των μπροστινών τροχών του οχήματος. Τέλος, ορίζεται μία ακέραια μεταβλητή με όνομα current_speed, στην οποία θα αποθηκεύεται η τρέχουσα ταχύτητα του οχήματος και η μεταβλητή flag2, τύπου Boolean, η οποία λειτουργεί ως σημαία και ο ρόλος της θα εξηγηθεί παρακάτω.

Ακολουθεί η εκτέλεση της συνάρτησης setup(). Η συνάρτηση αυτή αποτελεί βασικό κομμάτι, κάθε προγράμματος που αναπτύσσεται για Arduino. Χρησιμοποιείται για την αρχικοποίηση μεταβλητών, καταστάσεων των ακροδεκτών κ.τ.λ. και εκτελείται μία φορά, αφού ξεκινήσει η τροφοδοσία της πλακέτας ή γίνει επανεκκίνηση της. Κατά την εκτέλεση της συνάρτησης setup() εκτελούνται οι παρακάτω εντολές:

- **Serial.begin (9600);**

Γίνεται εκκίνηση της συριακής επικοινωνίας του Arduino με τον υπολογιστή με ρυθμό μετάδοσης 9600 bits/δευτερόλεπτο (τυπική ταχύτητα επικοινωνίας τέτοιου είδους). Η σειριακή επικοινωνία χρησιμοποιείται από την σειριακή κονσόλα που μας παρέχει το περιβάλλον Arduino IDE και μας βοηθάει να ελέγχουμε τις τιμές διάφορων παραμέτρων, μέσω της οθόνης του υπολογιστή.

- **pinMode(enA,OUTPUT);**
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT) ;
  pinMode(trigger,OUTPUT);
  pinMode(echo,INPUT);

Με την εντολή pinMode() ορίζουμε αν ένα pin θα συμπεριφέρεται ως είσοδος ή ως έξοδος. Έτσι οι ακροδέκτες που αφορούν την οδήγηση της H-bridge και ο ακροδέκτης που συνδέεται με τον ακροδέκτη Trig του αισθητήρα ορίζονται ως έξοδοι, καθώς στέλνουν παλμούς από τον Arduino στα ηλεκτρονικά που συνδέονται. Ο ακροδέκτης που συνδέεται με το pin Echo του αισθητήρα απόστασης, ορίζεται ως είσοδος, επειδή από εκεί γίνεται η μέτρηση από τον Arduino της διάρκειας του παλμού που στάλθηκε για να βρεθεί η απόσταση.

- **steering_servo.attach(3);**
  steering_servo.write(95);

Οι δύο παραπάνω εντολές συμπεριλαμβάνονται στη βιβλιοθήκη <Servo.h>. Με την πρώτη ορίζεται πως στο pin του Arduino με τον αριθμό 3 θα συνδεθεί ο σερβοκινητήρα με τη βοήθεια του οποίου στρίβει το όχημα. Στην επόμενη εντολή δίνεται στο pin αυτό η τιμή 95 κάτι που σημαίνει, όπως είδαμε σε προηγούμενο κεφάλαιο, πως οι μπροστά τροχοί είναι ευθυγραμμισμένοι με τους πίσω και κατά συνέπεια το όχημα θα κινηθεί σε ευθεία.

- **flag2= false;**

Αρχικοποιείται η δυαδική μεταβλητή flag2 με την τιμή false. Όταν η μεταβλητή έχει αυτή την τιμή, δεν εκτελείται ένα κομμάτι κώδικα, ενώ σε περίπτωση που λάβει την τιμή true εκτελείται.

Έπειτα εκτελείται η συνάρτηση loop() του προγράμματος. Όπως και η setup(), συμπεριλαμβάνεται και αυτή σε όλα τα προγράμματα που είναι γραμμένα για Arduino. Περιλαμβάνει τον βασικό κορμό των εντολών που εκτελούνται από τον μικροελεγκτή και όπως υποδηλώνει και το όνομα της εκτελείται συνεχώς για όσο τροφοδοτείται η πλακέτα. Επειδή, κάποιες εντολές

επαναλαμβάνονται πολλές φορές μέσα στη συνάρτηση loop(), θα αναφερθούν κάποια χαρακτηριστικά παραδείγματα, ώστε να γίνει κατανοητή η λειτουργεία τους.

- **long distance=0,sum=0,avg_distance=0;**
  int i,j,stopped=0;
  boolean flag=false;
  current_speed=0;

Στην αρχή της συνάρτησης ορίζονται και παίρνουν τις αρχικές τους τιμές ορισμένες μεταβλητές. Ο λόγος που ορίζονται εδώ και όχι στην αρχή του προγράμματος, είναι διότι θέλουμε σε κάθε εκτέλεση της loop(), να λαμβάνουν τις αρχικές τους τιμές.

- **for(i=0;i<5;i++){**

  distance=obstacle_distance();

  Serial.print('\n');

  Serial.print("i");

  Serial.print(":");

  Serial.print(i);

  Serial.print('\t');

  Serial.print("Value:");

  Serial.print (distance);

  sum=distance+sum;

  Serial.print('\t');

  Serial.print("Sum:");

  Serial.print (sum);

  delay(25);

  }

  avg_distance=sum/5;

Στη συνέχεια γίνεται ένας πρώτος έλεγχος της απόστασης. Για τον υπολογισμό της απόστασης χρησιμοποιήθηκε η συνάρτηση obstacle_distance (), η οποία περιγράφεται παρακάτω. Η συνάρτηση αυτή καλείται πέντε φορές, χάρη σε ένα βρόγχο επανάληψης for και στη συνέχεια υπολογίζεται ο μέσος όρος από τις τιμές που μετρήθηκαν. Αυτό συμβαίνει για να είναι πιο ακριβής η τιμή που θα βρεθεί και έτσι να είμαστε περισσότερο σίγουροι πως όντως υπάρχει εμπόδιο ή όχι και πως δεν πρόκειται για μια τυχαία μέτρηση του αισθητήρα.

- **if (avg_distance>=100){…}**

Αυτό το κομμάτι του προγράμματος, εκτελείται σε περίπτωση που ο μέσος όρος των αποστάσεων βρεθεί μεγαλύτερος ή ίσος με 100 cm (1 μέτρο). Τότε θεωρούμε πως η απόσταση είναι ασφαλής, ώστε να κινηθεί προς εκείνη την κατεύθυνση το όχημα.

