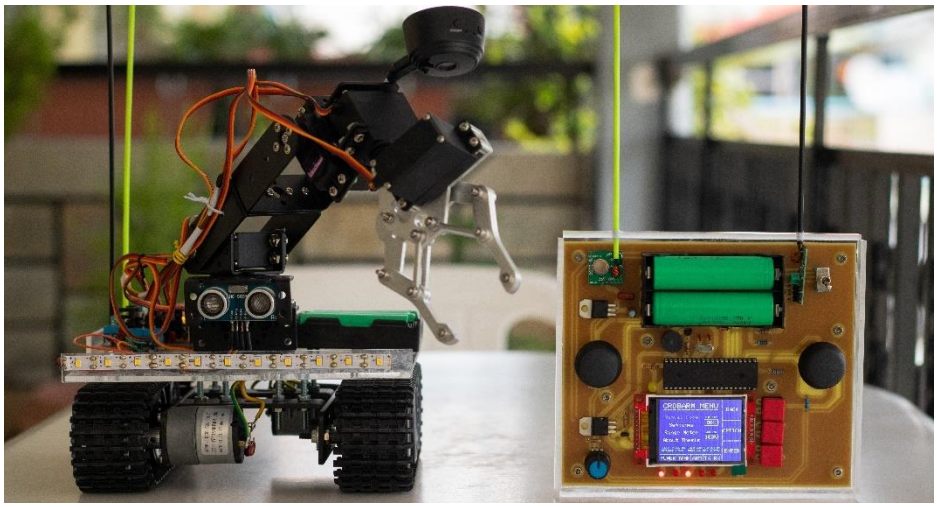
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

# Μελέτη Κατασκευή και Εφαρμογή Έξυπνου Ερπυστριοφόρου Ρομπότ με Βραχίονα Ελεγχόμενο Απομακρυσμένα Μέσω RF



***Φοιτητής***
***Στέφανος Λιάκας***
***Αριθμός Μητρώου: 517079***

***Επιβλέπων***
***Άγγελος Γιακουμής***
***Βαθμίδα: Λέκτορας***

**Θεσσαλονίκη,**
**Φεβρουάριος 2021**

Τίτλος Π.Ε: Μελέτη Κατασκευή και Εφαρμογή Έξυπνου Ερπυστριοφόρου Ρομπότ
με Βραχίωνα Ελεγχόμενο Απομακρυσμένα Μέσω RF
Κωδικός Π.Ε: 21108
Ονοματεπώνυμο φοιτητή: Στέφανος Λιάκας
Ονοματεπώνυμο εισηγητή: Άγγελος Γιακουμής
Ημερομηνία ανάληψης Π.Ε: 09/02/2021
Ημερομηνία περάτωσης Π.Ε: 13/09/2021

*Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.*

*Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Στέφανου Λιάκα που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.*

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

*Αφιερώνεται*

*στην οικογένεια μου*

*και τους φίλους μου*

# Πρόλογος

Η παρούσα πτυχιακή εργασία με τίτλο «Μελέτη Κατασκευή και Εφαρμογή Έξυπνου Ερπυστριοφόρου Ρομπότ με Βραχίωνα Ελεγχόμενο Απομακρυσμένα Μέσω RF», εκπονήθηκε στα πλαίσια της ολοκλήρωσης των προϋποθέσεων, για την λήψη του πτυχίου Ηλεκτρονικού Μηχανικού Α.Τ.Ε.Ι.Θ. Χάρη στην ταχύτατη εξέλιξη της τεχνολογίας, η ανθρωπότητα έχει κατορθώσει σημαντικά άλματα καινοτομίας με την άνοδο της επιστήμης της ρομποτικής, καθώς θα επηρεάσει σημαντικά τις ζωές μας μέσα στα επόμενα χρόνια. Ως επακόλουθο, θεωρήθηκε επιθυμητό η δημιουργία ενός χερσαίου κινούμενου ρομποτικού συστήματος, με την ονομασία Crobarm. Η κατηγορία που κατατάσσεται είναι τα Μη Επανδρωμένα Οχήματα (Unmanned Ground Vehicle).

Η επιλογή αυτού του θέματος, σήμαινε μια ευκαιρία για εξερεύνηση στον αναπτυσσόμενο τομέα της ρομποτικής. Επιθυμούσα μια πρόκληση, με αποτέλεσμα την απόκτηση νέων γνώσεων και εμπειριών, κατά την διάρκεια περάτωσης αυτού του έργου. Συμπληρωματικά, πιστεύω πως επωφελήθηκα αρκετά, λόγω του απαραίτητου συνδυασμού διάφορων πτυχών της ηλεκτρονικής, με απώτερο σκοπό να φτάσω σε ένα επιθυμητό ποιοτικό αποτέλεσμα. Η πτυχιακή εργασία βασίστηκε σε τομείς όπως τα Ενσωματωμένα Συστήματα (Embedded Systems), Τυπωμένα Κυκλώματα Πλακέτας (Printed Circuit Board), Αναλογικά Ηλεκτρονικά Κυκλώματα (Analog Electronics Circuit), Μικροελεγκτές (Microcontroller Unit), Ρομποτική (Robotics), Ασύρματες Επικοινωνίες (Wireless Communication), Ηλεκτροκίνηση (Electromobility) και Προγραμματισμό (Programing).

# Περίληψη

Σκοπός της πτυχιακής εργασίας ήταν να γίνει έρευνα, για τη κατασκευή ενός έξυπνου ερπυστριοφόρου ρομπότ, αποτελούμενο από βραχίονα καθώς και μια ξεχωριστή συσκευή τηλεχειρισμού. Η κύρια χρήση του Crobarm προορίζεται για έρευνα, μικρές εργασίες, παρατήρηση φυσικού περιβάλλοντος και μεταβίβαση πληροφοριών.

Πραγματοποιήθηκε ο σχεδιασμός, των απαραίτητων ηλεκτρονικών κυκλωμάτων ολόκληρου του συστήματος, με τις αντίστοιχες δυνατότητες και μηχανισμούς για το κάθε τμήμα. Σε επόμενο στάδιο, στήθηκαν τα τυπωμένα κυκλώματα πλακέτας και έγινε ο προγραμματισμός των μικροελεγκτών PIC. Η επικοινωνία μεταξύ των δύο συστημάτων επιτυγχάνεται ασύρματα, με την μετάδοση και τη λήψη κωδικοποιημένων ραδιοσυχνοτήτων κατά ASK. Οι πομποί έχουν εμβέλεια μέχρι τα 60 μέτρα σε ανοιχτό χώρο και εκπεμπόμουν συχνότητες των 315MHz. Μέσω μιας ειδικής φιλικής διεπαφής του τηλεχειριστήριου, υπάρχουν ποικίλες προσαρμογές για το μενού. Ο χρήστης επιλέγει οποιαδήποτε λειτουργία επιθυμεί, για τον άμεσο χειρισμό του ρομπότ. Βασικά στοιχεία είναι οι μοχλοί και το πόμολο ελέγχου, απαραίτητα για την κίνηση του τανκ ή του βραχίονα. Επίσης, περιλαμβάνεται αισθητήρας υπερήχων για την αποτροπή σύγκρουσης και μέτρηση απόστασης από αντικείμενα. Τέλος, η εγκατεστημένη κάμερα στέλνει ζωντανή μετάδοση βίντεο, από το ρομπότ στο έξυπνο τηλέφωνο.

Στο Κεφάλαιο 1 γίνεται μια γενική ανασκόπηση σε πέντε κύρια ρομποτικά συστήματα. Το Κεφάλαιο 2 περιγράφει την αρχιτεκτονική δομή και τις μονάδες λειτουργίας του μικροϋπολογιστικού συστήματος PIC. Στο Κεφάλαιο 3 παρουσιάζεται ο τρόπος οργάνωσης και λειτουργίας του συστήματος Crobarm, μαζί με τη δραστηριότητα μενού. Το Κεφάλαιο 4 επεξηγεί περαιτέρω, μερικά κρίσιμα χαρακτηριστικά των ηλεκτρονικών εξαρτημάτων που χρησιμοποιούνται. Στο Κεφάλαιο 5 αναφέρεται ο σχεδιασμός των PCB, οι κώδικες για το προγραμματισμό των μικροελεγκτών και οι προσομοιώσεις-πειραματισμοί, καταλήγοντας στην τελική εφαρμογή. Το Κεφάλαιο 6 κλείνει με συμπεράσματα και προτάσεις για μελλοντική βελτίωση.

# Research Construction and Application of Intelligent Crawler Robot with Arm Controlled Remotely Throw RF

## Abstract

The purpose of this thesis was to be performed research, in order to create a crawler robot, consisted of an arm as well as a separate device for remote control. The main use of Crobarm was meant for research, small tasks, observation of the natural ambient and information transfer.

The design of essential electronic circuits, was carried out for the whole system, with corresponding abilities and mechanisms in every section. In the next stage, printed circuit boards were set up and PIC microcontrollers were programmed. Communication between the two systems was achieved wireless, by sending and receiving coded radio frequencies over ASK. So, telecommunication is undertaken from modules of transmitter-receiver, for the incoming-outgoing data. Transmitters have a range of up to 60 meters in open space and work in 315MHz frequency. Remote control comes with a special friendly interface, which concentrates various adjustments for the menu. The user may select any function, for the direct operation of the robot. Joysticks and control knob are basic elements, for the necessary move of tank or arm. Also, there is an ultrasonic sensor to prevent crash and measures the distance from different objects. Finally, the installed camera sends live video stream, from the robot to the smartphone.

In Chapter 1 it is provided an overview of five main robotic systems. Chapter 2 describes the architecture structure and operating units of PIC's microcomputer system. In Chapter 3 it is presented, how the Crobarm system is organized and operated along with the menu activity. Chapter 4 clarifies furthermore, some crucial characteristics of the electronic components that are used to. In Chapter 5 it is referred the design of PCBs, codes for the programming part of microcontrollers and simulations-experiments, in order to obtain the final application. Chapter 6 closes with conclusions and suggestions for future improvement.

This work was accomplished by the undergraduate student Stefanos Liakas, at the International Hellenic University, School of Engineering, Department of Information and Electronic Engineering.

# Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Γιακουμή Άγγελο, για την εμπιστοσύνη που μου έδειξε, αναλαμβάνοντας το συγκεκριμένο θέμα. Οι συμβουλές και η υποστήριξη του αποτέλεσαν σημαντικά εργαλεία, κατά την διάρκεια περάτωσης αυτού του μεγάλου έργου. Σε αυτό το σημείο, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του τμήματος, για τις γνώσεις και τα εφόδια που μου πρόσφεραν.

Επίσης, οφείλω να ευχαριστήσω τον συνάδελφο ηλεκτρολόγο μηχανικό Αριστείδη Κιμιρτζή, για την αξιοσημείωτη συνδρομή του, στο κατασκευαστικό μέρος των τυπωμένων κυκλώματάτων. Ήταν μια πρωτόγνωρη εμπειρία για μένα, οπότε με τη βοήθεια του μπόρεσα να μάθω την διαδικασία.

Επιπλέον, θέλω να ευχαριστήσω τους φίλους μου που με στήριξαν ηθικά, ξεπερνώντας έτσι όλα τα εμπόδια που αντιμετώπιζα κάθε φορά. Κλείνοντας, ένα μεγάλο ευχαριστήσω στην οικογένεια μου, για την συμπαράσταση και την κατανόηση που μου έδειξε αυτούς τους δύσκολους μήνες, δίνοντάς μου δύναμη, κουράγιο και οικονομική αρωγή για την ολοκλήρωση της πτυχιακής εργασίας.

# Περιεχόμενα

## Κεφάλαιο 3ο: Ανάλυση Λειτουργίας Συστήματος Crobarm



## Κεφάλαιο 4ο: Ιδιότητες Ηλεκτρονικών Εξαρτημάτων

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες

| | |
|---|---|
| ΔΙ.ΠΑ.Ε | Διεθνές Πανεπιστήμιο Ελλάδος |
| Α.Τ.Ε.Ι.Θ. | Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλονίκης |
| Π.Ε. | Πτυχιακή Εργασία |
| Crobarm | Crawler Robot with Arm |
| UGV | Unmanned Ground Vehicle |
| UAV | Unmanned Aerial Vehicle |
| ROV | Remote Operating Underwater Vehicle |
| Li-ion | Lithium-ion battery |
| IDE | Integrated Development Environment |
| R&D | Research and Development |
| MCU | Microcontroller Unit |
| PWM | Pulse Width Modulation |
| LED | Light Emitting Diode |
| TX | Transmit |
| RX | Receive |
| REM-CONT | Remote Control |
| ROB-SYS | Robotic System |
| SCARA | Selective Compliance Articulated Robot Arm |
| RPA | Remotely Piloted Aircraft |
| IC | Integrated Circuit |
| MCU | Microcontroller Unit |
| CMOS | Complementary Metal Oxide Semiconductor |
| ALU | Arithmetic Logic Unit |
| RAM | Random Access Memory |
| ROM | Reading Only Memory |
| CCP | Capture/Compare/PWM |
| CPU | Central Processing Unit |
| ΚΜΕ | Κεντρική Μονάδα Επεξεργασίας |
| MC | Machine Cycle |
| CISC | Complex Instruction Set Computer |

| | |
|---|---|
| RISC | Reduced Instruction Set Computer |
| SFRs | Special Function Registers |
| GFRs | General Function Registers |
| PLL | Phase-Locked Loop |
| MSSP | Master Synchronous Serial Port |
| SPI | Serial Peripheral Interface |
| $I^2C$ | Inter-Integrated Circuit |
| BJT | Bipolar Junction Transistor |
| TTL | Transistor-Transistor Logic |
| OP AMP | Operational Amplifier |
| ASK | Amplitude Shift Keying |
| AM | Amplitude Modulation |
| CP | Crobarm Protocol |
| PCB | Printed Circuit Board |
| EDA | Electronic Design Automation |
| UV | Ultraviolet |
| CCS | Custom Computer Services |
| MOSFET | Metal Oxide Semiconductor Field Effect Transistor |
| PIC | Peripheral Interface Controller |
| DC | Direct Current |
| AC | Alternating Current |
| FPS | Frame per second |

# Κεφάλαιο 1ο: Ρομποτικά Συστήματα

## 1.1 Εισαγωγή στα Ρομπότ

Τα τελευταία χρόνια η επιστήμη της ρομποτικής (Science of Robotics) με μεγάλα βήματα προόδου, έφερε σημαντικές και θετικές αλλαγές στην ζωή του ανθρώπου κάνοντάς την πιο εύκολη. Πρόκειται για έναν σχετικά νέο τεχνολογικό κλάδο, αποτελούμενο από το συνδυασμό τριών κύριων επιστημών που είναι η πληροφορική, ηλεκτρολογία και μηχανολογία. Επί της ουσίας, το ρομπότ είναι μια προγραμματιζόμενη μηχανή, προοριζόμενη στο να εκτελεί μια σειρά από απαιτητικές ενέργειες, με μεγάλη ακρίβεια. Έχει την δυνατότητα να λειτουργεί αυτόνομα ή υπό τον έλεγχο χειριστή. Επίσης, μπορεί να φέρει εις πέρας εργασίες κάτω από δύσκολες καιρικές συνθήκες, σε επικίνδυνα ή μη προσβάσιμα για τον άνθρωπο μέρη.

Μια ρομποτική μηχανή αποτελείται από δύο βασικά σκέλη: το υλικό (Hardware) και το λογισμικό (Software). Ο όρος υλικό αναφέρεται στα εργοστασιακά κατασκευασμένα εξαρτήματα, τα οποία εκτελούν ποικίλες λειτουργίες μιας μεγαλύτερης εργασίας. Για παράδειγμα, χρησιμοποιούν μπαταρίες λιθίου (LI-ion), κινητήρες σέρβο (Servo Motors), αισθητήρες (Sensors), σύστημα ελέγχου (Control System), μικροελεγκτή (Microcontroller). Από την άλλη πλευρά το λογισμικό δεν έχει υλική υπόσταση, είναι ένα πλήθος εντολών το οποίο δημιουργεί ο άνθρωπος σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Παραδείγματος χάριν: στο Atmel Studio, MPLAB, Arduino Software. Ο επονομαζόμενος αυτός κώδικας, αποθηκεύεται στην μνήμη του μικροελεγκτή ή μικροεπεξεργαστή του ρομπότ, για να μπορέσει να γίνει εκτέλεση του προγράμματος.

Είναι γεγονός πως λόγω της ευρείας χρήσης των ρομποτικών συστημάτων (Robotic Systems), έχει δημιουργηθεί μεγάλο πλήθος εφαρμογών. Όμως, στο παρόν κεφάλαιο θα αναλυθούν μόνο οι πέντε κύριες κατηγορίες. Μελλοντικά, είναι βέβαιο πως οι επόμενες γενιές μηχανικών θα αναπτύξουν περαιτέρω τις λειτουργίες των ρομπότ, ώστε να έχουμε νέα είδη, ανάλογα με τις απαιτήσεις που θα προκύπτουν. Όπως τα νανορομπότ (Nanorobotics) στην ιατρική.

Γενικότερα, δραστηριοποιείται ένας μεγάλος αριθμός εταιριών στο τομέα της ρομποτικής, τόσο με τμήματα έρευνας και ανάπτυξης (R&D) όσο και στο επιχειρησιακό τομέα. Εταιρίες όπως η Boston Dynamics, Hanson Robotics, Kuka Robotics, Fanuc Robotics, Gesar Inc, Autonomous Solutions, Oceaneering International, Rovco, Parrot, Skydio και αρκετές άλλες.

Σε αυτό το κεφάλαιο θα εξεταστούν οι παρακάτω 5 κατηγορίες:

- Μη Επανδρωμένο Επίγειο Όχημα
- Ρομποτικός Βραχίονας
- Ανθρωποειδή Ρομπότ
- Μη Επανδρωμένο Αεροσκάφος
- Τηλεχειριζόμενο Υποβρύχιο Όχημα

## 1.2 Μη Επανδρωμένο Επίγειο Όχημα

### 1.2.1 Ορισμός UGV

Τα Μη Επανδρωμένα Επίγεια Οχήματα (UGV) είναι μια ιδιαίτερη οικογένεια ρομπότ, όπου λειτουργούν χωρίς την ανθρώπινη παρουσία επί παντός εδάφους. Είναι κατάλληλα για να οδηγούνται σε δύσβατες περιοχές με μυστικότητα και ευκινησία. Τροφοδοτούνται από πολλαπλές πηγές ενέργειας, όπως είναι οι μπαταρίες, τα καύσιμα και η ηλιακή ενέργεια. Με ένα σύστημα τηλεχειρισμού, ο άνθρωπος δύναται να ελέγχει το κινούμενο ρομπότ μέσω μιας διεπαφής (Interface). Δηλαδή, όλες οι ενέργειες καθορίζονται από τον χειριστή, ο οποίος μπορεί να έχει άμεση οπτική επαφή ή απομακρυσμένα με την χρήση ψηφιακής βιντεοκάμερας. Επιπλέον, κάποια από αυτά μπορούν να προσαρμοστούν στο περιβάλλον τους και να δραστηριοποιηθούν αυτόνομα αν απαιτηθεί. Για την ακρίβεια μαθαίνουν μέσα από αυτό, αξιοποιώντας τη μηχανική μάθηση (Machine Learning). Οι εταιρίες δημιουργούν εφαρμογές κυρίως για το στρατό, αστυνομία, πολιτεία και διαστημικές εξερευνήσεις [20].

Σχήμα 1.1 FLIR Centaur, Medium-Sized UGV [38]

### 1.2.2 Πλεονεκτήματα UGV

Το κύριο πλεονέκτημα των UGV στο στρατό ξηράς, είναι η έρευνα που παρέχουν για ύποπτο εξοπλισμό σταθερών βομβών όπως νάρκες. Ειδικά εκπαιδευμένο προσωπικό με τη χρήση του ρομπότ, μπορεί να ανακαλύψει πλήθος εκρηκτικών και να τα αφοπλίσει. Σε περίπτωση που δεν γίνεται επιτυχώς ο αφοπλισμός, τα UGV χρησιμοποιούνται για να πυροδοτήσουν την έκρηξη. Με αυτό τον τρόπο αποτρέπεται ο θάνατος των στρατιωτών και η καταστροφή τεθωρακισμένων οχημάτων [21].

Ο ρόλος τους είναι σημαντικός για να εντοπίζουν την θέση του εχθρού, προτού έρθουν σε επαφή οι στρατιώτες και τραυματιστούν στην μάχη. Υπάρχει μεγάλη πιθανότητα να συναντήσουν μια ενέδρα ή περίπολο στο δρόμο τους, ωστόσο το ρομπότ είναι ικανό να σταματήσει την σύγκρουση, καθώς θεωρούνται ανώτερες μηχανές μάχης. Με την παραπάνω μέθοδο, εξασφαλίζεται η ζωή των ενόπλων δυνάμεων και συνεχίζουν την επιχειρησιακή τους δράσης (Army Mission) [21].

Τα μεγαλύτερα Μη Επανδρωμένα Επίγεια Οχήματα που στην πλειοψηφία τους είναι αμφίβια, βοηθάνε στο πεδίο μάχης με πυρά από πολυβόλα/πυραύλους ή κουβαλάνε εφόδια. Παραδείγματος χάρη πρώτες βοήθειες, τρόφιμα, πυρομαχικά, εκρηκτικά, τηλεπικοινωνίες και άλλα [21].

### 1.2.3 Μειονεκτήματα UGV

Σημαντικό μειονέκτημα για τα χειριστήρια των UGV, θεωρείται μια σειρά σφαλμάτων που καλούνται αισθητή καθυστέρηση (lagging). Ορισμένες φορές λόγο της αποτυχίας συγχρονισμού, προκύπτει αργός χρόνος αντίδρασης για τον χειριστή και το ρομπότ. Με αποτέλεσμα, τον εντοπισμό του από τον εχθρό, το πάτημα μιας νάρκης αλλά και γενικότερα η πρόκληση οποιουδήποτε είδους ζημιά [21].

Αρκεί μια στιγμή για να προκύψει κάποιο τεχνικό πρόβλημα στο UGV, οπότε κατά την διάρκεια πολέμου καθιστώντας το άχρηστο. Ένα ξαφνικό λάθος στο σύστημα, μπορεί να σημαίνει την δραματική πιθανότητα τραυματισμού ή το θάνατο των στρατιωτών από τον εχθρό [21].

Ένα ακόμη μειονέκτημα είναι το κόστος για την δημιουργία των UGV. Είναι αρκετά δαπανηρή η διαδικασία για τη παραγωγή τέτοιων ρομπότ, διότι απαιτούνται πολλοί μηχανικοί ξοδεύοντας αμέτρητο χρόνο για σχεδιασμό και δοκιμές (Testing). Τέλος, για το κάθε όχημα χρειάζεται να γίνει ειδικός προγραμματισμός, σύμφωνα με την εκάστοτε δομή και εξόπλισή του [21].

