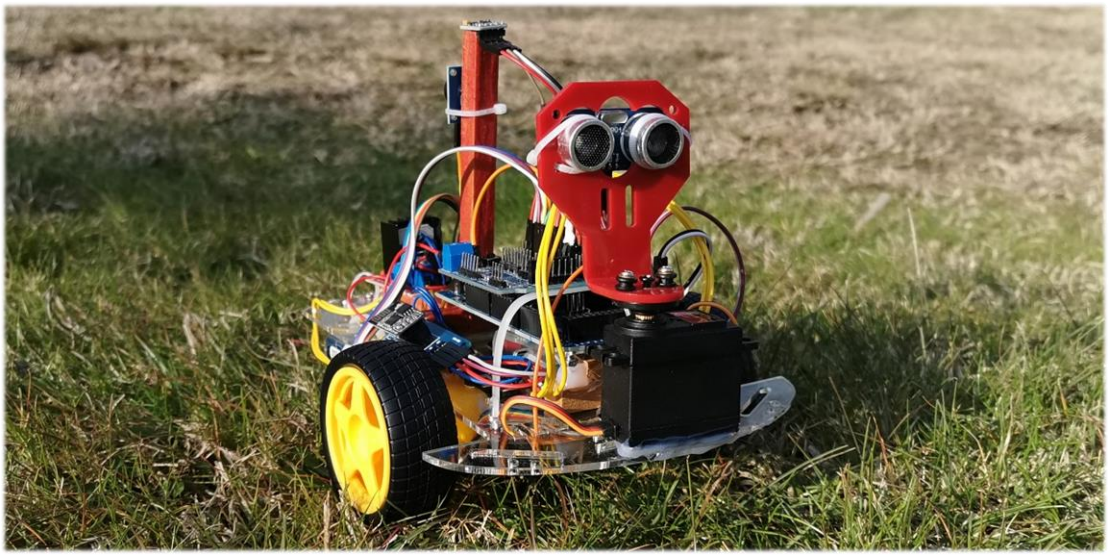
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ (ΔΙ.ΠΑ.Ε.)

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΣΕΡΡΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΣΤΗ ΡΟΜΠΟΤΙΚΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΑΠΟΦΥΓΗΣ ΕΜΠΟΔΙΩΝ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΑΥΤΟΚΙΝΟΥΜΕΝΟ ΡΟΜΠΟΤ ΜΕ ΤΗΝ ΜΕΘΟΔΟ VECTOR FIELD HISTOGRAM – VFH



Φοιτητής:

Αλτιντζής Γεώργιος (Α.Μ.: 64)

Επιβλέπων καθηγητής:

Καζαρλής Σπύρος, Καθηγητής

ΣΕΡΡΕΣ, ΦΕΒΡΟΥΑΡΙΟΣ 2023

ΠΡΟΛΟΓΟΣ

Αυτή η διπλωματική εργασία έχει ως σκοπό την μελέτη και παρουσίαση της μεθόδου Vector Field Histogram (VFH). Για να γίνει αυτό, κατασκευάστηκε ένα αυτόνομο ρομπότ το οποίο μπορεί να κινείται και να αποφασίζει μόνο του για την καλύτερη διαδρομή χρησιμοποιώντας τη μέθοδο VFH.

Θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Καζαρλή Σπύρο για την πολύτιμη και καθοριστική συμβολή του από τη θέση του επιβλέποντα της διπλωματικής μου εργασίας, τους γονείς μου Αναστάσιο και Γεωργία και πάνω από όλους τον Θεό που με βοήθησε να φτάσω ως αυτό το σημείο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Στα κεφάλαια που ακολουθούν θα περιγράψω την υλοποίηση αλγορίθμων αποφυγής εμποδίων με την μέθοδο «Ιστογράμματος Διανυσμάτων Πεδίου» (Vector Field Histogram ή VFH). Θα ξεκινήσω αναφερόμενος στην ιστορία των μικροελεγκτών. Έπειτα, θα περιγράψω τον μικροελεγκτή Arduino. Επίσης, θα παρουσιάσω αναλυτικά τους αισθητήρες και τις περιφερειακές συσκευές που χρησιμοποίησα στην κατασκευή μου μαζί με τις βιβλιοθήκες, τα προγράμματα και τις ρυθμίσεις που απαιτούνται για την ολοκληρωμένη διαμόρφωση και ορθή χρήση τους. Στο τέλος, θα μελετήσω την υλοποίηση της μεθόδου VFH σε ένα πραγματικό αυτοκινούμενο ρομπότ. Το ρομπότ ως πλατφόρμα υλοποίησης της μεθόδου VFH θα φέρει αισθητήρες αποστασιομέτρησης και θα ελέγχεται μέσω Arduino. Θα παρουσιάσω τις βασικές ρουτίνες χαμηλού επιπέδου για την κίνηση και τον έλεγχο των αισθητήρων, καθώς και τις ρουτίνες υψηλού επιπέδου για την υλοποίηση του αλγόριθμου VFH και localization.

ΚΕΦΑΛΑΙΟ 1: ΜΙΚΡΟΕΛΕΓΚΤΕΣ

Σχέση μικροελεγκτών με μικροεπεξεργαστές

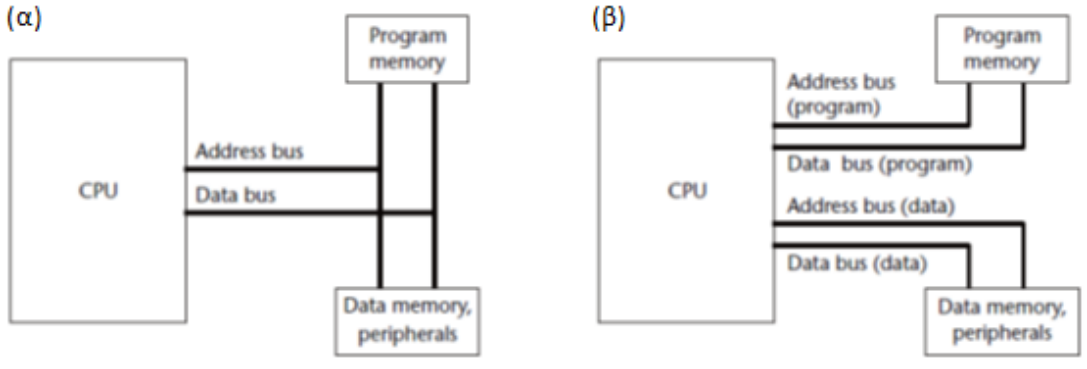
Ο μικροελεγκτής (microcontroller) ανήκει στους μικροεπεξεργαστές και λειτουργεί με πολύ λίγα παρελκόμενα επειδή ενσωματώνει διάφορες μονάδες. Τοποθετείται στα ενσωματωμένα συστήματα (embedded systems) ελέγχου μικρομεσαίου κόστους όπως για παράδειγμα στους αυτοματισμούς, στα ηλεκτρονικά προϊόντα, στις ηλεκτρικές μηχανές και στα οχήματα.

Οι σύγχρονοι μικροεπεξεργαστές μη ενσωματωμένων συστημάτων, όπως είναι οι μικροεπεξεργαστές των προσωπικών υπολογιστών, διαθέτουν υψηλή υπολογιστική ισχύ και το είδος της εργασίας διέπεται από τα εξαρτήματα που συνδέονται σε αυτούς. Οπότε, αυτοί οι μικροεπεξεργαστές δεν είναι εξειδικευμένοι σε αντίθεση με τους μικροελεγκτές που έχουν χαμηλή ισχύ, μηδαμινό πλήθος παρελκόμενων και εξειδικευμένη λειτουργία.

Οι μικροελεγκτές έχουν τα ακόλουθα πλεονεκτήματα:

- Αυξημένη αξιοπιστία λόγω απουσίας παρελκόμενων
- Αυτονομία επειδή ενσωματώνουν τα συστήματα που χρειάζονται (όπως είναι η μνήμη και η κάρτα γραφικών) και δεν συνδέονται εξωτερικά με αυτά
- Ελάχιστη ενεργειακή κατανάλωση
- Ελαχιστοποίηση διαστάσεων τελικών συσκευών που τους ενσωματώνουν
- Εύκολη ανάπτυξη λογισμικού για τις συσκευές που τους ενσωματώνουν γιατί απουσιάζουν τα εξωτερικά υποσυστήματα
- Μέγιστη φορητότητα
- Μικρό κόστος για τις συσκευές που τους περιέχουν
- Σχεδόν ανύπαρκτες ηλεκτρομαγνητικές παρεμβολές από αυτούς και σχεδόν καμία επιρροή από τις ηλεκτρομαγνητικές παρεμβολές αφού δεν συνδέονται εξωτερικά με άλλα εξαρτήματα και λειτουργούν με μειωμένη ισχύ
- Υψηλή διαθεσιμότητα σε αισθητήρες επειδή δεν ενσωματώνουν πολλά εξαρτήματα

Ως προς τη βασική αρχιτεκτονική μνήμης των μικροελεγκτών, αυτή είναι όμοια με την αντίστοιχη των συνηθισμένων μικροεπεξεργαστών. Ωστόσο, στους συνηθισμένους μικροεπεξεργαστές επικρατεί η αρχιτεκτονική τύπου Von Neumann-Princeton (α) όπου πρόγραμμα και δεδομένα ταυτίζονται ενώ στους μικροελεγκτές χρησιμοποιείται τακτικότερα η αρχιτεκτονική μνήμης τύπου Harvard (β) όπου πρόγραμμα και δεδομένα είναι εντελώς ανεξάρτητα.



Εικόνα 1: Διαγράμματα αρχιτεκτονικών μνήμης (πηγή: Σπύρος Καζαρλής)

Συνηθισμένα υποσυστήματα

Το κύκλωμα ενός κοινού μικροεπεξεργαστή μπορεί να αποτελείται μόνο από την Λογική και Αριθμητική Μονάδα (ALU), βασικούς καταχωρητές (registers), προσωρινή μνήμη RAM εξαιρετικά μεγάλης ταχύτητας (cache memory) και ελεγκτή μνήμης (memory controller). Για να δημιουργηθεί μία ολοκληρωμένη υπολογιστική διάταξη χρειάζονται και άλλα παρελκόμενα συστήματα όπως είναι τα εξής:

- Διάταξη αυτόνομων ψηφιακών εισόδων και εξόδων (PIO)
- Έναν ή πολλούς μετρητές (hardware timer-counter) που προκαλούν καθυστερήσεις ώστε να μετρούνται τα γεγονότα χρονικά και ποσοτικά
- Επόπτη αιτήσεων διακοπής (interrupt request controller) από τα εξωτερικά συστήματα
- Μη προγραμματιζόμενη μνήμη (όπως ROM, FLASH, EPROM κ.ά.) που ενσωματώνει το λογισμικό του συστήματος
- Μνήμη συγκράτησης τιμών από τις παραμέτρους λειτουργίας (όπως EEPROM ή NVRAM) που δύναται να αποθηκεύεται στον πυρήνα του μικροελεγκτή
- Ρολόι πραγματικού χρόνου (RTC) που λειτουργεί με αυτόνομη μπαταρία
- Σύστημα επανεκκίνησης (reset)
- Σύστημα επιτήρησης λειτουργίας (watchdog timer) το οποίο επανεκκινεί το σύστημα όταν αυτό κολλήσει (hang)
- Σύστημα επιτήρησης τροφοδοσίας (brown-out detection) το οποίο παρακολουθεί την τάση λειτουργίας και επανεκκινεί το συνολικό σύστημα όταν αυτή χαμηλώσει κάτω από ένα συγκεκριμένο όριο, επιτυγχάνοντας την προστασία των δεδομένων
- Σύστημα συνδετικής λογικής (glue logic) για την ένωση εξωτερικών μνημών και λοιπών παρελκομένων με τον δίαυλο δεδομένων (data bus) του επεξεργαστή
- Τοπικό ταλαντωτή για την διοχέτευση παλμών χρονισμού (clock)
- Υψηλό μέγεθος μνήμης RAM

Σε γενικές γραμμές, το σύνολο των μικροελεγκτών περιέχει πολλά από τα αναφερόμενα περιφερειακά με εξαιρέσεις όπως την ύπαρξη ή μη ενσωματωμένης μνήμης προγράμματος και το είδος της. Βάσει αυτών των στοιχείων μπορούμε να τους κατατάξουμε σε:

- Μικροελεγκτές απούσας μνήμης προγράμματος (ROM-less). Αυτοί συνδέονται πάντα με έναν παράλληλο δίαυλο δεδομένων (data bus) και αυτός, με τη σειρά του, συνδέεται με εξωτερικές μνήμες προγράμματος και RAM. Οι μικροελεγκτές αυτοί είναι κατάλληλοι για δυνατά υπολογιστικά συστήματα ελέγχου με υψηλές απαιτήσεις μνήμης.
- Μικροελεγκτές με μνήμη ROM η οποία περιέχει το πρόγραμμα εξαρχής (Mask ROM) ή το πρόγραμμα μπορεί να γραφτεί στην μνήμη μόνο μια φορά (One Time Programmable, OTP). Έχουν το πλεονέκτημα του πολύ μικρού κόστους όταν αποκτούνται μαζικά.
- Μικροελεγκτές με μνήμη FLASH που είναι επαναπρογραμματίσιμη. Πλέον ο προγραμματισμός της μνήμης πραγματοποιείται απευθείας στο κύκλωμα της ίδιας της ενσωματωμένης (embedded) εφαρμογής (δυνατότητα In Circuit Programming, ISP). Παλαιότερα αυτό γινόταν με μνήμη τύπου EPROM κάνοντας χρήση υπεριώδους ακτινοβολίας.

Επιπλέον λειτουργίες

Η τελική χρήση του μικροελεγκτή καθορίζει και τις επιπλέον λειτουργίες του όπως:

- Κύκλωμα προγραμματισμού (τύπου ISP) και ελέγχου (συχνά με τη θύρα JTAG). Είναι δυνατή η εγκατάσταση λογισμικού χωρίς να προϋπάρχει κάποιο προηγούμενο. Αυτό ωφελεί την αναγνώριση σφαλμάτων (bugs)
- Κύκλωμα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP). Με αυτό υπάρχει η δυνατότητα ενημέρωσης του λογισμικού από εξωτερική πηγή ή μέσω διαδικτύου. Για να υλοποιηθεί αυτό πρέπει να υπάρχει μη προγραμματίσιμο σταθερό τμήμα λογισμικού (bootstrap) εντός του μικροελεγκτή
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter ή DAC)
- Μία ή πολλές ασύγχρονες σειριακές πόρτες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART)
- Μονάδα κινητής υποδιαστολής (Floating Point Processing Unit ή FPU) γρηγορότερη από την ALU του επεξεργαστή
- Οθόνη υγρών κρυστάλλων (Liquid Crystal Display ή LCD)
- Πάνω από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter ή ADC)
- Σύγχρονες σειριακές πόρτες επικοινωνίας (I2C, SPI, Ethernet κ.ά.)
- Τελικά κυκλώματα για την χρήση, μετά από αναβάθμιση από υλικολογισμικό (firmware), των πολυπλοκότερων πρωτοκόλλων επικοινωνίας (CAN, HDLC, ISDN, ADSL κ.ά.)

Συνηθισμένοι τύποι

Οι μικροελεγκτές παράγονται πλέον μαζικά και εξειδικεύονται ώστε να ικανοποιήσουν τον ανταγωνισμό και την υψηλή τους απορρόφηση στις καθημερινές μας ηλεκτρικές και ηλεκτρονικές συσκευές:

- Μικροελεγκτές (των 4-bit ή, κυρίως, των 8-bit) οι οποίοι είναι φθηνοί, προωθούνται για την ικανοποίηση πολλών εφαρμογών και φέρουν ελάχιστους ακροδέκτες. Είναι αρκετά αυτόνομοι και η ενεργειακή τους κατανάλωση υπερβολικά χαμηλή. Το λογισμικό τους δεν είναι εύκολα αντιγράψιμο και η μνήμη τους δεν είναι επεκτάσιμη. Τέτοιοι μικροελεγκτές είναι οι PIC (της Microchip), οι AVR (της Atmel) και οι 8051 (της Intel, της Atmel κ.ά.).
- Μικροελεγκτές (των 8-bit ή, κυρίως, των 16 ή 32-bit) οι οποίοι είναι φθηνοί, προωθούνται για την ικανοποίηση πολλών εφαρμογών και φέρουν αρκετούς ακροδέκτες όπως θύρες UART, I2C, SPI ή CAN και μετατροπείς αναλογικού σήματος σε ψηφιακό και αντίστροφα. Η μνήμη τους είναι επεκτάσιμη. Τέτοιοι μικροελεγκτές τακτικά διαθέτουν οθόνη υγρών κρυστάλλων και πληκτρολόγιο. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.
- Μικροελεγκτές (κυρίως των 32-bit) οι οποίοι είναι μέτριου κόστους, προωθούνται για την ικανοποίηση πολλών εφαρμογών και φέρουν πολλούς ακροδέκτες. Είναι αρκετά αυτόνομοι, πολύ γρήγοροι, προορίζονται για την υποστήριξη πολύ μεγάλης μνήμης και διαθέτουν άριστη μεταφερσιμότητα λογισμικού (portability) όσον αφορά το λογισμικό τους μεταξύ των διάφορων κατασκευαστών.
- Μικροελεγκτές εξειδικευμένων εφαρμογών που φέρουν κυρίως συγκεκριμένα πρωτόκολλα επικοινωνίας υλοποιημένα μόνο σε hardware. Αυτού του τύπου οι μικροελεγκτές είναι ιδανικοί σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

Γλώσσες προγραμματισμού, εργαλεία ανάπτυξης και κατασκευαστές

Κάθε οικογένεια μικροελεγκτών μπορεί να διακριθεί και γίνει ευρέως αποδεκτή όταν προσφέρει ικανοποιητικά εργαλεία ανάπτυξης όπως είναι οι μεταφραστές και τα εργαλεία εκσφαλμάτωσης (debuggers). Τα εργαλεία αυτά δεν είναι μόνο το σχετικό λογισμικό αλλά και το αντίστοιχο υλικό όπως είναι οι προγραμματιστές της εσωτερικής μνήμης (μέσω θύρας JTAG ή USB) αλλά και ολοκληρωμένες πλακέτες (evaluation boards) με διάφορες εξόδους. Η δημιουργία των εργαλείων εμπίπτει όχι μόνο στους κατασκευαστές αυτούς καθ' αυτούς αλλά και σε ολόκληρες εταιρείες. Η κύρια γλώσσα προγραμματισμού των μικροελεγκτών είναι η C και η C++. Παλαιότερα κυριαρχούσε η Assembly η οποία εκτοπίζεται με την ανάπτυξη της τεχνολογίας και την χρηστική της δυσκολία.

Οι πιο γνωστοί κατασκευαστές μικροελεγκτών είναι οι εξής:

- ARM (δεν κατασκευάζει αλλά παραχωρεί δικαιώματα χρήσης του πυρήνα)
- Atmel
- Epson
- Freescale Semiconductor (πρώην Motorola)
- Hitachi
- Maxim (μετά την εξαγορά της Dallas)
- Microchip
- NEC
- Texas Instruments
- Toshiba

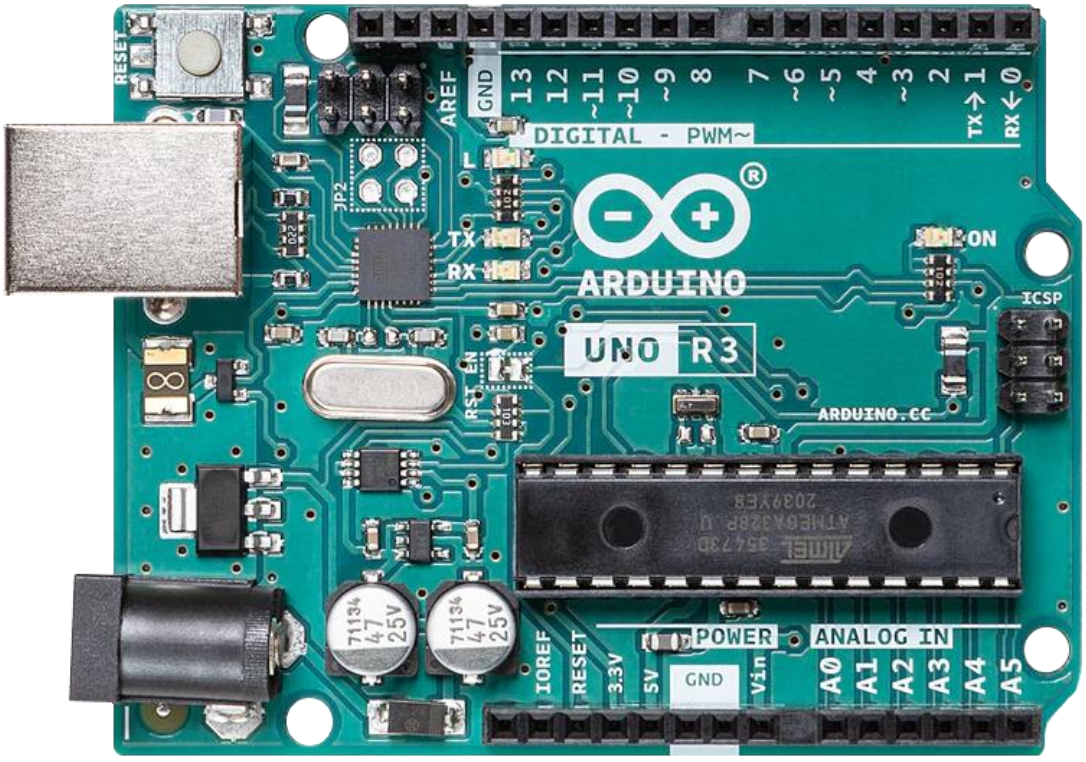
ΚΕΦΑΛΑΙΟ 2: ARDUINO

Εισαγωγή

Η εταιρεία Arduino παράγει πλακέτες ενσωματωμένων συστημάτων ανοικτής αρχιτεκτονικής και κώδικα. Το 2005 δημιούργησε την πλακέτα Arduino ως ένα υλικολογισμικό ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους το οποίο είναι προγραμματίσιμο με τη γλώσσα Wiring (μια γλώσσα προγραμματισμού ίδια με την C++ και ένα σετ από βιβλιοθήκες δημιουργημένες στην C++). Παράλληλα παρέχει στους χρήστες ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Arduino IDE).

Κάθε πλακέτα Arduino στηριζόταν παλαιότερα στον μικροελεγκτή ATmega8, πλέον στον μικροελεγκτή ATmega328 και πιο πρόσφατα στον ATmega168 της εταιρείας Atmel AVR. Περιέχει ένα ρυθμιστή τάσης των 5V και, συνήθως, έναν κρυσταλλικό ταλαντωτή των 16MHz. Ο μικροελεγκτής είναι προ-προγραμματισμένος με bootloader. Ο προγραμματισμός των πλακετών γίνεται κυρίως με αφαιρούμενο USB-to-Serial καλώδιο ή με RS-232 σειριακή σύνδεση. Αν χρησιμοποιηθούν τα παραδοσιακά εργαλεία των μικροελεγκτών τότε, αντί για το Arduino IDE, ο προγραμματισμός πραγματοποιείται AVR ISP.

Ο ATmega328 είναι ένας μικροελεγκτής 8-bit αρχιτεκτονικής επεξεργαστή AVR RISC, με αρχιτεκτονική μνήμης Harvard, απόδοση 20MIPS στα 20MHz. Διαθέτει SRAM 2 KB για μεταβλητές, μνήμη Flash EEPROM 32 KB για προγράμματα και επιπλέον EEPROM 1 KB για μόνιμη αποθήκευση μεταβλητών. Διαθέτει 23 ψηφιακές γραμμές I/O και 32 καταχωρητές. Διαθέτει 3 ευέλικτους χρονιστές, διακοπές (interrupts), σειριακές διεπαφές (interfaces) (USART, I2C, SPI), 10-bit μετατροπέα A/D πλήθους 6 καναλιών για ανάγνωση αναλογικών σημάτων και λειτουργεί στην τάση των 5V.



Εικόνα 2: Arduino Uno Rev3 (πηγή: Arduino Official Store)

Η πλακέτα **Arduino Uno Rev3**:

- Αναγνωρίζει ένα σετ 131 εντολών του ATmega328P που κάθε μία απαριθμείται από έναν αριθμό των 16-bit
- Διαθέτει 14 ψηφιακές ακίδες I/O (D0-D13) από τις οποίες οι 6 έχουν δυνατότητα εξόδου σήματος PWM (Pulse Width Modulation) και σημειώνονται στην πλακέτα με μία περισπωμένη «~»
- Διαθέτει USB θύρα για σύνδεση με Η/Υ μέσω της οποίας είναι η πλακέτα είναι προγραμματίσιμη. Η θύρα λειτουργεί στα 16MHz
- Διαθέτει τον ATmega328P που ήδη περιλαμβάνει πρόγραμμα bootloader μεγέθους 2KB το οποίο διευκολύνει την επικοινωνία με τον Η/Υ και το περιβάλλον Arduino IDE
- Εκτελεί πρόγραμμα σε γλώσσα μηχανής το οποίο έχει μεταφραστεί από την γλώσσα υψηλού επιπέδου μέσω του Arduino IDE
- Παρέχει 6 αναλογικές ακίδες I/O (A0-A5) ανάλυσης 10-bit (δηλαδή 1024 επιπέδων) μετατροπής αναλογικού σήματος σε ψηφιακό
- Περιέχει το ολοκληρωμένο ATmega16U2 που αναλαμβάνει την σειριακή επικοινωνία μέσω USB και εμφανίζεται στον Η/Υ ως εικονική σειριακή θύρα (virtual COM port)
- Τροφοδοτείται με τάση DC 6 – 20V (προτείνεται η τάση των DC 7 – 12V)

Πλακέτες

Οι περισσότερες πλακέτες του Arduino είναι εμπορικές και έτοιμες προς χρήση. Ωστόσο, τα διαγράμματα και όλες οι απαραίτητες πληροφορίες για το υλικό είναι στην διάθεση του κοινού και μπορούν να χρησιμοποιηθούν για την κατασκευή πλακετών από τον καθένα. Η πιο δημοφιλής πλακέτα είναι η Arduino Uno Rev3 και η επόμενη πιο δημοφιλής είναι η **Arduino Mega 2560 Rev3** η οποία και χρησιμοποιείται στην κατασκευή μου. Ως την στιγμή που γράφω αυτήν την εργασία, οι πλακέτες Arduino που έχουν ποτέ κατασκευαστεί είναι οι:

- Arduino Due
- Arduino Due without Headers
- Arduino Edge Control
- Arduino Leonardo with Headers
- Arduino Leonardo without Headers
- Arduino Mega 2560 Rev3
- Arduino Micro
- Arduino Micro without headers
- Arduino MKR GSM 1400
- Arduino MKR IoT Carrier
- Arduino MKR IoT Carrier Rev2
- Arduino MKR NB 1500
- Arduino MKR Vidor 4000
- Arduino MKR WAN 1300 (LoRa® connectivity)
- Arduino MKR WAN 1310
- Arduino MKR WiFi 1010
- Arduino MKR ZERO (I2S bus & SD for sound, music & digital audio data)
- Arduino Nano
- Arduino Nano 33 BLE
- Arduino Nano 33 BLE Sense
- Arduino Nano 33 BLE Sense Rev2
- Arduino Nano 33 BLE Sense Rev2 with headers
- Arduino Nano 33 BLE Sense with headers
- Arduino Nano 33 BLE with headers
- Arduino Nano 33 IoT
- Arduino Nano 33 IoT with headers
- Arduino Nano Every
- Arduino Nano Every - Pack
- Arduino Nano Every with headers
- Arduino Nano RP2040 Connect
- Arduino Nano RP2040 Connect with headers
- Arduino Uno Mini Limited Edition
- Arduino Uno Rev3

- Arduino Uno Rev3 SMD
- Arduino Uno WiFi Rev2
- Arduino Yún Rev 2
- Arduino Zero
- Nicla Sense ME
- Nicla Vision
- Portenta Breakout
- Portenta H7
- Portenta H7 Lite
- Portenta H7 Lite Connected
- Portenta Machine Control
- Portenta Max Carrier

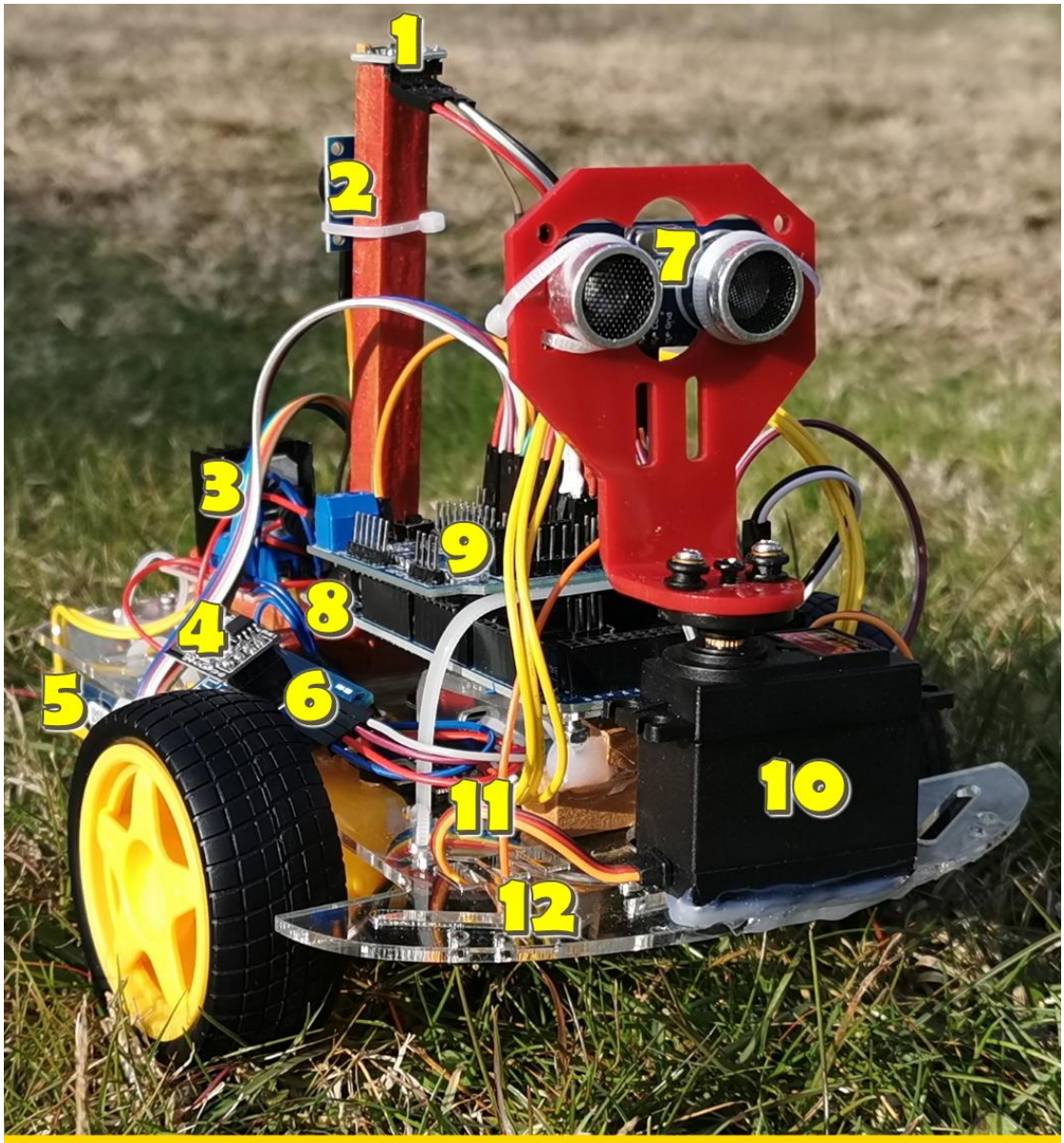
Arduino Mega 2560 Rev3

Εικόνα 3: Arduino Mega 2560 Rev3 (πηγή: Arduino Official Store)

Από τις πλακέτες Arduino που ανέφερα, εγώ χρησιμοποίησα την **Arduino Mega 2560 Rev3** στην κατασκευή μου.