- **if (flag2=false){**

```
steering_servo.write(95);
delay(20);
}
else{
delay(100);
steering_servo.write(95);
delay(50);
steering_servo.write(120);
delay(200);
flag2=false;
}
```

Γίνεται έλεγχος της τιμής που έχει λάβει η δυαδική μεταβλητή flag2. Σε περίπτωση που έχει πάρει την τιμή false, δεν έχει αλλάξει η πορεία του οχήματος και έτσι θέτουμε στο σέρβο που είναι υπεύθυνο για την στροφή των

μπροστινών τροχών την τιμή 95, με την οποία το όχημα κινείται σε ευθεία. Στην αντίθετη περίπτωση, όταν δηλαδή η τιμή της μεταβλητής είναι true, το όχημα έχει κινηθεί υπό γωνία προς τα πίσω στην προηγούμενη εκτέλεση της συνάρτησης loop. Έτσι, για να καταφέρει να αποφύγει το εμπόδιο θα πρέπει να κινηθεί προς τα εμπρός υπό την αντίθετη γωνία (120). Τέλος η μεταβλητή flag2 παίρνει την προκαθορισμένη της τιμή (false).

- **digitalWrite(in1,LOW);**

  digitalWrite(in2,HIGH);

  for(j=current_speed;j<=180;j=j+10){

  analogWrite(enA,j);

  Serial.println(j);

  delay(20);

  }

  delay(3000);

  steering_servo.write(95);

  flag=true;

Ορίζεται η φορά περιστροφής του κινητήρα μέσω των μεταβλητών in1 και in2, ώστε το όχημα να κινηθεί προς τα εμπρός. Η ταχύτητα περιστροφής του κινητήρα αυξάνεται σταδιακά μέχρι μία μέγιστη τιμή, μέσω ενός βρόγχου επανάληψης. Η εντολή analogWrite() γράφει μία αναλογική τιμή σε ένα pin που υποστηρίζει την παραγωγή παλμών τύπου PWM και από εκεί αποστέλλεται το σήμα στον αντίστοιχο ακροδέκτη της H-bridge (ENA). Αρχικοποιείται η τιμή της μεταβλητής flag, η χρησιμότητα της οποίας θα σχολιαστεί στη συνέχεια.

```
while(flag=true){
distance=0,sum=0,avg_distance=0;
for(i=0;i<5;i++){
distance=obstacle_distance();
Serial.print('\n');
Serial.print("i");
Serial.print(":");
Serial.print(i);
Serial.print('\t');
Serial.print("Value:");
Serial.print (distance);
sum=distance+sum;
Serial.print('\t');
Serial.print("Sum:");
Serial.print (sum);
delay(100);
}
avg_distance=sum/5;
Serial.print('\n');
Serial.print("Average value:" );
Serial.print(avg_distance);
Serial.print('\n');
if (avg_distance>=100){
analogWrite(enA,180);
```

```
current_speed=180;
}
else{
flag=false;
break;
}
}
}
```

Το παραπάνω κομμάτι κώδικα επαναλαμβάνεται όσο η μεταβλητή flag είναι ίση με true, δηλαδή για όσο η απόσταση από πιθανά εμπόδια παραμένει μεγαλύτερη του 100. Ο αισθητήρας συνεχίζει να μετρά αποστάσεις από τα αντικείμενα που βρίσκονται μπροστά στο όχημα και να βρίσκει το μέσο όρο τους, όπως και παραπάνω. Για όσο χρονικό διάστημα οι μέσοι όροι των αποστάσεων που μετρά ο αισθητήρας είναι μεγαλύτεροι από το επιτρεπόμενο όριο το όχημα θα κινείται ευθύγραμμα με σταθερή ταχύτητα, καθώς στέλνονται στον κινητήρα παλμοί PWM σταθερής διάρκειας (180). Σε περίπτωση που ο μέσος όρος των αποστάσεων είναι μικρότερος από το όριο, η μεταβλητή flag λαμβάνει την τιμή false και γίνεται έξοδος από τον βρόγχο while με την εντολή break.

```
❖ else{
  flag2=true;
   for(j=current_speed;j>=80;j=j-10){
  analogWrite(enA,j);
  Serial.println(j);
  delay(5);
  }
  current_speed=80;
  delay(100);
  steering_servo.write(70);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  for(j=current_speed;j<=180;j=j+10){
  analogWrite(enA,j);
  Serial.println(j);
  delay(15);
  }
  delay(1000);
  current_speed=180;
  }
  }
```

Σε περίπτωση που εντοπιστεί ένα αντικείμενο σε πολύ κοντινή απόσταση (<100 cm) εκτελείται το παραπάνω κομμάτι κώδικα. Αρχικά γίνεται true η τιμή της μεταβλητής flag2, ώστε στην επόμενη εκτέλεση της συνάρτησης loop() να εκτελεστεί το κομμάτι του προγράμματος, για το οποίο

λειτουργεί ως σημαία. Στη συνέχεια αρχίζει να μειώνεται σταδιακά η ταχύτητα περιστροφής του κινητήρα συνεχούς ρεύματος και το όχημα ακινητοποιείται στιγμιαία. Μέσω του σερβοκινητήρα αλλάζει η γωνία των μπροστινών τροχών (70), ενώ αλλάζει και η φορά περιστροφής του κινητήρα συνεχούς ρεύματος. Η ταχύτητα περιστροφής του κινητήρα αυξάνεται σταδιακά και έτσι το όχημα κινείται προς τα πίσω με σταδιακά αυξανόμενη ταχύτητα και υπό τη γωνία που αναφέρθηκε. Έπειτα, εκτελείται και πάλι η συνάρτηση loop() από την αρχή.

Για τον υπολογισμό της απόστασης από τα αντικείμενα που βρίσκονται μπροστά από το όχημα χρησιμοποιήθηκε η συνάρτηση obstacle_distance. Χάρη στη συνάρτηση αυτή ο αισθητήρας αρχικά στέλνει έναν υπερηχητικό παλμό, ο οποίος ανακλάται από το αντικείμενο που βρίσκεται μπροστά, πίσω στον αισθητήρα. Ο αισθητήρας λαμβάνει τον ανακλώμενο παλμό και η συνάρτηση υπολογίζει το συνολικό χρόνο που χρειάστηκε για την παραπάνω διαδικασία. Με βάση το χρόνο αυτό υπολογίζεται η απόσταση από το αντικείμενο.

Η συνάρτηση είναι τύπου long, όπως και οι δύο τοπικές μεταβλητές που δηλώνονται στην αρχή της συνάρτησης. Στη μεταβλητή duration θα αποθηκευτεί ο χρόνος που “ταξίδεψε” ο παλμός και στη μεταβλητή distance θα αποθηκευτεί η απόσταση.