Σχήμα 1.2 Rheinmetall Mission Master SP [39]

## 1.3 Ρομποτικός Βραχίωνας

### 1.3.1 Ορισμός Ρομποτικού Βραχίονα

Ο ρομποτικός βραχίονας (Robotic Arm) θυμίζει τον ανθρώπινο βραχίονα, επειδή μιμείται τις κινήσεις του με παρόμοιες λειτουργίες. Το ρομποτικό σύστημα, μπορεί να αποτελείται μόνο από το βραχίονα ή να είναι κομμάτι μιας πιο περίπλοκης μηχανικής δομής. Απαρτίζεται από μια κινηματική αλυσίδα στερεών κομματιών, που συνδέονται μεταξύ τους με αρθρώσεις. Πιο συγκεκριμένα, χωρίζεται σε δύο τύπους: τη περιστροφική άρθρωση και τη πρισματική άρθρωση. Η περιστροφική άρθρωση, κάνει σχετική στροφή μεταξύ δύο γειτονικών συνδέσμων, γύρω από τον άξονα της άρθρωσης. Ενώ, η πρισματική άρθρωση επιτρέπει την σχετική μετατόπιση, κατά μήκος του άξονα της άρθρωσης. Επίσης, στο άκρο του πάντα έχει ένα εργαλείο τελικής δράσης (Αρπάγη). Με το συνδυασμό αυτών των δύο, δημιουργούνται συγκεκριμένες δομές βραχίονα, σύμφωνα με την εφαρμογή που θα έχει. Χρησιμοποιούνται συχνά για την εκτέλεση βαριών εργασιών και για επαναλαμβανόμενες διαδικασίες εκτεταμένης χρονικής περιόδου. Γενικότερα, είναι πολύτιμα για τη βιομηχανική παραγωγή και συνηθίζεται να έχουν 4 έως 6 βαθμούς ελευθερίας [4] [19].

Σχήμα 1.3 Βιομηχανικός Κυλινδρικός Ρομποτικός Βραχίονας [40]

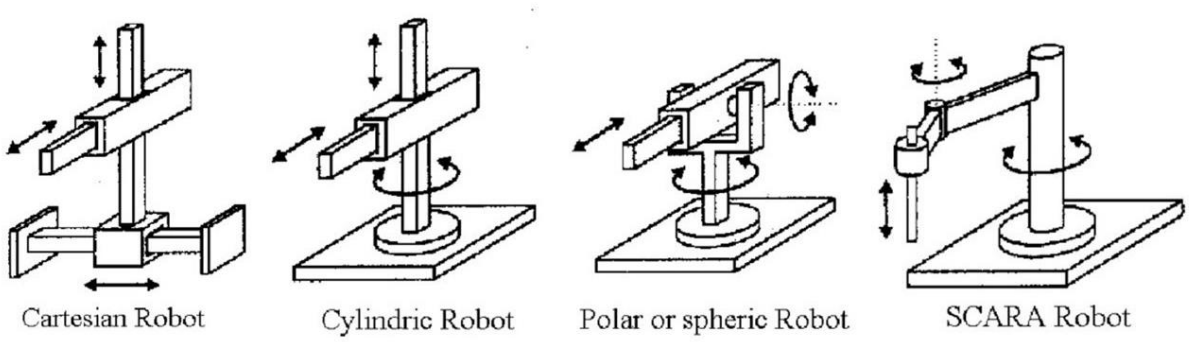
### 1.3.2 Κατηγορίες Ρομποτικού Βραχίονα

Η αγορά διαθέτει διαφορετικούς τύπους ρομποτικών βραχιόνων και ο καθένας από αυτούς έχει σχεδιαστεί με σημαντικές ικανότητες. Όλες οι κατηγορίες είναι μοναδικές στο τομέα τους και κατάλληλες να εξυπηρετούν αντίστοιχους σκοπούς. Για αυτό τον λόγο μια παραγωγική γραμμή υψηλής τεχνολογίας, έχει σε κάθε θέση επίλεκτες μηχανές όπου είναι πλήρως εξιδεικευμένες. Παρακάτω αναφέρονται τέσσερα είδη ρομποτικού βραχίονα:

1. Καρτεσιανός Ρομποτικός Βραχίονας. (Cartesian Robotic Arm) Αποτελείται από τρείς πρισματικές αρθρώσεις, που προγραμματίζονται σύμφωνα με τις συντεταγμένες X, Y και Z. Προφανώς ο καθορισμός μιας γραμμικής κίνησης, γίνεται σε τρείς διαστάσεις κατά μήκος αυτών των τριών αξόνων. Δηλαδή, κινείται προς τα πάνω-κάτω (Z), μπροστά-πίσω (Y) και μέσα-έξω (X) [19].

2. Κυλινδρικός Ρομποτικός Βραχίονας. (Cylindrical Robotic Arm) Οι άξονες του βραχίονα σχηματίζουν ένα κυλινδρικό σύστημα συντεταγμένων, με τη πρώτη άρθρωση στη βάση να είναι περιστροφική. Συναντάται συχνά σε εργασίες συναρμολόγησης, χειρισμό εργαλειομηχανών, συγκόλλησης και άλλα συναφή [19].

3. Σφαιρικός Ρομποτικός Βραχίονας. (Spherical Robotic Arm) Οι πρώτες δύο αρθρώσεις είναι περιστροφικές, ώστε να λειτουργούν σε ένα σφαιρικό χώρο εργασίας. Είναι χρήσιμος όπως ο κυλινδρικός για την εκτέλεση παρόμοιων εργασιών [19].




Σχήμα 1.4 Κατηγορίες Ρομποτικού Βραχίονα [26]

4. SCARA. (Selective Compliance Articulated Robot Arm) Το ρομπότ SCARA μεταφράζεται στα ελληνικά, ως Επιλεκτική Συμμόρφωση Συναρμολόγησης Ρομπότ Βραχίονα. Στερεώνεται στον άξονα Z της βάσης μια περιστροφική άρθρωση όπως και ο αμέσως επόμενος σύνδεσμος. Μόνο το τελικό εργαλείο αποτελείται από πρισματική. Το ρομπότ είναι δύσκαμπτο προς τον άξονα Z αλλά εύκαμπτο στον X και Y [19].

## 1.4 Ανθρωποειδής Ρομπότ

### 1.4.1 Ορισμός Ανθρωποειδών Ρομπότ

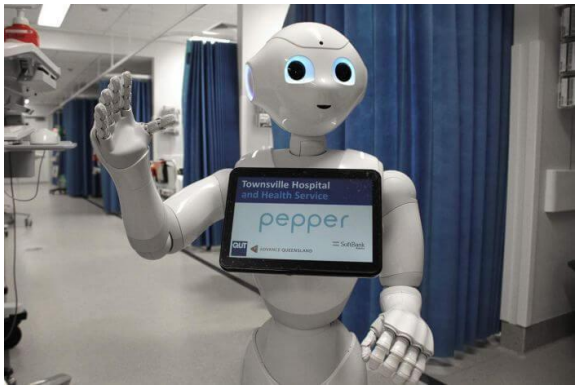
Η λογική του ανθρωποειδούς ρομπότ, είναι να αντικατοπτρίζει τόσο το ανθρώπινο σώμα, όσο και την συμπεριφορά. Αποτελούμενο από ένα κεφάλι, κορμό, δύο χέρια και δύο πόδια. Μπορεί να επικοινωνεί ερμηνεύοντας την κάθε πληροφορία που λαμβάνει, εκτελώντας δραστηριότητες σύμφωνα με αυτές που του έχει προκαθορίσει ο άνθρωπος. Ορισμένες μορφές ρομπότ, αποτελούνται μόνο από συγκεκριμένα μέρη του σώματος, για παράδειγμα, μπορεί να μην έχουν πόδια. Αναπαράγουν τα χαρακτηριστικά του ανθρώπινου προσώπου όπως τα μάτια και το στόμα. Αισθητικά, μιμούνται σε υψηλό βαθμό τον άνθρωπο, αλληλοεπιδρούν αρκετά με το περιβάλλον τους, αναγνωρίζοντας πρόσωπα, συνομιλώντας και πραγματοποιώντας ενέργειες. Όλα τα παραπάνω, γίνονται με τη χρήση της τεχνητής νοημοσύνης, καθώς με την πάροδο του χρόνου γίνεται πιο έξυπνο, μαθαίνοντας καινούργια πράγματα συνεχώς. Όμως, υπάρχουν και ρομπότ τα οποία είναι προ-προγραμματιζόμενα (pre-programmable), για να κάνουν μια σειρά από περιορισμένες δραστηριότητες [25].

Σχήμα 1.5 Hanson Robotics, Sophia [28]

### 1.4.2 Κατηγορίες Ανθρωποειδών Ρομπότ

Τα ανθρωποειδή ρομπότ ορίζονται σε τρία πεδία εφαρμογών, καθότι είναι αρκετά χρήσιμα στην υγειονομική περίθαλψη, στην εκπαίδευση και στην κοινωνικοποίηση. Όσον αφορά την υγειονομική περίθαλψη, το ρομπότ υφίσταται για να βοηθάει τον ασθενή. Παρακολουθεί την ιατρική κατάσταση του ανθρώπου και χορηγεί την κατάλληλη φαρμακευτική αγωγή, είτε στο σπίτι είτε στο νοσοκομείο. Επιπλέον, είναι ικανό να υπηρετήσει τη μόρφωση σε κάθε εκπαιδευτικό φορέα όπως γυμνάσια, λύκεια

πανεπιστήμια. Με αυτόν τον τρόπο βελτιστοποιείται και εξελίσσεται η ποιότητα γνώσης που προσφέρεται στα παιδιά. Τα κοινωνικά ανθρωποειδή ρομπότ χρησιμοποιούνται από οργανισμούς ή προσωπικό, για να εξυπηρετούν τους ανθρώπους στις καθημερινές τους δραστηριότητες. Επιπρόσθετα, κάνουν εποικοδομητικές συζητήσεις υψηλού επιπέδου, με αποτέλεσμα οι άνθρωποι να μάθουν πράγματα που πιθανόν να μην γνώριζαν [25].




Σχήμα 1.6 SoftBank Robotics, Pepper [29]

## 1.5 Μη Επανδρωμένο Αεροσκάφος

### 1.5.1 Ορισμός UAV

Τα Μη Επανδρωμένα Αεροσκάφη (UAV) λειτουργούν χωρίς την άμεση παρουσία του ανθρώπου. Με άλλα λόγια δεν απαιτείται πιλότος ή πλήρωμα στο αεροσκάφος. Αξίζει δε να σημειωθεί πως ο περισσότερος κόσμος, αναφέρεται σε αυτά τα ρομπότ ως ντρόουν (Drone). Λόγω του ότι διαθέτουν ένα ειδικό σύστημα απομακρυσμένου ελέγχου για το αεροσκάφος (RPA), το προσωπικό έχει τη δυνατότητα να το διαχειρίζεται από μια ασφαλή στρατιωτική βάση (ή μια οποιαδήποτε τοποθεσία), δίχως να κινδυνεύει η ζωή του. Βέβαια, υπάρχουν κάποιοι μηχανισμοί που μπορούν να τεθούν σε εφαρμογή, όπως η βοήθεια αυτόματου πιλότου. Επίσης, δεν απαιτείται από τον χρήστη να επιτηρεί συνέχεια ή να παρεμβάλλει στο εναέριο ρομπότ, αλλά να ενεργοποιήσει την πλήρη αυτονομία του. Το UAV θεωρείται

Σχήμα 1.7 General Atomics MQ-9 Reaper [30]

σύμμαχος για τη πολεμική αεροπορία, διότι διαθέτουν αισθητήρες, προσδιοριστές στόχου, πυραυλικό εξοπλισμό και ηλεκτρονικούς πομπούς που παρεμβάλουν τα συστήματα των εχθρικών στόχων. Αυτά τα χαρακτηρίστηκα τα καθιστά πολύ πιο αποδοτικά, προσφέροντας μεγαλύτερη εμβέλεια και αντοχή σε σύγκριση με τα επανδρωμένα αεροσκάφη. Εξυπακούεται, πως μεταχειρίζονται στην κοινωνία από τους πολίτες για επιτήρηση, παραγωγή βίντεο, εναέριες φωτογραφίες, μεταφορά προϊόντων (delivery), παρακολούθηση και άλλα [22][26].

### 1.5.2 Πλεονεκτήματα UAV

Ένα σημαντικό πλεονέκτημα που έχουν τα UAV, είναι ότι μπορούν να πετάξουν σε περιοχές υψηλής επικινδυνότητας, αντί για το πιλότο. Επιπλέον, είναι ικανά για να παραμένουν στους αιθέρες μέχρι και 30 ώρες, εκτελώντας επαναλαμβανόμενες ενέργειες με ακριβή απόδοση. Σαρώνουν μια συγκεκριμένη περιοχή μέρα-νύχτα, ακόμη και με βροχή ή ομίχλη. Με αυτόν τον τρόπο προστατεύεται η ζωή του πιλότου. Το θετικό είναι πως τα UAV κοστολογούνται φτηνότερα σε σχέση με τα συνηθισμένα μαχητικά αεροσκάφη [13].

Πρέπει να σημειωθεί ότι τα UAV, είναι πολύτιμο εργαλείο στο κομμάτι της κατασκοπίας, της επιτήρησης, στην αναγνώριση και στην αυξημένη απόκτηση στρατιωτικής νοημοσύνης (Military Intelligence). Είναι γεγονός ότι είναι το πιο εύκολο και γρήγορο μέσο για να αναλάβει στρατιωτική δράση, μέχρι και την τελευταία στιγμή. Αποδεικνύεται ότι έχουν υψηλότερη ακρίβεια για να χτυπήσουν το στόχο τους από μεγάλες αποστάσεις, μειώνοντας έτσι την πιθανότητα να πεθάνουν πολίτες ή να καταστραφούν άλλες υποδομές. Σημειώνεται πως παραμένουν τόσο θανατηφόρα όπλα, όσο είναι και τα επανδρωμένα πολεμικά αεροσκάφη [13].

Σχήμα 1.8 Τετρακόπτερο, Phantom 4 Pro [31]

### 1.5.3 Μειονεκτήματα UAV

Τα UAV μειονεκτούν ως προς τον έλεγχο τους, διότι αν ο άνθρωπος σε μια χρονική στιγμή κάνει κάποιο λάθος, μπορεί να προκληθεί ζημία ή ακόμη και η καταστροφή του. Ένα άλλο πρόβλημα που θα μπορούσε να προκύψει, είναι να χτυπήσει κάποιο σφάλμα τα ηλεκτρονικά συστήματά του ή το λογισμικό του. Συνεπώς, καταλήγει να καταστραφεί το ιπτάμενο όχημα εφόσον πέσει κάτω στο έδαφος.

Η ζημία που προκαλείται, είναι της τάξης των εκατομμυρίων ευρώ για ένα στρατιωτικό drone ή μερικές εκατοντάδες ευρώ αν πρόκειται για πολιτικό drone. Όμως, το πιο σημαντικό είναι πως θα έχουμε ανθρώπινες απώλειες και καταστροφή περιουσιών. Τα UAV δεν είναι τέλεια συστήματα, δηλαδή αν υπάρξει κάποια δυσλειτουργία, χάνεται ο έλεγχος του αεροσκάφους από τον άνθρωπο, με άγνωστες τις συνέπειες. Είναι χαρακτηριστικό, πως μερικοί εγκληματίες εκμεταλλεύονται τα drone χρησιμοποιώντας τα για τρομοκρατικές επιθέσεις. Ένα άλλο παράδειγμα, είναι η μεταφορά εγκληματικών αγαθών όπως ναρκωτικά, κάνναβη και τα λοιπά. Αξίζει να αναφερθεί το γεγονός, πως υπάρχει ενδεχόμενο να έρθει σε επαφή με κάποιο εμπόδιο, αεροσκάφος, ελικόπτερο, κτίριο λόγω των κρυφών γωνιών [13].

## 1.6 Τηλεχειριζόμενο Υποβρύχιο Όχημα

### 1.6.1 Ορισμός ROVs

Το Τηλεχειριζόμενο Υποβρύχιο Όχημα (ROV) είναι ένα μη επανδρωμένο ρομποτικό σκάφος, το οποίο ενεργεί κάτω από το νερό σε μεγάλα ή μικρά βάθη, αναλόγως την έρευνα που διεξάγει κάθε φορά. Ο έλεγχός του υποβρυχίου επιτυγχάνεται απομακρυσμένα από το χειριστή του, ο οποίος πλοηγεί το ρομπότ μέσα από ένα ειδικό σύστημα διεπαφής. Γενικά, το ROV διαφέρει σε σχέση με υπόλοιπα ρομπότ, λόγω του ότι απαιτείται η σύνδεση ενός καλωδίου (Umbilical Cable) μεταξύ αυτού και του πλοίου. Πρόκειται για ένα ανθεκτικό καλώδιο, που περιλαμβάνεται από ηλεκτρικούς αγωγούς και οπτικές ίνες, για την κάλυψή των αναγκών του ρομπότ σε τροφοδοσία και επικοινωνίες. Ως υποβρύχια ρομπότ είναι αρκετά ευέλικτα, καθώς μπορούν να κινηθούν σε όλους τους άξονες, τουτέστιν, σε οποιοδήποτε σημείο της θάλασσας. Μετακινούνται με ταχύτητες συνήθως από 0.5-3 κόμβους, σύμφωνα με το βάθος και το ρεύμα της θάλασσας. Φτάνουν σε αξιοπρόσεχτα βάθη μέχρι και τα 4.500 μέτρα. Διαθέτουν, πλήθος οργάνων όπως αισθητήρα υπερήχων, μαγνητόμετρο, ψηφιακή βιντεοκάμερα, βραχίονες, δειγματολήπτες νερού, προβολείς, προωθητές και αισθητήρα θερμοκρασίας [14].

Το ROV χρειάζεται τρία με τέσσερα άτομα τουλάχιστον, για να διαχειριστή σωστά στην ανοικτή θάλασσα. Ο κυβερνήτης είναι υπεύθυνος για την αποτελεσματική πλοήγηση του ρομπότ. Ο συγκυβερνήτης μπορεί να βοηθήσει στη χρήση των βραχιόνων και να προσέχει παράλληλα την θέση του οχήματος. Πάντα υπάρχει μια ομάδα επιστημών που κάθεται πίσω από τους χειριστές, η οποία είναι

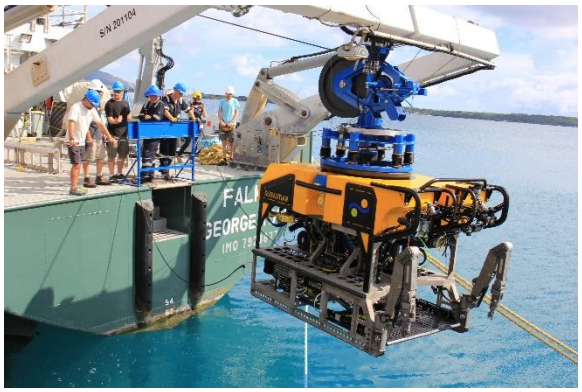

Σχήμα 1.9 ECA GROUP, H1000, ROV [32]

αρμόδια για να κρατάει σημειώσεις, να καταγράφει δεδομένα, στοιχεία, να λαμβάνει αποφάσεις άλλα και να δίνει οδηγίες πλοήγησης στους χειριστές [14].

### 1.6.2 Εφαρμογές ROVs

Υπάρχει πλήθος βιομηχανιών όπου εκμεταλλεύονται τις δυνατότητες των ROV, διευκολύνοντας πολλές φορές τις ενέργειες ή εργασίες που θέλουν να εκτελέσουν. Αναφέρονται οι παρακάτω περιπτώσεις [23]:

- Υδατοκαλλιέργειες
- Εμπορικές Καταδύσεις
- Δημοτικές Υποδομές
- Στρατός
- Επιστήμη του Ωκεανού
- Πετρέλαιο και Ενέργεια
- Ναυτιλία
- Υποβρύχια Εξερεύνηση



Σχήμα 1.10 Schmidt Ocean Institute, ROV [33]

# Κεφάλαιο 2ο: Μικροϋπολογιστικό Σύστημα PIC

## 2.1 Εισαγωγή στους Μικροελεγκτές

Η λέξη μικροελεγκτής, παραπέμπει σε ένα υπολογιστικό σύστημα που έχει τη μορφή ολοκληρωμένου κυκλώματος (Integrated Circuit), καλυπτόμενο από ένα πλαστικό περίβλημα. Προγραμματίζεται με σκοπό να ελέγχει ηλεκτρονικές συσκευές, μηχανήματα ή περίπλοκα ενσωματωμένα συστήματα. Για αυτό τον λόγο πολλές φορές αναφέρεται και ως ενσωματωμένος ελεγκτής, επειδή εργοστασιακά δημιουργούνται και τοποθετούνται απευθείας στις συσκευές που προορίζονται. Τυπικά κάθε μικροελεγκτής πρέπει να διαθέτει τουλάχιστον μια κεντρική μονάδα επεξεργασίας (CPU), μνήμη προγράμματος (Program Memory), μνήμη δεδομένων (Data Memory) και τις πόρτες εισόδου/εξόδου (Input/Output). Χρησιμοποιήθηκαν δύο 8 bit μικροελεγκτές PIC18F46K20 της αμερικάνικης εταιρίας Microchip Technology. Πρόκειται για μια οικογένεια υψηλής απόδοσης Συμπληρωματικού Ημιαγωγού Μετάλλου Οξειδίου (CMOS) [1].