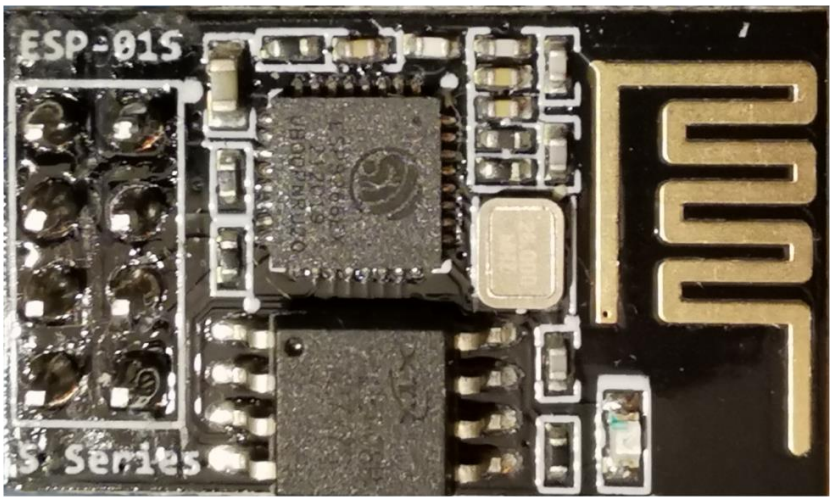
Τεχνικά χαρακτηριστικά	
Ακίδα ενσωματωμένου LED πλακέτας (LED_BUILTIN)	13
Βάρος	0.037kg
Διαστάσεις	101.52 x 53.3mm
Ένταση ρεύματος ακίδας 3.3V	50mA
Ένταση ρεύματος ανά ακίδα I/O	20mA
Μνήμη EEPROM	4KB
Μνήμη Flash	256KB
Μνήμη Flash EEPROM	32KB
Ολοκληρωμένο κύκλωμα (IC)	ATmega2560
Πλήθος αναλογικών εισόδων/εξόδων (Analog I/O ports)	16
Πλήθος ψηφιακών εισόδων/εξόδων (Digital I/O ports)	54 (15 με έξοδο PWM)
Ποσότητα μνήμης από τη μνήμη Flash για bootloader	8KB
Στατική μνήμη (SRAM)	8KB
Τάση εισόδου (προτείνεται)	DC 7 – 12V
Τάση εισόδου (όριο)	DC 6 – 20V
Τάση λειτουργίας	DC 5V
Ταχύτητα ρολογιού (Clock speed)	16MHz

ΚΕΦΑΛΑΙΟ 3: ΑΙΣΘΗΤΗΡΕΣ & ΠΕΡΙΦΕΡΕΙΑΚΕΣ ΣΥΣΚΕΥΕΣ



1 Μαγνητόμετρο GY-271
2 Βομβητής KY-006
3 Κύκλωμα οδήγησης κινητήρων L298N
4 Αισθητήρας WiFi ESP-01S
5 Μπαταρία λιθίου Gens Ace
6 Προσαρμογέας ESP-01
7 Αποστασιόμετρο HC-SR04
8 Arduino Mega 2560 Rev3
9 Πλακέτα επέκτασης Arduino
10 Σερβο-κινητήρας MG996R
11 Αισθητήρας ταχύτητας κινητήρα
12 Δίτροχο ρομπότ/αυτοκίνητο για Arduino

Αισθητήρας WiFi (WiFi module) ESP-01S



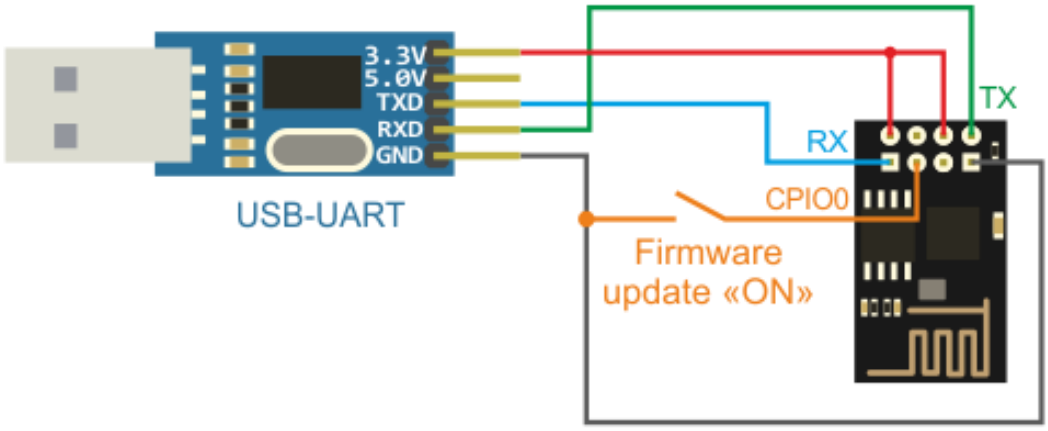
Το **ESP-01S** είναι μία μονάδα (module) WiFi με ενσωματωμένο IC ESP8266 και αποτελεί ένα αυτόνομο SOC με δυνατότητες TCP/IP. Χάρη σε αυτές μπορεί να δώσει πρόσβαση σε οποιονδήποτε μικροελεγκτή σε ένα δίκτυο WiFi. Το ESP8266 μπορεί είτε να φιλοξενήσει μια εφαρμογή είτε να δικτυωθεί μέσω WiFi. Το **ESP-01S** κυκλοφορεί προ-προγραμματισμένο με firmware που λειτουργεί με χρήση εντολών AT. Μπορούμε να το συνδέσουμε με μία πλακέτα Arduino και να αποκτήσει η τελευταία τόσες δυνατότητες σε WiFi όσες μπορεί να αποκτήσει αν τη συνδέσουμε με ένα WiFi Shield.

Τεχνικά χαρακτηριστικά	
Αρχιτεκτονική επεξεργαστή	RISC
Διαστάσεις	11.5 x 11.5mm
Ένταση ρεύματος	0.5μA – 215mA
Θερμοκρασία λειτουργίας	-40° – 125°C
Μνήμη RAM διαθέσιμη για δεδομένα	96KB
Μνήμη RAM διαθέσιμη για εντολές	64KB
Ολοκληρωμένο κύκλωμα (IC)	ESP8266
Συχνότητα λειτουργίας	80MHz
Τάση λειτουργίας	DC 3.3V
Τύπος WiFi	2.4 GHz, 802.11 b/g/n

Τα γνωστά firmwares που υπάρχουν για το module **ESP-01S** είναι τα ακόλουθα:

- Το firmware AT όπως το σύνολο εντολών Hayes στα παλιά μόντεμ
- Το firmware Espruino το οποίο περιλαμβάνει διερμηνέα JavaScript
- Το firmware MicroPython το οποίο περιλαμβάνει διερμηνέα Python
- Το firmware NodeMCU το οποίο περιλαμβάνει διερμηνέα LUA

Προκειμένου να χρησιμοποιήσουμε το **ESP-01S** αρχικά εγκαθιστούμε το τελευταίο κατάλληλο firmware για τις εντολές AT. Για να θέσουμε το module σε λειτουργία προγραμματισμού ενώνοντας την ακίδα GND με την ακίδα GPIO0.



Εικόνα 4: Διάγραμμα προγραμματισμού ESP-01S με USB-UART (πηγή: RemoteXY)

Εικόνα 5: Προγραμματισμός ESP-01S με προσαρμογέα ESP-01 και ένωση των ακίδων GND με GPIO0

Η Espressif σταμάτησε την ενημέρωση της σειράς ESP8266 όσον αφορά τις μονάδες (modules) μεγέθους 1 MB όπως φαίνεται από την ανάρτησή τους (https://docs.espressif.com/projects/esp-at/en/release-v2.2.0.0_esp8266/AT_Binary_Lists/ESP8266_AT_binaries.html).

Κάνουμε λήψη το firmware από την ιστοσελίδα της Espressif επισκεπτόμενοι τον σύνδεσμο https://www.espressif.com/en/products/socs/esp8266ex/resources.

Από το τμήμα **TOOLS** κάνουμε λήψη την τελευταία έκδοση του **Flash Download Tools** που εκτελείται σε **Windows**. Πρόκειται για την v3.9.3 [2022.08.26].

Από το τμήμα **SDKs & Demos** κάνουμε λήψη την τελευταία έκδοση του **ESP8266 NONOS SDK**. Αυτό μας οδηγεί στην ιστοσελίδα https://github.com/espressif/ESP8266_NONOS_SDK/releases. Πρόκειται για την **v3.0.5** [2021.10.18]. Τελευταία έκδοση firmware της σειράς ESP8266 μεγέθους 1 MB είναι η **v1.7.5**. Η **Espressif** όπως μας πληροφορεί η ίδια (https://docs.espressif.com/projects/esp-at/en/release-v2.2.0.0_esp8266/AT_Binary_Lists/ESP8266_AT_binaries.html) σταμάτησε την ενημέρωση της σειράς **ESP8266** μεγέθους 1 MB.

Για να φορτώσουμε το τελευταίο firmware στο ESP-01S ακολουθούμε τις οδηγίες από τον οδηγό που κάνουμε λήψη από την ιστοσελίδα https://www.espressif.com/sites/default/files/documentation/2a-esp8266-sdk_getting_started_guide_en.pdf. Στη σελίδα 21 του οδηγού εντοπίζουμε το πίνακα «Table 4-4. Download Addresses for OTA Firmware (unit: KB)» που μας πληροφορεί ποια στοιχεία από πακέτο χρειαζόμαστε και σε ποιες διευθύνσεις. Προφανώς εμείς κάνουμε χρήση των αρχείων και των διευθύνσεων που υπάρχουν στην 2η στήλη των 1024 KB (= 1 MB).

Table 4-4 lists the download addresses for the OTA firmware.

Table 4-4. Download Addresses for OTA Firmware (unit: KB)

Binaries	Download addresses in flash with different capacities							
	512	1024	2048		4096		8192	16384
			512+512	1024+1024	512+512	1024+1024	1024+1024	1024+1024
blank.bin	0x7B000	0xFB000	0x1FB000		0x3FB000		0x7FB000	0xFFB00
esp_init_data _default.bin	0x7C000	0xFC000	0x1FC000		0x3FC000		0x7FC000	0xFFC000
blank.bin	0x7E000	0xFE000	0x1FE000		0x3FE000		0x7FE000	0xFFE000
boot.bin					0x00000			
user1.bin					0x01000			
user2.bin	0x41000	0x81000	0x81000	0x101000	0x81000	0x101000	0x101000	0x101000

Η τελευταία έκδοση του **ESP8266 NONOS SDK** περιέχει τα ακόλουθα αρχεία από τα οποία χρησιμοποιούμε μόνο τα έντονα χρωματισμένα:

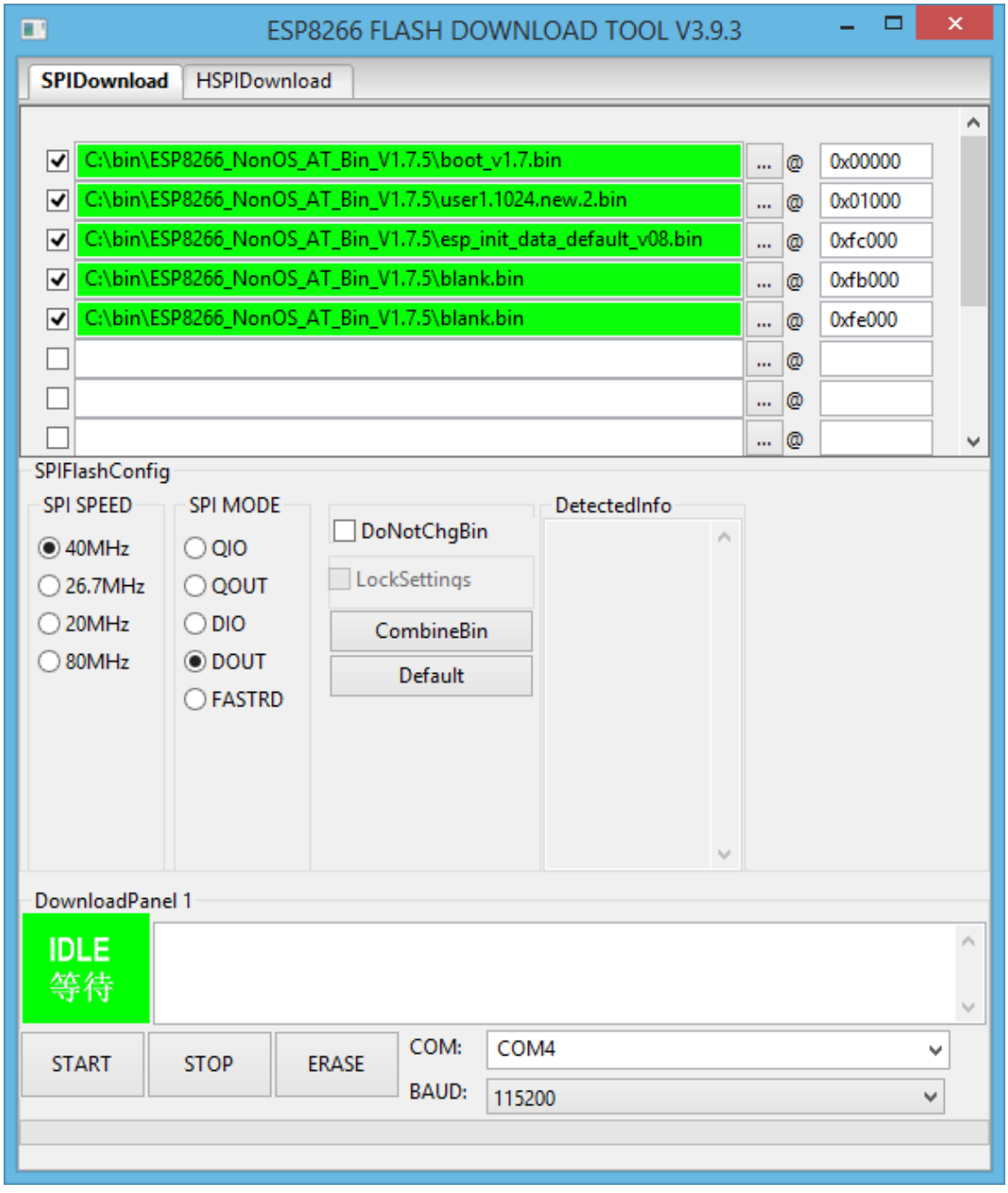
- ESP8266_NonOS_AT_Bin_V1.7.5\.DS_Store
- ESP8266_NonOS_AT_Bin_V1.7.5\bin
- ESP8266_NonOS_AT_Bin_V1.7.5\ESP8266 NonOS AT Release Note.pdf
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\.DS_Store
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at_sdio
- **ESP8266_NonOS_AT_Bin_V1.7.5\bin\blank.bin**
- **ESP8266_NonOS_AT_Bin_V1.7.5\bin\boot_v1.7.bin**
- **ESP8266_NonOS_AT_Bin_V1.7.5\bin\esp_init_data_default_v08.bin**
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\.DS_Store
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\1024+1024
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\512+512
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\README.md
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\1024+1024\user1.2048.new.5.bin
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\1024+1024\user2.2048.new.5.bin
- **ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\512+512\user1.1024.new.2.bin**
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at\512+512\user2.1024.new.2.bin
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at_sdio\.DS_Store
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at_sdio\1024+1024
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at_sdio\README.md
- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at_sdio\1024+1024\user1.2048.new.5.bin

- ESP8266_NonOS_AT_Bin_V1.7.5\bin\at_sdio\1024+1024\user2.2048.new.5.bin

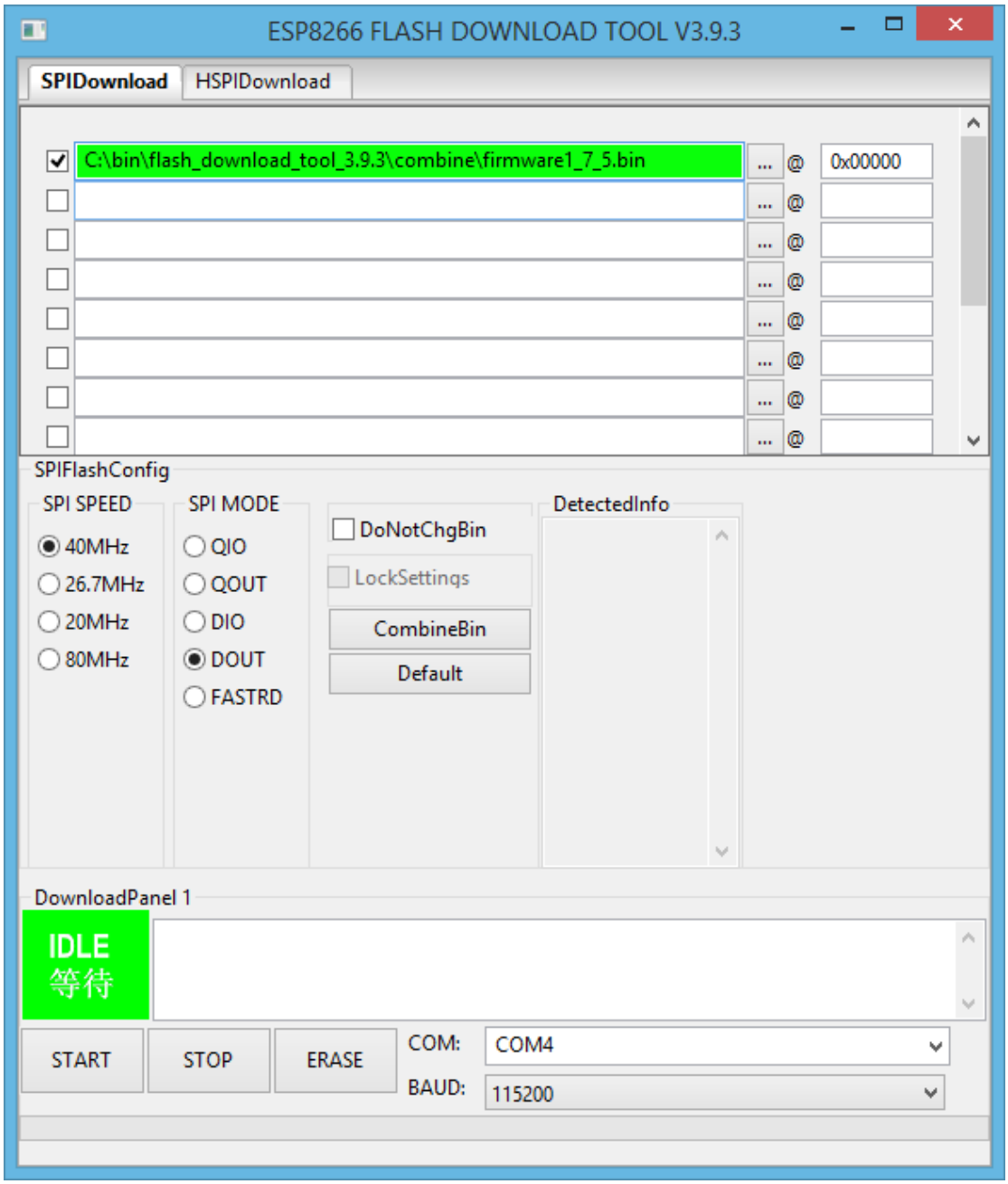
Συνεπώς, στέλνουμε τα παραπάνω αρχεία στο module με το **Flash Download Tool** σε συγκεκριμένες διευθύνσεις σύμφωνα με τον παρακάτω πίνακα:

ΑΡΧΕΙΟ	ΔΙΕΥΘΥΝΣΗ
boot_v1.7.bin	0x00000
user1.1024.new.2.bin	0x01000
esp_init_data_default_v08.bin	0xFC000
blank.bin	0xFB000
blank.bin	0xFE000

Σχετικά με τις υπόλοιπες ρυθμίσεις, επιλέγουμε τα **40MHz** από τις ταχύτητες SPI, επιλέγουμε το **DOUT** από τους τύπους SPI και το πεδίο **DoNotChgBin** δεν το επιλέγουμε.

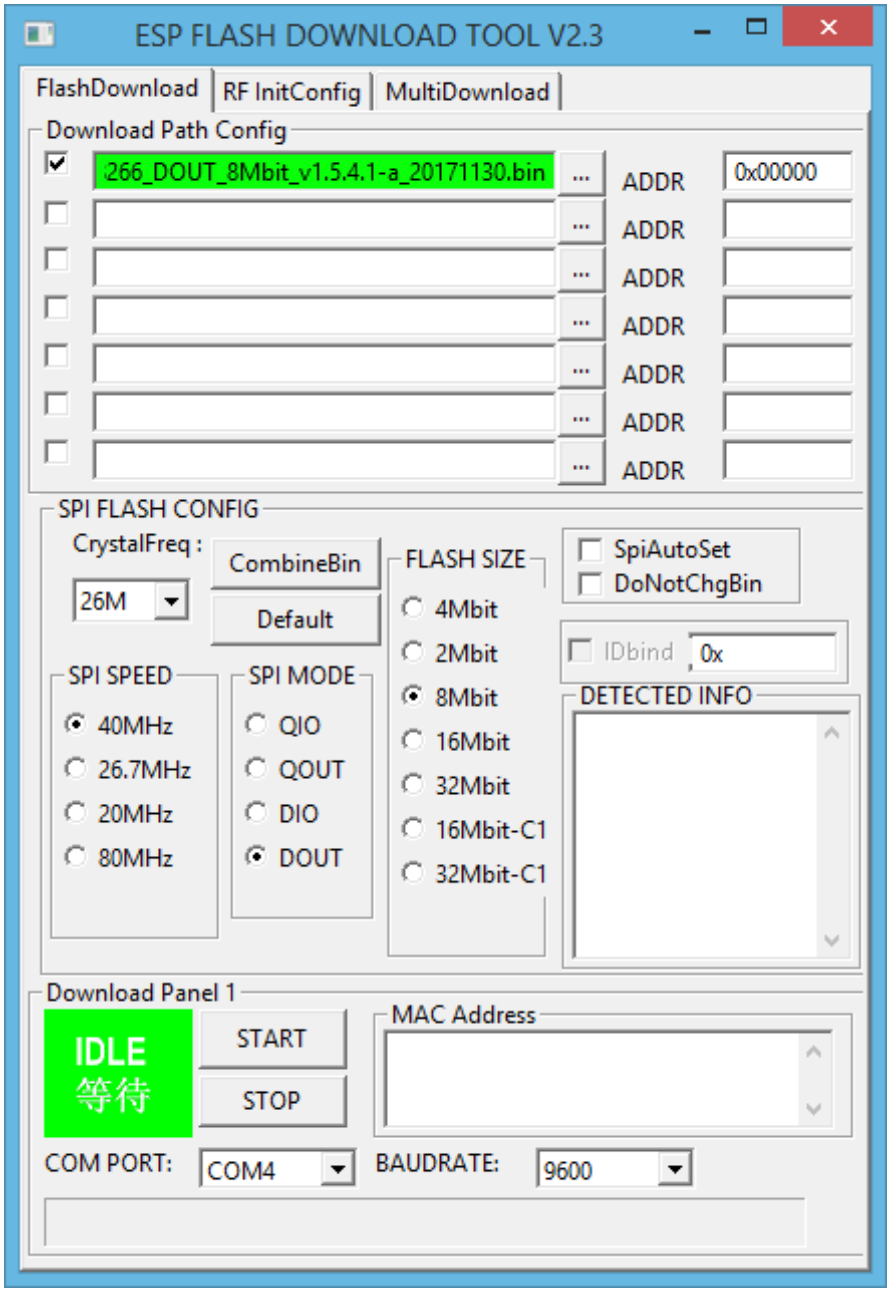


ΣΗΜΑΝΤΙΚΟ! Αν πατήσουμε το κουμπί **CombineBin** τότε δημιουργείται στον φάκελο **combine** του **Flash Download Tool** ένα νέο αρχείο τύπου **.bin** που περιέχει όλα τα παραπάνω αρχεία με τις δηλωμένες διευθύνσεις τους. Αυτό το αρχείο μπορούμε να το χρησιμοποιήσουμε για την αναβάθμιση του firmware του module απευθείας στην διεύθυνση **0x00000** αντί όλων των παραπάνω από τον προηγούμενο πίνακα. Οι υπόλοιπες ρυθμίσεις παραμένουν οι ίδιες.

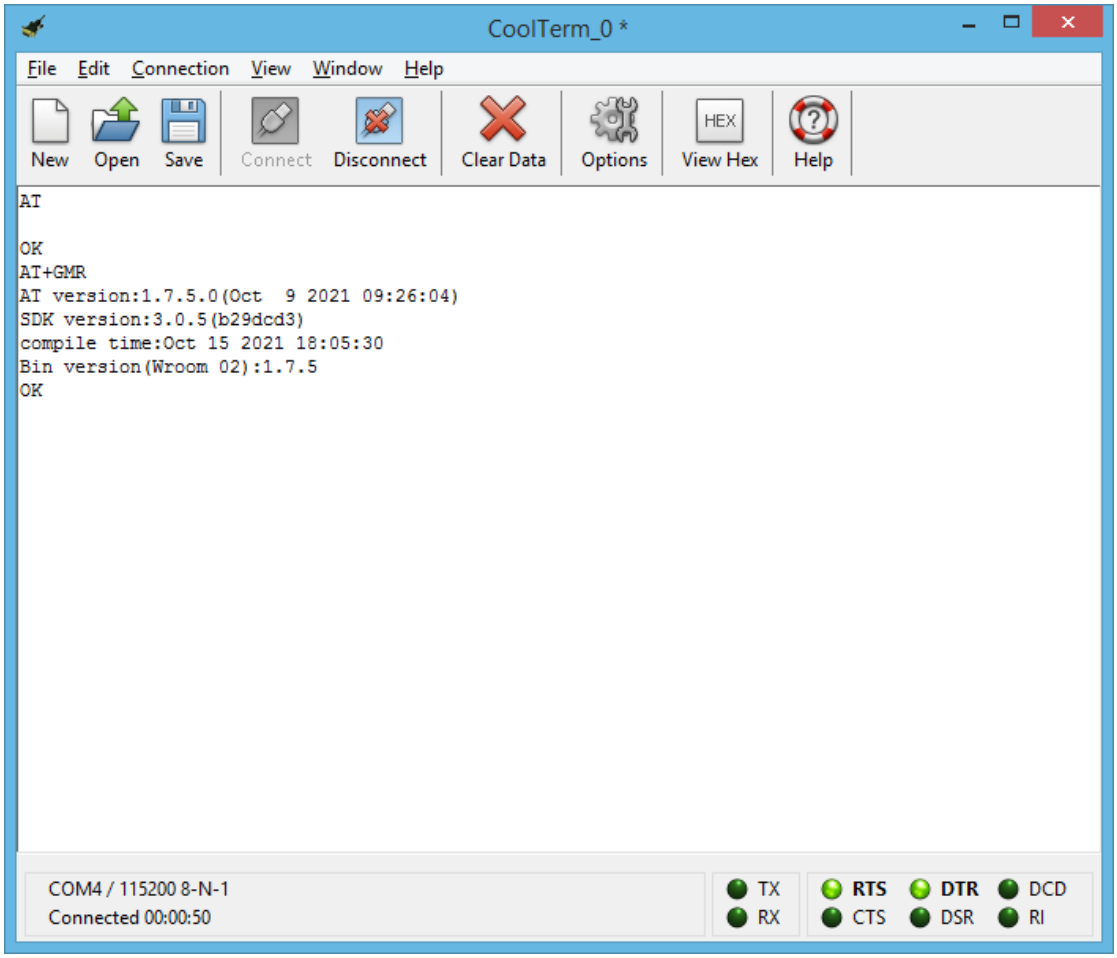


ΠΡΟΣΟΧΗ! Η τελευταία έκδοση του **Flash Download Tool** μπορεί να προγραμματίσει το **ESP-01S** μόνο σε ταχύτητες (baud rate) από **115200** και πάνω. Σε περίπτωση που έχουμε αλλάξει την ταχύτητα του module σε κάποια μικρότερη και έχουμε αντικαταστήσει το AT firmware με κάποιο sketch ή άλλο (μη AT) firmware τότε θα πρέπει να ανατρέξουμε σε πολύ παλιότερη έκδοση **Flash Download Tool** όπως π.χ. την **v2.4** [2015.09.24]. Τη συγκεκριμένη μπορούμε να τη λάβουμε από την ιστοσελίδα https://bbs.espressif.com/viewtopic.php?f=57&t=433. Αυτή υποστηρίζει ταχύτητες από **9600** και πάνω. Πλέον μπορούμε να γράψουμε ένα παλιότερο AT

firmware, το **v1.5.4.1-a** [2017.11.30] που θα βρούμε στην ιστοσελίδα https://docs.ai-thinker.com/en/esp8266/sdk. Έπειτα, αφού αλλάξουμε την ταχύτητα στην **115200,** μπορούμε να προχωρήσουμε στο αναβαθμισμένο firmware **v1.7.5**.

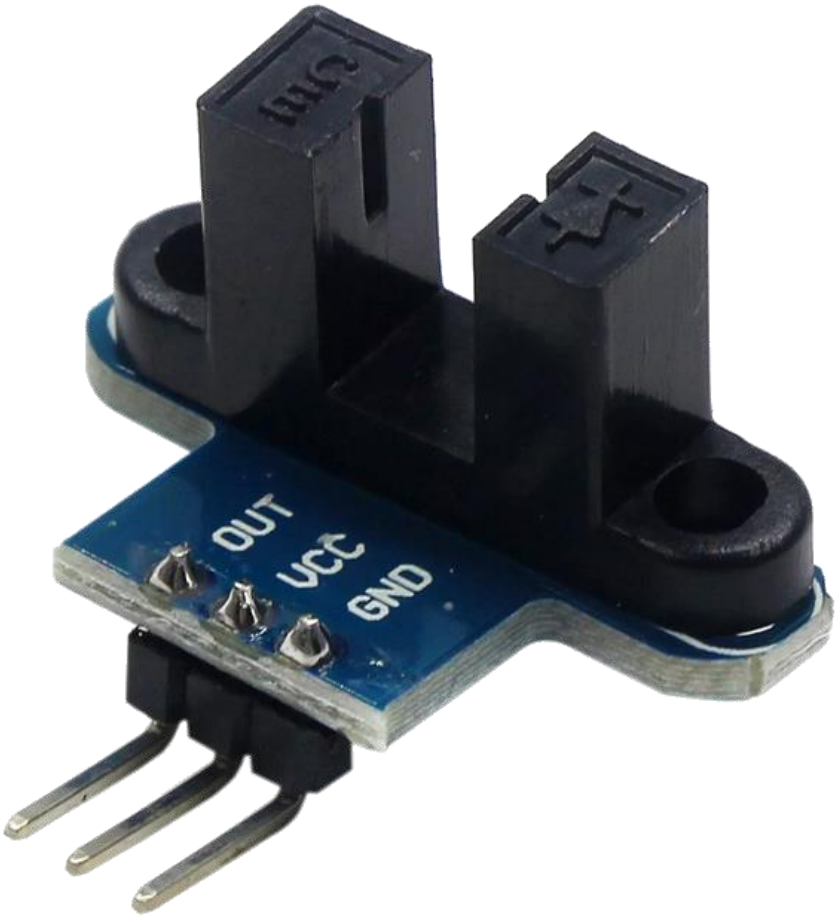


Με χρήση της εφαρμογής υπολογιστή **CoolTerm** μπορώ να ελέγξω ότι προγραμμάτισα σωστά το **ESP-01S**. Επαληθεύω ότι το **ESP-01S** δέχεται εντολές AT παίρνοντας ως απάντηση τη λέξη «`OK`» αφού εκτελέσω την εντολή «**`AT`**». Επαληθεύω ότι το module διαθέτει την επιθυμητή έκδοση firmware παίρνοντας ως απάντηση το κείμενο «`AT version … 1.7.5`» ακολουθούμενο από τη λέξη «`OK`» αφού εκτελέσω την εντολή «**`AT+GMR`**».



Εικόνα 6: Παράδειγμα χρήσης των εντολών «AT» και «AT+GMR» στην εφαρμογήυπολογιστή CoolTerm

Αισθητήρας ταχύτητας κινητήρα (IR slotted optical speed measuring sensor)



Εικόνα 7: Αισθητήρας ταχύτητας κινητήρα (πηγή: AliExpress)

Ο αισθητήρας ταχύτητας του κινητήρα περιέχει το IC LM393 και μετράει τους παλμούς, δηλαδή τις στροφές του κινητήρα.

Τεχνικά χαρακτηριστικά	
Βάρος	0.01kg
Διαστάσεις κωδικοποιητή δίσκου	32 x 14mm
Μορφή εξόδου	Έξοδος ψηφιακής μεταγωγής (0 και 1)
Ολοκληρωμένο κύκλωμα (IC)	LM393
Πλάτος υποδοχής	5mm
Τάση λειτουργίας	DC 3.3 – 5V

Πρόκειται για έναν απλό αισθητήρα υπερύθρων ο οποίος υπολογίζει τις περιστροφές ανά δευτερόλεπτο του κινητήρα κάθε φορά που μπλοκάρεται η δέσμη υπερύθρων αυξάνοντας έναν μετρητή. Βρίσκει εφαρμογή στους κινητήρες συνεχούς ρεύματος επειδή οι συγκεκριμένοι ανάλογα με την ποσότητα ρεύματος κινούνται πιο γρήγορα ή πιο αργά αλλά δεν είναι ακριβείς όπως είναι οι βηματικοί κινητήρες. Οι τελευταίοι είναι ακριβείς επειδή ο αριθμός των βημάτων τους είναι πάντα γνωστός. Γνωρίζοντας την ταχύτητα του κινητήρα, μπορώ στον κώδικά μου να ρυθμίσω την ταχύτητα με βάση τον αριθμό περιστροφής που

κάνει και να υπολογίσω κατά προσέγγιση την απόσταση μετακίνησης του ρομπότ μου. Στον κώδικά μου χρησιμοποιώ διακοπή η οποία καλείται κάθε φορά που μπλοκάρεται η δέσμη υπερύθρων ώστε να αυξάνεται ένας μετρητής. Ο δικός μου κωδικοποιητής δίσκος αποτελείται από 20 τρύπες και αυτό σημαίνει ότι για να υπολογίσω τις περιστροφές του κινητήρα ανά δευτερόλεπτο αρκεί να διαιρέσω την τιμή του μετρητή με τον αριθμό 20. Επειδή διαθέτω δύο κινητήρες χρησιμοποιώ δύο διακοπές.

Εικόνα 8: Κωδικοποιητής δίσκος 20 οπών (πηγή: Amazon.co.uk)

Ο αισθητήρας ταχύτητας διαθέτει τις ακόλουθες τρεις (3) ακίδες:

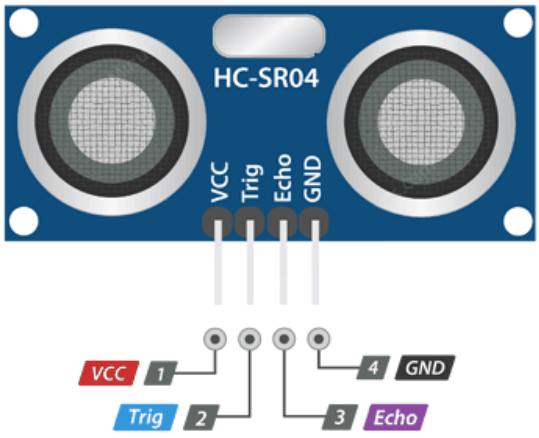
- GND: Ακίδα γείωσης του αισθητήρα που την συνδέουμε με τη γείωση του Arduino.
- OUT: Ακίδα που χρησιμοποιείται για την ενημέρωση του Arduino ότι άλλαξε η στάθμη του αισθητήρα από χαμηλή σε υψηλή που συμβαίνει όταν μπλοκάρεται η σχισμή από την οποία διέρχεται η δέσμη υπερύθρων.
- VCC: Ακίδα τροφοδοσίας του αισθητήρα που την συνδέουμε με την ακίδα των 5V του Arduino.

Αποστασιόμετρο (Ultrasonic distance sensor) HC-SR04

Εικόνα 9: Αποστασιόμετρο HC-SR04 (πηγή: https://www.why.gr/)

Το αποστασιόμετρο HC-SR04 είναι ένας αισθητήρας υπερήχων και χρησιμοποιείται στον υπολογισμό απόστασης. Η απόσταση που μπορεί να υπολογίσει κυμαίνεται από 2cm έως 400cm με ακρίβεια 1cm και απόκλιση 3mm.