- **digitalWrite(trigger,LOW);**
  delayMicroseconds(2);
  digitalWrite(trigger,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger,LOW);

Με το παραπάνω κομμάτι κώδικα γίνεται αποστολή ενός παλμού από τον αισθητήρα. Αρχικά θέτουμε στη μεταβλητή trigger, δηλαδή στον ακροδέκτη 12 του Arduino που είναι συνδεμένος με τον ακροδέκτη Trig του αισθητήρα, την τιμή LOW, ώστε στη συνέχεια να σταλεί ένας καθαρός παλμός HIGH. Μετά από μία μικρή καθυστέρηση διάρκειας 2μs, η μεταβλητή παίρνει την τιμή HIGH για 10μs και έπειτα μεταβαίνει και πάλι σε κατάσταση LOW. Ο χρόνος που χρειάζεται αυτός ο θετικός παλμός, από την αποστολή του έως

και την επιστροφή του στον αισθητήρα έπειτα από ανάκλαση, είναι ο χρόνος που μας ενδιαφέρει για τον υπολογισμό της απόστασης. Θέτουμε την καθυστέρηση στα 10μs, καθώς αυτή είναι η ελάχιστη διάρκεια παλμού για την οποία εκκινείτε ο αισθητήρας.

- **duration=pulseIn(echo,HIGH,30000);**

Με την παραπάνω γραμμή κώδικα αποθηκεύεται στη μεταβλητή duration ο χρόνος που χρειάστηκε για να διανύσει ο υπερηχητικός παλμός την απόσταση από τον αισθητήρα έως το αντικείμενο και πίσω. Η ανάγνωση του παλμού γίνεται μέσω της συνάρτησης pulseIn(). Η συνάρτηση αυτή δέχεται τρεις παραμέτρους. Στην πρώτη ορίζεται το pin του Arduino από το οποίο γίνεται η ανάγνωση του παλμού. Στην περίπτωσή μας είναι το pin 11 που συνδέεται με τον ακροδέκτη Echo του αισθητήρα, γι' αυτό και η αντίστοιχη μεταβλητή έχει το ίδιο όνομα. Με τη δεύτερη παράμετρο ορίζεται αν ο παλμός που θα αναγνωσθεί είναι θετικός (HIGH) ή αρνητικός (LOW) και τέθηκε η τιμή HIGH, καθώς είχε παραχθεί θετικός παλμός. Η τελευταία παράμετρος είναι προαιρετική, ορίζει το χρόνο αναμονής μέχρι την εκκίνηση του παλμού και ορίστηκε στα 30000 μs.

- **if(duration==0)**

```
{
pinMode(echo,OUTPUT);
delay(10);
digitalWrite(echo,LOW);
delay(10);
pinMode(echo,INPUT);
delay(10);
}
```

Το παραπάνω κομμάτι κώδικα είναι προαιρετικό και εκτελείται όταν για κάποιο λόγο ο ακροδέκτης Echo επιστρέφει συνεχώς την τιμή μηδέν. Το πρόβλημα αυτό, παρουσιάζεται συνήθως όταν η απόσταση από το εμπόδιο είναι πολύ μεγάλη (>4 μέτρα) ή βρίσκεται υπό γωνία, με αποτέλεσμα να μην πραγματοποιείται η ανάκλαση της δέσμης υπερήχων. Ο παλμός δεν επιστρέφει ποτέ στην αρχική του κατάσταση, κάτι που περιμένει η pulseIn().

Τη λύση σε αυτό το πρόβλημα τη δίνει η χρήση της τρίτης παραμέτρου της pulseln() που περιεγράφηκε παραπάνω. Όμως, η επιστρεφόμενη τιμή σε αυτή την περίπτωση είναι μηδέν κάτι που διορθώνεται με το παραπάνω κομμάτι κώδικα. Ο ακροδέκτης σε περίπτωση που χρόνος είναι ίσος με μηδέν, ορίζεται ως έξοδος, λαμβάνει την τιμή LOW και ορίζεται ξανά ως είσοδος.

- **distance=duration/29/2;**
  return distance;

Με την πρώτη εντολή υπολογίζεται με βάση το χρόνο και αποθηκεύεται στην μεταβλητή distance, η απόσταση από το αντικείμενο. Η διαίρεση με το 29 γίνεται γιατί η ταχύτητα του ήχου στον αέρα είναι 340 m/sec, δηλαδή 29 cm/ms. Στη συνέχεια διαιρούμαι με το 2, καθώς ο παλμός διανύει την ίδια απόσταση δύο φορές, μία όταν “ταξιδεύει” προς το αντικείμενο και μία όταν επιστρέφει στον αισθητήρα μετά την ανάκλαση. Η δεύτερη εντολή επιστρέφει σε μία μεταβλητή ίδιου τύπου, του βασικού κορμού του προγράμματος, το περιεχόμενο της μεταβλητής distance.

Σε πολλά σημεία του προγράμματος καλείται η συνάρτηση Serial.println() για να μπορούμε να επιβεβαιώνουμε πως το όχημα λειτουργεί με τον επιθυμητό τρόπο, με βάση τα αποτελέσματα που εμφανίζονται στη σειριακή κονσόλα, στην οθόνη του υπολογιστή. Επίσης, χρησιμοποιείται συχνά η συνάρτηση delay(), μέσω της οποίας προσθέτουμε τις απαραίτητες καθυστερήσεις που χρειάζονται για τη σωστή λειτουργεία του προγράμματος.

ΚΕΦΑΛΑΙΟ 5: Τελικά συμπεράσματα

Μετά την ολοκλήρωση της εργασίας, καταλήξαμε σε ορισμένα χρήσιμα συμπεράσματα που θα μπορούσαν να βοηθήσουν στην ανάπτυξη παρόμοιων εφαρμογών. Όμως, υπήρξαν και προβλήματα που οφείλονται σε διάφορους παράγοντες και περιόρισαν την περαιτέρω ανάπτυξη της εργασίας. Επίσης, στο μέλλον υπάρχει η δυνατότητα επέκτασης των λειτουργειών του οχήματος και βελτίωσης του συστήματος που υπάρχει ήδη.

5.1 Μερικά χρήσιμα συμπεράσματα

Συμπερασματικά θα μπορούσαμε να πούμε πως η ανάπτυξη ενός αυτόνομου οχήματος δεν είναι μία απλή διαδικασία. Για να αναπτυχθεί ένα τέτοιο όχημα, θα πρέπει να γίνει πρώτα η κατάλληλη έρευνα, ώστε να κατανοηθεί ο τρόπος λειτουργείας των επιμέρους εξαρτημάτων, ο τρόπος προγραμματισμού, η βασική δομή του αλγορίθμου κ.τ.λ.. Επίσης, για να υπάρξει το καλύτερο δυνατό αποτέλεσμα θα πρέπει να γίνουν πολλές δοκιμές και ενδεχομένως αρκετές αλλαγές τόσο στον κώδικα που θα χρησιμοποιηθεί, όσο και στα διάφορα μέρη και στο κύκλωμα του οχήματος. Τελικά θα πρέπει να κατασκευαστεί ένα σύστημα με σταθερή συμπεριφορά, το οποίο θα εκτελεί τις εργασίες για τις οποίες είναι σχεδιασμένο, με το βέλτιστο δυνατό τρόπο.