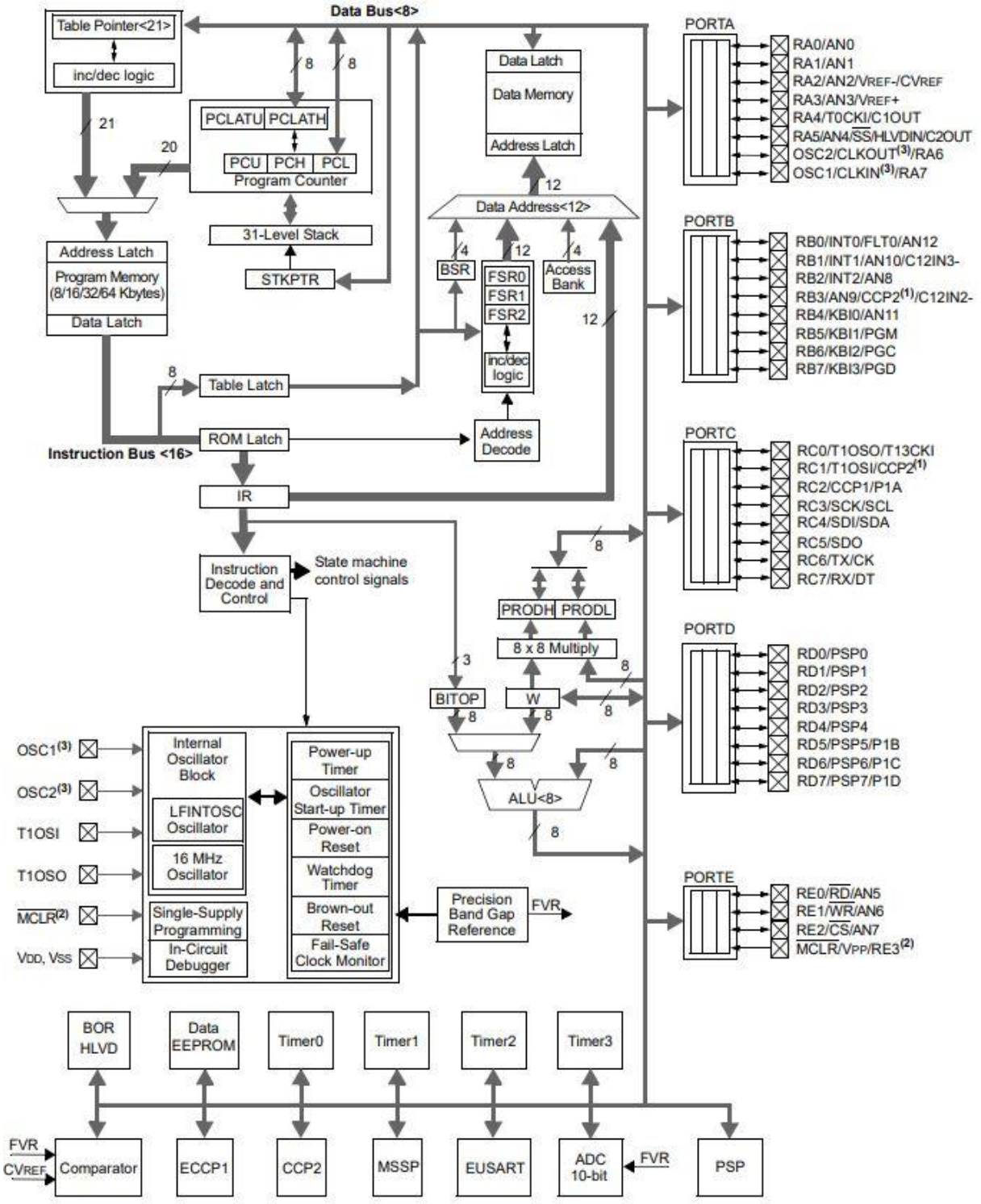

Σχήμα 2.1 Μικροελεγκτής PIC 40-PIN PDIP (Dual In-Line Package) [34]

## 2.2 Αρχιτεκτονική Μικροελεγκτή

Ο μικροελεγκτής αποτελείται από διάφορα στοιχεία, τα οποία έχουν σχεδιαστεί για να εξυπηρετούν μια συγκεκριμένη λειτουργία στο σύστημα. Όλα τα παρακάτω είναι κυκλώματα που περιλαμβάνει ο PIC με λειτουργίες στο πυρήνα του, τις περιφερειακές μονάδες του και τα ειδικά χαρακτηριστικά του [7]:

- Μετατροπέας Αναλογικού Σήματος σε Ψηφιακό (A/D Converter)
- Πόρτες Εισόδου/Εξόδου (Port I/O)
- Διακοπές (Interrupts)
- Χρονιστές (Timers)
- Μνήμη Δεδομένων RAM
- Μνήμη Προγράμματος ROM
- Αριθμητική & Λογική Μονάδα (ALU)
- Συγκριτής (Comparator)
- Διάφοροι Καταχωρητές (Registers): εργασίας, κατάστασης, μετρητής προγράμματος κτλ

- Καταγραφή, Σύγκριση και Διαμόρφωση Εύρους Παλμών (CCP)
- Μονάδα Ταλαντωτή (Oscillator Module)
- Κύρια Σύγχρονη Σειριακή Θύρα (MSSP)

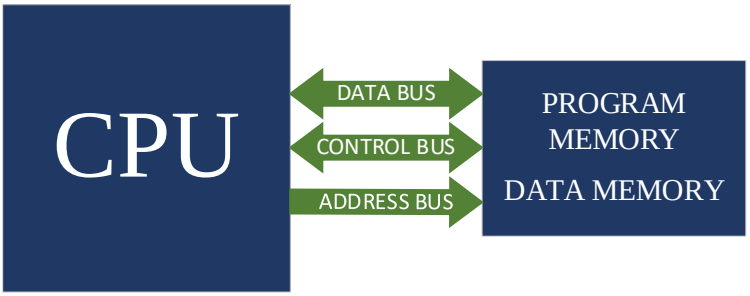



Σχήμα 2.2 Μπλοκ Διάγραμμα Αρχιτεκτονικής Δομής PIC18F46K20 [12]

Υπάρχουν δύο κύριες αρχιτεκτονικές στις οποίες μπορούν να διαχωριστούν οι μικροελεγκτές. Η πρώτη αρχιτεκτονική είναι η Von Neumann ενώ η δεύτερη είναι η Harvard. Ο μικροελεγκτής PIC βασίζεται στην αρχιτεκτονική Harvard.

Η αρχιτεκτονική Von Neumann αποτελείται από μια κοινή μνήμη, η οποία εμπεριέχει τις εντολές του προγράμματος καθώς και τα αποθηκευμένα δεδομένα. Η εκτέλεση των εντολών γίνεται με σχετικά πιο αργό ρυθμό, επειδή υπάρχει μόνο ένας δίαυλος και για τις δύο μνήμες. Αδυνατεί να προσπελάσει στη μνήμη ταυτόχρονα για τις εντολές και τα δεδομένα. Σχετικά με την εκτέλεση μιας εντολής απαιτούνται περισσότεροι κύκλοι μηχανής (MC). Επίσης, ο επεξεργαστής ανήκει στην οικογένεια των λεγόμενων Υπολογιστών με Διευρυμένο Ρεπερτόριο Εντολών (CISC). Ένας μικροελεγκτής CISC περιλαμβάνει πιο περίπλοκες εντολές, με συνέπεια να εκτελεί πιο πολύπλοκες διαδικασίες. Ωστόσο, όσο μεγαλύτερο είναι το πλήθος του συνόλου των εντολών, τόσο πιο πολύπλοκη είναι η σχεδίαση του, με περισσότερη καθυστέρηση για την εκτέλεση της κάθε εντολής [7][24].

Από την άλλη πλευρά, η αρχιτεκτονική Harvard αποτελείται από ξεχωριστές μνήμες για το πρόγραμμα και για τα δεδομένα. Οι εντολές εκτελούνται με γρηγορότερο ρυθμό, λόγο του ότι υπάρχουν δύο μοναδικοί δίαυλοι που επικοινωνούν με την ΚΜΕ, καθιστώντας το αρκετά πιο αποδοτικό. Η εκτέλεση μιας εντολής χρειάζεται λιγότερους κύκλους μηχανής. Ο μικροελεγκτής σε αυτήν την περίπτωση αποτελεί μέρος των Υπολογιστών Μειωμένου Ρεπερτορίου Εντολών (RISC). Η κατηγορία RISC παρέχει μικρό σύνολο εντολών, οι οποίες εκτελούνται ταχύτατα. Έτσι, η σχεδίαση του μικροελεγκτή απλοποιείται σημαντικά, ώστε οι εντολές να εκτελούνται αρκετά πιο γρήγορα σε σχέση με τους CISC [7][24].

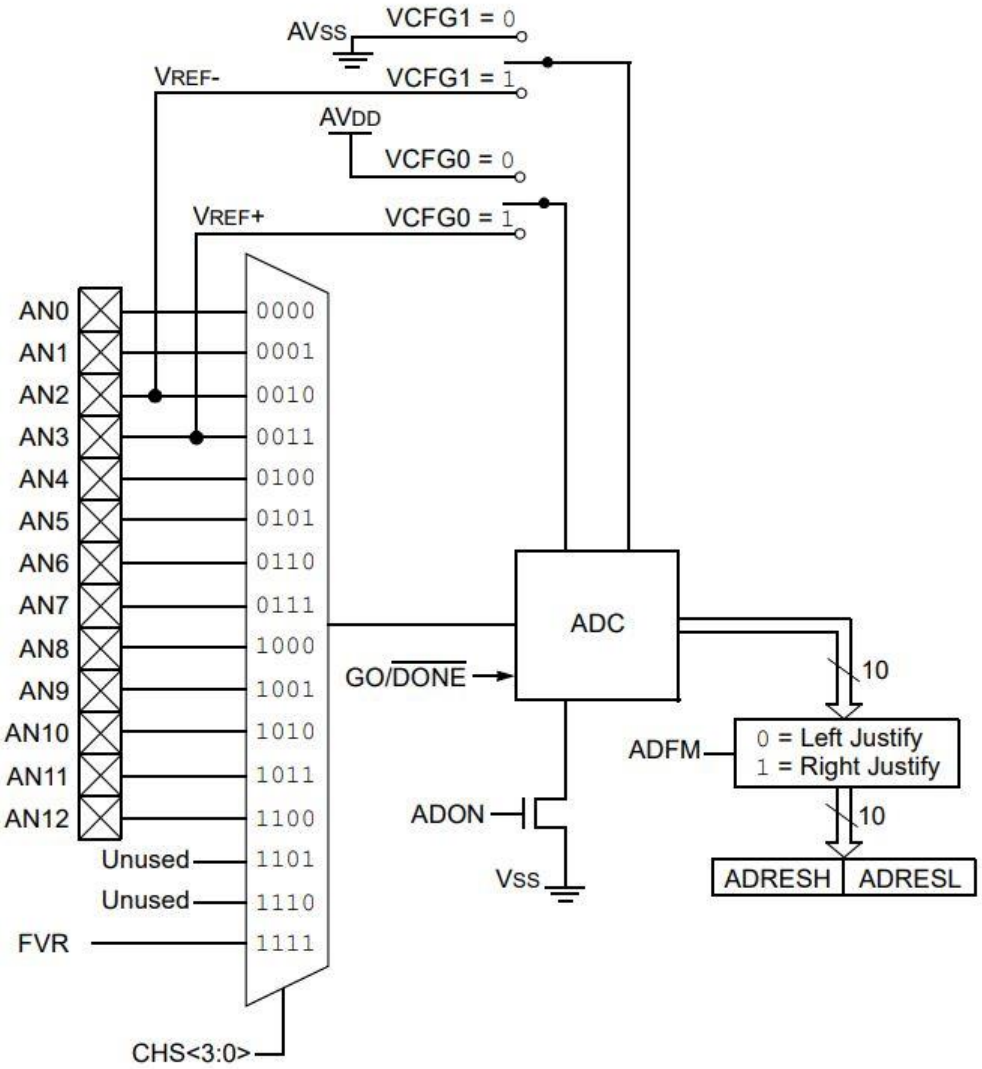



Σχήμα 2.3 Αρχιτεκτονική Von Neumann


Σχήμα 2.4 Αρχιτεκτονικής Harvard

## 2.3 Μετατροπή Αναλογικού Σήματος σε Ψηφιακό (A/D Converter)

Ο μετατροπέας από αναλογικό σε ψηφιακό σήμα, επιτρέπει την τροποποίηση ενός αναλογικού σήματος σε ψηφιακό, για να μπορέσει να το επεξεργαστεί κατάλληλα ο μικροελεγκτής. Επί της ουσίας, το ψηφιακό σήμα αντιπροσωπεύει μια προσεγγιστική τιμή των 10 bit. Το αποτέλεσμα αποθηκεύεται στους δύο καταχωρητές του μετατροπέα ADC (ADRESL και ADRESH). Η διαδικασία μετατροπής αρχίζει μόλις ο πολυπλέκτης επιλέξει το σήμα εισόδου που θέλει να διαβάσει. Στην συνέχεια γίνεται επεξεργασία για να ληφθεί η τελική μετατροπή, διαβάζοντας τα ψηφιακά δεδομένα που προκύπτουν. Δέχεται μονοπολική τάση εισόδου από 0V έως 5V, αποτελείται από 13 κανάλια αναλογικής εισόδου και από πέντε καταχωρητές. Οι A/D έχουν πεπερασμένο Χρόνο Προσπέλασης (Conversion Time), καθώς είναι ο χρόνος που απαιτείται για την μετατροπή της αναλογικής τάσης σε ψηφιακή τιμή στην έξοδο. Επιπλέον, έχει συγκεκριμένη Διακριτική Ικανότητά (Resolution), όπου είναι ο αριθμός των bits στην έξοδο του. Το βήμα κβάντισης είναι πάντα πιο μικρό, όταν είναι μεγαλύτερη η διακριτική ικανότητα. Οπότε έχει μεγαλύτερη Ακρίβεια (Accuracy) από την καταγραφή της δειγματοληψίας [12] [2].



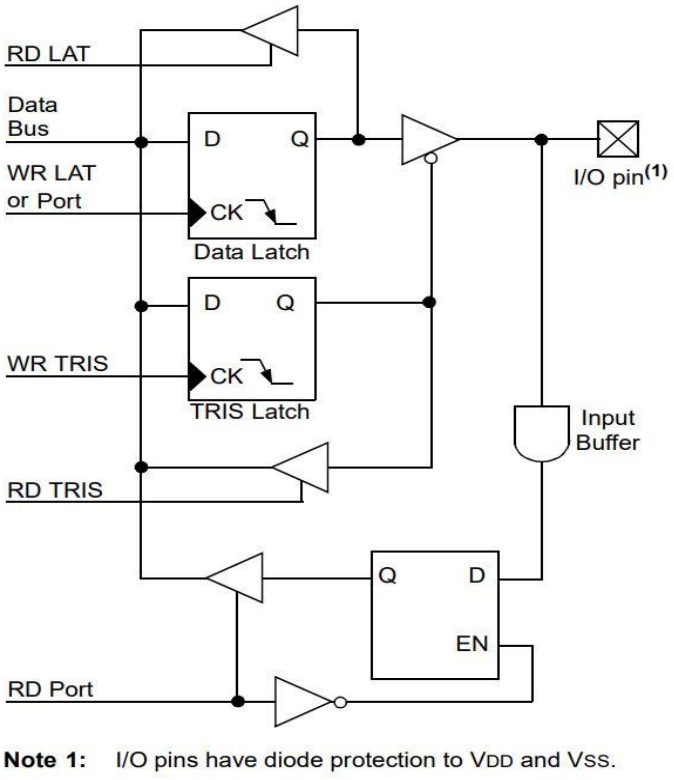

Σχήμα 2.5 Μπλοκ Διάγραμμα Μετατροπέα ADC [12]

Ο μετατροπέας A/D έχει δομηθεί με τους εξής καταχωρητές [12]:

1. A/D Result High Register (ADRESH)
2. A/D Result Low Register (ADRESL)
3. A/D Control Register 0 (ADCON0)
4. A/D Control Register 1 (ADCON1)
5. A/D Control Register 2 (ADCON2)

## 2.4 Πόρτες Εισόδου/Εξόδου (Port I/O)

Οι πόρτες εισόδου/εξόδου είναι καταχωρητές δεδομένων PORTX διπλής κατεύθυνσης, που σημαίνει ότι ο κάθε ακροδέκτης ορίζεται είτε σαν είσοδο, είτε σαν έξοδο από τον καταχωρητή κατεύθυνσης TRISX. Ο καταχωρητής δεδομένων εμφανίζει τα δεδομένα του στην έξοδο της πόρτας, δηλαδή στους ακροδέκτες. Έχει τη δυνατότητα να διαβάζει από τη θύρα, είτε να γράψει σε αυτή λογικό “0” ή “1”. Με τον καταχωρητή κατεύθυνσης ορίζεται η συμπεριφορά των ακροδεκτών. Εφόσον είναι μονάδα κάποιο bit στον καταχωρητή κατεύθυνσης, τότε ο αντίστοιχος ακροδέκτης της πόρτας θεωρείται είσοδος. Αν το bit είναι μηδέν τότε ο ακροδέκτης είναι έξοδος. Ο PIC απαρτίζεται από τέσσερις 8 bit θύρες και μια 4 bit: PORTA, PORTB, PORTC, PORTD και PORTE (4 bit). Οι αντίστοιχοι καταχωρητές κατεύθυνσης: TRISA, TRISB, TRISC, TRISD και TRISE. Η κάθε πόρτα διαθέτει τις δικές της ικανότητες [12][2].




Σχήμα 2.6 Γενικό Κύκλωμα Λειτουργίας Πόρτας Εισόδου/Εξόδου [12]

| Pin | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RB0/INT0/FLT0/ AN12 | RB0 | 0 | O | DIG | LATB<0> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTB<0> data input; Programmable weak pull-up. Disabled when analog input enabled.(1) |
| | INT0 | 1 | I | ST | External interrupt 0 input. |
| | FLT0 | 1 | I | ST | Enhanced PWM Fault input (ECCP1 module); enabled by software. |
| | AN12 | 1 | I | ANA | A/D input channel 12.(1) |
| RB1/INT1/AN10/ C12IN3-/P1C | RB1 | 0 | O | DIG | LATB<1> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTB<1> data input; Programmable weak pull-up. Disabled when analog input enabled.(1) |
| | INT1 | 1 | I | ST | External Interrupt 1 input. |
| | AN10 | 1 | I | ANA | ADC input channel 10.(1) |
| | C12IN3- | 1 | I | ANA | Comparators C1 and C2 inverting input, channel 3. Analog select is shared with ADC. |
| | P1C | 0 | O | DIG | ECCP PWM output (28-pin devices only). |
| RB2/INT2/AN8/ P1B | RB2 | 0 | O | DIG | LATB<2> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTB<2> data input; Programmable weak pull-up. Disabled when analog input enabled.(1) |
| | INT2 | 1 | I | ST | External interrupt 2 input. |
| | AN8 | 1 | I | ANA | ADC input channel 8.(1) |
| | P1B | 0 | O | DIG | ECCP PWM output (28-pin devices only). |
| RB3/AN9/C12IN2-/ CCP2 | RB3 | 0 | O | DIG | LATB<3> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTB<3> data input; Programmable weak pull-up. Disabled when analog input enabled.(1) |
| | AN9 | 1 | I | ANA | ADC input channel 9.(1) |
| | C12IN2- | 1 | I | ANA | Comparators C1 and C2 inverting input, channel 2. Analog select is shared with ADC. |
| | CCP2(2) | 0 | O | DIG | CCP2 compare and PWM output. |
| | | 1 | I | ST | CCP2 capture input |
| RB4/KBI0/AN11/ P1D | RB4 | 0 | O | DIG | LATB<4> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTB<4> data input; Programmable weak pull-up. Disabled when analog input enabled.(1) |
| | KBI0 | 1 | I | TTL | Interrupt-on-pin change. |
| | AN11 | 1 | I | ANA | ADC input channel 11.(1) |
| | P1D | 0 | O | DIG | ECCP PWM output (28-pin devices only). |
| RB5/KBI1/PGM | RB5 | 0 | O | DIG | LATB<5> data output. |
| | | 1 | I | TTL | PORTB<5> data input; Programmable weak pull-up. |
| | KBI1 | 1 | I | TTL | Interrupt-on-pin change. |
| | PGM | x | I | ST | Single-Supply Programming mode entry (ICSP™). Enabled by LVP Configuration bit; all other pin functions disabled. |

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Configuration on POR is determined by the PBADEN Configuration bit. Pins are configured as analog inputs by default when PBADEN is set and digital inputs when PBADEN is cleared.

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is '0'. Default assignment is RC1.

**3:** All other pin functions are disabled when ICSP or ICD are enabled.

Σχήμα 2.7 Χαρακτηριστικές Ιδιότητες Πόρτας B [12]

Αναλύονται τα ειδικά χαρακτηριστικά της πόρτας Β παρακάτω:

- Η πόρτα Β έχει εσωτερικές pull-up αντιστάσεις για κάθε ακροδέκτη και συνηθέστερα προορίζονται να ενεργοποιηθούν όταν χρησιμοποιούνται εξωτερικοί διακόπτες. Προσφέρουν υψηλή λογική κατάσταση όταν είναι ενεργοποιημένες και με το κλείσιμο της επαφής του συνδεδεμένου διακόπτη, η είσοδος θα διαβάσει λογικό "0" [12].
- Τα τέσσερα πιο σημαντικά bit (MSB) είναι RB7, RB6, RB5, RB4. Διατίθεται μια λειτουργία εξωτερικής διακοπής KBIX κατά την αλλαγή της τιμής στις εισόδους τους,. Όμως, υπάρχουν και οι ξεχωριστές διακοπές INTX των RB0, RB1, RB2 που εκτελούν την ίδια διαδικασία με την διαφορά πως ορίζονται τα μέτωπα (αρνητικό ή θετικό) [12].
- Οι ακροδέκτες RB0, RB1, RB2, RB3, RB4, RB5 ρυθμίζονται ως αναλογικά κανάλια εισόδου τάσης για να γίνει δειγματοληψία. Τα κανάλια είναι τα εξής: AN12, AN11, AN10, AN9, AN8 [12].
- Το ποδαράκι RB3 με την μορφή του CCP2 μπορεί να παράγει παλμούς διαμορφωμένου εύρους (PWM) [12].