Τεχνικά χαρακτηριστικά	
Ακρίβεια μέτρησης	3mm
Γωνία μέτρησης	15°
Διαστάσεις	45 x 20 x 15mm
Ελάχιστο εύρος	2cm
Ένταση ρεύματος	15mA
Μέγιστο εύρος	4m
Σήμα εισόδου ενεργοποίησης	Παλμός 10μs TTL
Συχνότητα λειτουργίας	40kHz
Τάση λειτουργίας	DC 5V



Εικόνα 10: Σχεδιάγραμμα ακίδων αποστασιόμετρου (πηγή: Last Minute Engineers)

Το αποστασιόμετρο **HC-SR04** διαθέτει τις ακόλουθες τέσσερις (4) ακίδες:

- Echo: Αυτή η ακίδα πηγαίνει στην θέση HIGH όταν παραχθεί ένα σετ 8 παλμών και επιστρέφει στην θέση LOW όταν οι παλμοί γυρίσουν πίσω. Η απόσταση προκύπτει από την διάρκεια που η ακίδα βρισκόταν στην θέση HIGH.
- GND: Ακίδα γείωσης του αποστασιόμετρου που την συνδέουμε με τη γείωση του Arduino.
- Trig: Ακίδα που χρησιμοποιείται για την ενεργοποίηση των υπερηχητικών παλμών. Παράγουμε ένα υπερηχητικό παλμό θέτοντας την ακίδα στην θέση HIGH για 10μs.
- VCC: Ακίδα τροφοδοσίας του αποστασιόμετρου που την συνδέουμε με την ακίδα των 5V του Arduino.

Λειτουργία αποστασιόμετρου

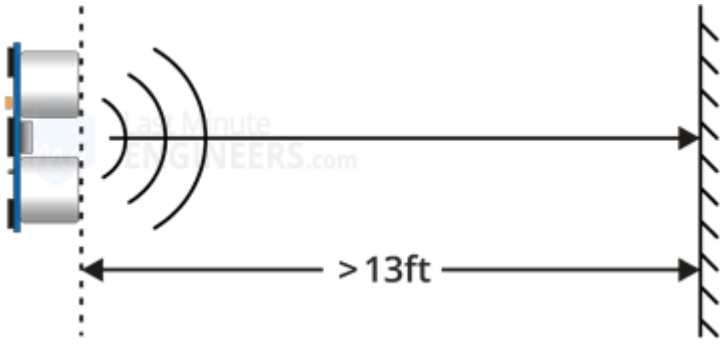
Όλα ξεκινούν όταν η ακίδα **Trig** έχει ρυθμιστεί στην θέση HIGH για 10μs. Τότε ο αισθητήρας εκπέμπει μια δέσμη υπερήχων 8 παλμών στα 40kHz. Αυτό το σετ των 8 παλμών είναι σχεδιασμένο με τέτοιον τρόπο ώστε ο δέκτης να μπορεί να διακρίνει τους μεταδιδόμενους παλμούς από τον υπερηχητικό θόρυβο του περιβάλλοντος. Αυτοί οι 8 παλμοί υπερήχων ταξιδεύουν μέσω του αέρα μακριά από τον πομπό. Παράλληλα η ακίδα **Echo** βρίσκεται στην θέση HIGH για να εκκινήσει το σήμα της ηχούς.

Εάν αυτοί οι παλμοί δεν αντανακλώνται πίσω, το σήμα ηχούς λήγει και μειώνεται μετά από 38ms. Έτσι, ένας παλμός 38ms υποδεικνύει ότι δεν υπάρχει εμπόδιο εντός της εμβέλειας του αισθητήρα. Εάν αυτοί οι παλμοί αντανακλώνται πίσω, η ακίδα ηχούς χαμηλώνει μόλις ληφθεί το σήμα. Αυτό δημιουργεί έναν παλμό στην ακίδα **Echo** του οποίου το πλάτος κυμαίνεται από 150μs έως 25ms, ανάλογα με το χρόνο που χρειάστηκε για τη λήψη του σήματος.

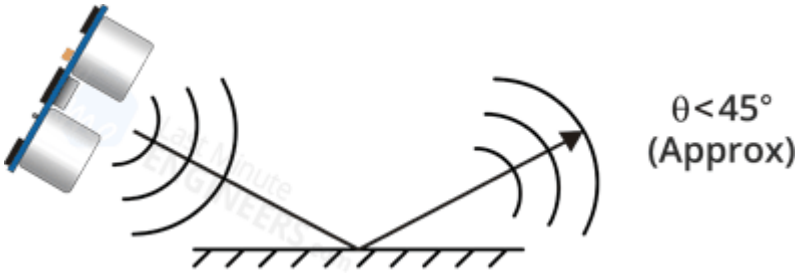
Ο γενικός τύπος της απόστασης δίνεται από την σχέση: Απόσταση = Ταχύτητα x Χρόνος

- Η ταχύτητα του ήχου είναι η 340 m/s ή 0.034 cm/μs
- Ο χρόνος που απαιτήθηκε αφορά την αποστολή και την επιστροφή του σήματος.
- Συνοψίζοντας τα παραπάνω ο τύπος της απόστασης μετασχηματίζεται στην νέα σχέση:
  [Απόσταση σε cm] = (0.034 cm/μs x [Χρόνος σε ms]) / 2

Περιορισμοί στην χρήση του αποστασιόμετρου

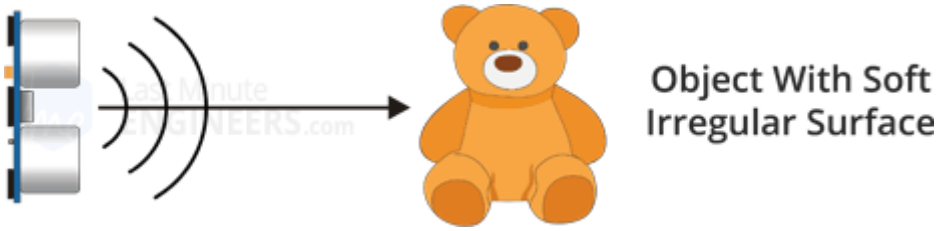


Εικόνα 11: Η απόσταση από το αντικείμενο ξεπερνάει τα 4m (πηγή: Last Minute Engineers)



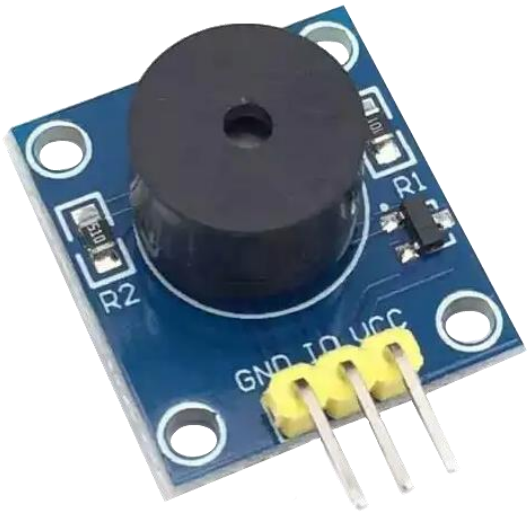
Εικόνα 12: Ο ήχος δεν επιστρέφεται στο αποστασιόμετρο (πηγή: Last Minute Engineers)

Εικόνα 13: Το αντικείμενο είναι πολύ μικρό για να μπορέσει να αντανακλάσει τον ήχο του αποστασιόμετρου (πηγή: Last Minute Engineers)



Εικόνα 14: Το αντικείμενο απορροφά τον ήχο του αποστασιόμετρου (πηγή: Last Minute Engineers)

Βομβητής (Buzzer) KY-006



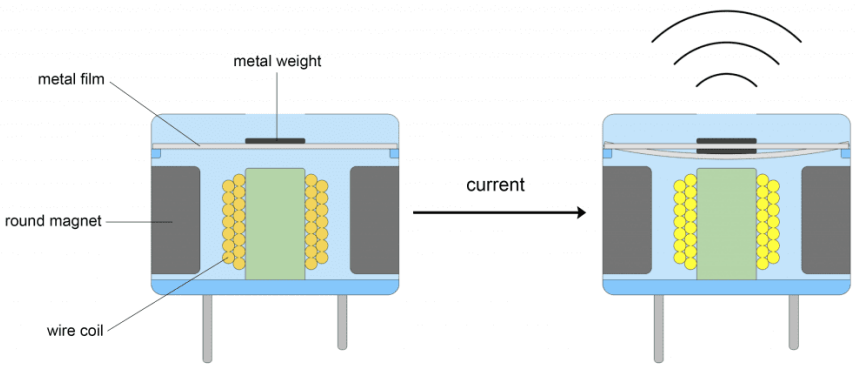
Εικόνα 15: Βομβητής (πηγή: Zazootek)

Ο βομβητής **KY-006** χρησιμοποιείται για να παράγει διαφορετικούς ήχους, ήχους τύπου beep ή ήχους ειδοποίησης. Στην κατασκευή μου χρησιμοποίησα παθητικό βομβητή (passive buzzer). Αυτός δεν διαθέτει πηγή ταλάντωσης. Για να μπορέσει να παράγει ήχο πρέπει να οδηγηθεί από τετραγωνικό κύμα 2 – 5kHz (PWM).

Τεχνικά χαρακτηριστικά	
Διαστάσεις	20 x 15mm
Είδος & συχνότητα κύματος οδήγησης	τετραγωνικό κύμα 2 – 5kHz
Τάση λειτουργίας κινητήρων	DC 3.3 – 5V
Τρανζίστορ	9012

Ο βομβητής διαθέτει τις ακόλουθες τρεις (3) ακίδες:

- GND: Ακίδα γείωσης του βομβητή που την συνδέουμε με τη γείωση του Arduino.
- IO: Ακίδα εισόδου/εξόδου του βομβητή που την συνδέουμε με ακίδα εισόδου/εξόδου του Arduino.
- VCC: Ακίδα τροφοδοσίας του βομβητή που την συνδέουμε με την ακίδα των 5V του Arduino.



Εικόνα 16: Σχεδιάγραμμα παραγωγής ήχου από βομβητή (πηγή: Circuit Basics)

Δίτροχο ρομπότ/αυτοκίνητο για Arduino (2WD Robot car chassis kit for Arduino)

Εικόνα 17: Δίτροχο ρομπότ/αυτοκίνητο για Arduino (πηγή: Amazon.com)

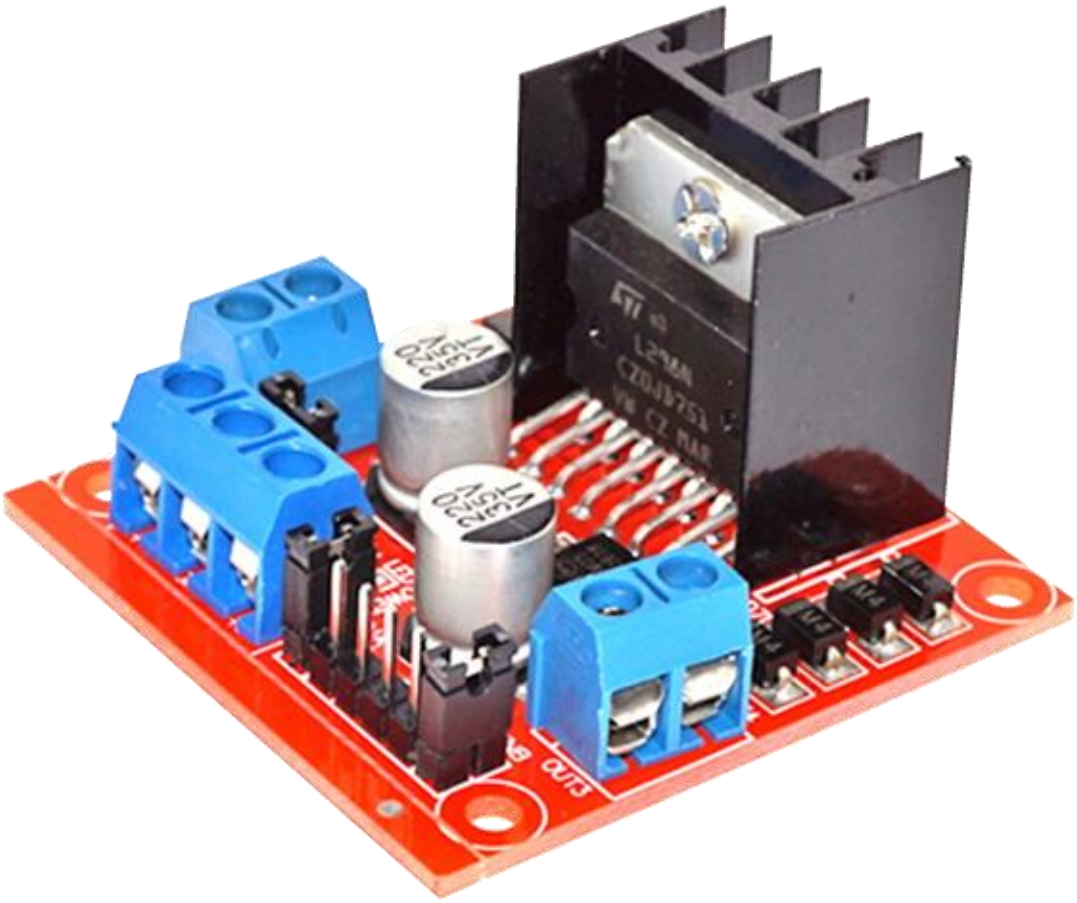
Αυτό το πακέτο του δίτροχου ρομπότ/αυτοκινήτου είναι πολύ εύκολο στη συναρμολόγησή του. Στην ουσία πρόκειται για μια πλατφόρμα αυτοκινήτου στην οποία προσθέτουμε μερικούς μικροελεγκτές (όπως το Arduino) και αισθητήρες, και στο τέλος το προγραμματίζουμε.

Τεχνικά χαρακτηριστικά	
Βάρος	0.289kg
Διαστάσεις	221 x 155 x 39mm
Τάση λειτουργίας κινητήρων	DC 3 – 6V

Το πακέτο του δίτροχου ρομπότ/αυτοκινήτου περιέχει:

- 2 x DC κινητήρες μετάδοσης (1:48)
- 14 x Βίδες
- 1 x Διακόπτη κυκλώματος
- 1 x Κουμπί λειτουργίας
- 1 x Κουτί μπαταρίας
- 2 x Κωδικοποιητές δίσκους
- 10 x Παξιμάδια
- 1 x Σασί αυτοκινήτου
- 4 x Συνδετήρες
- 1 x Τροχό τύπου castor
- 2 x Τροχούς με λάστιχο

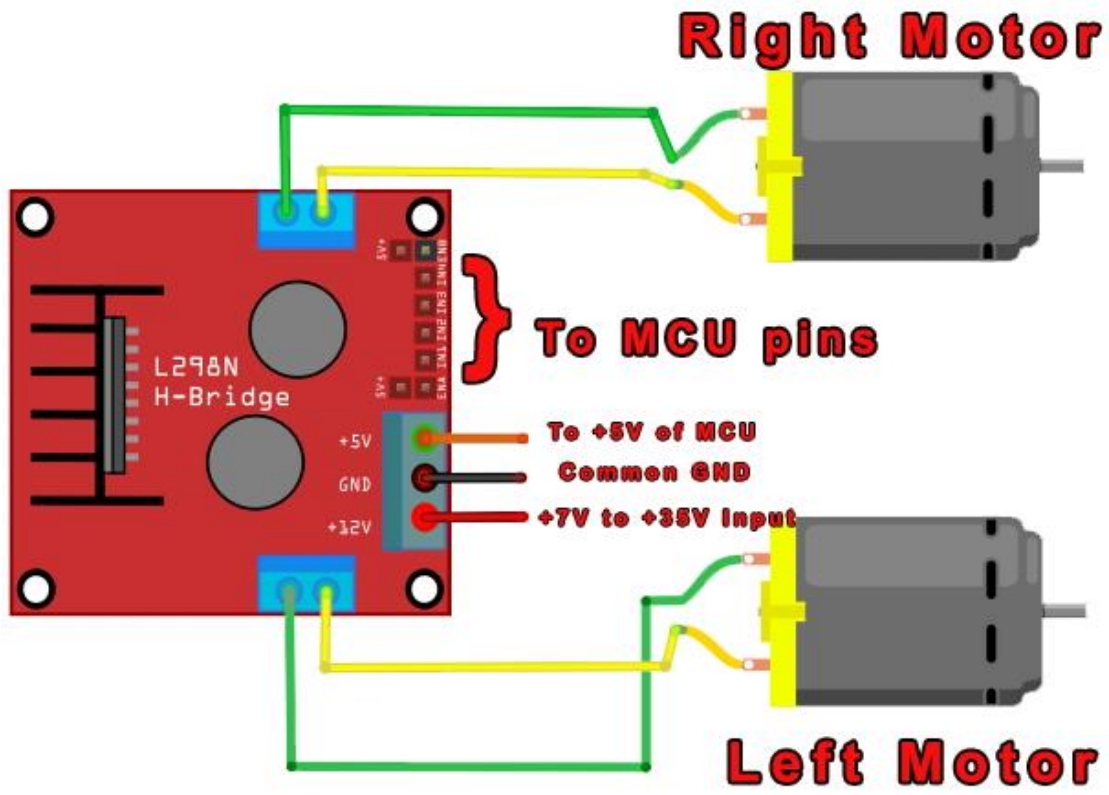
Κύκλωμα οδήγησης κινητήρων (Dual motor driver module) L298N



Εικόνα 18: Κύκλωμα οδήγησης κινητήρων L298N (πηγή: Ηλεκτρονικά Dme)

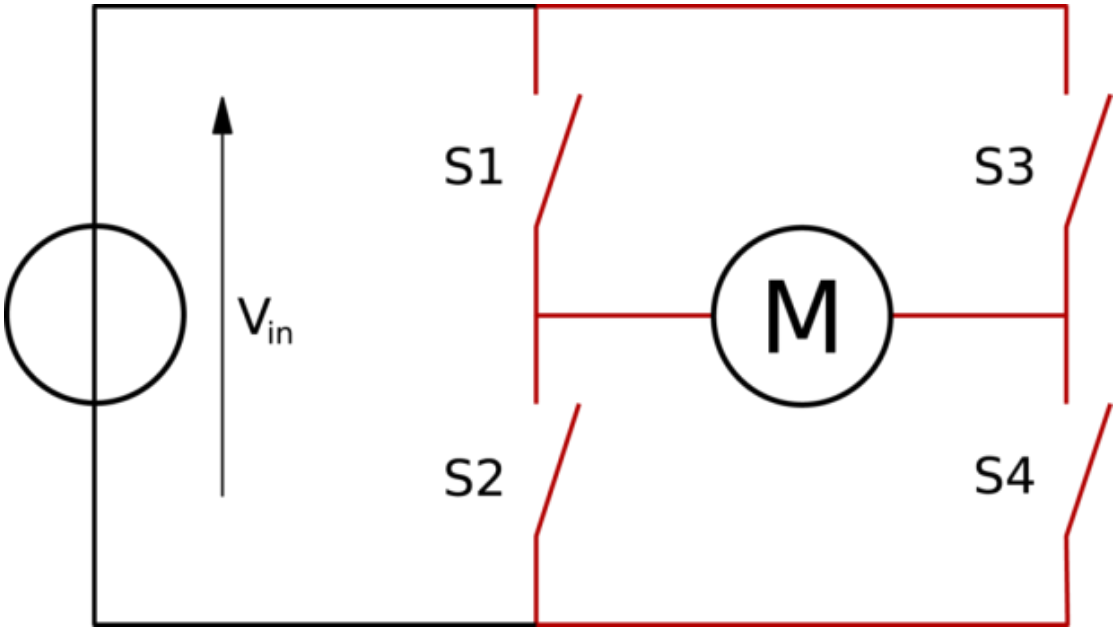
Το κύκλωμα οδήγησης κινητήρων L298N χρησιμοποιείται κυρίως στον έλεγχο της ταχύτητας και της κατεύθυνσης των κινητήρων. Ωστόσο, μπορεί να χρησιμοποιηθεί στον έλεγχο και άλλων ηλεκτρικών συσκευών όπως οι συστοιχίες LED. Όσον αφορά τους κινητήρες, το κύκλωμα L298N μπορεί να οδηγήσει ένα σερβο-κινητήρα 2 φάσεων, ένα σερβο-κινητήρα 4 φάσεων ή δύο κινητήρες συνεχούς ρεύματος. Το ρεύμα των 20mA που παρέχει το Arduino είναι αρκετό για να φωτίσει ένα λαμπτήρα LED αλλά δεν επαρκεί για να καλύψει τις ανάγκες των κινητήρων. Για αυτό χρησιμοποιούμε το κύκλωμα L298N.

Τεχνικά χαρακτηριστικά	
Βάρος (καθαρό / μεικτό)	0.026kg / 0.036kg
Διαστάσεις	55 x 60 x 30mm
Ένταση ρεύματος / κανάλι	2A
Ένταση ρεύματος λογικού τμήματος	36mA
Θερμοκρασία λειτουργίας	-25° – 130°C
Μέγιστη κατανάλωση ισχύος	25W
Ολοκληρωμένο κύκλωμα (IC)	L298N
Τάση εισόδου λογικού τμήματος	DC 4.5 – 5.5V
Τάση εξόδου κινητήρα	DC 5 – 35V
Τάση εξόδου κινητήρα (προτείνεται)	DC 7 – 12V



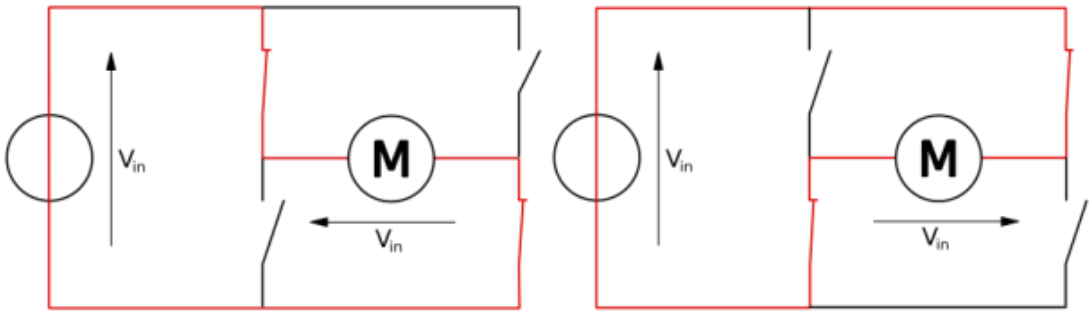
Εικόνα 19: Διάγραμμα σύνδεσης κυκλώματος L298N με δύο DC κινητήρες (πηγή: pantou.mysch.gr)

Το κύκλωμα οδήγησης κινητήρων **L298N** χρησιμοποιεί την διάταξη της «γέφυρας τύπου Η» (H-Bridge) για τον έλεγχο της κατεύθυνσης περιστροφής ενός DC κινητήρα.



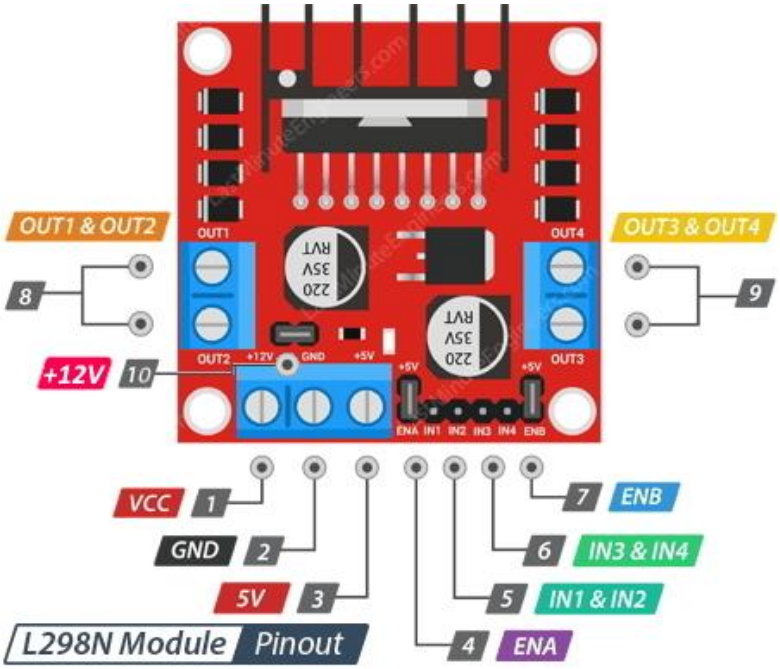
Εικόνα 20: Διάγραμμα διάταξης «γέφυρας τύπου Η» (πηγή: pantou.mysch.gr)

Οι διακόπτες στην διάταξη «γέφυρας τύπου Η» ορίζουν την κατεύθυνση περιστροφής του κινητήρα. Με κλειστούς τους διακόπτες S1 και S4 ο κινητήρας περιστρέφεται προς μία κατεύθυνση ενώ με κλειστούς τους διακόπτες S2 και S3 ο κινητήρας περιστρέφεται προς άλλη κατεύθυνση.



Εικόνα 21: Αλλαγή φοράς ρεύματος με την διάταξη της «γέφυρας τύπου Η» (πηγή: pantou.mysch.gr)

Ένα άλλο προτέρημα της διάταξης της «γέφυρας τύπου Η» είναι η δυνατότητα παροχής εξωτερικής πηγής ρεύματος στο κύκλωμα των κινητήρων επειδή όπως ήδη αναφέρθηκε το Arduino δεν επαρκεί ενεργειακά για την κάλυψή του.



Εικόνα 22: Διάγραμμα ακίδων κυκλώματος L298N (πηγή: Last Minute Engineers)

Το κύκλωμα **L298N** διαθέτει τέσσερις (4) εισόδους αντίστοιχες των τεσσάρων (4) διακοπτών της διάταξης της «γέφυρας τύπου Η» (OUT1, OUT2, OUT3, OUT4). Ο πίνακας ελέγχου έχει τερματικά των 12V και 5V (VCC, 5V). Το τερματικό των 12V μπορεί να δεχτεί τάσεις από 7V έως 12V όταν ο βραχυκυκλωτήρας των 12V (+12V) είναι στην θέση του ενώ από 12V έως 35V χωρίς τον βραχυκυκλωτήρα. Στην πρώτη περίπτωση ενεργοποιείται ο ρυθμιστής τάσης του κυκλώματος **L298N** και μπορεί να δεχθεί 5V από το Arduino. Οι ακροδέκτες των κινητήρων συνδέονται με τους ακροδέκτες IN1, IN2, IN3 και IN4 του κυκλώματος **L298N**. Για να χρησιμοποιήσουμε σερβο-κινητήρες διατηρούμε τους βραχυκυκλωτήρες στις ακίδες ENA και ENB. Διαφορετικά, για να χρησιμοποιήσουμε κινητήρες συνεχούς ρεύματος, αφαιρούμε τους βραχυκυκλωτήρες και αυτές τις ακίδες τις συνδέουμε με ακίδες PWM του Arduino.

IN1	IN2	IN3	IN4	Κατεύθυνση
0	0	0	0	Να σταματήσει
1	0	1	0	Προς τα εμπρός
0	1	0	1	ΑΝΤΙΣΤΡΟΦΗ
1	0	0	1	Αριστερά
0	1	1	0	Δεξιά

Εικόνα 23: Ακίδες κυκλώματος οδήγησης κινητήρων L298N (πηγή: pantou.mysch.gr)

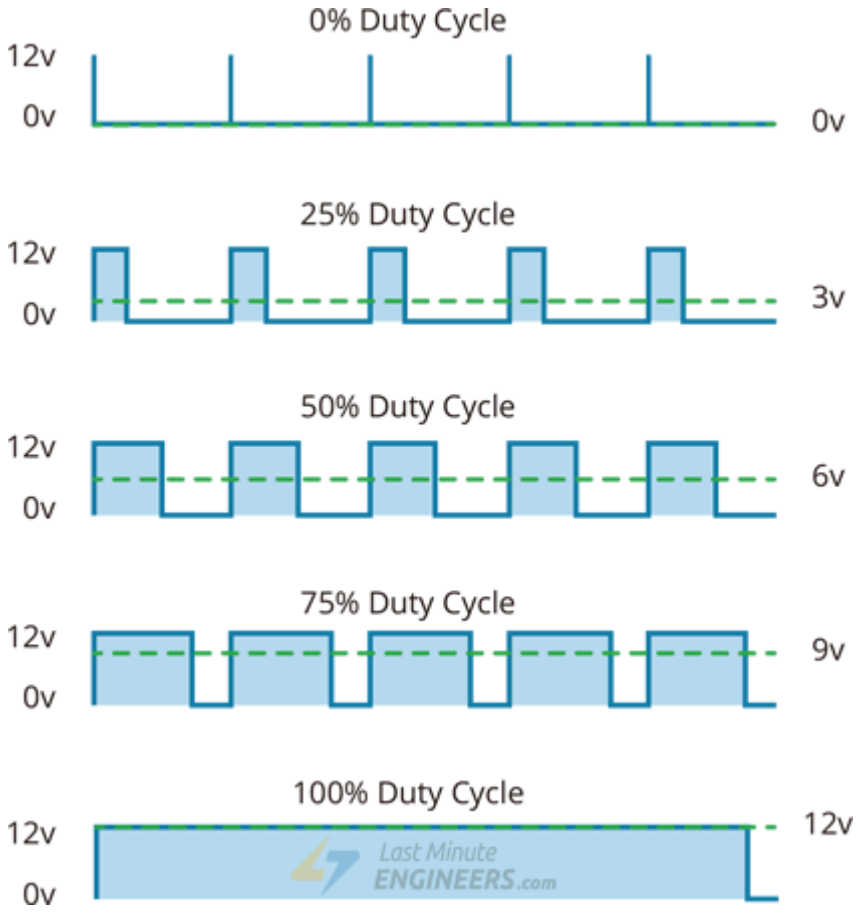
Για να περιστρέψουμε τον αριστερό κινητήρα προς μία κατεύθυνση αρκεί να εφαρμόσουμε έναν υψηλό παλμό στην ακίδα IN1 και έναν χαμηλό παλμό στην ακίδα IN2. Για να αντιστρέψουμε την κατεύθυνση, αρκεί να εφαρμόσουμε αντίστροφα τους παλμούς τις ακίδες IN1 και IN2. Ομοίως, περιστρέφουμε και τον δεξιό κινητήρα με τις ακίδες IN3 και IN4.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev2, Nano Every	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3, A4	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3, A4	732 Hz
Zero *	3 - 13, A0, A1	732 Hz
Nano 33 IoT *	2, 3, 5, 6, 9 - 12, A2, A3, A5	732 Hz
Nano 33 BLE/BLE Sense	1 - 13, A0 - A7	500 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

Εικόνα 24: Ακίδες PWM στο Arduino Mega (πηγή: Arduino)

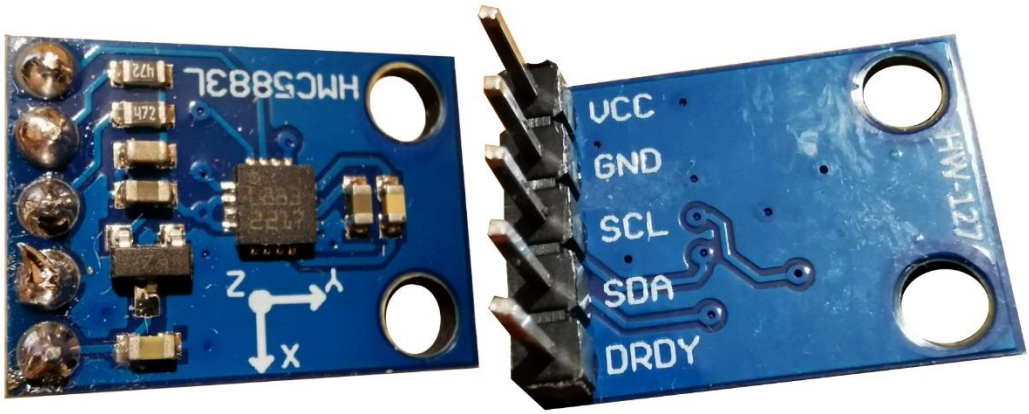
Στην παραπάνω εικόνα σημειώνω με μπλε χρώμα τις ακίδες PWM που υπάρχουν στο Arduino Mega. Η συχνότητα αυτών γενικά διαμορφώνεται στα 490Hz με εξαίρεση αυτών που σημειώνω με κόκκινο χρώμα (ακίδες 4 και 13) των οποίων η συχνότητα διαμορφώνεται στα 980Hz. Στην κατασκευή μου χρησιμοποιώ τις ακίδες των 980Hz για να ελέγξω την ταχύτητα του δεξιού και του αριστερού κινητήρα.

Η ταχύτητα του κινητήρα είναι ανάλογη με το πλάτος των παλμών. Οι παλμοί μεγάλου πλάτους περιστρέφουν τον κινητήρα γρήγορα και αντίστροφα. Ας σημειωθεί ότι ίδιοι κινητήρες που δέχονται παλμούς ίδιου πλάτους δεν περιστρέφονται ακριβώς με την ίδια ταχύτητα.



Εικόνα 25: Διαμόρφωση πλάτους παλμών (πηγή: Last Minute Engineers)

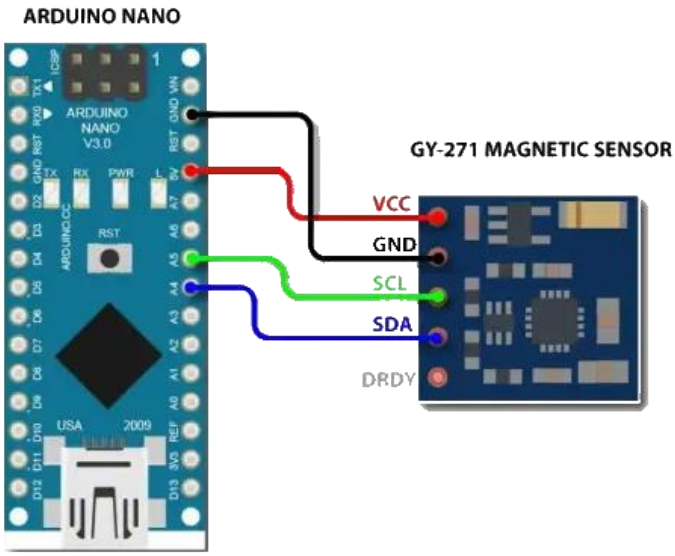
Μαγνητόμετρο (Magnetic field sensor) GY-271



Εικόνα 26: Μαγνητόμετρο GY-271

Το μαγνητόμετρο **GY-271** είναι ένας αισθητήρας μαγνητικού πεδίου και χρησιμοποιείται σε διάφορα gadgets. Επίσης, μπορεί να χρησιμοποιηθεί σε ρομπότ για τη λήψη πληροφοριών κατεύθυνσης, σε ρομπότ αυτό-εξισορρόπησης, αυτόνομα οχήματα κ.τ.λ. Τέτοιου τύπου αισθητήρες χρησιμοποιούνται οπουδήποτε χρειάζεται η πληροφορία της κατεύθυνσης. Το μαγνητόμετρο φέρει επάνω του IC το οποίο διαθέτει υψηλή ανάλυση βάθους 12-bit ADC και η ακρίβειά του κυμαίνεται από 0.3 έως 2 μοίρες. Ο ενσωματωμένος ρυθμιστής τάσης του μπορεί να δεχθεί τάση από 3.3V έως 6V DC.

