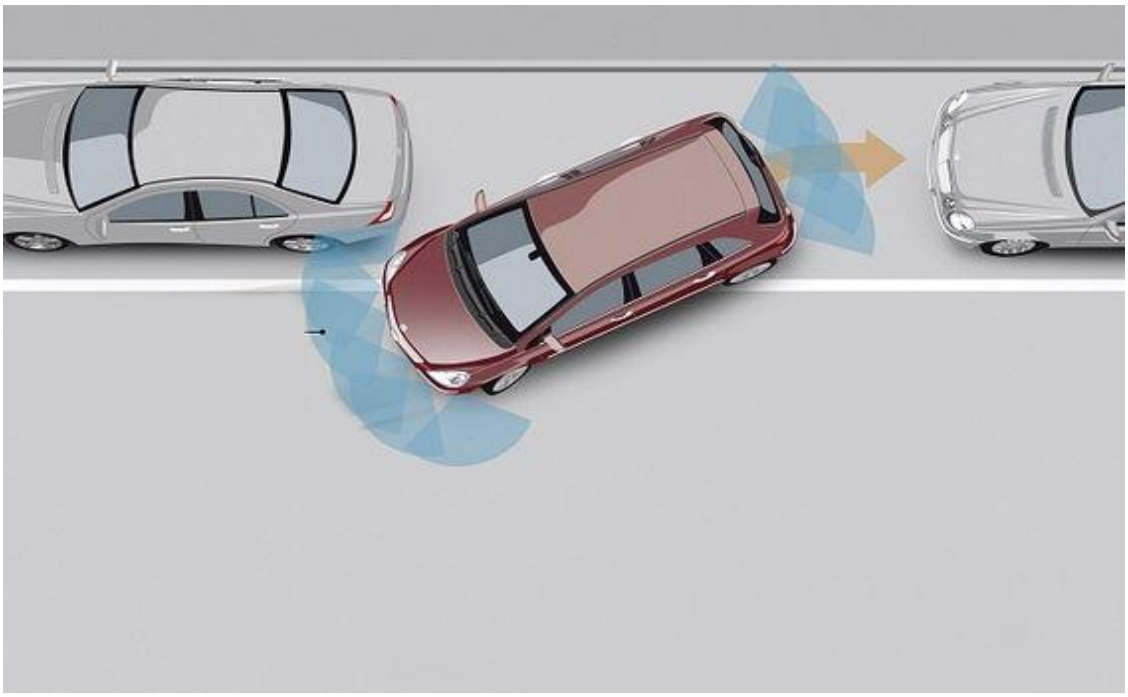
**ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ**
**ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ**
**ΤΕΧΝΟΛΟΓΙΚΟΥ ΤΟΜΕΑ**
**Α.Ε.Ι. Πειραιά Τ.Τ.**

ΤΜΗΜΑ
ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ
ΤΕΧΝΟΛΟΓΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ

# ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

## *Αυτόματο παρκάρισμα αυτοκινήτου με χρήση επεξεργαστή Arduino*



| | | |
|---|---|---|
| **ΕΠΩΝΥΜΟ:** | Μάλλης | Βογιατζάκης |
| **ΟΝΟΜΑ:** | Ιωάννης | Ιωάννης |
| **Α.Μ.:** | 41431 | 41552 |

**ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ :** Βαρσάμης Χρήστος-Πλάτωνας

**ΑΘΗΝΑ 2018**

**ΕΥΧΑΡΙΣΤΙΕΣ**

Θεωρούμε υποχρέωσή μας να ευχαριστήσουμε τον επιβλέποντα καθηγητή κύριο Βαρσάμη Χρήστο-Πλάτωνα για την πολύτιμη καθοδήγησή του. Επιπλέον, οφείλουμε να αφιερώσουμε την πτυχιακή μας εργασία στους γονείς μας που μας συμπαραστάθηκαν όλα τα χρόνια της φοίτησης μας στο Α.Ε.Ι Πειραιά τ.τ.

## Περίληψη

Ο σκοπός της παρούσας εργασίας είναι να παρουσιάσει ένα απλό πρωτότυπο σύστημα στάθμευσης, το οποίο μπορεί να εκτελέσει αυτόνομους παράλληλους χειρισμούς στάθμευσης. Αυτό το σύστημα περιλαμβάνει μια σειρά αισθητήρων εγγύτητας καθώς και έναν κεντρικό μικροεπεξεργαστή που ελέγχει το αυτοκίνητο.

Το αυτόνομο αυτοκίνητο στάθμευσης εντοπίζει πρώτα το περιβάλλον του με μια σειρά υπέρυθρων αισθητήρων εγγύτητας που τοποθετούνται γύρω από τα όρια του αυτοκινήτου. Αυτοί οι αισθητήρες συνδέονται με έναν μικροεπεξεργαστή Arduino, ο οποίος διαβάζει τις τάσεις εξόδου/εισόδου από τους αισθητήρες. Στη συνέχεια το αυτοκίνητο κινείται ανάλογα, βάσει των τιμών που δέχεται από τους αισθητήρες.

Η κατεύθυνση του αυτοκινήτου καθορίζεται από τους μπροστινούς τροχούς του και οδηγείται από τους πίσω τροχούς. Ο εμπρός και ο πίσω άξονας συνδέονται με έναν ηλεκτροκινητήρα αντίστοιχα και με τη διαμόρφωση της γέφυρας "Η". Η γέφυρα "Η" επιτρέπει στους κινητήρες να αλλάζουν κατευθύνσεις, αριστερά και δεξιά για τους μπροστινούς τροχούς, εμπρός και πίσω για τους πίσω τροχούς καθορίζοντας πλήρως τις κινήσεις του οχήματος.

## Abstract

The purpose of this paper is to present a simple prototype parking system, which can perform autonomous parallel parking operations. This system includes a range of proximity sensors as well as a central microprocessor that controls the car.

The stand-alone parking car first locates its environment with a range of infrared proximity sensors located around the car's limits. These sensors are connected to an Arduino microprocessor, which reads the output / input voltages from the sensors. Then the car moves accordingly, based on the values it receives from the sensors.

The direction of the car is determined by its front wheels and driven by the rear wheels. The front and rear axles are connected to an electric motor, respectively, and to the "H" bridge configuration. The H-bridge allows the engines to shift left and right for the front wheels, front and rear for the rear wheels, fully defining the vehicle's movements.

## Πίνακας περιεχομένων

## Εισαγωγή

Το αυτόνομο σύστημα στάθμευσης μπορεί να αποτελέσει πρόκληση για τους οδηγούς. Το τυπικό μοντέρνο αυτοκίνητο δεν περιέχει καθόλου συστήματα που να διευκολύνουν τη στάθμευση. Μέχρι στιγμής η Ford με το ''Active Park Assist'' , η Toyota με το ''Intelligent Parking Assist'' και η BMW με το ''Parking Assistant'' είναι οι εταιρίες που έχουν εγκαταστήσει σε κάποια μοντέλα τους στην αγορά ένα αυτόματο σύστημα παρκαρίσματος όπου το μόνο που έχει να κάνει ο οδηγός του οχήματος είναι να ελέγχει το γκάζι και το φρένο.

Ο κύριος στόχος της παρούσας εργασίας είναι να σχεδιάσει ένα απλό πρωτότυπο σύστημα στάθμευσης, το οποίο μπορεί να πραγματοποιεί αυτόματους παράλληλους χειρισμούς στάθμευσης. Αυτό το σύστημα περιλαμβάνει μια σειρά αισθητήρων εγγύτητας καθώς και έναν κεντρικό μικροεπεξεργαστή που ελέγχει το αυτοκίνητο. Το συγκεκριμένο σύστημα θα λειτουργούσε ιδανικά τόσο σε μοντέλο κλίμακας ενός αυτοκινήτου, όσο και σε αυτοκίνητο μεγέθους ζωής. Αυτό το έργο διερευνά πώς η είσοδος αισθητήρα και ένας αλγόριθμος μπορούν να χρησιμοποιηθούν για πρακτικές εφαρμογές. Αυτή η εργασία καλύπτει τα διάφορα στοιχεία που χρησιμοποιούνται για τη δημιουργία του συστήματος στάθμευσης, συμπεριλαμβανομένου ενός μικροεπεξεργαστή Arduino, υπερηχητικών αισθητήρων, γεφυρών "Η" και ηλεκτροκινητήρων. Περιλαμβάνει επίσης τον αλγόριθμο στάθμευσης που εφαρμόζεται στο σύστημα.

# ΚΕΦΑΛΑΙΟ 1

## Εισαγωγή στον επεξεργαστή Arduino

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


Στην ουσία, πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται στον μικροελεγκτή ATmega της Atmel του οποίου όλα τα σχέδια, καθώς και το software που χρειάζεται για την λειτουργία του, διανέμονται ελεύθερα και

δωρεάν ώστε να μπορεί να κατασκευαστεί από τον καθένα (απ' όπου και ο περίεργος -για hardware- χαρακτηρισμός «ανοικτού κώδικα»). Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, καθώς ο χρήστης μπορεί να συνδέσει επάνω του πολλαπλές μονάδες εισόδου/εξόδου και να προγραμματίσει τον μικροελεγκτή να δέχεται δεδομένα από τις μονάδες εισόδου, να τα επεξεργάζεται και να στέλνει κατάλληλες εντολές στις μονάδες εξόδου.

Ο Arduino βέβαια, δεν είναι ούτε ο μοναδικός, ούτε και ο καλύτερος δυνατός τρόπος για την δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής. Όμως το κύριο πλεονέκτημά του είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, συντηρήσει και επεκτείνει μια ανάλογου μεγέθους online γνωσιακή βάση. Έτσι, παρότι ένας έμπιρος ηλεκτρονικός μπορεί να προτιμήσει διαφορετική πλατφόρμα ή εξαρτήματα ανάλογα με την εφαρμογή που έχει στον νου του, το Arduino, με το εκτενές documentation, καταφέρνει να κερδίσει όλους αυτούς των οποίων οι γνώσεις στα ηλεκτρονικά περιορίζονται σε αυτές τις λίγες που απέκτησαν στο σχολείο.

## 1.1 Πλεονεκτήματα επιλογής του Arduino

Χάρη στην απλή και προσβάσιμη εμπειρία του χρήστη, ο Arduino έχει χρησιμοποιηθεί σε χιλιάδες διαφορετικά έργα και εφαρμογές. Το λογισμικό Arduino είναι εύκολο στη χρήση για αρχάριους, αλλά αρκετά ευέλικτο για τους προχωρημένους χρήστες. Εκτελείται σε Mac, Windows και Linux. Οι εκπαιδευτικοί και οι φοιτητές το χρησιμοποιούν για να κατασκευάσουν επιστημονικά όργανα χαμηλού κόστους, να αποδείξουν τις αρχές της χημείας και της φυσικής ή να ξεκινήσουν τον προγραμματισμό και τη ρομποτική. Οι σχεδιαστές και οι αρχιτέκτονες κατασκευάζουν διαδραστικά πρωτότυπα, ενώ μουσικοί και καλλιτέχνες το χρησιμοποιούν για εγκαταστάσεις και πειραματισμό με νέα μουσικά όργανα. Ο Arduino είναι ένα βασικό εργαλείο για να μάθετε νέα πράγματα. Οποιοσδήποτε μπορεί να ξεκινήσει να παίζει ακολουθώντας βήμα προς βήμα τις οδηγίες ενός κιτ ή να μοιράζεται ιδέες online με άλλα μέλη της κοινότητας Arduino.

Βασικά πλεονεκτήματα πλατφόρμας Arduino

- **Οικονομική:** Οι πλακέτες Arduino είναι σχετικά φθηνές σε σύγκριση με άλλες πλατφόρμες μικροελεγκτών. Η λιγότερο δαπανηρή έκδοση του module Arduino μπορεί να συναρμολογηθεί με το χέρι και ακόμη και οι προ-συναρμολογημένες μονάδες Arduino έχουν αρκετά χαμηλό κόστος.

- **Μεταφέρσιμη:** Το λογισμικό Arduino (IDE) λειτουργεί σε λειτουργικά συστήματα Windows, Macintosh OSX και Linux. Τα περισσότερα συστήματα μικροελεγκτών περιορίζονται στα Windows.

- Απλό και σαφές περιβάλλον προγραμματισμού: Το λογισμικό Arduino (IDE) είναι εύκολο στη χρήση για αρχάριους, αλλά αρκετά ευέλικτο για να επωφεληθούν και οι προηγμένοι χρήστες. Για τους εκπαιδευτικούς είναι βολικά βασισμένο στο περιβάλλον προγραμματισμού επεξεργασίας, έτσι ώστε οι μαθητές που μαθαίνουν να προγραμματίζουν σε αυτό το περιβάλλον να είναι εξοικειωμένοι με τον τρόπο λειτουργίας του IDE του Arduino.

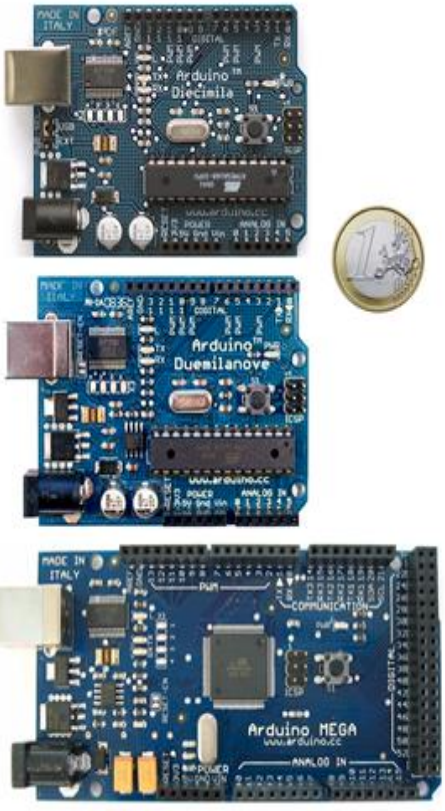
- **Ανοιχτού κώδικα και επεκτάσιμο Λογισμικό:** Το λογισμικό Arduino είναι φημισμένο ως ανοικτού κώδικα, διαθέσιμο για επέκταση από έμπειρους προγραμματιστές. Η γλώσσα μπορεί να επεκταθεί μέσω των βιβλιοθηκών C ++ και οι άνθρωποι που θέλουν να κατανοήσουν τις τεχνικές λεπτομέρειες μπορούν να κάνουν το άλμα από το Arduino στη γλώσσα προγραμματισμού AVR C στην οποία βασίζεται. Ομοίως μπορείτε να προσθέσετε τον κώδικα AVR-C απευθείας στα προγράμματα Arduino, αν θέλετε.

- **Ανοιχτού κώδικα και επεκτάσιμο υλικό:** Τα σχέδια για τις πλακέτες του Arduino δημοσιεύονται με άδεια Creative Commons, έτσι οι έμπειροι σχεδιαστές κυκλωμάτων μπορούν να δημιουργήσουν τη δική τους έκδοση της ενότητας, να την επεκτείνουν και να τη βελτιώσουν.

Ο Arduino κυκλοφορεί σε πολλές διαφορετικές εκδόσεις. Από τις επίσημες εκδόσεις (Duemilanove, Diecimila, Nano, Mega, Bluetooth, LilyPad, Mini, Mini USB, Pro, Pro Mini, Serial και Serial SS) συνιστάται κυρίως η αγορά του Arduino Duemilanove ή τουλάχιστον των Diecimila ή Mega, επειδή διαθέτουν υποδοχή USB και είναι συμβατές με τα shield.

Διαφορές στις προτεινόμενες εκδόσεις του Arduino.

Ο Arduino Diecimila έχει ουσιαστικά δύο βασικές διαφορές με το Duemilanove:




- Βασίζεται στον μικροελεγκτή ATmega168, ο οποίος διαθέτει ακριβώς τη μισή μνήμη από τον ATmega328, δηλαδή 1Kb SRAM, 512bytes EEPROM και 16Kb Flash (14 ελεύθερα λόγω του bootloader).
- Δεν επιλέγει αυτόματα μεταξύ της εξωτερικής τροφοδοσίας και της τροφοδοσίας μέσω της θύρας USB. Το Diecimila διαθέτει ειδικό jumper με το οποίο μπορείτε να επιλέξετε χειροκίνητα την πηγή τροφοδοσίας.

Το Arduino Mega είναι η πιο εξελιγμένη έκδοση με τον μικροελεγκτή ATmega1280 και έχει αρκετά μεγαλύτερο μέγεθος. Οι διαφορές του από το Duemilanove είναι:

- Τετραπλάσια μνήμη (8Kb SRAM, 4Kb EEPROM, 128Kb Flash).
- 40 επιπλέον ψηφιακά pin εισόδου/εξόδου (σύνολο 54)
- 10 επιπλέον pin αναλογικής εισόδου (σύνολο 16)
- Υποστήριξη ψευδοαναλογικής εξόδου PWM σε 8 ακόμα ψηφιακά pin (σύνολο 14 PWM pin)
- Υποστήριξη εξωτερικού interrupt σε 4 ακόμα ψηφιακά pin (σύνολο 6 interrupt)
- 3 επιπλέον σειριακά interface (σύνολο 4) από τα οποία το ένα προωθείται στον ελεγκτή Serial-Over-USB όπως στο Duemilanove για σύνδεση με τον υπολογιστή.

## 1.2 Γλώσσα προγραμματισμού

Η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring, μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει

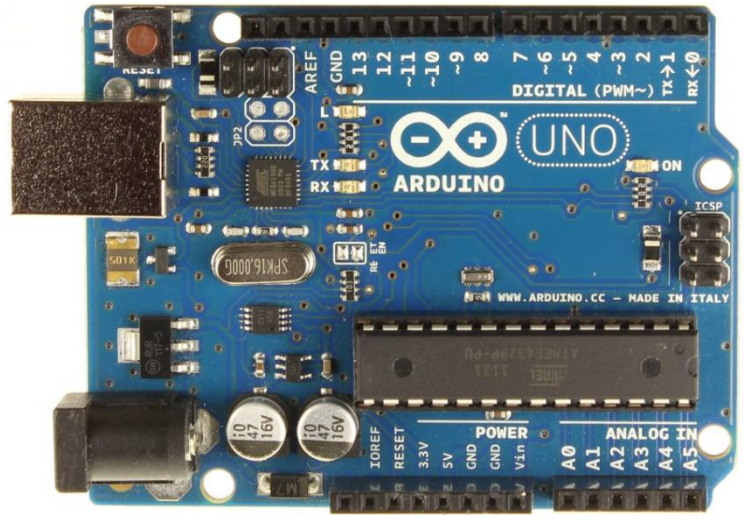
όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc.

Λόγω της καταγωγής της από την C, στην γλώσσα του Arduino μπορούν να χρησιμοποιηθούν ουσιαστικά οι ίδιες βασικές εντολές και συναρτήσεις με την ίδια σύνταξη, τους ίδιους τύπους δεδομένων και τους ίδιους τελεστές όπως και στην C. Πέρα από αυτές όμως, υπάρχουν κάποιες ειδικές εντολές, συναρτήσεις και σταθερές που βοηθούν για την διαχείριση του ειδικού hardware του Arduino.

# ΚΕΦΑΛΑΙΟ 2

## Arduino Uno

Το "Uno" σημαίνει ένα στην ιταλική γλώσσα και επιλέχθηκε για να σηματοδοτήσει την κυκλοφορία του Arduino Software (IDE) 1.0. Η πλατφόρμα Uno και η έκδοση 1.0 του λογισμικού Arduino (IDE) ήταν οι εκδόσεις αναφοράς του Arduino, που τώρα εξελίχθηκαν σε νεότερες εκδόσεις.




Εικόνα 2.1 Arduino Uno

Η πλακέτα διαθέτει μικροελεγκτή Atmel ATmega328 που λειτουργεί στα 5 V με 2 Kb μνήμης RAM, 32 Kb μνήμης flash για αποθήκευση προγραμμάτων και 1 Kb EEPROM για αποθήκευση παραμέτρων. Η ταχύτητα χρονισμού είναι 16 MHz, η οποία μεταφράζεται σε εκτέλεση περίπου 300.000 γραμμών ανά δευτερόλεπτο του πηγαίου κώδικα C. Διαθέτει 14 ψηφιακές ακίδες εισόδου / εξόδου (από τις οποίες 6 μπορούν να χρησιμοποιηθούν ως έξοδο PWM), 6 αναλογικές εισόδους, **κρυστάλλων quartz** 16 MHz, σύνδεση USB, υποδοχή τροφοδοσίας για σύνδεση με εξωτερική πηγή ώστε να εκτελείται το πρόγραμμα χωρίς να είναι συνδεδεμένο στον κεντρικό υπολογιστή, μια κεφαλή ICSP και ένα κουμπί επαναφοράς. Περιέχει όλα τα απαραίτητα για την υποστήριξη του μικροελεγκτή, απλά πρέπει να συνδεθεί με έναν υπολογιστή με καλώδιο USB ή να τροφοδοτηθεί με έναν

προσαρμογέα εναλλασσόμενου ρεύματος ή με μπαταρία για να ξεκινήσετε. Ο Uno προσφέρει τη δυνατότητα στον χρήστη να πειραματιστεί χωρίς να ανησυχεί για το αν θα κάνει κάτι λάθος.

Ενα σημαντικό χαρακτηριστικό του Arduino είναι ότι επιτρέπει στον χρήστη να δημιουργήσει ένα πρόγραμμα ελέγχου στον κεντρικό υπολογιστή. Αρκεί να το κατεβάσει στον Arduino και θα τρέξει αυτόματα. Έπειτα, αφαιρείται η σύνδεση καλωδίου USB από τον υπολογιστή και το πρόγραμμα συνεχίζει να εκτελείται από την αρχή κάθε φορά που πιέζεται το κουμπί επαναφοράς (reset). Η μπαταρία αφαιρείται και ο Arduino αποθηκεύεται για έξι μήνες. Όταν επανασυνδέσει ο χρήστης την μπαταρία, το τελευταίο πρόγραμμα που αποθηκεύτηκε θα τρέξει κανονικά.

Αυτό σημαίνει ότι ο χρήστης συνδέει την πλακέτα στον κεντρικό υπολογιστή προκειμένου να αναπτύξει και να διορθώσει το πρόγραμμά του, αλλά μόλις γίνει αυτό, δεν χρειάζεται πλέον τον υπολογιστή για να εκτελέσει το πρόγραμμα.

**Τι χρειάζεστε για ένα λειτουργικό σύστημα**

1. Πλακέτα Arduino
2. Καλώδιο προγραμματισμού USB
3. Μπαταρία 9V ή εξωτερική παροχή ρεύματος (για αυτόνομη λειτουργία)
4. Πλακέτα για εξωτερικά κυκλώματα και συμπαγές καλώδιο για συνδέσεις
5. Κεντρικό υπολογιστή που θα εκτελεί το περιβάλλον ανάπτυξης του Arduino. Υπάρχουν εκδόσεις για Windows, Mac και το Linux

## 2.1 Είσοδοι και έξοδοι

Κάθε μία από τις 14 ψηφιακές ακίδες στον Uno μπορεί να χρησιμοποιηθεί ως είσοδος ή έξοδος, χρησιμοποιώντας τις εντολές pinMode(), digitalWrite() και digitalRead(). Λειτουργούν στα 5Volt. Κάθε ακίδα μπορεί να παρέχει ή να λαμβάνει ένα μέγιστο 40mA και έχει μια εσωτερική αντίσταση pull-up (αποσυνδεδεμένη) 20-50 KOhms. Επιπλέον, μερικές ακίδες έχουν εξειδικευμένες λειτουργίες:


**Εικόνα 2.2 Μέρη της πλακέτας Arduino Uno**

- **Σειριακές: 0 (RX) και 1 (TX).** Τα pin 0 και 1 λειτουργούν ως RX και TX της σειριακής όταν το πρόγραμμα ενεργοποιεί τη σειριακή θύρα. Έτσι, όταν λόγου χάρη το πρόγραμμα στέλνει δεδομένα στη σειριακή, αυτά προωθούνται και στη θύρα USB μέσω του ελεγκτή Serial-Over-USB αλλά και στο pin 0 για να τα διαβάσει ενδεχομένως μια άλλη συσκευή (π.χ. ένα δεύτερο Arduino στο δικό του pin 1).

  Αυτό φυσικά σημαίνει ότι αν στο πρόγραμμά του ο χρήστης ενεργοποιήσει το σειριακό interface, χάνει 2 ψηφιακές εισόδους/εξόδους.
- **Εξωτερικοί διακόπτες**: **2 και 3**. Λειτουργούν και ως εξωτερικά interrupt (interrupt 0 και 1 αντίστοιχα). Πιο συγκεκριμένα, μπορεί να τα ρυθμίσει ο χρήστης μέσα από το πρόγραμμά του ώστε να λειτουργούν αποκλειστικά ως ψηφιακές είσοδοι στις οποίες όταν συμβαίνουν συγκεκριμένες αλλαγές, η κανονική ροή του προγράμματος σταματάει *άμεσα* και εκτελείται μια συγκεκριμένη συνάρτηση. Τα εξωτερικά interrupt είναι ιδιαίτερα χρήσιμα σε εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.
- **PWM: 3, 5, 6, 9, 10 και 11**. Οι ακίδες 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδοαναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation), δηλαδή το ίδιο σύστημα που διαθέτουν οι μητρικές των υπολογιστών για να ελέγχουν τις ταχύτητες των ανεμιστήρων. Έτσι, μπορεί να συνδεθεί λόγου χάρη ένα LED σε κάποιο από αυτά τα pin και να ελεγχθεί πλήρως η φωτεινότητά του με ανάλυση 8bit ,με τη χρήση της analogWrite(), (256

καταστάσεις από 0-σβηστό ως 255-πλήρως αναμμένο) αντί να δίνεται απλά η δυνατότητα αναμμένο-σβηστό που παρέχουν οι υπόλοιπες ψηφιακές έξοδοι.

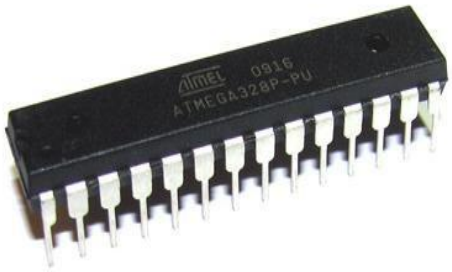
- **SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Αυτές οι ακίδες υποστηρίζουν την επικοινωνία SPI χρησιμοποιώντας τη βιβλιοθήκη SPI.
- **LED: 13**. Υπάρχει μια ενσωματωμένη λυχνία Led συνδεδεμένη στην ψηφιακή θύρα 13. Όταν ο ακροδέκτης έχει υψηλή τιμή τότε η λυχνία Led είναι αναμμένη, όταν ο ακροδέκτης έχει χαμηλή τιμή η λυχνία είναι απενεργοποιημένη.

Επιπλέον, ο Arduino διαθέτει 6 αναλογικές εισόδους, με την ένδειξη A0 έως A5, καθεμιά από τις οποίες παρέχει 10 bit ανάλυση (δηλ. 1024 διαφορετικές τιμές) 0 (όταν η τάση στο pin είναι 0V) μέχρι 1023 (όταν η τάση στο pin είναι 5V). Από προεπιλογή, μετρούν στα 5Volt, αν και είναι δυνατό να αλλάξει, χρησιμοποιώντας τον ακροδέκτη AREF και την εντολή analogReference(). Τέλος καθεμιά από τις 6 αυτές ακίδες μπορούν να μετατραπούν σε ψηφιακές ακίδες εισόδου/εξόδου όπως τα 14 που βρίσκονται στην απέναντι πλευρά και τα οποία περιγράφηκαν πριν. Σε αυτή την περίπτωση οι ακίδες μετονομάζονται από 0~5 σε 14~19 αντίστοιχα.