5.2 Περιορισμοί της έρευνας

Ορισμένοι από τους λόγους που περιορίστηκε η έρευνα αναφέρονται στη συνέχεια:

- Οι μετρήσεις από τον αισθητήρα απόστασης που χρησιμοποιήθηκε (HC-SR04), σε ορισμένες περιπτώσεις δεν έχουν την απαραίτητη ακρίβεια με αποτέλεσμα να λαμβάνονται λάθος αποφάσεις. Το πρόβλημα αυτό παρουσιάζεται συνήθως όταν το αντικείμενο βρίσκεται

σε μεγάλη απόσταση ή υπό γωνία. Παρατηρείται επίσης σε περιπτώσεις που η επιφάνεια του αντικειμένου δεν είναι αρκετά λεία. Οι παραπάνω παράγοντες επηρεάζουν την ανάκλαση της δέσμης υπερήχων και οδηγούν σε λάθος μετρήσεις. Για να περιοριστεί αυτό το πρόβλημα λαμβάνονται πολλές τιμές και βρίσκεται ο μέσος όρος τους. Για περαιτέρω περιορισμό του προβλήματος θα μπορούσε να βελτιστοποιηθεί περισσότερο ο αλγόριθμος ή να χρησιμοποιηθεί κάποιος αισθητήρας που παρέχει μεγαλύτερη σταθερότητα και ακρίβεια.

- Ο ελεγκτής ταχύτητας και το μοτέρ συνεχούς ρεύματος, χάρη στο οποίο κινείται το όχημα, δεν έχουν πάντα σταθερή συμπεριφορά. Υπάρχει δηλαδή η πιθανότητα, για παλμούς PWM συγκεκριμένης διάρκειας, να μην παρατηρείται η αναμενόμενη συμπεριφορά. Ορισμένοι παράγοντες που είναι πιθανό να επηρεάσουν τη συμπεριφορά τους είναι το επίπεδο φόρτισης της μπαταρίας, ο χρόνος χρήσης, εξωτερικοί παράγοντες όπως η θερμοκρασία και η υγρασία κ.τ.λ. Για το λόγο αυτό καλό είναι πριν από τη χρήση του οχήματος να γίνεται πάντα έλεγχος της κατάστασης των μπαταριών και να βεβαιώνεται πως το μοτέρ τροφοδοτείται με επαρκές ρεύμα. Μία καλύτερη λύση θα ήταν η προσθήκη ενός συστήματος μέτρησης των στροφών του κινητήρα και αποστολή κατάλληλων παλμών, με βάση τη μέτρηση αυτή.
- Το όχημα δεν γνωρίζει τη θέση του στο χώρο, καθώς δεν διαθέτει κάποιο σύστημα χαρτογράφησης. Έτσι, κινείται μόνο με βάση τις πληροφορίες που λαμβάνει από τον αισθητήρα απόστασης που είναι σταθερός. Κατά συνέπεια μπορεί να εντοπίζει μόνο αντικείμενα τα οποία βρίσκονται μπροστά του και για αυτό προτείνεται ο χώρος κίνησής του να είναι όσο το δυνατόν άδειος.

5.3 Προτάσεις για μελλοντική έρευνα

Στη συνέχεια αναφέρονται ορισμένες προσθήκες και βελτιώσεις που θα μπορούσαν να γίνουν, ώστε να βελτιωθεί ο τρόπος λειτουργείας του οχήματος ή να προστεθούν νέες δυνατότητες:

- Χρήση ενός σερβοκινητήρα, με τη βοήθεια του οποίου θα κινείται ο αισθητήρας απόστασης. Με αυτόν τον τρόπο σαρώνεται μία περιοχή μπροστά από το όχημα, εντοπίζονται τα πιθανά εμπόδια και κινείται προς την κατεύθυνση που απέχει περισσότερο.
- Προσθήκη περισσότερων αισθητήρων απόστασης. Έτσι, μπορεί να γίνεται έλεγχος για εμπόδια που βρίσκονται πίσω ή πλάγια του οχήματος. Επίσης, μπορούν να χρησιμοποιηθούν δύο αισθητήρες που ελέγχουν προς μία κατεύθυνση για περισσότερη ακρίβεια.
- Εξοπλισμός του οχήματος με ειδικούς αισθητήρες οδομετρίας, χάρη στους οποίους γνωρίζει τη θέση του στο χώρο και τον προσανατολισμό του. Ο υπολογισμός της οδομετρίας, μπορεί να γίνεται από μία πλακέτα Raspberry Pie που θα συνδεθεί και θα ανταλλάζει πληροφορίες με τον Arduino.
- Προσθήκη κάποιου πιο εξελιγμένου αισθητήρα για τη μέτρηση της απόστασης ή ακόμη και κάμερας.
- Προσθήκη κάποιου συστήματος για τη μέτρηση των στροφών του κινητήρα συνεχούς ρεύματος. Ανάλογα με τη μέτρηση αυτή αποφασίζεται η διάρκεια παλμού PWM, που πρέπει να σταλεί στον κινητήρα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Autonomous Robot. Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Autonomous_robot
2. Autonomous Car. Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Autonomous_car
3. Rover (space exploration). Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Rover_(space_exploration)
4. Automated mining. Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Automated_mining
5. Autonomous underwater vehicle. Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Autonomous_underwater_vehicle
6. Ultrasonic Ranging Module HC-SR04 User Guide. Διαθέσιμο στη διεύθυνση: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
7. SC-15WP Waterproof Electronic Speed Control instructions. Διαθέσιμο στη διεύθυνση: https://www.hpiracing.com/assets/documents/instruction_manuals/106570-sc15wp_esc_glb-m-v1.pdf
8. L298 Dual full-bridge driver datasheet. Διαθέσιμο στη διεύθυνση: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf
9. Brushless DC Electric Motor. Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Brushless_DC_electric_motor
10. Servomotor. Διαθέσιμο στη διεύθυνση: https://en.wikipedia.org/wiki/Servomotor
11. Arduino. Διαθέσιμο στη διεύθυνση: https://el.wikipedia.org/wiki/Arduino
12. Πληροφορίες σχετικά με τις εντολές του Arduino. Διαθέσιμο στη διεύθυνση: https://www.arduino.cc