Τεχνικά χαρακτηριστικά	
Ακρίβεια μέτρησης	0.3^o – 2^o
Βάρος (καθαρό / μεικτό)	~1g / ~2g
Διαστάσεις	14 x 13 x 4mm
Ελάχιστο εύρος μαγνητικού πεδίου	-1.3G
Μέγιστο εύρος μαγνητικού πεδίου	8 G
Τάση λειτουργίας	DC 3.3 – 6V



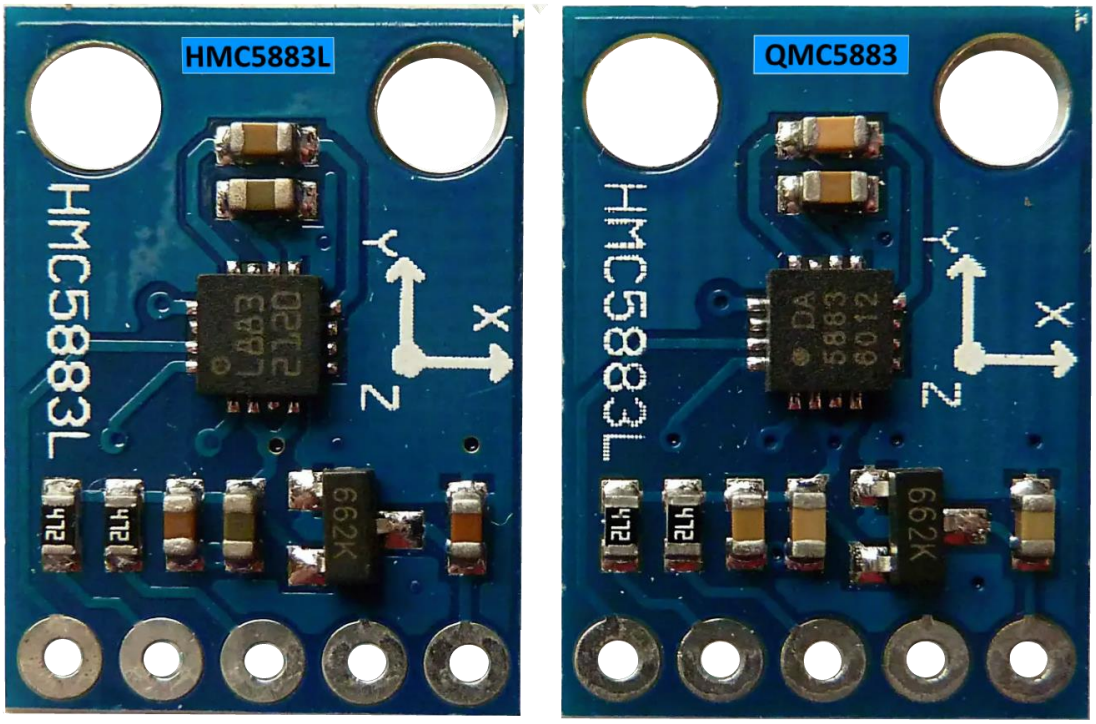
Εικόνα 27: Διάγραμμα σύνδεσης μαγνητόμετρου με Arduino (πηγή: Electric Diy Lab)

Το μαγνητόμετρο διαθέτει τις ακόλουθες πέντε (5) ακίδες:

- DRDY: Αυτή η ακίδα δημιουργεί μια διακοπή όταν τα δεδομένα εξόδου είναι διαθέσιμα.
- GND: Ακίδα γείωσης του αποστασιόμετρου που την συνδέουμε με τη γείωση του Arduino.
- SCL: Ακίδα ρολογιού τύπου I2C.
- SDA: Ακίδα δεδομένων τύπου I2C.
- VCC: Ακίδα τροφοδοσίας του μαγνητόμετρου που την συνδέουμε με την ακίδα των 5V του Arduino.

Οι δύο τύποι μαγνητόμετρου

Στην αγορά κυκλοφορούν δύο εκδόσεις μαγνητόμετρων. Αυτό που τις κάνει να διαφέρουν ο τύπος τους αν είναι τύπου **HMC5883L** ή **QMC5883**. Αυτές που διαθέτουν το IC **HMC5883L** έχουν μεγαλύτερη ακρίβεια στην αποτύπωση του μαγνητικού πεδίου της γης με αποκλίσεις που πειραματικά δεν υπερβαίνουν τις **0.3°** ενώ αυτές που διαθέτουν το IC **QMC5883** έχουν αισθητά μικρότερη ακρίβεια με αποκλίσεις που πειραματικά υπερβαίνουν τη **1°**.



Εικόνα 28: Οι δύο τύποι μαγνητόμετρου (πηγή: Surtr Technology)

Για να αναγνωρίσουμε αν το μαγνητόμετρο είναι τύπου **HMC5883L**:

- αρκεί να συνδέσουμε το μαγνητόμετρο με τον υπολογιστή, να τρέξουμε το sketch **I2cScanner** στο **Arduino IDE** και η διεύθυνση του μαγνητόμετρου αποδειχθεί ότι είναι η «**0x1E**» από το αποτέλεσμα της εκτέλεσης
- αρκεί το IC που βρίσκεται επάνω στην πλακέτα του μαγνητόμετρου να φέρει την ένδειξη **L883**

Για να αναγνωρίσουμε αν το μαγνητόμετρο είναι τύπου **QMC5883**:

- αρκεί να συνδέσουμε το μαγνητόμετρο με τον υπολογιστή, να τρέξουμε το sketch **I2cScanner** στο **Arduino IDE** και η διεύθυνση του μαγνητόμετρου αποδειχθεί ότι είναι η «**0x0D**» από το αποτέλεσμα της εκτέλεσης
- αρκεί το IC που βρίσκεται επάνω στην πλακέτα του μαγνητόμετρου να φέρει την ένδειξη **DA5883**

Το sketch **I2cScanner** σαρώνει το **I2C-bus** για συσκευές. Εάν εντοπιστεί μία συσκευή τότε αυτή καταγράφεται στη σειριακή οθόνη του **Arduino IDE**. Αυτό το sketch είναι το πρώτο βήμα για να λειτουργήσει η επικοινωνία **I2C**. Το sketch παρουσιάζει τις διευθύνσεις των 7-bit συσκευών που βρέθηκαν ως δεκαεξαδικές τιμές. Η τιμή που προκύπτει από το sketch μπορεί να χρησιμοποιηθεί για τη συνάρτηση «Wire.begin» η οποία χρησιμοποιεί τη διεύθυνση 7-bit. Ορισμένα datasheets χρησιμοποιούν διευθύνσεις των 8-bit και ορισμένα sketches χρησιμοποιούν δεκαδικές διευθύνσεις.

Πριν χρησιμοποιήσουμε το μαγνητόμετρο πρέπει αρχικά να αναζητήσουμε την κατάλληλη βιβλιοθήκη του βάσει του τύπου του (**HMC5883L** / **QMC5883**). Οι βιβλιοθήκες που λειτουργούν για τα δικά μου μαγνητόμετρα είναι η **Arduino-HMC5883L** (https://github.com/jarzebski/Arduino-HMC5883L) για τον τύπο **HMC5883L** και η **QMC5883LCompass** (https://github.com/mprograms/QMC5883LCompass) για τον τύπο **QMC5883**.

Έπειτα, πρέπει να βαθμονομήσουμε τα μαγνητόμετρά μας (διαδικασία calibration).

Για τον τύπο **HMC5883L**, τρέχουμε το sketch HMC5883L_calibrate.ino και κάνουμε περιστροφές του μαγνητόμετρου ως προς τις 3 διαστάσεις του χώρου. Όταν θεωρήσουμε ότι είμαστε έτοιμοι, αντιγράφουμε τις τιμές των πεδίων «*offX*» και «*offY*» και τις χρησιμοποιούμε στην συνάρτηση «*compass.setOffset(offX, offY);*» των κύριων sketches.

Για τον τύπο **QMC5883**, τρέχουμε το sketch calibration.ino και κάνουμε περιστροφές του μαγνητόμετρου ως προς τις 3 διαστάσεις του χώρου. Αφού ολοκληρωθεί το sketch μετά από την παρέλευση 5s, αντιγράφουμε την γραμμή που μας προτείνει με την προτροπή «*DONE. Copy the line below and paste it into your projects sketch.*» και την εισάγουμε αμέσως μετά από την γραμμή «*compass.init();*» των κύριων sketches.

Βιβλιοθήκη «QMC5883L Compass» (QMC5883L Compass Arduino Library)

Στην δική μου κατασκευή χρησιμοποιήθηκε μαγνητόμετρο τύπου **QMC5883L**.

Η βιβλιοθήκη «**QMC5883L Compass**» είναι μια βιβλιοθήκη Arduino που προορίζεται για την χρήση αισθητήρα της σειράς **QMC5583L** ως μαγνητόμετρου. Υποστηρίζει τις ακόλουθες ενέργειες:

- Λήψη κατεύθυνσης εδράνου αζιμούθιου 16 σημείων (0 – 15)
- Λήψη ονομάτων που φέρουν αζιμούθιο 16 σημείων (N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW)

- Λήψη τιμών του άξονα XYZ
- Εξομάλυνση των ενδείξεων XYZ μέσω κυλιόμενου μέσου όρου και ελάχιστης / μέγιστης αφαίρεσης
- Προαιρετικές λειτουργίες chipset
- Υπολογισμός αζιμούθιου

Παράδειγμα κώδικα για χρήση του μαγνητόμετρου με τη βιβλιοθήκη «**QMC5883L Compass**»:

```
// Φόρτωση βιβλιοθήκης μαγνητόμετρου
#include <QMC5883LCompass.h>

// Ορισμός μαγνητόμετρου
QMC5883LCompass compass;

// Αρχικοποίηση του μαγνητόμετρου στη συνάρτηση setup()
void setup(){
  compass.init();
}

// Κλήση του μαγνητόμετρου στη συνάρτηση loop():
void loop()
{
  int az = compass.getAzimuth();
  Serial.println((String)"Azimuth = " + az);
}
```

Μπαταρία λιθίου (LiPo battery pack) Gens Ace

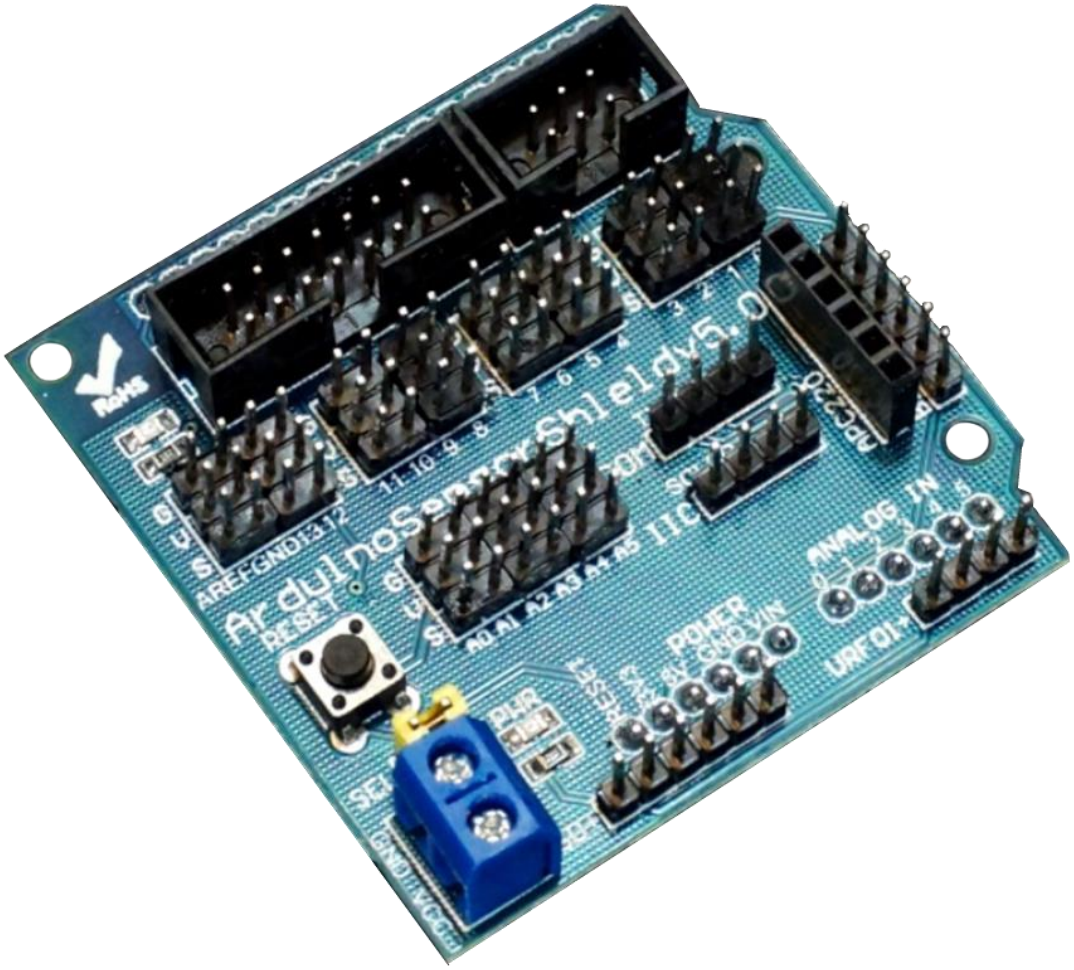
Εικόνα 29: Μπαταρία λιθίου Gens Ace 1600mAh 7.4V 45C 2S1P με πρίζα XT60 (πηγή: Technokap)

Η μπαταρία λιθίου που χρησιμοποιώ στο ρομπότ μου είναι η μπαταρία **Gens Ace** LiPo χωρητικότητας 1600mAh και τάσης 7.4V με πρίζα XT60. Η **Gens Ace** είναι μία από τις διασημότερες εταιρείες μπαταριών με μεγάλη εμπειρία. Όλες οι μπαταρίες της **Gens Ace** είναι βελτιστοποιημένες για πολλές κλίμακες των μοντέλων RC, όπως ηλεκτρικά και ηλεκτρονικά οχήματα, ηλεκτρικά εργαλεία, παιχνίδια, ψηφιακά προϊόντα κ.ά.

Τεχνικά χαρακτηριστικά	
Βάρος	0.102kg
Διαστάσεις	105 x 35 x 14mm
Ρυθμός εκφόρτισης	45C
Τάση	7.4V / 2S
Χωρητικότητα	1600mAh

Η φόρτιση της μπαταρίας πραγματοποιείται με τον φορτιστή μπαταριών iMAX B6AC.

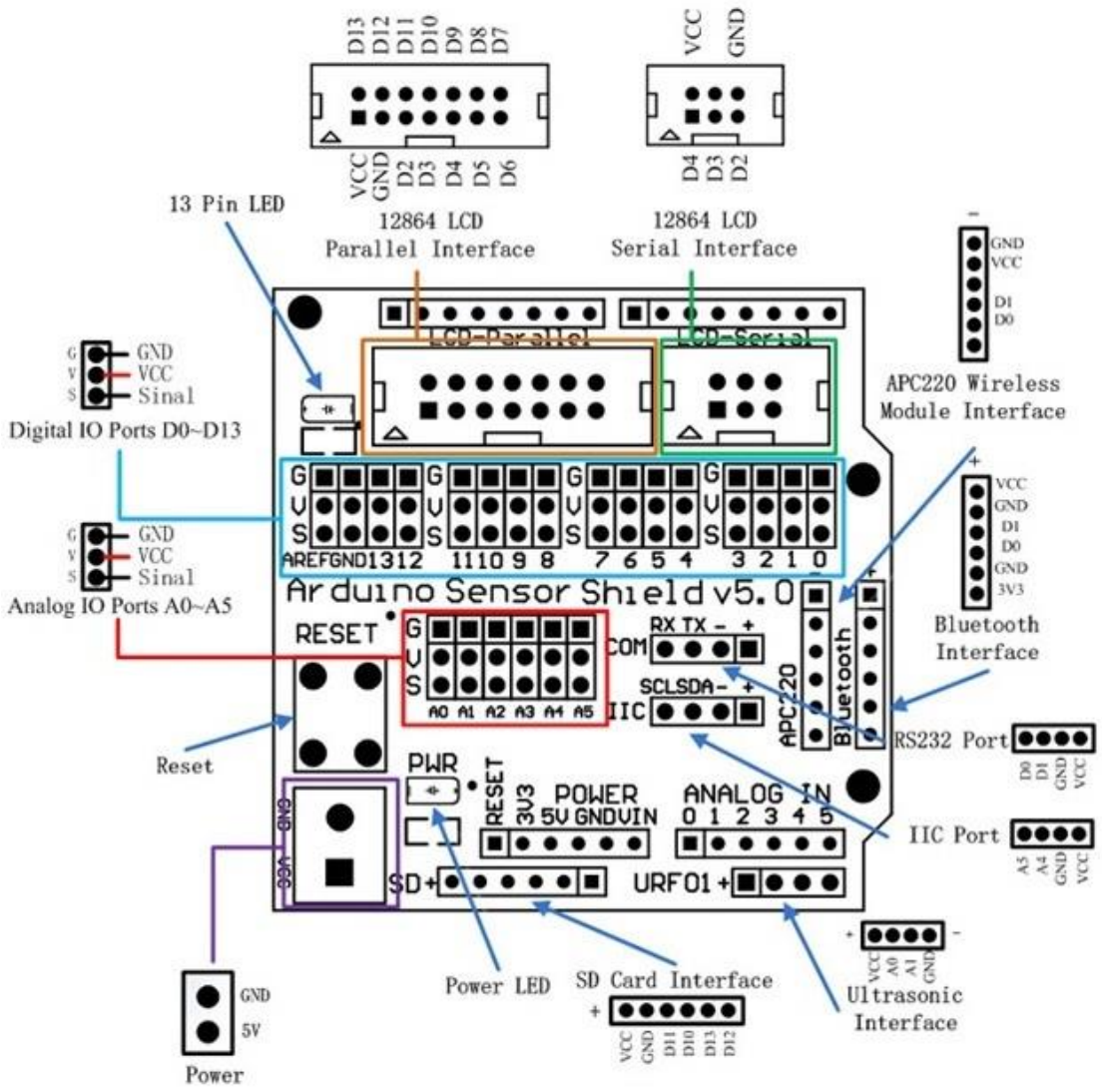
Πλακέτα επέκτασης Arduino (Sensor shield v5.0)



Εικόνα 30: Πλακέτα επέκτασης Arduino (πηγή: ARSSHIELD - Datasheet)

Η πλακέτα επέκτασης δίνει την δυνατότητα σύνδεσης του Arduino με διάφορες άλλες μονάδες όπως αισθητήρες, κουμπιά, ποτενσιόμετρα, ρελέ, σερβομηχανισμούς κ.ά. Κάθε μονάδα έχει μία ακίδα VCC και μία ακίδα GND για εύκολη σύνδεση και τροφοδοσία. Ένα πλεονέκτημα είναι η σύνδεση με εξωτερική πηγή ρεύματος για συσκευές που απαιτούν υψηλή ισχύ, όπως οι σερβομηχανές και οι κινητήρες. Δίπλα στην είσοδο ρεύματος είναι ένας βραχυκυκλωτήρας (jumper) που χρησιμοποιείται για την επιλογή μεταξύ της εσωτερικής τροφοδοσίας (μέσω του Arduino) ή της εξωτερικής (από μία εξωτερική ηλεκτρική πηγή). Η πλακέτα επέκτασης πρέπει να τροφοδοτείται από εξωτερική ηλεκτρική πηγή όταν χρησιμοποιούνται σερβομηχανές και κινητήρες επειδή δεν επαρκεί το ρεύμα του Arduino και επειδή δημιουργούν πάρα πολύ ηλεκτρικό θόρυβο. Ας σημειωθεί ότι η συγκεκριμένη πλακέτα επέκτασης είναι συμβατή μόνο το Arduino Uno και το Arduino Mega.

Τεχνικά χαρακτηριστικά	
Βάρος	0.025kg
Διαστάσεις	57 x 19 x 57mm
Ένδειξη λειτουργίας LED	ενσωματωμένη
Πλήθος αναλογικών εισόδων/εξόδων (Analog I/O ports)	6
Πλήθος ψηφιακών εισόδων/εξόδων (Digital I/O ports)	14

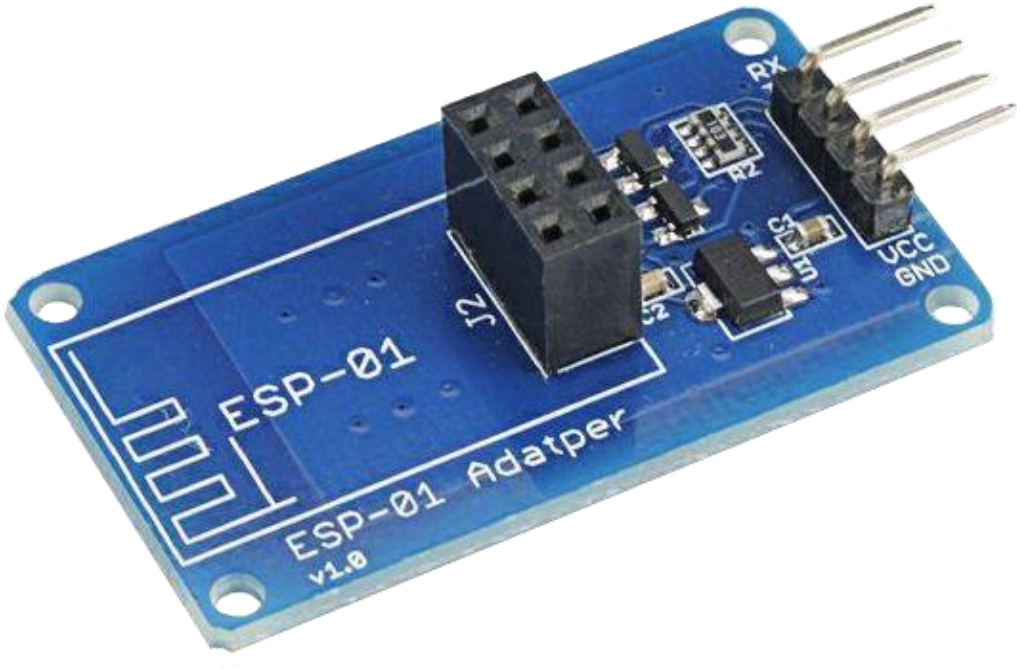


Εικόνα 31: Διάγραμμα πλακέτας επέκτασης Arduino (πηγή: GRobotronics)

Οι συνδέσεις που προσφέρει η πλακέτα επέκτασης συνοψίζονται στις ακόλουθες:

- PIN13 LED Pilot
- Αναλογικές είσοδοι/έξοδοι (Digital IO ports): A0-A5
- Διεπαφή APC220
- Διεπαφή Bluetooth
- Διεπαφή IIC
- Διεπαφή RS232 (TTL)
- Διεπαφή κάρτας μνήμης τύπου SD
- Διεπαφή υπερήχων (Ultrasonic)
- Είσοδος πηγής ρεύματος (Power In)
- Ένδειξη λειτουργίας LED (Power LED)
- Κουμπί επανεκκίνησης (Reset)
- Παράλληλη διεπαφή 12864 για οθόνη LCD
- Σειριακή διεπαφή 12864 για οθόνη LCD
- Ψηφιακές είσοδοι/έξοδοι (Analog IO ports): D0-D13

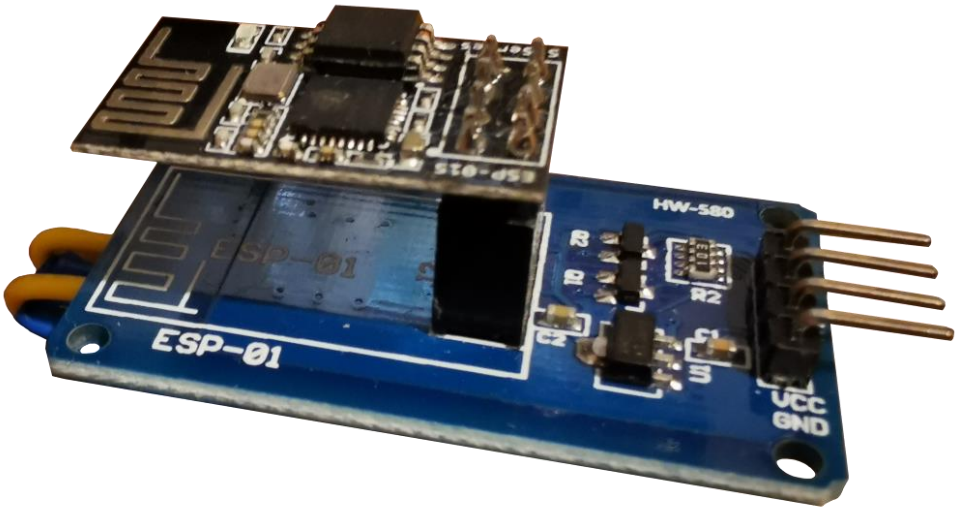
Προσαρμογέας ESP-01 (Adapter module ESP8266 3.3V/5V)



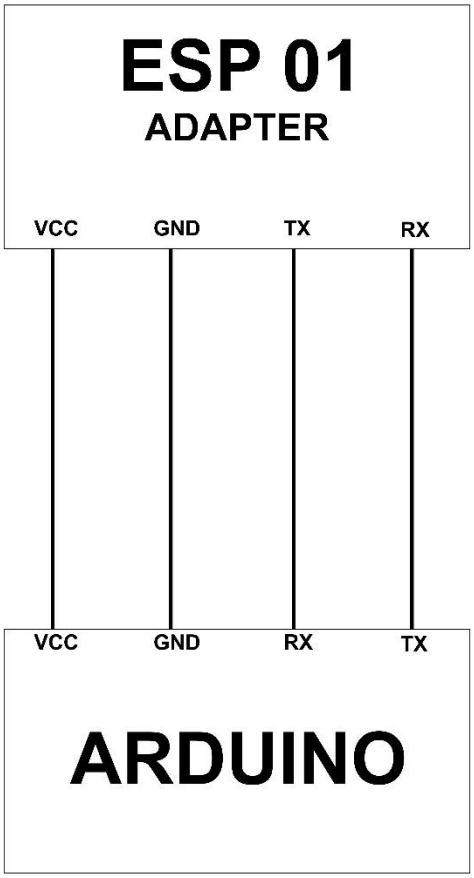
Εικόνα 32: Ο προσαρμογέας ESP-01 (πηγή: Top Electronics)

Τεχνικά χαρακτηριστικά	
Διαστάσεις	47.5 x 24 x 11.5mm
Ένταση ρεύματος	240mA
Συμβατότητα με ESP8266	ESP-01 / ESP-01S
Τάση λειτουργίας	DC 4.5 – 5V

Ο προσαρμογέας ESP-01 χρησιμοποιείται για να διευκολύνει την σύνδεση του ESP-01S με το Arduino επειδή παρέχει την απαραίτητη τάση των 3.3V στον αισθητήρα. Αυτός λειτουργεί με 5V από το Arduino αφού το τελευταίο δεν μπορεί να παρέχει απευθείας 3.3V στον ESP-01S.



Εικόνα 33: ESP-01S συνδεδεμένο με προσαρμογέα ESP-01



Εικόνα 34: Διάγραμμα σύνδεσης προσαρμογέα ESP-01 με Arduino (πηγή: GRobotronics)

Ο προσαρμογέας ESP-01 διαθέτει τις ακόλουθες τέσσερις (4) ακίδες:

- GND: Ακίδα γείωσης του κινητήρα που την συνδέουμε με τη γείωση του Arduino.
- RX: Ακίδα λήψης που την συνδέουμε με την ακίδα αποστολής TX του Arduino.
- TX: Ακίδα αποστολής που την συνδέουμε με την ακίδα λήψης RX του Arduino.
- VCC: Ακίδα τροφοδοσίας του κινητήρα που την συνδέουμε με την ακίδα των 5V του Arduino.

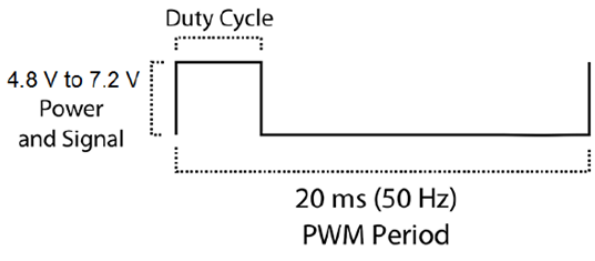
Η σύνδεση του προσαρμογέα με τον υπολογιστή γίνεται μέσω USB to Serial Converter.

Σερβο-κινητήρας (Metal gear servo**)** MG996R

Εικόνα 35: Σερβο-κινητήρας MG996R (πηγή: MagicDuino)

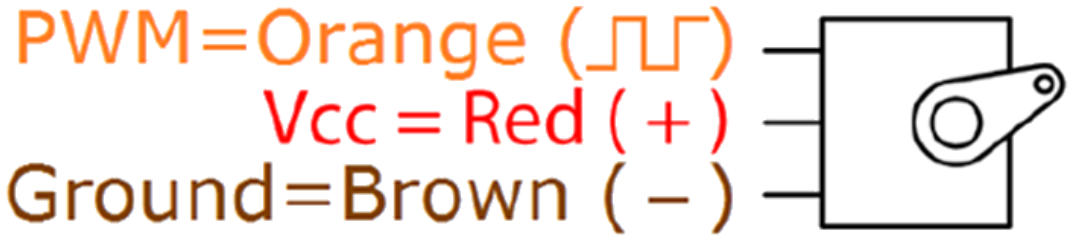
Ο σερβο-κινητήρας **MG996R** αποτελεί την εξέλιξη του σερβο-κινητήρα MG995. Η υψηλή του ροπή ξεπερνάει τα 10kg/cm. Ο σερβο-κινητήρα **MG996R** ενσωματώνει μεταλλικό γρανάζι και αντικραδασμική προστασία. Κυκλοφορεί με καλώδιο 30cm και θηλυκό βύσμα κεφαλής τύπου «S» πλήθους 3 ακίδων. Συνοδεύεται από βραχίονες και εξοπλισμό για αποδοτικότερους συνδυασμούς.

Τεχνικά χαρακτηριστικά	
Βάρος	0.055kg
Γωνία περιστροφής	180°
Διαστάσεις	40.7 x 19.7 x 42.9mm
Ένταση ρεύματος λειτουργίας	500 – 900mA (6V)
Ένταση ρεύματος ροπής	2.5A (6V)
Θερμοκρασία λειτουργίας	0° – 55°C
Πλάτος νεκρής ζώνης	5μs
Ροπή	9kg/cm (4.8V) – 11kg/cm (6V)
Τάση λειτουργίας	DC 4.8 – 6V
Ταχύτητα λειτουργίας	0.17s/60° (4.8V) – 0.14s/60° (7.2V)



Εικόνα 36: Παλμός σερβο-κινητήρα MG996R (πηγή: MagicDuino)

Η θέση των «0°» (1.5ms παλμός) είναι η μέση, η θέση των «90°» (~2ms παλμός) είναι τέρμα δεξιά και η θέση των «-90°» (~1ms παλμός) είναι τέρμα αριστερά.



Εικόνα 37: Ακίδες σερβο-κινητήρα MG996R (πηγή: MagicDuino)

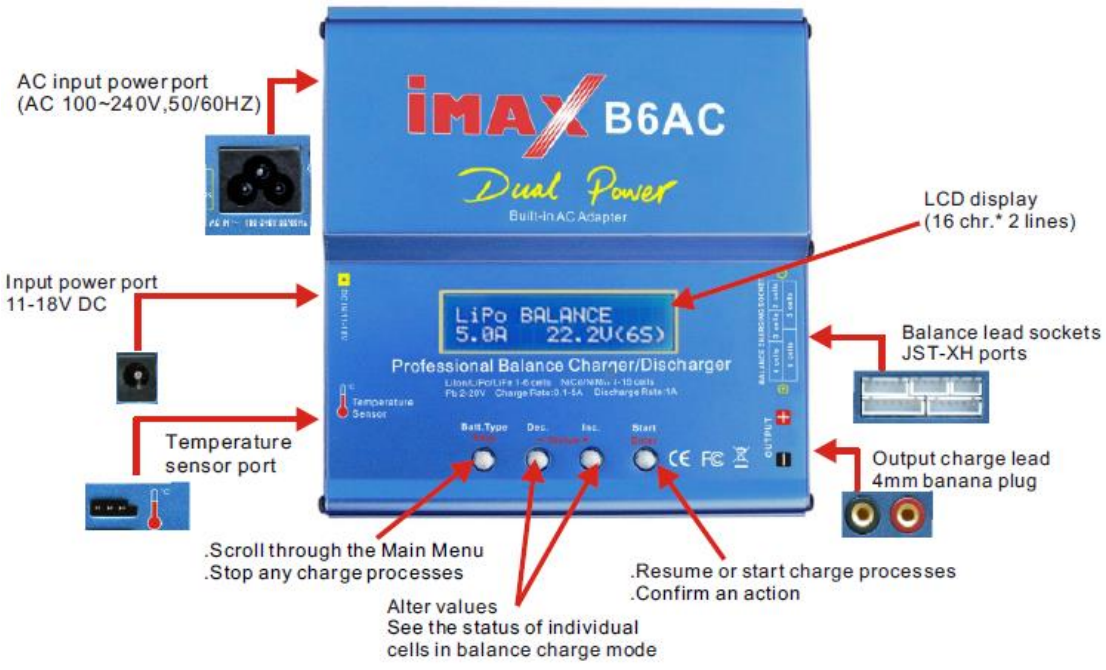
Ο σερβο-κινητήρας **MG996R** διαθέτει τις ακόλουθες τρεις (3) ακίδες:

- Ground: Ακίδα γείωσης του κινητήρα που την συνδέουμε με τη γείωση του Arduino. Το αντίστοιχο καλώδιο έχει συνήθως μαύρο ή καφέ χρώμα.
- PWM: Ακίδα παλμών του κινητήρα. Το αντίστοιχο καλώδιο έχει συνήθως κίτρινο, πορτοκαλί ή λευκό χρώμα.
- Vcc: Ακίδα τροφοδοσίας του κινητήρα που την συνδέουμε με την ακίδα των 5V του Arduino. Το αντίστοιχο καλώδιο έχει συνήθως κόκκινο χρώμα.