## 2.2 Επικοινωνία

Ο Arduino Uno διαθέτει αρκετές εγκαταστάσεις για επικοινωνία με έναν υπολογιστή, ένα άλλο Arduino, ή άλλους μικροελεγκτές.

Το ATmega328 παρέχει σειριακή επικοινωνία UART TTL (5V), η οποία είναι διαθέσιμη στις ψηφιακές ακίδες 0 (RX) και 1 (TX). Στην πλακέτα ο ATmega16U2 διοχετεύει αυτή τη σειριακή επικοινωνία μέσω USB και εμφανίζεται στο λογισμικό ως εικονική θύρα COM του υπολογιστή.




Εικόνα 2.3 Μικροελεγκτής ATmega328

Το λογισμικό '16U2 χρησιμοποιεί τα πρότυπα προγράμματα οδήγησης USB COM και δεν χρειάζεται εξωτερικό πρόγραμμα οδήγησης. Ωστόσο, στα Windows απαιτείται ένα αρχείο .inf. Το λογισμικό του Arduino περιλαμβάνει μια σειριακή οθόνη η οποία επιτρέπει, απλά δεδομένα κειμένου να σταλούν από και προς τον Arduino. Οι ενδεικτικές λυχνίες RX και TX στην πλακέτα αναβοσβήνουν όταν

μεταφέρονται δεδομένα μέσω του USB-to-serial και της σύνδεσης USB στον υπολογιστή.

Η SoftwareSerial library επιτρέπει την σειριακή επικοινωνία σε οποιαδήποτε από τις ψηφιακές ακίδες του Uno. Το ATmega328 υποστηρίζει επίσης την επικοινωνία I2C (TWI) και SPI. Το λογισμικό Arduino περιλαμβάνει μια wire library για απλοποίηση της χρήσης του διαύλου I2C. Για την επικοινωνία SPI, χρησιμοποιήστε την SPI library.

Βασικές συναρτήσεις σειριακής θύρας.

| | |
|---|---|
| begin() | αρχικοποίηση της σειριακής |
| end() | κλείσιμο της σειριακής |
| available() | έλεγχος αν υπάρχουν δεδομένα να διαβαστούν |
| read() | ανάγνωση των εισερχόμενων σειριακών δεδομένων |
| peek() | επιστρέφει το επόμενο byte από την σειριακή |
| flush() | άδειασμα του buffer της σειριακής από δεδομένα που έχει |
| print() | γράψιμο δεδομένων στη σειριακή |
| println() | γράψιμο δεδομένων στη σειριακή με αλλαγή γραμμής στο τέλος |
| write() | γράφει δυαδικά δεδομένα στη σειριακή |

## 2.3 Μνήμη

Οι πλατφόρμες Arduino διαθέτουν τρείς βασικές μνήμες:

- **Flash memory** (32 Kbytes) είναι εκεί όπου αποθηκεύεται κάθε φορά το πρόγραμμα που πρόκειται να εκτελεστεί.
- **SRAM memory** (στατική μνήμη τυχαίας προσπέλασης 2 Kbytes) είναι εκεί όπου το σκίτσο δημιουργεί και χειρίζεται μεταβλητές όταν εκτελείται.

- **EEPROM memory** (1 Kbyte) είναι χώρος μνήμης που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για την αποθήκευση μακροπρόθεσμων πληροφοριών.

Η μνήμη FLASH και η μνήμη EEPROM είναι σταθερές (οι πληροφορίες παραμένουν μετά την απενεργοποίηση της τροφοδοσίας). Η μνήμη SRAM είναι ασταθής και πληροφορίες χάνονται όταν κυκλοφορήσει το ρεύμα.

Το τσιπ ATmega328 που βρίσκεται στον Uno έχει τα εξής ποσά μνήμης:

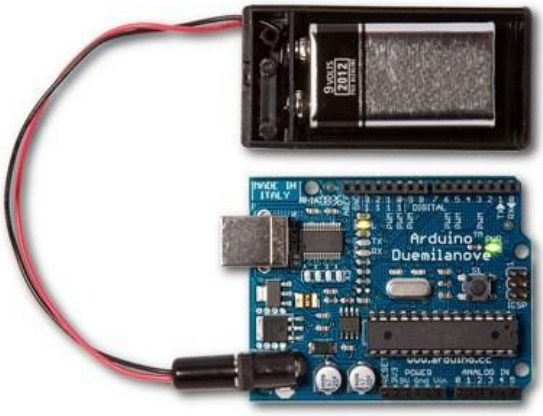
- Flash 32k byte (από τα οποία το .5k χρησιμοποιείται για το bootloader)
- SRAM 2k bytes
- EEPROM 1k byte

Αν εξαντληθεί η μνήμη SRAM, το πρόγραμμα ενδέχεται να αποτύχει με απρόβλεπτο τρόπο. Θα εμφανιστεί να φορτώνει με επιτυχία, αλλά να μην τρέχει ή να τρέχει παράξενα. Για να ελεγχθεί αν συμβαίνει αυτό, μπορούν να μειωθούν τα σχόλια ή οι συμβολοσειρές ή άλλες δομές δεδομένων στο sketch (χωρίς να αλλάξει ο κώδικας). Εάν στη συνέχεια τρέχει με επιτυχία, πιθανότατα είχε εξαντληθεί η μνήμη SRAM. Ενας τρόπος για να αντιμετωπιστεί αυτό το πρόβλημα είναι να μεταφερθούν δεδομένα ή υπολογισμοί στον υπολογιστή μειώνοντας έτσι το φορτίο στον Arduino. Εάν υπάρχουν πίνακες αναζήτησης ή άλλες μεγάλες συστοιχίες, μπορεί να χρησιμοποιηθεί ο μικρότερος τύπος δεδομένων που απαιτείται για την αποθήκευση των τιμών που χρειάζονται.

## 2.4 Τροφοδοσία

Ο Arduino Uno μπορεί να τροφοδοτηθεί μέσω σύνδεσης USB ή με εξωτερικό τροφοδοτικό. Η επιλογή της πηγής γίνεται αυτόματα. Η εξωτερική (μη-USB) ισχύς μπορεί να προέρχεται είτε από προσαρμογέα εναλλασσόμενου ρεύματος (DC) είτε από μπαταρία. Ο προσαρμογέας μπορεί να συνδεθεί με ένα βύσμα 2,1 mm στην υποδοχή τροφοδοσίας της πλακέτας. Επίσης, οι ακροδέκτες από μια μπαταρία θα μπορούσαν να τοποθετηθούν στις υποδοχές Gnd και Vin της πλακέτας


Εικόνα 2.4 Μετασχηματιστής




Εικόνα 2.5 Σύνδεση με μπαταρία 9V

Η πλακέτα μπορεί να λειτουργεί σε εξωτερική τροφοδοσία 6 έως 20Volt. Εάν τροφοδοτούνται λιγότερα από 7V, ο ακροδέκτης 5V μπορεί να παρέχει λιγότερα από 5Volt και ο πίνακας ενδέχεται να γίνει ασταθής. Εάν χρησιμοποιηθούν περισσότερα από 12V, ο ρυθμιστής τάσης μπορεί να υπερθερμανθεί και να προκαλέσει βλάβη στην πλακέτα. Το συνιστώμενο εύρος τιμών κυμαίνεται από 7 έως 12Volt.

Οι ακίδες ισχύος είναι οι εξής:

- **VIN**. Είναι η τάση εισόδου στην πλακέτα Arduino όταν χρησιμοποιείται μια εξωτερική πηγή τροφοδοσίας (σε αντίθεση με τα 5Volt από τη σύνδεση USB ή από άλλη πηγή τροφοδοσίας). Μπορεί να τροφοδοτηθεί τάση μέσω αυτής της υποδοχής ή εάν τροφοδοτείται με τάση μέσω της υποδοχής τροφοδοσίας, μπορούμε να έχουμε πρόσβαση μέσω αυτής της εισόδου.

- **5V**. Αυτός ο ακροδέκτης δίνει στην έξοδο 5V από τον ρυθμιστή της πλακέτας. Η πλακέτα μπορεί να τροφοδοτηθεί με ισχύ από την υποδοχή τροφοδοσίας DC (7 - 12V), τη θύρα USB (5V) ή από τον ακροδέκτη VIN της πλακέτας (7-12V). Η παροχή τάσης μέσω των ακροδεκτών 5V ή 3.3V παρακάμπτει τον ρυθμιστή και μπορεί να βλάψει την πλακέτα. Δεν το συνιστούμε.
- **3V3**. Μια τροφοδοσία 3,3Volt που παράγεται από τον ρυθμιστή. Η μέγιστη ροή ρεύματος είναι 50 mA.
- **GND**. Ακροδέκτες γείωσης.
- **Reset**: Όταν γειωθεί έχει σαν αποτέλεσμα την επανεκκίνηση του Arduino.

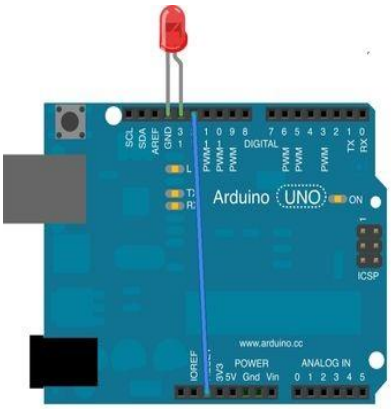
## 2.5 Προστασία υπερέντασης USB

Όσον αφορά τις θύρες USB του υπολογιστή ο Arduino Uno διαθέτει προστασία από βραχυκύκλωμα και υπερφόρτωση, αν και οι περισσότεροι υπολογιστές παρέχουν τη δική τους εσωτερική προστασία. Εάν εφαρμοστούν περισσότερα από 500 mA στη θύρα USB, η ασφάλεια θα κόψει αυτόματα τη σύνδεση μέχρι να διορθωθεί το βραχυκύκλωμα ή η υπερφόρτωση.

## 2.6 Αυτόματη Επαναφορά συστήματος

Ο Arduino Uno είναι σχεδιασμένος με τέτοιο τρόπο που να επιτρέπει την επαναφορά του από το λογισμικό που εκτελείται στον συνδεδεμένο υπολογιστή χωρίς να απαιτείται χρήση του πλήκτρου επαναφοράς πριν από τη μεταφόρτωση. Μια από τις γραμμές ελέγχου ροής υλικού (DTR) του ATmega8U2 / 16U2 συνδέεται στη γραμμή επαναφοράς του ATmega328 μέσω πυκνωτή 100 nanofarad. Όταν αυτή η γραμμή δείξει (Low), η γραμμή επαναφοράς πέφτει για να επαναφέρει το τσιπ. Το λογισμικό του Arduino χρησιμοποιεί αυτή τη δυνατότητα για να επιτρέψει τη μεταφόρτωση του κώδικα, πατώντας απλά το κουμπί φόρτωσης, στο περιβάλλον του Arduino.

Υπάρχουν δυο τρόποι επαναφοράς συστήματος.




1. **Ηλεκτρονικά**, χρησιμοποιώντας ένα καλώδιο που συνδέει έναν ακροδέκτη output με τον ακροδέκτη Reset. Σε αυτό το παράδειγμα, ο ακροδέκτης 13, που συνδέεται με τον εσωτερικό πείρο Led 13, αναβοσβήνει. Ο ακροδέκτης 12 συνδέεται με τον ακροδέκτη Reset με ένα καλώδιο. Στη συνάρτηση setup (), το πρώτο πράγμα που κάνουμε είναι να γράψουμε HIGH στον ακροδέκτη 12, ο οποίος καλείται ακίδα επαναφοράς (digitalWrite (resetPin, HIGH), κάνοντας έτσι τον ακροδέκτη Reset HIGH.
2. **Με χρήση λογισμικού**. Σε αυτό το παράδειγμα, δεν χρειάζεται επιπλέον καλωδίωση, αρκεί η Αρχικοποίηση της λειτουργίας επαναφοράς και κατόπιν η κλήση της ώστε να γίνει επαναφορά.

void (* resetFunc) (void) = 0; (όπου δηλώνει τη λειτουργία επαναφοράς
στη διεύθυνση 0)

.....

resetFunc (); ( καλούμε να γίνει επαναφορά)

## 2.7 Γλώσσα προγραμματισμού

Η γλώσσα του Arduino βασίζεται στη γλώσσα Wiring, μια παραλλαγή C/C++ για μικροελεγκτές αρχιτεκτονικής AVR όπως ο ATmega, και υποστηρίζει όλες τις βασικές δομές της C καθώς και μερικά χαρακτηριστικά της C++. Για compiler χρησιμοποιείται ο AVR gcc και ως βασική βιβλιοθήκη C χρησιμοποιείται η AVR libc.

Στην γλώσσα του Arduino κάθε πρόγραμμα αποτελείται από δύο βασικές ρουτίνες ώστε να έχει την γενική δομή:

```
// Ενσωματώσεις βιβλιοθηκών, δηλώσεις μεταβλητών...
void setup()
{
  // ...
}
void loop()
{
  // ...
}
// Υπόλοιπες συναρτήσεις...
```

Η void setup() εκτελείται μια φορά μόνο κατά την εκκίνηση του προγράμματος ενώ η void loop() περιέχει τον βασικό κορμό του προγράμματος και η εκτέλεσή της επαναλαμβάνεται συνέχεια.

## 2.8 Εγκατάσταση του προγράμματος

Ο Arduino Uno μπορεί να προγραμματιστεί με το λογισμικό Arduino IDE. Το ATmega328 στον Arduino Uno έρχεται προεγκατεστημένο με ένα bootloader που επιτρέπει να ανεβεί ο κώδικας χωρίς τη χρήση εξωτερικού προγράμματος.

Μπορεί επίσης να παρακάμψει τον bootloader και να γίνει ο προγραμματισμός του μικροελεγκτή μέσω του ICSP (In-Circuit Serial Programming).

Arduino IDE και σύνδεση με τον υπολογιστή.

Ό, τι είναι απαραίτητο για την διαχείριση του Arduino από τον υπολογιστή το παρέχει το Arduino IDE, η τελευταία έκδοση του οποίου υπάρχει διαθέσιμη προς λήψη για καθένα από τα τρία δημοφιλέστερα λειτουργικά συστήματα στο επίσημο site του Arduino.

Το Arduino IDE είναι βασισμένο σε Java και συγκεκριμένα παρέχει:

- ένα πρακτικό περιβάλλον για την συγγραφή των προγραμμάτων σας (τα οποία ονομάζονται sketch στην ορολογία του Arduino) με συντακτική χρωματική σήμανση,
- αρκετά έτοιμα παραδείγματα,
- μερικές έτοιμες βιβλιοθήκες για προέκταση της γλώσσας και για να χειρίζεστε εύκολα μέσα από τον κώδικά σας τα εξαρτήματα που συνδέετε στο Arduino,
- τον compiler για την μεταγλώττιση των sketch σας,
- το serial monitor που είναι ένα αναδυόμενο παράθυρο που λειτουργεί ως ξεχωριστό τερματικό και απεικονίζει τη λήψη και αποστολή σειριακών δεδομένων στον Arduino.
- και την επιλογή να ανεβεί το μεταγλωττισμένο sketch στο Arduino.

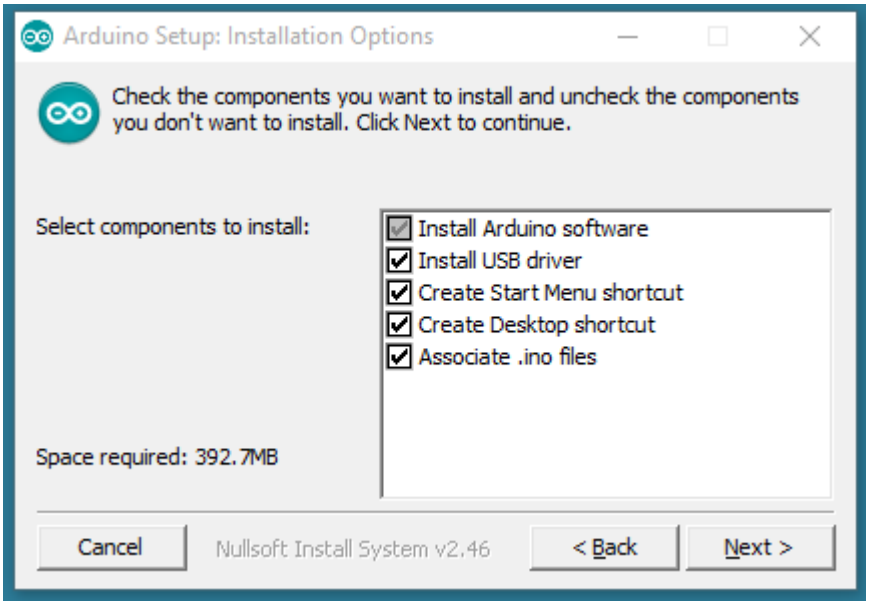
Για τα δύο τελευταία χαρακτηριστικά βέβαια, ο Arduino πρέπει να έχει συνδεθεί σε μια από τις θύρες USB του υπολογιστή και, λόγω του ελεγκτή Serial-over-USB, θα πρέπει να αναγνωριστεί από το λειτουργικό σύστημα ως εικονική σειριακή θύρα.

Για την σύνδεση είναι απαραίτητο ένα καλώδιο USB από Type A σε Type B, όπως αυτό των εκτυπωτών. Για την αναγνώριση από το λειτουργικό θα χρειαστεί η εγκατάσταση του οδηγού του FTDI chip (δηλαδή του ελεγκτή Serial-over-USB) ο οποίος υπάρχει στον φάκελο drivers του Arduino IDE. Η τελευταία έκδοση αυτού του οδηγού για κάθε λειτουργικό σύστημα από το site της FTDI.

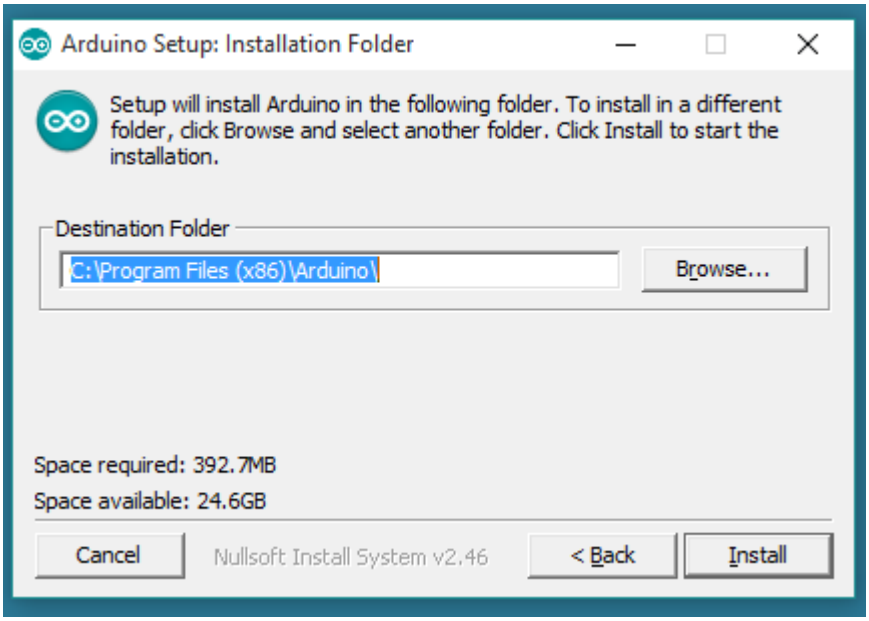
Λήψη του λογισμικού Arduino (IDE).

Αποκτήστε την πιο πρόσφατη έκδοσης από τη σελίδα λήψης . Υπάρχει δυνατότητα επιλογής μεταξύ του πακέτου Installer (.exe) και του Zip. Προτείνεται η χρήση της πρώτης που εγκαθιστά άμεσα όλα όσα χρειάζονται για τη χρήση του λογισμικού Arduino (IDE), συμπεριλαμβανομένων των προγραμμάτων οδήγησης. Με το πακέτο Zip θα πρέπει να γίνει εγκατάσταση των προγραμμάτων οδήγησης με μη αυτόματο τρόπο. Το αρχείο Zip είναι επίσης χρήσιμο στο να δημιουργηθεί μια φορητή εγκατάσταση .

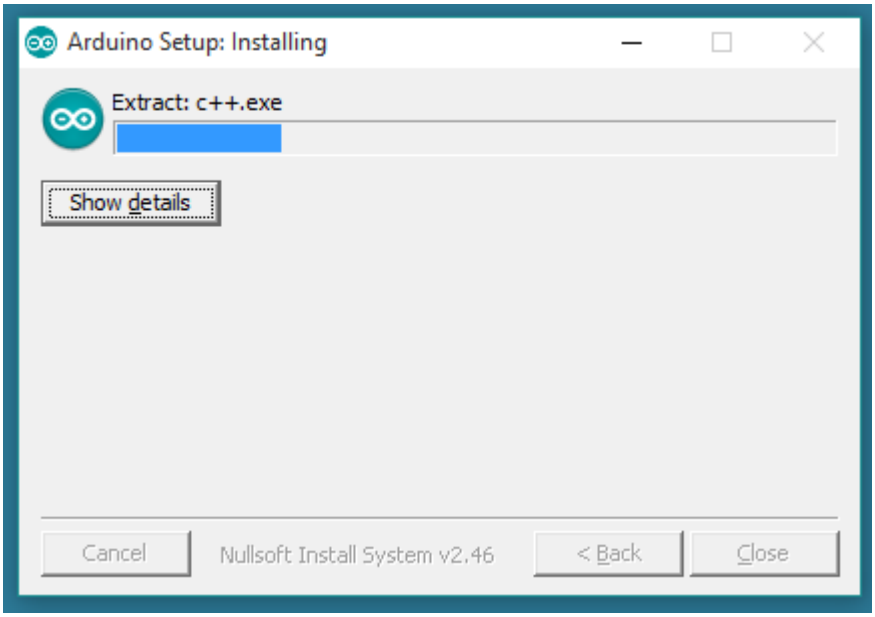
Μετά την ολοκλήρωση της λήψης, σειρά έχει η εγκατάσταση αρκεί όταν εμφανιστεί ειδοποίηση από το λειτουργικό σύστημα, να γίνει αποδεκτή η διαδικασία εγκατάστασης του προγράμματος οδήγησης.




Γίνεται επιλογή των στοιχείων που θα εγκατασταθούν




Επιλογή του προορισμού εγκατάστασης.




Η διαδικασία θα εξάγει και θα εγκαταστήσει όλα τα απαιτούμενα αρχεία για να εκτελέσει σωστά το λογισμικό Arduino (IDE)

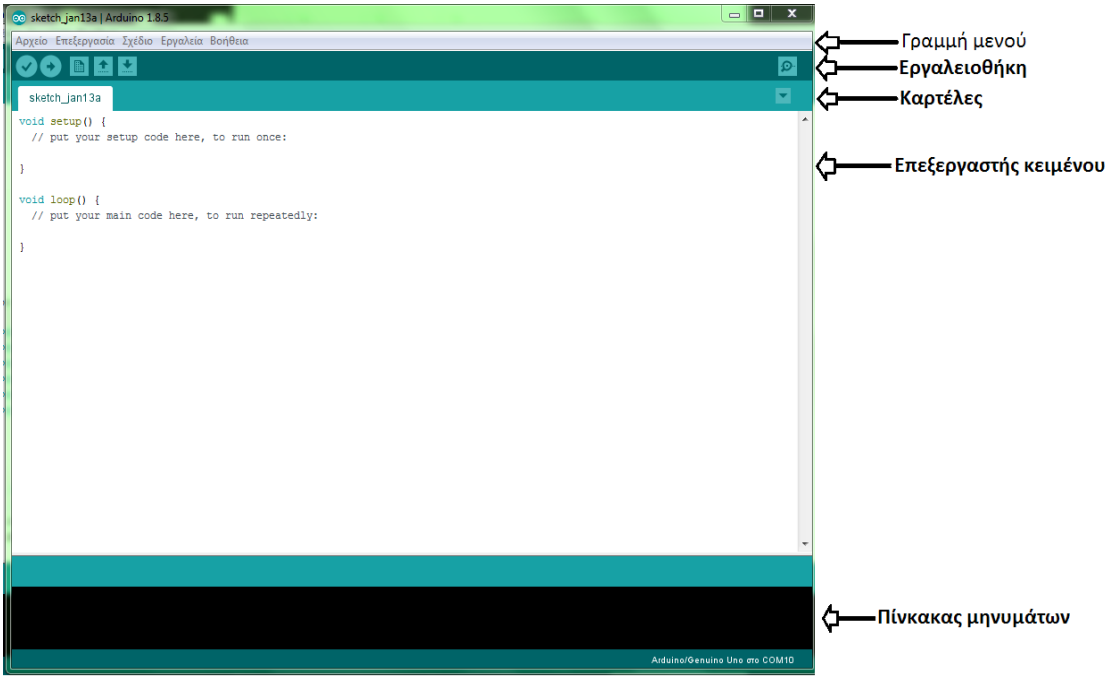
Αν όλα έγιναν σωστά, όταν γίνει η εκτέλεση του προγράμματος θα εμφανιστεί το κεντρικό παράθυρο του Arduino IDE και στο μενού Tools –> Serial Port θα πρέπει να εμφανίζεται η εικονική σειριακή θύρα (συνήθως COM# για τα Windows, /dev/ttyusbserial## για το MacOS και /dev/ttyusb## για το Linux).

Γίνεται η επιλογή της εικονικής θύρας και στην συνέχεια η επιλογή του τύπου πλακέτας του Arduino από το μενού Tools–>Board. Το Arduino είναι πλέον έτοιμο να δεχτεί τα sketch. Αν εμφανίστηκε οποιοδήποτε πρόβλημα υπάρχουν αναλυτικές οδηγίες εγκατάστασης για κάθε λειτουργικό σύστημα στη διεύθυνση http://arduino.cc/en/Guide/HomePage.

## 2.9 Περιβάλλον ανάπτυξης του Arduino

Το λογισμικό Arduino Integrated Development Environment ή αλλιώς Arduino (IDE) περιέχει ένα πρόγραμμα επεξεργασίας κειμένου για την εγγραφή κώδικα, μια περιοχή μηνυμάτων, μια κονσόλα κειμένου, μια γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες και μια σειρά από μενού. Συνδέεται με το υλικό Arduino και Genuino για τη μεταφόρτωση προγραμμάτων και την επικοινωνία μαζί τους.

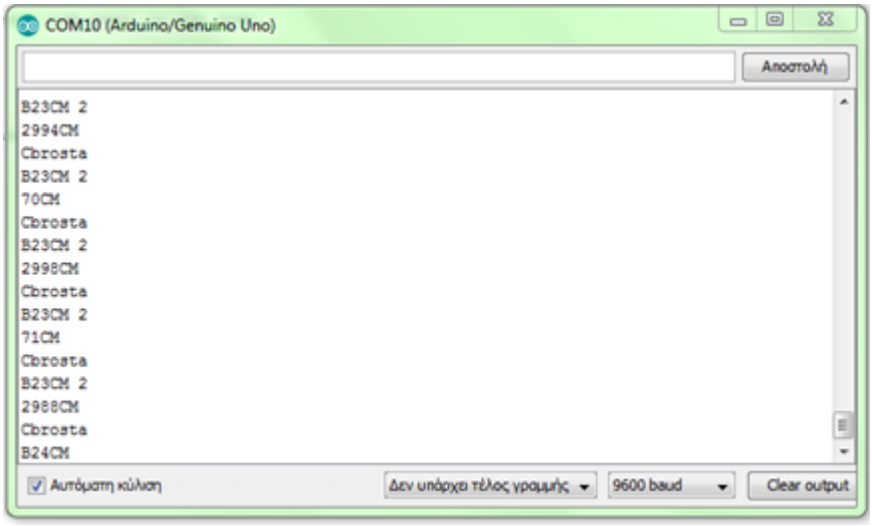
Τα προγράμματα που γράφονται χρησιμοποιώντας το λογισμικό Arduino (IDE) ονομάζονται sketch. Αυτά τα sketch γράφονται στον επεξεργαστή κειμένου και αποθηκεύονται με την επέκταση αρχείου .ino. Ο επεξεργαστής διαθέτει χαρακτηριστικά για την αντιγραφή/επικόλληση και την αναζήτηση/αντικατάσταση κειμένου. Η περιοχή μηνυμάτων παρέχει, ανατροφοδότηση κατά την αποθήκευση και εξαγωγή και εμφανίζει επίσης σφάλματα. Η κονσόλα εμφανίζει την έξοδο κειμένου από το λογισμικό Arduino (IDE), συμπεριλαμβανομένων πλήρων μηνυμάτων σφάλματος και άλλων πληροφοριών. Στην κάτω δεξιά γωνία του παραθύρου εμφανίζεται η διαμορφωμένη πλακέτα και η σειριακή θύρα. Τα κουμπιά της γραμμής εργαλείων επιτρέπουν την επαλήθευση και την φόρτωση των προγραμμάτων, την δημιουργία νέου sketch, το άνοιγμα και την αποθήκευση των sketch και το άνοιγμα της σειριακής οθόνης.