13. Arduino DC Motor Control Tutorial – L298N | PWM | H-Bridge
Διαθέσιμο στη διεύθυνση:
https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/
14. Ι. Καλόμοιρου, Εργαστηριακές Σημειώσεις για τον Προγραμματισμό Συστημάτων Πραγματικού Χρόνου, ΤΕΙ Κ. Μακεδονίας, 2012
Διαθέσιμο στη διεύθυνση:
https://elearning.teicm.gr/course/view.php?id=502
15. Σπ. Καζαρλή, Εργαστηριακές Σημειώσεις του Μαθήματος Ενσωματωμένα Συστήματα (Πρόγραμμα Μεταπτυχιακών Σπουδών στη Ρομποτική), ΤΕΙ Κ. Μακεδονίας, 2017
Διαθέσιμο στη διεύθυνση:
https://elearning.teicm.gr/course/view.php?id=467
16. Παπάζογλου,Παν.,Λιωνής,Σπ.Πολ.*Ανάπτυξη εφαρμογών με Arduino*.Εκδόσεις Τζιόλα,2η Έκδοση, 2018

ΠΑΡΑΡΤΗΜΑ Α

Το πρόγραμμα που ακολουθεί περιλαμβάνει τον βασικό αλγόριθμο που χρησιμοποιείται, ώστε να μπορεί το όχημα να κινείται στο χώρο, εντοπίζοντας εμπόδια, που στη συνέχεια αποφεύγει:

Room_Navigation_L298.ino:

```
//Συμπεριλαμβάνεται στο πρόγραμμα η βιβλιοθήκη που αφορά την οδήγηση σερβοκινητήρων

#include <Servo.h>

//Ορισμός των pin ελέγχου του αισθητήρα

//Τα pin του Arduino συνδέονται με τους αντίστοιχους ακροδέκτες του αισθητήρα που υποδηλώνει το όνομα της κάθε μεταβλητής

int trigger=12;

int echo=11;

// Ορισμός μεταβλητής τύπου Servo

Servo steering_servo;

//Ορισμός μεταβλητών για τη σύνδεση του Arduino με τους ακροδέκτες του ελεγκτή L298N

//Τα pin του Arduino συνδέονται με τους αντίστοιχους ακροδέκτες του ελεγκτή που υποδηλώνει το όνομα κάθε μεταβλητής

int enA=6;

int in1=4;

int in2=2;

//Ορισμός μεταβλητής όπου αποθηκεύεται η τρέχουσα ταχύτητα περιστροφής του κινητήρα
```

```
int current_speed;
//Ορισμός μεταβλητής-σημαίας
boolean flag2;

void setup() {
//Αρχικοποίηση και έναρξη της σειριακής κονσόλας
  Serial.begin(9600);
//Ορισμός των pin ελέγχου της H-bridge ως εξόδους
  pinMode(enA,OUTPUT);
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
//Ορισμός του pin Trig του αισθητήρα ως έξοδο
  pinMode(trigger,OUTPUT);
//Ορισμός του pin Echo του αισθητήρα ως είσοδο
  pinMode(echo,INPUT);
//Ορισμός του pin για τον έλεγχο του σερβοκινητήρα
  steering_servo.attach(3);
//Θέτουμε αρχική τιμή στο σέρβο, ώστε οι μπροστινοί τροχοί να βρίσκονται σε
ευθεία με τους πίσω
  steering_servo.write(95);
  flag2=false;
}

void loop() {
//Ορισμός και αρχικοποίηση μεταβλητών
```

```
  long distance=0,sum=0,avg_distance=0;
  int i,j,stopped=0;
  boolean flag=false;
  current_speed=0;
//Πρώτος έλεγχος απόστασης και εύρεση μέσης τιμής
  Serial.print("First check");
  Serial.print('\n');
  for(i=0;i<5;i++){
    distance=obstacle_distance();
    Serial.print('\n');
    Serial.print("i");
    Serial.print(":");
    Serial.print(i);
    Serial.print('\t');
    Serial.print("Value:");
    Serial.print (distance);
    sum=distance+sum;
    Serial.print('\t');
    Serial.print("Sum:");
    Serial.print (sum);
    delay(25);
  }
  avg_distance=sum/5;
  Serial.print('\n');
```

```
  Serial.print("Average value:" );
  Serial.print(avg_distance);
  Serial.print('\n');
  Serial.print("----------------");
  Serial.print('\n');
//Επιταχυνόμενη κίνηση προς τα μπροστά αν η απόσταση είναι μεγαλύτερη
από ορισμένη τιμή
  if (avg_distance>=100){
//Αν δεν έχει προηγηθεί κίνηση προς τα πίσω το όχημα κινείται σε ευθεία
    if (flag2=false){
      steering_servo.write(95);
      delay(20);
    }
//Αλλιώς στρίβει ώστε να αποφύγει το εμπόδιο που βρίσκεται μπροστά του
    else{
      delay(100);
      steering_servo.write(95);
      delay(50);
      steering_servo.write(120);
      delay(200);
      flag2=false;
    }
//Ορισμός της φοράς περιστροφής του κινητήρα για κίνηση προς τα εμπρός
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
```

//Σταδιακή αύξηση της ταχύτητας περιστροφής του κινητήρα μέχρι μία μέγιστη τιμή

```
    for(j=current_speed;j<=180;j=j+10){
      analogWrite(enA,j);
      Serial.println(j);
      delay(20);
    }
   delay(3000);
   steering_servo.write(95);
   flag=true;
```

//Επανάληψη όσο η μεταβλητή flag είναι true, δηλαδή όσο η απόσταση ειναι μεγαλύτερη ή ίση του 100

```
    while(flag=true){
```