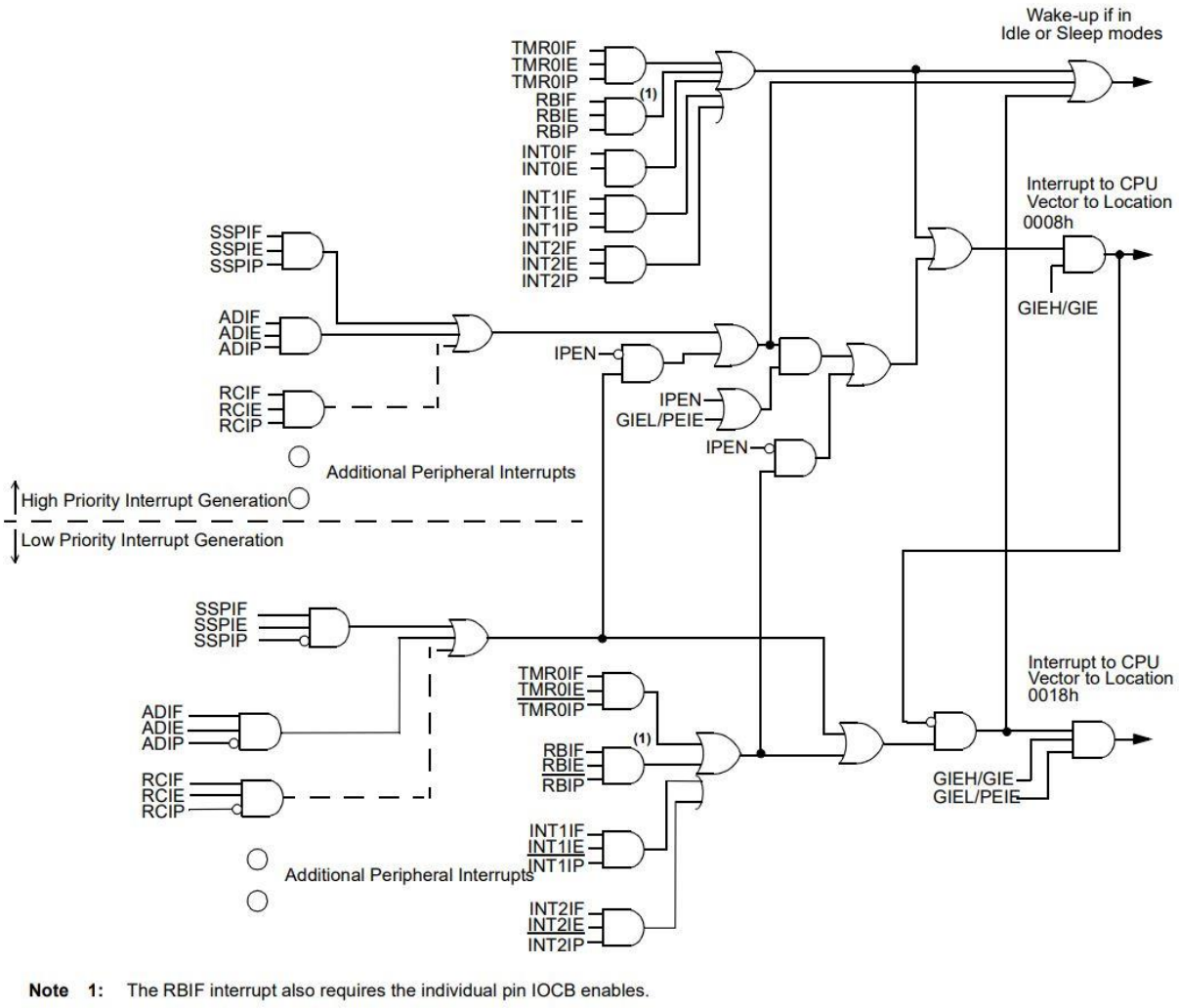
## 2.5 Διακοπές (Interrupts)

Η διακοπή είναι μια μέθοδος, με την οποία διακόπτεται η ροή της εκτέλεσης του τρέχοντος προγράμματος στον μικροελεγκτή, εξαιτίας κάποιου εσωτερικού ή εξωτερικού συμβάντος. Άμα συμβεί διακοπή λόγο κάποιου ακροδέκτη ή χρονιστή, τότε ο μικροελεγκτής έχει ως αποτέλεσμα να απαντήσει σε αυτή την αίτηση. Πραγματοποιείται αλλαγή διεύθυνσης του τρέχοντος προγράμματος σε μια συγκεκριμένη διεύθυνση της μνήμης, για να εκτελεστεί η υπορουτίνα ειδικής εξυπηρέτησης διακοπής. Μόλις ολοκληρωθεί αυτή η διαδικασία, τότε επαναφέρεται το πρόγραμμα στο σημείο από όπου έγινε η διακοπή [1][2].

Οι διακοπές χωρίζονται στις επικαλυπτόμενες (Maskable) και τις μη επικαλυπτόμενες (Non Maskable). Στην επικαλυπτόμενη διακοπή, ο χρήστης έχει την δυνατότητα να την απενεργοποιήσει μέσω του προγράμματος. Οι διακοπές που έχουν μάσκα, εκτελούνται με πιο γρήγορο ρυθμό και βοηθάνε στην διαχείριση εργασιών με μικρότερη προτεραιότητα (Lower Priority). Σε αντίθεση με την μη επικαλυπτόμενη διακοπή, δεν μπορεί να απενεργοποιηθεί και ο χρόνος απόκρισης είναι πιο αργός. Επίσης, προορίζονται για να ελέγχουν εργασίες με μεγαλύτερη προτεραιότητα (Higher Priority) [2].

Ο μικροελεγκτής PIC18F46K20 αποτελείται από 10 καταχωρητές που χρησιμοποιούνται για τον έλεγχο της λειτουργίας των διακοπών [12]:

- Reset Control Register (RCON)
- Interrupt Control Register (INTCON)
- Interrupt Control Register 2 (INTCON2)
- Interrupt Control Register 3 (INTCON3)
- Peripheral Interrupt Request (Flag) Register 1&2 (PIR1, PIR2)
- Peripheral Interrupt Enable (Flag) Register 1&2 (PIE1, PIE2)
- Peripheral Interrupt Priority Register 1&2 (IPR1, IPR2)



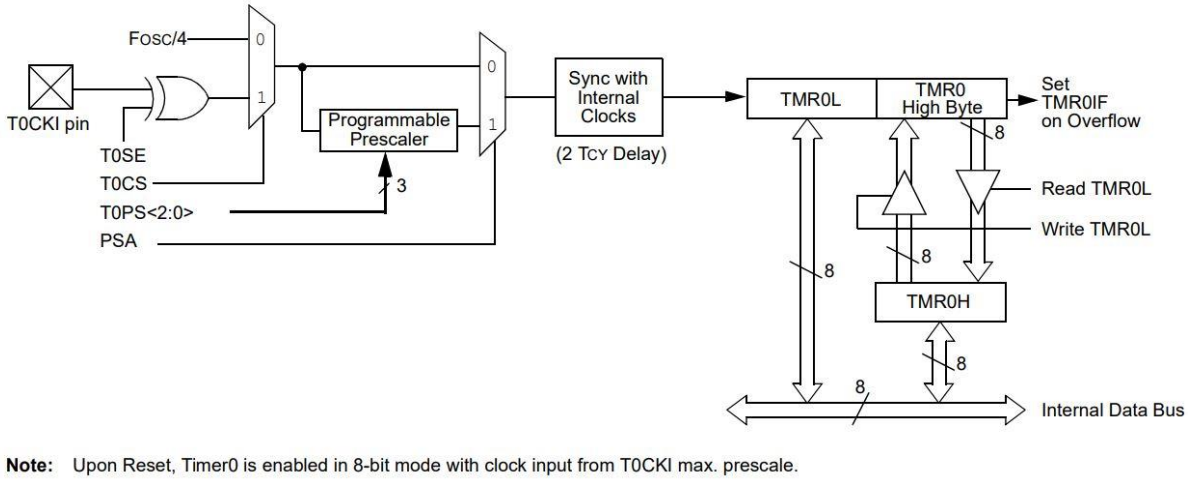

Σχήμα 2.8 Λογική Δομή του Συστήματος Διακοπής [12]

## 2.6 Χρονιστές (Timers)

Χρονιστές είναι ειδικευμένα περιφερειακά κυκλώματα, που αυξάνουν ή μειώνουν την τιμή ενός μετρητή, δημιουργώντας χρονικές καθυστερήσεις με τη συχνότητα του ρολογιού. Η τιμή του μετρητή μεγαλώνει ή μικραίνει κατά μια μονάδα σε κάθε κύκλο μηχανής και είναι αρκετά ευέλικτοι με την δυνατότητα προγραμματισμού διαιρέτη (Programmable Prescaler). Μια εξωτερική πηγή χρονισμού, χρησιμοποιείται για τον συγχρονισμό με την εσωτερική φάση του ρολογιού. Μερικοί χρονιστές λειτουργούν είτε σαν μετρητές (Counter), έτσι ώστε να αυξάνεται η τιμή τους με κάθε ανοδικό ή καθοδικό παλμό, από έναν ορισμένο εξωτερικό ακροδέκτη. Στον PIC18F46K20 υπάρχουν συνολικά οι παρακάτω 4 χρονιστές [1]:

- Timer 0 – Επιλογή για την λειτουργία του ως χρονιστή ή μετρητή παλμών σε 8 ή 16 bits.
- Timer 1 – Επιλογή για την λειτουργία του ως χρονιστή ή μετρητή παλμών σε 16 bits.
- Timer 3 – Επιλογή για την λειτουργία του ως χρονιστή ή μετρητή παλμών σε 8 bits ή 16 bits.

❖ Timer 2 - Επιλογή για την λειτουργία του ως χρονιστή σε 8 bits και για την παραγωγή παλμών διαμορφωμένου εύρους από το CCP.




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

Σχήμα 2.9 Μπλοκ Διάγραμμα του Timer0 [12]

## 2.7 Μνήμη Δεδομένων RAM

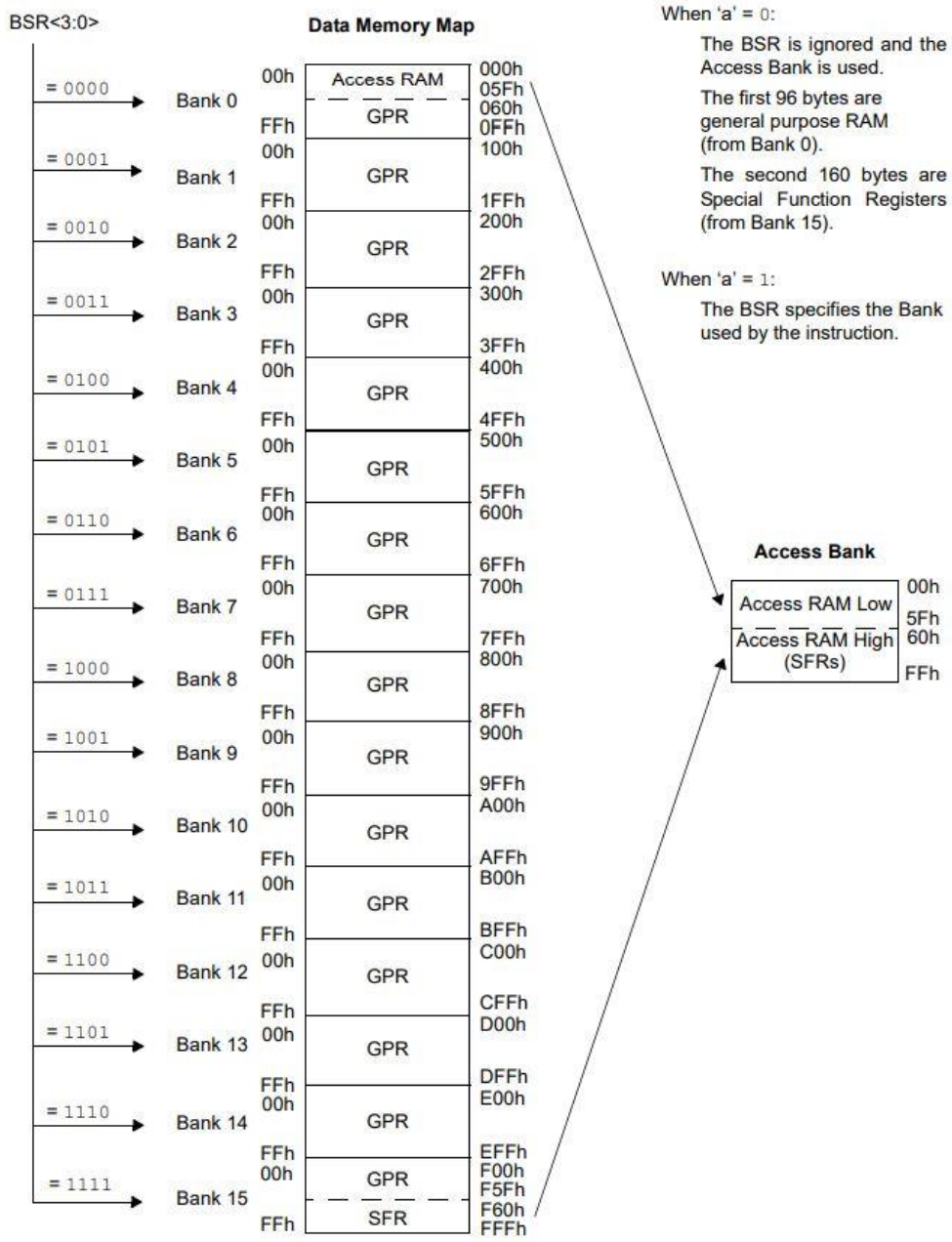
Η μνήμη δεδομένων στους PIC18 εφαρμόζονται ως στατική RAM. Γίνεται εγγραφή και ανάγνωση δεδομένων, σε οποιαδήποτε τυχαίο ή συγκεκριμένο μέρος διευθύνσεων, κατά την διάρκεια εκτέλεσης του προγράμματος. Είναι απαραίτητη για την αποθήκευση δεδομένων, όπως μεταβλητές για τον έλεγχο του προγράμματος. Κάθε καταχωρητής της μνήμης έχει μια διεύθυνση 12 bit, επιτρέποντας μέχρι και 4096 bytes προς αποθήκευση και οργανώνεται σε λέξεις των 8 bit. Ο χώρος της μνήμης χωρίζεται σε 16 τράπεζες (Banks) όπου η κάθε μια περιέχει 256 bytes. Το ρεπερτόριο εντολών και η αρχιτεκτονική, εξασφαλίζουν την δυνατότητα διεργασιών σε όλες τις τράπεζες. Ολόκληρη η μνήμη μπορεί να προσπελαστεί με τρόπο άμεσο (Direct Addressing), έμμεσο (Indirect Addressing) ή μέσω ευρετηρίου (Index Addressing) [12].

Επιπρόσθετα, η μνήμη προγράμματος προσφέρει θέσεις με καταχωρητές ειδικής λειτουργίας (Special Function Registers, SFRs) και γενικού σκοπού (General Function Registers, GFRs). Οι Ειδικοί SFR ρυθμίζουν την κατάσταση ελεγκτή και τον έλεγχό του. Οι γενικοί GFRs χρησιμοποιούνται για την καταχώρηση δεδομένων και για λειτουργίες υψηλής απόδοσης καταχωριστών της KME (Scratchpad Memory) [12].

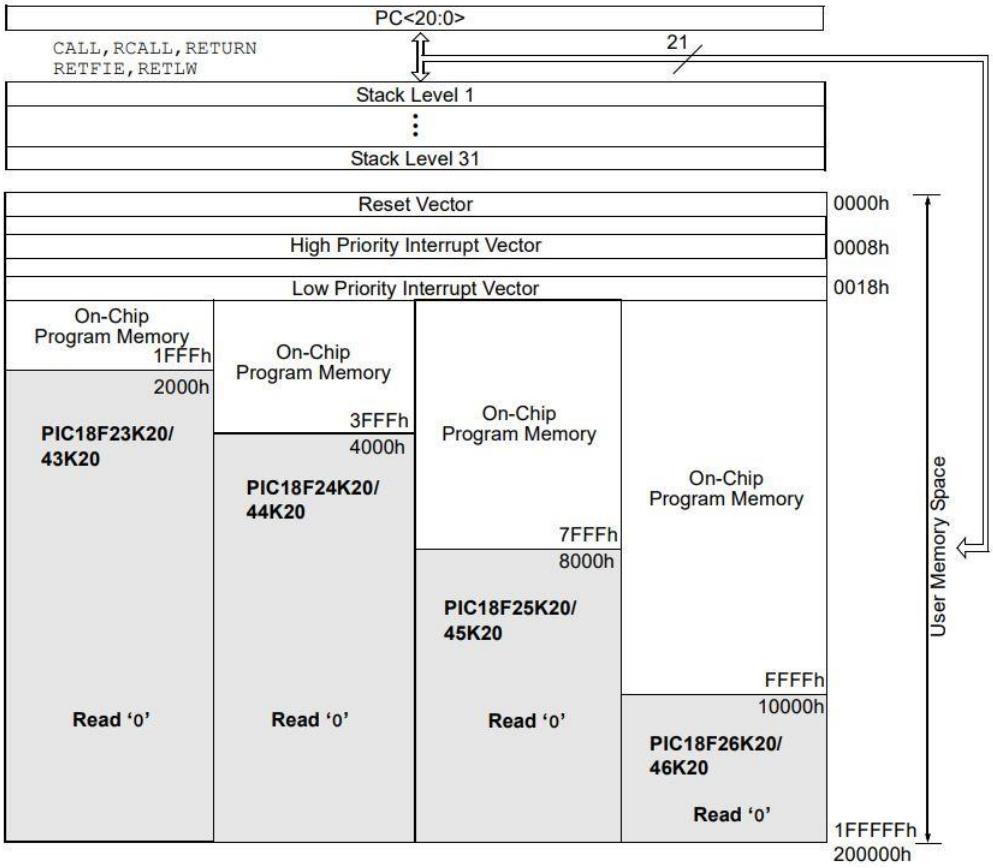
## 2.8 Μνήμη Προγράμματος ROM

Η μνήμη προγράμματος στους PIC18 αποτελεί τη μνήμη ROM. Επιτρέπεται μόνο η ανάγνωση δεδομένων από συγκριμένες θέσεις μνήμης, περιέχοντας αποθηκευμένες εντολές (Instructions), ώστε να πραγματοποιηθεί εκτέλεση του προγράμματος. Αφενός, πραγματοποιείται εγγραφή στην μνήμη του

μικροελεγκτή, όμως μόνο για την τοποθέτηση του νέου κώδικα (Code) και όχι κατά την διάρκεια που τρέχει το πρόγραμμα. Αρχικά, ο δίαυλος διευθύνσεων (Address Bus) είναι της τάξης των 21 bit και απευθύνεται σε εντολές, που οργανώνονται με κωδικοποιημένες λέξεις των 16 bit. Παρά τους 21 αγωγούς που διατίθενται από τον μετρητή προγράμματος (Program Counter), επικοινωνεί μονάχα από την διεύθυνση $0000_h$ έως τη διεύθυνση $FFFF_h$. Οι θέσεις μνήμης από τη $10000_h$ έως τη διεύθυνση $1FFFF_h$ δεν υφίστανται, δηλαδή ισχύει Read "0". Ο PIC18 διαθέτει δύο δείκτες διακοπής (Interrupt Vector), τον χαμηλής προτεραιότητας στη θέση $0018_h$ (Low Priority) και τον υψηλής προτεραιότητας (High Priority) στη διεύθυνση $0008_h$. Επίσης, έναν δείκτη επανεκκίνησης διεύθυνσης (Reset Vector Address) στη θέση $0000_h$. Τέλος, οι συνολικά διαθέσιμές θέσεις μνήμης είναι $2^{16}$=64K bytes [12].




Σχήμα 2.10 Διάγραμμα Μνήμης Δεδομένων RAM [12]




Σχήμα 2.11 Διάγραμμα Μνήμης Προγράμματος ROM [12]

## 2.9 Αριθμητική και Λογική Μονάδα (ALU)

Η Αριθμητική και Λογική Μονάδα (ALU) είναι υπεύθυνη για την εκτέλεση αριθμητικών και λογικών πράξεων. Μπορεί να κάνει πράξεις, όπως δυαδική πρόσθεση, αφαίρεση, τις λογικές πράξεις NOT, AND, OR, πολλαπλασιασμούς, διαιρέσεις, συμπλήρωμα ως προς 1 (NOT) και 2, ολίσθηση και περιστροφή. Πάντοτε υπάρχει ένας καταχωρητή εργασίας W (Work Register), ο οποίος είναι 8 bit και βρίσκεται μέσα στην Κεντρική Μονάδα Επεξεργασίας. Ο καταχωρητής W δύναται να του φορτωθεί μια δυαδική τιμή και στη συνέχεια να αποθηκευτεί σε μια θέση της μνήμης δεδομένων. Όταν πρέπει να εκτελεστεί μια εντολή με ένα όρισμα μόνο, μπορεί να γίνει είτε από τον καταχωρητή W είτε από οποιοδήποτε άλλο που το αφορά. Ολοκληρώνεται μια πράξη μεταξύ του καταχωρητή W, πάντα σε σχέση με κάποιον άλλον καταχωρητή. Σύμφωνα με την εντολή ή την πράξη που εκτελείται κάθε φορά από την ALU, ο καταχωρητής κατάστασης (Status Register) δίνει κάποιες πληροφορίες σχετικά με το αποτέλεσμα της τελευταίας ενέργειας, μαζί με κάποιες σημαίες (Flags) [1][2].

## 2.10 Συγκριτής (Comparator)

Ο Συγκριτής (Comparator) είναι ένα αναλογικό κύκλωμα, το οποίο συγκρίνει την τάση δύο σημάτων και εμφανίζει στην έξοδό του μια ψηφιακή στάθμη λογικού "0" ή "1". Πρέπει να τονιστεί πως οι είσοδοι

χωρίζονται στην μη αναστρέφουσα (Non-inverted) και την αναστρέφουσα (Inverted). Σε περίπτωση που η αναλογική τάση του ακροδέκτη $V_{IN+}$ γίνει μεγαλύτερη από την τάση του $V_{IN-}$, τότε το ψηφιακό επίπεδο της εξόδου θα είναι λογικό "1". Όμως, εάν η αναλογική τάση για το ποδαράκι $V_{IN+}$ γίνει μικρότερη από την τάση του $V_{IN-}$, αυτό θα έχει ως αποτέλεσμα το ψηφιακό επίπεδο της εξόδου να αλλάξει σε λογικό "0". Ο Συγκριτής περιλαμβάνει τα παρακάτω χαρακτηριστικά [12]:

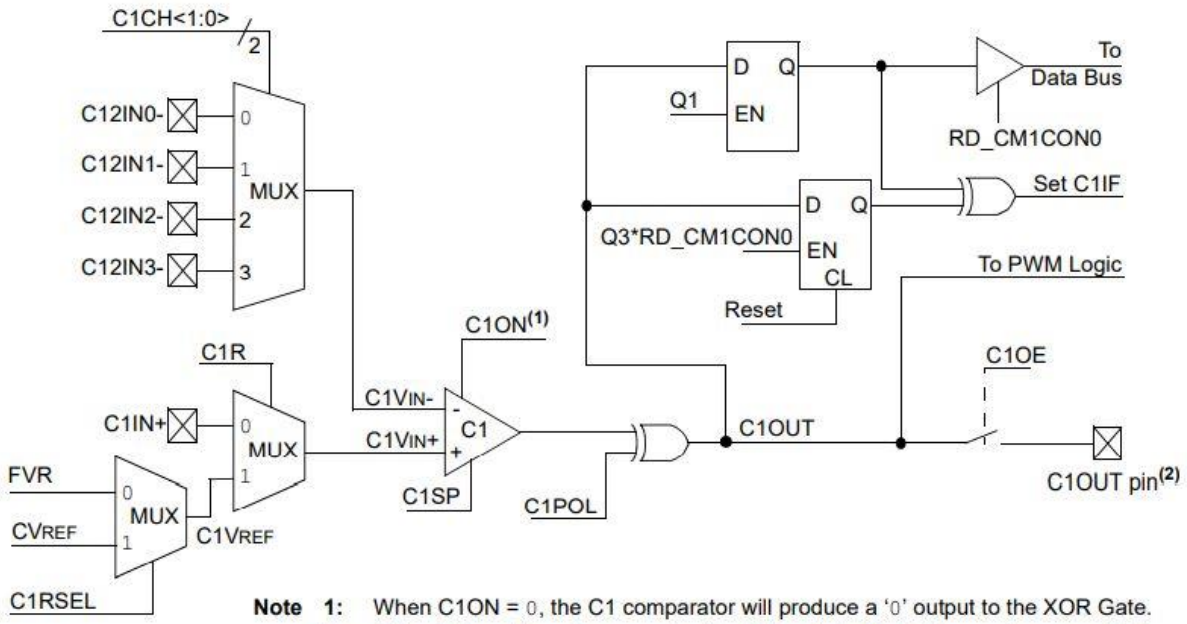
- Ανεξάρτητος Έλεγχος Συγκριτή
- Προγραμματιζόμενη Επιλογή Εισόδου
- Η Σύγκριση Εξόδου είναι διαθέσιμη Εξωτερικά/Εσωτερικά
- Προγραμματιζόμενη Πολικότητα Εξόδου
- Διακοπή Κατά την Αλλαγή
- Αφύπνιση από τον Ύπνο
- Προγραμματιζόμενη Βελτιστοποίηση Ταχύτητας/Ισχύος
- Απενεργοποίηση PWM
- Προγραμματιζόμενη και Σταθερή Αναφορά Τάσης

## 2.11 Καταχωρητές (Registers)

Οι Καταχωρητές (Registers) είναι εκείνοι που απαρτίζουν ολόκληρο τον μικροελεγκτή PIC, παρέχοντας έναν γρήγορο τρόπο για τη συλλογή και την φύλαξη δεδομένων. Επίσης, χρησιμοποιούνται για την μετέπειτα επεξεργασία των λειτουργιών, που θέλουν να επισπεύσουν στο μικροϋπολογιστικό τους σύστημα. Επειδή είναι μεγάλο το πλήθος των καταχωριτών, θα αναλυθούν 2 από τους πιο σημαντικούς.