**Εικόνα 2.6 Περιβάλλον ανάπτυξης Arduino IDE**

Κουμπιά γραμμής εργαλείων:

-  Επικύρωση: Ελέγχει για τυχόν λάθη στον κώδικα.
-  Ανέβασμα: Ανεβάζει τον κώδικα στον μικροελεγκτή.
-  Δημιουργία: Δημιουργεί ένα νέο sketch.
-  Άνοιγμα: Ανοίγει το μενού μ όλα τα αποθηκευμένα sketch.
-  Αποθήκευση: Αποθηκεύει το sketch.
-  Σειριακή οθόνη: Ανοίγει την σειριακή οθόνη ώστε να φαίνονται τα δεδομένα που δίνει και παίρνει ο Arduino.




Εικόνα 2.7 Σειριακή θύρα επικοινωνίας.

## 2.10 Δομή και λειτουργίες προγραμματισμού.

Δομές ελέγχου

| | |
|---|---|
| **break** | **εντολή διακοπής μιας επαναληπτικής δομής** |
| **continue** | εντολή παράλειψης της τρέχουσας επανάληψης |
| **do...while** | δομή επαναληπτικού ελέγχου συνθήκης |
| **else** | δομή μεγαλύτερου έλεγχου ροής κώδικα |
| **for** | δομή επαναληπτικού ελέγχου συνθήκης |
| **goto** | εντολή μετάβασης σε κάποιο σημείο του κώδικα |
| **if...else** | δομή ελέγχου πολλαπλών συνθηκών |
| **if** | δομή ελέγχου μίας συνθήκης |
| **switch...case** | δομή ελέγχου περιπτώσεων |
| **while** | δομή επαναληπτικού ελέγχου συνθήκης |
| **return** | εντολή επιστροφής από μία συνάρτηση |

Αριθμητικοί χειριστές

| | |
|---|---|
| **%** | υπόλοιπο ακεραίας διαίρεσης |
| ***** | πολλαπλασιασμού |
| **+** | πρόσθεσης |
| **-** | αφαίρεσης |
| **/** | διαίρεσης |
| **=** | εκχώρησης |

Τελεστές σύγκρισης

| | |
|---|---|
| **!=** | ανισότητα |
| **<** | μικρότερο |
| **<=** | μικρότερο ή ίσο |
| **==** | ισότητα |
| **>** | μεγαλύτερο |
| **>=** | μεγαλύτερο ή ίσο |

Λογικοί τελεστές

| | |
|---|---|
| **!** | λογική άρνηση |
| **&&** | λογική σύζευξη |
| **\|\|** | λογική διάζευξη |

Χειριστές δεικτών

| | |
|---|---|
| **&** | απόκτησης διεύθυνσης |
| ***** | απόκτησης περιεχομένου |

Δυαδικοί χειριστές

| | |
|---|---|
| **&** | σύζευξης |
| **<<** | αριστερής ολίσθησης |
| **>>** | δεξιάς ολίσθησης |
| **^** | αποκλειστικής διάζευξης |
| **\|** | διάζευξης |
| **~** | άρνησης |

Σύνθετοι χειριστές

| | |
|---|---|
| **&=** | σύνθετο και |
| ***=** | σύνθετος πολλαπλασιασμός |
| **++** | προσαύξησης |
| **+=** | προσθήκη ένωσης |
| **-** | μείωσης |
| **-=** | αφαίρεση ένωσης |
| **/=** | σύνθετης διαίρεσης |
| **\|** | σύνθετο ή |

Σταθερές

| | |
|---|---|
| **HIGH** | τιμή υψηλή (5V) για επαφή εισόδου ή εξόδου |
| **LOW** | τιμή χαμηλή (0V) για επαφή εισόδου ή εξόδου |
| **INPUT** | ορισμό επαφής σαν είσοδο |
| **OUTPUT** | ορισμό επαφής σαν έξοδο |
| **true** | επίπεδο αληθείας σε μια συνθήκη |
| **false** | επίπεδο ψεύδους σε μια συνθήκη |

Τύποι εντολών και συναρτήσεων που χρησιμοποιούνται

| Όρισμα | Είδος | Τύπος | Παράμετροι | Περιγραφή |
|---|---|---|---|---|
| **pinmode** | Εντολή | – | (*pin*, *mode*) | Καθορίζει αν το συγκεκριμένο ψηφιακό *pin* θα είναι pin εισόδου ή pin εξόδου ανάλογα με την τιμή που δίνεται στην παράμετρο *mode*(INPUT ή OUTPUT αντίστοιχα). |
| **digitalWrite** | Εντολή | – | (*pin*, *pinstatus*) | Θέτει την κατάσταση *pinstatus*(HIGH ή LOW) στο συγκεκριμένο ψηφιακό *pin*. |
| **digitalRead** | Συνάρτηση | int | (*pin*) | Επιστρέφει την κατάσταση του συγκεκριμένου ψηφιακού *pin* (0 για LOW και 1 για HIGH) εφόσον αυτό είναι pin εισόδου. |
| **analogReference** | Εντολή | – | (*type*) | Δέχεται τις τιμές DEFAULT, INTERNAL ή EXTERNAL στην παράμετρο *type* για να καθορίσει την τάση αναφοράς (Vref) των αναλογικών εισόδων (5V, 1.1V ή η εξωτερική τάση με την οποία τροφοδοτείται το pin AREF αντίστοιχα) |
| **aalogRead** | Συνάρτηση | int | (*pin*) | Επιστρέφει έναν ακέραιο από 0 εώς 1023, ανάλογα με την τάση που τροφοδοτείται το συγκεκριμένο *pin* αναλογικής εισόδου στην κλίμακα 0 ως Vref. |
| **analogWrite** | Εντολή | – | (*pin*, *value*) | Θέτει το συγκεκριμένο ψηφιακό *pin* σε κατάσταση ψευδοαναλογικής εξόδου |

| | | | | |
|---|---|---|---|---|
| | | | | (PWM). Η παράμετρος *value* καθορίζει το πλάτος του παλμού σε σχέση με την περίοδο του παραγόμενου σήματος στην κλίμακα από 0 ως 255 (π.χ. με *value* 127, το πλάτος του παλμού είναι ίσο με μισή περίοδο). |
| **millis** | Συνάρτηση | unsigned long | () | Μετρητής που επιστρέφει το χρονικό διάστημα σε ms από την στιγμή που άρχισε η εκτέλεση του προγράμματος. Λάβετε υπόψη ότι λόγω του τύπου μεταβλητής (unsigned long δηλ. 32bit) θα γίνει overflow σε 2^32ms δηλαδή περίπου σε 50 μέρες, οπότε ο μετρητής θα ξεκινήσει πάλι από το μηδέν. |
| **delay** | Εντολή | – | (*time*) | Σταματά προσωρινά την ροή του προγράμματος για *time* ms. Η παράμετρος *time* είναι unsigned long (από 0 ως 2^32). |
| **attachInterrupt** | Εντολή | – | (*interrupt, function,triggermode*) | Θέτει σε λειτουργία το συγκεκριμένο *interrupt*, ώστε να ενεργοποιεί την συνάρτηση *function*, κάθε φορά που ικανοποιείται η συνθήκη που ορίζεται από την παράμετρο *triggermode*:<br>LOW (ενεργοποίηση όταν η κατάσταση του pin που αντιστοιχεί στο συγκεκριμένο interrupt γίνει LOW)<br>RISING (όταν από LOW γίνει HIGH)<br>FALLING (όταν από HIGH γίνει LOW) |

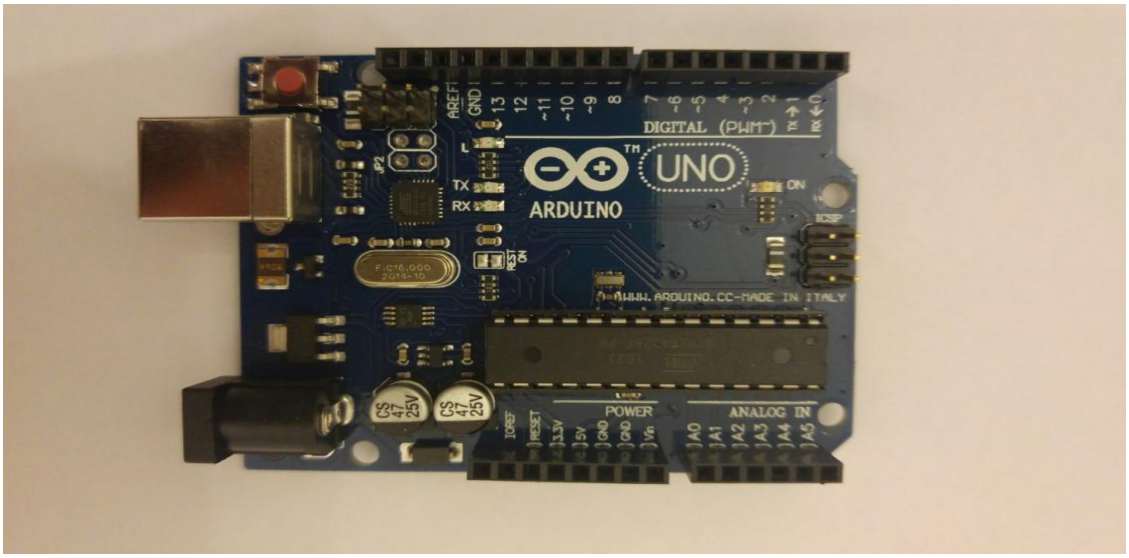
| | | | |
|---|---|---|---|
| | | | CHANGE (όταν αλλάξει κατάσταση γενικά) |
| **detachInterrupt** | Εντολή | – | Απενεργοποιεί το συγκεκριμένο *interrupt*. |
| **noInterrupts** | Εντολή | – | Σταματά προσωρινά την λειτουργία όλων των interrupt |
| **interrupts** | Εντολή | – | Επαναφέρει την λειτουργία των interrupt που διακόπηκε προσωρινά από μια εντολή noInterrupts. |
| **Serial.begin** | Μέθοδος κλάση | – | Θέτει τον ρυθμό μεταφοράς δεδομένων του σειριακού interface (σε baud) |
| **Serial.println** | Μέθοδος κλάση | – | Διοχετεύει τα δεδομένα *data* για αποστολή μέσω του σειριακού interface. Η παράμετρος *data*μπορεί να είναι είτε αριθμός είτε αλφαριθμητικό. |

# ΚΕΦΑΛΑΙΟ 3

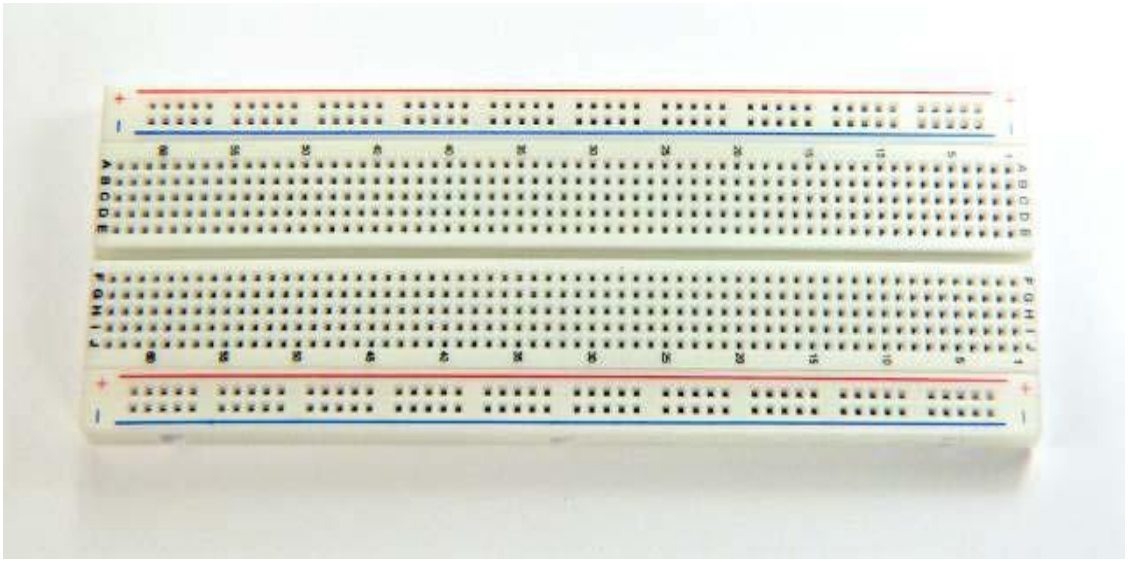
## Υλικά κατασκευής

Σε αυτό το κεφάλαιο περιγράφονται τα απαραίτητα υλικά που χρησιμοποιούνται για την ολοκλήρωση του πρακτικού μέρους αυτής της πτυχιακής εργασίας.

**1. Arduino Uno**: Ο κεντρικός επεξεργαστής από τον οποίο ξεκινάει ο έλεγχος του αυτοκινήτου.



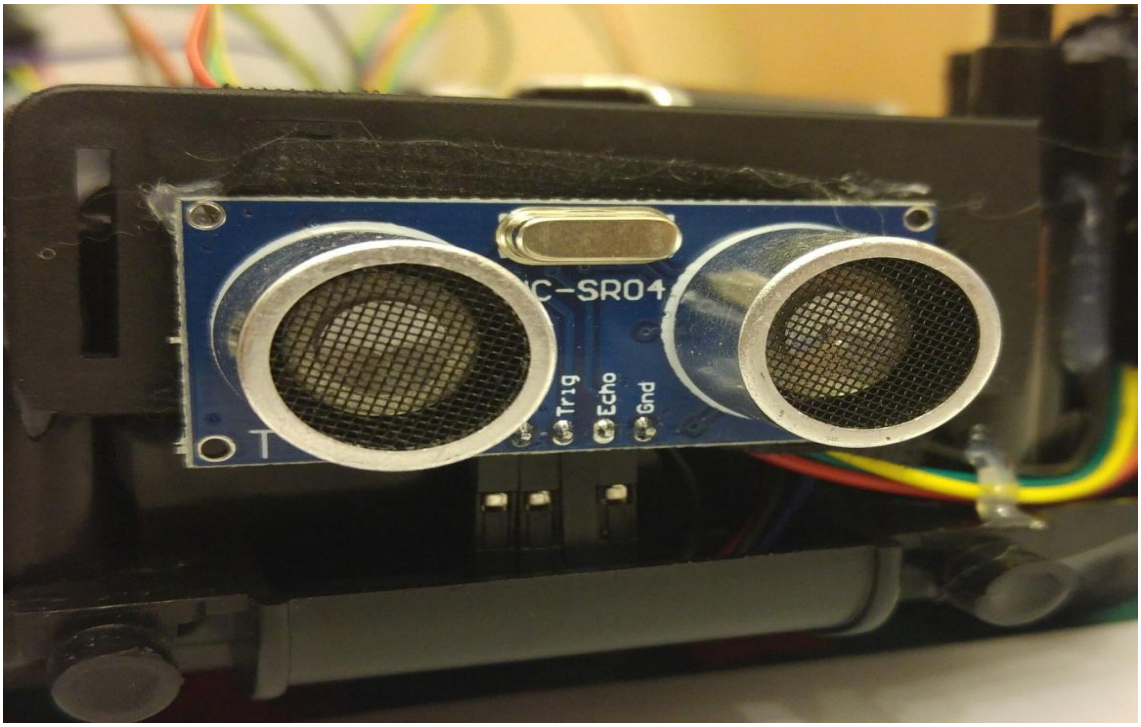

**2. Breadboard**: Πάνω σε αυτή την πλακέτα φτιάχνουμε το κύκλωμά μας στο στάδιο της έρευνας και της ανάπτυξης του συστήματός μας.



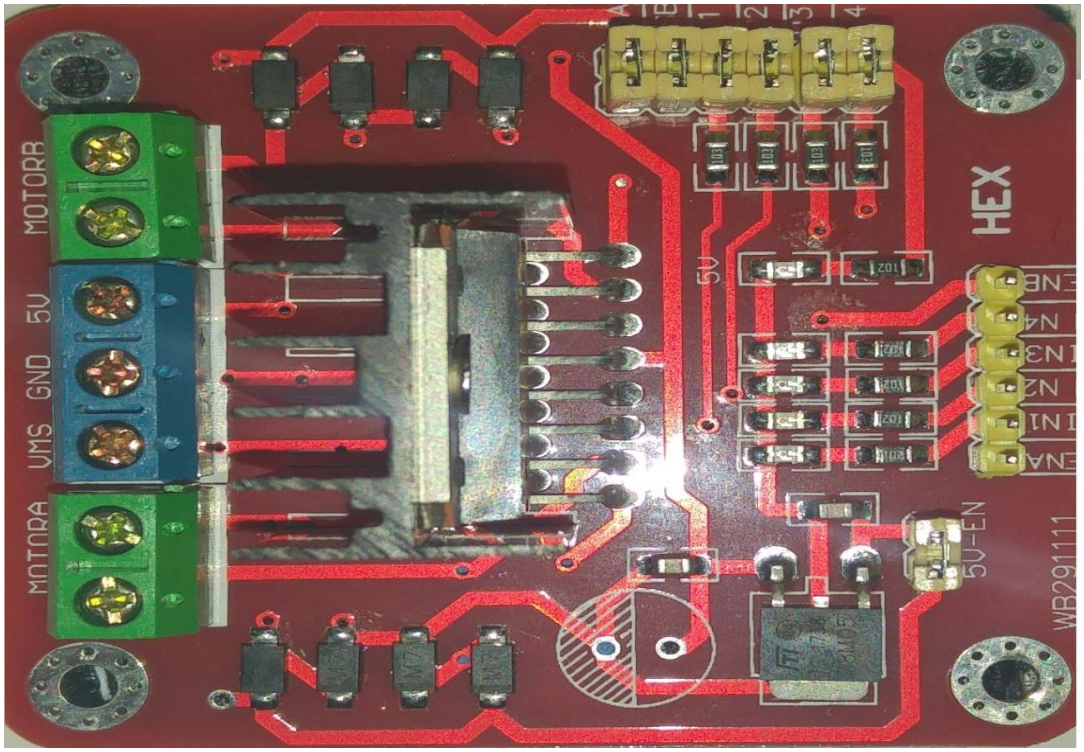
Λειτουργικά ένα breadboard μπορεί να βραχυκυκλώνει μεταξύ τους καλώδια, κάτι που είναι πολύ χρήσιμο όταν έχουμε πολλά καλώδια για βραχυκύκλωση διαφορετικά, άλλωστε δεν υπάρχουν τόσες ελεύθερες είσοδοι ή έξοδοι στην πλακέτα μας.

Οι οριζόντιες γραμμές «+» και «-» σε κάθε μεριά (πάνω και κάτω, με κόκκινο και μπλε χρώμα όπως την βλέπουμε) είναι βραχυκυκλωμένες μεταξύ τους, ενώ στις στήλες (που είναι συνήθως αριθμημένες από το 1 μέχρι το 30) είναι βραχυκυκλωμένες οι πέντε κάθετες υποδοχές (συνήθως με γράμματα a, b, c, d, e καθώς και f, g, h, i, j) μεταξύ τους σε κάθε στήλη. Έτσι, για παράδειγμα μπορούμε να συνδέσουμε στο «–» το Gnd του Arduino και στο breadboard να συνδέουμε όλες τις επιστροφές Gnd των κυκλωμάτων μας.

**3.HC-SR04 Ultrasonic Sensor**: Ο αισθητήρας απόστασης ο οποίος διαβάζει την απόσταση και στέλνει τις τιμές στον κεντρικό επεξεργαστή.

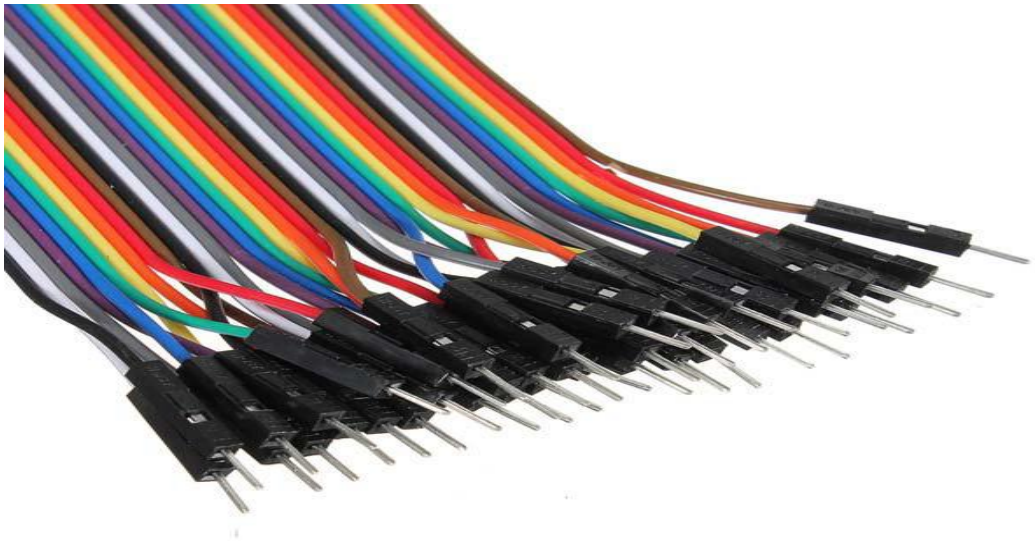


**4. H-Bridge L298N**: Η πλακέτα η οποία ελέγχει πλήρως τους ηλεκτροκινητήρες όσον αφορά την κατεύθυνση και την ταχύτητα περιστροφής τους.




**5. Ηλεκτροκινητήρας**: Ο κινητήρας που χρησιμοποιείται για την κίνηση και την κατεύθυνση του αυτοκινήτου.

**6. Καλώδια**: Είναι οι αγωγοί που οδηγούν το ρεύμα και κατ' επέκταση τις εντολές από τον Arduino στις συσκευές.

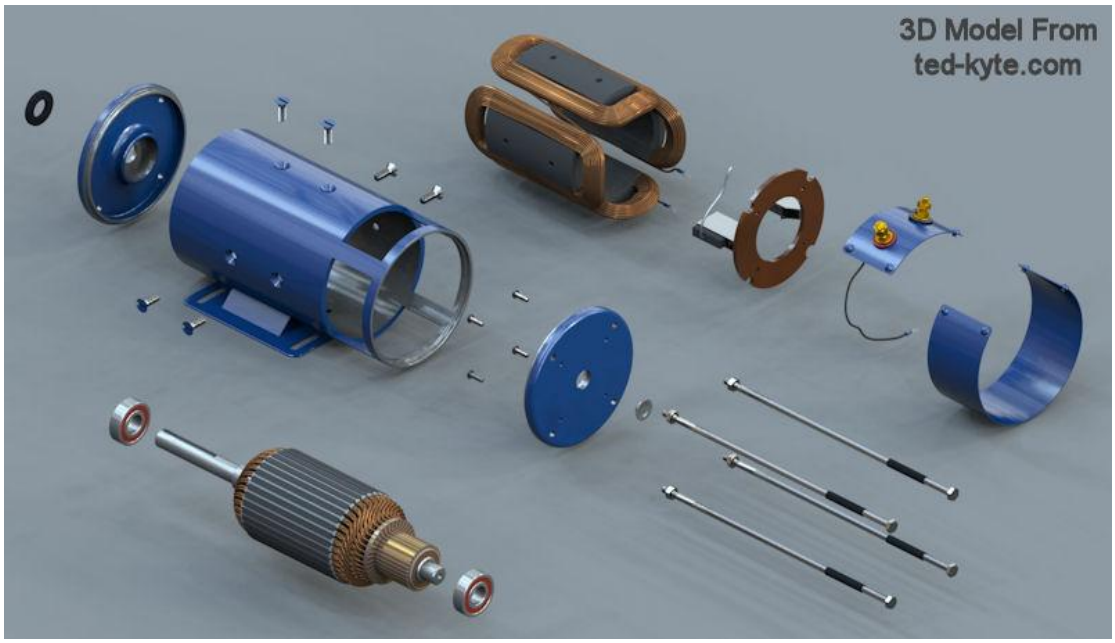


**7. Καλώδιο τροφοδοσίας USB A to B**: Είναι το καλώδιο που χρησιμοποιείται για την τροφοδοσία του Arduino αλλά και για τον προγραμματισμό του.

**8. Αλκαλικές μπαταρίες**: Χρησιμοποιούνται για την τροφοδοσία του κυκλώματος αλλά και για την ανεξάρτητη τροφοδοσία του Arduino.


## 3.1 DC ΚΙΝΗΤΗΡΕΣ

Ο πρώτος πρακτικός κινητήρας DC (Direct Current) εφευρέθηκε από τον Βρετανό επιστήμονα William Sturgeon το 1832. Από τότε οι κινητήρες συνεχούς ρεύματος αποτελούσαν μέρος αναρίθμητου εξοπλισμού και μηχανημάτων. Η γκάμα των κινητήρων συνεχούς ρεύματος είναι ευρεία και αποτελείται από τεράστια μοντέλα που χρησιμοποιούνται στον βιομηχανικό εξοπλισμό έως μικροσκοπικές συσκευές που μπορούν να χωρέσουν στην παλάμη του χεριού μας. Οι μικρής κλίμακας κινητήρες είναι οικονομικοί και αυτό τους καθιστά ιδανικούς για χρήση σε project ρομποτικής, κινούμενων κατασκευών και πολλών ακόμη εργασιών.




Σε αντίθεση με τα LED δεν γίνεται απλά να συνδεθεί ένας κινητήρας DC σε έναν από τους ακροδέκτες εξόδου του Arduino και να αναμένεται να λειτουργήσει. Οι κινητήρες DC έχουν απαιτήσεις ρεύματος και τάσης που είναι πέρα από τις δυνατότητες του μικροελεγκτή ή του μικροϋπολογιστή. Είναι απαραίτητο να χρησιμοποιηθούν κάποια εξωτερικά ηλεκτρονικά για να οδηγηθεί και να ελεγχθεί ο κινητήρας, και πιθανότατα θα χρειαστεί και ένα ξεχωριστό τροφοδοτικό.