//Έλεγχος απόστασης και εύρεση μέσης τιμής

```
      distance=0,sum=0,avg_distance=0;
      for(i=0;i<5;i++){
        distance=obstacle_distance();
        Serial.print('\n');
        Serial.print("i");
        Serial.print(":");
        Serial.print(i);
        Serial.print('\t');
        Serial.print("Value:");
        Serial.print (distance);
        sum=distance+sum;
```

```
        Serial.print('\t');
        Serial.print("Sum:");
        Serial.print (sum);
        delay(100);
      }
    avg_distance=sum/5;
    Serial.print('\n');
    Serial.print("Average value:" );
    Serial.print(avg_distance);
    Serial.print('\n');
//Κίνηση προς τα μπροστά με σταθερή ταχύτητα αν ικανοποιείται η συνθήκη
    if (avg_distance>=100){
      analogWrite(enA,180);
      current_speed=180;
    }
//Αλλιώς έξοδος από το βρόγχο επανάληψης
    else{
      flag=false;
      break;
      }
    }
  }
//Ύπαρξη εμποδίου σε απόσταση μικρότερη του 100
  else{
```

```
    flag2=true;
//Σταδιακή μείωση της ταχύτητας
    for(j=current_speed;j>=80;j=j-10){
      analogWrite(enA,j);
      Serial.println(j);
      delay(5);
    }
    current_speed=80;
    delay(100);
//Αλλαγή της γωνίας των μπροστινών τροχών
    steering_servo.write(70);
//Αλλαγή της φοράς περιστροφής του dc motor-Κίνηση προς τα πίσω
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
//Σταδιακή αύξηση της ταχύτητας περιστροφής του κινητήρα μέχρι μία μέγιστη
τιμή
    for(j=current_speed;j<=180;j=j+10){
      analogWrite(enA,j);
      Serial.println(j);
      delay(15);
    }
    delay(1000);
    current_speed=180;
  }
}
```

```
//Συνάρτηση υπολογισμού της απόστασης από εμπόδιο
  long obstacle_distance(){
  long duration,distance;
  digitalWrite(trigger,LOW);
  delayMicroseconds(2);
  digitalWrite(trigger,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger,LOW);
  duration=pulseIn(echo,HIGH,30000);
  if(duration==0)
  {
    pinMode(echo,OUTPUT);
    delay(10);
    digitalWrite(echo,LOW);
    delay(10);
    pinMode(echo,INPUT);
    delay(10);
  }
  distance=duration/29/2;
  return distance;
}
```

ΠΑΡΑΡΤΗΜΑ Β

Το πρόγραμμα που ακολουθεί βοήθησε στην κατανόηση του τρόπου λειτουργείας του αισθητήρα απόστασης HC-SR04. Πιο συγκεκριμένα, όταν εκτελείται το παρακάτω πρόγραμμα, εμφανίζεται στη σειριακή κονσόλα η διάρκεια του παλμού που δέχεται ο αισθητήρας σε microseconds, καθώς και η απόσταση από το αντικείμενο σε εκατοστά. Να σημειωθεί εδώ πως ο τρόπος μέτρησης της απόστασης στο βασικό αλγόριθμο, βασίστηκε στη λειτουργεία αυτού του προγράμματος.

Distance_Sensor.ino:

```
//Προβολή στη σειριακή κονσόλα της διάρκειας παλμού και της απόστασης
που μετριέται από τον αισθητήρα απόστασης

//Καθορισμός των ακροδεκτών του Arduino που θα συνδεθούν με τους
ακροδέκτες trigger και echo του αισθητήρα

int trigger=12;

int echo=11;


void setup() {

//Αρχικοποίηση και έναρξη της σειριακής κονσόλας

Serial.begin(9600);

//Ορισμός του pin 12 ως έξοδο και του pin 11 ως είσοδο

pinMode(trigger,OUTPUT);

pinMode(echo,INPUT);

}


void loop() {

//Ορισμός δύο μεταβλητών τύπου long για την αποθήκευση της διάρκειας
παλμού και της απόστασης

long duration,distance;
```

```
//Παρέχουμε στο pin 12 έναν σύντομο παλμό LOW ώστε στη συνέχεια να παραχθεί ένας καθαρός παλμός HIGH

digitalWrite(trigger,LOW);

delayMicroseconds(2);

digitalWrite(trigger,HIGH);

delayMicroseconds(10);

digitalWrite(trigger,LOW);

//Ανάγνωση του παλμού που ανακλάται από το αντικείμενο, από το pin echo του αισθητήρα

duration=pulseIn(echo,HIGH,30000);

//Επαναφορά του pin 11 σε κατάσταση LOW, σε περίπτωση που η διάρκεια παλμού είναι ίση με μηδέν

if(duration==0)

  {

    pinMode(echo,OUTPUT);

    delay(10);

    digitalWrite(echo,LOW);

    delay(10);

    pinMode(echo,INPUT);

    delay(10);

  }

//Υπολογισμός της απόστασης σύμφωνα με τη διάρκεια του παλμού

  distance=duration/29/2;

//Εκτύπωση των αποτελεσμάτων στη σειριακή κονσόλα

Serial.print("Διάρκεια παλμού:");

Serial.print(duration);

Serial.print(", ");

Serial.print("Απόσταση:");

Serial.print(distance);

Serial.print("cm");

Serial.println();
```

```
//Καθυστέρηση ενός δευτερολέπτου μέχρι την επόμενη εκτέλεση της loop()
delay(1000);
}
```

ΠΑΡΑΡΤΗΜΑ Γ

Τα προγράμματα που παρουσιάζονται στη συνέχεια χρησιμοποιήθηκαν για την καλύτερη κατανόηση του τρόπου οδήγησης κινητήρων συνεχούς ρεύματος μέσω της H-bridge. Στην πρώτη περίπτωση το όχημα DIY αρχικά κινείται προς τα εμπρός με σταθερή ταχύτητα και αφού ακινητοποιηθεί για μερικά δευτερόλεπτα, εκτελεί την ίδια κίνηση αλλά αυτή τη φορά προς τα πίσω. Κατά την εκτέλεση του δεύτερου προγράμματος το όχημα αρχικά κινείται επιταχυνόμενα προς τα μπροστά, με τους κινητήρες να περιστρέφονται από την ελάχιστη έως της μέγιστη δυνατή ταχύτητα. Έπειτα το όχημα επιβραδύνεται μέχρι να σταματήσει τελείως για μερικά δευτερόλεπτα και στη συνέχεια, αφού αλλάξει η φορά περιστροφής των κινητήρων επαναλαμβάνεται η παραπάνω κίνηση με αντίθετη κατεύθυνση.