Φορτιστής/Αποφορτιστής μπαταριών (Battery charger/discharger) iMAX B6AC

Εικόνα 38: Φορτιστής μπαταριών iMAX B6AC (πηγή: Top Electronics)

Ο φορτιστής/αποφορτιστής μπαταριών **iMAX B6AC** είναι ένας ταχυφορτιστής με μικροεπεξεργαστή πολύ μεγάλης απόδοσης και διαθέτει ειδικό λογισμικό. Έχει τη δυνατότητα φόρτισης και αποφόρτισης μπαταριών. Επίσης, επιδιορθώνει τις μπαταρίες από το φαινόμενο μνήμης και επανενεργοποιεί τα αδρανή σημεία της μπαταρίας.

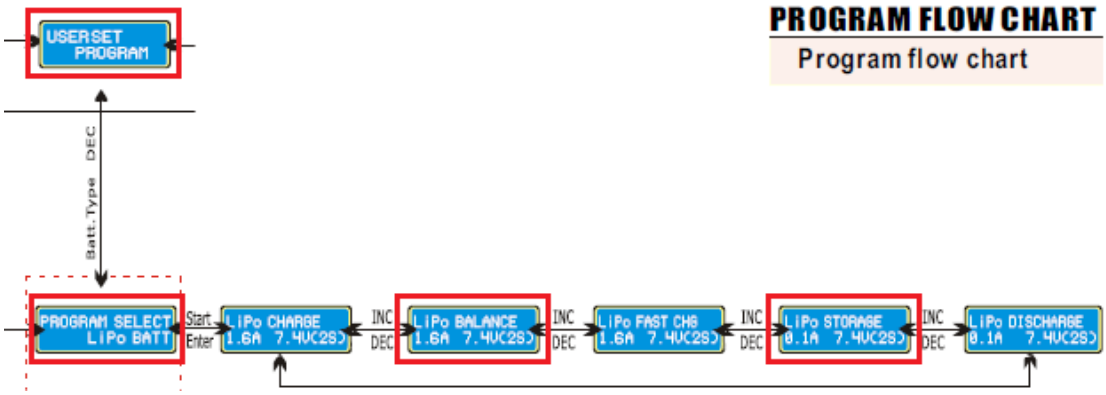


Εικόνα 39: Επεξήγηση μερών του φορτιστή μπαταριών iMAX B6AC με ετικέτες (πηγή: Pololu)

Τεχνικά χαρακτηριστικά	
Βάρος	0.531kg
Διαστάσεις	134 x 142 x 36mm
Ένταση ρεύματος αποφόρτισης	0.1 – 1.0A
Ένταση ρεύματος φόρτισης	0.1 – 5.0A
Μέγιστη ισχύς αποφόρτισης	5W
Μέγιστη ισχύς φόρτισης	50W
Παρελκόμενα συσκευασίας	6 x Καλώδια
Τάση λειτουργίας	DC 11 – 18V / AC 110 – 220V
Υποστηριζόμενοι τύποι μπαταρίας	LiFe, LiLo, LiPo, Pb, NiCD, NiMH

Εικόνα 40: Συνδεσμολογία φορτιστή μπαταριών iMAX B6AC με μπαταρία λιθίου Gens Ace

Ο φορτιστής αυτός χρησιμοποιείται για την φόρτιση της μπαταρίας λιθίου Gens Ace. Για να φορτίσω την μπαταρία λιθίου της κατασκευής μου, επιλέγω **LiPo BALANCE** με ένταση 1.6A και τάση 7.4V. Για να την αποθηκεύσω την μπαταρία λιθίου της κατασκευής μου, επιλέγω **LiPo STORAGE** με ένταση 0.1A και τάση 7.4V. Στο παρακάτω διάγραμμα σημείωσα την διαδρομή που πρέπει να ακολουθήσει κάποιος για να φορτίσει ή να αποθηκεύσει τη συγκεκριμένη μπαταρία.



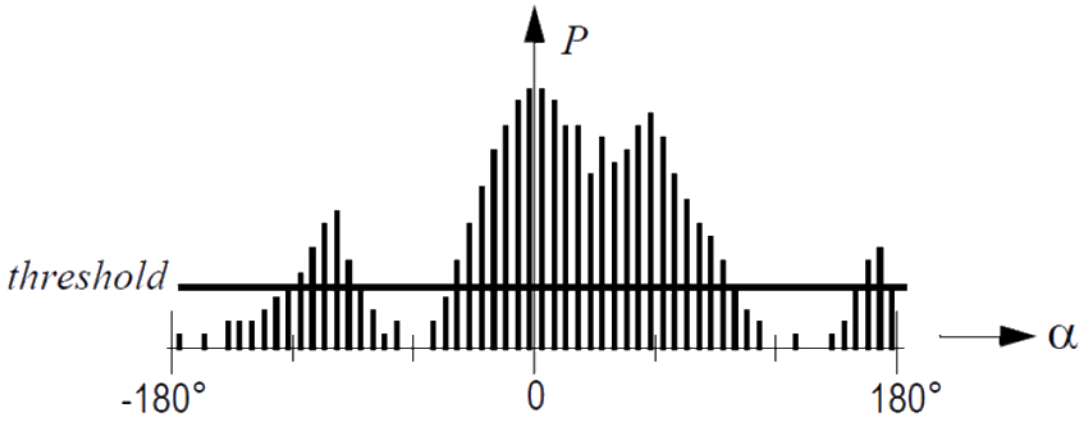
Εικόνα 41: Διάγραμμα ροής του προγράμματος του φορτιστή μπαταριών iMAX B6AC (πηγή: Pololu)

ΚΕΦΑΛΑΙΟ 4: VECTOR FIELD HISTOGRAM (VFH)

Ο αλγόριθμος «Ιστογράμματος Διανυσμάτων Πεδίου» (Vector Field Histogram ή **VFH**) αφορά την σχεδίαση μιας διαδρομής σε τοπικό επίπεδο. Για να σχεδιαστεί χρησιμοποιούνται αισθητήρες απόστασης που λαμβάνουν μετρήσεις γύρω από το ρομπότ σε ακτίνα 360^o. Μπορούμε με αυτόν τον τρόπο να σχεδιάσουμε ένα πιθανοτικό πλέγμα (occupancy grid) ώστε να χαρτογραφηθούν όλα τα εμπόδια τριγύρω.

Έπειτα, δημιουργούμε έναν πίνακα των πιθανοτήτων σύγκρουσης με κάθε εμπόδιο ανά γωνία μέτρησης. Στη συνέχεια, σχεδιάζουμε ένα ιστόγραμμα από αυτόν τον πίνακα. Στο ιστόγραμμα, ο οριζόντιος άξονας κυμαίνεται από -180^o έως 180^o και αφορά τις γωνίες για τις οποίες διαθέτουμε μετρήσεις ενώ ο κάθετος άξονας κυμαίνεται από 0 έως 1 και αφορά τις πιθανότητες σύγκρουσης. Ας σημειωθεί ότι στην κατασκευή μου ο οριζόντιος άξονας κυμαίνεται από 50^o έως 130^o με κέντρο τις 90 ^{o} στην αρχή των αξόνων επειδή κινούμαστε μόνο εμπρός.

Το επόμενο βήμα είναι να βρούμε το κατώφλι (threshold) του ιστογράμματος σε συνδυασμό με τις πραγματικές διαστάσεις του ρομπότ ώστε να εντοπίσουμε τα ανοίγματα μέσα από τα οποία μπορεί να διέλθει το ρομπότ.



Εικόνα 42: Παράδειγμα ιστογράμματος που δημιουργήθηκε με τον αλγόριθμο VFH (πηγή: Σπύρος Καζαρλής)

Έπειτα, δημιουργούμε έναν καινούργιο πίνακα πιθανοτήτων στον οποίο διατηρούνται όλες οι τιμές των πιθανοτήτων που βρίσκονται πάνω από το κατώφλι και μηδενίζονται αυτές που είναι κάτω από το κατώφλι.

Με χρήση του τελευταίου πίνακα αποφασίζουμε για την καλύτερη διαδρομή που θα ακολουθήσει το ρομπότ ως προς την απόσταση που αυτό καλείται να διανύσει. Διακρίνουμε τρεις βασικές περιπτώσεις:

1. Την περίπτωση να υπάρχει μονοπάτι ακριβώς μπροστά και εφόσον το πλάτος του είναι μεγαλύτερο από αυτό του ρομπότ να διέλθει το ρομπότ.
2. Την περίπτωση να υπάρχει ένα μονοπάτι δεξιά ή αριστερά από τη μέση και εφόσον το πλάτος του είναι μεγαλύτερο από αυτό του ρομπότ να διέλθει το ρομπότ.
3. Την περίπτωση να υπάρχουν δύο ή περισσότερα μονοπάτια δεξιά και αριστερά από τη μέση και εφόσον το πλάτος τους είναι μεγαλύτερο από αυτό του ρομπότ, να βαθμολογηθούν με μία συνάρτηση κόστους G και να διέλθει το ρομπότ από το μονοπάτι με την μικρότερη τιμή κόστους.

Η συνάρτηση κόστους G είναι η ακόλουθη:

$$\boldsymbol{G = a * target\_direction + b * wheel\_orientation + c * previous\_direction}$$

όπου:

- $target\_direction$: είναι η απόκλιση της εκάστοτε κατεύθυνσης της κίνησής μας σε μοίρες από τις 0^o της βασικής ευθείας του στόχου
- $wheel\_orientation$: είναι η απόκλιση σε μοίρες της στροφής των τροχών σε ένα τετράτροχο όχημα από αυτήν που έχουν ήδη
- $previous\_direction$: είναι η γωνία κίνησης του ρομπότ σε μοίρες από την προηγούμενη φορά
- a, b, c: είναι οι συντελεστές-βάρη της συνάρτησής μας και ορίζονται από εμάς βάση της βαρύτητας που θέλουμε να δώσουμε στις προηγούμενες μεταβλητές

Λόγω του μοντέλου κίνησης των δύο (2) τροχών με castor του ρομπότ της κατασκευής μου, αντί τεσσάρων (4) τροχών ισχύει ότι:

$$wheel\_orientation = 0$$

Οπότε η συνάρτηση G απλοποιείται και παίρνει την ακόλουθη μορφή:

$$G = a * target\_direction + c * previous\_direction$$

Η δική μου συνάρτηση κόστους G που χρησιμοποιώ μαζί με τα βάρη της είναι η:

$$\boldsymbol{G = 0.67 * target\_direction + 0.33 * previous\_direction}$$

ΚΕΦΑΛΑΙΟ 5: ΔΙΤΡΟΧΟ ΡΟΜΠΟΤ ΜΕ CASTOR

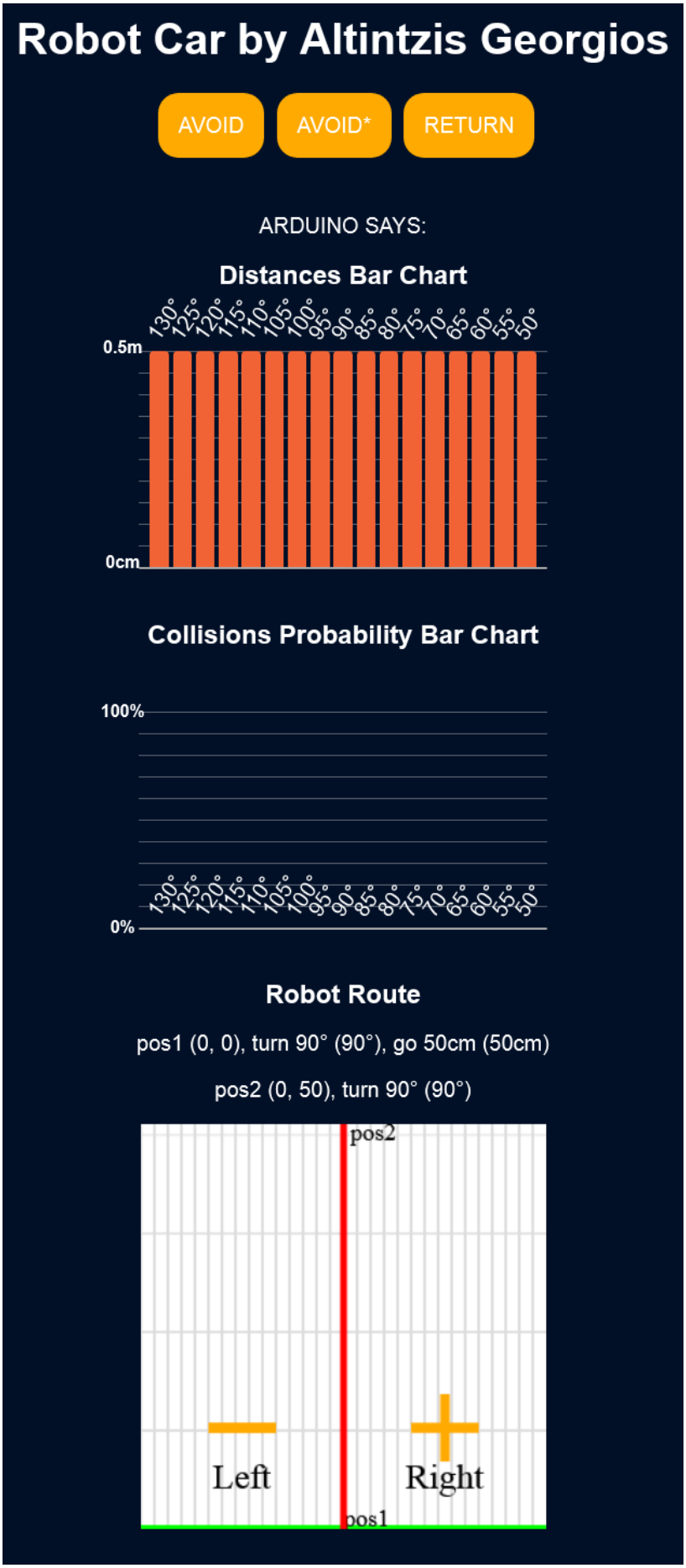
Κίνηση χωρίς εμπόδια

Αποφυγή εμποδίων (Εκτέλεση της επιλογής «AVOID»)

Εικόνα 43: Θέση ρομπότ: (0, 0), κανένα εμπόδιο

Εικόνα 44: Ολοκλήρωση σάρωσης εμποδίων από δεξιά προς τα αριστερά

Εικόνα 45: Κίνηση ρομπότ εμπρός κατά 50cm, θέση ρομπότ: (0, 50)


Robot Car by Altintzis Georgios
AVOID
AVOID*
RETURN
ARDUINO SAYS:
Distances Bar Chart
0.5m
0cm
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Collisions Probability Bar Chart
100%
0%
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Robot Route
pos1 (0, 0), turn 90° (90°), go 50cm (50cm)
pos2 (0, 50), turn 90° (90°)
pos2
Left
Right
pos1

Κίνηση με εμπόδια στα δεξιά

Αποφυγή εμποδίων (Εκτέλεση της επιλογής «AVOID»)

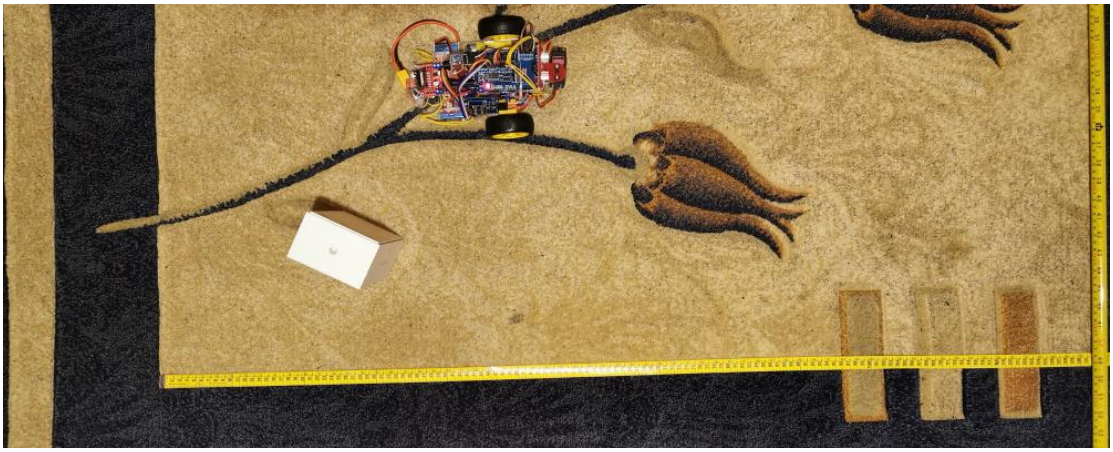
Εικόνα 46: Θέση ρομπότ: (0, 0), εμπόδιο δεξιά στα 20cm

Εικόνα 47: Ολοκλήρωση σάρωσης εμποδίων από δεξιά προς τα αριστερά

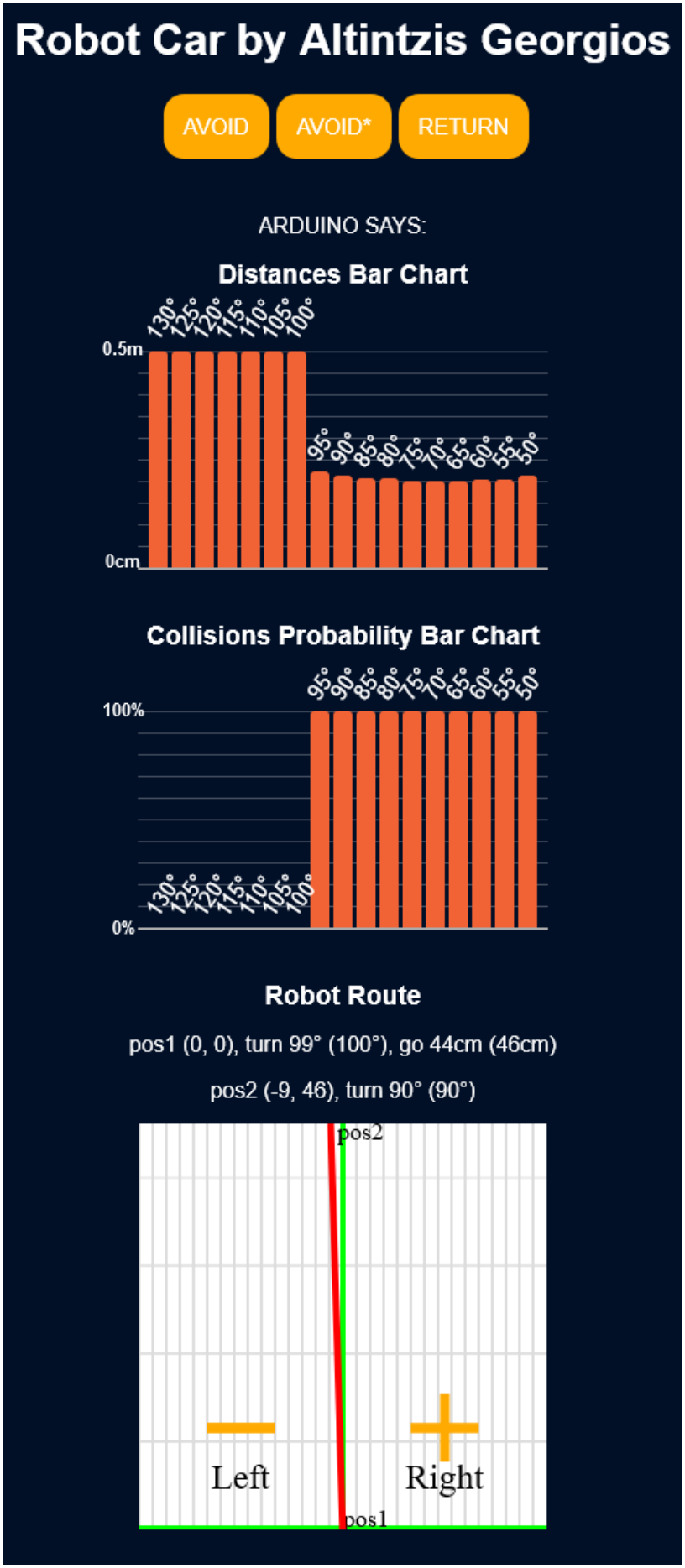
Εικόνα 48: Περιστροφή ρομπότ κατά 100°

Εικόνα 49: Έλεγχος ύπαρξης εμποδίων εμπρός

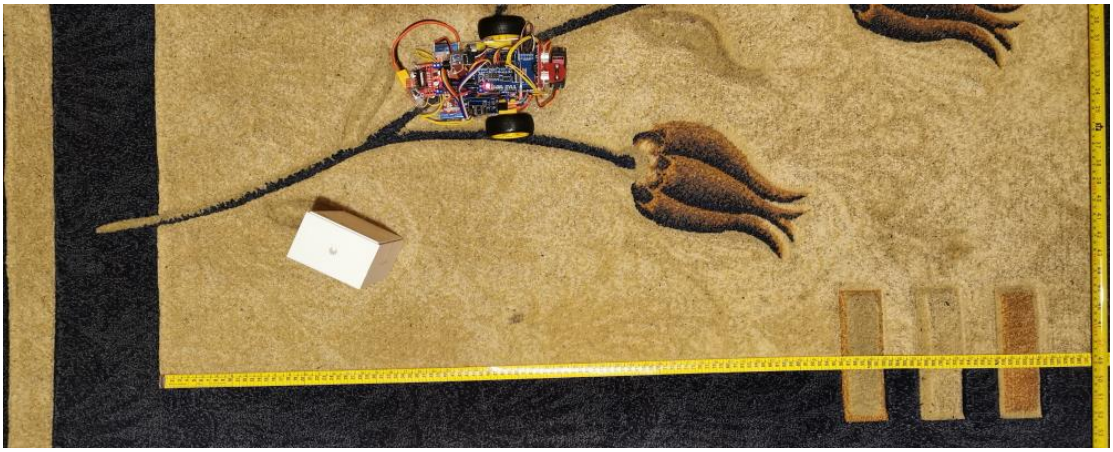
Εικόνα 50: Κίνηση ρομπότ εμπρός κατά 46cm



Εικόνα 51: Περιστροφή ρομπότ κατά 90°, θέση ρομπότ: (-9, 46)


Robot Car by Altintzis Georgios
AVOID
AVOID*
RETURN
ARDUINO SAYS:
Distances Bar Chart
0.5m
0cm
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Collisions Probability Bar Chart
100%
0%
95° 90° 85° 80° 75° 70° 65° 60° 55° 50° 130° 125° 120° 115° 110° 105° 100°
Robot Route
pos1 (0, 0), turn 99° (100°), go 44cm (46cm)
pos2 (-9, 46), turn 90° (90°)
pos2
Left
Right
pos1

Επιστροφή στη βασική ευθεία κίνησης (Εκτέλεση της επιλογής «RETURN»)

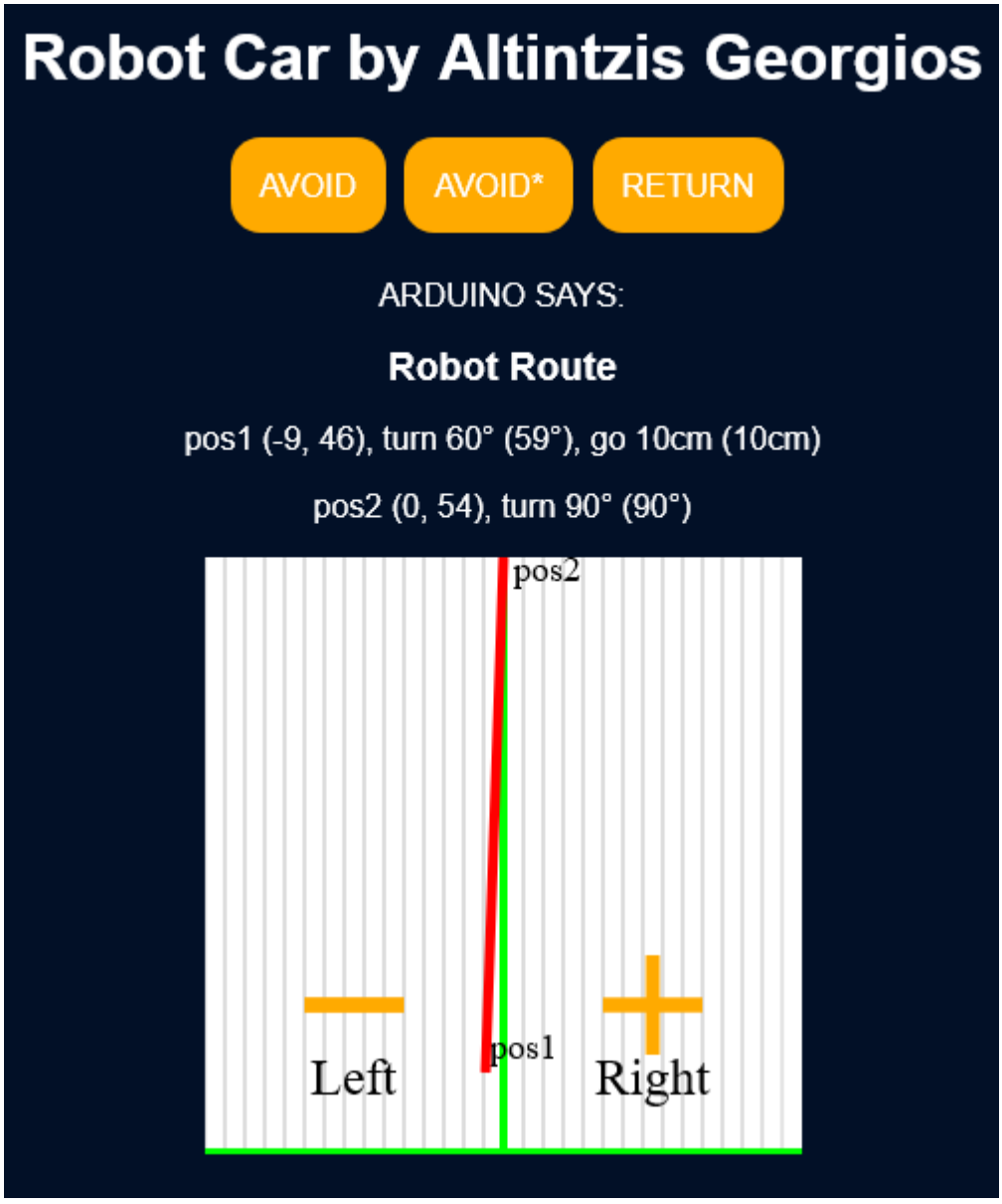


Εικόνα 52: Θέση ρομπότ: (-9, 46)

Εικόνα 53: Περιστροφή ρομπότ κατά 59°

Εικόνα 54: Κίνηση ρομπότ εμπρός κατά 10cm

Εικόνα 55: Περιστροφή ρομπότ κατά 90°, θέση ρομπότ: (0, 54)



Κίνηση με εμπόδια στα αριστερά

Αποφυγή εμποδίων (Εκτέλεση της επιλογής «AVOID»)

Εικόνα 56: Θέση ρομπότ: (0, 0), εμπόδιο αριστερά στα 20cm

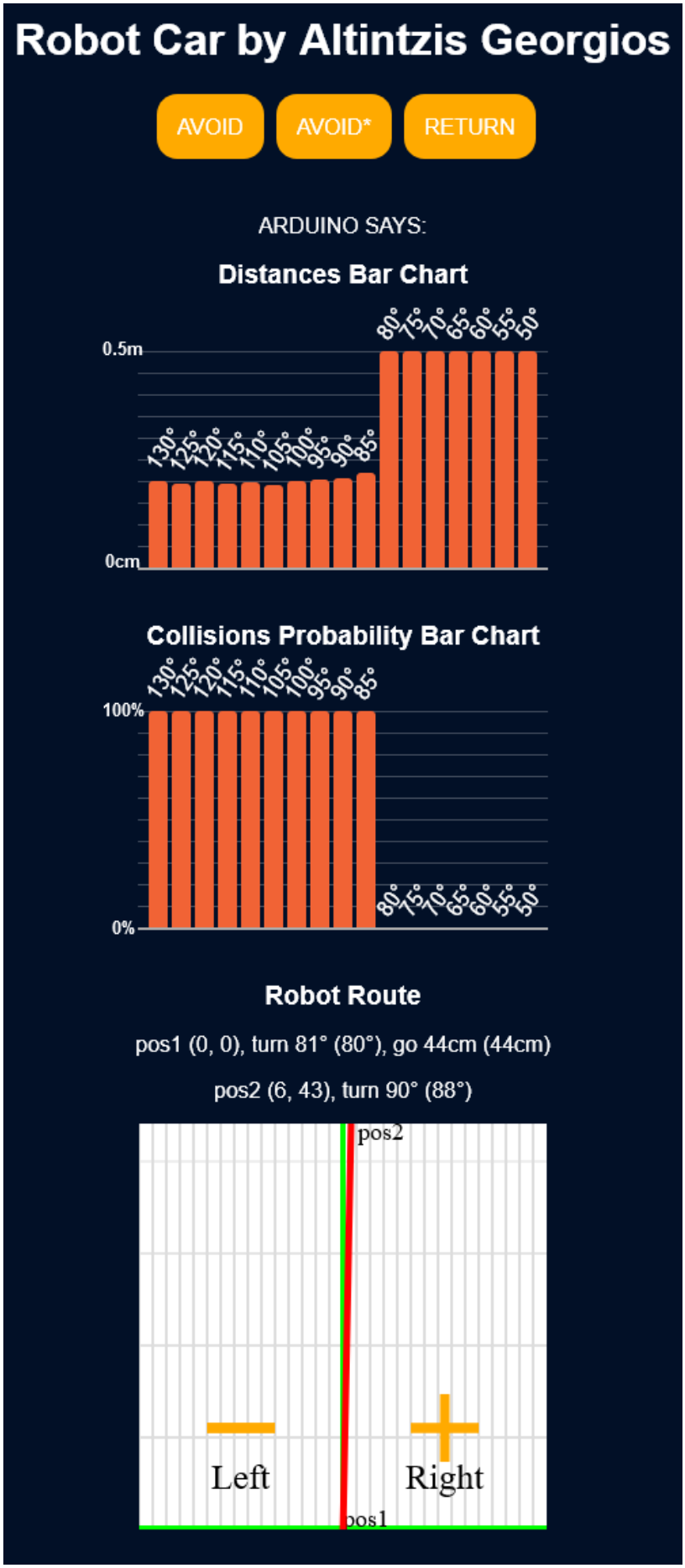
Εικόνα 57: Ολοκλήρωση σάρωσης εμποδίων από δεξιά προς τα αριστερά

Εικόνα 58: Περιστροφή ρομπότ κατά 80°

Εικόνα 59: Έλεγχος ύπαρξης εμποδίων εμπρός

Εικόνα 60: Κίνηση ρομπότ εμπρός κατά 44cm

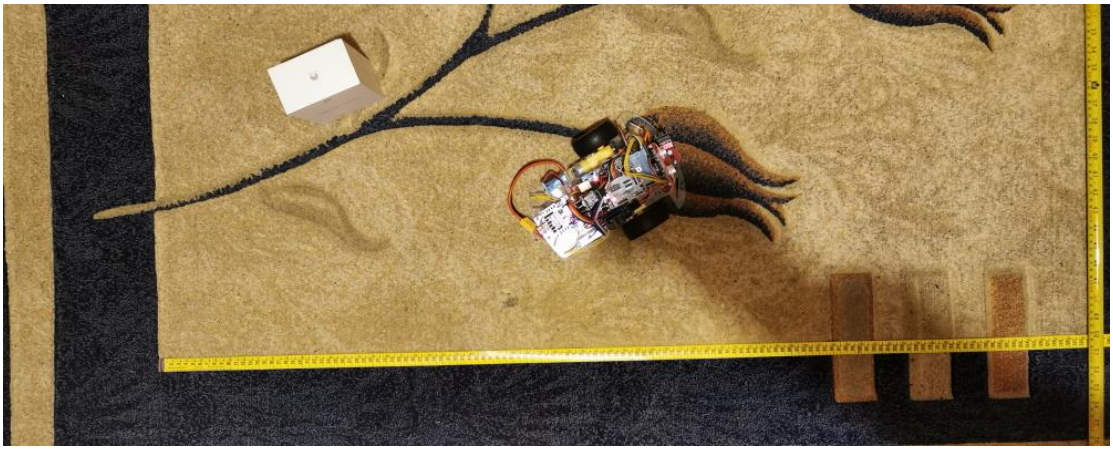
Εικόνα 61: Περιστροφή ρομπότ κατά 88°, θέση ρομπότ: (6, 43)


Robot Car by Altintzis Georgios
AVOID
AVOID*
RETURN
ARDUINO SAYS:
Distances Bar Chart
0.5m
0cm
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Collisions Probability Bar Chart
100%
0%
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Robot Route
pos1 (0, 0), turn 81° (80°), go 44cm (44cm)
pos2 (6, 43), turn 90° (88°)
pos2
Left
Right
pos1

Επιστροφή στη βασική ευθεία κίνησης (Εκτέλεση της επιλογής «RETURN»)

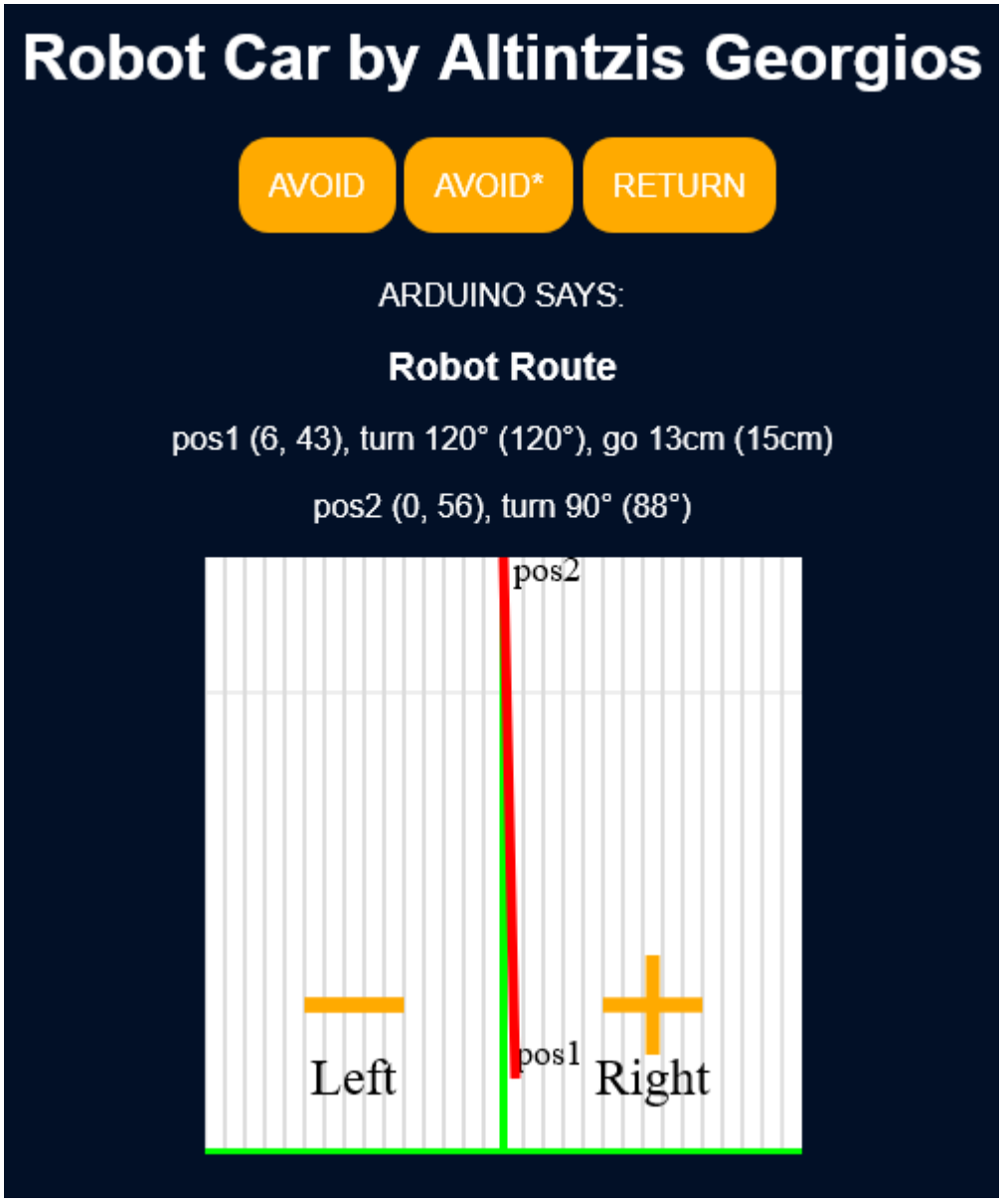
Εικόνα 62: Θέση ρομπότ: (6, 43)

Εικόνα 63: Περιστροφή ρομπότ κατά 120°



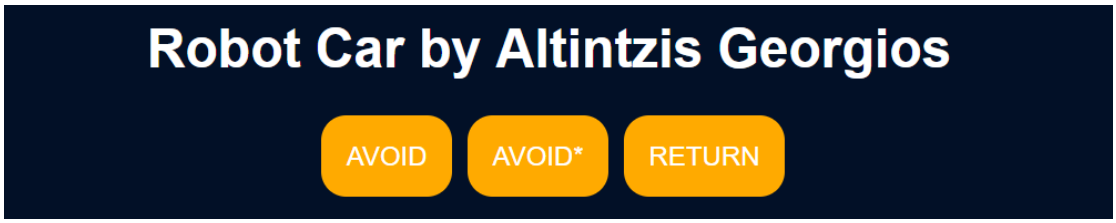
Εικόνα 64: Κίνηση ρομπότ εμπρός κατά 15cm

Εικόνα 65: Περιστροφή ρομπότ κατά 88°, θέση ρομπότ: (0, 56)



ΚΕΦΑΛΑΙΟ 6: ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΟ ΡΟΜΠΟΤ

Το ρομπότ μεταδίδει μέσω WiFi την δική του ιστοσελίδα και δίνει 3 επιλογές με κουμπιά στην ιστοσελίδα του (ως server): **AVOID**, **AVOID*** και **RETURN**. Προφανώς, το ρομπότ μπορεί να ελεγχθεί, πέρα από έναν Η/Υ, και με την χρήση οποιασδήποτε άλλης συσκευής που έχει τη δυνατότητα να προβάλλει ιστοσελίδες με τον δικό της περιηγητή όπως π.χ. ένα κινητό τηλέφωνο ή ένα tablet.