Υπάρχουν διάφοροι τρόποι για τον χειρισμό ενός μοτέρ DC από την έξοδο του υπολογιστή. Όταν υπάρχει ανάγκη ελέγχου μόνο της ταχύτητας περιστροφής του κινητήρα τότε αυτός μπορεί να οδηγηθεί χρησιμοποιώντας ένα και μόνο

τρανζίστορ. Στην περίπτωση όμως που εκτός από την ταχύτητα χρειάζεται να ελεγχθεί και η κατεύθυνση περιστροφής δεν επαρκεί αυτός ο τρόπος. Ένα πιο ευέλικτο σύστημα ελέγχου ενός κινητήρα συνεχούς ρεύματος είναι με τη χρήση ενός κυκλώματος που ονομάζεται "H Bridge".

Η "H Bridge" αποτελείται από διάταξη τρανζίστορ που επιτρέπει τον έλεγχο τόσο της κατεύθυνση όσο και της ταχύτητας του κινητήρα. Μία από τις πιο κοινές μονάδες "H Bridge" είναι εκείνη που βασίζεται στο ολοκληρωμένο κύκλωμα L298N.

### 3.1.1 Πώς λειτουργούν οι ηλεκτροκινητήρες

Σε έναν απλό κινητήρα συνεχούς ρεύματος υπάρχουν δύο βασικά εξαρτήματα, ο "στάτης" και ο "οπλισμός". Ο στάτης είναι ένας μόνιμος μαγνήτης ο οποίος παρέχει ένα σταθερό μαγνητικό πεδίο. Ο οπλισμός, που αποτελεί το περιστρεφόμενο τμήμα, είναι ένα απλό πηνίο.

Ο οπλισμός είναι συνδεδεμένος σε μια πηγή συνεχούς ρεύματος χρησιμοποιώντας έναν δακτύλιο 2 τεμαχίων που είναι τοποθετημένος γύρω από τον άξονα του κινητήρα. Αυτά τα τμήματα δακτυλίων ονομάζονται "δακτύλιοι μεταγωγού". Τα δύο τεμάχια των δακτυλίων μεταγωγού συνδέονται σε κάθε άκρο του πηνίου οπλισμού. Συνεχές ρεύμα κατάλληλης τάσης εφαρμόζεται στους δακτύλιους του συλλέκτη μέσω δύο "βουρτσών" που τρίβονται με αντίθετη φορά στους δακτύλιους.

Εφαρμόζοντας συνεχές ρεύμα στους δακτυλίους συλλέκτη, αυτό ρέει μέσω του πηνίου οπλισμού παράγοντας ένα μαγνητικό πεδίο. Αυτό το πεδίο προσελκύεται από τον μαγνήτη του στάτη (αντίθετες μαγνητικές πολικότητες έλκονται, οι όμοιες απωθούνται) και ο άξονας του κινητήρα αρχίζει να περιστρέφεται. Ο άξονας του κινητήρα περιστρέφεται μέχρι να φτάσει στη διακλάδωση μεταξύ των δύο μισών του μεταγωγού. Σε αυτό το σημείο οι βούρτσες έρχονται σε επαφή με το άλλο μισό των δακτυλίων του διακόπτη, αντιστρέφοντας την πολικότητα του πηνίου οπλισμού. Σε αυτό το σημείο ο άξονας του κινητήρα έχει περιστραφεί κατά 180 μοίρες και οι πολικότητες του μαγνητικού πεδίου πρέπει να αντιστραφούν για να συνεχίσει η περιστροφή του

κινητήρα. Αυτή η διαδικασία επαναλαμβάνεται επ' αόριστον μέχρι να απομακρυνθεί το ρεύμα από τα πηνία οπλισμού.

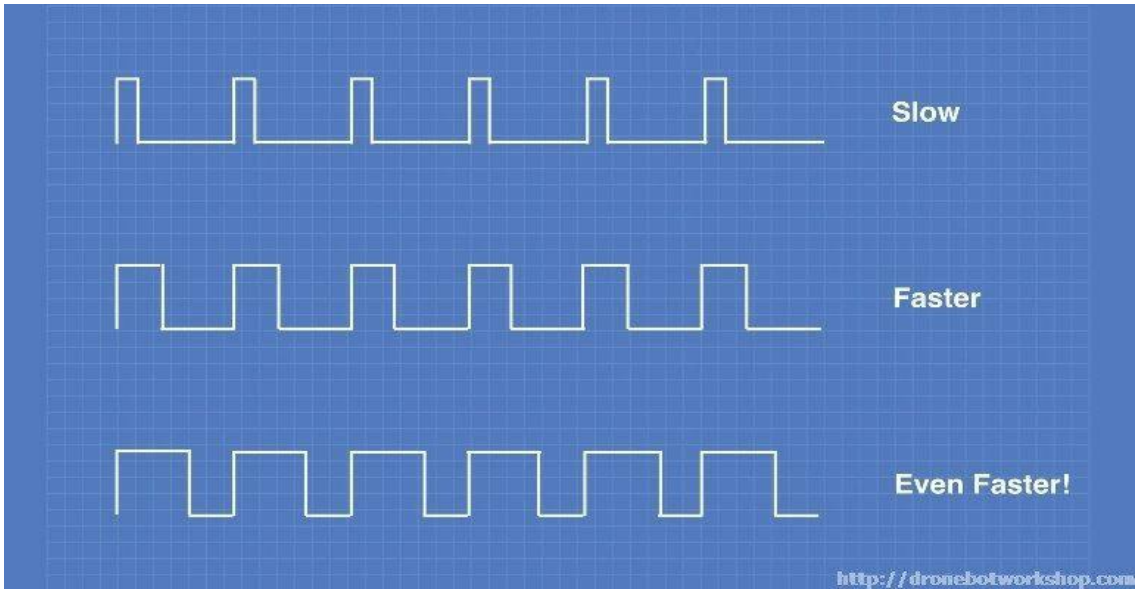
Ο κινητήρας που αναφέρεται παραπάνω καλείται ως ‘’Brushed DC Motor’’ γιατί έχει βούρτσες. Οι βούρτσες όμως μπορούν να δημιουργήσουν πρόβλημα στην εύρυθμη λειτουργία του κινητήρα. Όσο φθείρονται με την πάροδο του χρόνου, τρίβουν τον άξονα του κινητήρα και μπορεί να προκαλέσουν σπινθήρα καθώς ο κινητήρας γερνάει. Οι καλύτερης ποιότητας κινητήρες συνεχούς ρεύματος δεν έχουν βούρτσες. Οι κινητήρες χωρίς βούρτσες χρησιμοποιούν μια πιο περίπλοκη διάταξη πηνίων και δεν απαιτούν μεταγωγό. Το κινούμενο τμήμα του κινητήρα συνδέεται με τον μόνιμο μαγνήτη. Επειδή δεν περιέχουν βούρτσες, αυτοί οι κινητήρες έχουν μεγαλύτερη διάρκεια ζωής και παράγουν λιγότερο θόρυβο από τους κινητήρες με βούρτσες.

Οι κινητήρες συνεχούς ρεύματος καθορίζονται από το επίπεδο τάσης στο οποίο λειτουργούν. Οι συνήθεις κινητήρες συνεχούς ρεύματος για ερασιτεχνική χρήση-hobby τρέχουν σε 6Volt ή 12Volt . Για να αντιστραφεί η κατεύθυνση στην οποία περιστρέφεται ο κινητήρας συνεχούς ρεύματος αρκεί να αντιστραφεί η πολικότητα του συνεχούς ρεύματος που εφαρμόζεται σε αυτόν. Το δεύτερο βασικό στοιχείο του κινητήρα για το οποίο δημιουργείται ανάγκη ελέγχου είναι η ταχύτητά του. Μια μέθοδος αλλαγής της ταχύτητας ενός κινητήρα συνεχούς ρεύματος είναι η απλή μείωση της τάσης τροφοδοσίας. Ενώ αυτό θα λειτουργήσει σε κάποιο βαθμό, ωστόσο δεν είναι στην πραγματικότητα μια πολύ καλή μέθοδος ελέγχου της ταχύτητας του κινητήρα, καθώς η μείωση της τάσης θα μειώσει επίσης τη ροπή που μπορεί να παράγει ο κινητήρας. Επίσης, μόλις η τάση πέσει κάτω από ένα συγκεκριμένο σημείο, ο κινητήρας δεν θα περιστραφεί καθόλου.

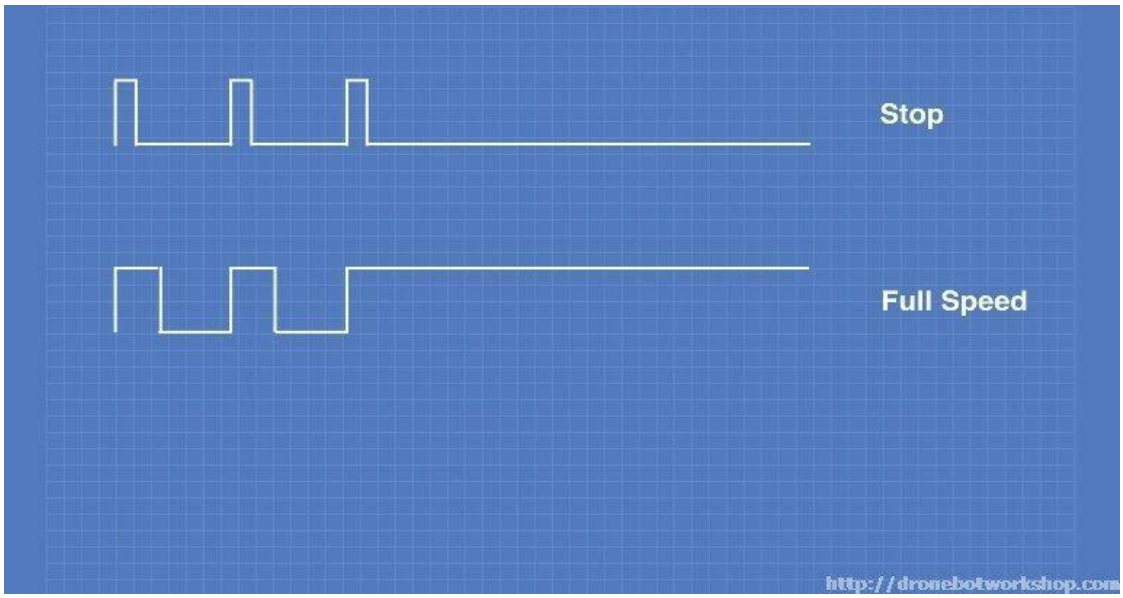
### 3.1.2 Ρύθμιση πλάτους παλμού (PWM)

Μια πολύ καλύτερη μέθοδος ελέγχου των κινητήρων συνεχούς ρεύματος είναι η χρήση διαμόρφωσης πλάτους παλμού ή PWM. Με το PWM ο κινητήρας στέλνει μια σειρά παλμών. Κάθε παλμός είναι της πλήρους τάσης που μπορεί να χειριστεί ο κινητήρας, έτσι ώστε ένας 6-volt κινητήρας να στείλει παλμούς 6Volt ενώ ένας κινητήρας 12Volt να στείλει παλμούς 12 Volt. Το πλάτος των παλμών μεταβάλλεται για τον έλεγχο της ταχύτητας του κινητήρα, οι παλμοί με ένα στενό

πλάτος προκαλούν την αργή περιστροφή του κινητήρα. Η αύξηση του πλάτους των παλμών θα αυξήσει την ταχύτητα του κινητήρα, όπως φαίνεται στην παρακάτω εικόνα.




Για να σταματήσει τελείως ο κινητήρας, αρκεί να σταλεί μηδενικός παλμός, στέλνοντας ουσιαστικά 0Volt. Για να περιστραφεί με την πλήρη ταχύτητα, αρκεί να σταλεί παλμός πλήρους τάσης, χωρίς να σταλεί άλλος παλμός.

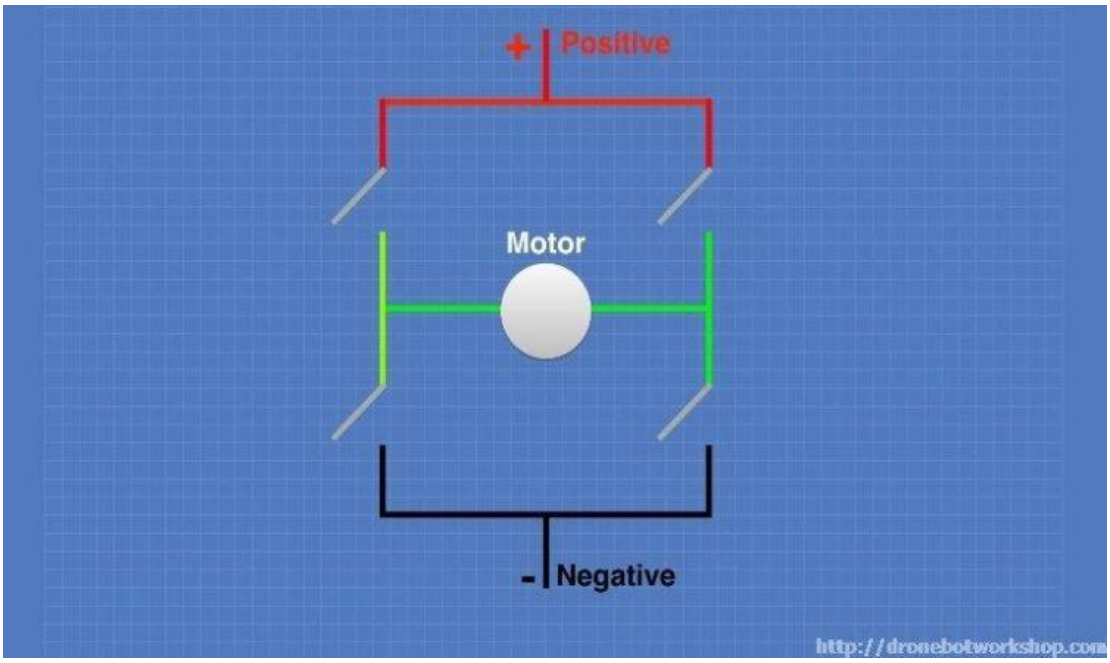



Το Arduino έχει μια λειτουργία που ονομάζεται "analogWrite", η οποία χρησιμοποιείται για την οδήγηση οποιωνδήποτε εξόδων που είναι συμβατές με PWM (το Arduino Uno διαθέτει 6 ψηφιακές εξόδους που είναι επίσης ικανές για PWM).

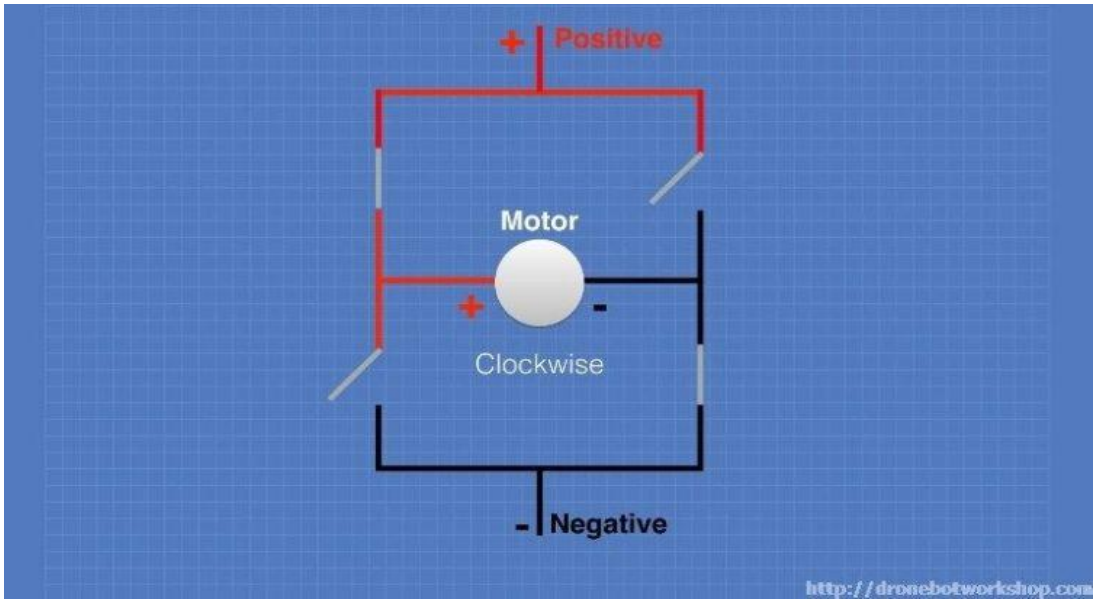
## 3.2 Ανάλυση πλακέτας «γέφυρας Η» (H-Bridge)

Η "H-Bridge" είναι απλώς μια διάταξη μεταγωγής της πολικότητας της τάσης που εφαρμόζεται σε έναν κινητήρα συνεχούς ρεύματος, ελέγχοντας έτσι την κατεύθυνση περιστροφής του. Για να γίνει κατανοητή η λειτουργία της γέφυρας στο παράδειγμα παρακάτω θα χρησιμοποιηθούν κάποιοι διακόπτες. Στην πραγματικότητα μια "H-Bridge" κατασκευάζεται συνήθως χρησιμοποιώντας τρανζίστορ. Η χρήση τρανζίστορ επιτρέπει τον έλεγχο της ταχύτητας του κινητήρα με PWM.

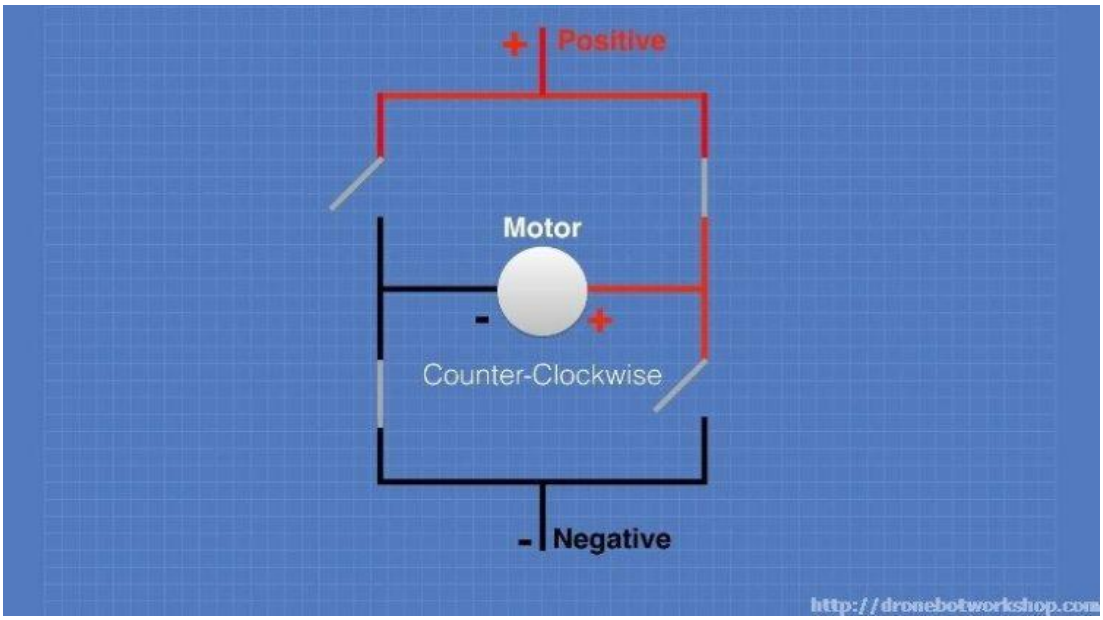
Στο πρώτο διάγραμμα υπάρχουν τέσσερις διακόπτες οι οποίοι είναι όλοι ανοιχτοί (off). Στο κέντρο του κυκλώματος υπάρχει ένας κινητήρας DC. Στο κύκλωμα παρατηρείται ότι έχει σχεδιαστεί ένα γράμμα "H", με τον κινητήρα συνδεδεμένο στο κέντρο. Έτσι προέκυψε ο όρος "H-Bridge".




Εάν κλείσουν (δηλαδή ανάψουν) δύο από τους διακόπτες εφαρμόζεται η τάση στον κινητήρα, προκαλώντας την περιστροφή του δεξιόστροφα.




Στο επόμενο σκίτσο ανοίγουν αυτοί οι διακόπτες και κλείνουν οι άλλοι δύο. Αυτό προκαλεί την αντιστροφή πολικότητας της τάσης που εφαρμόζεται στον κινητήρα, με αποτέλεσμα ο κινητήρας να περιστρέφεται αριστερόστροφα.




Αυτό είναι αρκετά απλό αλλά αποτελεσματικό. Στην πραγματικότητα, το μόνο που χρειάζεται είναι η σχεδίαση ενός κυκλώματος για να κινηθεί ο κινητήρας σε πλήρη ταχύτητα και προς οποιαδήποτε κατεύθυνση. Αλλά στην παρούσα κατάσταση ο έλεγχος του κινητήρα θα γίνεται χρησιμοποιώντας έναν Arduino. Επομένως, αυτό που χρειάζεται είναι οι διακόπτες να αντικατασταθούν από τρανζίστορ.

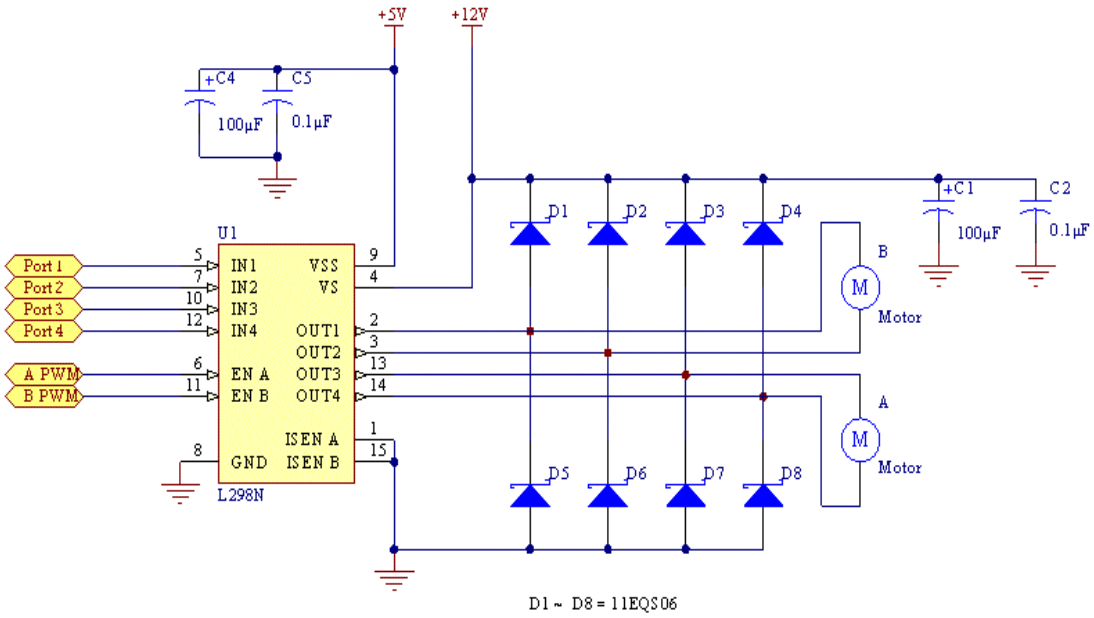
### 3.2.1 H-Bridge τύπου L298N

Ενώ μπορούν να χρησιμοποιηθούν διακριτά τρανζίστορ για να γίνει μια «H-Bridge», η χρήση ενός ολοκληρωμένου κυκλώματος παρέχει περισσότερα πλεονεκτήματα. Ένας αριθμός από «H-Bridge» ελέγχου κινητήρα είναι διαθέσιμες και όλες τους λειτουργούν με τον ίδιο τρόπο. Ένα από τα πιο δημοφιλή ολοκληρωμένα κυκλώματα είναι το L298N.

Το L298N είναι μέλος μιας οικογένειας IC (integrated circuit) που έχουν όλοι την ονομασία "L298". Η διαφορά μεταξύ των μελών της οικογένειας είναι στο ύψος του ρεύματος που μπορούν να χειριστούν. Το L298N μπορεί να χειριστεί έως και 3Amperes στα 35Volt DC, το οποίο είναι κατάλληλο για τους περισσότερους κινητήρες.

Το L298N περιέχει δύο ολοκληρωμένα κυκλώματα «H-Bridge», έτσι ώστε να μπορεί να κινεί ένα ζεύγος κινητήρων συνεχούς ρεύματος. Αυτό το καθιστά ιδανικό για ρομποτικά έργα, καθώς τα περισσότερα ρομπότ έχουν είτε δύο είτε τέσσερις κινητήρες.

Διάγραμμα ενός ολοκληρωμένου κυκλώματος L298N:



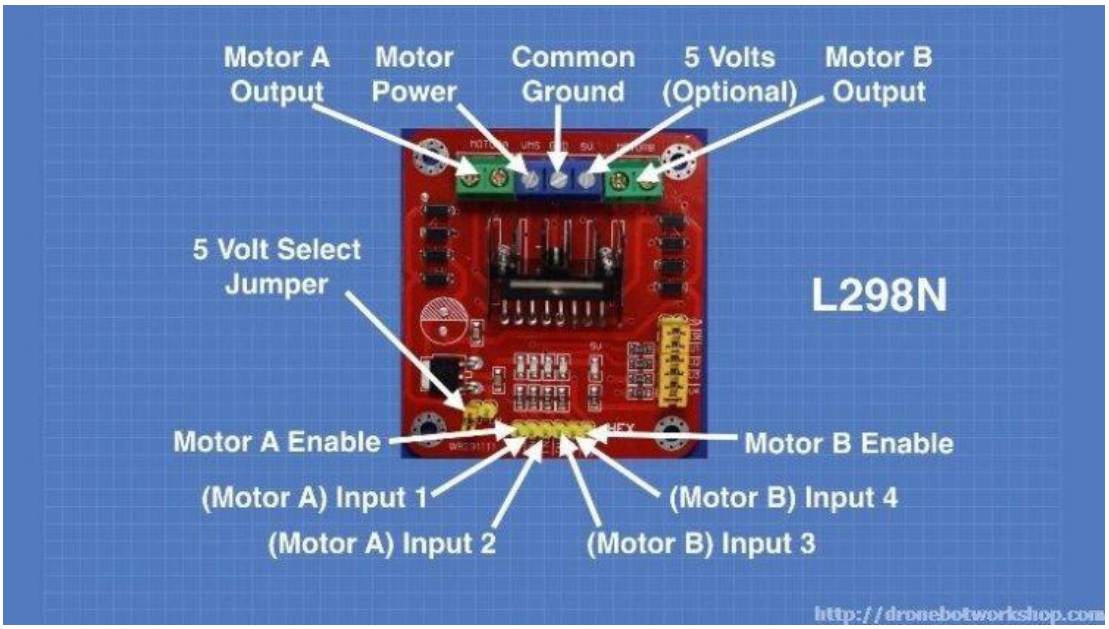

Είναι πολύ πιο εύκολη η αγορά ενός ολοκληρωμένο κύκλωμα L298N, το οποίο είναι ενσύρματο και ολοκληρωμένο με συνδετήρες για κινητήρες, τροφοδοτικά και λογική εισόδου. Οι πίνακες αυτοί έχουν επίσης έναν ρυθμιστή

τάσης 5Volt, ο οποίος μπορεί να χρησιμοποιηθεί για την τροφοδοσία των λογικών κυκλωμάτων.