Ο Καταχωρητής Προγράμματος (Program Counter), καθορίζει τη διεύθυνση της εντολής που πρόκειται να ανακτήσει ώστε να εκτελεστεί. Ο καταχωρητής προγράμματος είναι 21 bit και χωρίζεται σε 3 ξεχωριστούς καταχωρητές των 8 bit. Ο πρώτος είναι low byte, δηλαδή ο PCL καταχωρητής στον οποίο γίνεται εγγραφή και ανάγνωση (8 bit). Ο δεύτερος είναι high byte, με την ονομασία PCH εμπεριέχοντας 15:8 bits (8 bits) και δεν είναι απευθείας αναγνώσιμος, εγγράψιμος. Ανανεώνονται τα δεδομένα του PCH, μέσω του καταχωρητή PCLATH. Ο τρίτος είναι upper byte, καλούμενος ως PCU και διαθέτει 20:16 bits (4 bits). Επίσης, ο PCU δεν έχει απευθείας πρόσβασης για εγγραφή, ανάγνωση και χρησιμοποιείται ο καταχωρητής PCLATU για να πραγματοποιεί αυτές τις ενέργειες [12].

Ο Καταχωρητής Κατάστασης (Status Register), περιλαμβάνει αριθμητικές πληροφορίες στατιστικών, που λήφθηκαν από την Αριθμητική και Λογική Μονάδα. Εφόσον βρεθεί το αποτέλεσμα μιας εντολή μέσα στο καταχωρητή, θα επηρεαστούν οι τιμές μερικών σημαιών (flags). Το αρνητικό bit (Negative bit, N), γίνεται λογικό "0" αν το αποτέλεσμα της πράξης ήταν θετικό, στην περίπτωση του αρνητικού θα γίνει λογικό "1". Αν προκύψει υπερχείλιση στο αριθμητικό αποτέλεσμα (signed), η σημαία υπερχείλισης (Overflow bit, OV) θα είναι λογικό "1", αλλιώς θα παραμείνει λογικό "0". Άμα μια εντολή καταλήξει με μηδενικό αποτέλεσμα, τότε η σημαία μηδενισμού (Zero bit, Z) είναι λογικό "1", διαφορετικά λογικό "0". Τέλος, η σημαία κρατουμένου δίνει λογικό "0", όταν δεν υπάρχει κρατούμενο από πρόσθεση ή δανεικό από αφαίρεση, αλλιώς βγάζει λογικό "1". Η παραπάνω σημαία, ισχύει για το Digit Carry/Borrow του τέταρτου λιγότερου σημαντικού bit αποτελέσματος και για το Carry/Borrow του πιο σημαντικού bit (MSB) [12].




**Note 1:** When C1ON = 0, the C1 comparator will produce a '0' output to the XOR Gate.
**2:** Output shown for reference only. See I/O port pin block diagram for more detail.
**3:** Q1 and Q3 are phases of the four-phase system clock (FOSC).
**4:** Q1 is held high during Sleep mode.

Σχήμα 2.12 Δομή Διαγράμματος Συγκριτή C1 [12]

| U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | N | OV | Z | DC(1) | C(1) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (two's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (two's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

**Note 1:** For $\overline{\text{Borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high-order or low-order bit of the source register.

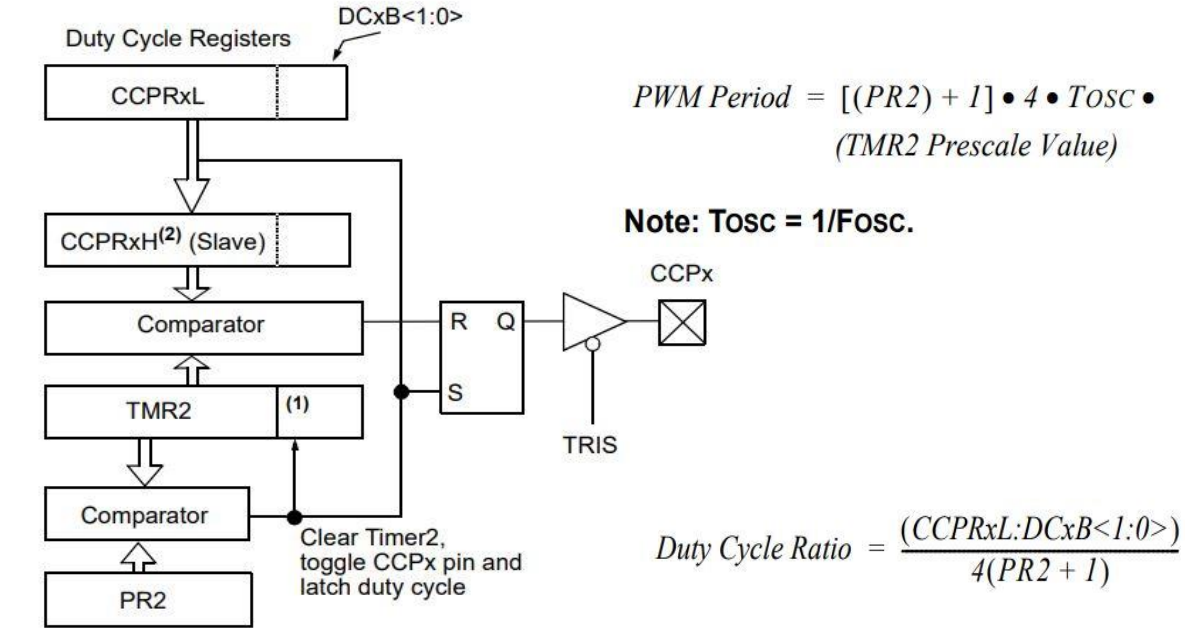
Σχήμα 2.13 Σχέδιο Καταχωρητή Κατάστασης (Status Register) [12]

## 2.12 Καταγραφή, Σύγκριση και Διαμόρφωση Εύρους Παλμών (CCP)

Η μονάδα Καταγραφής, Σύγκρισης και Διαμόρφωση Εύρους Παλμών (CCP), εργαλειοποιεί τους Timers 1, 2, 3, αναλόγως με τη λειτουργία που επιλέγεται. Οι Timer 1 & 3 είναι διαθέσιμοι στην Σύλληψη ή Σύγκριση, ενώ ο Timer 2 διατίθεται για την Διαμόρφωση Εύρους Παλμού. Ο μικροελεγκτής έχει δύο ανεξάρτητα CCP Modes (CCP1, CCP2) και μπορούν να ενεργοποιηθούν ταυτόχρονα, μοιράζοντας από κοινού τον ίδιο Timer.

Η Λειτουργία Καταγραφής (Capture), αναλαμβάνει την καταγραφή μιας δυαδικής τιμής των 16 bit, από τους καταχωρητές του Timer 1 ή 3, τοποθετώντας τα στους δύο καταχωρητές CCPRXH, CCPRXL. Προβαίνει σε ανάγνωση από τον ακροδέκτη CCPX, για κάθε παλμό χαμηλής ή υψηλής στάθμης. Επίσης, ανά κάθε 4 ή 16 παλμούς χαμηλής στάθμης. Αποφασίζεται πιο πριν η παραπάνω επιλογή, από τα bits CCPXM<3:0> του καταχωρητή CCPXCON. Μόλις γίνει μια σύλληψη, η σημαία CCPRX παίρνει τιμή λογικό "1" και μπορεί να μηδενιστή το bit μέσω του λογισμικού. Εφόσον καταγραφή καινούργια τιμή στο καταχωρητή CCPRX, η παλιά θα αντικατασταθεί από την καινούργια [12].

Στην Λειτουργία Σύγκρισης (Compare) ο καταχωρητής CCPRX των 16 bit, συγκρίνει συνέχεια μια προκαθορισμένη τιμή που έχει, με το περιεχόμενο των καταχωριτών Timer 1 ή 3. Όταν προκύψει ο κοινός αριθμός μεταξύ αυτών, τότε ο ακροδέκτης CCPX μπορεί να οδηγήσει την έξοδό του, σε υψηλή στάθμη, χαμηλή στάθμη, αναστρέψιμή λογική ή να παραμείνει αμετάβλητη. Η δράση του ακροδέκτη ορίζεται από τα bits στο CCPXM<3:0> και ταυτόχρονα η σημαία CCPXIF bit αποκτάει λογικό "1" [12].




**Note 1:** The 8-bit timer TMR2 register is concatenated with the 2-bit internal system clock (FOSC), or 2 bits of the prescaler, to create the 10-bit time base.

**2:** In PWM mode, CCPRxH is a read-only register.

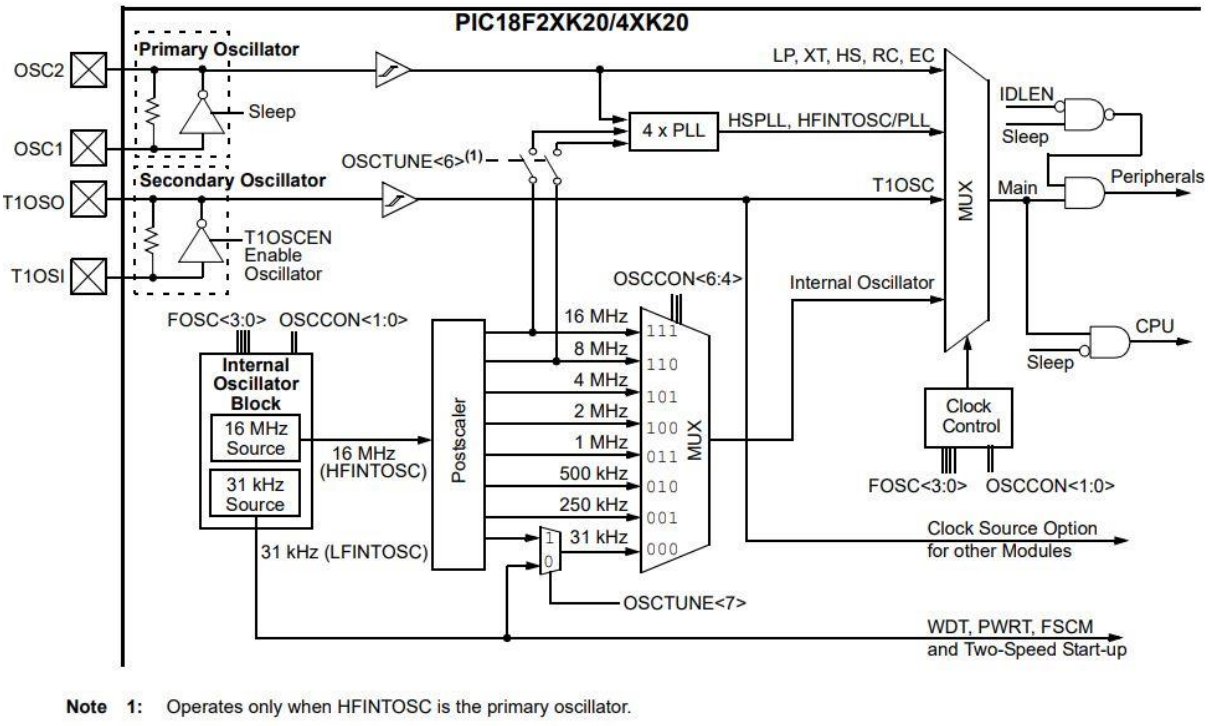
Σχήμα 2.14 Μπλοκ Διάγραμμα PWM και Φόρμουλες Υπολογισμού [12]

Η Λειτουργία PWM, αναπαράγει σήματα Διαμορφωμένου Εύρους Παλμών, δηλαδή είναι μια μέθοδος που πραγματοποιεί ψηφιακή διαμορφωμένη παλμών στην έξοδο της, για να ελέγχει αναλογικές συσκευές. Στα ποδαράκι CCP1, CCP2, παράγει μέχρι και 10 bit ανάλυση PWM σήματος. Ο καταχωρητής κατεύθυνσης TRIS, πρέπει να είναι μηδενισμένος για εκείνους τους ακροδέκτες, ώστε να λειτουργήσουν ως έξοδοι. Το PWM σήμα, βασίζεται σε μια σταθερή περίοδο (Period) και έναν κύκλο εργασίας, που μεταβάλλεται ως προς το χρόνο (Duty Cycle). Η περίοδος του σήματος, καθορίζεται από τον καταχωρητή PR2 του Timer 2 και υπολογίζεται από ειδική φόρμουλα [12].

## 2.13 Μονάδα Ταλαντωτή (Oscillator Module)

Η Μονάδα Ταλαντωτή (Oscillator Module), έχει μια μεγάλη ποικιλία από χαρακτηριστικά για τη ρύθμιση του ρολογιού, μεγιστοποιώντας την απόδοση και ελαχιστοποιώντας την καταναλώμενη ενέργεια. Ως πηγή ρολογιού, ορίζεται ένα εξωτερικό κύκλωμα (External Clock Source), δομημένο από ταλαντωτή των 16MHz, με δύο κεραμικούς πυκνωτές 22pF. Ενεργοποιείται η λειτουργία HSPLL, για να μπορέσει να δουλέψει με συχνότητες μέχρι τα 16 MHz. Μέσα από ένα Βρόγχο Κλειδωμένης Φάσης (Phase-Locked Loop), πολλαπλασιάζεται η συχνότητα εξόδου του ταλαντωτή κατά 4 φορές, για να παραχθεί εσωτερική συχνότητα ρολογιού 64 MHz. Το PLL είναι αρκετά χρήσιμο, γιατί δίνει την δυνατότητα στο χρήστη, να εκμεταλλευτεί ταλαντωτή χαμηλότερης συχνότητας [12].

Συμπληρωματικά, υπάρχει και η λειτουργία εσωτερικής πηγής ρολογιού (Internal Clock Source), όπου εμπεριέχεται μέσα στο μπλοκ ταλαντωτή. Χωρίζεται σε 8 διαφορετικές συχνότητες, επιλέγοντας μια από τις παρακάτω: 31KHz, 250KHz, 500KHz, 1MHz, 2MHz, 4MHz, 8MHz, 16MHz [12].

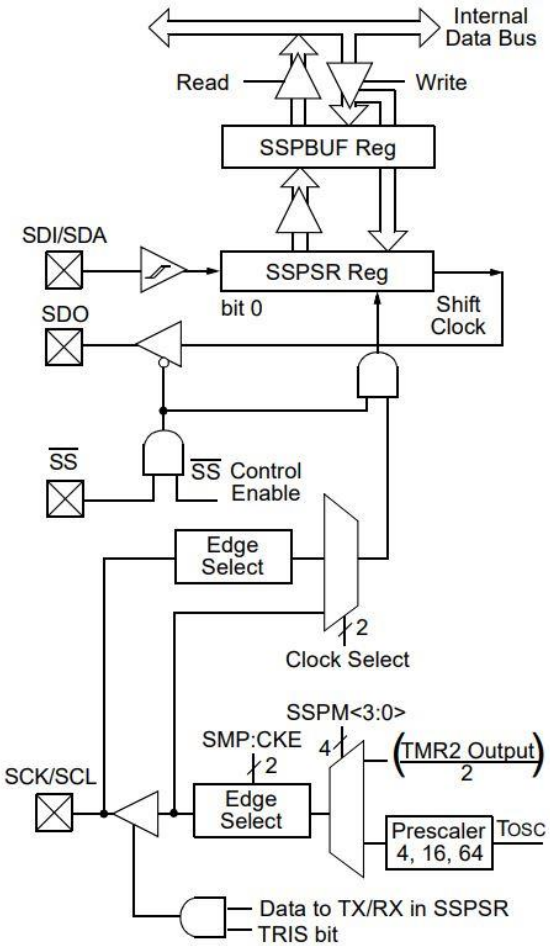



Σχήμα 2.15 Μπλοκ Διάγραμμα Πηγής Ρολογιού [12]

## 2.14 Μονάδα Σύγχρονης Σειριακής Θύρας (MSSP)

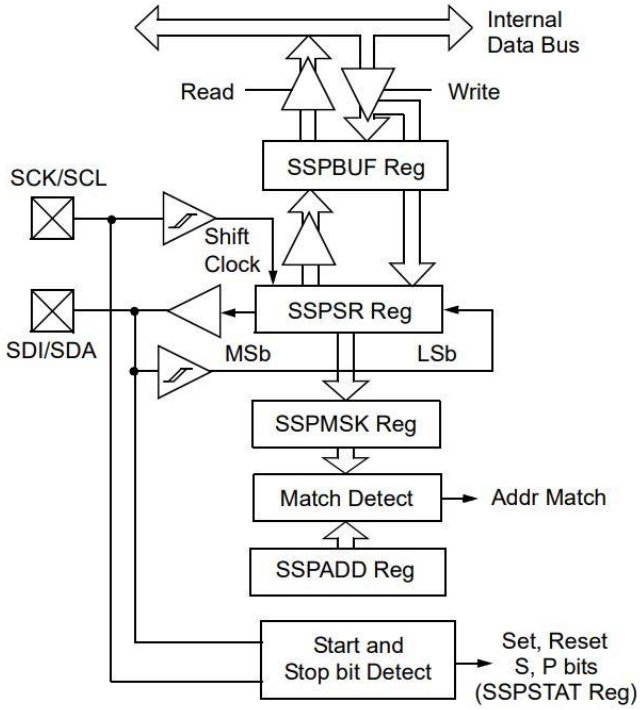
Η Μονάδα Σύγχρονης Σειριακής Θύρας (MSSP), είναι μια μονάδα σειριακής διεπαφής, για την επικοινωνία με άλλες περιφερειακές συσκευές ή μικροελεγκτές. Μερικά από αυτά τα παραδείγματα, είναι η μνήμη EEPROM, καταχωρητές ολίσθησης, οδηγοί οθονών, A/D Μετατροπής, SD Κάρτες, USB συσκευές, αισθητήρες και τα λοιπά. Το MSSP λειτουργεί με έναν από τους δυο τρόπους: Σειριακή Διεπαφή Περιφερειακών (SPI) ή Μέσω-Ολοκληρωμένου Κυκλώματος ($I^2C$) [2].

Αρχικά, ο δίαυλος SPI δημιουργήθηκε από την αμερικάνικη εταιρία Motorola Inc. Επιτρέπει την ταυτόχρονη μεταφορά δεδομένων, για 8 bit με σύγχρονη μετάδοση και λήψη. Κάνει χρήση δύο σημάτων ελέγχου SS, SCL και δύο σήματα δεδομένων SDO, SDI. Ο ακροδέκτης Slave Select (SS) είναι μια γραμμή ελέγχου, που χρησιμοποιείται για την επιλογή μιας εξωτερικής συσκευής (Slave), ώστε να γίνει ανταλλαγή δεδομένων με τη κύρια (Master). Το ποδαράκι Serial Clock (SCL), είναι υπεύθυνο για την κατάλληλη παραγωγή χρονισμού, μεταφέροντας δεδομένα μεταξύ του Master και του Slave. Από το Serial Data Out (SDO) του μικροελεγκτή, ο Master είναι εκείνους που μεταδίδει δεδομένα, δηλαδή σήμα προς τον Slave. Αντιστοίχως, η ακίδα Serial Data In θεωρείται από τον Master, είσοδος των εξερχόμενων δεδομένων που στέλνονται από τον Slave [1][2].




Σχήμα 2.16 Μπλοκ Διάγραμμα Συστήματος SPI [12]

Όσον αφορά το σειριακό δίαυλο I²C, επινοήθηκε από την ολλανδική εταιρία Philips και χρησιμοποιείται για την σύγχρονη επικοινωνία περιφερειακών συσκευών. Σε αντίθεση με το SPI, απαιτούνται μόνο δύο ακροδέκτες για να επιτευχθεί επικοινωνία στο I²C. Το ποδαράκι SDA (Serial Data), επιτρέπει την αμφίδρομη μεταφορά δεδομένων για τον Master και τον Slave. Η ακίδα SCL (Serial Clock), είναι η πηγή παλμών ρολογιού για τον συγχρονισμό των bit που στέλνονται ή λαμβάνονται. Είναι χαρακτηριστικό, πως τα εξαρτήματα που συνδέονται στον δίαυλο, αποκτάνε μια μοναδική διεύθυνση το καθένα. Διαθέτει έναν ειδικό αλγόριθμο για την ανίχνευση σύγκρουσης και επιφύλαξης. Με άλλα λόγια, ελέγχεται αν αλλοιωθούν τα δεδομένων από την ταυτόχρονη μετάδοση, λόγο των δυο Master ή παραπάνω [1][2].




Σχήμα 2.17 Μπλοκ Διάγραμμα Συστήματος I²C [12]

# Κεφάλαιο 3ο: Ανάλυση Λειτουργίας Συστήματος Crobarm

## 3.1 Εισαγωγή στο Crobarm

Πρώτα απ' όλα, θα αναλυθεί συνοπτικά, από τι αποτελείται το ρομπότ και το χειριστήριο, διότι είναι η θεωρητική βάση για την περαιτέρω κατανόηση, ολόκληρης της λογικής του συστήματος. Επεξηγείται με σαφήνεια η χρησιμότητα που έχει το κάθε στοιχείο, σε συνεργασία με όλα τα υπόλοιπα, καθώς και το σκοπό που εξυπηρετεί. Η δομή που περιγράφεται είναι σημαντική για τη συνέχεια του επόμενου κεφαλαίου. Θα μελετηθούν τα υλικά με περισσότερες λεπτομέρειες, για τις δυνατότητες που προσφέρουν, αλλά και τα προβλήματα που μπορεί να εμφανίσουν.

Έπειτα, σειρά έχει το μενού, όπου είναι καθοριστικό για τον πλήρη έλεγχο ολόκληρου του ρομποτικού συστήματος. Θεωρείται κύρια προτεραιότητα για το χειριστή, να αναγνωρίζει τις ενέργειες που μπορεί να εκτελέσει ο κάθε εντολέας, ώστε να αλληλοεπιδρά με τους καταλόγους γρήγορα και αποδοτικά. Για αυτόν το λόγο, ερμηνεύεται επαρκώς από την εκκίνηση, μέχρι την εισαγωγή στο κύριο μενού, αλλά και την ανάλυση όλων των υπομενού. Σχολιάζεται, με διεξοδικό τρόπο το κάθε γραφικό σύμβολο, ειδική συμβουλή ή χρήσιμη πληροφορία που αναφέρεται.