Με το κλικ σε κάποια από τις τρεις (3) επιλογές, το ρομπότ παράγει, σχεδόν ακαριαία, έναν χαρακτηριστικό ήχο τύπου beep ώστε να μας ενημερώσει ότι έλαβε και αναγνώρισε την επιλογή μας και να μας αποτρέψει από την επανάληψη του κλικ. Αμέσως μετά ξεκινάει την εκτέλεση των εντολών που αφορούν την εκάστοτε επιλογή μας. Το ρομπότ, πριν από τις κινήσεις που θα κάνει για να υλοποιήσει την επιλογή μας, επανεκμπέμπει την αρχική ιστοσελίδα του αλλά με μία πολύ σημαντική διαφορά. Πλέον ενσωματώνει στον κώδικά της, στο τμήμα «head», την ακόλουθη γραμμή κώδικα:

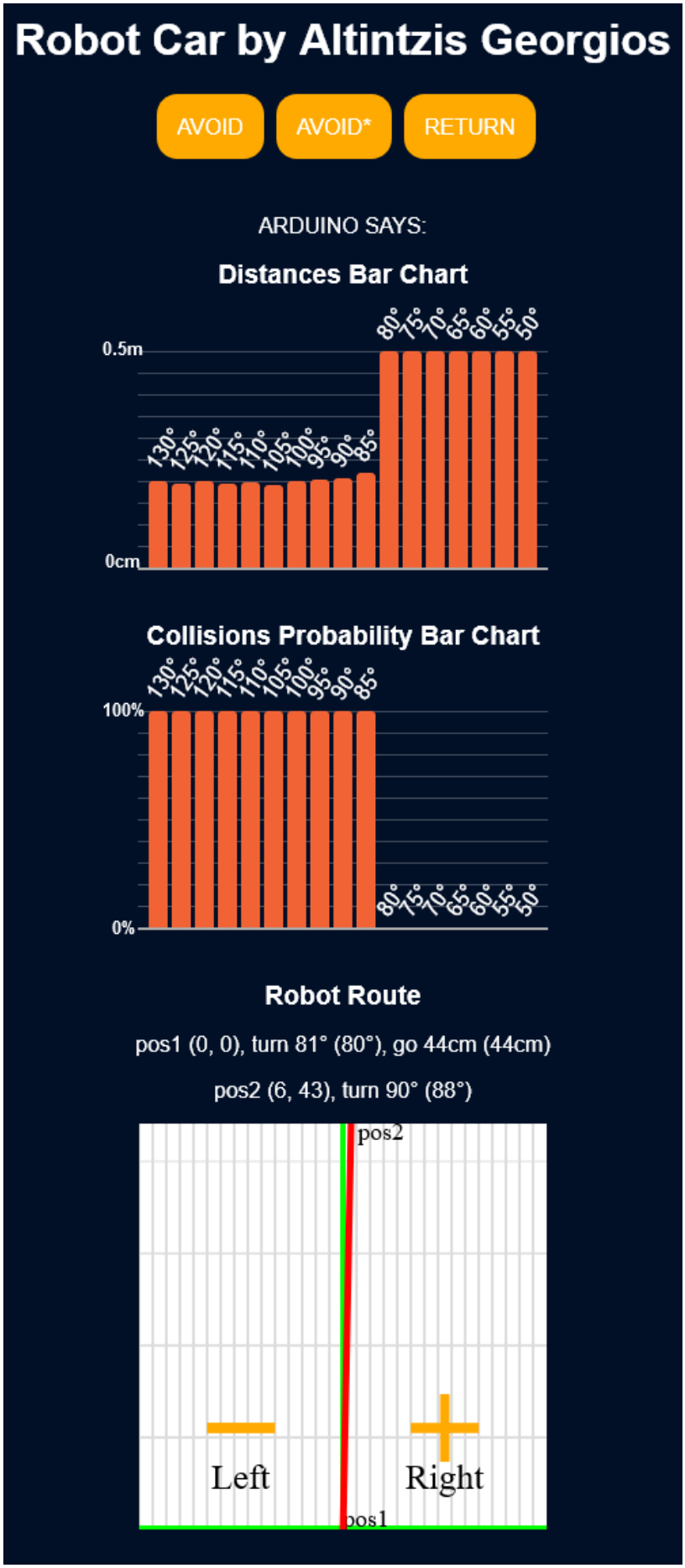
```
// Ανανέωση της αρχικής ιστοσελίδας κάθε 10 δευτερόλεπτα
<meta http-equiv=”refresh” content=”10; url=/”>
```

Με αυτήν την γραμμή κώδικα, θα γίνεται, αυτόματα στον περιηγητή μας, ανανέωση της ιστοσελίδας του ρομπότ με τέτοιον τρόπο ώστε από την 1^{η} κιόλας φορά που θα γίνει ανανέωση στα επόμενα 10s, θα λάβουμε τα αποτελέσματα από τις κινήσεις του ρομπότ με βάση την επιλογή μας. Η καινούργια ιστοσελίδα που θα προκύψει δεν θα περιέχει αυτήν την γραμμή κώδικα και έτσι δεν θα ανανεώνεται αλλά θα περιμένει την επόμενη επιλογή μας.

Επιλογή AVOID και AVOID*

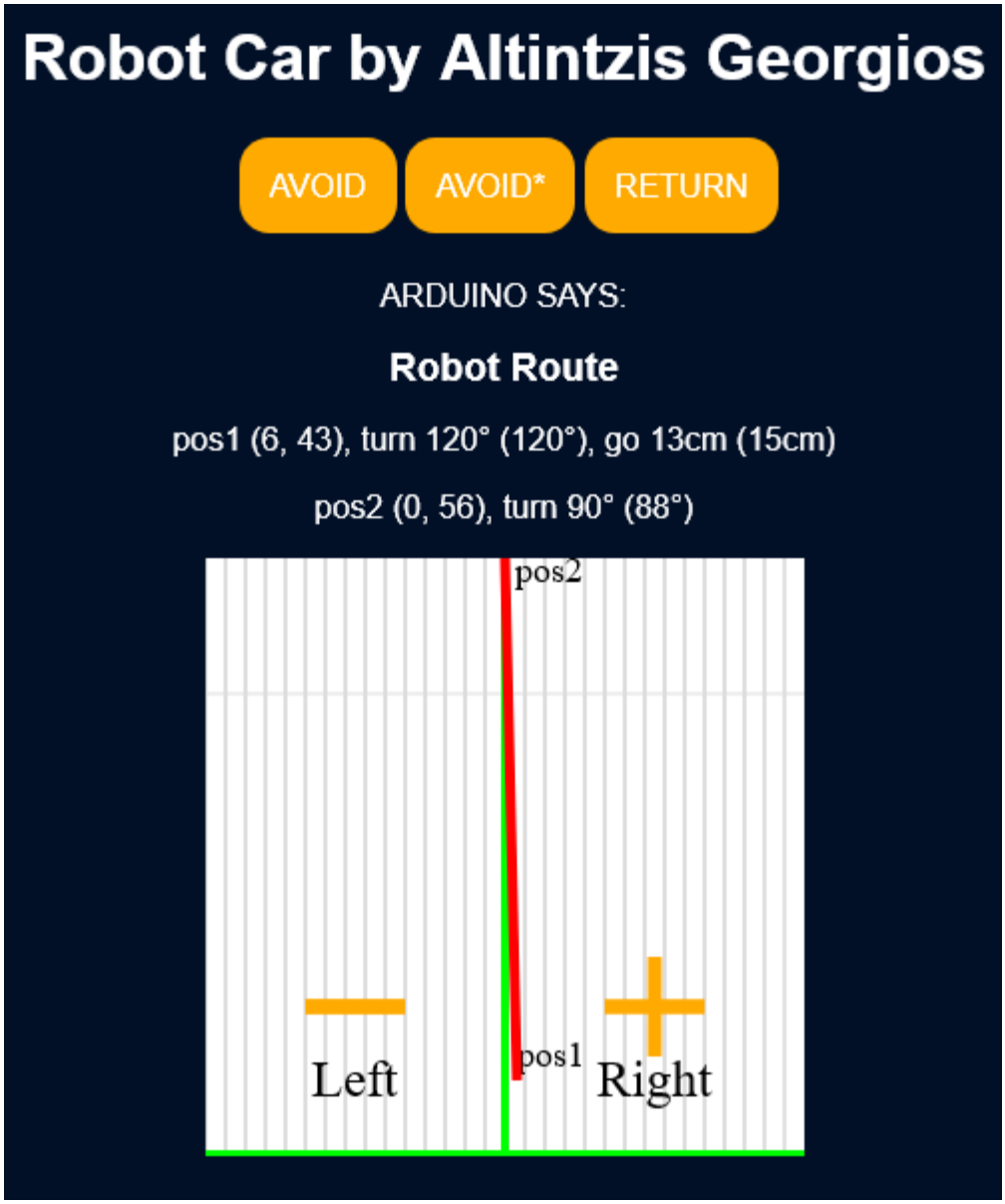
Με την επιλογή **AVOID**, το ρομπότ αποφασίζει για την καλύτερη διαδρομή και αποφεύγει τα εμπόδια κάθε φορά. Όταν ολοκληρώσει την κίνησή **AVOID**, το ρομπότ στρέφεται σε ευθεία παράλληλη με τη βασική ευθεία κίνησης. Ως απάντηση το ρομπότ επιστρέφει μία νέα ιστοσελίδα στην οποία παρουσιάζει με γράφημα τις αποστάσεις που συνάντησε και σε ποιες γωνίες μπροστά από αυτό. Επίσης, παρουσιάζει με γράφημα τι πιθανότητες σύγκρουσης ανά γωνία μπροστά από αυτό. Επιπλέον, δίνει αναλυτικά τις αποστάσεις που χρειάστηκε να διανύσει και τις γωνίες που χρειάστηκε να περιστραφεί. Μέσα στην κάθε παρένθεση αναφέρεται η πραγματική απόσταση που τελικά διένυσε το ρομπότ και η πραγματική γωνία με την οποία τελικά περιστράφηκε το ρομπότ. Τέλος, δίνει και μία γραφική απεικόνιση της πορείας του (με χρήση κλίμακας). Την επιλογή **AVOID**, μπορούμε να την χρησιμοποιήσουμε όσες φορές το επιθυμούμε για να αποφύγουμε διαδοχικά όλα τα εμπόδια της βασικής του ευθείας προς το στόχο. Το ρομπότ, μετά από κάθε κίνησή του, αποθηκεύει την θέση του στον χώρο αφού υπολογίσει τις συντεταγμένες του με μαθηματικούς υπολογισμούς.

Με την επιλογή **AVOID***, το ρομπότ πράττει ό,τι κάνει και με την επιλογή **AVOID** αλλά δεν σαρώνει πραγματικά την περιοχή εμπρός για εμπόδια. Η επιλογή **AVOID*** δημιουργεί τυχαία εικονικά εμπόδια.


Robot Car by Altintzis Georgios
AVOID
AVOID*
RETURN
ARDUINO SAYS:
Distances Bar Chart
0.5m
0cm
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Collisions Probability Bar Chart
100%
0%
130° 125° 120° 115° 110° 105° 100° 95° 90° 85° 80° 75° 70° 65° 60° 55° 50°
Robot Route
pos1 (0, 0), turn 81° (80°), go 44cm (44cm)
pos2 (6, 43), turn 90° (88°)
pos2
Left
Right
pos1

Επιλογή RETURN

Με την επιλογή RETURN επιστρέφει στη βασική ευθεία κίνησης όπου y=0 και x=[η συνολική κάθετη απόσταση του ρομπότ]. Επιπλέον, δίνει αναλυτικά τις αποστάσεις που χρειάστηκε να διανύσει και τις γωνίες που χρειάστηκε να περιστραφεί. Μέσα στην κάθε παρένθεση αναφέρεται η πραγματική απόσταση που τελικά διένυσε το ρομπότ και η πραγματική γωνία με την οποία τελικά περιστράφηκε το ρομπότ. Τέλος, δίνει και μία γραφική απεικόνιση της πορείας του (με χρήση κλίμακας).



ΚΕΦΑΛΑΙΟ 7: ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ

Εισαγωγή

Οι συναρτήσεις που δημιούργησα μπορούν να ομαδοποιηθούν για ευκολία σε επτά (7) βασικές κατηγορίες:

- Βασικές συναρτήσεις
- Βοηθητικές συναρτήσεις
- Συναρτήσεις VFH
- Συναρτήσεις ασύρματης επικοινωνίας με WiFi
- Συναρτήσεις θέσης
- Συναρτήσεις κίνησης
- Συναρτήσεις μαγνητικού πεδίου

Βασικές συναρτήσεις	
void setup ()	Υπορουτίνα αρχικοποίησης πλακέτας Arduino (setup)
void loop ()	Υπορουτίνα επανάληψης (loop)
void Left_ISR ()	Διακοπή (interrupt) αριστερού κινητήρα
void Right_ISR ()	Διακοπή (interrupt) δεξιού κινητήρα

Βοηθητικές συναρτήσεις	
float deg2rad (float degrees)	Υπορουτίνα υπολογισμού ακτινίων από μοίρες
void playBuzzer (String buzzerStatus)	Υπορουτίνα αναπαραγωγής ήχων
float rad2deg (float radians)	Υπορουτίνα υπολογισμού μοιρών από ακτίνια

Συναρτήσεις VFH	
int calcChordLength (byte leftLimit, byte rightLimit, word &angle, word &r)	Υπορουτίνα υπολογισμού χορδής νοητού κύκλου
float calcProbability (int previous, int current, int next)	Υπορουτίνα υπολογισμού πιθανοτήτων σύγκρουσης
char chooseBetweenGaps (int angleLeft, int angleRight)	Υπορουτίνα απόφασης G για την καλύτερη διαδρομή
void createProbabilityArrays ()	Υπορουτίνα δημιουργίας πίνακα πιθανοτήτων σύγκρουσης
bool findBestGap (word &chordLength, word &angle, word &r)	Υπορουτίνα εύρεσης του καλύτερου κενού για διέλευση του ρομπότ
word readDistance ()	Υπορουτίνα μέτρησης απόστασης με χρήση του αποστασιόμετρου
void readNextDistance ()	Υπορουτίνα μέτρησης απόστασης ανά γωνία εμπρός
void scanArea ()	Υπορουτίνα σάρωσης περιοχής
void scanVirtualArea ()	Υπορουτίνα εικονικής σάρωσης περιοχής

Συναρτήσεις ασύρματης επικοινωνίας με WiFi	
String buildAWebpage (String aReply)	Υπορουτίνα αποστολής ιστοσελίδας στο ESP8266 με δυνατότητα λήψης εντολής από τη συσκευή-πελάτη
bool initESP8266 ()	Υπορουτίνα αρχικοποίησης του ESP8266
bool initWiFi (byte CWMODE)	Υπορουτίνα αρχικοποίησης της σύνδεσης του ESP8266 σε ασύρματο δίκτυο WiFi
bool sendACommand (String command, const char* reply, unsigned int waitA, unsigned int waitB, boolean debug)	Υπορουτίνα αποστολής εντολών στο ESP8266 με δυνατότητα ελέγχου της απάντησης του ESP8266
bool sendAWebPage (int connectionId, String webpage, word timeDelay)	Υπορουτίνα αποστολής ιστοσελίδας στο ESP8266

Συναρτήσεις θέσης	
word calcDistanceToStop (int curAngle)	Υπορουτίνα υπολογισμού υποτείνουσας κίνησης στην θέση (0,y)
void calcPosXY (int targetAngle, word distance)	Υπορουτίνα υπολογισμού συντεταγμένων
word recalcTargetAngle (int targetAngle, word r)	Υπορουτίνα επαναϋπολογισμού γωνίας περιστροφής λόγω ασσυμετρίας κατασκευής

Συναρτήσεις κίνησης	
void go (byte dirAngle, byte speedInit, word targetDistance)	Υπορουτίνα κίνησης προς συγκεκριμένη κατεύθυνση
void moveForward (byte speedRight, byte speedLeft)	Υπορουτίνα κίνησης προς τα μπροστά
void startMoving ()	Υπορουτίνα εκκίνησης κινητήρων
void stopMoving ()	Υπορουτίνα σταματήματος
int turnByAngle (int endAngle, byte speed)	Υπορουτίνα περιστροφής

Συναρτήσεις μαγνητικού πεδίου	
int calcAzimuthFix ()	Υπορουτίνα υπολογισμού απόκλισης μαγνητικού πεδίου σε σχέση με τον προσανατολισμό του ρομπότ ώστε το ρομπότ να ξεκινάει πάντα με γωνία 90°
int getFixedAzimuth ()	Υπορουτίνα διόρθωσης τιμής μαγνητικού πεδίου
int getAzimuth ()	Υπορουτίνα λήψης απόκλισης σε μοίρες από τον Βορρά
void initCompass ()	Υπορουτίνα αρχικοποίησης μαγνητόμετρου