### 3.2.2 Θήρες εξόδου για το L298N

Υπάρχουν διαφορετικά στυλ πλακετών L298N αλλά όλα λειτουργούν με τον ίδιο τρόπο. Η πλακέτα περιέχει ένα L298N τοποθετημένο σε μια ψήκτρα, ένας ρυθμιστής τάσης 5 volt "προαιρετικά" για να παρέχει ισχύ στα λογικά κυκλώματα της πλακέτας, υποστηρίζοντας διόδους, πυκνωτές και συνδετήρες ως εξής:

- Λογικές εισόδους για κάθε κύκλωμα H-Bridge
- Εισαγωγές τροφοδοσίας ρεύματος για την τροφοδοσία του κινητήρα
- Προαιρετική είσοδος ισχύος 5 volt για τα λογικά κυκλώματα.
- Εξόδους για κάθε κινητήρα συνεχούς ρεύματος




**Εικόνα 1 Πλακέτα L298N**

Στην πλακέτα παρατηρούνται επίσης αρκετές βραχυκυκλώσεις, οι οποίες αναγράφονται παρακάτω:

- **CSA**: Αυτή είναι η λειτουργία "ανίχνευσης ρεύματος" (current sensing) για τον κινητήρα Α. Μεταξύ αυτού του πείρου και της γείωσης συνδέεται η αντίσταση

που ελέγχει το ρεύμα φορτίου. Στις περισσότερες περιπτώσεις αυτός ο βραχυκυκλωτής μένει στην θέση του.

- **CSB**: Αυτή είναι η λειτουργία "ανίχνευσης ρεύματος" για τον κινητήρα Β. Και πάλι συνήθως αυτός ο βραχυκυκλωτής παραμένει στη θέση του.
- **U1:** Αντίσταση ανύψωσης εισόδου 1. Αυτή ενεργοποιεί μια αντίσταση ανύψωσης 10k για την είσοδο.
- **U2**: Αντίσταση ανύψωσης εισόδου 2.
- **U3**: Αντίσταση ανύψωσης εισόδου 3.
- **U4**: Αντίσταση ανύψωσης εισόδου 4.
- **5V-EN**: Αυτός είναι ο μόνος βραχυκυκλωτής στον οποίο πρέπει να δοθεί προσοχή. Όταν αυτός ο βραχυκυκλωτής είναι στη θέση του, επιτρέπει ρύθμιση 5Volt στα λογικά κυκλώματα 78M05, παρέχοντας λογική ισχύ από την τροφοδοσία του κινητήρα. Όταν βρίσκεται στη θέση του αυτός ο βραχυκυκλωτής, δεν χρειάζεται να τροφοδοτηθεί με 5Volts το τερματικό εισόδου 5V. Όταν αφαιρεθεί ο βραχυκυκλωτής, θα χρειαστεί τροφοδοσία 5Volts στο τερματικό εισόδου 5V.

Αν χρησιμοποιηθεί ο εσωτερικός ρυθμιστής τάσης, τότε θα πρέπει η τροφοδοσία του κινητήρα να είναι τουλάχιστον 7,5Volts. Η τροφοδοσία του κινητήρα θα πρέπει να έχει λίγο υψηλότερη η τάση από τις πραγματικές απαιτήσεις. Αυτό οφείλεται στην εσωτερική πτώση τάσης στα τρανζίστορ που σχηματίζουν το κύκλωμα H-Bridge. Η συνδυασμένη πτώση τάσης είναι 1,4Volt οπότε αν χρησιμοποιηθούν κινητήρες 6Volt στην πλακέτα θα χρειαστεί τάση 7,4Volt. Αν χρησιμοποιηθούν κινητήρες 12Volt τότε η τάση τροφοδοσίας στην πλακέτα θα πρέπει να είναι 13,4Volt.

Η πλακέτα έχει τέσσερις ακροδέκτες εισόδου και δύο τερματικά ενεργοποίησης. Αυτοί οι ακροδέκτες είναι που χρησιμοποιούνται για τον έλεγχο της κατεύθυνσης και της ταχύτητας κάθε κινητήρα.

- **ENA**: Είσοδος τροφοδοσίας για τον κινητήρα Α
- **IN1**: Είσοδος 1 για τον κινητήρα Α
- **IN2**: Είσοδος 2 για τον κινητήρα Α
- **IN3**: Είσοδος 3 για τον κινητήρα Β

- **IN4**: Είσοδος 4 για τον κινητήρα Β
- **ENB**: Είσοδος τροφοδοσίας για τον κινητήρα Β

Στον παρακάτω πίνακα απεικονίζεται ένας απλός τρόπος πώς χρησιμοποιούνται αυτές οι είσοδοι για να επιτευχθεί η αλλαγή φοράς περιστροφής ενός ηλεκτροκινητήρα.

Σε περίπτωση που υπάρχει ένας ηλεκτροκινητήρας "Α" θα χρησιμοποιηθούν οι 3 πρώτες είσοδοι. Στην είσοδο ΕΝΑ συνδέεται η τροφοδοσία του κινητήρα, στις εισόδους IN1 και IN2 καθορίζεται η κίνηση του κινητήρα «εμπρός» ή «πίσω» βάζοντας τιμές (5Volt) ή (Ground).

| INPUT 1 | INPUT 2 | ΚΑΤΑΣΤΑΣΗ ΚΙΝΗΤΗΡΑ |
|---|---|---|
| Ground | Ground | Κλειστός |
| 5Volt | Ground | Εμπρός |
| Ground | 5Volt | Πίσω |
| 5Volt | 5Volt | Δεν λειτουργεί |

Όπως φαίνεται μόνο δύο συνδυασμοί χρησιμοποιούνται στην πραγματικότητα για τον έλεγχο της κατεύθυνσης περιστροφής των κινητήρων.

Η γραμμή ενεργοποίησης μπορεί να χρησιμοποιηθεί για να ενεργοποιήσει τον κινητήρα, να τον απενεργοποιήσει και να ελέγξει την ταχύτητά του. Όταν η γραμμή ενεργοποίησης είναι στα 5Volt, ο κινητήρας θα είναι αναμμένος. Γειώνοντας τη γραμμή ενεργοποίησης (Ground) θα απενεργοποιηθεί ο κινητήρας.

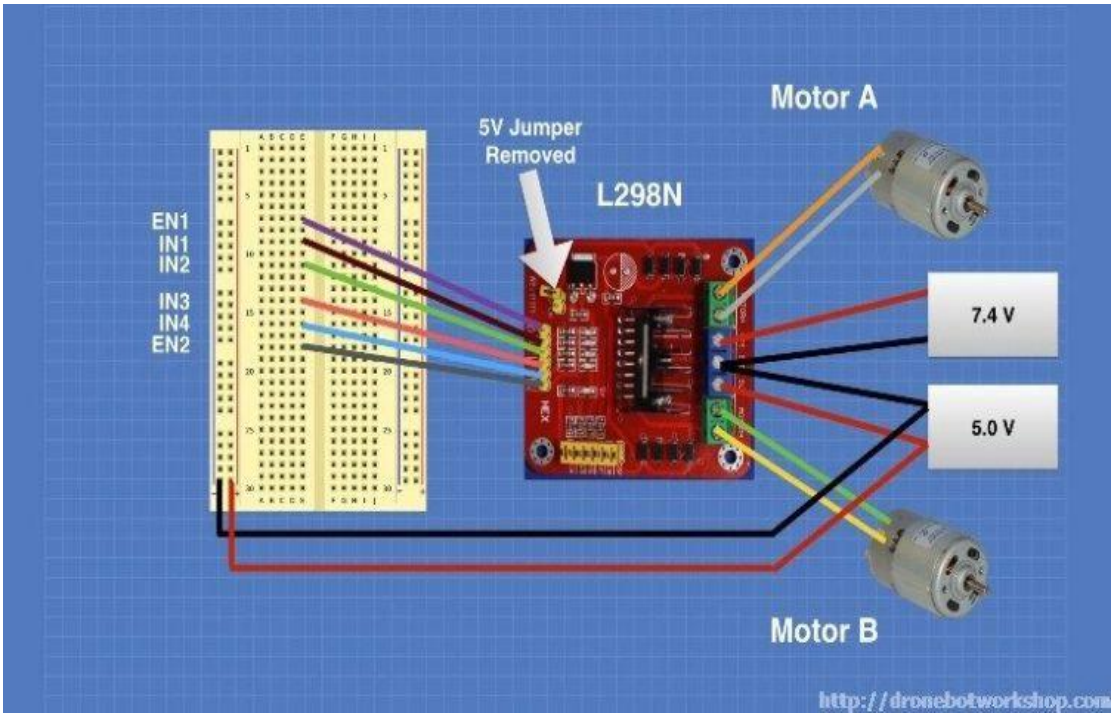
Για τον έλεγχο της ταχύτητας του κινητήρα, μπορεί να εφαρμοστεί ένα σήμα διαμόρφωσης πλάτους παλμού (PWM) στη γραμμή ενεργοποίησης. Όσο πιο σύντομο είναι το πλάτος του παλμού, τόσο πιο αργά θα περιστραφεί ο κινητήρας.

### 3.2.3 Πειράματα με την πλακέτα L298n σε Breadboard

Για τον έλεγχο μιας μονάδας L298N αρκεί ένα breadboard, ένα ζεύγος κινητήρων και μερικά καλώδια.

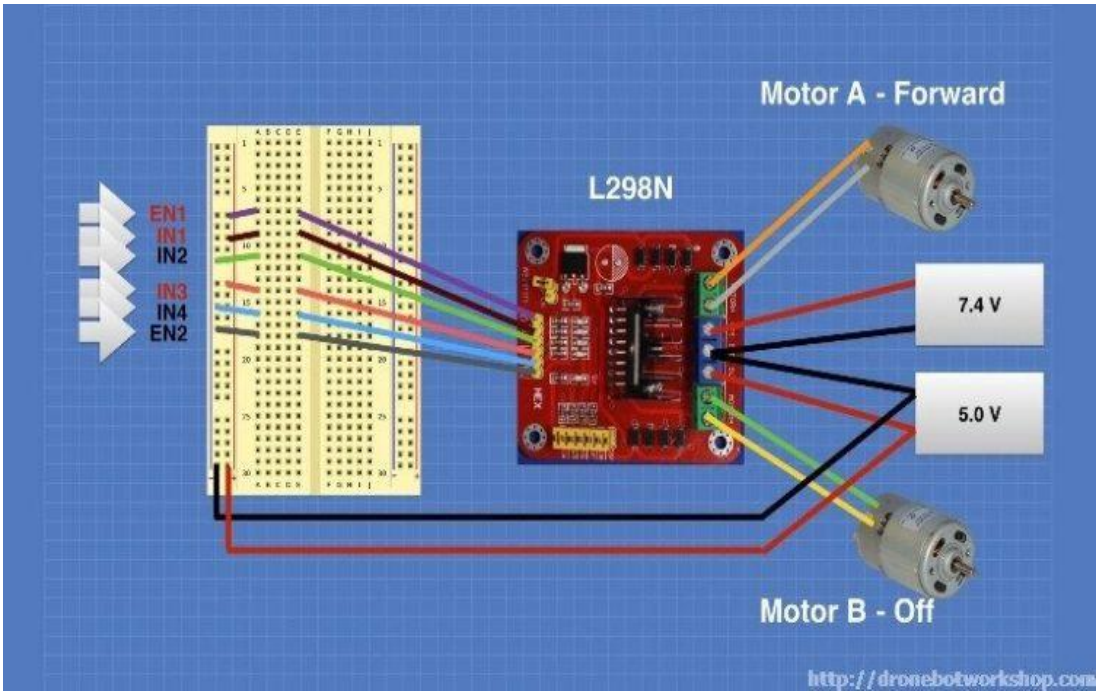
Θα χρειαστεί επίσης τροφοδοσία ισχύος για τους κινητήρες. Στο πείραμα, χρησιμοποιείται και δεύτερη τροφοδοσία των 5Volt για την τροφοδοσία του λογικού κυκλώματος, αλλά θα μπορεί να χρησιμοποιηθεί μία τροφοδοσία τόσο για τους κινητήρες όσο και για το λογικό κύκλωμα. Απλά πρέπει να ρυθμιστεί σωστά ο βραχυκυκλωτής 5Volt (αν χρησιμοποιηθεί μία τροφοδοσία τότε θα πρέπει να είναι στη θέση του, αν χρησιμοποιηθούν δύο, πρέπει να αφαιρεθεί ο βραχυκυκλωτής).

Η αρχική σύνδεση εμφανίζεται ως εξής:



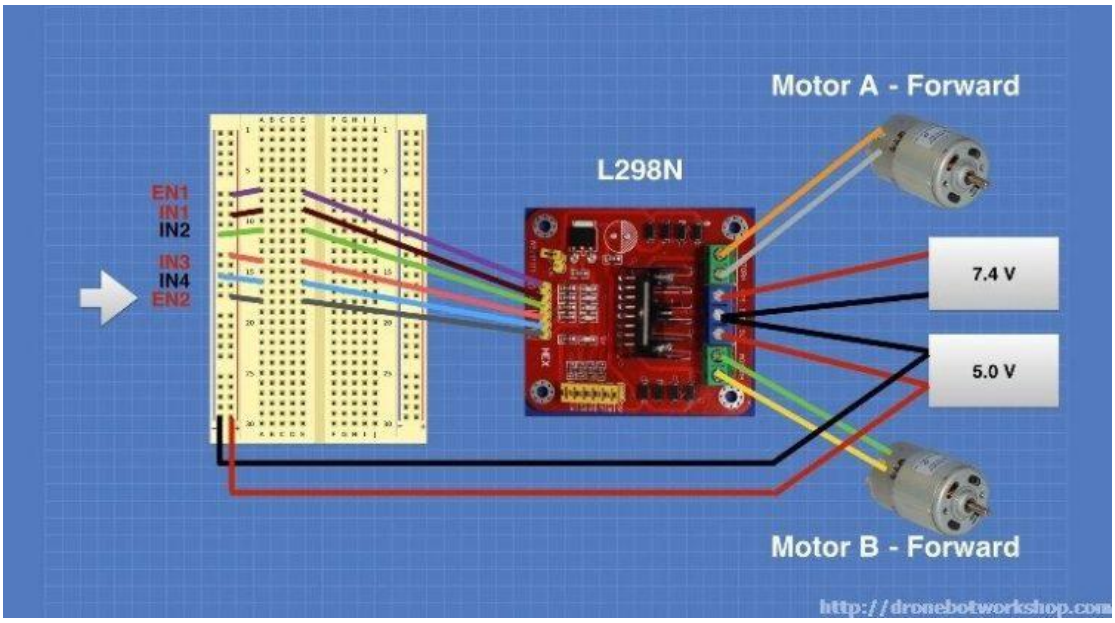

Σημειώνεται ότι η παροχή τόσο των κινητήρων όσο και του λογικού κυκλώματος έχουν την ίδια γείωση. Η τροφοδοσία των 5Volt και η γείωση (Ground) θα συνδεθούν στο breadboard, έτσι ώστε να μπορεί να γίνει η ρύθμιση για τα λογικά κυκλώματα.

Για να λειτουργήσει θα πρέπει να γίνει η σύνδεση των 5Volt και γείωσης (Ground) στις γραμμές εισαγωγής και ενεργοποίησης. Στο πρώτο πείραμα συνδέονται τα EN1, IN1 και IN3 στα 5Volt. Οι υπόλοιπες γραμμές εισόδου και ενεργοποίησης θα συνδεθούν με τη γείωση.

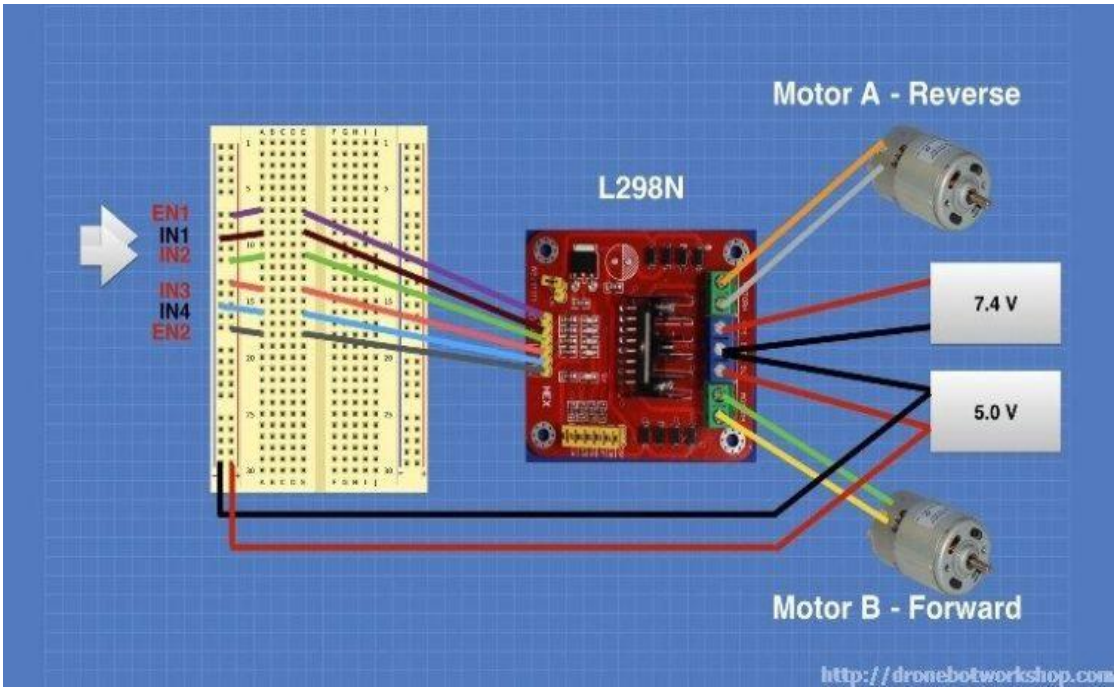



Αυτή η συνδεσμολογία θα προκαλέσει την κίνηση του κινητήρα Α προς τα εμπρός. Καθώς η γραμμή ενεργοποίησης για τον κινητήρα Β (ENB) βρίσκεται στη γείωση, θα παραμείνει απενεργοποιημένος.

Για την ενεργοποίηση του κινητήρα Β πρέπει να συνδεθεί το ENB στα 5Volt.



Αυτό θα προκαλέσει την περιστροφή και των δύο κινητήρων προς τα εμπρός. Τώρα θα γίνει αλλαγή στις εισόδους για τον κινητήρα Α, θέτοντας το IN1 στη γείωση και το IN2 στα 5Volt. Οι άλλες συνδέσεις θα παραμείνουν όπως είναι.

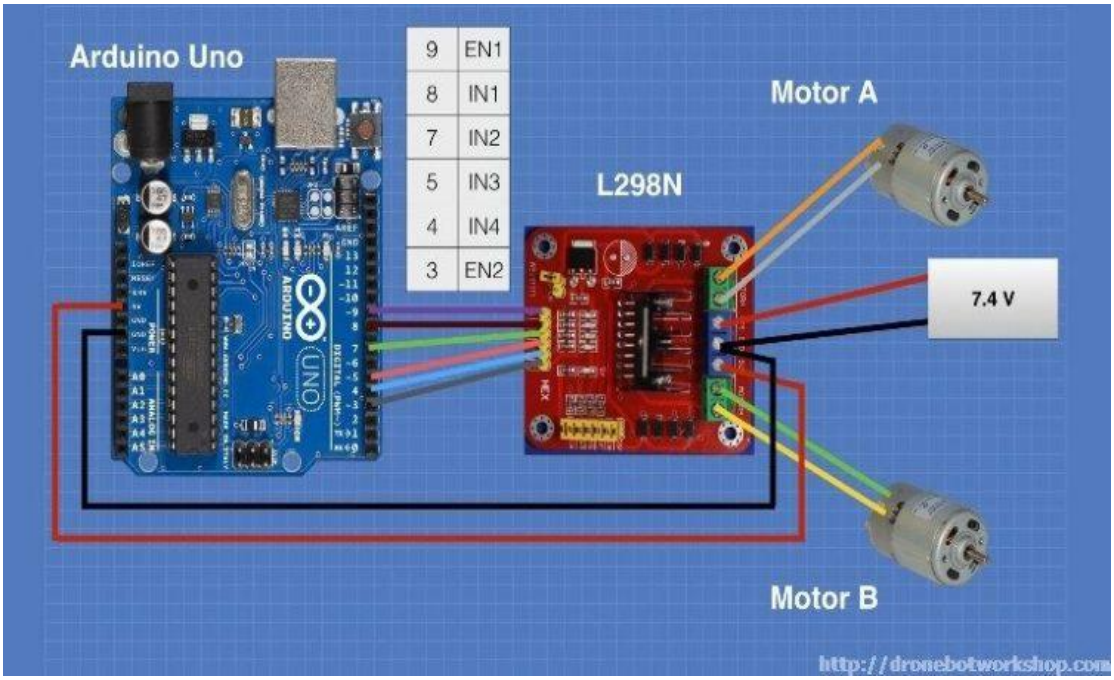



Σε αυτό το γράφημα ο κινητήρας Α κινείται προς τα πίσω, ενώ ο κινητήρας Β συνεχίζει να κινείται προς τα εμπρός.

Με τη ρύθμιση αυτή μπορεί κανείς να ελέγξει την κατεύθυνση, καθώς και να ενεργοποιήσει ή να απενεργοποιήσει τους δύο κινητήρες. Αλλά ένα πράγμα που δεν μπορεί να ελεγχθεί με αυτόν τον τρόπο είναι ο έλεγχος της ταχύτητας του κινητήρα. Για το σκοπό αυτό θα χρειαστεί η εφαρμογή ενός σήματος PWM στις γραμμές ενεργοποίησης ENA και ENB.

### 3.2.4 Συνδέοντας έναν Arduino με την πλακέτα L298N

Χρησιμοποιώντας έναν Arduino ή παρόμοιο μικροελεγκτή καθιστά δυνατό τον έλεγχο της κατεύθυνσης και της ταχύτητας του κάθε μοτέρ. Στο παράδειγμα χρησιμοποιείται ένας Arduino Uno. Ο Arduino Uno διαθέτει 14 ακροδέκτες εισόδου/εξόδου (I / O), έξι από τους οποίους είναι σε θέση να παρέχουν σήμα PWM. Το παρακάτω διάγραμμα δείχνει πώς είναι συνδεδεμένος ο Arduino Uno με την πλακέτα L298N.




Σημειώνεται ότι τα 5Volt για την πλακέτα L298N τροφοδοτούνται τώρα από την έξοδο Arduino 5Volt. Ο Arduino τροφοδοτείται μέσω του καλωδίου USB, το οποίο θα επιτρέψει με τη σειρά του τη φόρτωση του sketch για να λειτουργήσει το L298n. Αφού ανεβεί το sketch, αφαιρείται το καλώδιο USB και τροφοδοτείται ο Arduino με εξωτερική τροφοδοσία. Οι γραμμές εισόδου και ενεργοποίησης στο L298N ελέγχονται από έξι ακροδέκτες ψηφιακής εξόδου Arduino, ως εξής:

| Arduino | L298n |
|---|---|
| 9 | ENA |
| 8 | IN1 |
| 7 | IN2 |
| 5 | IN3 |
| 4 | IN4 |
| 3 | ENB |