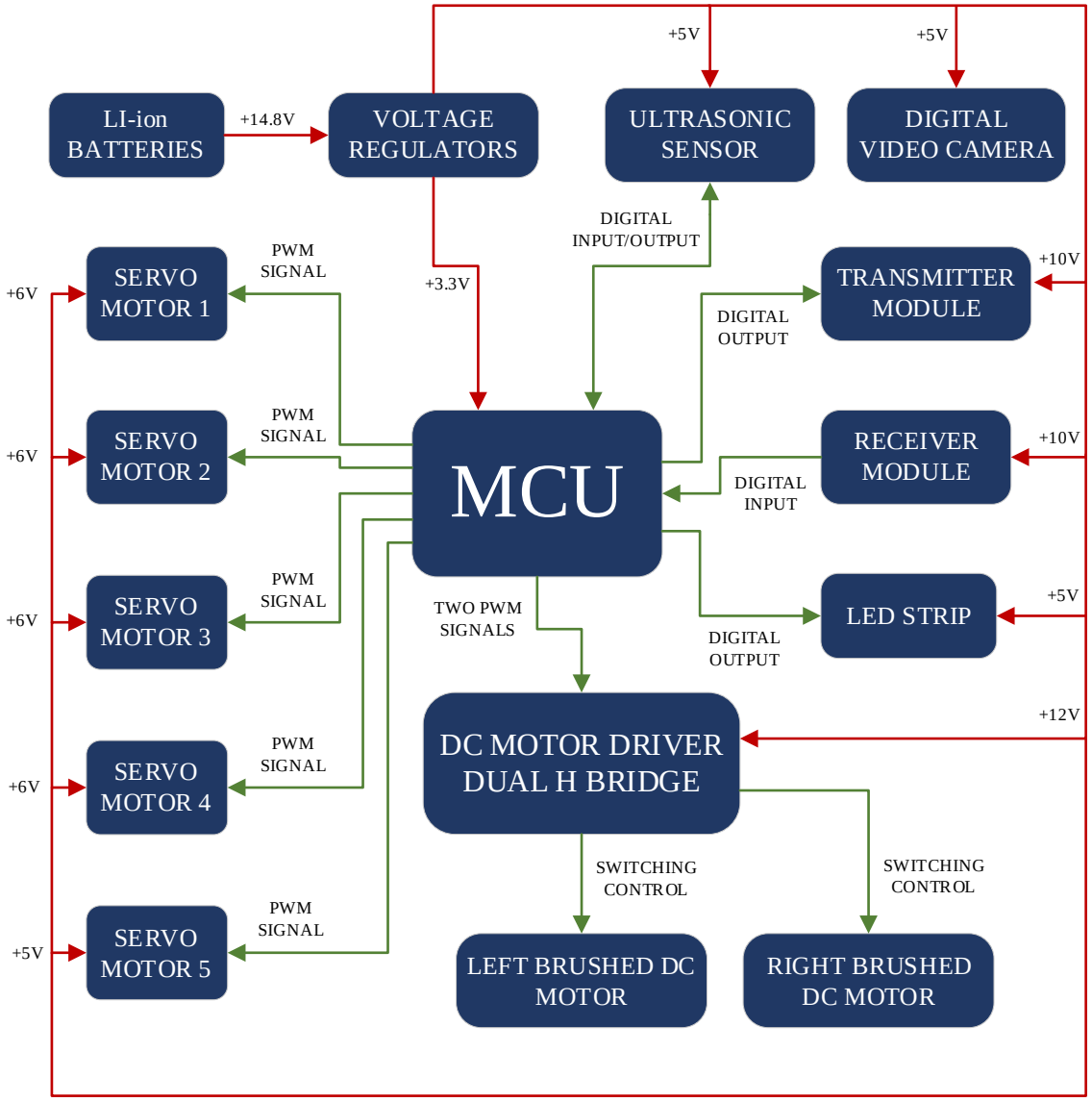
Ολοκληρώνοντας, στο τελευταίο μέρος διευκρινίζεται με μεθοδικότητα, το πως θα πρέπει να περιστρέφονται οι μοχλοί, σύμφωνα με την εργασία που είναι επιλέγεται. Υπάρχει συγκεκριμένο μοτίβο για να ελέγχεται το τανκ όπως και ο βραχίονας. Με βολικό τρόπο δε, ώστε να το μάθει σε λίγα μόλις λεπτά ο χρήστης. Οι μοχλοί του τηλεχειριστήριου, προγραμματίστηκαν σύμφωνα με τη φιλοσοφία που επικρατεί στα τηλεκατευθυνόμενα αυτοκίνητα, διαχειρίζοντάς το έτσι με περισσότερη οικειότητα.

## 3.2 Θεμελιώδη Οργάνωση Συστήματος

### 3.2.1 Βασική Δομή Ρομπότ

Το Crobarm διαθέτει μια ποικιλία υλικών και ηλεκτρονικών εξαρτημάτων, τα οποία είναι απαραίτητα για τη σωστή λειτουργεία του. Αρχικά, το σασί του μικρού οχήματος διαθέτει ερπύστριες, καθώς και ένα ρομποτικό βραχίονα σταθερής βάσης. Ένας μικροελεγκτής (MCU) είναι ο εγκέφαλος του όλου συστήματος, διότι ελέγχει, εκτελεί και επικοινωνεί συνέχεια με τις υπόλοιπες περιφερειακές μονάδες. Η ηλεκτρική πηγή ενέργειας με την οποία τροφοδοτείται, είναι οι επαναφορτιζόμενες μπαταρίες τύπου LI-ion. Επιπλέον, υπάρχουν μερικοί σταθεροποιητές τάσης (Voltage Regulators), για να εξασφαλίζουν ένα εύρος τιμών σταθερής τάσης.

Ο ρομποτικός βραχίονας αποτελείται από σκελετό, ο οποίος υποστηρίζει πέντε κινητήρες σέρβο. Λαμβάνοντας κατάλληλα σήματα PWM (Pulse Width Modulation), το καθένα από αυτά έχει τη δυνατότητα να περιστραφεί είτε δεξιόστροφα είτε αριστερόστροφα. Το ερπυστριοφόρο μπορεί να κινείται, χρησιμοποιώντας δύο κινητήρες συνεχούς ρεύματος με ψήκτρες (Brushed DC Motor). Ο οδηγός διπλής γέφυρας Η (Driver Dual H Bridge) διαβάζει τα σήματα PWM, οπότε έχει τη δυνατότητα να παρέχει και το κατάλληλο διακοπτικό έλεγχο (Control Switching) για τους κινητήρες.




Σχήμα 3.1 Μπλοκ Διάγραμμα Ρομπότ

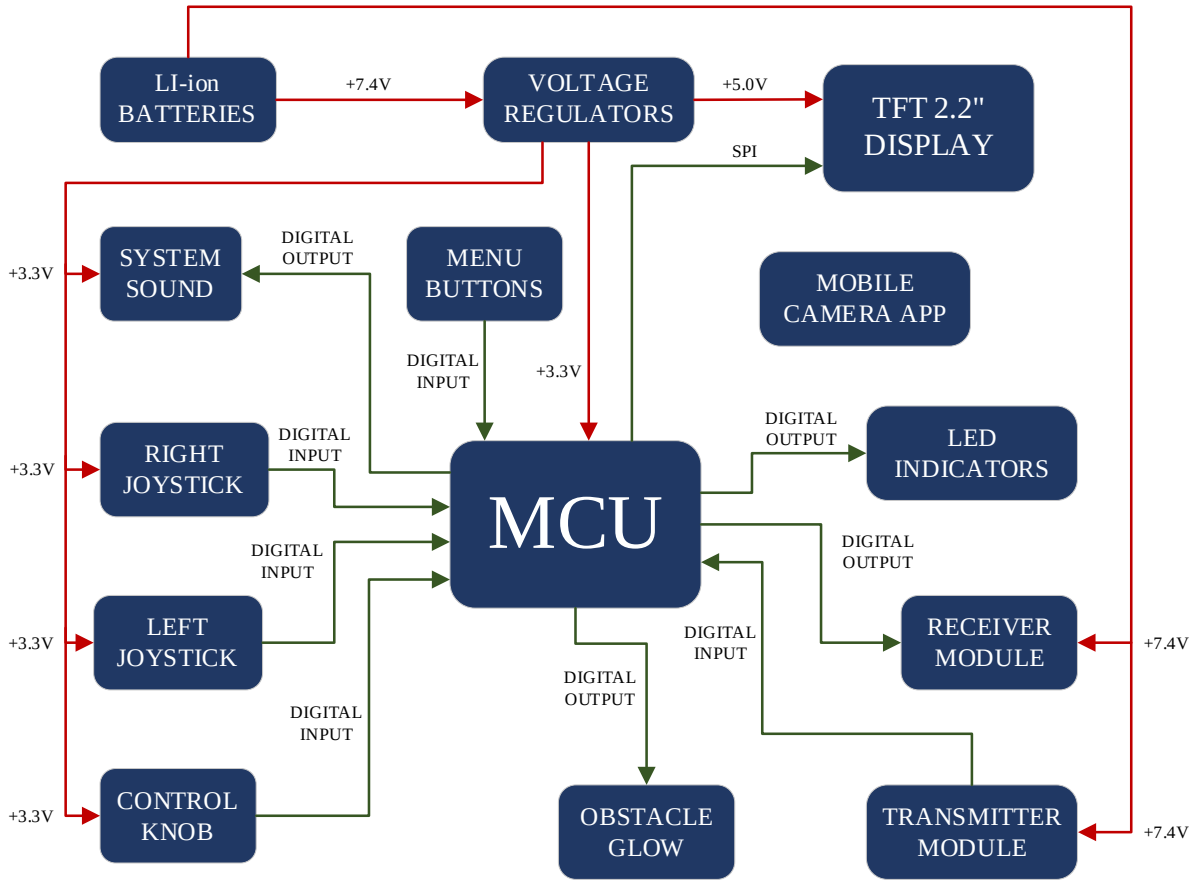
Το σύστημα πραγματοποιεί εκτεταμένη χρήση του αισθητήρα υπερήχων (Ultrasonic Sensor). Όταν δεχθεί έναν ψηφιακό παλμό, θα μετρήσει μια απόσταση από το εκάστοτε στόχο αντικείμενου και στην συνέχεια θα στείλει πίσω το τελικό ψηφιακό παλμό. Επιπλέον, φέρει μια ταινία LED (LED Strip), η οποία ελέγχεται από ένα ψηφιακό σήμα και υπάρχει ώστε να φωτίζει τη μπροστινή θέση, με αποτέλεσμα να προσαρμόζεται ακόμη και τη νύχτα. Σημαντική προσθήκη είναι και η ψηφιακή βιντεοκάμερα (Digital Video Camera), προσφέροντας απομακρυσμένα ασφαλή παρατήρηση περιβάλλοντος, αλλά και καλύτερη ακρίβεια προς την εκτέλεση εργασιών.

Τέλος, απομένει το κομμάτι της ασύρματης επικοινωνίας, όπου καθορίζεται από τον πομπό και το δέκτη. Ο δέκτης (Receiver Module), είναι υπεύθυνος για την λήψη κωδικοποιημένων ραδιοκυμάτων, φέροντας πληροφορία. Έπειτα, το στέλνει στον μικροελεγκτή, για να γίνει αποκωδικοποίηση της εντολής που έλαβε, ώστε να ενεργήσουν αντίστοιχα οι περιφερειακές μονάδες. Από την άλλη πλευρά, ο μικροελεγκτής στέλνει μια ψηφιακή κωδικοποιημένη εντολή προς τον πομπό (Transmitter Module), με αποτέλεσμα να παράγει και να μεταδίδει το ανάλογο ραδιοκύμα.

### 3.2.2 Βασική Δομή Τηλεχειριστήριου

Το τηλεχειριστήριο είναι ένα εργαλείο, που δίνει πρόσβαση στους μηχανισμούς και τις ρυθμίσεις του Crobarm. Δηλαδή, απομακρυσμένα έχει την ικανότητα να το ελέγχει πλήρως. Υπάρχει αναπαραγωγή ίδιων ηλεκτρονικών διατάξεων, όπως είναι ο μικροελεγκτής, οι ρυθμιστές τάσης, οι μπαταρίες LI-ion, ο πομπός και ο δέκτης. Οι μοχλοί δεξιά και αριστερά (Right/Left Joysticks), μαζί με το πόμολο ελέγχου (Control Knob), στέλνουν μια αναλογική τάση στο μικροελεγκτή και χρησιμοποιούνται είτε για να κινήσει το ρομπότ, είτε για να διαχειριστεί το βραχίωνα.

Κατόπιν, υπάρχει μια οθόνη (TFT Display), η οποία εμφανίζει με γραφικό τρόπο όλες τις πληροφορίες και τα θεμελιώδη περιεχόμενα. Να σημειωθεί ότι ο χρήστης μπορεί να περιηγηθεί και να επιλέγει διάφορες λειτουργίες για το σύστημα, με την βοήθεια τριών κύριων κουμπιών μενού (Menu Buttons). Εφόσον, πατηθεί ένα κουμπί, αποστέλλει ένα σήμα και εκτελείται ο αντίστοιχος εντολέας, προκαλώντας μια ενέργεια στο σύστημα. Ταυτοχρόνως, εκτελείται ο ήχος συστήματος (System Sound), όπου στην προκυμμένη περίπτωση είναι ένας βομβητής (Buzzer), δημιουργώντας τρείς ξεχωριστούς ήχους, δηλαδή όσα είναι και τα κουμπιά. Οι ενδείκτες Δίοδοι Εκπομπής Φωτός (LED Indicators) είναι πέντε, όπου μαζί με την συνδρομή της οθόνης, ανάβουν ή σβήνουν ανάλογα με τα δεδομένα λήψης. Επίσης, η λάμψη εμποδίου (Obstacle Glow) ανάβει ή σβήνει, όταν το ρομπότ τύχη να συναντήσει μπροστά του κάποιο εμπόδιο. Προαιρετικά, μέσω μιας εφαρμογής για κάμερα από ένα έξυπνο τηλέφωνο (Mobile Camera App), υπάρχει ζωντανή μετάδοση εικόνας από την πλευρά του ρομπότ.



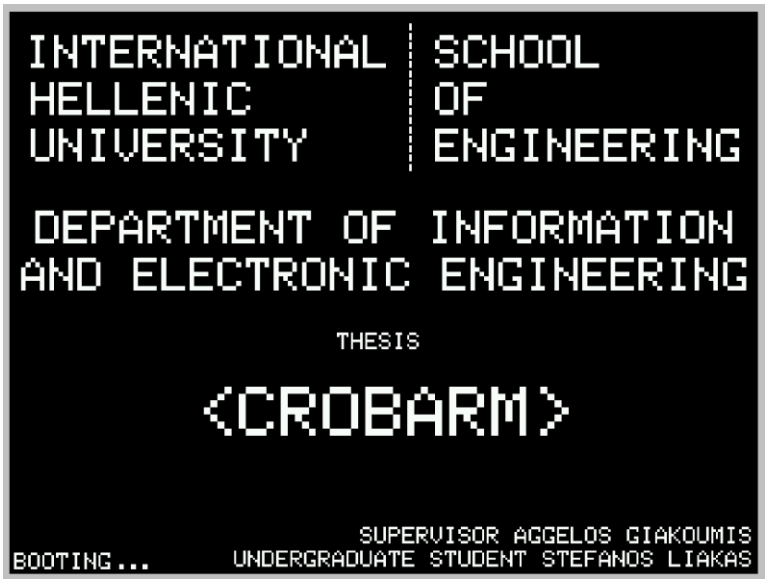

Σχήμα 3.2 Μπλοκ Διάγραμμα Τηλεχειριστήριου

## 3.3 Επεξήγηση Δραστηριότητας Μενού

### 3.3.1 Εκκίνηση Μενού

Το μενού (Menu) εμφανίζεται στην οθόνη του τηλεχειριστήριου και έχει καταλόγους με υπηρεσίες που μπορεί να επιλέξει ο χρήστης, ανάλογα με την δραστηριότητα που θέλει να διεξάγει. Θα γίνει ανάλυση βήμα προς βήμα, για τις δυνατότητες που προσφέρει. Επίσης, πρέπει να σημειωθεί ότι το μενού, είναι γραμμένα στα αγγλικά και όχι στα ελληνικά. Η βιβλιοθήκη (Header File) που χρησιμοποιήθηκε για τα γραφικά της οθόνης, δεν διαθέτει ελληνικούς χαρακτήρες.

Κατά την εκκίνηση του συστήματος, παρουσιάζεται ένα μαύρο φόντο με λευκά κεφαλαία γράμματα. Αντικρίζεται ο τίτλος του πανεπιστήμιου, της σχολής και του τμήματος. Έπειτα, αναφέρεται η λέξη της πτυχιακής εργασίας (Thesis), μαζί με την συντομογραφία του έργου. ως Crobarm. Πέρα από αυτό, κάτω δεξιά διακρίνεται το όνομα του επιβλέπων και του φοιτητή. Ωστόσο, από την αριστερή πλευρά, η εκκίνηση (Booting) αναπαρίσταται από τρείς τελείες, όπου ανάβουν η μια μετά την άλλη και σβήνουν σε επαναλαμβανόμενο χρόνο. Εφόσον περάσει ο χρόνος των 7.5 δευτερολέπτων η οθόνη εισέρχεται στο Crobarm μενού (Main Menu).



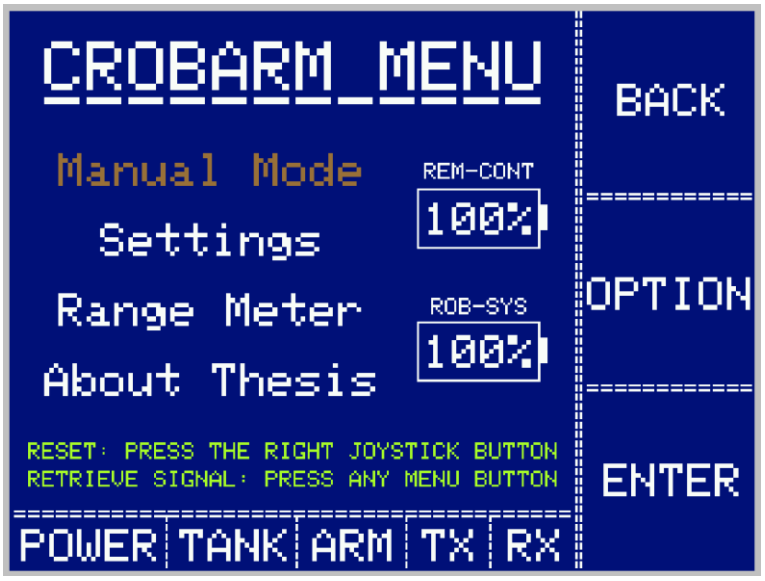

Σχήμα 3.3 Οθόνη Εκκίνησης (Boot Screen)

### 3.3.2 Crobarm Μενού

Ύστερα, σε επόμενο στάδιο εμφανίζεται ο τίτλος του αρχικού μενού (Crobarm Menu) και περιλαμβάνει ένα σκούρο μπλε φόντο με λεύκα γράμματα. Τα χρυσαφένια υποδεικνύουν, ότι ο δείκτης βρίσκεται σε εκείνη τη συγκεκριμένη θέση και μπορεί να αλλάξει επιλογή, με κατεύθυνση από πάνω προς τα κάτω. Τα πράσινα γράμματα, πάντα παρέχουν ένα μικρό κομμάτι χρήσιμης πληροφορίας ή ειδικής συμβουλής. Η διαμόρφωση του μενού έγινε με τέτοιο τρόπο, ώστε από την κάτω πλευρά να έχει τις ενδείξεις, από δεξιά τις επιλογές και ενδιάμεσα τα υπόλοιπα υπομενού.

Καταρχάς, η οριζόντια μπάρα αναγράφει τη τροφοδοσία (Power), το τανκ (Tank), το βραχίονα (Arm), την αποστολή (TX) και τη λήψη (RX). Στο τυπωμένο κύκλωμα υπάρχουν πέντε LED ακριβώς κάτω από αυτές τις λέξεις, που αντιπροσωπεύουν την κατάσταση τους (ON/OFF). Η τροφοδοσία ανάβει (ON) αμέσως μόλις ενεργοποιηθεί το τηλεχειριστήριο και σβήνει (OFF) μόνο όταν απενεργοποιηθεί. Από την άλλη, με βάση τον έλεγχο που επιλέγεται για τανκ ή για βραχίονα, ανάβει το αντίστοιχο LED παρατεταμένα. Μόλις ανάψει το TX-LED, ειδοποιείται ο χρήστης ότι γίνεται αποστολή ενός πακέτου (Package) και σβήνει με την ολοκλήρωση της διαδικασίας. Αντίστοιχα ανάβει το RX-LED, προκυμμένου να ληφθεί ένα πακέτο και μετά απενεργοποιείται.

Η δεξιά κάθετη λωρίδα, αποτυπώνει τις ονομασίες των κουμπιών πίσω (Back), επιλογή (Option), «εισάγω» (Enter), διευκολύνοντας στην περιήγηση ολόκληρου του μενού. Το τυπωμένο κύκλωμα διαθέτει τρείς διακόπτες τακτ (Tact Switch), δίπλα από αυτές τις ονομασίες. Αφού πατηθεί το κουμπί πίσω, η οθόνη επιστρέφει στο Crobarm μενού, με την επιλογή αλλάζει η θέση του δείκτη και με το «εισάγω» εισέρχεται στα υπομενού ή επιλέγεται διαφορετική κατάσταση ενός εντολέα.