Κώδικας

```
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                        */
/*      ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ (ΔΙΠΑΕ)        */
/*    ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ     */
/*               ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΣΕΡΡΩΝ                   */
/*                                                        */
/*                ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ:                    */
/*                                                        */
/*      ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΑΠΟΦΥΓΗΣ ΕΜΠΟΔΙΩΝ ΣΕ          */
/*   ΠΡΑΓΜΑΤΙΚΟ ΑΥΤΟΚΙΝΟΥΜΕΝΟ ΡΟΜΠΟΤ ΜΕ ΤΗΝ ΜΕΘΟΔΟ        */
/*            VECTOR FIELD HISTOGRAM - VFH                */
/*                                                        */
/*            ΦΟΙΤΗΤΗΣ: ΑΛΤΙΝΤΖΗΣ ΓΕΩΡΓΙΟΣ               */
/*             ΚΑΘΗΓΗΤΗΣ: ΚΑΖΑΡΛΗΣ ΣΠΥΡΟΣ                */
/*                                                        */
/*                  ΦΕΒΡΟΥΑΡΙΟΣ 2023                      */
/*                                                        */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++ */



/************************/
/* ΟΡΙΣΜΟΣ ΒΙΒΛΙΟΘΗΚΩΝ */
/************************/

// Φόρτωση βιβλιοθήκης μαγνητόμετρου
#include <QMC5883LCompass.h>
// Ορισμός μαγνητόμετρου
QMC5883LCompass compass;

// Φόρτωση βιβλιοθήκης σερβο-κινητήρα
#include <Servo.h>
// Ορισμός σερβο-κινητήρα
Servo servo;



/************************************/
/* ΟΡΙΣΜΟΣ ΤΙΜΗΣ ΣΤΑΘΕΡΩΝ ΜΕΤΑΒΛΗΤΩΝ */
/************************************/

// Πλάτος ρομπότ [mm]
#define ROBOT_WIDTH 170
// Ακτίνα τροχού του ρομπότ [mm]
#define ROBOT_WHEEL_RADIUS 33
// Arduino Mega 2560 Rev3 Serial2 (RX2,TX2) <-> ESP8266 TX,RX
#define esp8266 Serial2
int serialCommunicationSpeed = 9600;
// Ρυθμός ανανέωσης συνάρτησης loop
int LOOPDELAY = 1000;
```

```
/*************************/
/* ΟΡΙΣΜΟΣ ΑΚΙΔΩΝ (PINS) */
/*************************/

// Ακίδες κυκλώματος οδήγησης κινητήρων L298N
// Έλεγχος ταχύτητας PWM δεξιού κινητήρα (καφέ καλώδιο)
byte enRightPin = 4;
// Έλεγχος ταχύτητας PWM αριστερού κινητήρα (μπλε καλώδιο)
byte enLeftPin  = 13;
// Πρόσθια κατεύθυνση δεξιού κινητήρα (κόκκινο καλώδιο)
byte in1Pin = 6;
// Οπίσθια κατεύθυνση δεξιού κινητήρα (πορτοκαλί καλώδιο)
byte in2Pin = 7;
// Πρόσθια κατεύθυνση αριστερού κινητήρα (κίτρινο καλώδιο)
byte in3Pin = 8;
// Οπίσθια κατεύθυνση αριστερού κινητήρα (πράσινο καλώδιο)
byte in4Pin = 9;

// Ακίδα σερβο-κινητήρα που κατευθύνει το αποστασιόμετρο
byte servoPin = 32; // Έξοδος PWM σερβο-κινητήρα

// Ακίδες αποστασιομέτρου
byte trigPin = 12;
byte echoPin = 11;


/************************/
/* ΥΠΟΛΟΙΠΕΣ ΜΕΤΑΒΛΗΤΕΣ */
/************************/

// Μεταβλητές για τις γωνίες στροφής του σερβο-κινητήρα
// Πλήθος γωνιών
#define NUM_ANGLES 17
// Πίνακας γωνιών [°]
byte sensorAngle[NUM_ANGLES] = {50, 55, 60, 65, 70, 75, 80,
85, 90, 95, 100, 105, 110, 115, 120, 125, 130};
word distances[NUM_ANGLES];      // Πίνακας αποστάσεων [mm]
byte angleIndex = 0;             // Δείκτης τρέχουσας γωνίας
char step = 1;                   // Κατεύθυνση σάρωσης χώρου
bool scanVirtualAreaSet = false; // Εικονική σάρωση εμποδιών

// Πίνακας πιθανοτήτων με κατώφλι
word probabilityT[NUM_ANGLES];

// Ασφαλής απόσταση εμπρός [mm]
int FAR_AWAY = 500;

// Ορισμός απόκλισης μαγνητικού πεδίου
int azimuthFix = 0;

// Ορισμός μετρητών διακοπών
volatile int left_intr, right_intr;

// Ορισμός μεταβλητών VFH
word previous_direction = 0;
word distance, max_distance, min_distance;
```

```
// Ορισμός συντεταγμένων θέσης (x και y) του ρομπότ
int posX, posY, positions[2][2];
word turnAngles[2][2], distancesArray[1][2];

// Ορισμός απόστασης στόχου
word tgtDist = 0;

// Ορισμός μεταβλητών επικοινωνίας
String aCommand, aReply;



/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*              || ΒΑΣΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ ||                   */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */


/*
 * Υπορουτίνα αρχικοποίησης πλακέτας Arduino (setup)
 */
void setup() {
  // Ακίδες αισθητήρων ταχύτητας
  byte speedLeftPin  = 2;
  byte speedRightPin = 3;

  // Αρχικοποίηση σειριακής επικοινωνίας
  //Serial.begin(serialCommunicationSpeed);
  //Serial.println(F("SETUP!"));

  // Αρχικοποίηση ασύρματης επικοινωνίας με WiFi
  initESP8266();

  // Αρχικοποίηση μαγνητόμετρου
  initCompass();

  // Αρχικοποίηση μεταβλητών VFH
  max_distance = min_distance = 0;

  // Αρχικοποίηση μεταβλητών VFH
  posX = posY = 0;

  // Αρχικοποίηση μετρητών διακοπών
  left_intr = right_intr = 0;

  // Ενεργοποίηση ακίδων αποστασιόμετρου
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Ενεργοποίηση ακίδων κυκλώματος οδήγησης κινητήρων L298N
  pinMode(enLeftPin, OUTPUT);
  pinMode(enRightPin, OUTPUT);
```

```
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);
  digitalWrite(in2Pin, LOW);
  digitalWrite(in4Pin, LOW);

  // Ενεργοποίηση διακοπών
  attachInterrupt(digitalPinToInterrupt(speedLeftPin),
Left_ISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(speedRightPin),
Right_ISR, CHANGE);

  // Ενεργοποίηση ακίδας σερβο-κινητήρα
  delay(100);
  servo.attach(servoPin);
  servo.write(90);
  //servo.detach();
  delay(1000);

  //Serial.println(F("hello world!"));
}



/*
 * Υποροutίνα επανάληψης (loop)
 */
void loop()
{
  //Serial.println((String)"+ aCommand = ["+aCommand+"],
aReply = ["+aReply+"]");

  // Έλεγχος νέων εντολών
  aCommand = buildAWebpage(aReply);


  // Επιλογή της εντολής AVOID* από τον χρήστη
  if ((String)"N" == aCommand)
  {
    // Ενεργοποίηση τυχαίας εμφάνισης εμποδίων
    scanVirtualAreaSet = true;
    // Επιλογή της εντολής AVOID από το πρόγραμμα
    aCommand = "A";
  }


  // Επιλογή της εντολής AVOID από τον χρήστη
  if ((String)"A" == aCommand)
  {
    aReply = "";
    byte i;
    word chordLength, targetDistance, r, targetAngle;

        // Αρχικοποίηση όλων των πινάκων
    for (i=0; i<2; i++) {
```

```
      turnAngles[i][0] = turnAngles[i][1] = 0;
      positions[i][0]  = positions[i][1]  = 0;
    }
    distancesArray[0][0] = distancesArray[0][1] = 0;

    // Σάρωση μπροστινής περιοχής
    if (!scanVirtualAreaSet) {
      scanArea();
    } else {
      scanVirtualAreaSet = false;
      scanVirtualArea();
    }

    // Δημιουργία πινάκων πιθανοτήτων
    createProbabilityArrays();

    // Χρήση των τρέχουσων συντεταγμένων ως αφετηρία
    positions[0][0] = posX;  positions[0][1] = posY;

    bool fbg = findBestGap(chordLength, targetAngle, r);
    if (fbg)
    {
      //Serial.print(F("chordLength = "));
//Serial.println((String)chordLength);
      //Serial.print(F("targetAngle = "));
//Serial.println((String)targetAngle);
      previous_direction = previous_direction + (targetAngle -
90);

      targetDistance = min_distance + 250; // 250mm μήκος του
ρομπότ
      targetDistance = (FAR_AWAY > targetDistance ?
targetDistance : FAR_AWAY);
      //Serial.print(F("targetDistance = "));
//Serial.println((String)targetDistance);

      if (90 != targetAngle)  targetAngle =
recalcTargetAngle(targetAngle, min_distance);
      //Serial.print(F("min_distance = "));
//Serial.println((String)min_distance);
      //Serial.print(F("targetAngle (new) = "));
//Serial.println((String)targetAngle);

      // Περιστροφή για απόκλιση από την ευθεία κίνησης
      // εφόσον αυτό απαιτείται
      //Serial.println(F("turnByAngle..."));
      playBuzzer("ok"); delay(500);
      bool turned = false;
      if ((89 > targetAngle) || (91 < targetAngle)) {
        turnAngles[0][0] = targetAngle;
        turnAngles[0][1] = turnByAngle(targetAngle, 90);
        turned = true;
      } else {
        targetAngle = turnAngles[0][0] = turnAngles[0][1] =
90;
      }
```

```
      // Ευθύγραμμη κίνηση χωρίς αποκλίσεις
      //Serial.println(F("go..."));
      playBuzzer("ok"); delay(500);
      distancesArray[0][0] = targetDistance;
      go(targetAngle, 100, targetDistance);
      distancesArray[0][1] = distance;
      calcPosXY(targetAngle, distance);
      // positions[1][0] = posX;  positions[1][1] = posY;
      // Διόρθωση μετακίνησης ρομπότ
      // λόγω της τελευταίας περιστροφής
      // Διόρθωση για κίνηση σε πλακάκια
      // if (turned)  positions[1][0] = (positions[0][0] <
posX ? posX + 100: posX - 100);
      // Διόρθωση για κίνηση σε χαλί
      if (turned)  positions[1][0] = (positions[0][0] < posX ?
posX + 50: posX - 50);
      positions[1][1] = posY;

      // Παραλληλοποίηση ως προς τη βασική ευθεία κίνησης
      //Serial.println(F("turnBy90d..."));
      playBuzzer("ok"); delay(500);
      turnAngles[1][0] = 90;
      turnAngles[1][1] = turnByAngle(90, 90);
    }

    // Ολοκλήρωση εντολής
    aReply = "AVOID done!";
  }


  // Επιλογή της εντολής RETURN από τον χρήστη
  else if ((String)"R" == aCommand)
  {
    aReply = "";
    byte i;
    word targetDistance, targetAngleBack;
    // Αρχικοποίηση όλων των πινάκων
    for (i=0; i<2; i++) {
      turnAngles[i][0] = turnAngles[i][1] = 0;
      positions[i][0]  = positions[i][1]  = 0;
    }
    distancesArray[0][0] = distancesArray[0][1] = 0;

    // Χρήση των τρέχουσων συντεταγμένων ως αφετηρία
    positions[0][0] = posX;  positions[0][1] = posY;

    // Στροφή προς τη βασική ευθεία κίνησης
    //Serial.println(F("turnByAngleBack..."));
    playBuzzer("ok"); delay(500);
    targetAngleBack = (0 < posX ? 120 : 60);
    turnAngles[0][0] = targetAngleBack;
    turnAngles[0][1] = turnByAngle(targetAngleBack, 90);
    // Επιστροφή στο σημείο x = 0
    // με ευθύγραμμη κίνηση χωρίς αποκλίσεις
    //Serial.println(F("go RetAngle..."));
```

```
    playBuzzer("ok"); delay(500);

    // Διόρθωση μετακίνησης ρομπότ
    // λόγω της τελευταίας περιστροφής
    // Διόρθωση για κίνηση σε πλακάκια
    // posX = (0 < posX ? posX - 100 : posX + 100);
    // Διόρθωση για κίνηση σε χαλί
    posX = (0 < posX ? posX - 50 : posX + 50);

    targetDistance = calcDistanceToStop(turnAngles[0][1]);
    //Serial.print(F("targetDistance = "));
    //Serial.println((String)targetDistance);
    distancesArray[0][0] = targetDistance;
    go(targetAngleBack, 100, targetDistance);
    distancesArray[0][1] = distance;
    calcPosXY(targetAngleBack, distance);
    positions[1][0] = posX;  positions[1][1] = posY;

    // Παραλληλοποίηση ως προς τη βασική ευθεία κίνησης
    //Serial.println(F("turnBy90d..."));
    playBuzzer("ok"); delay(500);
    turnAngles[1][0] = 90;
    turnAngles[1][1] = turnByAngle(90, 90);

    // Ολοκλήρωση εντολής
    aReply = "RETURN done!";
  }

  delay(LOOPDELAY);
}



/*
 * Διακοπή (interrupt) αριστερού κινητήρα
 */
void Left_ISR()
{
  left_intr++; delay(10);
  // Οδομετρία
  distance = 2*3.141*ROBOT_WHEEL_RADIUS*left_intr/40;
  // Διακοπή κίνησης:
  // εάν έχει οριστεί απόσταση στόχου και
  // εάν η διανυθείσα απόσταση ξεπερνάει την απόσταση στόχου
  if (0 != tgtDist)  if (distance >= tgtDist)  stopMoving();
}



/*
 * Διακοπή (interrupt) δεξιού κινητήρα
 */
void Right_ISR()
{
  right_intr++; delay(10);
}
```

```
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*             || ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ ||                 */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */


/*
 * Υποροutίνα υπολογισμού ακτινίων από μοίρες
 */
float deg2rad(float degrees)
{
  return degrees / 180 * PI;
}



/*
 * Υποροutίνα υπολογισμού μοιρών από ακτίνια
 */
float rad2deg(float radians) {
  return radians * 180 / PI;
}



/*
 * Υποροutίνα αναπαραγωγής ήχων
 */
void playBuzzer(String buzzerStatus)
{
  // Ακίδα βομβητή (buzzer)
  byte buzzerPin = 10;
  pinMode(buzzerPin, OUTPUT);
  if      (String("ok")    == buzzerStatus) tone(buzzerPin,
500);
  else if (String("error") == buzzerStatus) tone(buzzerPin,
3000);
  delay(100);
  noTone(buzzerPin);
  delay(100);
}


/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*                 || ΣΥΝΑΡΤΗΣΕΙΣ VFH ||                   */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
```

```
/*
 * Υπορουτίνα σάρωσης περιοχής
 */
void scanArea()
{
  byte i, max_pos, min_pos;
  //servo.attach(servoPin);
  for(i=0; i<NUM_ANGLES; i++) {
    readNextDistance();
    delay(500);
  }
  //servo.detach();

  max_distance = min_distance = distances[0];
  max_pos = 0;
  for (i=0; i<NUM_ANGLES; i++)
  {
    if (max_distance<distances[i])
    {
      max_distance = distances[i];
      max_pos = i;
    }
    if (min_distance>distances[i])
    {
      min_distance = distances[i];
      min_pos = i;
    }
  }
  //Serial.println((String)"MAX
distance["+sensorAngle[max_pos]+"] = "+max_distance+", MIN
distance["+sensorAngle[min_pos]+"] = "+min_distance);
}



/*
 * Υπορουτίνα μέτρησης απόστασης ανά γωνία εμπρός:
 * Κάθε τιμή απόστασης αποθηκεύεται σε αντίστοιχη θέση πίνακα
 * με βάση την γωνία του σερβο-κινητήρα. Ο κινητήρας
 * κινείται αριστερόστροφα ή δεξιόστροφα ανάλογα με την θέση
 * που βρίσκεται από την τελευταία του φορά κίνησής του. Αφού
 * ληφθεί η τιμή της απόστασης ο σερβο-κινητήρας περιστρέφεται
 * στην επόμενη γωνία του.
 */
void readNextDistance()
{
  distances[angleIndex] = readDistance();
  //Serial.println((String)"reading... distance["+
sensorAngle[angleIndex] + "\xC2\xB0] = " +
distances[angleIndex]/10+"cm");
  if (step == 1 and angleIndex == NUM_ANGLES - 1) {
    step = -1;
  }
  else if (step == -1 and angleIndex == 0) {
```

```
    step = 1;
  }
  else {
    angleIndex += step;
    servo.write(sensorAngle[angleIndex]);
  }
}


/*
 * Υπορουτίνα μέτρησης απόστασης με χρήση του αποστασιόμετρου:
 * Η απόσταση μετριέται σε χιλιοστά (mm).
 * Η ταχύτητα του ήχου στον αέρα στους 20°C είναι 343m/s.
 * Η συνάρτηση pulseIn επιστρέφει την χρονική διάρκεια ενός
(1)
 * παλμού σε μικροδευτερόλεπτα (1μs = 10^-6s).
 * 2d = p * 10^-6s * 343m/s = p * 0.00343m = p * 0.343mm/μs
 */
word readDistance()
{
  playBuzzer("ok");
  // Ο αισθητήρας εκκινείται με παλμό HIGH διάρκειας 10 ή
  // περισσότερων μικροδευτερολέπτων. Παρέχουμε έναν σύντομο
  // παλμό LOW για να αποσταλεί ένας καθαρός παλμός HIGH.
  // Το πλάτος του μέγιστου παλμού δείχνει την απόσταση.
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  unsigned long period = pulseIn(echoPin, HIGH);
  // Επαναφορά της στάθμης του echoPin αν λήφθηκε η τιμή 0
  if (0 == period)
  {
    pinMode(echoPin, OUTPUT);
    delay(10);
    digitalWrite(echoPin, LOW);
    delay(10);
    pinMode(echoPin, INPUT);
    delay(10);
  }
  // Μετατροπή του χρόνου σε απόσταση [mm]
  return period * 343 / 2000;
}



/*
 * Υπορουτίνα εικονικής σάρωσης περιοχής
 */
void scanVirtualArea()
{
  byte i, max_pos, min_pos;
  for(i=0; i<NUM_ANGLES; i++) {
    distances[i] = random(200, 1000);
  }
```

```
  max_distance = min_distance = distances[0];
  max_pos = 0;
  for (i=0; i<NUM_ANGLES; i++)
  {
    if (max_distance<distances[i])
    {
      max_distance = distances[i];
      max_pos = i;
    }
    if (min_distance>distances[i])
    {
      min_distance = distances[i];
      min_pos = i;
    }
  }
  playBuzzer("ok"); playBuzzer("ok");
  //Serial.println((String)"MAX
distance["+sensorAngle[max_pos]+"] = "+max_distance+", MIN
distance["+sensorAngle[min_pos]+"] = "+min_distance);
}



/*
 * Υπορουτίνα δημιουργίας πίνακα πιθανοτήτων σύγκρουσης:
 * Καλείται η υπορουτίνα υπολογισμού πιθανοτήτων σύγκρουσης
 * calcProbability. Το κατώφλι προκύπτει από το άθροισμα όλων
 * των πιθανοτήτων σύγκρουσης προς το πλήθος των γωνιών.
 */
void createProbabilityArrays()
{
  byte i;
  int previous, current, next;
  float sum = 0, threshold;
  // Πίνακας πιθανοτήτων
  float probability[NUM_ANGLES];

  for (i=0; i<NUM_ANGLES; i++)
  {
    // Σε οποιαδήποτη μέτρηση απόστασης (εκτός ακραίων γωνιών)
    // έχουμε υπολογισμένες τιμές απόστασης
    if ((0 != i) and (NUM_ANGLES - 1 != i))
    {
      previous = distances[i-1];
      current  = distances[i];
      next     = distances[i+1];
    }

    // Πριν από την μέτρηση της απόστασης της πρώτης γωνίας
    // δεν υπάρχει τιμή και θέτω ως προηγούμενη τιμή το -1
    else if (0 == i)
    {
      previous = -1;
      current  = distances[i];
      next     = distances[i+1];
```

```
    }

    // Μετά από την μέτρηση της απόστασης της τελευταίας
γωνίας
    // δεν υπάρχει τιμή και θέτω ως επόμενη τιμή το -1
    else if (NUM_ANGLES - 1 == i)
    {
      previous = distances[i-1];
      current  = distances[i];
      next     = -1;
    }
    probability[i] = calcProbability(previous, current, next);
    sum = sum + probability[i];
  }
  //for (i=0; i<NUM_ANGLES; i++)
  //  Serial.println((String)"probability["+sensorAngle[i]+"]
= "+probability[i]);

  threshold = sum / NUM_ANGLES;
  //Serial.println((String)"threshold = "+threshold);

  for (i=0; i<NUM_ANGLES; i++)  probabilityT[i] = (threshold <
probability[i] ? 1 : 0);
}



/*
 * Υπορουτίνα υπολογισμού πιθανοτήτων σύγκρουσης
 * Οι πιθανότητες σύγκρουσης κατηγοριοποιούνται ως εξής:
 * α) 0%: όλες οι διαδοχικές αποστάσεις > FAR_AWAY
 * β) 33%: στις ενδιάμεσες αποστάσεις μία απόσταση < FAR_AWAY
 * γ) 50%: στις αποστάσεις με μία άκρη μία απόσταση < FAR_AWAY
 * δ) 67%: στις ενδιάμεσες αποστάσεις δύο αποστάσεις <
FAR_AWAY
 * ε) 100%: όλες οι διαδοχικές αποστάσεις < FAR_AWAY
 */
float calcProbability(int previous, int current, int next)
{
  float probability;
  // Περίπτωση ακραίων αποστάσεων
  if (-1 == previous or -1 == next)
  {
    if (-1 != previous)
    {
      probability = (FAR_AWAY < previous ? 0.0 : 0.5);
      probability = probability + (FAR_AWAY < current ? 0.0 :
0.5);
    }
    else
    {
      probability = (FAR_AWAY < next ? 0.0 : 0.5);
      probability = probability + (FAR_AWAY < current ? 0.0 :
0.5);
    }
  }
```

```
  // Περίπτωση ενδιάμεσων αποστάσεων
  else
  {
      probability = (FAR_AWAY < previous ? 0.0 : 0.33);
      probability = probability + (FAR_AWAY < current ? 0.0 :
0.33);
      probability = probability + (FAR_AWAY < next ? 0.0 :
0.33);
  }
  // Στρογγυλοποίηση των πιθανοτήτων:
  // από 0.66 σε 0.67 και
  // από 0.99 σε 1
  probability = (0.66 == probability ? 0.67 : probability);
  probability = (0.99 == probability ? 1 : probability);
  return probability;
}


/*
 * Υπορουτίνα εύρεσης του καλύτερου κενού για διέλευση του
ρομπότ
 */
bool findBestGap(word &chordLength, word &angle, word &r)
{
  char i;
  byte middle, leftLimit, rightLimit;
  bool obsFoundLeft = false, obsFoundRight = false;
  word chordLengthMiddle = 0;

  // Α΄ Περίπτωση: Το κενό βρίσκεται μπροστά
  middle = (NUM_ANGLES - 1) / 2;
  leftLimit = middle;
  rightLimit = middle;
  byte c = 1;
  if (0 == probabilityT[middle])
  {
    i = 1;
    while (i <= middle and (!obsFoundLeft or !obsFoundRight))
    {
      if (1 == c)
      {
        if (0 == probabilityT[middle + i])
        {
          leftLimit = middle+i;

          // Εξετάζουμε και το συζυγές ζεύγος ώστε να μην
                // χρειαστεί να περιστραφεί καθόλου το ρομπότ
                // και να περάσει από την μέση
          if ((!obsFoundRight) and (0 == probabilityT[middle -
i]))
          {
            rightLimit = middle - i;
            c++;
          }
          // Υπολογίζουμε κάθε φορά την χορδή ώστε να
```

```
                // γνωρίζουμε αν επαρκεί για την διέλευση του
ρομπότ
          chordLengthMiddle = calcChordLength(leftLimit,
rightLimit, angle, r);
          if (ROBOT_WIDTH < chordLengthMiddle)  break;
        }
        else  obsFoundLeft = true;
      }
      else if (2 == c)
      {
        if (0 == probabilityT[middle-i])
        {
          rightLimit = middle - i;
          // Υπολογίζουμε κάθε φορά την χορδή ώστε να
                // γνωρίζουμε αν επαρκεί για την διέλευση του
ρομπότ
          chordLengthMiddle = calcChordLength(leftLimit,
rightLimit, angle, r);
          if (ROBOT_WIDTH < chordLengthMiddle)  break;
        }
        else  obsFoundRight = true;
      }
      else
      {
        if      (!obsFoundLeft and !obsFoundRight) c = 0;
        else if (obsFoundLeft  and !obsFoundRight) c = 1;
        else if (!obsFoundLeft and obsFoundRight)  c = 0;
        i++;
      }
      c++;
    }
  }

  //Serial.println((String)" chordLengthMiddle=
"+chordLengthMiddle);
  if (ROBOT_WIDTH < chordLengthMiddle)
  {
    //Serial.println((String)" findBestGap1... angle =
"+angle+", r = "+r+", chordLengthMiddle =
"+chordLengthMiddle);
    return true;
  }


  // Β΄ Περίπτωση: Υπάρχει κενό δεξιά ή/και αριστερά ή πουθενά
  else
  {
    word chordLengthLeft, chordLengthRight, angleLeft,
angleRight;
    chordLengthLeft = chordLengthRight = 0;

    // Β1΄: Εύρεση αριστερού κενού
    leftLimit = middle;
    rightLimit = middle;
    for (i=middle+1; i<NUM_ANGLES; i++)
    {
```

```
      // B1α΄: Βρίσκουμε 0
      if (0 == probabilityT[i])
      {
        if (middle == rightLimit)  rightLimit = i;
        else
        {
          leftLimit = i;
          chordLengthLeft = calcChordLength(leftLimit,
rightLimit, angleLeft, r);
          if (ROBOT_WIDTH < chordLengthLeft)  break;
        }
      }

      // B1β΄: Δεν βρίσκουμε 0
      else
      {
        // B1β1΄: Βρήκαμε τα άκρα, εξετάζουμε μήκος χορδής,
              // τερματίζουμε την επανάληψη αν επαρκεί
              // για την διέλευση του ρομπότ
        if ((middle != leftLimit) and (middle != rightLimit))
        {
          chordLengthLeft = calcChordLength(leftLimit,
rightLimit, angleLeft, r);
          if (ROBOT_WIDTH < chordLengthLeft)  break;
          // επαναφέρουμε τις τιμές των άκρων
          else
          {
            leftLimit = middle;
            rightLimit = middle;
          }
        }
        // B1β2΄: Δεν βρήκαμε τα άκρα. Οπότε,
              // επαναφέρουμε τις τιμές τους
        else
        {
          leftLimit = middle;
          rightLimit = middle;
        }
      }
    }
       //Serial.println((String)" chordLengthLeft  =
"+chordLengthLeft);


    byte leftLimit2, rightLimit2;
    // B1΄: Εύρεση δεξιού κενού
    leftLimit2 = middle;
    rightLimit2 = middle;
    for (i=middle-1; i>=0; i--)
    {
      // B2α΄: Βρίσκουμε 0
      if (0 == probabilityT[i])
      {
        if (middle == leftLimit2)  leftLimit2 = i;
        else
        {
```

```
          rightLimit2 = i;
          chordLengthRight = calcChordLength(leftLimit2,
rightLimit2, angleRight, r);
          if (ROBOT_WIDTH < chordLengthRight)  break;
        }
      }

      // B2β΄: Δεν βρίσκουμε 0
      else
      {
        // B1β1΄: Βρήκαμε τα άκρα, εξετάζουμε μήκος χορδής,
        // τερματίζουμε την επανάληψη αν επαρκεί
        // για την διέλευση του ρομπότ
        if ((middle != leftLimit2) and (middle !=
rightLimit2))
        {
          chordLengthRight = calcChordLength(leftLimit2,
rightLimit2, angleRight, r);
          if (ROBOT_WIDTH < chordLengthRight)  break;
          // επαναφέρουμε τις τιμές των άκρων
          else
          {
            leftLimit2 = middle;
            rightLimit2 = middle;
          }
        }
        // B2β2΄: Δεν βρήκαμε τα άκρα. Οπότε,
        // επαναφέρουμε τις τιμές τους
        else
        {
          leftLimit2 = middle;
          rightLimit2 = middle;
        }
      }
    }
       //Serial.println((String)" chordLengthRight =
"+chordLengthRight);


    // ΕΞΕΤΑΣΗ ΚΑΛΥΤΕΡΟΥ ΚΕΝΟΥ
    if ((0 != chordLengthLeft) or (0 != chordLengthRight))
    {
      if ((ROBOT_WIDTH < chordLengthLeft) and (ROBOT_WIDTH <
chordLengthRight))
      {
        // ΚΡΙΤΗΡΙΑ
        char cbg    = chooseBetweenGaps(angleLeft,
angleRight);
        chordLength = ('l' == cbg ? chordLengthLeft :
chordLengthRight);
        angle       = ('l' == cbg ? angleLeft       :
angleRight);
        //Serial.println((String)" findBestGap2... angle  =
"+angle+", r  = "+r);
        return true;
      }
```

```
      else if ((ROBOT_WIDTH >= chordLengthLeft) and
(ROBOT_WIDTH >= chordLengthRight))
      {
        //Serial.println(F(" findBestGap3... false"));
        return false;
      }
      else if ((ROBOT_WIDTH < chordLengthLeft) and
(ROBOT_WIDTH >= chordLengthRight))
      {
        chordLength = chordLengthLeft;
        angle = angleLeft;
        //Serial.println((String)" findBestGap4... angle  =
"+angle+", r  = "+r);
        return true;
      }
      else //if ((ROBOT_WIDTH >= chordLengthLeft) and
(ROBOT_WIDTH < chordLengthRight))
      {
        chordLength = chordLengthRight;
        angle = angleRight;
        //Serial.println((String)" findBestGap5... angle  =
"+angle+", r  = "+r);
        return true;
      }
    }
    else
    {
        //Serial.println(F(" findBestGap6... false"));
        return false;
    }
  }
}



/*
 * Υπορουτίνα υπολογισμού χορδής νοητού κύκλου:
 * Ο κύκλος έχει ως κέντρο το αποστασιόμετρο του ρομπότ.
 */
int calcChordLength(byte leftLimit, byte rightLimit, word
&angle, word &r)
{
  word angleDif;
  // Λαμβάνω υπόψη την μικρότερη απόσταση ως ακτίνα του κύκλου
  r = (distances[leftLimit] < distances[rightLimit] ?
distances[leftLimit] : distances[rightLimit]);
  // Επίκεντρη γωνία κύκλου που αφορά την χορδή
  angleDif = abs(sensorAngle[leftLimit] -
sensorAngle[rightLimit]);
  // Γωνία στην πρέπει να στραφεί το ρομπότ για να προχωρήσει
  // κάθετα ως προς την χορδή του κύκλου
  angle = sensorAngle[rightLimit] + int(angleDif/2);
  //Serial.println((String)" calcChordLength... angleDif  =
"+angleDif+", angle  = "+angle+", r  = "+r);
  return round(2 * r * sin( deg2rad(angleDif/2) ));
}
```

```
/*
 * Υποροutίνα απόφασης G για την καλύτερη διαδρομή:
 * G = a*target_direction
 *   + b*wheel_orientation
 *   + c*previous_direction
 * Η εξίσωση G τροποποιείται ως:
 * G = a*target_direction + c*previous_direction,
 * επειδή wheel_orientation = 0 λόγω 2τροχου μοντέλου κίνησης
 */
char chooseBetweenGaps(int angleLeft, int angleRight)
{
  float G_L, G_R, a = 0.67, c = 0.33;
  G_L = a * (angleLeft - 90)  + c * abs(int(previous_direction
+ angleLeft  - 90));
  G_R = a * (90 - angleRight) + c * abs(int(previous_direction
+ angleRight - 90));
  return (G_L < G_R ? 'l' : 'r');
}



/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*  || ΣΥΝΑΡΤΗΣΕΙΣ ΑΣΥΡΜΑΤΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕ WIFI ||      */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */


/*
 * Υποροutίνα αρχικοποίησης του ESP8266
 */
bool initESP8266()
{
  // 0 => Σημείο πρόσβασης, αλλιώς => ασύρματος σταθμός
  byte CWMODE = 0;

  //Serial.println("Initializing ESP8266...");
  esp8266.begin(serialCommunicationSpeed);
  // Έλεγχος επικοινωνίας με το ESP8266
  while (!esp8266) {
    //Serial.println(F("Waiting for ESP8266 to be
available..."));
    delay(1000);
  }
  // Σύνδεση σε ασύρματο δίκτυο WiFi
  if (initWiFi(CWMODE))  { /*Serial.println(F("initWiFi()...
ok!"));*/ playBuzzer("ok"); playBuzzer("ok"); }
  else                   { /*Serial.println(F("initWiFi()...
failed!"));*/ playBuzzer("error"); }
}
```

```
/*
 * Υπορουτίνα αρχικοποίησης της σύνδεσης του ESP8266
 * σε ασύρματο δίκτυο WiFi
 */
bool initWiFi(byte CWMODE)
{
  // Στοιχεία νέου δικτύου για την ρύθμιση του σημείου
  // πρόσβασης
  String AP_ssid     = "Robot_Car_by_Altintzis_Georgios";
  String AP_password = "12345678";
  // Στοιχεία τοπικού δικτύου για την σύνδεση σε αυτό και
  // την ρύθμιση του ασύρματου σταθμού
  String WSM_ssid     = "WiFi-SSID";
  String WSM_password = "WiFi-PASSWORD";

  // Έλεγχος λειτουργίας του ESP8266
  if (!sendACommand("AT\r\n", "OK", 400, 200))  return false;

  // Επανεκκίνηση του ESP8266
  if (!sendACommand("AT+RST\r\n", "ready", 5000, 200))  return
false;

  // Επιλογή λειτουργίας του ESP8266
  if (0 == CWMODE) {
    /*
     * ESP8266: Σημείο πρόσβασης (ACCESS POINT)
     */
    // Ρύθμιση του ESP8266 ως σημείο πρόσβασης
    // στην διεύθυνση 192.168.4.1
    if (!sendACommand("AT+CWMODE=2\r\n", "OK", 400, 200))
return false;
    // Ορισμός ονόματος και κωδικού ασύρματου δικτύου,
    // καναλιού εκπομπής [από 1 έως 10] και
    // τύπου ασφαλείας [επιλογές: 0 - Ελεύθερο, 2 - WPA_PSK,
    // 3 - WPA2_PSK, 4 - WPA_WPA2_PSK]
    if (!sendACommand("AT+CWSAP=\""+ AP_ssid +"\",\""+
AP_password +"\",8,4\r\n", "OK", 10000, 800))  return false;
  } else {
    /*
     * ESP8266: Ασύρματος σταθμός (WIRELESS STATION MODE)
     */
    // Ρύθμιση του ESP8266 ως ασύρματος σταθμός
    if (!sendACommand("AT+CWMODE=1\r\n", "OK", 400, 200))
return false;
    // Ορισμός ονόματος και κωδικού ασύρματου δικτύου
    if (!sendACommand("AT+CWJAP=\""+ WSM_ssid +"\",\""+
WSM_password +"\"\r\n", "CONNECTED", 10000, 800))  return
false;
  }

  // Λήψη διεύθυνσεων IP και MAC
  if (!sendACommand("AT+CIFSR\r\n", "CIFSR", 500, 200))
return false;
  // Ενεργοποίηση της δυνατότητας επιλογής πολλαπλών συνδέσεων
```

```
  if (!sendACommand("AT+CIPMUX=1\r\n", "OK", 400, 200))
return false;
  // Ενεργοποίηση του server με θύρα επικοινωνίας την 80:
  // η θύρα 80 χρησιμοποιείται για την επικοινωνία
  // με τους διακομιστές web μέσω αιτημάτων http
  if (!sendACommand("AT+CIPSERVER=1,80\r\n", "OK", 400, 200))
return false;

  return true;
}



/*
 * Υποροutίνα αποστολής εντολών στο ESP8266 με
 * δυνατότητα ελέγχου της απάντησης του ESP8266
 */
bool sendACommand(String command, const char* reply, unsigned
int waitA, unsigned int waitB)
{
  // Εκκαθάριση προσωρινής μνήμης
  uint8_t rl = strlen(reply);
  uint8_t idx = 0;
  uint8_t c = 0;
  int i = 0;
  int j = 0;

  // Προεπιλεγμένοι χρόνοι αναμονής
  if (!waitA) waitA = 1000;
  if (!waitB) waitB = 100;

  // Ορισμός κατωφλίου χρόνου αναμονής
  unsigned long timeout = millis() + waitA;

  esp8266.print(command);
  while (millis() < timeout)
  {
    while (esp8266.available())
    {
      // Ανάγνωση ενός χαρακτήρα
      c = esp8266.read();
      i++;
      //Serial.print((char)c);
      /*
      // Break on LF
      if (c == 10) {
        if (idx < rl) idx = 0;
        break;
      }
      */

      // Ταίριασμα του χαρακτήρα
      if ((c == reply[idx]) && (idx < rl)) {
        // Ο χαρακτήρας εντοπίστηκε και τώρα ψάχνουμε τον
        // επόμενο χαρακτήρα
        idx++;
```

```
      } else {
        // Ο χαρακτήρας δεν εντοπίστηκε / επαναφορά της
        // διαδικασίας αντιστοίχισης
        if (idx != rl)  idx = 0;
      }
    }

    if (j>0) {
      if (i>0) {
        // Λήφθηκαν μερικοί χαρακτήρες: ενημέρωση πλήθους,
              // αναμονή περισσότερων και επανάληψη του βρόχου
ξανά
        j += i;
        i = 0;
      } else {
        // Δεν λήφθηκε τίποτα, έξοδος από τον βρόχο
        break;
      }
    }
    delay(waitB);
  }

  //Serial.println();
  //Serial.print(F("_REPORT (")); //Serial.print(reply);
//Serial.print(F(") rl=")); //Serial.print(rl);
  //Serial.print(F(", idx=")); //Serial.print(idx);
  //Serial.println();

  if (idx == rl) {
      // Η απάντηση αντιστοιχήθηκε
      //Serial.println(F("sendACommand: true"));
      return true;
  } else {
    // Η απάντηση δεν αντιστοιχήθηκε
    //Serial.println(F("sendACommand: false"));
    return false;
  }
}



/*
 * Υπορουτίνα αποστολής ιστοσελίδας στο ESP8266
 * με δυνατότητα λήψης εντολής από τη συσκευή-πελάτη
 */
String buildAWebpage(String aReply)
{
  // Έλεγχος μηνύματος από το ESP8266
  if (esp8266.available())
  {
    //Serial.println("esp8266.available!");
    if (esp8266.find("+IPD,"))
    {
      int connectionId, webpage_size;
      String msg, myCommand, webpage, cipSend;
      char reply_to_check[10];
```

```
          // Καθυστέρηση ώστε να γεμίσει το buffer με δεδομένα
        delay(1000);
        // Ανάγνωση του url που στάλθηκε από τη συσκευή-πελάτη
        // Αφαιρώ το 48 επειδή η συνάρτηση read() επιστρέφει
        // την δεκαδική τιμή ASCII και το 0 (ο πρώτος δεκαδικός
        // αριθμός) ξεκινά από το 48
        /*//Serial.println(esp8266.read());*/
        connectionId = esp8266.read() - 48;
        // Εύρεση του κειμένου "?" στην απάντηση του ESP8266
        esp8266.find("?");
        // Καθυστέρηση ώστε να γεμίσει το buffer με δεδομένα
        delay(100);
        msg = esp8266.readStringUntil(' ');
        myCommand = msg.substring(0);

        webpage = "<html><head>";
        if (0 < myCommand.length())
        {
          // Ανανέωση της αρχικής ιστοσελίδας κάθε 10
δευτερόλεπτα
          webpage += "<meta http-equiv=\"refresh\" content=\"10;
url=/\">";
          playBuzzer("ok");
        }
        webpage += "<meta http-equiv='content-type'
content='text/html; charset=windows-1252'>";
        webpage += "<title>Robot Car by Altintzis
Georgios</title>";

        // Αποστολή του 1ου σταθερού τμήματος της ιστοσελίδας
        if (sendAWebPage(connectionId, webpage, 3000))
        {
          delay(100);

          // Αποστολή του τμήματος «style» της ιστοσελίδας
          webpage = "<style>body{background-
color:#021027;color:#fff;font-family:sans-serif}h2{font-
size:1.2em}.button{background-
color:#fa0;border:none;color:#fff;padding:15px;display:inline-
block;text-decoration:none;border-radius:15px;margin:0 0 .4em
.3em}.button:hover{box-shadow:rgba(255,255,255,.25) 0 8px
15px;transform:translateY(-2px)}.button:active{box-
shadow:none;transform:translateY(0)}.centre{text-
align:center}.arduino_says{margin-left:auto;margin-
right:auto}</style>";
          if (sendAWebPage(connectionId, webpage, 4000))
          {
            delay(100);

            if ((0 == myCommand.length()) && ((String("SCAN
done!") == aReply) || (String("AVOID done!") == aReply)))
            {
              webpage = "<style>.chart-wrap .grid::after,.chart-
wrap .grid::before{font-size:.8em;font-
weight:700;position:absolute;top:-1.5em}.chart-wrap{margin-
```

```
bottom:-14em}.title{font-weight:700;font-size:1.2em;padding:0
0 7em;text-align:center;white-space:nowrap}.chart-wrap.vert
.grid{transform:translateY(-175px) translateX(0) rotate(-
90deg)}.chart-wrap.vert .grid .b::after{transform:translateY(-
45%) rotate(35deg);display:block}.chart-wrap.vert
.grid::after,.chart-wrap.vert
.grid::before{transform:translateX(-.2em) rotate(90deg)}";
            if (sendAWebPage(connectionId, webpage, 4000))
            {
              delay(100);
              webpage = ".chart-wrap .grid{margin-
left:auto;margin-right:auto;position:relative;padding:5px
0;width:10em;border-left:2px solid #aaa;background:repeating-
linear-gradient(90deg,transparent,transparent
9.5%,rgba(255,255,255,.7) 10%)}.chart-wrap
.grid::before{content:attr(min);left:-.5em}.chart-wrap
.grid::after{content:attr(max);right:-1.5em}.chart-wrap
.b{width:var(--v);height:14px;margin:3px 0;background-
color:#f16335;border-radius:0 3px 3px 0}.chart-wrap
.b::after{content:attr(d);margin-
left:100%;padding:10px;display:inline-block;white-
space:nowrap}</style>";
              sendAWebPage(connectionId, webpage, 4000);
            }
          }
          delay(100);

          // Αποστολή του τμήματος «body» της ιστοσελίδας
          webpage = "</head><body class='centre'><h1>Robot Car
by Altintzis Georgios</h1><a class='button'
href='?A'>AVOID</a><a class='button' href='?N'>AVOID*</a><a
class='button' href='?R'>RETURN</a>";
          if (sendAWebPage(connectionId, webpage, 4000))
          {

            // Αποστολή του υπόλοιπου τμήματος της ιστοσελίδας
χωρισμένο σε κομμάτια
            if (0 == myCommand.length() && (0 !=
aReply.length()))
            {

              if ((String("AVOID done!") == aReply) ||
(String("SCAN done!") == aReply))
              {
                float dis1, yLimit;
                int i;
                yLimit = FAR_AWAY/1000.0;
                String yLimitS;
                if (int(yLimit) == yLimit)   yLimitS =
String(int(yLimit));
                else                         yLimitS =
String(yLimit, 1);
                for (i=NUM_ANGLES-1; i>=0; i--)
                {
                  dis1 = (FAR_AWAY > distances[i] ?
distances[i] : FAR_AWAY);
```

```
                  dis1 = 100*dis1/FAR_AWAY;
                  if (NUM_ANGLES-1 == i)  webpage =
"<br><br><p>ARDUINO SAYS:</p><div class='chart-wrap vert'><h2
class='title'>Distances Bar Chart</h2><div class='grid'
min='0cm' max='"+ yLimitS +"m'>";
                  else                    webpage = "";
                  webpage += "<div class='b' style='--v:"+
(String)dis1 + "%;' d='" + (String)sensorAngle[i] + char(176)
+ "'></div>";
                  if (0 == i)  webpage += "</div></div>";
                  sendAWebPage(connectionId, webpage, 500);
                }
                delay(100);

                for (i=NUM_ANGLES-1; i>=0; i--)
                {
                  if (NUM_ANGLES-1 == i)  webpage = "<div
class='chart-wrap vert'><h2 class='title'>Collisions
Probability Bar Chart</h2><div class='grid' min='0%'
max='100%'>";
                  else                    webpage = "";
                  webpage += "<div class='b' style='--v:"+
(String)(probabilityT[i]*100) + "%;' d='" +
(String)sensorAngle[i] + char(176) + "'></div>";
                  if (0 == i)  webpage += "</div></div>";
                  sendAWebPage(connectionId, webpage, 500);
                }
              }


              if ((String("AVOID done!") == aReply) ||
(String("RETURN done!") == aReply))
              {
                byte i;
                int posCXprev, posCYprev, posCX, posCY,
yHeight;
                float multY;
                webpage = "";
                if (String("RETURN done!") == aReply)  webpage
+= "<p>ARDUINO SAYS:</p>";
                webpage += "<h2>Robot Route</h2><div>";
                for (i=0; i<2; i++)
                {
                  webpage += "<p>pos" + (String)(i+1) + " (" +
(String)(positions[i][0]/10) + ", " +
(String)(positions[i][1]/10) + "), ";
                  webpage += "turn " +
(String)turnAngles[i][0] + char(176) + " (" +
(String)turnAngles[i][1] + char(176) + ")";
                  if (1 > i) webpage += ", go " +
(String)(distancesArray[i][0]/10) + "cm (" +
(String)(distancesArray[i][1]/10) + "cm)</p>";
                  else       webpage += "</p>";
                }
```

```
                webpage += "</div><canvas id='canv'
width='300' height='300' style='backround-
color:#FFF;'></canvas>";
                sendAWebPage(connectionId, webpage, 500);
                delay(100);

                //multY = (int)(positions[2][1] / 300) + 1;
                multY = (posY-positions[0][1]) / 300.0;
                if (0 == multY)  multY = 1;
                yHeight = 2*50/multY;
                webpage = "<script>var canvas =
document.getElementById('canv');var
c=canvas.getContext('2d');c.fillStyle='#fff';c.fillRect(0,0,30
0,300);c.strokeStyle='#ddd';for(i=0;i<300;i+=10){c.moveTo(i,0)
;c.lineTo(i,300);c.stroke();}for(i=300;i>0;i-=" +
String(yHeight) +
"){c.moveTo(0,i);c.lineTo(300,i);c.stroke();}";
                sendAWebPage(connectionId, webpage, 1000);
                delay(100);

                webpage =
"c.beginPath();c.strokeStyle='#0f0';c.lineWidth=4;c.moveTo(150
,300);c.lineTo(150,0);c.moveTo(0,299);c.lineTo(300,299);c.stro
ke();";
                sendAWebPage(connectionId, webpage, 1000);
                delay(100);

                webpage =
"c.beginPath();c.strokeStyle='#fa0';c.lineWidth=7;c.moveTo(50,
225);c.lineTo(100,225);c.stroke();c.moveTo(200,225);c.lineTo(2
50,225);c.stroke();c.moveTo(225,200);c.lineTo(225,250);c.strok
e();";
                sendAWebPage(connectionId, webpage, 1000);
                delay(100);

                webpage = "c.fillStyle='#000';c.font='26px
Bold
Courier';c.fillText('Left',53,270);c.fillText('Right',196,270)
;";
                webpage +=
"c.beginPath();c.strokeStyle='#f00';c.lineWidth=5;c.font='18px
Bold Courier';";
                posCXprev = 150 + positions[0][0]/10;
                posCYprev = 300 - positions[0][1]/10;
                webpage += "c.fillText('pos1',"+
(String)posCXprev +","+ (String)(posCYprev-2) +");";
                sendAWebPage(connectionId, webpage, 1000);
                delay(100);

                webpage = "";
                for (i=1; i<2; i++)
                {
                  posCX = 150 + positions[i][0]/10;
                  if (1 > i)  posCY = 300 -
positions[i][1]/multY;
                  else        posCY = 0;
```

```
                    webpage += "c.moveTo(" + (String)posCXprev +
", " + (String)(posCYprev+5) + ");c.lineTo(" + (String)posCX +
", " + (String)posCY + ");";
                    posCXprev = posCX;
                    posCYprev = posCY;
                    if (1 > i)  webpage += "c.fillText('pos" +
(String)(i+1) + "'," + (String)(posCXprev+5) + "," +
(String)(posCYprev-2) + ");";
                    else        webpage += "c.fillText('pos" +
(String)(i+1) + "'," + (String)(posCXprev+5) + "," +
(String)(posCYprev+12) + ");";
                  }
                  webpage += "c.stroke();</script>";
                  sendAWebPage(connectionId, webpage, 3000);
                  delay(100);
                }

                webpage = "<br><br></body></html>";
                sendAWebPage(connectionId, webpage, 500);
              }
            }
          }
        }


        // Ολοκλήρωση σύνδεσης με τη συσκευή-πελάτη
        String closeCommand = "AT+CIPCLOSE=" +
String(connectionId) + "\r\n";
        sendACommand(closeCommand, "CLOSED", 2000, 300);

        //Serial.println((String)"+ myCommand = ["+myCommand+"],
aReply = ["+aReply+"]");
        return myCommand;
      }
      return "";
    }
    return "";
}



/*
 * Υπορουτίνα αποστολής ιστοσελίδας στο ESP8266
 */
bool sendAWebPage(int connectionId, String webpage, word
timeDelay)
{
    int webpage_size;
    String cipSend;
    char reply_to_check[10];

    webpage_size = webpage.length();
    cipSend = "AT+CIPSEND=" + String(connectionId) + "," +
String(webpage_size) + "\r\n";
    if (sendACommand(cipSend, "OK", 300, 200))
    {
```

```
      itoa(webpage_size, reply_to_check, 10);
      strcat(reply_to_check, " bytes");
      if (sendACommand(webpage, reply_to_check, timeDelay,
300))  return true;
      else  return false;
    }
    else  return false;
}



/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*               || ΣΥΝΑΡΤΗΣΕΙΣ ΘΕΣΗΣ ||                  */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */


/*
 * Υποροutίνα επαναϋπολογισμού γωνίας περιστροφής
 * λόγω ασσυμετρίας κατασκευής:
 * Υπολογίζουμε ξανά την γωνία περιστροφής επειδή το
 * αποστασιόμετρο απέχει 130mm από το μαγνητόμετρο.
 */
word recalcTargetAngle(int targetAngle, word r)
{
  float angleDifRad;
  word x, y, newAngleDif;
  angleDifRad = deg2rad( abs(targetAngle - 90));
  y = round( r * sin( angleDifRad));
  x = round( sqrt(pow(y,2) + pow(r,2)));
  //return 90-round( rad2deg( atan2(y,x+130) ) );
  newAngleDif = round( rad2deg( atan2(y,x+130)));
  //Serial.println((String)" + angleDif = "+abs(targetAngle -
90)+", y = "+y+", x = "+x+", newAngleDif = "+newAngleDif);
  return (90 > targetAngle ? 90-newAngleDif : 90+newAngleDif);
}


/*
 * Υποροutίνα υπολογισμού συντεταγμένων:
 * Υπολογίζουμε την θέση του ρομπότ χρησιμοποιώντας
 * τη γωνία περιστροφής και την διανυόμενη απόσταση.
 */
void calcPosXY(int targetAngle, word distance)
{
  //byte angleDif;
  int posX_new, posY_new;
  float angleDifRad;
  angleDifRad = deg2rad( abs(targetAngle - 90) );
  posX_new = int(distance * sin(angleDifRad));
  posY_new = int(distance * cos(angleDifRad));
  posX = (90 > targetAngle ? posX + posX_new : posX -
posX_new);
```

```
  posY = posY + posY_new;
  //Serial.println((String)" + posX = "+posX+", posY =
"+posY);
}



/*
 * Υπορουτίνα υπολογισμού υποτείνουσας κίνησης στην θέση (0,y)
 */
word calcDistanceToStop(int curAngle)
{
  float angleDifRad;
  angleDifRad = deg2rad( abs(curAngle - 90) );
  return abs(posX) / sin(angleDifRad);
}



/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*              || ΣΥΝΑΡΤΗΣΕΙΣ ΚΙΝΗΣΗΣ ||                  */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */


/*
 * Υπορουτίνα περιστροφής:
 * Περιστρέφουμε το ρομπότ προς μία πλευρά περιστρέφοντας
 * έναν κινητήρα την φορά. Αν το ρομπότ φτάσει στην γωνία
 * endAngle τότε η περιστροφή ολοκληρώνεται.
 */
int turnByAngle(int endAngle, byte speed)
{
  byte speedLowLimit;
  int curAngle;
  speedLowLimit = 90;
  // Λήψη γωνίας κίνησης
  curAngle = getFixedAzimuth();
  //Serial.println((String)+" START curAngle = "+curAngle+",
endAngle = "+endAngle);

  while (endAngle != curAngle)
  {
    speed = (speed-1 > speedLowLimit ? speed-1 : speed);
    if (endAngle > curAngle)  moveForward(speed, 0);
    else                      moveForward(0, speed);
    startMoving();
        // Λήψη γωνίας κίνησης
    curAngle = getFixedAzimuth();
        // Σταμάτημα των κινητήρων
        if (endAngle == curAngle)  stopMoving();
  }
  delay(50);
```

```
  // Λήψη γωνίας κίνησης
  curAngle = getFixedAzimuth();

  //Serial.println((String)" !!! curAngle = "+curAngle+",
endAngle = "+endAngle);
  return curAngle;
}



/*
 * Υπορουτίνα σταματήματος:
 * Διακόπτουμε την κίνηση των κινητήρων.
 */
void stopMoving()
{
  // Σταμάτημα των κινητήρων
  digitalWrite(in1Pin, LOW);
  digitalWrite(in2Pin, LOW);
  digitalWrite(in3Pin, LOW);
  digitalWrite(in4Pin, LOW);
}



/*
 * Υπορουτίνα εκκίνησης κινητήρων:
 * Ενεργοποιούμε την κίνηση των κινητήρων.
 */
void startMoving()
{
  // Εκκίνηση του δεξιού κινητήρα
  digitalWrite(in1Pin, HIGH);
  digitalWrite(in2Pin, LOW);
  // Εκκίνηση του αριστερού κινητήρα
  digitalWrite(in3Pin, HIGH);
  digitalWrite(in4Pin, LOW);
}



/*
 * Υπορουτίνα κίνησης προς τα μπροστά:
 * Το ρομπότ κινείται εμπρός.
 */
void moveForward(byte speedRight, byte speedLeft)
{
  // Ορισμός ταχύτητας δεξιού κινητήρα
  analogWrite(enRightPin, speedRight);
  // Ορισμός ταχύτητας αριστερού κινητήρα
  analogWrite(enLeftPin, speedLeft);
}
```

```
/*
 * Υποροutίνα κίνησης προς συγκεκριμένη κατεύθυνση:
 * dirAngle: Γωνία κατεύθυνσης προορισμού
 * curAngle: Μεταβαλλόμενη γωνία κατεύθυνσης κίνησης
 * angleDif: Διαφορά της γωνίας κίνησης με τη γωνία προορισμού
 * speedInit: Αρχική ταχύτητα εκκίνησης
 * speedLeft: Ταχύτητα αριστερού τροχού
 * speedRight: Ταχύτητα δεξιού τροχού
 * speedLowLimit: Κατώτατο όριο ταχύτητας
 * speedHighLimit: Ανώτατο όριο ταχύτητας
 * safeSpeedDif: Ποσότητα μείωσης ταχύτητας όταν η ταχύτητα
 * και των δύο τροχών έχει υπερβεί την ταχύτητα εκκίνησης
 * speedCor: Ποσότητα διόρθωσης ταχύτητας όταν παρατηρείται
 * απόκλιση μεταξύ κατεύθυνσης κίνησης και προορισμού
 * speedCorNum: Ποσότητα διόρθωσης
 */
void go(byte dirAngle, byte speedInit, word targetDistance)
{
  int curAngle, angleDif;
  byte speedLeft, speedRight, speedLowLimit, speedHighLimit,
safeSpeedDif, speedCor, speedCorNum;
  speedLowLimit = speedInit-15;
  speedHighLimit = speedInit+15;
  safeSpeedDif = 5;
  speedCorNum = 7;
  speedLeft = speedRight = speedInit;

  curAngle = getFixedAzimuth();
  //Serial.println((String)" -> curAngle = "+curAngle);
  //Serial.println((String)" -> targetDistance =
"+targetDistance+" ? distance = "+distance);

  left_intr = right_intr = distance = 0;
  // Διόρθωση με αφαίρεση της ακτίνας του τροχού του ρομπότ
  targetDistance -= ROBOT_WHEEL_RADIUS;
  tgtDist = targetDistance;
  servo.write(90); delay(1000);

  // Προχωρούμε μόνο αν δεν υπάρχει εμπόδιο εμπρός
  // τουλάχιστον σε απόσταση targetDistance
  if (targetDistance < readDistance())
  {
    playBuzzer("ok"); delay(500);
    moveForward(speedInit, speedInit);
    startMoving();

    while (targetDistance > distance)
    {
      angleDif = abs(curAngle-dirAngle);
      //Serial.println((String)" -> curdistance =
"+distance+"...");

        // Διακοπή κίνησης αν το ρομπότ κινηθεί πολύ στραβά
      if (15 <= angleDif)
      {
        playBuzzer("error");
```

```
        break;
      }
         // Αλλιώς γίνεται προσπάθεια διόρθωσης της κίνησης
      else if (1 < angleDif)
      {
        //Serial.println(F(" * 1 < angleDif"));
        speedCor = speedCorNum*angleDif;
        if (dirAngle > curAngle)
        {
          // Αν το ρομπότ κινείται δεξιά
          playBuzzer("error");
          //Serial.println((String)" * "+dirAngle+" <
"+curAngle+": L--: ");
          if (speedLeft > speedInit)  speedLeft  = (speedLeft-
speedCor  > speedLowLimit  ? speedLeft-speedCor  :
speedLowLimit);
          else                        speedRight =
(speedRight+speedCor < speedHighLimit ? speedRight+speedCor :
speedHighLimit);
        }
        else if (dirAngle < curAngle)
        {
          // Αν το ρομπότ κινείται αριστερά
          playBuzzer("ok");
          //Serial.println((String)" * "+dirAngle+" <
"+curAngle+": L++: ");
          if (speedLeft < speedInit)  speedLeft  =
(speedLeft+speedCor  < speedHighLimit ? speedLeft+speedCor :
speedHighLimit);
          else                        speedRight =
(speedRight-speedCor > speedLowLimit  ? speedRight-speedCor:
speedLowLimit);
        }

        if (speedLeft > speedInit+safeSpeedDif and speedRight
> speedInit+safeSpeedDif)
          {speedLeft=speedLeft-safeSpeedDif;
speedRight=speedRight-safeSpeedDif;}
        else if (speedLeft < speedInit-safeSpeedDif and
speedRight < speedInit-safeSpeedDif)
          {speedLeft=speedLeft+safeSpeedDif;
speedRight=speedRight+safeSpeedDif;}
      }
      else if (0 == angleDif)
      {
        speedLeft = speedRight = speedInit;
      }

      // Κίνηση εμπρός
      moveForward(speedRight, speedLeft);

      // Λήψη γωνίας κίνησης
      curAngle = getFixedAzimuth();
    }
  }
  else
```

```
  {
    playBuzzer("error");
  }

  //Serial.println((String)"_ STOP @ distance = "+distance);
  // Σταμάτημα των κινητήρων
  tgtDist = 0;
  stopMoving();

  // Διόρθωση με πρόσθεση της ακτίνας του τροχού του ρομπότ
  if (targetDistance <= distance)  distance +=
ROBOT_WHEEL_RADIUS;
}



/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*                                                          */
/*         || ΣΥΝΑΡΤΗΣΕΙΣ ΜΑΓΝΗΤΙΚΟΥ ΠΕΔΙΟΥ ||         */
/*                                                          */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/* xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx */


/*
 * Υπορουτίνα αρχικοποίησης μαγνητόμετρου
 */
void initCompass()
{
  //Serial.println(F("Initializing QMC5883L..."));
  compass.init();
  // MODE CONTROL (MODE): Continuous (0x01)
  // OUTPUT DATA RATE (ODR): 200Hz (0x0C)
  // FULL SCALE (RNG): 8G (0x10)
  // The lowest field range has the highest sensitivity,
  // therefore, higher resolution.
  // OVER SAMPLE RATIO (OSR): 512 (0x00)
  // Larger OSR value leads to smaller filter bandwidth,
  // less in-band noise and higher power consumption.
  compass.setMode(0x01, 0x0C, 0x10, 0x00);
  compass.setSmoothing(1, true);
  compass.setCalibration(-881, 1723, -228, 2610, 0, 3718);
  delay(500);

  azimuthFix = calcAzimuthFix();
  delay(100);
  while (90 != getFixedAzimuth())
  {
    azimuthFix = calcAzimuthFix();
    delay(100);
    //Serial.println(F("correcting..."));
  }
  //Serial.print(F("-> azimuthFix = "));
//Serial.println(azimuthFix);
}
```

```
/*
 * Υποροutίνα λήψης απόκλισης σε μοίρες από τον Βορρά
 */
int getAzimuth() {
  // Λήψη τιμών μαγνητόμετρου
  compass.read();
  // Λήψη τιμής αζιμουθίου
  int az = compass.getAzimuth();
  // Αντιστροφή αζιμουθίου
  return (180 >= az ? 180 - az : 540 - az);
}



/*
 * Υποροutίνα υπολογισμού απόκλισης μαγνητικού πεδίου
 * σε σχέση με τον προσανατολισμό του ρομπότ ώστε το
 * ρομπότ να ξεκινάει πάντα σε γωνία 90° από τον Βορρά
 */
int calcAzimuthFix()
{
  return 90 - getAzimuth();
}



/*
 * Υποροutίνα διόρθωσης τιμής μαγνητικού πεδίου
 */
int getFixedAzimuth()
{
  int raz = getAzimuth();
  int fraz = (360 >= raz+azimuthFix ? raz+azimuthFix :
azimuthFix-abs(360-raz));
  fraz = (0 <= fraz ? fraz : 360-abs(fraz));
  return fraz;
}
```

ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα παρέχω χρήσιμο υλικό που λειτουργεί συμπληρωματικά στην εργασία μου καθώς και προτάσεις για μελλοντική εξέλιξη.

Βιβλιοθήκη SoftwareSerial

Η βιβλιοθήκη **SoftwareSerial** επιτρέπει τη σειριακή επικοινωνία μέσω των ψηφιακών ακίδων μιας πλακέτας Arduino, χρησιμοποιώντας λογισμικό για την αναπαραγωγή αυτής της λειτουργικότητας.

Αρχικά πρέπει να την εισάγουμε στον κώδικά μας με την εντολή «**`#include <SoftwareSerial.h>`**» και την χρησιμοποιούμε με την εντολή «**`SoftwareSerial (rxPin, txPin, inverse_logic)`**» όπου η μεταβλητή **rxPin** είναι η εικονική ακίδα RX του Arduino, η **txPin** είναι η εικονική ακίδα TX του Arduino και η προαιρετική **inverse_logic** είναι μια τιμή που προκαλεί αντιστροφή των εικονικών ακίδων με προεπιλογή την τιμή false (0). Αυτή η βιβλιοθήκη φαίνεται να λειτουργεί απρόσκοπτα σε ταχύτητες μικρότερες ή ίσες των **9600**. Για να θέσουμε την ταχύτητα του module στις **9600** εκτελούμε στο περιβάλλον του module την AT εντολή «**`AT+UART_DEF=9600,8,1,0,3`**».

Η βιβλιοθήκη **SoftwareSerial** δεν είναι αναγκαία στην πλακέτα **Arduino Uno Rev3** επειδή διαθέτει μόνο ένα (1) ζεύγος και το δεσμεύουμε με την σειριακή θύρα USB. Συνεπώς δημιουργούμε 1 εικονικό ζεύγος ακίδων RX – TX.

Παράδειγμα κώδικα για χρήση της βιβλιοθήκης **SoftwareSerial** στο **Arduino Uno Rev3**:

```
// Φόρτωση βιβλιοθήκης SoftwareSerial
#include <SoftwareSerial.h>

// Arduino Uno Rev3 (ακίδες 2, 3 ως RX, TX) <-> ESP8266 TX, RX
#SoftwareSerial esp8266(2, 3);
```

Η βιβλιοθήκη **SoftwareSerial** δεν είναι απαραίτητη στην πλακέτα της κατασκευής μου, την **Arduino Mega 2560 Rev3**, επειδή ήδη μας παρέχει τρία (3) ζεύγη ακίδων RX - TX από τα οποία δεσμεύουμε το ένα (1) ζεύγος με την σειριακή θύρα USB. Όπως φαίνεται στην γραμμή του κώδικά μου εγώ κάνω χρήση του ζεύγους ακίδων RX - TX με όνομα `Serial2`:

```
// Arduino Mega 2560 Rev3 Serial2 (RX2, TX2) <-> ESP8266 TX, RX
#define esp8266 Serial2
```

Προτάσεις για μελλοντική επέκταση

Είναι γεγονός ότι η αντικατάσταση των τρεχόντων αισθητήρων με άλλους μεγαλύτερης ακρίβειας θα συντελέσει στην καλύτερη απεικόνιση της μεθόδου VFH όπως για παράδειγμα η αντικατάσταση του αποστασιόμετρου με ένα καλύτερο και ακριβότερο μοντέλο. Ωστόσο, αυτή η ενέργεια θα ανεβάσει αρκετά το κόστος. Τέλος, η ανάπτυξη μίας εφαρμογής για Η/Υ ή/και για κινητό που θα επικοινωνεί απευθείας με το ρομπότ μέσω sockets θα οδηγούσε σε οικονομία δεδομένων μετάδοσης και σε καλύτερη επεξεργασία αυτών των δεδομένων.

ΒΙΒΛΙΟΓΡΑΦΙΑ – ΔΙΚΤΥΟΓΡΑΦΙΑ

Ελληνόγλωσση

- Cableworks. Λήμμα «2WD Smart Robot Car Chassis Kit for Arduino». Διαθέσιμο στον δικτυακό τόπο: https://www.cableworks.gr/ilektronika/kit/robot-car-kit/2wd-smart-robot-car-chassis-kit-for-arduino/. [Πρόσβαση στις 1/08/2021].
- GRobotronics. Λήμμα «ESP-01 Adapter Module 3.3-5V». Διαθέσιμο στον δικτυακό τόπο: https://grobotronics.com/esp-01-adapter-module-3.3-5v.html. [Πρόσβαση στις 27/09/2022].
- GRobotronics. Λήμμα «Sensor Shield V5.0 for Arduino». Διαθέσιμο στον δικτυακό τόπο: https://grobotronics.com/sensor-shield-v5.0-for-arduino.html. [Πρόσβαση στις 12/10/2022].
- KalemisBros. Λήμμα «KBS 80W iMAX B6AC Dual Power LiPo Ni-Cd NiMH RC Battery Balance Charger Discharger». Διαθέσιμο στον δικτυακό τόπο: https://www.kalemisbros.gr/sports-and-outdoors/drones-and-rc/spare-and-accessories/kbs-80w-imax-b6ac-dual-power-lipo-ni-cd-nimh-rc-battery-balance-charger-discharger-r/. [Πρόσβαση στις 28/08/2021].
- pantou.mysch.gr. Λήμμα «Πώς να χρησιμοποιήσετε το κύκλωμα οδήγησης κινητήρα L298N». Διαθέσιμο στον δικτυακό τόπο: http://www.pantou.mysch.gr/kyklvmata/L298N.htm. [Πρόσβαση στις 11/08/2021].
- Technokap. Λήμμα «Gens ace 1600mAh 7.4V 45C 2S1P Lipo Battery Pack with XT60 Plug». Διαθέσιμο στον δικτυακό τόπο: https://technokap.gr/?product=gens-ace-1600mah-7-4v-45c-2s1p-lipo-battery-pack-with-xt60-plug. [Πρόσβαση στις 28/08/2021].
- Top Electronics. Λήμμα «Charger iMax B6AC 80W w/ power supply + adapters». Διαθέσιμο στον δικτυακό τόπο: https://topelectronics.gr/new-products/imax-chrager-imax-b6ac-80w-w-power-supply-adapters/. [Πρόσβαση στις 02/08/2021].
- Top Electronics. Λήμμα «ESP8266 ESP-01 Adaptor Module 3.3V/5V». Διαθέσιμο στον δικτυακό τόπο: https://topelectronics.gr/arduino/esp8266/esp8266-esp-01-adaptor-module-3.3v-5v/. [Πρόσβαση στις 25/09/2022].
- Why.gr - Διερευνητική Μάθηση. Λήμμα «Αισθητήρας Απόστασης Υπερήχων 2 – 400cm HC-SR04». Διαθέσιμο στον δικτυακό τόπο: https://www.why.gr/καταστημα/open-hardware/αισθητήρες/αισθητήρας-απόστασης-υπερήχων-hc-sr04/. [Πρόσβαση στις 01/02/2022].
- Βικιπαίδεια. Λήμμα «Arduino». Διαθέσιμο στον δικτυακό τόπο: https://el.wikipedia.org/wiki/Arduino. Τελευταία ενημέρωση στις 28 Σεπτεμβρίου 2022. [Πρόσβαση στις 11/11/2022].
- Βικιπαίδεια. Λήμμα «Μικροελεγκτής». Διαθέσιμο στον δικτυακό τόπο: https://el.wikipedia.org/wiki/Μικροελεγκτής. Τελευταία ενημέρωση στις 21 Δεκεμβρίου 2020. [Πρόσβαση στις 01/09/2022].
- Ηλεκτρονικά Dme. Λήμμα «DC οδηγός βηματικού κινητήρα L298N». Διαθέσιμο στον δικτυακό τόπο: https://dme.gr/product/dc-odigos-vimatikou-kinitira-l298n. [Πρόσβαση στις 11/08/2021].
- Σπύρος Καζαρλής. Σημειώσεις μαθήματος ΠΜΣ «Αυτόνομα Ρομποτικά Οχήματα», Μάθημα 3ο: Σχεδιασμός Διαδρομών - Path Planning, Διεθνές Πανεπιστήμιο της

Ελλάδος (Σέρρες), Τμήμα Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών.

- Σπύρος Καζαρλής. Σημειώσεις μαθήματος ΠΜΣ «Ενσωματωμένα Συστήματα», Παρουσίαση και εργαστήριο 4 και 5, Διεθνές Πανεπιστήμιο της Ελλάδος (Σέρρες), Τμήμα Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών.

Ξενόγλωσση

- AliExpress. Λήμμα «Ir Infrared Slotted Optical Speed Measuring Sensor Detection Optocoupler Module For Motor Test». Διαθέσιμο στον δικτυακό τόπο: https://www.aliexpress.com/i/32805007505.html. [Πρόσβαση στις 06/03/2022].
- Amazon.com. Λήμμα «diymore 2WD Smart Robot Car Chassis Kit with 2 Motor (1:48) Speed Encoder Battery Box». Διαθέσιμο στον δικτυακό τόπο: https://www.amazon.com/diymore-Smart-Robot-Chassis-Motor/dp/B01LWYUQPH. [Πρόσβαση στις 28/08/2021].
- Amazon.co.uk. Λήμμα «Youmile 5Pcs Speed Measuring Sensor IR Infrared Slotted Optical Optocoupler Module Photo Interrupter Sensor for Motor Speed Detection or Arduino with Encoders». Διαθέσιμο στον δικτυακό τόπο: https://www.amazon.co.uk/Youmile-Measuring-Optocoupler-Interrupter-Detection/dp/B0817FM4BJ. [Πρόσβαση στις 06/03/2022].
- Andromina robot V.2.0. Λήμμα «Encoder and Arduino. Tutorial about the IR speed sensor module with the comparator LM393 (Encoder FC-03)». Διαθέσιμο στον δικτυακό τόπο: https://androminarobot-english.blogspot.com/2017/03/encoder-and-arduinotutorial-about-ir.html. Τελευταία ενημέρωση στις 2 Μαρτίου 2017. [Πρόσβαση στις 05/03/2022].
- Arduino. Λήμμα «Analogwrite». Διαθέσιμο στον δικτυακό τόπο: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/. Τελευταία ενημέρωση στις 7 Σεπτεμβρίου 2020. [Πρόσβαση στις 10/02/2022].
- Arduino Docs. Λήμμα «SoftwareSerial Library». Διαθέσιμο στον δικτυακό τόπο: https://docs.arduino.cc/learn/built-in-libraries/software-serial. Τελευταία ενημέρωση στις 19 Ιανουαρίου 2023. [Πρόσβαση στις 19/01/2023].
- Arduino Official Store. Λήμμα «Arduino Mega 2560 Rev3». Διαθέσιμο στον δικτυακό τόπο: https://store.arduino.cc/products/arduino-mega-2560-rev3. [Πρόσβαση στις 10/01/2023].
- Arduino Official Store. Λήμμα «Arduino Uno Rev3». Διαθέσιμο στον δικτυακό τόπο: https://store.arduino.cc/products/arduino-uno-rev3. [Πρόσβαση στις 10/01/2023].
- Arduino Official Store. Λήμμα «Boards». Διαθέσιμο στον δικτυακό τόπο: https://store.arduino.cc/collections/boards. [Πρόσβαση στις 10/01/2023].
- Arduino Playground. Λήμμα «Attachinterrupt». Διαθέσιμο στον δικτυακό τόπο: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/. Τελευταία ενημέρωση στις 21 Μαΐου 2021. [Πρόσβαση στις 07/03/2022].
- Arduino Playground. Λήμμα «I2cScanner». Διαθέσιμο στον δικτυακό τόπο: https://playground.arduino.cc/Main/I2cScanner/. Τελευταία ενημέρωση στις 14 Νοεμβρίου 2018. [Πρόσβαση στις 08/10/2022].
- ARSSHIELD - Datasheet. Λήμμα «Sensor Shield V5.0». Διαθέσιμο στον δικτυακό τόπο: https://www.diyelectronics.co.za/store/index.php?controller=attachment&id_attachment=558. [Πρόσβαση στις 15/02/2022].

- Circuit Basics. Λήμμα «How to Use Active and Passive Buzzers on the Arduino». Διαθέσιμο στον δικτυακό τόπο: https://www.circuitbasics.com/how-to-use-active-and-passive-buzzers-on-the-arduino/. [Πρόσβαση στις 15/06/2022].
- Electric Diy Lab. Λήμμα «How to use GY-271 Magnetic field sensor with Arduino». Διαθέσιμο στον δικτυακό τόπο: https://electricdiylab.com/how-to-use-gy-271-magnetic-field-sensor-with-arduino/. Τελευταία ενημέρωση τον Μάιο 2021. [Πρόσβαση στις 02/03/2022].
- Electronic Clinic. Λήμμα «LM393 Speed sensor with Arduino using L9110 motor driver, circuit and Code explained». Διαθέσιμο στον δικτυακό τόπο: https://www.electroniclinic.com/lm393-speed-sensor-with-arduino-using-l9110-motor-driver-circuit-and-code-explained/. Τελευταία ενημέρωση στις 21 Μαΐου 2021. [Πρόσβαση στις 05/03/2022].
- Electronics Hub. Λήμμα «How to Interface LM393 Speed Sensor with Arduino?». Διαθέσιμο στον δικτυακό τόπο: https://www.electronicshub.org/interfacing-lm393-speed-sensor-with-arduino/. Τελευταία ενημέρωση στις 7 Σεπτεμβρίου 2018. [Πρόσβαση στις 06/03/2022].
- Electronics Hub. Λήμμα «How to Update Flash ESP8266 Firmware – Flashing Official AT Firmware». Διαθέσιμο στον δικτυακό τόπο: https://www.electronicshub.org/update-flash-esp8266-firmware/. Τελευταία ενημέρωση στις 16 Δεκεμβρίου 2017. [Πρόσβαση στις 01/10/2022].
- ElectroPeak. Λήμμα «Interfacing LM393 Infrared Speed Sensor with Arduino». Διαθέσιμο στον δικτυακό τόπο: https://electropeak.com/learn/interfacing-lm393-infrared-speed-sensor-with-arduino/. [Πρόσβαση στις 08/03/2022].
- Embedded Laboratory. Λήμμα «Update the AT Firmware in Your ESP8266 WiFi Module». Διαθέσιμο στον δικτυακό τόπο: https://embeddedlaboratory.blogspot.com/2021/02/update-at-firmware-in-your-esp8266-wifi.html. Τελευταία ενημέρωση στις 6 Φεβρουαρίου 2021. [Πρόσβαση στις 02/10/2022].
- GitHub. Λήμμα «ESP8266 - Wiki Public». Διαθέσιμο στον δικτυακό τόπο: https://github.com/esp8266/esp8266-wiki/wiki. Τελευταία ενημέρωση στις 6 Μαρτίου 2015. [Πρόσβαση στις 12/10/2022].
- GitHub. Λήμμα «HMC5883L Triple Axis Digital Compass Arduino Library». Διαθέσιμο στον δικτυακό τόπο: https://github.com/jarzebski/Arduino-HMC5883L. Τελευταία ενημέρωση στις 31 Ιουλίου 2017. [Πρόσβαση στις 09/03/2022].
- GitHub. Λήμμα «QMC5883L Compass is a Arduino library for using QMC5583L series chip boards as a compass». Διαθέσιμο στον δικτυακό τόπο: https://github.com/mprograms/QMC5883LCompass. Τελευταία ενημέρωση στις 7 Ιουλίου 2020. [Πρόσβαση στις 09/03/2022].
- Instructables. Λήμμα «Restore or Upgrade Firmware on ESP8266 (ESP-01) Module Using Arduino MKR». Διαθέσιμο στον δικτυακό τόπο: https://www.instructables.com/Restore-or-Upgrade-Firmware-on-ESP8266-ESP-01-Modu/. [Πρόσβαση στις 01/10/2022].
- Last Minute Engineers. Λήμμα «How HC-SR04 Ultrasonic Sensor Works & Interface It With Arduino». Διαθέσιμο στον δικτυακό τόπο: https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/. [Πρόσβαση στις 05/02/2022].

- Last Minute Engineers. Λήμμα «Interface L298N DC Motor Driver Module with Arduino». Διαθέσιμο στον δικτυακό τόπο: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/. [Πρόσβαση στις 05/02/2022].
- MagicDuino. Λήμμα «MG996R All Metal Gear Servo Motor». Διαθέσιμο στον δικτυακό τόπο: http://magicduino.com/Images/ItemsMedia/File/7203.pdf. [Πρόσβαση στις 05/04/2022].
- Pololu. Λήμμα «iMAX B6AC manual». Διαθέσιμο στον δικτυακό τόπο: https://www.pololu.com/file/0J525/iMAXB6ACmanual.pdf. [Πρόσβαση στις 02/08/2021].
- Rajguru Electronics. Λήμμα «Lm393 Motor Speed Measuring Sensor Module For Arduino». Διαθέσιμο στον δικτυακό τόπο: https://electropeak.com/learn/download/lm393-infrared-speed-sensor-datasheet/. [Πρόσβαση στις 08/03/2022].
- RemoteXY. Λήμμα «ESP8266 firmware update». Διαθέσιμο στον δικτυακό τόπο: https://remotexy.com/en/help/esp8266-firmware-update/. [Πρόσβαση στις 02/10/2022].
- Robert Oostenveld's blog. Λήμμα «Restoring the AT firmware on the ESP8266». Διαθέσιμο στον δικτυακό τόπο: https://robertoostenveld.nl/esp8266-at-firmware/. [Πρόσβαση στις 01/10/2022].
- ShenZhen2U. Λήμμα «ESP-01S ESP8266 WiFi Module». Διαθέσιμο στον δικτυακό τόπο: https://www.shenzhen2u.com/ESP-01S-ESP8266-WIFI-MODULE. [Πρόσβαση στις 25/09/2022].
- Surtr Technology. Λήμμα «Interfacing HMC5883L / QMC5883 Digital compass with Arduino». Διαθέσιμο στον δικτυακό τόπο: https://surtrtech.com/2018/02/01/interfacing-hmc8553l-qmc5883-digital-compass-with-arduino/. Τελευταία ενημέρωση στις 1 Φεβρουαρίου 2018. [Πρόσβαση στις 01/03/2022].
- TeachMeMicro. Λήμμα «Use LM393 IR Module as Motor Speed Sensor». Διαθέσιμο στον δικτυακό τόπο: https://www.teachmemicro.com/lm393-ir-module-motor-speed-sensor/. [Πρόσβαση στις 06/03/2022].
- Tutorials Brainy-Bits. Λήμμα «How to use a Speed Sensor with Arduino». Διαθέσιμο στον δικτυακό τόπο: https://www.brainy-bits.com/post/how-to-use-a-speed-sensor-with-arduino. Τελευταία ενημέρωση στις 23 Οκτωβρίου 2020. [Πρόσβαση στις 06/03/2022].
- Zazootek. Λήμμα «Passive Speaker Buzzer Module for Arduino». Διαθέσιμο στον δικτυακό τόπο: https://www.zazootek.com.ng/product/passive-speaker-buzzer-module-for-arduino/. [Πρόσβαση στις 15/06/2022].