Επιλέχθηκαν αυτές οι έξοδοι, γιατί οι ακίδες του Arduino 9 και 3 είναι και οι δύο ικανές για διαμόρφωση εύρους παλμού (PWM). Παρακάτω αντικατοπτρίζεται ένα σχετικό sketch.

```
// Motor A
int enA = 9;
int in1 = 8;
int in2 = 7;

// Motor B
int enB = 3;
int in3 = 5;
int in4 = 4;

void setup() {
// Set all the motor control pins to outputs

  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

}
void loop() {
// Turn on motor A
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
//Θέτουμε ταχύτητα 200 (εύρος τιμών 0-255)
  analogWrite(enA, 200);



  // Turn on motor B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

  //Θέτουμε ταχύτητα 200 (εύρος τιμών 0-255)
  analogWrite(enB, 200);

  delay(2000);

  // Αλλαγή κατεύθυνσης

  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);

  delay(2000);

  // Turn off motors

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```

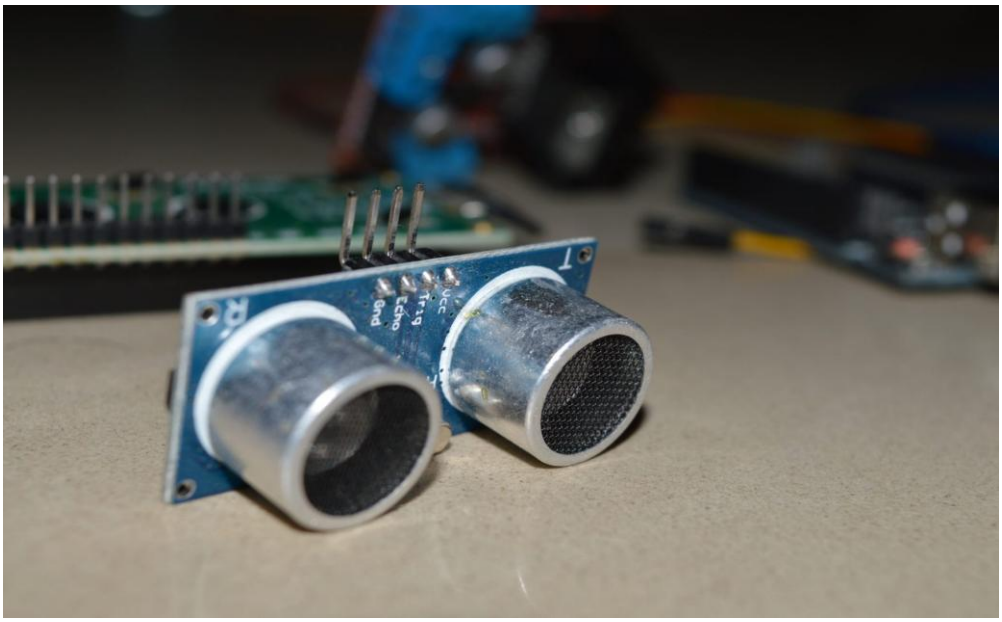
Αυτό το σκίτσο δείχνει μια σειρά από ενέργειες που μπορούν να γίνουν προκειμένου να ελεγχθεί μια γέφυρα H-L298N με τη χρήση Arduino. Θα μπορούσε επίσης να χρησιμοποιηθεί και σε άλλους ελεγκτές H-Bridge, μιας και όλοι λειτουργούν με παρόμοιο τρόπο.

Αρχικά καθορίζονται οι ακέραιοι αριθμοί που αντιπροσωπεύουν τους ακροδέκτες ψηφιακής εξόδου που θα χρησιμοποιηθούν για να οδηγήσουν τον ελεγκτή H-Bridge. Αυτή είναι μια καλή πρακτική κωδικοποίησης, καθώς επιτρέπει την εναλλαγή των θηρών αλλάζοντας απλώς αυτές τις τιμές, επιτρέποντας έτσι να χρησιμοποιηθεί το ίδιο σκίτσο σε διαφορετικό Arduino. Βασική προϋπόθεση αποτελεί ότι οι δύο θήρες ενεργοποίησης, που ορίζονται ως ENA και ENB, πρέπει να συνδεθούν με εξόδους που υποστηρίζουν PWM.

Στη συνάρτηση setup ορίζονται όλοι αυτοί οι ακροδέκτες ως ψηφιακές έξοδοι. Στη συνάρτηση loop ορίζεται η λειτουργία. Στο συγκεκριμένο παράδειγμα αυτή η λειτουργία περιστρέφει τους δύο κινητήρες προς τα εμπρός με ταχύτητα 200 για δύο δευτερόλεπτα. Στη συνέχεια, αναστρέφει τους κινητήρες και τους περιστρέφει για άλλα δύο δευτερόλεπτα. Τέλος, απενεργοποιεί τους κινητήρες. Η συνάρτηση analogWrite χρησιμοποιείται για την παραγωγή σημάτων PWM στις ψηφιακές εξόδους.

## 3.3 Αισθητήρας απόστασης HC-SR04 Ultrasonic

Ο αισθητήρας απόστασης HC-SR04 είναι μια οικονομική συσκευή που είναι πολύ χρήσιμη για τα ρομποτικά και τα τεστ εξοπλισμού δοκιμών. Αυτός ο μικροσκοπικός αισθητήρας είναι σε θέση να μετρήσει την απόσταση μεταξύ του ίδιου και του πλησιέστερου στερεού αντικειμένου. Το HC-SR04 μπορεί να συνδεθεί απευθείας σε ένα Arduino ή άλλο μικροελεγκτή και λειτουργεί στα 5Volt.



Αυτός ο αισθητήρας απόστασης υπερήχων είναι ικανός να μετρά αποστάσεις μεταξύ 2cm και 400cm. Είναι μια συσκευή χαμηλής κατανάλωσης ρεύματος, ώστε να είναι κατάλληλη για συσκευές με μπαταρία.

### 3.3.1 Πώς λειτουργεί ο HC-SR04 Ultrasonic

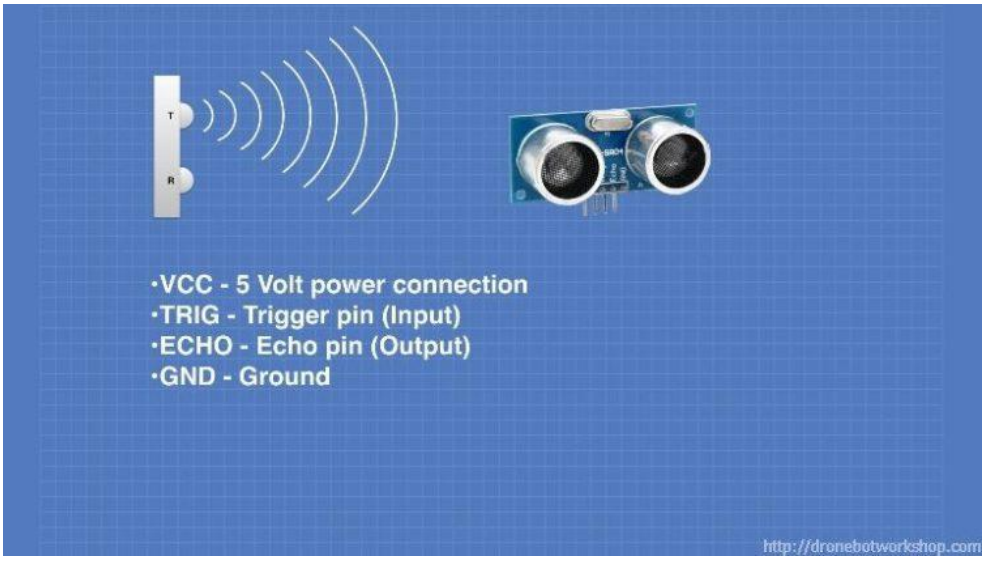
Οι υπερηχητικοί αισθητήρες απόστασης χρησιμοποιούν παλμούς υπερηχητικού ήχου (ήχου πάνω από το εύρος της ανθρώπινης ακοής) για να ανιχνεύσουν την απόσταση μεταξύ των ίδιων και των κοντινών στερεών αντικειμένων. Οι αισθητήρες αποτελούνται από δύο κύριες συνιστώσες:

- **Ένας υπερηχητικός πομπός**: Αυτός μεταδίδει τους υπέρηχους παλμούς ήχου και λειτουργεί σε 40 KHz
- **Ένας υπερηχητικός δέκτης**: Ο δέκτης που ακούει τους μεταδιδόμενους παλμούς. Όταν τους λαμβάνει, παράγει έναν παλμό εξόδου το πλάτος του οποίου

μπορεί να χρησιμοποιηθεί για τον προσδιορισμό της απόστασης που διάνυσε ο παλμός.

Το HC-SR04 έχει τις ακόλουθες τέσσερις συνδέσεις:

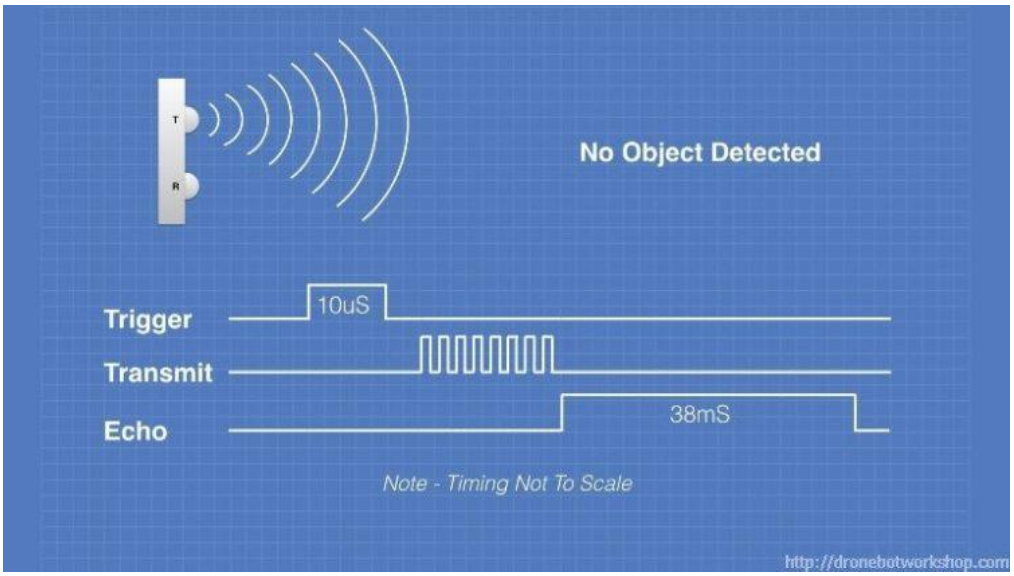
- **VCC**: Παροχή ισχύος 5Volt.
- **Trig**: Ο ακροδέκτης "Trigger" που χρησιμοποιείται για να στέλνει τους παλμούς υπερήχων.
- **Echo**: Ο ακροδέκτης που παράγει έναν παλμό όταν ληφθεί το ανακλώμενο σήμα. Το μήκος του παλμού είναι ανάλογο του χρόνου που χρειαζόταν για την ανίχνευση του μεταδιδόμενου σήματος.
- **GND**: Ο ακροδέκτης γείωσης.

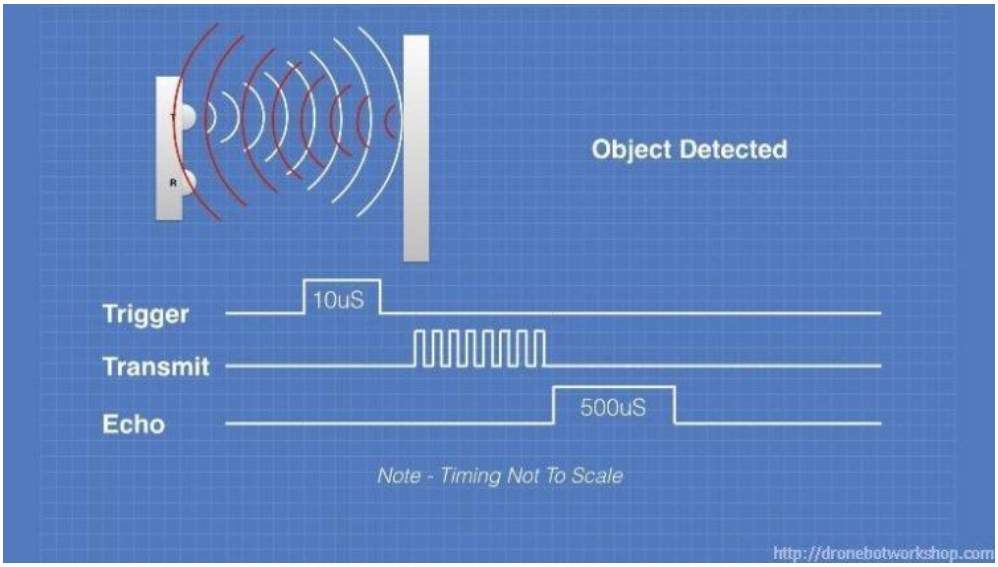



Η συσκευή λειτουργεί ως εξής:

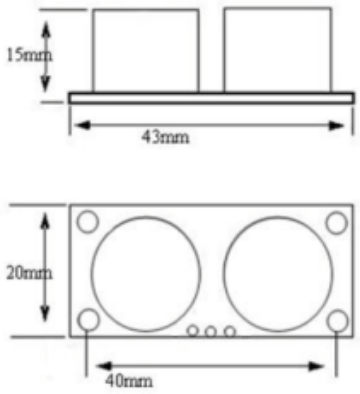
1. Ένας παλμός 5Volt, τουλάχιστον 10μS σε διάρκεια εφαρμόζεται στην θύρα Trigger.
2. Ο αισθητήρας αποκρίνεται με τη μετάδοση μιας έκρηξης οκτώ παλμών στα 40KHz. Αυτό το μοτίβο 8 παλμών καθιστά την "υπερηχητική υπογραφή" από τη συσκευή μοναδική, επιτρέποντας έτσι στον δέκτη να μπορεί να διακρίνει μεταξύ του μεταδιδόμενου σχεδίου και των εξωτερικών θορύβων.

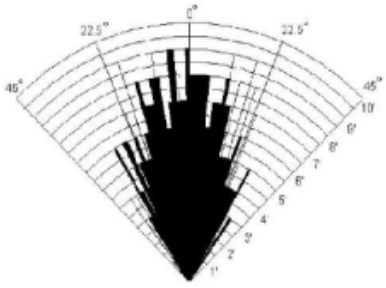
3. Οι οκτώ παλμοί υπερήχων μετακινούνται στον αέρα από τον πομπό. Εν τω μεταξύ, ο ακροδέκτης Echo ενεργοποιείται για να αρχίσει να σχηματίζεται η αρχή του σήματος.
4. Εάν ο παλμός δεν αντικατοπτρίζεται, τότε το σήμα Echo θα λήξει σε χρονικό όριο μετά από 38mS (38 milliseconds) και θα κλείσει. Αυτό παράγει έναν παλμό 38mS που δεν δείχνει κανένα εμπόδιο στην περιοχή του αισθητήρα.
5. Εάν ο παλμός αντανακλάται πίσω, ο ακροδέκτης Echo κλείνει όταν ληφθεί το σήμα. Αυτό παράγει έναν παλμό του οποίου το πλάτος κυμαίνεται μεταξύ 150μS έως 25 mS, ανάλογα με το χρόνο που χρειάζεται για να ληφθεί το σήμα.
6. Το πλάτος του λαμβανόμενου παλμού χρησιμοποιείται για τον υπολογισμό της απόστασης στο ανακλώμενο αντικείμενο. Ο παλμός δείχνει το χρόνο που χρειάστηκε το σήμα για να σταλεί και να ανακλαστεί. Για να βρεθεί λοιπόν η απόσταση χρειάζεται η διαίρεση του αποτελέσματος στο μισό.






Η παρακάτω εικόνα δείχνει τις διαστάσεις του αισθητήρα απόστασης HC-SR04 καθώς και την πραγματική γωνία λειτουργίας. Ο αισθητήρας είναι πιο ακριβής όταν το αντικείμενο που θα ανιχνευθεί βρίσκεται ακριβώς μπροστά του, ωστόσο μπορεί να αντιληφθεί αντικείμενα που βρίσκονται μέσα σε ένα εύρος 45 μοιρών. Για πιο ακριβείς ενδείξεις συνιστάται να περιοριστεί το παράθυρο σε 30 μοίρες (15 μοίρες σε κάθε πλευρά).



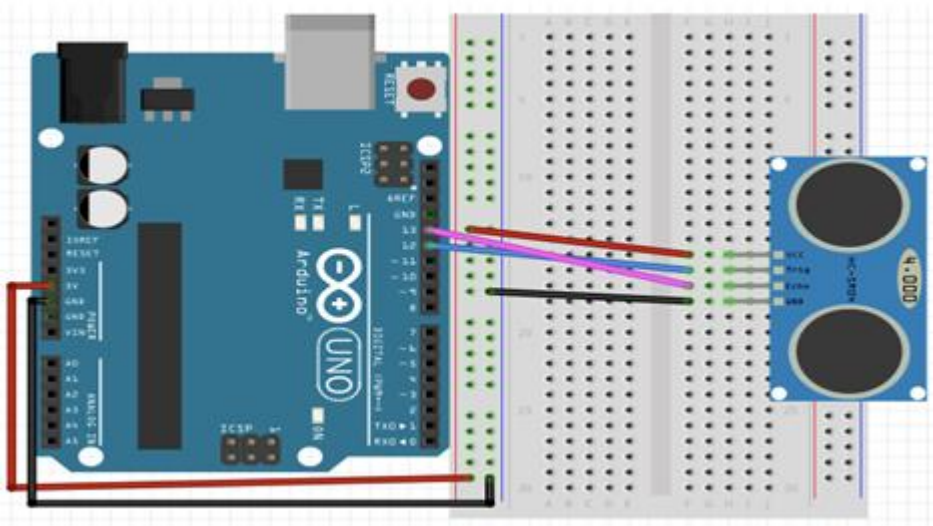




### 3.3.2 Σύνδεση αισθητήρα HC-SR04 Ultrasonic με Arduino UNO

Υπάρχουν τέσσερις ακίδες που χρησιμοποιούνται για τη διασύνδεση του αισθητήρα με τον Arduino: VCC, Trig (pin εξόδου σήματος), Echo (pin εισόδου σήματος) και GND. Κάθε ένας από τους τέσσερις ακροδέκτες (pin) συνδέεται με τον Arduino ως ακολούθως:

- VCC: συνδέεται στον ακροδέκτη 5V του επεξεργαστή
- Trig: συνδέεται σε ψηφιακό ακροδέκτη (ορισμένο ως είσοδο) του επεξεργαστή
- Echo: συνδέεται σε ψηφιακό ακροδέκτη (ορισμένο ως έξοδο) του επεξεργαστή
- GND: συνδέεται σε ακροδέκτη GND (γείωση) του επεξεργαστή

Για την διασύνδεσή του με τον Arduino δεν χρειάζονται αντιστάσεις.



### 3.3.3 Προγραμματισμός του αισθητήρα

Πρώτα πρέπει να οριστούν οι ακροδέκτες Trig και Echo. Σε αυτή την περίπτωση είναι οι αριθμοί 9 και 10 στην πλακέτα του Arduino και ονομάζονται trigPin και echoPin. Στη συνέχεια, χρειάζεται μια μεταβλητή Long, που θα ονομαστεί "duration", η διάρκεια δηλαδή του χρόνου ταξιδιού που θα ληφθεί από τον αισθητήρα και μια μεταβλητή int για την απόσταση "distance".

Στην "setup" πρέπει να οριστεί το trigPin ως έξοδος και το echoPin ως είσοδος και επίσης να ξεκινήσει η σειριακή επικοινωνία για την εμφάνιση των αποτελεσμάτων στη σειριακή οθόνη.

Στην "Loop" πρέπει πρώτα να μηδενίσει το trigPin, οπότε χρειάζεται η ρύθμιση του ακροδέκτη σε κατάσταση LOW για μόλις 2μs. Τώρα για τη δημιουργία του ηχητικού κύματος πρέπει να ρυθμιστεί το trigPin σε κατάσταση HIGH για 10μs. Χρησιμοποιώντας τη λειτουργία **pulseIn ()** διαβάζεται ο χρόνος

ταξιδιού και η τιμή του μπαίνει στη μεταβλητή "duration". Αυτή η λειτουργία έχει 2 παραμέτρους, η πρώτη είναι το όνομα echopin και η δεύτερη είναι η τιμή που ορίζεται (HIGH ή LOW). Στην περίπτωση αυτή, η τιμή HIGH σημαίνει ότι η συνάρτηση **pulsIn ()** θα περιμένει να πάρει ο ακροδέκτης την τιμή HIGH που προκαλείται από το ηχητικό κύμα και θα ξεκινήσει τη χρονομέτρηση. Έπειτα, θα περιμένει τον ακροδέκτη να πάρει την τιμή LOW. Οταν το ηχητικό κύμα τελειώσει τότε θα σταματήσει τη χρονομέτρηση και τέλος θα επαναφέρει το μήκος του παλμού σε microseconds. Για τη λήψη της απόσταση θα χρειαστεί να πολλαπλασιαστεί η διάρκεια κατά 0.034 και να διαιρεθεί κατά 2. Στο τέλος θα εκτυπωθεί η τιμή της απόστασης στην σειριακή οθόνη.

```
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;

// defines variables
long duration;
int distance;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}

void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);

// Calculating the distance
distance= duration*0.034/2;

// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
}
```

Ο Ultrasonic HC-SR04 μπορεί να λειτουργήσει και σε αναλογική θύρα του Arduino, με την λήψη και ενσωμάτωση της κατάλληλης βιβλιοθήκης στο sketch όπως φαίνεται παρακάτω. Στο συγκεκριμένο παράδειγμα οι αναλογικές θύρες που χρησιμοποιούνται είναι η A0 για το Trig και η A1 για το Echo και μια μεταβλητή int για την απόσταση "distance". Στη "setup" πρέπει να ξεκινήσει η σειριακή επικοινωνία για την εμφάνιση των αποτελεσμάτων στη σειριακή οθόνη. Στη "loop" αρχικά ορίζεται η μονάδα μέτρησης για την απόσταση και το κείμενο που θα εμφανίζει η σειριακή οθόνη επικοινωνίας.

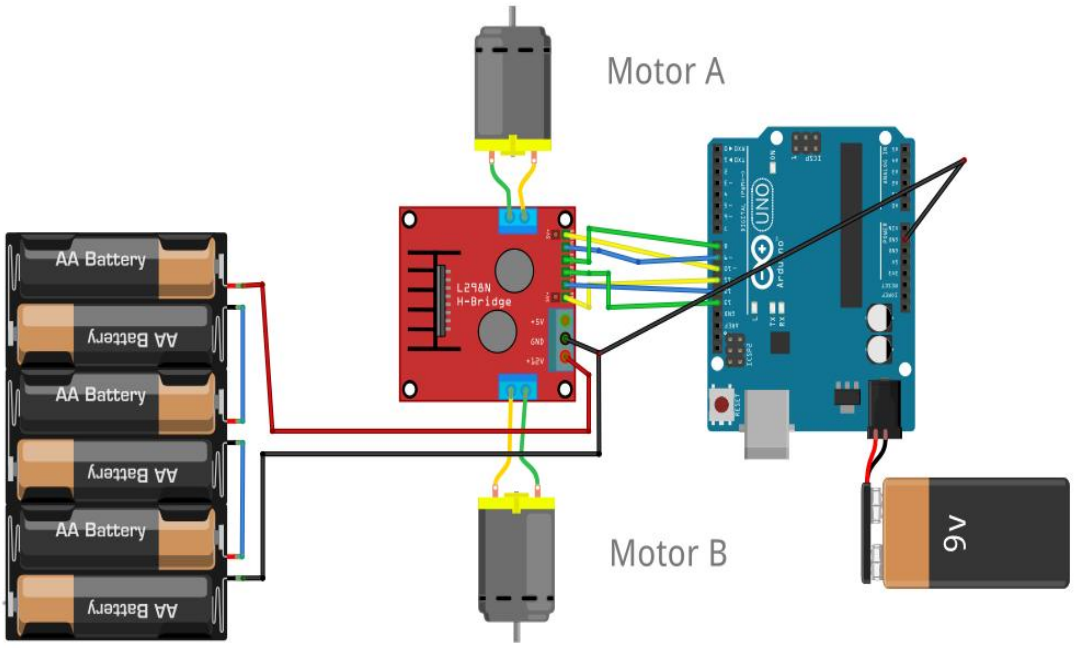
```
//Libraries
#include "Ultrasonic.h"

//Define pins ultrasonic(trig,echo)
Ultrasonic ultrasonic(A0,A1);

//Variables
int distance;