Σχήμα 3.4 Crobarm Μενού (Crobarm Menu)

Το κύριο μενού χωρίζεται στα παρακάτω τέσσερα υπομενού:

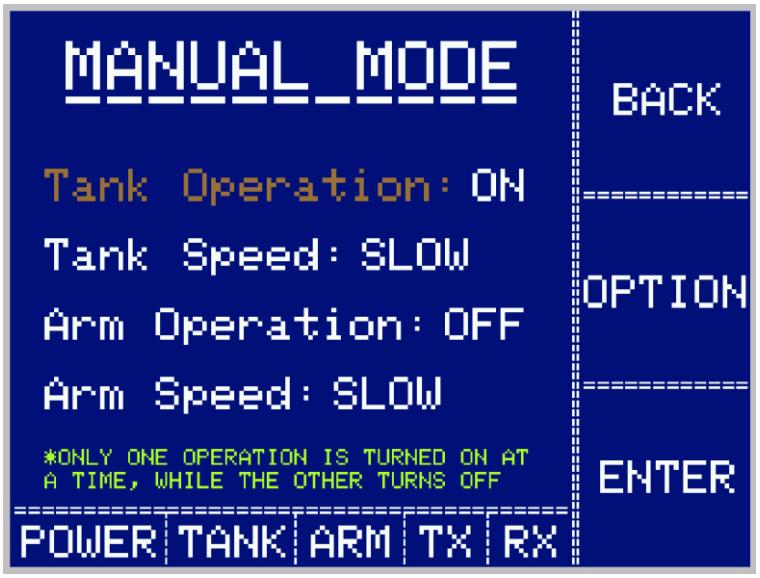
- Χειροκίνητη Λειτουργία (Manual Mode)
- Ρυθμίσεις (Settings)
- Μετρητής Απόστασης (Range Meter)
- Σχετικά με την Π.Ε. (About Thesis)

Εάν προκύψει κάποιο πρόβλημα και χαθεί η επικοινωνία μεταξύ ρομπότ-τηλεχειριστήριου, γίνεται ανάκτηση σήματος (Retrieve Signal) με το πάτημα οποιουδήποτε κουμπιού. Άμα πατηθεί το μπουτόν του δεξιού μοχλού, το σύστημα αποβαίνει σε επανεκκίνηση (Reset). Εμφανίζονται δύο ποσοστά επί τις

εκατό, για την υπολειπόμενη μπαταρία τηλεχειριστήριου (REM-CONT) και ρομπότ (ROB-SYS). Η κλίμακα έχει τις εξής τιμές: 100%, 80%, 60%, 40%, 20%. Οι αριθμοί ανανεώνονται ανά ένα δευτερόλεπτο. Μόλις φτάσει στο 20% η μπαταρία σε ένα από τα δύο, συστήνεται η απενεργοποίηση αυτών. Διαφορετικά με την περαιτέρω μείωση τάσης τροφοδοσίας, η τιμή αλλάζει σε LOW και τα συστήματα δεν θα δουλεύουν ορθά ή ακόμη και καθόλου. Γενικά, το μενού δημιουργήθηκε με στόχο να είναι φιλικό προς το χρήστη (user-friendly), μαθαίνοντάς και κατανοώντας το εύκολα.

### 3.3.3 Υπομενού Χειροκίνητης Λειτουργίας

Αρχικά, η χειροκίνητη λειτουργία αποτελείται από τέσσερις μηχανισμούς, που είναι η εργασία τανκ (Tank Operation), η ταχύτητα τανκ (Tank Speed), η εργασία βραχίωνα (Arm Operation) και τέλος η ταχύτητα βραχίονα (Arm Speed). Ο χειριστής ενημερώνεται από την προειδοποίηση, πως εκτελείται μόνο μια από τις δύο εργασίες κάθε φορά, ενώ η άλλη αυτομάτως απενεργοποιείται. Αυτό σημαίνει παραχώρηση ελέγχου είτε για το τανκ, είτε για το βραχίονα, αλλά ποτέ και στα δύο την ίδια στιγμή. Στην προκυμμένη περίπτωση η εργασία τανκ είναι ON και η εργασία βραχίονα είναι OFF. Πέρα από αυτό, έχουν προστεθεί μέχρι τρείς ταχύτητες, οι οποίες ορίζονται αυθαίρετα. Η ταχύτητα τανκ βρίσκεται στο αργό (Slow), όμως μπορεί να αλλάξει σε κανονική (Normal) ή γρήγορη (Fast). Ωστόσο, η ταχύτητα βραχίονα περιορίζεται σε δύο, με επιλογή για αργό και κανονικό.




Σχήμα 3.5 Υπομενού Χειροκίνητη Λειτουργία (Manual Mode)

### 3.3.4 Υπομενού Ρυθμίσεων

Έπειτα, σειρά έχει το υπομενού ρυθμίσεις, για να προσφέρει προσαρμογή σε κάποιες παραμέτρους, που φέρει τόσο το τηλεχειριστήριο, όσο και το ρομπότ. Κατά την εκκίνηση του μενού, η ασφάλεια κρούσης (Crash Safety) είναι αυτόματα στην λειτουργία ON και αναμενόμενα επιδέχεται απενεργοποίηση. Είναι

απαραίτητο να υφίσταται η παραπάνω ιδιότητα, ώστε σε πολύ κρίσιμες περιπτώσεις να κλείνει ο χειριστής αυτόν τον αυτοματισμό προστασίας. Χαρακτηριστική περίπτωση, είναι να βρίσκεται το ρομπότ σε ένα περιβάλλον με πυκνή βλάστηση, δημιουργώντας πρόβλημα σε αυτό και να πρέπει να απενεργοποιηθεί η ασφάλεια κρούσης. Αυτό συμβαίνει λόγω των φυτών όπου θα εντοπίζονται συνέχεια από τον αισθητήρα υπερήχων, παρόλο που δεν διατρέχει κανέναν κίνδυνο για να συγκρουστεί. Ο μικροελεγκτής του ηλεκτρονικού συστήματος Crobarm, καταλήγει να κολλάει μονίμως μέσα στην συγκεκριμένη διεργασία.

Συμπληρωματικά, αναφέρεται ότι η ασφάλεια κρούσης, αποτρέπει την πιθανότητα σύγκρουσης του ρομπότ. Όταν η μετρούμενη απόσταση γίνει μικρότερη ίση με 15cm, διακόπτεται η επικοινωνία από το τηλεχειριστήριο και κατόπιν το όχημα κάνει όπισθεν για μερικά εκατοστά. Υπενθυμίζεται ότι η παραπάνω διενέργεια, εκτελείται αποκλειστικά κατά τη χειροκίνητη λειτουργια, με την εργασία τανκ να είναι ON.

Τα μπροστινά φώτα (Headlights), είναι αναγκαία για να υπάρχει καλύτερη ορατότητα στο σκοτάδι. Εξαρχής, έχουν ρυθμιστεί με απενεργοποίηση και μονάχα όταν κριθεί απαραίτητο θα τα ενεργοποιήσει ο χειριστής. Επιπρόσθετα, είναι δυνατή η ένταση του εκπεμπόμενου φωτός από τη ταινία LED και θεωρείται ως φώτα μεγάλης σκάλας.


Σχήμα 3.6 Υπομενού Ρυθμίσεις (Settings)

Ύστερα, με το ενεργοποιημένο σύστημα ήχου (System Sound), κάθε φορά και αναλόγως ποιο κουμπί πατιέται, ακούγονται τρείς τόνοι με διαφορετική συχνότητά μεταξύ τους. Αυτό συμβαίνει ως μια ηχητική ανατροφοδότηση, κατά το πάτημα ενός εκ των τριών μπουτόν. Δύναται η επιλογή για απενεργοποίηση του ήχου.

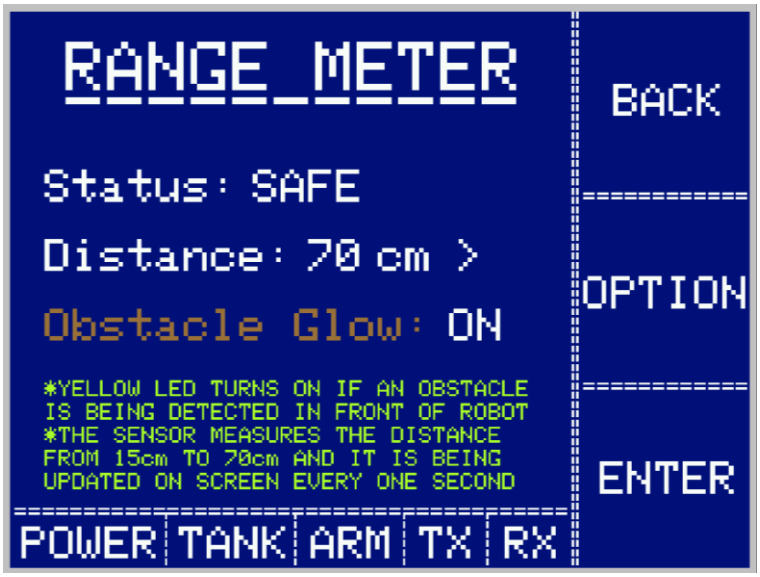
Η φωτεινότητα (Brightness) της οθόνης, ρυθμίζεται για αύξηση ή μείωση προδιαθέτοντας πέντε τιμές από: 100%, 80%, 60%, 40%, 20%. Ανάλογα με το φωτισμό περιβάλλοντος, ο χρήστης θέτει την τιμή προτίμησής του.

### 3.3.5 Υπομενού Μετρητή Απόστασης

Όσον αφορά το μετρητή απόστασης, είναι ένα εργαλείο που χρησιμοποιείται για να καθορίσει με ακρίβεια, την απόσταση που έχει το ρομπότ από ένα αντικείμενο. Περιλαμβάνονται σε αυτό, κάποιες πληροφορίες κατάστασης (Status), οπότε ο χρήστης γνωρίζει ανά πάσα στιγμή πως η περιοχή στην οποία υφίσταται είναι ασφαλής (Safe), χρήζει ιδιαίτερης προσοχής (Caution) ή κινδυνεύει (Danger) να χτυπήσει σε εμπόδιο. Επιπλέον, εμφανίζεται και ανανεώνεται η τιμή της μετρούμενης απόστασης, σε εκατοστά ανά ένα δευτερόλεπτο. Καλύπτει ένα διάστημα από 15cm μέχρι και 70cm.

Πρέπει να τονιστεί πως η απόσταση που μετριέται, έχει ανοχή ενός εκατοστού προς τα πάνω ή προς τα κάτω. Αυτή η ανοχή εξαρτάται από το εάν μικραίνει ή αυξάνεται η απόσταση. Παραδείγματος χάριν, αν βρισκόμαστε ακριβώς στα 28 cm και το ρομπότ ξεκινήσει να πηγαίνει προς τα πίσω, υπάρχει μια νεκρή περιοχή 10 χιλιοστών, η οποία δεν εμφανίζεται στην οθόνη. Εφόσον φτάσει με ακρίβεια στα 29 cm, τότε μόνο θα εμφανιστεί η προαναφερόμενη τιμή. Η κατάσταση στην οποία θα βρίσκεται το ρομπότ, αλλάζει σύμφωνα με τα όρια που έχουν τεθεί προγενέστερα. Από τα 15 cm και κάτω, η κατάσταση θα αλλάξει με την ένδειξη του κινδύνου, επειδή επρόκειτο να χτυπήσει σε κάποιο εμπόδιο. Ενώ, από τα 16 cm μέχρι και τα 70 cm, η σήμανση θα αναγράφει στον χρήστη να είναι προσεκτικός. Άμα η απόσταση είναι από 70 cm και πάνω, τότε είναι ασφαλής.

Η λάμψη εμποδίου (Obstacle Glow) παρουσιάζεται ενεργοποιημένη, καθότι είναι ρυθμισμένο έτσι από πριν. Σύμφωνα με την αναφορά, αν το ρομπότ αντικρίσει κάποιο εμπόδιο κοντά του, τότε θα ανάψει το κίτρινο LED, διαφορετικά θα παραμείνει σβηστό. Εξυπακούεται η δυνατότητα επιλογής, για απενεργοποίηση της προαναφερόμενης ρύθμισης.

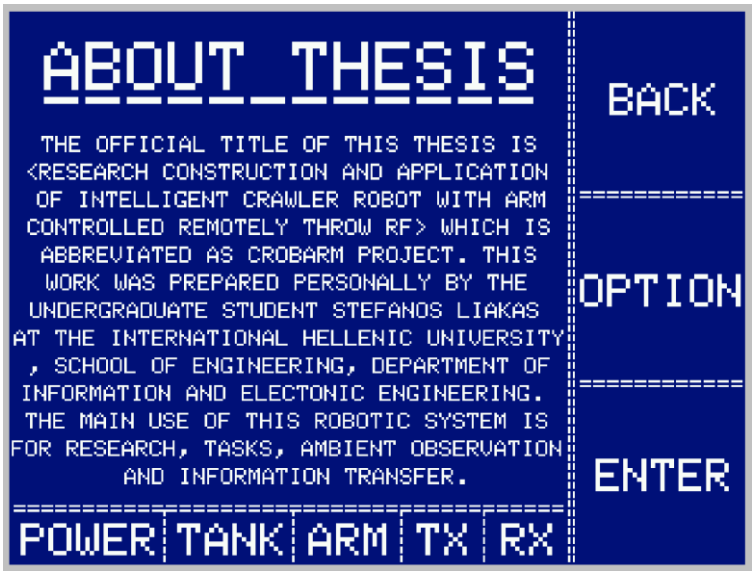



Σχήμα 3.7 Υπομενού Μετρητή Απόστασης (Range Meter)

### 3.3.6 Υπομενού Σχετικά με την Π.Ε.

Κλείνοντας, στο τελευταίο υπομενού δεν περιέχεται κάποια λειτουργία, αλλά ένα γενικό κείμενο το οποίο σχετίζεται με τη πτυχιακή εργασία. Ο αναγνώστης διαπιστώνει το λόγο που δημιουργήθηκε και

τη χρησιμότητά της. Η πλήρης μετάφραση είναι η εξής: Ο επίσημος τίτλος αυτής της Π.Ε. είναι «Μελέτη Κατασκευή και Εφαρμογή Έξυπνου Ερπυστριοφόρου Ρομπότ με Βραχίωνα Ελεγχόμενο Απομακρυσμένα Μέσω RF» συντομογραφημένο ως εργασία Crobarm. Αυτό το έργο ετοιμάστηκε προσωπικά από τον προπτυχιακό φοιτητή Στέφανο Λιάκα στο Διεθνές Πανεπιστήμιο της Ελλάδος, Σχολή Μηχανικών, Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Η κύρια χρήση αυτού του ρομποτικού συστήματος προορίζεται για έρευνα, εργασίες, παρατήρηση περιβάλλοντος και μεταφορά πληροφοριών.

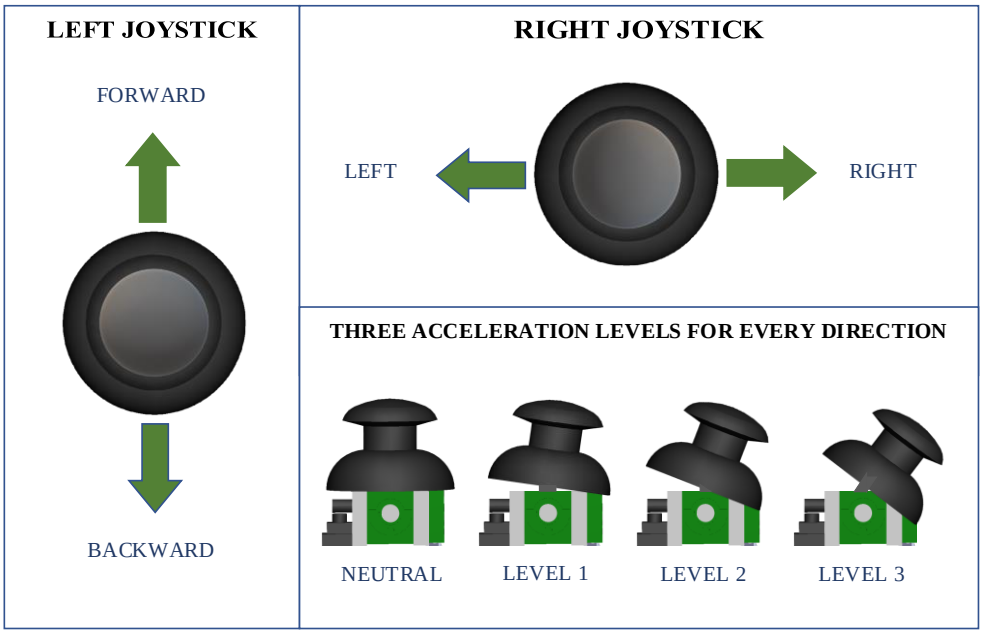



Σχήμα 3.8 Υπομενού Σχετικά με την Π.Ε. (About Thesis)

## 3.4 Έλεγχος Ρομπότ Μέσω Μοχλών

### 3.4.1 Διαχείριση Τανκ

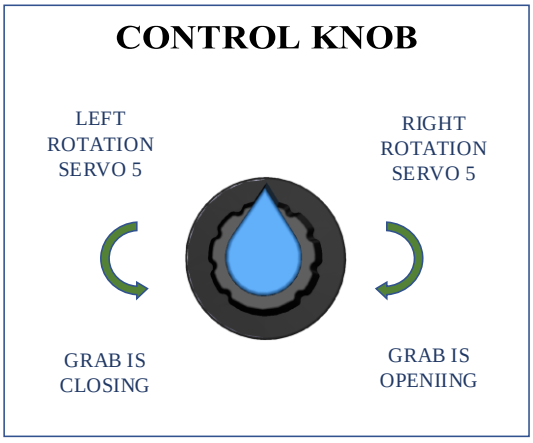
Όπως προαναφέρθηκε νωρίτερα, το τηλεχειριστήριο αποτελείται από δύο μοχλούς, όπου ο ένας βρίσκεται στα αριστερά και ο άλλος στα δεξιά. Εφόσον ο μηχανισμός εργασίας οριστεί για το τανκ και κουνηθεί ο αριστερό μοχλός προς τα πάνω, τότε το όχημα μετακινείται μπροστά. Αντιθέτως, για να μπορέσει να πάει προς τα πίσω, ο μοχλός πρέπει να αλλάξει θέση προς τα κάτω. Πέρα από αυτό, έχει την ικανότητα στροφής με πορεία είτε προς τα δεξιά, είτε προς τα αριστερά. Αφού μετατοπιστεί στην αντίστοιχη κατεύθυνση ο άξονας του δεξιού μοχλού, τότε επιτυγχάνεται η στροφή . Άμα δεν κουνηθούν οι μοχλοί και στις δύο περιπτώσεις, τότε το τανκ θα παραμείνει σε αδράνεια. Όσο πιο μπροστά, πίσω, δεξιά ή αριστερά μετατοπίζεται η θέση στους μοχλούς, τόσο πιο γρήγορα θα κινείται το όχημα. Αυτό συμβαίνει λόγω του ότι και στα τρία επίπεδα ταχύτητας που διαθέτει, η κάθε μια από αυτές έχει τρία επίπεδα επιτάχυνσης για το μπροστά, πίσω, δεξιά, αριστερά. Δεν χρησιμοποιήθηκαν περισσότερες ταχύτητες και επίπεδα επιτάχυνσης, γιατί ο υπάρχων μηχανισμός καλύπτει τις ανάγκες του ρομπότ για μετακίνηση.



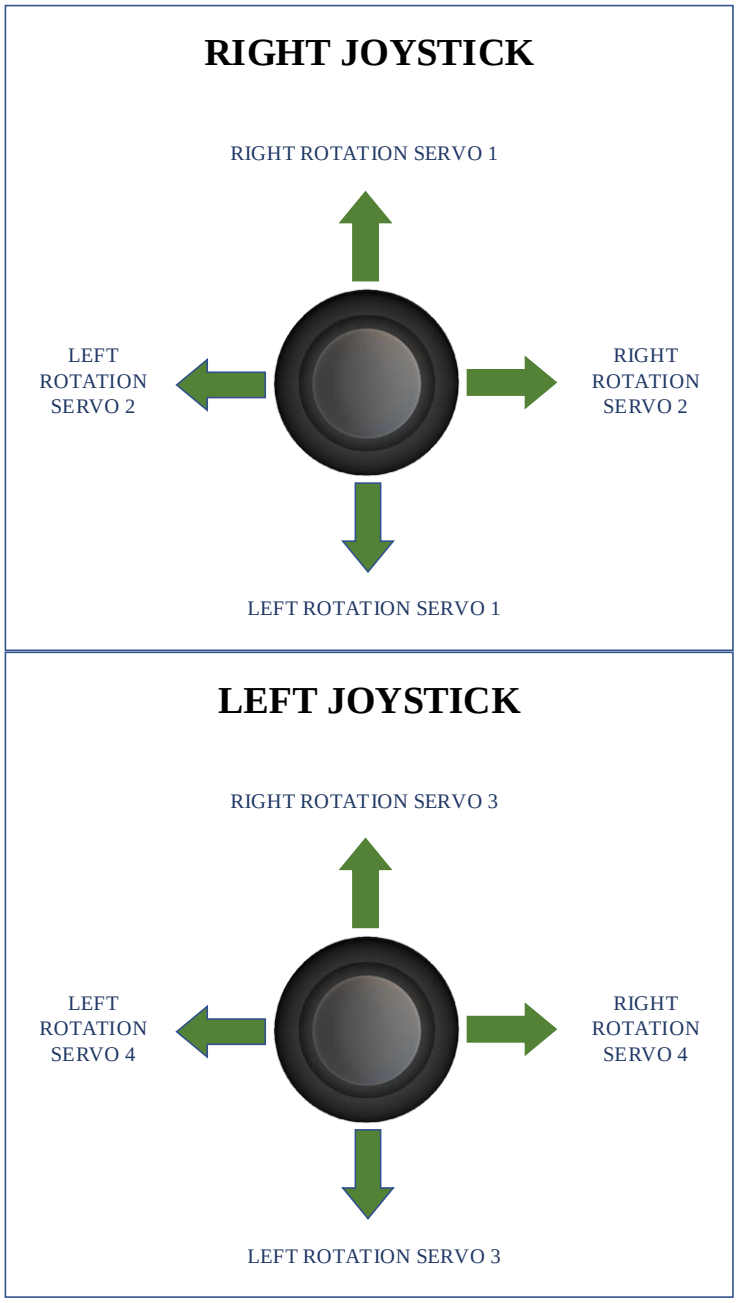

Σχήμα 3.9 Τρόπος Χρήσης Μοχλών για Τανκ

### 3.4.2 Διαχείριση Ρομποτικού Βραχίονα

Στη βασική δομή του ρομπότ, είχε αναγνωριστεί πως ο βραχίονας αποτελείται, από πέντε κινητήρες σέρβο. Οι δύο μοχλοί, καλύπτουν τις απαιτήσεις σε διαχείριση μόνο στους τέσσερις κινητήρες. Ο πέμπτος σέρβο, όπου είναι το άκρο του βραχίονα, δουλεύει ανοίγοντας και κλείνοντας την αρπάγη από το πόμολο ελέγχου. Από τη στιγμή που τεθεί σε εφαρμογή η εργασία βραχίονας, είναι εφικτό για το δεξιό μοχλό, αλλάζοντας τη θέση του κάθετα, να περιστρέφεται ο άξονας του πρώτου σέρβο, ενώ για τον δεύτερο μετατοπίζεται ο μοχλός οριζόντια. Ο τρίτος σέρβο γυρίζει τον άξονα του με την κάθετη κίνηση του αριστερού μοχλού όμως για τον τέταρτο κουνάμε το μοχλό οριζόντια. Τέλος, με την περιστροφή του πόμολου δεξιά ή αριστερά ελέγχεται η αρπάγη. Επιπλέον, ισχύει η ίδια λογική των τριών επιπέδων μετακίνησης για τα δύο επίπεδα ταχύτητας, που προσφέρει το μενού στο βραχίονα.