dcmotor_control.ino:

```
//Κίνηση του οχήματος προς τα εμπρός με σταθερή ταχύτητα και στη συνέχεια κίνηση προς τα πίσω με την ίδια ταχύτητα

//Ορισμός μεταβλητών για τη σύνδεση του Arduino με τους ακροδέκτες του ελεγκτή L298N

//Τα pin του Arduino συνδέονται με τους αντίστοιχους ακροδέκτες του ελεγκτή που υποδηλώνει το όνομα της κάθε μεταβλητής

int enA=10;

int in1=9;

int in2=8;

int enB=5;

int in3=6;

int in4=7;
```

```
void setup() {
//Ορισμός των pin ελέγχου της H-bridge ως εξόδους
  pinMode(enA,OUTPUT);
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(enB,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
}
//Συνάρτηση κατά την εκτέλεση της οποίας το όχημα κινείται προς τα πίσω
void Reverse(){
//Ορισμός της φοράς περιστροφής των κινητήρων
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
//Ορισμός της ταχύτητας περιστροφής των κινητήρων
  analogWrite(enA,130);
  analogWrite(enB,130);
//Καθυστέρηση τεσσάρων δευτερολέπτων-Το όχημα κινείται όπως ορίστηκε παραπάνω
  delay(4000);
//Οι κινητήρες σταματούν να περιστρέφονται-Ακινητοποίηση του οχήματος
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
```

```
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
}
//Συνάρτηση κατά την εκτέλεση της οποίας το όχημα κινείται προς τα μπροστά
void Forward(){
//Ορισμός της φοράς περιστροφής των κινητήρων
  digitalWrite(in1,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
//Ορισμός της ταχύτητας περιστροφής των κινητήρων
  analogWrite(enA,130);
  analogWrite(enB,130);
//Καθυστέρηση τεσσάρων δευτερολέπτων-Το όχημα κινείται όπως ορίστηκε παραπάνω
  delay(4000);
//Οι κινητήρες σταματούν να περιστρέφονται-Ακινητοποίηση του οχήματος
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
}

void loop() {
//Καλείται η συνάρτηση Forward()
```

```
  Forward();

  delay(1000);

//Καλείται η συνάρτηση Reverse()

  Reverse();

  delay(1000);

}
```

dc_motor_2.ino:

```
//Επιταχυνόμενη κίνηση του οχήματος προς τα εμπρός και προς τα πίσω

//Ορισμός μεταβλητών για τη σύνδεση του Arduino με τους ακροδέκτες του ελεγκτή L298N

//Τα pin του Arduino συνδέονται με τους αντίστοιχους ακροδέκτες του ελεγκτή που υποδηλώνει το όνομα της κάθε μεταβλητής

int enA=10;

int in1=9;

int in2=8;

int enB=5;

int in3=6;

int in4=7;

int i;

void setup() {

//Ορισμός των pin ελέγχου της H-bridge ως εξόδους

  pinMode(enA,OUTPUT);

  pinMode(in1,OUTPUT);

  pinMode(in2,OUTPUT);

  pinMode(enB,OUTPUT);

  pinMode(in3,OUTPUT);

  pinMode(in4,OUTPUT);

}
```

```
void loop(){
//Ορισμός της φοράς περιστροφής των κινητήρων για κίνηση του οχήματος προς τα εμπρός
  digitalWrite(in1,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
//Επιταχυνόμενη περιστροφή των κινητήρων από την ελάχιστη έως και τη μέγιστη τιμή
  for(i=0;i<256;i++){
    analogWrite(enA,i);
    analogWrite(enB,i);
    delay(20);
    }
//Επιβραδυνόμενη περιστροφή των κινητήρων από τη μέγιστη έως και την ελάχιστη τιμή
   for(i=255;i>=0;i--){
    analogWrite(enA,i);
    analogWrite(enB,i);
    delay(20);
    }
//Καθυστέρηση δύο δευτερολέπτων
  delay(2000);
//Αλλαγή της φοράς περιστροφής των κινητήρων, για να μπορεί το όχημα να κινηθεί προς τα πίσω
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
```

```
//Επιταχυνόμενη περιστροφή των κινητήρων από την ελάχιστη έως και τη μέγιστη τιμή
  for(i=0;i<256;i++){
    analogWrite(enA,i);
    analogWrite(enB,i);
    delay(20);
    }
//Επιβραδυνόμενη περιστροφή των κινητήρων από τη μέγιστη έως και την ελάχιστη τιμή
   for(i=255;i>=0;i--){
    analogWrite(enA,i);
    analogWrite(enB,i);
    delay(20);
    }
//Ακινητοποίηση του οχήματος
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
  delay(500);
}
```

ΠΑΡΑΡΤΗΜΑ Δ

Στη συνέχεια παρατίθεται ένα από τα προγράμματα που χρησιμοποιήθηκαν για την οδήγηση ενός κινητήρα τύπου σέρβο (SG90) και την κατανόηση του τρόπου λειτουργείας του. Ο άξονας του σέρβο περιστρέφεται προς τη μία κατεύθυνση καλύπτοντας όλες τις πιθανές γωνίες και αμέσως μετά εκτελεί την ίδια κίνηση προς την αντίθετη κατεύθυνση.

servo_control:

```
//Περιστροφή του άξονα ενός σερβοκινητήρα, καλύπτοντας όλες τις πιθανές γωνίες

//Συμπεριλαμβάνεται στο πρόγραμμα η βιβλιοθήκη που αφορά την οδήγηση σερβοκινητήρων

#include <Servo.h>

//Ορίζεται μία μεταβλητή τύπου Servo

Servo servo_1;

//Ορίζεται μία ακέραια μεταβλητή και γίνεται αρχικοποίηση της θέση του άξονα περιστροφής

int servo_position=0;

void setup() {

//Ορίζεται το pin ελέγχου του σέρβο

  servo_1.attach(3);

}
```

```
void loop() {
//Η θέση του άξονα αλλάζει σταδιακά από την ελάχιστη έως τη μέγιστη τιμή

for(servo_position=0;servo_position<=180;servo_position=servo_position+5){
    servo_1.write(servo_position);
    delay(1000);
  }
//Ο άξονας περιστροφής επιστρέφει στην αρχική του θέση, εκτελώντας την προηγούμενη κίνηση με αντίθετη φορά
  for(servo_position=180;servo_position>=0;servo_position=servo_position-5){
    servo_1.write(servo_position);
    delay(1000);
  }
}
```

ΠΑΡΑΡΤΗΜΑ Ε

Τα δύο προγράμματα που ακολουθούν μπορούν να χρησιμοποιηθούν ώστε να βρεθούν οι τιμές λειτουργείας ενός dc κινητήρα ή ενός σερβοκινητήρα. Τα δύο προγράμματα έχουν παρόμοιο τρόπο λειτουργείας. Οι αντίστοιχοι κινητήρες περιστρέφονται ανάλογα με τις τιμές που εισάγονται από το χρήστη μέσω της σειριακής κονσόλας, εφόσον οι τιμές αυτές είναι έγκυρες. Η βασική διαφορά των δύο προγραμμάτων είναι πως το πρώτο αφορά την οδήγηση κινητήρων μέσω του ελεγκτή ταχύτητας HPI EN-1 ή κάποιου αντίστοιχου, ενώ το δεύτερο αφορά την περίπτωση οδήγησης με το ολοκληρωμένο L298N (H-bridge).