void setup() {
  Serial.begin(9600);
}

void loop()
{
  distance = ultrasonic.Ranging(CM); //Use 'CM' for centimeters or 'INC' for inches
  //Print distance...
  Serial.print("Object found at: ");
  Serial.print(distance);
  Serial.println("cm");
  //every 1sec.
  delay(1000);
}
```

# ΚΕΦΑΛΑΙΟ 4

## 4.1 Συνδεσμολογία κυκλώματος H-Bridge L298N με Arduino.

Όπως έχει προαναφερθεί, με τη βοήθεια του επεξεργαστή Arduino μπορούμε με πολύ εύκολο τρόπο και χωρίς πλήθος εξαρτημάτων και ωρών εργασίας να χειριστούμε δυο ηλεκτροκινητήρες ως προς την ταχύτητα και την κατεύθυνση περιστροφής τους.

Παρακάτω απεικονίζεται η συνδεσμολογία του επεξεργαστή και των ηλεκτροκινητήρων με την πλακέτα L298N.




Αναλυτική συνδεσμολογία πλακέτας L298N:

- **MotorA:** Σε αυτές τις θύρες συνδέεται ο ηλεκτροκινητήρας Α όπου στη συγκεκριμένη περίπτωση είναι αυτός που καθορίζει την κίνηση του αμαξιού μπρος ή πίσω.

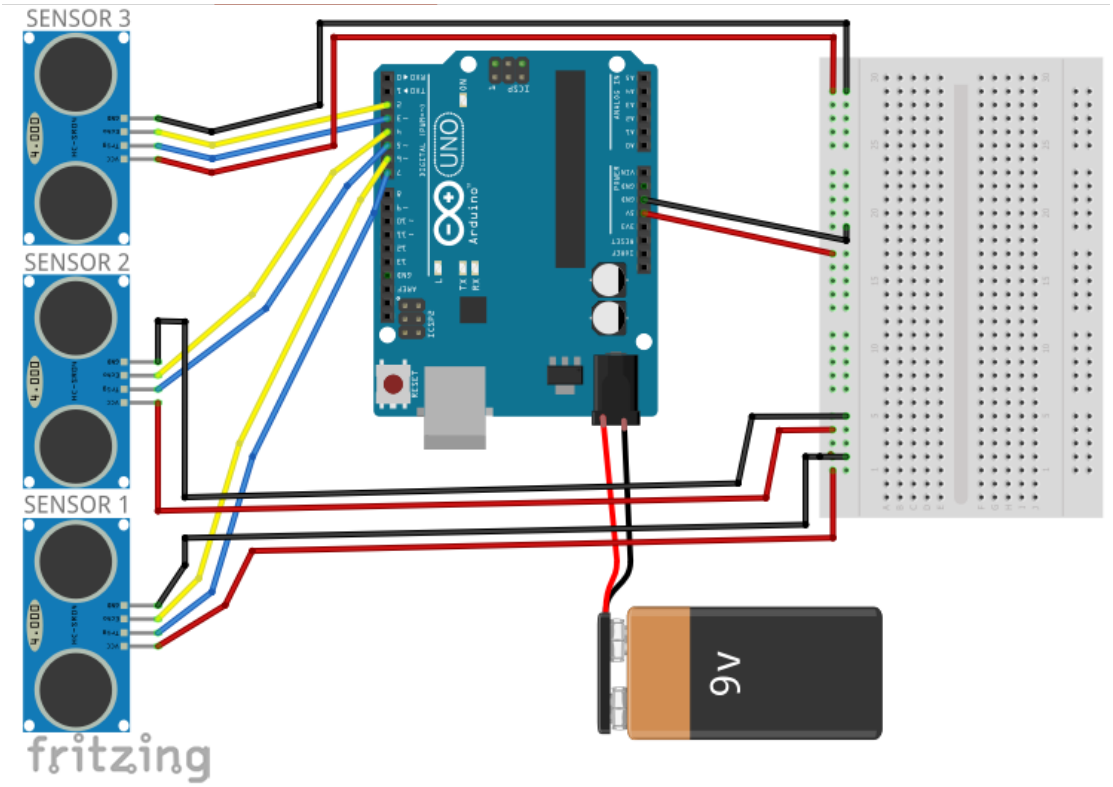
- **MotorB**: Σε αυτές τις θύρες συνδέεται ο ηλεκτροκινητήρας Β όπου στη συγκεκριμένη περίπτωση είναι αυτός που καθορίζει την κατεύθυνση του αμαξιού δεξιά ή αριστερά.
- **Vms**: Αυτή η είσοδος είναι για την τροφοδοσία των ηλεκτροκινητήρων στην οποία συνδέονται οι μπαταρίες οι οποίες συνολικά έχουν τάση 9V.
- **Gnd**: Συνδέεται με την γείωση ή αλλιώς την αρνητική πόλωση των μπαταριών.
- **5V**: Αυτή η είσοδος είναι για την τροφοδοσία του λογικού κυκλώματος της πλακέτας. Σε αυτή την περίπτωση επιλέξαμε να μην υπάρχει δεύτερη τροφοδοσία για τον λόγο ότι οι κινητήρες είναι ρυθμισμένοι να καταναλώνουν τη μισή τάση απ' αυτήν που μπορεί να δώσει η L298N. Συνεπώς, επαρκεί η κύρια πηγή και βραχυκυκλώνοντας τους ακροδέκτες 5V-EN μπορούμε να έχουμε τροφοδοσία και για το λογικό κύκλωμα.
- **ENA**: Αυτή η είσοδος συνδέεται με την ψηφιακή θύρα 11 του Arduino η οποία είναι τύπου PWM έτσι ώστε να μπορεί να ρυθμιστεί η τάση που θα δέχεται ο ηλεκτροκινητήρας Α και κατά συνέπεια και η ταχύτητά του.
- **IN1**: Αυτή η είσοδος συνδέεται στην ψηφιακή θύρα 12 του επεξεργαστή ώστε να δέχεται τις τιμές 0 και 1 (0V και 5V) και σε συνδυασμό με τις τιμές που θα δέχεται η είσοδος IN2 να αλλάζει κατεύθυνση ο ηλεκτροκινητήρας Α.
- **IN2**: Αυτή η είσοδος συνδέεται στην ψηφιακή θύρα 13 του επεξεργαστή ώστε να δέχεται τις τιμές 0 και 1 (0V και 5V) και σε συνδυασμό με τις τιμές που θα δέχεται η είσοδος IN1 να αλλάζει κατεύθυνση ο ηλεκτροκινητήρας Α.
- **ENB**: Αυτή η είσοδος συνδέεται με την ψηφιακή θύρα 10 του Arduino η οποία είναι τύπου PWM έτσι ώστε να μπορεί να ρυθμιστεί η τάση που θα δέχεται ο ηλεκτροκινητήρας Β και συνεπώς και η ταχύτητα περιστροφής του.
- **IN3**: Αυτή η είσοδος συνδέεται στην ψηφιακή θύρα 8 του επεξεργαστή ώστε να δέχεται τις τιμές 0 και 1 (0V και 5V) και σε συνδυασμό με τις τιμές που θα δέχεται η είσοδος IN4 να αλλάζει κατεύθυνση ο ηλεκτροκινητήρας Β
- **IN4**: Αυτή η είσοδος συνδέεται στην ψηφιακή θύρα 9 του επεξεργαστή ώστε να δέχεται τις τιμές 0 και 1 (0V και 5V) και σε συνδυασμό με τις τιμές που θα δέχεται η είσοδος IN3 να αλλάζει κατεύθυνση ο ηλεκτροκινητήρας Β.

## 4.2 Συνδεσμολογία κυκλώματος αισθητήρων HC-SR04 Ultrasonic με τον επεξεργαστή Arduino.

Ο αισθητήρας υπερήχων χρησιμοποιεί δύο διατάξεις, μία για να στείλει ένα υπερηχητικό σήμα και μία για να το λάβει. Έτσι, από το χρόνο που μεσολαβεί από τη στιγμή που στέλνει μέχρι τη στιγμή που λαμβάνει το σήμα πίσω, μπορούμε να υπολογίσουμε την απόσταση στην οποία βρίσκεται μπροστά μας κάποιο αντικείμενο.

Για τη συνδεσμολογία του χρησιμοποιούμε ένα καλώδιο για την πηγή (Vcc – 5V), ένα για τη γείωση (GND), μια θύρα για να στέλνουμε σήμα (trigger) και μια θύρα για να λαμβάνουμε το σήμα που επιστρέφει (Echo). Η θύρα για το trigger θα οριστεί ως εξόδου, ενώ η θύρα για το echo ως εισόδου στη συνάρτηση setup(). Η αποστολή σήματος γίνεται δίνοντας τάση στο trigger.

Στην παρακάτω εικόνα απεικονίζεται η συνδεσμολογία που πραγματοποιήθηκε στην παρούσα εργασία.



Όσον αφορά την παραπάνω συνδεσμολογία, αρχικά συνδέουμε στην τροφοδοσία του Arduino μια μπαταρία των 9V. Από τις θύρες ισχύος του επεξεργαστή θα συνδεθούν δυο καλώδια με το breadboard. Στο πρώτο καλώδιο (κόκκινο στο παρόν σχέδιο) η μια άκρη του θα συνδεθεί στη θύρα 5V του Arduino και η άλλη άκρη του στην πρώτη βραχυκυκλωμένη γραμμή του breadboard. Στο δεύτερο καλώδιο (μαύρο στο παρόν σχέδιο) η μία άκρη θα συνδεθεί στη θύρα Gnd του επεξεργαστή και η άλλη άκρη στην δεύτερη βραχυκυκλωμένη γραμμή του breadboard.

Λόγω του ότι οι μπαταρίες της κατασκευής τροφοδοτούν το κεντρικό κύκλωμα με τάση 9V, συνδέοντας τους αισθητήρες με αυτό, υπάρχει κίνδυνος να καούν. Πρέπει λοιπόν να δημιουργηθεί η συνδεσμολογία με τον επεξεργαστή έτσι ώστε οι αισθητήρες να μπορούν τροφοδοτηθούν με τάση 5V χωρίς να υπάρχει κίνδυνος προβλήματος.

Για τον αισθητήρα ''sensor 3" :

- **Vcc**: Αυτή η θύρα που είναι η τροφοδοσία του αισθητήρα (κόκκινο καλώδιο) συνδέεται με την βραχυκυκλωμένη γραμμή του breadboard που τροφοδοτείται με τα 5V του Arduino.
- **Trig**: Αυτή η θύρα του αισθητήρα συνδέεται με την ψηφιακή θύρα 3 του Arduino.
- **Echo**: Αυτή η θύρα του αισθητήρα συνδέεται με την ψηφιακή θύρα 2 του Arduino.
- **Gnd**: Αυτή η θύρα η οποία είναι η γείωση του αισθητήρα (μαύρο καλώδιο) συνδέεται με την βραχυκυκλωμένη γραμμή του breadboard που συνδέεται με την γείωση του Arduino.

Για τον αισθητήρα ''sensor 2" :

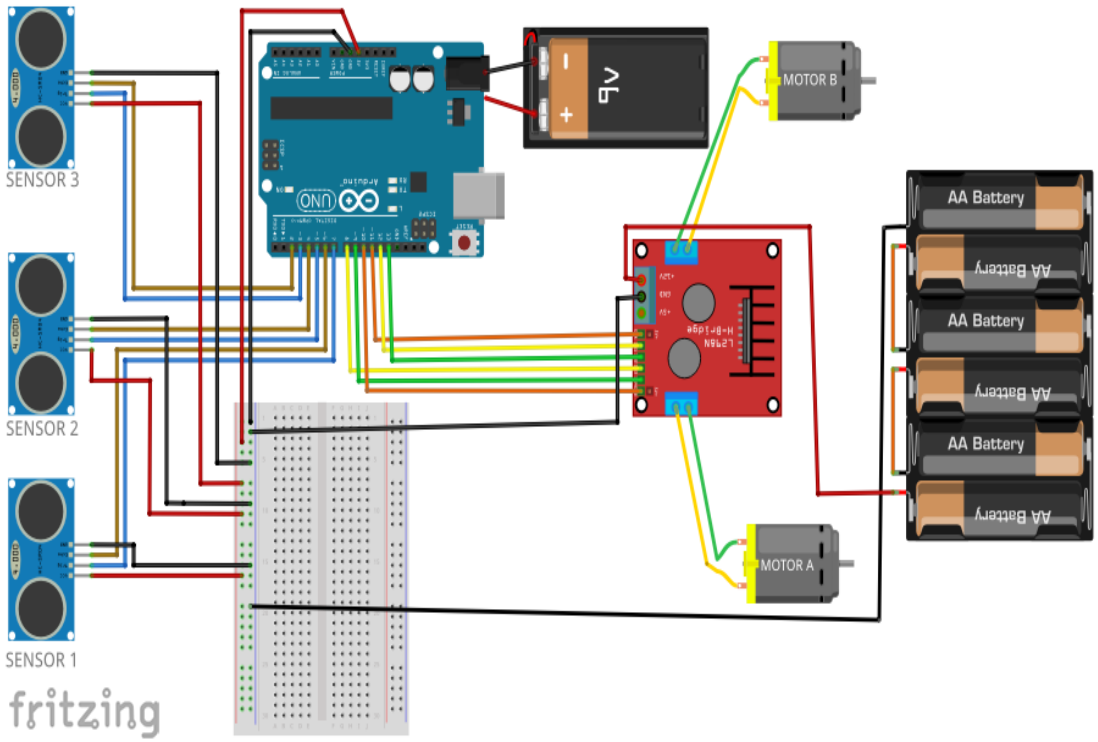
- Οι θύρες **Vcc** και **Gnd** έχουν την ίδια συνδεσμολογία με αυτές του αισθητήρα ''sensor 3".
- **Trig**: Αυτή η θύρα του αισθητήρα συνδέεται με την ψηφιακή θύρα 5 του Arduino.
- **Echo**: Αυτή η θύρα του αισθητήρα συνδέεται με την ψηφιακή θύρα 4 του Arduino.

Για τον αισθητήρα ''sensor 1" :

- Οι θύρες **Vcc** και **Gnd** έχουν την ίδια συνδεσμολογία με αυτές του αισθητήρα ''sensor 3" και "sensor 2".
- **Trig**: Αυτή η θύρα του αισθητήρα συνδέεται με την ψηφιακή θύρα 7 του Arduino.
- **Echo**: Αυτή η θύρα του αισθητήρα συνδέεται με την ψηφιακή θύρα 6 του Arduino.

### 4.3 Συνδεσμολογία πλήρους κυκλώματος κατασκευής

Παραπάνω παρουσιάστηκε και αναλύθηκε ο τρόπος σύνδεσης της πλακέτας L298N H-Bridge με τους ηλεκτροκινητήρες και τον Arduino καθώς και ο τρόπος σύνδεσης των αισθητήρων απόστασης HC-SR04 Ultrasonic με τον Arduino. Στην παρακάτω εικόνα απεικονίζεται το πλήρες κύκλωμα της παρούσας κατασκευής, που δεν είναι κάτι περισσότερο από το συνδυασμό των δύο παραπάνω κυκλωμάτων, με την διαφορά ότι σε αυτήν την περίπτωση το breadboard χρησιμοποιείται και για την σύνδεση της πλακέτας L298N H-Bridge με την γείωση.

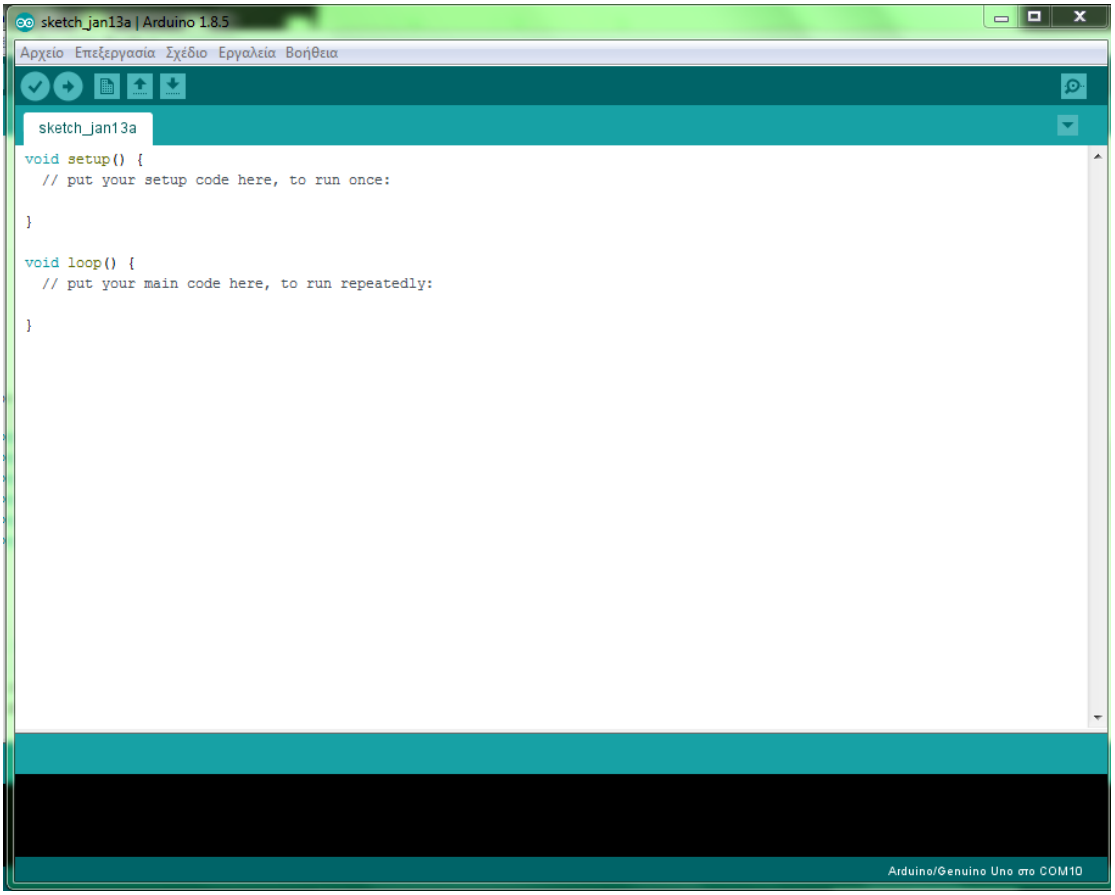


# Κεφάλαιο 5

# Υλοποίηση του κώδικα ελέγχου του Arduino.

## 5.1 Ξεκινώντας με τον Arduino IDE

Για να ξεκινήσει η συγγραφή του κώδικα ελέγχου, χρειάζεται αρχικά η λήψη του αρμόδιου προγράμματος "Arduino IDE". Το πρόγραμμα είναι πολύ εύκολο στην εγκατάστασή του και διατίθεται δωρεάν για κάθε τύπου λογισμικό στην ιστοσελίδα του Arduino, όπως έχει αναφερθεί και στο κεφάλαιο 1.



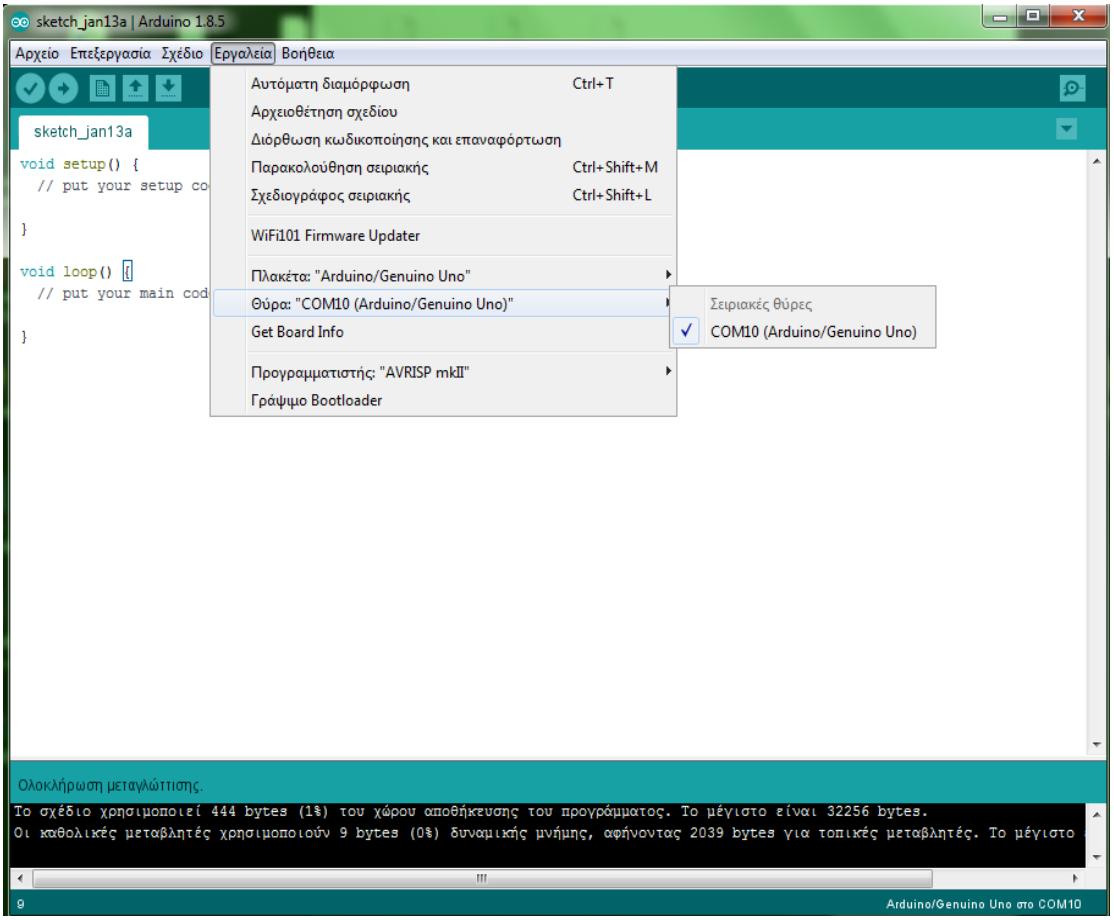

Εικόνα 4.1 Περιβάλλον εργασίας Arduino IDE

Έχοντας εγκαταστήσει το πρόγραμμα στον υπολογιστή, όταν το ανοίξετε θα δείτε αυτό που απεικονίζεται στην παραπάνω εικόνα (στη συγκεκριμένη

περίπτωση το πρόγραμμα είναι εγκατεστημένο σε λογισμικό Windows). Αυτό είναι το περιβάλλον στο οποίο γράφεται ο κώδικας του Arduino.

Η δομή του κώδικα είναι η εξής: σαν κύριες ρουτίνες έχουμε την «void setup» και την «void loop». Η setup είναι για τη δήλωση εισόδων και εξόδων του επεξεργαστή, ενώ η loop είναι το κύριο πρόγραμμα όπου θα τρέχει διαρκώς ο επεξεργαστής. Πριν από την setup είναι ο χώρος στον οποίο εισάγονται τυχόν βιβλιοθήκες (libraries) που ίσως χρειάζονται για την υλοποίηση κάποιου κώδικα. Επίσης, δηλώνονται οι μεταβλητές που χρησιμοποιούνται και ποιες θήρες είναι δεσμευμένες ώστε να μπορεί ο επεξεργαστής να τις αναγνωρίσει και να συμπεριληφθούν στον κεντρικό κώδικα. Παράλληλα μπορούν να οριστούν και κάποιες διαδικασίες οι οποίες διευκολύνουν την υλοποίηση του κύριου κώδικα.

Η επόμενη κίνηση που πρέπει να γίνει στο πρόγραμμα είναι η επιλογή της πλακέτας στην οποία θα τρέξει το πρόγραμμα (στο δικό μας παράδειγμα είναι ο Arduino Uno) και η σύνδεση του Arduino με τον υπολογιστή μέσω ενός καλωδίου usb. Για να ελεγχθεί ότι ο υπολογιστής διαβάζει κανονικά τον Arduino στην είσοδό του, θα πρέπει στην καρτέλα «Εργαλεία» → «θύρα» να εμφανιστεί ο αριθμός της θύρας όπου είναι συνδεδεμένος ο επεξεργαστής. Παράδειγμα στην εικόνα 4.2.

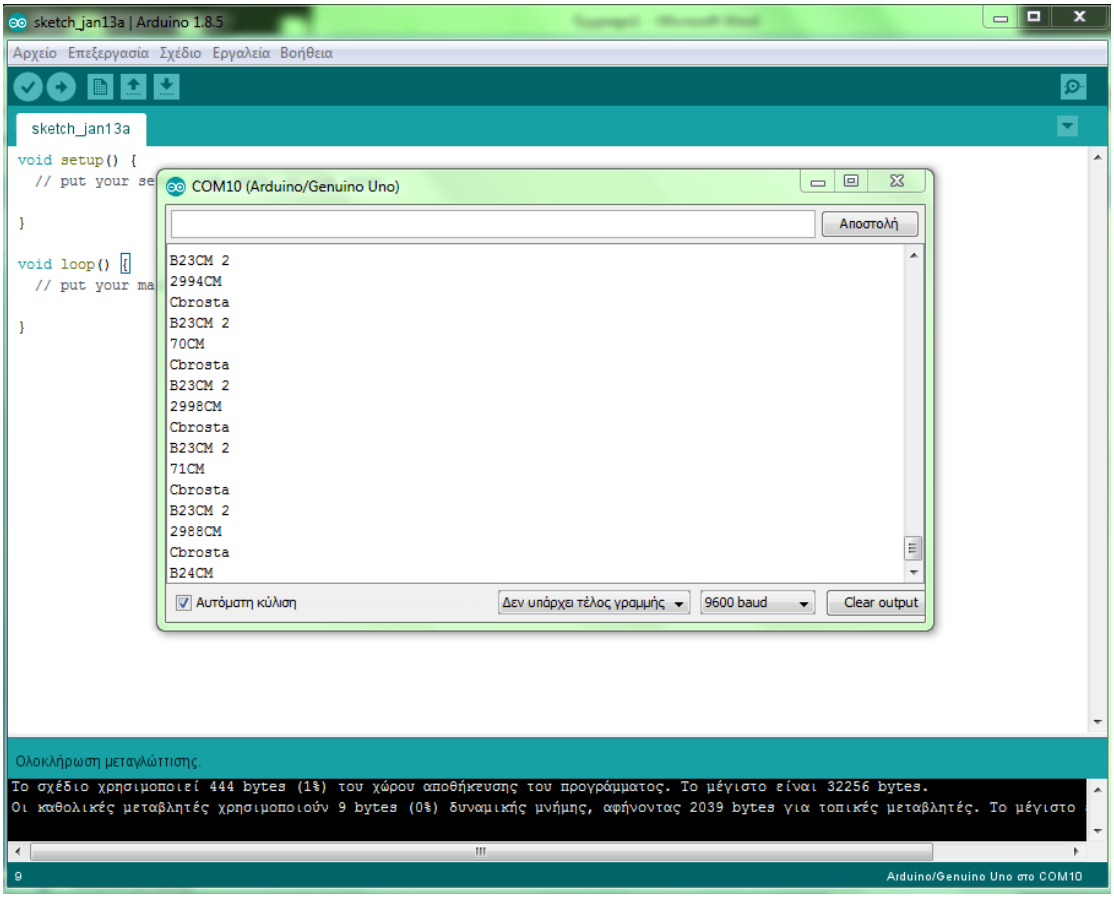



**Εικόνα 4.2 Θύρα εισόδου**

Μόλις φτιάξουμε τον κώδικα, ελέγχουμε αν είναι σωστός πατώντας το πλήκτρο «Επικύρωση» που υπάρχει επάνω αριστερά με το σύμβολο «√ ». Έπειτα πατάμε το πλήκτρο «Ανέβασμα» που υπάρχει δίπλα στο πλήκτρο επικύρωσης.

Απ' τη στιγμή που θα ανεβάσουμε τον κώδικα στον Arduino αυτός θα ξεκινήσει να τρέχει κανονικά. Παράλληλα μπορούμε να ελέγχουμε τα δεδομένα που στέλνονται στον επεξεργαστή μέσω της «Παρακολούθησης σειριακής θύρας» που βρίσκεται στην καρτέλα «Εργαλεία».

Στην παρακάτω εικόνα απεικονίζεται ένα παράδειγμα των δεδομένων που στέλνει και δέχεται ο Arduino.




**Εικόνα 4.3 Παρακολούθηση σειριακής**

## 5.2 Ανάλυση του κώδικα της παρούσας εργασίας

Όπως αναφέραμε προηγουμένως, το πρώτο σκέλος του κώδικα αποτελείται από μεταβλητές και βιβλιοθήκες. Στον συγκεκριμένο κώδικα δεν χρησιμοποιήθηκαν βιβλιοθήκες οπότε θα αναφερθούμε μόνο στις μεταβλητές.

```
1  // Variables...
2
3  // sensor 1 pins
4
5  int trigPin1 = 7;
6  int echoPin1 = 6;
7
8  // sensor 2 pins
9
10 int trigPin2 = 5;
11 int echoPin2 = 4;
12
13 // sensor 3 pins
14
15 int trigPin3 = 3;
16 int echoPin3 = 2;
17
18 int thesi;
19 int stamata;
20
21 // Motor A
22
23 int enA = 11;
24 int in1 = 12;
25 int in2 = 13;
26
27 // Motor B
28
29 int enB = 10;
30 int in3 = 8;
31 int in4 = 9;
32
33  long duration1, distance1;
34  long duration2, distance2;
35  long duration3, distance3;
36
```

Ο τύπος μεταβλητής «int» βγαίνει από την αγγλική λέξη integer που σημαίνει ακέραιος. Οι δυο κάθετες παύλες «//» χρησιμοποιούνται για τους σχολιασμούς στο περιβάλλον του κώδικα και ό, τι γραφτεί μετά από αυτές δεν συμπεριλαμβάνεται στην εκτέλεσή του.

Όσον αφορά τις μεταβλητές στις σειρές:

- **3 - 6**: Ορίζονται οι μεταβλητές για τον αισθητήρα απόστασης 1 όπου το trigPin1 = 7 δείχνει ότι ο ακροδέκτης που έρχεται από τη θύρα trig του αισθητήρα είναι συνδεδεμένος με την ψηφιακή θύρα 7 του Arduino. Το echoPin1 = 6 δείχνει ότι ο ακροδέκτης που έρχεται από την θύρα echo του αισθητήρα είναι συνδεδεμένος με την ψηφιακή θύρα 6 του Arduino.

- **8 - 11**: Ορίζονται οι μεταβλητές για τον αισθητήρα απόστασης 2 όπου το trigPin2 = 5 δείχνει ότι ο ακροδέκτης που έρχεται από την θύρα trigger του αισθητήρα είναι συνδεδεμένος με την ψηφιακή θύρα 5 του Arduino. Το echoPin2 = 4 δείχνει ότι ο ακροδέκτης που έρχεται από την θύρα echo του αισθητήρα είναι συνδεδεμένος με την ψηφιακή θύρα 4 του Arduino.
- **13 - 16**: Ορίζονται οι μεταβλητές για τον αισθητήρα απόστασης 3 όπου το trigPin3 = 3 δείχνει ότι ο ακροδέκτης που έρχεται από την θύρα trig του αισθητήρα είναι συνδεδεμένος με την ψηφιακή θύρα 3 του Arduino. Το echoPin3 = 2 δείχνει ότι ο ακροδέκτης που έρχεται από τη θύρα echo του αισθητήρα είναι συνδεδεμένος με την ψηφιακή θύρα 2 του Arduino.
- **18 - 19**: Ορίζονται οι μεταβλητές που θα είναι αρμόδιες στο να ενημερώνουν τον επεξεργαστή σε ποια θέση βρίσκεται το αυτοκινητάκι για να τρέξει τις ανάλογες εντολές.
- **21 - 25**: Ορίζονται οι μεταβλητές που θα είναι αρμόδιες για τον χειρισμό του ηλεκτροκινητήρα Α. Πιο συγκεκριμένα, (πρέπει) να αναφερθούν και οι θύρες του επεξεργαστή που είναι αρμόδιες γι' αυτόν τον σκοπό. Το enA της πλακέτας L298N που θα ορίζει την ταχύτητα του κινητήρα είναι συνδεδεμένο με την ψηφιακή - ψευδοαναλογική θύρα 11 του Arduino. Τα in1 και in2 της πλακέτας L298N που καθορίζουν τη φορά περιστροφής του κινητήρα είναι συνδεδεμένα στον Arduino στις ψηφιακές θύρες 12 και 13 αντίστοιχα.
- **27 - 31**: Ορίζονται οι μεταβλητές που θα είναι αρμόδιες για τον χειρισμό του ηλεκτροκινητήρα Β. Πιο συγκεκριμένα, (πρέπει) να αναφερθούν και οι θύρες του επεξεργαστή που είναι αρμόδιες γι' αυτόν τον σκοπό. Το enB της πλακέτας L298N που θα ορίζει την ταχύτητα του κινητήρα είναι συνδεδεμένο με την ψηφιακή - ψευδοαναλογική θύρα 10 του Arduino. Τα in3 και in4 της πλακέτας L298N που καθορίζουν τη φορά περιστροφής του κινητήρα είναι συνδεδεμένα στον Arduino στις ψηφιακές θύρες 8 και 9 αντίστοιχα.
- **33 - 35**: Ορίζονται οι μεταβλητές διάρκειας (duration1, duration2, duration3) και απόστασης (distance1, distance2, distance3) για τη βοήθεια λήψης τιμών απόστασης για τους αισθητήρες 1, 2 και 3 αντίστοιχα.

Παρακάτω θα δούμε τι ορίζουμε και πώς στη συνάρτηση setup.

```
37
38  void setup() {
39    Serial.begin(9600);
40
41     pinMode(trigPin1, OUTPUT);
42     pinMode(echoPin1, INPUT);
43     pinMode(trigPin2, OUTPUT);
44     pinMode(echoPin2, INPUT);
45     pinMode(trigPin3, OUTPUT);
46     pinMode(echoPin3, INPUT);
47     pinMode(enA, OUTPUT);
48     pinMode(in1, OUTPUT);
49     pinMode(in2, OUTPUT);
50     pinMode(enB, OUTPUT);
51     pinMode(in3, OUTPUT);
52     pinMode(in4, OUTPUT);
53
54  }
```

Όταν ορίζουμε μια συνάρτηση την "κλείνουμε" πάντα σε αγκύλες ({....}) όπως φαίνεται και στις σειρές 38 και 54. Στη setup αναφέρονται οι θύρες του Arduino που είναι δεσμευμένες και ορίζουμε αν είναι είσοδοι ή έξοδοι. Αυτό επιτυγχάνεται με τη βοήθεια της εντολής «pinMode();» η οποία συντάσσεται με πολύ εύκολο τρόπο: «pinMode(Pin, Mode);» όπου το pin είναι η δεσμευμένη θύρα που έχουμε ονομάσει στις μεταβλητές παραπάνω και το Mode είναι η κατάσταση στην οποία το ορίζουμε (INPUT , OUTPUT).

Ανάλυση ανά σειρά:

- **39**: Η εντολή «Serial.begin();» Ορίζει τον ρυθμό δεδομένων σε bits ανά δευτερόλεπτο (baud) για τη μετάδοση σειριακών δεδομένων. Για επικοινωνία με τον υπολογιστή, χρησιμοποιείται μία από αυτές τις ταχύτητες: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ή 115200.

- **41 - 46**: Ορίζονται οι θύρες για τους αισθητήρες 1, 2 και 3 σαν είσοδοι και έξοδοι με την εντολή «pinMode();». Όπως ξέρουμε (είναι γνωστό) στους αισθητήρες Ultrasonic το trig χρησιμοποιείται σαν έξοδος και το echo σαν είσοδος στον επεξεργαστή.
- **47 - 52**: Ορίζονται οι θύρες για τους ηλεκτροκινητήρες Α και Β σαν έξοδοι με τη βοήθεια πάλι της εντολής «pinMode();».

Το επόμενο στάδιο είναι η ανάλυση των διαδικασιών που δημιουργήθηκαν έτσι ώστε να γίνει καλύτερη και ευκολότερα διακριτή η δομή του κώδικα.

```
55  // sensor 1 on
56  void UltraSonic1On(){
57    digitalWrite(trigPin1,HIGH);
58    delayMicroseconds(1000);
59    digitalWrite(trigPin1, LOW);
60    duration1=pulseIn(echoPin1,HIGH);
61    distance1 =(duration1/2)/29.1;
62    Serial.print(distance1);
63    Serial.println("CM");
64    delay(50);
65  }
66  // sensor 1 off
67  void UltraSonic1Off(){
68     digitalWrite(trigPin1,LOW);
69    delayMicroseconds(1000);
70    digitalWrite(trigPin1, LOW);
71    duration1=pulseIn(echoPin1,LOW);
72    distance1 =(duration1/2)/29.1;
73    Serial.print(distance1);
74    Serial.println("CM");
75    delay(50);
76  }
```

Στις σειρές 56 - 65 είναι η διαδικασία ενεργοποίησης του αισθητήρα 1.

Στη σειρά 56 είναι η ονομασία που δίνεται σε αυτή τη διαδικασία με τη χρήση της λέξης **«void»** μπροστά.

Στη σειρά 57 δίνουμε την εντολή να ενεργοποιηθεί το trigger του αισθητήρα και να στείλει παλμό. Αυτό επιτυγχάνεται με την εντολή **«digitalWrite();»** στην οποία αν δοθεί η τιμή LOW ή HIGH δίνει στη θύρα που της έχουμε ορίσει τιμές 0V ή 5V αντίστοιχα. Η εντολή συντάσσεται ως εξής **«digitalWrite(Pin , Value);**»

Στη σειρά 58 με τη χρήση της εντολής **«delayMicroseconds();»** καθορίζουμε τον χρόνο σε microseconds που θα στέλνει παλμό το trig του αισθητήρα.

Στη σειρά 59 κλείνουμε το trigger του αισθητήρα με την εντολή **«digitalWrite();»** βάζοντας την τιμή LOW.

Στη σειρά 60 με την εντολή **«pulseIn();»** (η οποία συντάσσεται με τον ίδιο τρόπο όπως η **«digitalWrite();»**) η θύρα echo του αισθητήρα ενεργοποιείται και θα δώσει την τιμή της διάρκειας που έκανε να φτάσει ο παλμός από το trigger

στο echo. Αυτή την τιμή θα την ονομάσουμε duration1, η οποία έχει συμπεριληφθεί στη δήλωση μεταβλητών παραπάνω.

Στη σειρά 61 έχουμε το αποτέλεσμα της πράξης **[(duration1/2)/29.1]** το οποίο με μια ματιά είναι η διάρκεια που έκανε να φτάσει ο παλμός από το trigger στο echo διαιρώντας με το 2 ,διότι χρειαζόμαστε τη μισή διάρκεια, και τέλος το αποτέλεσμα το διαιρούμε με τον σταθερό αριθμό 29,1. Το αποτέλεσμα που θα μας δώσει αυτή η πράξη είναι η απόσταση που χρειαζόμαστε και την ονομάσαμε **distance1**.

Στη σειρά 62 ζητάμε να μας εμφανίζει στο παράθυρο σειριακής παρακολούθησης την τιμή distance1. Αυτό επιτυγχάνεται με την εντολή **Serial.print();**.

Στη σειρά 63 ζητάμε η τιμή που θα διαβάζει ο επεξεργαστής να είναι σε εκατοστά (cm).

Τέλος, στη σειρά 64 με την εντολή **«delay();»** ζητάμε πόσο συχνά θέλουμε να διαβάζει απόσταση ο αισθητήρας (η μέτρηση του χρόνου γίνεται σε microseconds).

Στις σειρές 67 - 76 αναφέρεται η διαδικασία απενεργοποίησης του αισθητήρα 1.

Ακολουθείται η ίδια διαδικασία με αυτήν που παρουσιάστηκε παραπάνω με μόνη διαφορά ότι στην πρώτη εντολή **«digitalWrite();»** για το trigger αντικαθιστούμε την τιμή HIGH με την τιμή LOW έτσι ώστε να σταματήσει να στέλνει παλμό και αντίστοιχα γίνεται η ίδια αλλαγή και στην σειρά 71 για το echo του αισθητήρα ώστε να μην μπορεί να διαβάσει παλμό.

Η ίδια διαδικασία ακολουθείται παρακάτω για τους αισθητήρες 2 και 3 αντίστοιχα. Με μόνη διαφορά ότι για τον αισθητήρα 3 δε χρειάστηκε να ορίσουμε διαδικασία απενεργοποίησης.

```
77  // sensor 2 on
78  void UltraSonic2On(){
79    digitalWrite(trigPin2,HIGH);
80    delayMicroseconds(1000);
81    digitalWrite(trigPin2, LOW);
82    duration2=pulseIn(echoPin2,HIGH);
83    distance2 =(duration2/2)/29.1;
84    Serial.print(distance2);
85    Serial.println("CM 2");
86    delay(50);
87  }
88  //sensor 2 off
89  void UltraSonic2Off(){
90     digitalWrite(trigPin2,LOW);
91    delayMicroseconds(1000);
92    digitalWrite(trigPin2, LOW);
93    duration2=pulseIn(echoPin2,LOW);
94    distance2 =(duration2/2)/29.1;
95    Serial.print(distance2);
96    Serial.println("CM");
97    delay(50);
98  }
99  // sensor 3 on
100 void UltraSonic3On()
101 {
102   digitalWrite(trigPin3,HIGH);
103   delayMicroseconds(1000);
104   digitalWrite(trigPin3, LOW);
105   duration3=pulseIn(echoPin3,HIGH);
106   distance3 =(duration3/2)/29.1;
107   Serial.print(distance3);
108   Serial.println("CM 2");
109   delay(50);
110 }
```

Στη συνέχεια θα αναλύσουμε τις διαδικασίες που ορίστηκαν για την κίνηση και την κατεύθυνση του αυτοκινήτου.

```
111 // go forward
112   void forward(){
113   digitalWrite(in1, HIGH);
114   digitalWrite(in2, LOW);
115   analogWrite(enA,120);
116
117   }
118   // go back
119   void backward(){
120   digitalWrite(in1, LOW);
121   digitalWrite(in2, HIGH);
122   analogWrite(enA, 120);
123   }
```

Στις σειρές 111 - 117 αναφέρεται η διαδικασία με την οποία το αυτοκίνητο κινείται προς τα εμπρός και στις 118 - 123 η διαδικασία με την οποία κινείται προς τα πίσω.

Οπως αναφέραμε και πιο πάνω για τον τρόπο λειτουργίας του ηλεκτροκινητήρα Α είναι αρμόδιες οι θύρες enA, in1 και in2. Το enA είναι συνδεδεμένο σε μια ψηφιακή θύρα PWM του Arduino έτσι ώστε να καθίσταται δυνατή η ρύθμιση του ρεύματος που θα δίνεται για την περιστροφή του

κινητήρα. Αυτή η ρύθμιση της τάσης γίνεται με την χρήση της εντολής **«analogWrite();»** η οποία βρίσκεται στη σειρά 115 και 122. Η σύνταξή της γίνεται ως εξής, «**analogWrite(Pin, value)»** η τιμή value θα πρέπει να είναι από 0 (0V) έως 255 (5V).

Για την αλλαγή της κατεύθυνσης χρησιμοποιούμε τις συναρτήσεις στις σειρές 113-114 και 120-121. Χρησιμοποιώντας την εντολή **«digitalWrite();» ,** με τον τρόπο που αναλύσαμε σε προηγούμενο παράδειγμα, αλλάζοντας τις τιμές HIGH και LOW

Παρακάτω με τον ίδιο τρόπο ρυθμίζουμε την τάση και τη φορά περιστροφής του κινητήρα Β. Σειρές 125-129 για στροφή του αυτοκινήτου αριστερά και 131-135 για στροφή δεξιά.

```
124   // stearing left
125   void left(){
126   digitalWrite(in3, LOW);
127   digitalWrite(in4, HIGH);
128   analogWrite(enB, 210);
129   }
130   // stearing right
131   void right(){
132   digitalWrite(in3, HIGH);
133   digitalWrite(in4, LOW);
134   analogWrite(enB, 220);
135   }
```

Τέλος η διαδικασία που σταματάει τη λειτουργία των κινητήρων, μηδενίζοντας την τάση στις θύρες enA και enB.

```
136   // car stops
137   void brake(){
138   analogWrite(enA, 0);
139   analogWrite(enB, 0);
140
141   }
```

Παρακάτω θα αναλύσουμε τη ροή του κύριου προγράμματος που θα τρέχει ο επεξεργαστής.

```
143  void loop()
144  {
145    thesi=0;
146    stamata=0;
147     while(thesi<5)
148     {
149
150    Serial.print(thesi);
151    UltraSonic1On();
152    if ((distance1<20)&&(thesi==0))
153    {
154      forward();
155
156    }
157    else if ((distance1>20)&&(thesi==0))
158    {
159      thesi=1;
160      while(thesi==1)
161       {
162        UltraSonic2On();
163         UltraSonic1On();
164         if (stamata==0)
165         {
166          forward();
167         }
168
169       if ((distance1>20)&&(distance2>20)&&(thesi==1))
170       {
171        forward();
172        stamata=1;
173       }
174       else if ((distance1<20)&&(distance2<20)&&(stamata==1))
175       {
176            brake();
177            stamata=2;
178
179       }
```

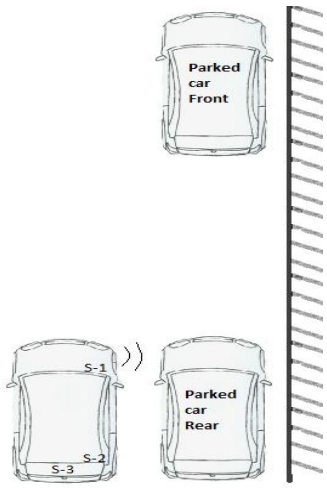
Ξεκινώντας ορίζουμε τις μεταβλητές **«thesi»** και **«stamata»** ίσες με το μηδέν. Αυτές οι μεταβλητές θα χρησιμοποιηθούν για να μπορεί ο Arduino να γνωρίζει σε ποια θέση βρίσκεται το αυτοκίνητο και να δώσει τις κατάλληλες εντολές.

Ξεκινώντας λοιπόν στη σειρά 147 με τη συνάρτηση **«while()»** η οποία για όσο θα διαβάζει την τιμή της μεταβλητής **«thesi»** μικρότερη από 5 θα εκτελεί το πρόγραμμα που βρίσκεται μέσα σε αυτήν.

Στη σειρά 151 καλούμε την διαδικασία που έχουμε ορίσει για την ενεργοποίηση το αισθητήρα 1 έτσι ώστε να ξεκινήσει να μετράει τις αποστάσεις που έχει στα δεξιά του το αυτοκίνητο.

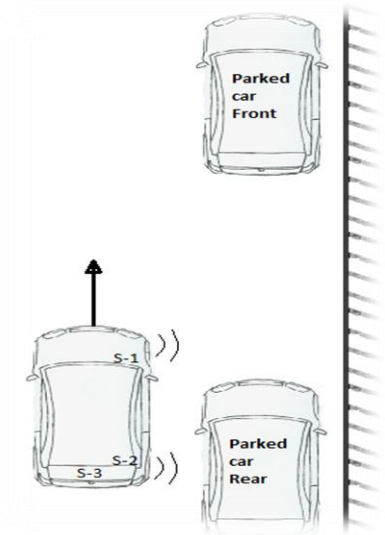
Στη σειρά 152 χρησιμοποιούμε την συνάρτηση **«if()»** , για να ορίσουμε στον Arduino ότι σε περίπτωση που θα διαβάζει την απόσταση του αισθητήρα 1

μικρότερη από 20cm και η μεταβλητή **thesi** είναι ίση με το μηδέν να κινήσει το αυτοκίνητο εμπρός καλώντας την διαδικασία **«forward();»**.



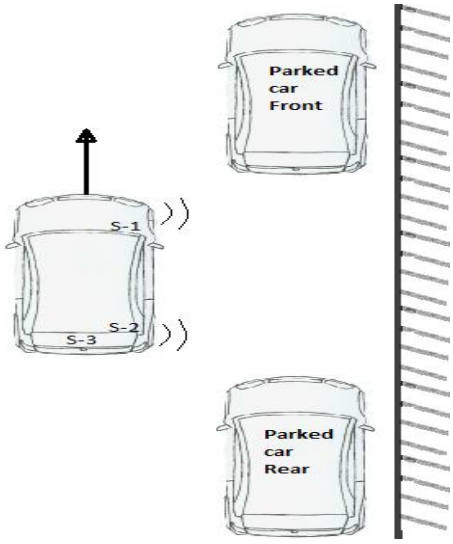

Στη σειρά 157 χρησιμοποιούμε την συνάρτηση **«else if()»**, για να ορίσουμε ότι σε περίπτωση που η τιμή της μεταβλητής **thesi** είναι 0 και ο αισθητήρας 1 διαβάσει απόσταση μεγαλύτερη από 20cm θα οριστεί η τιμή **thesi** ίση με 1.

Στη σειρά 160 χρησιμοποιούμε τη συνάρτηση **«while()»** για να καθορίσουμε ότι για όσο η **«thesi»** θα είναι ίση με 1, θα είναι ενεργοποιημένοι και οι 2 αισθητήρες απόστασης και το αυτοκίνητο θα ξεκινήσει την αναζήτηση του χώρου στάθμευσης.

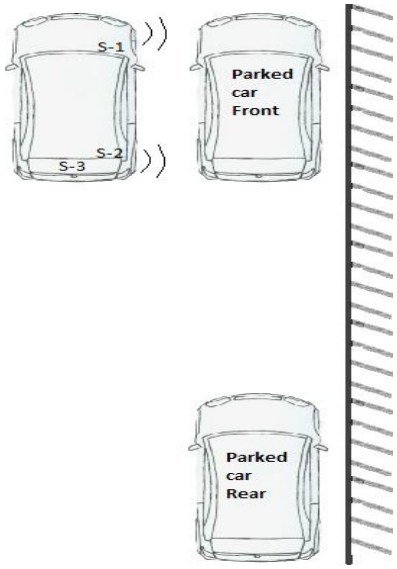


Συνεχίζουμε στη σειρά 164 χρησιμοποιώντας πάλι την συνάρτηση **«if()»** η οποία ελέγχει αν η μεταβλητή **«stamata»** είναι ίση με 0 και δίνει την εντολή στο αυτοκίνητο να συνεχίσει να κινείται προς τα εμπρός.

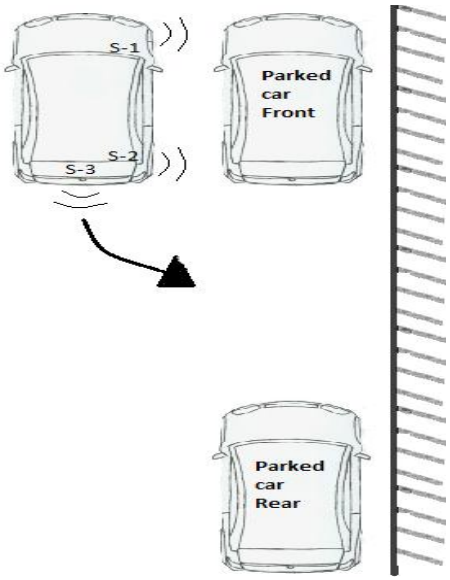
Έχοντας πλέον ενεργοποιημένους και τους δύο αισθητήρες απόστασης, στη σειρά 169 χρησιμοποιούμε τη συνάρτηση **«if()»** ώστε να ελέγξει, εάν οι αποστάσεις των αισθητήρων 1 και 2 είναι μεγαλύτερες από 20cm και η τιμή της μεταβλητής «**thesi»** είναι 1. Εάν ισχύει αυτή η συνθήκη τότε το αυτοκίνητο έχοντας αναγνωρίσει τον χώρο στάθμευσης, θα συνεχίσει την πορεία του προς τα εμπρός και η μεταβλητή **«stamata»** θα πάρει την τιμή 1.




Στη σειρά 174 χρησιμοποιώντας την συνάρτηση **«else if()»** η οποία ελέγχει εάν οι αποστάσεις που διαβάζουν οι αισθητήρες 1 και 2 είναι μικρότερες από 20cm και η μεταβλητή **«stamata»** είναι ίση με 1. Σ αυτή την περίπτωση το αυτοκίνητο θα σταματήσει και η μεταβλητή **«stamata»** θα πάρει την τιμή 2.




Εφόσον λοιπόν το αυτοκίνητο έχει αναγνωρίσει τον χώρο στον οποίο θα παρκάρει και έχει σταματήσει τη διαδικασία ανεύρεσης της θέσης, είναι έτοιμο να ξεκινήσει τη διαδικασία παρκαρίσματος η οποία βρίσκεται παρακάτω.



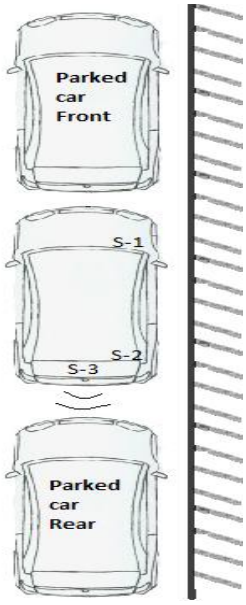
```
180         while(stamata==2)
181          {
182             UltraSonic2On();
183             UltraSonic1On();
184             UltraSonic3On();
185             if(distance3>=60)
186             {
187               right();
188               backward();
189             }
190             else if((distance3<=60)&&(distance3>15))
191             {
192               left();
193               backward();
194             }
195             else if((distance3<15)||(distance1=distance2))
196             {
197               brake();
198               stamata=3;
199             }
```

Ξεκινάει λοιπόν στη σειρά 180 τη διαδικασία του παρκαρίσματος, χρησιμοποιώντας τη συνάρτηση **«while()»** για να ορίσει ότι για όσο η μεταβλητή **«stamata»** θα είναι ίση με 2 θα τρέχει η παρακάτω διαδικασία. Ενεργοποιείται για αρχή ο αισθητήρας απόστασης 3 ο οποίος βρίσκεται στο πίσω μέρος του αυτοκινήτου.

Στη σειρά 185 χρησιμοποιώντας την συνάρτηση **«if()»** καθορίζουμε ότι στην περίπτωση που η απόσταση που θα διαβάσει ο αισθητήρας 3 είναι μεγαλύτερη ή ίση των 60cm να στρίψει δεξιά και να κάνει πίσω.

Στη σειρά 190 ορίζεται για τον αισθητήρα 3 μια απόσταση από 60cm μέχρι 15cm. Όσο λοιπόν ο αισθητήρας θα διαβάζει τιμές εντός αυτού του εύρους θα κινείται πίσω και αριστερά.

Τέλος στη σειρά 195, μόλις η απόσταση που θα διαβάσει ο αισθητήρας 3 είναι μικρότερη από 15cm ή οι αποστάσεις που διαβάζουν οι αισθητήρες 1 και 2 είναι ίσες το αυτοκίνητο θα σταματήσει καλώντας την διαδικασία **«brake();»** και θα δώσει στην μεταβλητή **«stamata»** την τιμή 3.




```
200                 while(stamata==3)
201                 {
202                   UltraSonic2On();
203                   UltraSonic1On();
204
205                   if (distance1>distance2)
206                   {
207                   right();
208                   forward();
209                   }
210                   else if (distance1<distance2)
211                   {
212                    left();
213                    forward();
214                   }
215                   else if (distance1=distance2)
216                   {
217                    brake();
218                   }
219                 }
220               }
221             }
222           }
223
224       }
225
226   }
```

Τέλος στη σειρά 200 , χρησιμοποιώντας τους δύο πλαϊνούς αισθητήρες απόστασης 1 και 2, ξεκινάει η διαδικασία με την οποία το αυτοκίνητο τοποθετείται σωστά στον χώρο στάθμευσης.

## Συμπεράσματα

Στην πράξη η εγκατάσταση ενός συστήματος αυτόνομου παρκαρίσματος δεν έχει γίνει ακόμη τόσο προσιτή μιας και χρειάζεται να υπάρχει πλήρης έλεγχος του οχήματος κατά τη διαδικασία ελιγμών παρκαρίσματος. Αυτό προϋποθέτει την ύπαρξη αυτόματου κιβωτίου ταχυτήτων και όχι μηχανικού έτσι ώστε να μπορεί να γίνει η εναλλαγή των ταχυτήτων και η ρύθμιση της κίνησης χωρίς να επεμβαίνει ο οδηγός του οχήματος. Η εγκατάσταση των συγκριμένων συστημάτων ακόμη και στις μικρές κατηγορίες θα ανέβαζε αισθητά το κόστος αγοράς τους με αποτέλεσμα τη μείωση του αγοραστικού κοινού σε αυτές τις κατηγορίες.

Εμείς προσομοιώσαμε το σύστημα αυτόματου παρκαρίσματος χρησιμοποιώντας προσιτά υλικά και κάνοντας τη διαδικασία παρκαρίσματος λίγο πιο διασκεδαστική απ ότι είναι στην πραγματικότητα.

Η παρούσα κατασκευή θα μπορούσε με ορισμένες βελτιώσεις, όπως προσθέτοντας κάποιους ακόμη αισθητήρες απόστασης για μεγαλύτερη ακρίβεια να χρησιμοποιηθεί και από σχολές οδηγών στο πλαίσιο της θεωρητικής εκπαίδευσης νέων οδηγών.

## Βιβλιογραφία


**Ιστοσελίδες**

https://deltahacker.gr/arduino-intro/

https://www.arduino.cc/en/Guide/Windows

https://www.arduino.cc/reference/en/#structure

https://www.arduino.cc/en/Tutorial/Memory

https://store.arduino.cc/arduino-uno-rev3

https://www.arduino.cc/en/Guide/Environment

https://dronebotworkshop.com/dc-motors-l298n-h-bridge/

https://dronebotworkshop.com/hc-sr04-ultrasonic-distance-sensor-arduino/

http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

http://www.ardumotive.com/how-to-use-the-hc-sr04-ultrasonic-sensor-en.html

**Συγγράμματα**

Hubbard, J.R. (2008). *Προγραμματισμός με C++.* (Δ. Τσιλογιάννης, μεταφρ.). Αθήνα: Κλειδάριθμος.

Πουλάκης , Ε. (2015). *Προγραμματίζοντας με τον μικροελεγκτή Arduino*. Ε. Πουλάκης: Ηράκλειο.

Χατζιγιαννάκης, Ν. (2008). *Η γλώσσα C++ σε βάθος.* Αθήνα: Κλειδάριθμος.

## Παράρτημα

```
// Variables...

// sensor 1 pins

int trigPin1 = 7;
int echoPin1 = 6;

// sensor 2 pins

int trigPin2 = 5;
int echoPin2 = 4;

// sensor 3 pins

int trigPin3 = 3;
int echoPin3 = 2;
int thesi;
int stamata;

// Motor A
int enA = 11;
int in1 = 12;
int in2 = 13;

// Motor B

int enB = 10;
```

```
int in3 = 8;
int in4 = 9;

 long duration1, distance1;
 long duration2, distance2;
 long duration3, distance3;


void setup() {
  Serial.begin(9600);

   pinMode(trigPin1, OUTPUT);
   pinMode(echoPin1, INPUT);
   pinMode(trigPin2, OUTPUT);
   pinMode(echoPin2, INPUT);
   pinMode(trigPin3, OUTPUT);
   pinMode(echoPin3, INPUT);
   pinMode(enA, OUTPUT);
   pinMode(in1, OUTPUT);
   pinMode(in2, OUTPUT);
   pinMode(enB, OUTPUT);
   pinMode(in3, OUTPUT);
   pinMode(in4, OUTPUT);
  }
// sensor 1 on
void UltraSonic1On(){
  digitalWrite(trigPin1,HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigPin1, LOW);
```

```
 duration1=pulseIn(echoPin1,HIGH);
 distance1 =(duration1/2)/29.1;
 Serial.print(distance1);
 Serial.println("CM");
 delay(50);
}
// sensor 1 off
void UltraSonic1Off(){
  digitalWrite(trigPin1,LOW);
 delayMicroseconds(1000);
 digitalWrite(trigPin1, LOW);
 duration1=pulseIn(echoPin1,LOW);
 distance1 =(duration1/2)/29.1;
 Serial.print(distance1);
 Serial.println("CM");
 delay(50);
}
// sensor 2 on
void UltraSonic2On(){
 digitalWrite(trigPin2,HIGH);
 delayMicroseconds(1000);
 digitalWrite(trigPin2, LOW);
 duration2=pulseIn(echoPin2,HIGH);
 distance2 =(duration2/2)/29.1;
 Serial.print(distance2);
 Serial.println("CM 2");
 delay(50);
}
//sensor 2 off
```

```
void UltraSonic2Off(){
  digitalWrite(trigPin2,LOW);
 delayMicroseconds(1000);
 digitalWrite(trigPin2, LOW);
 duration2=pulseIn(echoPin2,LOW);
 distance2 =(duration2/2)/29.1;
 Serial.print(distance2);
 Serial.println("CM");
 delay(50);
}
// sensor 3 on
void UltraSonic3On()
{
 digitalWrite(trigPin3,HIGH);
 delayMicroseconds(1000);
 digitalWrite(trigPin3, LOW);
 duration3=pulseIn(echoPin3,HIGH);
 distance3 =(duration3/2)/29.1;
 Serial.print(distance3);
 Serial.println("CM 2");
 delay(50);
}
// go forward
 void forward(){
 digitalWrite(in1, HIGH);
 digitalWrite(in2, LOW);
 analogWrite(enA,120);

 }
```

```
 // go back
 void backward(){
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
 analogWrite(enA, 120);
 }
 // stearing left
 void left(){
 digitalWrite(in3, LOW);
 digitalWrite(in4, HIGH);
 analogWrite(enB, 220);
 }
 // stearing right
 void right(){
 digitalWrite(in3, HIGH);
 digitalWrite(in4, LOW);
 analogWrite(enB, 220);
 }
 // car stops
 void brake(){
 analogWrite(enA, 0);
 analogWrite(enB, 0);

 }

void loop()
{
 thesi=0;
 stamata=0;
```

```
while(thesi<5)
{
   Serial.print(thesi);
   UltraSonic1On();

    if ((distance1<20)&&(thesi==0))
    {
     forward();

    }
   else if ((distance1>20)&&(thesi==0))
   {
     thesi=1;
     while(thesi==1)
     {
       UltraSonic2On();
       UltraSonic1On();
       if (stamata==0)
       {
        forward();
       }

       if ((distance1>20)&&(distance2>20)&&(thesi==1))
       {
         forward();
         stamata=1;
       }
       else if ((distance1<20)&&(distance2<20)&&(stamata==1))
       {
```

```
  brake();
  stamata=2;
 }

while(stamata==2)
 {
  UltraSonic2On();
  UltraSonic1On();
  UltraSonic3On();
  if(distance3>=60)
  {
   right();
   backward();
  }
  else if((distance3<=60)&&(distance3>10))
  {
   left();
   backward();
  }
  else if((distance3<15)||(distance1=distance2))
  {
   brake();
   stamata=3;
  }
  while(stamata==3)
  {
   UltraSonic2On();
   UltraSonic1On();
```

```
            if (distance1>distance2)
            {
            right();
            forward();
            }
            else if (distance1<distance2)
            {
             left();
             forward();
            }
            else if (distance1=distance2)
            {
             brake();
            }
           }
         }
       }
     }
   }
}
```