my_motor_test.ino:

```
//Το μοτέρ συνεχούς ρεύματος ή το σέρβο περιστρέφονται ανάλογα με τις τιμές που εισάγουμε από τη σειριακή κονσόλα

//Αφορά την οδήγηση μέσω του ελεγκτή ταχύτητας HPI EN-1

//Ορισμός μεταβλητής τύπου String

String readString;

//Συμπεριλαμβάνεται στο πρόγραμμα η βιβλιοθήκη που αφορά την οδήγηση σερβοκινητήρων

#include <Servo.h>

//Ορίζονται δύο μεταβλητές τύπου Servo

//Στην πρώτη αποθηκεύονται τιμές για την οδήγηση του dc motor, ενώ στη δεύτερη για το σέρβο

Servo myservo,myservo2;


void setup() {
```

```
//Αρχικοποίηση και έναρξη της σειριακής κονσόλας
  Serial.begin(9600);
//Ορισμός του pin για τον έλεγχο του κινητήρα συνεχούς ρεύματος
  myservo.attach(5);
//Θέτουμε αρχική τιμή στο μοτέρ, για την οποία δεν περιστρέφεται
  myservo.writeMicroseconds(1480);
//Ορισμός του pin για τον έλεγχο του σερβοκινητήρα
  myservo2.attach(3);
//Θέτουμε αρχική τιμή στο σέρβο, για την οποία το όχημα κινείται σε ευθεία
  myservo.write(95);
//Εμφάνιση μηνύματος για να γνωρίζει ο χρήστης τι τιμές να εισάγει
  Serial.println("dc motor or servo input (microseconds or angle)");
}

void loop() {
//Επαναλαμβάνεται όσο εισάγουμε χαρακτήρες
  while (Serial.available()) {
//Ανάγνωση ενός χαρακτήρα ενός byte από τη σειριακή κονσόλα
    char c = Serial.read();
//Μετατροπή της μεταβλητής τύπου char σε String
    readString += c;
//Καθυστέρηση ώστε να προλάβει ο buffer να δεχθεί τον επόμενο χαρακτήρα
    delay(2);
  }
```

```
//Έλεγχος πως έχει εισαχθεί String
  if (readString.length() >0) {
//Εκτύπωση του String για να επιβεβαιωθεί πως έχει πάρει την επιθυμητή τιμή
    Serial.println(readString);
//Μετατροπή του String σε ακέραια μεταβλητή
    int n = readString.toInt();
//Επιλογή με βάση την τιμή της μεταβλητής n
//Αν είναι μεγαλύτερη ή ίση του 500 σημαίνει πως θέλουμε να περιστραφεί ο
dc κινητήρας
    if(n >= 500)
    {
      Serial.print("writing Microseconds: ");
      Serial.println(n);
//Αποστολή παλμού τύπου PWM στο pin οδήγησης του κινητήρα με διάρκεια
n
      myservo.writeMicroseconds(n);
    }
//Για τιμές μικρότερες του 500 περιστρέφεται ο σερβοκινητήρας
    else
    {
      Serial.print("writing Angle: ");
      Serial.println(n);
//Αποστολή παλμού τύπου PWM στο pin οδήγησης του σέρβο με κατάλληλη
διάρκεια ώστε να περιστραφεί n μοίρες
      myservo2.write(n);
```

```
    }
//Το String αδειάζει για να δεχθεί την επόμενη τιμή
    readString="";
  }
}
```

my_motor_test_h_bridge.ino:

```
//Το μοτέρ συνεχούς ρεύματος περιστρέφεται ανάλογα με τις τιμές που εισάγουμε από τη σειριακή κονσόλα
//Αφορά την οδήγηση με H-bridge(L298N)
//Συμπεριλαμβάνεται στο πρόγραμμα η βιβλιοθήκη που αφορά την οδήγηση σερβοκινητήρων
#include <Servo.h>
//Ορισμός μεταβλητής τύπου Servo
Servo steering_servo;
//Ορισμός μεταβλητών για τη σύνδεση του Arduino με τους ακροδέκτες του ελεγκτή L298N
//Τα pin του Arduino συνδέονται με τους αντίστοιχους ακροδέκτες του ελεγκτή που υποδηλώνει το όνομα της κάθε μεταβλητής
int enA=6;
int in1=4;
int in2=2;
//Ορισμός μεταβλητής τύπου String
String readString;
```

```
void setup() {
//Αρχικοποίηση και έναρξη της σειριακής κονσόλας
  Serial.begin(9600);
//Ορισμός των pin ελέγχου της H-bridge ως εξόδους
  pinMode(enA,OUTPUT);
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
//Ορισμός του pin για τον έλεγχο του σερβοκινητήρα
  steering_servo.attach(3);
//Θέτουμε αρχική τιμή στο σέρβο, ώστε οι μπροστινοί τροχοί να βρίσκονται σε
ευθεία με τους πίσω
  steering_servo.write(95);
//Εμφάνιση μηνύματος για να γνωρίζει ο χρήστης τι τιμές να εισάγει
  Serial.println("servo speed input (0-255)");
}

void loop() {
//Επαναλαμβάνεται όσο εισάγουμε χαρακτήρες
  while (Serial.available()) {
//Ανάγνωση ενός χαρακτήρα ενός byte από τη σειριακή κονσόλα
    char c = Serial.read();
//Μετατροπή της μεταβλητής τύπου char σε String
    readString += c;
//Καθυστέρηση ώστε να προλάβει ο buffer να δεχθεί τον επόμενο χαρακτήρα
    delay(2);
```

```
  }
//Έλεγχος πως έχει εισαχθεί String
  if (readString.length() >0) {
//Εκτύπωση του String για να επιβεβαιωθεί πως έχει πάρει την επιθυμητή τιμή
    Serial.println(readString);
//Μετατροπή του String σε ακέραια μεταβλητή
    int n = readString.toInt();


//Έλεγχος της τιμής που έχει εισαχθεί
//Αν βρίσκεται εντός των ορίων εκτελείται το παρακάτω κομμάτι κώδικα
    if((n > 0)&&(n<=255))
    {
      Serial.print("Speed: ");
      Serial.println(n);
//Ορισμός της φοράς περιστροφής του κινητήρα για κίνηση προς τα εμπρός
      digitalWrite(in1,LOW);
      digitalWrite(in2,HIGH);
//Ορισμός της ταχύτητας περιστροφής του κινητήρα
      analogWrite(enA,n);
//Καθυστέρηση τριών δευτερολέπτων
      delay(3000);
//Ακινητοποίηση του οχήματος
      digitalWrite(in1,LOW);
      digitalWrite(in2,LOW);
```

```
    }
//Σε περίπτωση που η τιμή που έχει εισάγει ο χρήστης εμφανίζεται μήνυμα σφάλματος
    else
    {
      Serial.print("Wrong input value!!!");
    }
//Το String αδειάζει για να δεχθεί την επόμενη τιμή
    readString=""; //empty for next input
  }
}
```