Σχήμα 3.10 Τρόπος Χρήσης Πόμολου ελέγχου




Σχήμα 3.11 Τρόπος Χρήσης Μοχλών για Βραχίονα

# Κεφάλαιο 4ο: Ιδιότητες Ηλεκτρονικών Εξαρτημάτων

## 4.1 Εισαγωγή στα Ηλεκτρονικά Εξαρτήματα

Κάθε συσκευή ή μηχάνημα αποτελείται από ηλεκτρονικές διατάξεις, οι οποίες έχουν σχεδιαστεί και τοποθετηθεί με τέτοιο τρόπο, για να επιτελείται μια διεργασία. Ο παραπάνω κανόνας, αληθεύει στην περίπτωση των ηλεκτρονικών συστημάτων, που συν αποτελείται το Crobarm. Τα χαρακτηριστικά αυτών των εξαρτημάτων, εξετάστηκαν προσεκτικά προτού επιλεχθούν ως τελικά υλικά. Αυτό έγινε γιατί αναλόγως με τη χρήση και την ποιότητα τους, προβλέπονται καθοριστικά για την σωστή λειτουργικότητα του ρομπότ. Με άλλα λόγια, παίζει σημαντικό ρόλο η κατάλληλη επιλογή ηλεκτρονικών υλικών με οδηγό τα φύλλα δεδομένων (datasheet).

Το παρόν κεφάλαιο επεξηγεί και διευκρινίζει τα παρακάτω κρίσιμα εξαρτήματα του ρομπότ:

- Κινητήρας Συνεχούς Ρεύματος με Ψήκτρες
- Κινητήρας Σέρβο
- Οδηγός Διπλής Πλήρους Γέφυρας
- Σταθεροποιητής Τάσης
- Αισθητήρας Υπερήχων
- Πομπός και Δέκτης Ραδιοσυχνοτήτων
- Ασύρματη Κάμερα Wi-Fi

## 4.2 Κινητήρας Συνεχούς Ρεύματος με Ψήκτρες

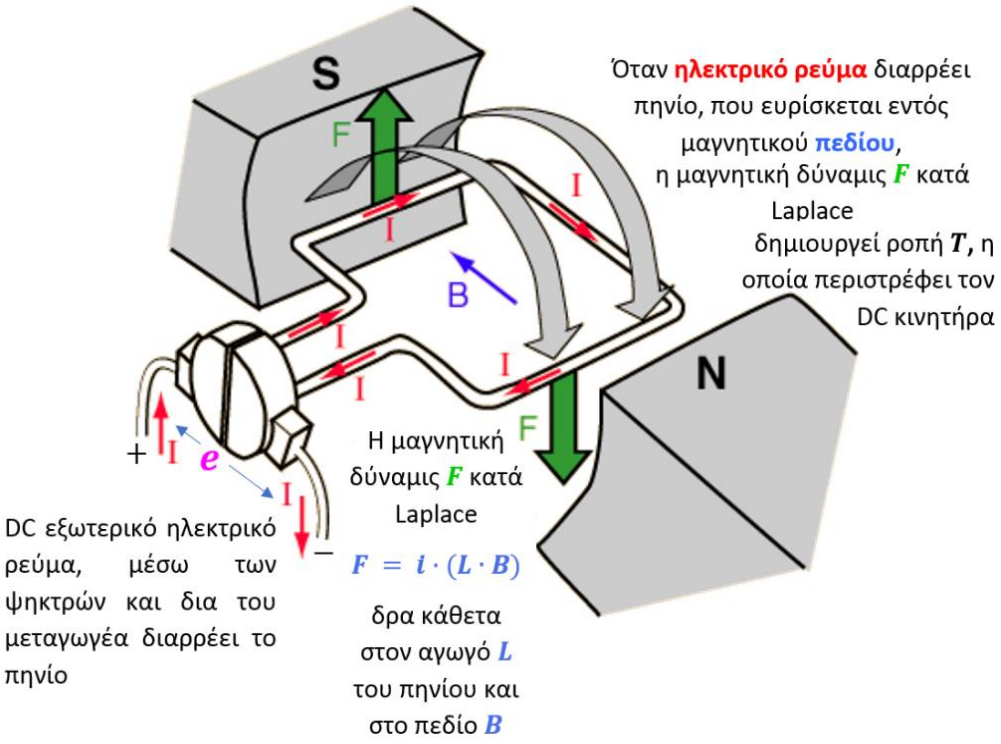
Ένας Κινητήρας Συνεχούς Ρεύματος (DC Motor), μετατρέπει την ηλεκτρική ενέργεια συνεχούς ρεύματος σε μηχανική ενέργεια. Βασικός κανόνας για την κατανόηση της λειτουργίας του κινητήρα, είναι η αρχή λειτουργίας του, όπου βασίζεται στην δύναμη κατά Laplace. Ορίζεται από τον τύπο $F_L = I(L \times B)$. Το σύμβολο B (Μονάδα μέτρησης: Tesla) ορίζει τις γραμμές του μαγνητικού πεδίου, το I (Μονάδα μέτρησης: Ampere) την ένταση του ηλεκτρικού ρεύματος και το L (Μονάδα μέτρησης: Meter) το μήκος του αγωγού. Στον ηλεκτρομαγνητισμό (Electromagnetism), όταν ένας αγωγός τυχαίου μήκους αρχίσει να διαρρέεται από ρεύμα και βρίσκεται σε ομογενές μαγνητικό πεδίο, τότε δημιουργείται η δύναμη κατά Laplace $F_L$. Η επιστημονική κοινότητα πραγματοποιεί εκτεταμένη χρήση του κανόνα του δεξιού χεριού, για την καλύτερη κατανόηση του φαινομένου [3].

Ο Κινητήρας DC έχει ως στοιχειώδης δομή τα εξής χαρακτηριστικά: δρομέας, στάτης, τυλίγματα, ψήκτρες και μεταγωγέας. Ο δρομέας (Rotor) είναι το περιστρεφόμενο πλαίσιο, που γυρίζει γύρω από τον άξονά του. Ο στάτης (Stator) είναι το σταθερό μέρος του κινητήρα, αποτελούμενο από μόνιμο μαγνήτη για την δημιουργία μαγνητικού πεδίου. Μεταξύ δρομέα και στάτη, πρέπει να υπάρχει ένα κενό αέρος για να μπορεί να περιστρέφεται. Τα τυλίγματα (Windings) είναι πολλά χάλκινα καλώδια, που χωρίζονται σε ομάδες πηνίων (Coils) και τοποθετούνται γύρω από μαγνητικό πυρήνα μαλακού σιδήρου,

επάνω στο δρομέα. Η ψήκτρα (Brush) αποτελεί μια ηλεκτρική επαφή για την μεταφορά ρεύματος, από ακίνητους κλώνους καλωδίων στα τυλίγματα του δρομέα. Ο μεταγωγέας (Commutator) είναι ένας δακτύλιος με δύο κενά, περιστρεφόμενος μαζί με τον δρομέα [18].

Χρησιμοποιήθηκαν δύο Κινητήρες Συνεχούς Ρεύματος με Ψήκτρες 33GB-520. Πρόκειται για μοτέρ μεταλλικού μηχανισμού και υψηλής ταχύτητας, καλύπτοντας τις απαιτήσεις για ροπή. Τροφοδοτούνται με τάση εύρους 6V-12V και ζυγίζουν 100 γραμμάρια το καθένα. Χωρίς φορτίο μπορεί να τραβήξει ρεύμα μέχρι 100mA και να φτάσει τις στροφές των 170-350 rpm. Παρόλα ταύτα, δοκιμάστηκε και με φορτίο, καταλήγοντας να απορροφήσει μέχρι 700mA περίπου.


Σχήμα 4.1 DC Κινητήρας 33GB-520 [35]

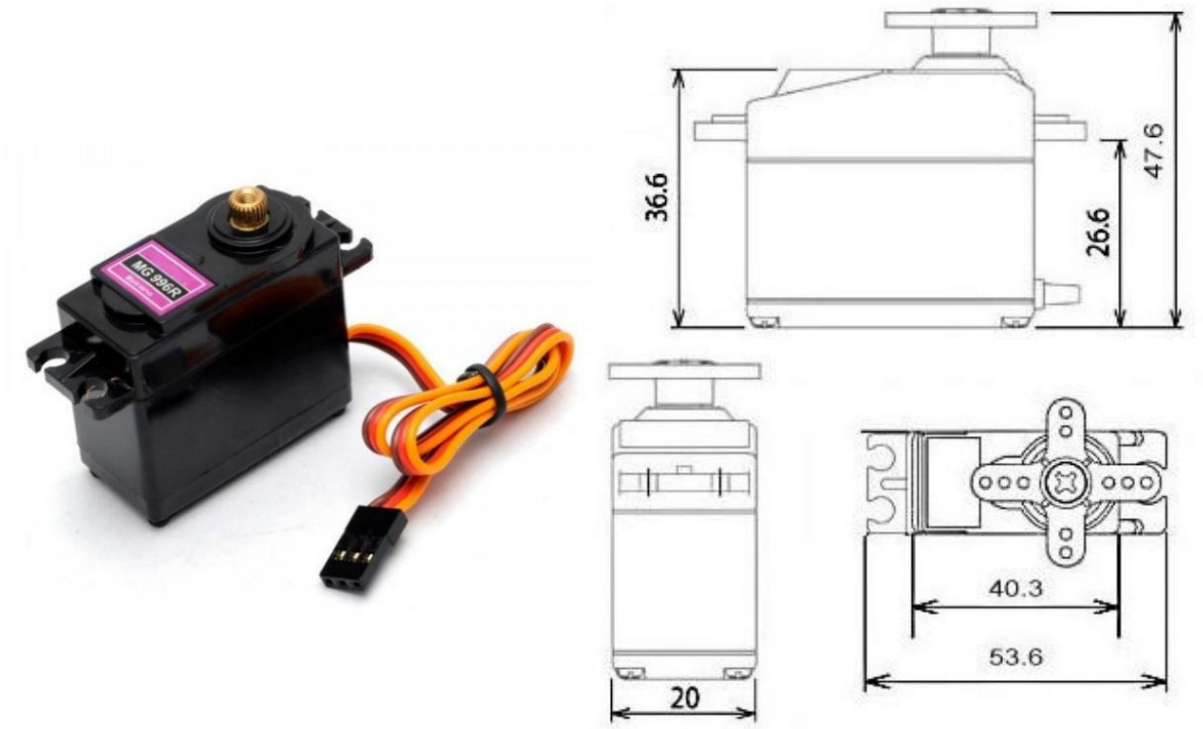



Σχήμα 4.2 Θεμελιώδης Λειτουργία DC Κινητήρα [3]

## 4.3 Κινητήρας Σέρβο

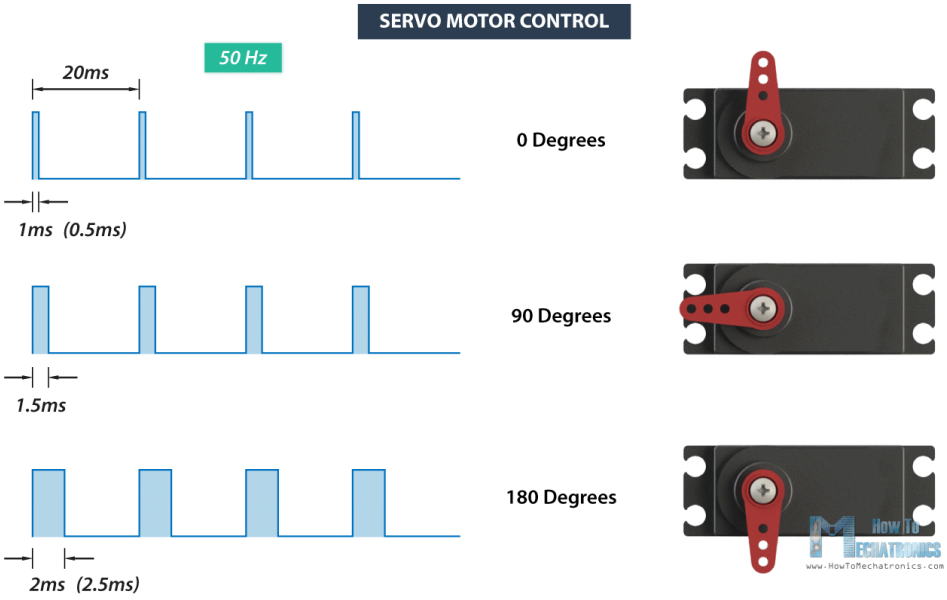
Ο Κινητήρας Σέρβο παρέχει ένα μηχανισμό, βασιζόμενο στο σύστημα ελέγχου κλειστού βρόγχου (Closed Loop), για την οδήγηση ενός κινητήρα του. Επιτρέπει τον ακριβή έλεγχο της γωνιακής θέσης του άξονα, σε ταχύτητα και επιτάχυνση. Μέσα στο Σέρβο βρίσκονται τέσσερα κύρια εξαρτήματα: DC κινητήρας, ελεγκτής, ποτενσιόμετρο θέσης και το κιβώτιο ταχυτήτων. Ο Κινητήρας DC είναι υψηλής ταχύτητας με χαμηλή ροπή και ο άξονάς του συνδέεται με το κιβώτιο ταχυτήτων. Αυτό έχει ως αποτέλεσμα να μειωθεί η ταχύτητα και να αυξηθεί η ροπή του. Το Ποτενσιόμετρο (Potentiometer) συνδέεται με τον εξωτερικό άξονα και ανάλογος με την περιστροφή που γίνεται λόγο του κινητήρα, παράγει μια τάση σύμφωνα με την γωνία που θα βρίσκεται κάθε φορά. Ο Ελεγκτής (Controller) συγκρίνει την τάση του ποτενσιόμετρου, με αυτή του σήματος που δέχεται από έναν εξωτερικό μικροελεγκτή. Εφόσον διαφέρουν τα δύο σήματα μεταξύ τους, ο ελεγκτής θα ενεργοποιήσει αντίστοιχα την εσωτερική πλήρης γέφυρα, ώστε να περιστραφεί ο άξονας δεξιά ή αριστερά. Εάν η διαφορά τάσης των δύο εισόδων είναι 0V, στέλνεται σήμα για την σταθεροποίηση του άξονα στο τωρινό σημείο [15].

Έγινε χρήση πέντε Κινητήρων Σέρβο MG996R. Κατέχουν τρία καλώδια, με το πορτοκαλί χρώμα να συμβολίζει το σήμα PWM, το κόκκινο την τάση τροφοδοσίας και το καφέ την γείωση. Τροφοδοτείται με τάση 4.8V-7.2V, το μέγιστο ρεύμα που μπορεί να τραβήξει είναι 500mA-900mA (6V) και ζυγίζει 55 γραμμάρια. Στα 6V η ροπή αγγίζει μέχρι και τα 11 κιλά ανά εκατοστό. Επίσης, μπορεί να περιστραφεί περίπου μέχρι 180 μοίρες. Το Σέρβο ελέγχεται με σήματα Μεταβλητού Εύρους Παλμών για την γωνία, την ταχύτητα και την επιτάχυνση. Συνήθως, θα έχουν μια σταθερή συχνότητα των 50Hz




Σχήμα 4.3 Κινητήρας Σέρβο και Διαστάσεις [8]

και με την μεταβολή του κύκλου εργασίας κατά 1-2 ms. Στην περίπτωση του Crobarm είναι από τα 0.4ms έως τα 2.4ms. Θεωρητικά ο άξονας περιστρέφεται από 0 έως 180 μοίρες [8].



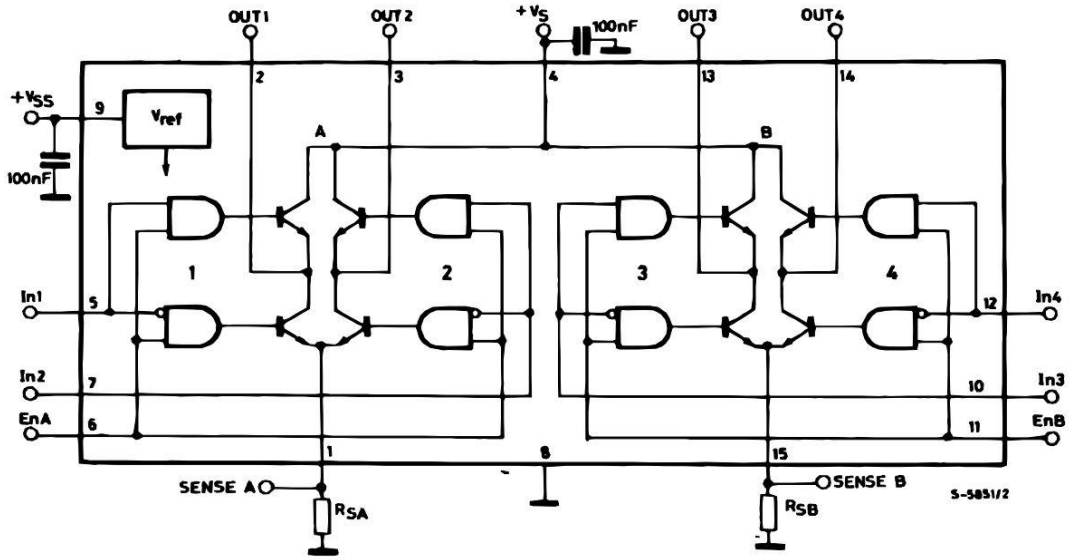

Σχήμα 4.4 Έλεγχος Κινητήρα Σέρβο με PWM Σήματα [36]

## 4.4 Οδηγός Διπλής Πλήρους Γέφυρας

Ο Οδηγός Διπλής Πλήρης Γέφυρας (Driver Dual H Bridge), είναι ένα εξάρτημα που διαχειρίζεται το πλήρη έλεγχο σε ταχύτητα και περιστροφή κατεύθυνσης, για δύο Κινητήρες DC. Ο έλεγχος της ταχύτητας επιτυγχάνεται με σήματα PWM, ενώ ο έλεγχος της κατεύθυνσης με πλήρης γέφυρες Η. Τα ποδαράκια ENA και ENB δέχονται σήματα PWM. Είναι μια τεχνική που χρησιμοποιείται για τον έλεγχο της τάσης εισόδου, με διακοπτικός παλμούς, παράγοντας υψηλή (ON) και χαμηλή (OFF) στάθμη. Η λογική αυτού του μηχανισμού έχει την κύρια ιδιότητα, για μεγαλύτερο κύκλο εργασίας να αυξάνεται και η εφαρμοζόμενη τάση στον κινητήρα. Αντιθέτως, με την πτώση του κύκλου εργασίας, μειώνεται η τάση στα άκρα του, καταλήγοντας στην πτώση στροφών του κινητήρα σε σχέση με πριν [13][16].

Η πλήρης γέφυρα Η, κατασκευάστηκε για να θέτει την λειτουργία του DC κινητήρα, σε δύο κατευθύνσεις. Η δομή της γέφυρας ορίζεται από τέσσερις κρυσταλοτριόδους. Τα BJT συμπεριφέρονται ως διακόπτες, μόλις εφαρμοστεί τάση στις εισόδους B (Base), η οποία τάση προέρχεται από λογικές πύλες τεχνολογίας TTL. Αναλόγως τους διακόπτες που ενεργοποιούνται, εξαρτάται η περιστροφή του κινητήρα αν θα είναι δεξιόστροφη (Clockwise) ή αριστερόστροφη (Anti-Clockwise) [13].

Το L298N είναι ένα μονολιθικό ολοκληρωμένο κύκλωμα και δουλεύει με τάση μέχρι τα 46V, με ρεύμα μέχρι τα 4 A (2A κάθε κινητήρας), έχει χαμηλό κορεσμό τάσης, διαθέτει προστασία από υπερθέρμανση και δομείται από 2 πλήρης γέφυρες Η. Για την τροφοδοσία τάσης των λογικών πυλών ορίζεται από τα 4.5V-7V. Η κατεύθυνση του πρώτου κινητήρα ελέγχεται από τους ακροδέκτες IN1 και IN2. Ενώ, για το δεύτερο κινητήρα ορίζονται οι ακροδέκτες IN3 και IN4 [13][16].




Σχήμα 4.5 Μπλοκ Διάγραμμα Οδηγού L298N [13]

MOTOR A

| IN1 | IN2 | Spinning Direction |
|---|---|---|
| 0 | 0 | Stop |
| 0 | 1 | Left |
| 1 | 0 | Right |
| 1 | 1 | Stop |

MOTOR B

| IN3 | IN4 | Spinning Direction |
|---|---|---|
| 0 | 0 | Stop |
| 0 | 1 | Left |
| 1 | 0 | Right |
| 1 | 1 | Stop |

Σχήμα 4.6 Ακροδέκτες Ελέγχου για τον Οδηγό L298N

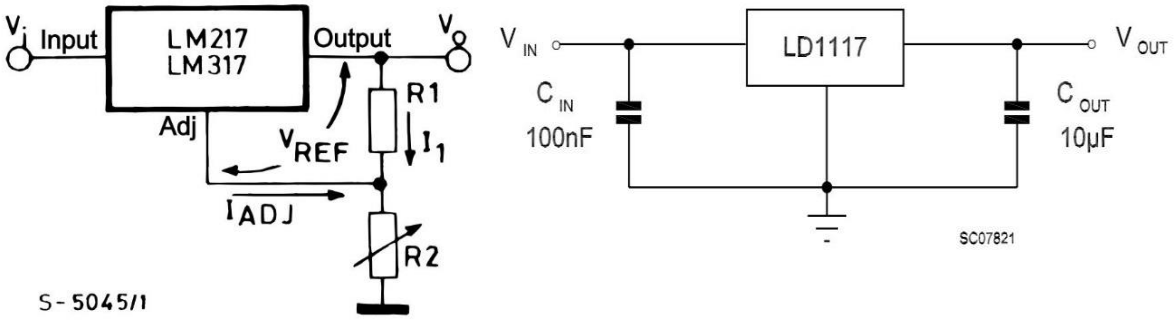
## 4.5 Σταθεροποιητής Τάσης

Ο Σταθεροποιητής Τάσης (Voltage Regulator) είναι ένα ολοκληρωμένο κύκλωμα για την αυτόματη διατήρηση τάσης, σε μια συγκεκριμένη τιμή. Εντοπίζονται δύο κατηγορίες Regulators.

Οι ρυθμιζόμενοι που τοποθετήθηκαν στο ρομπότ είναι τα LM317, ικανά να τροφοδοτήσουν μέχρι 1.5A, με τάσεις από 1.25V-37V. Με δύο εξωτερικούς αντιστάτες και τη χρήση μιας φόρμουλας, ρυθμίζεται η τάση εξόδου $\text{Vout} = 1.25(1 + (\text{R2}/\text{R1}))$. Σύμφωνα με τα κυκλώματα που διαθέτει, θεωρείται ο καλύτερος τρόπος για την σταθεροποίηση φορτίου. Επίσης, λόγο των εσωτερικών κυκλωμάτων προστασίας, γίνεται περιορισμός του ρεύματος για να μην υπερφορτωθεί και ξεπεράσει τα 1.5A. Εξίσου σημαντικό είναι η θερμική προστασία που κατέχει, εάν γίνει υπερβολική κατανάλωση ισχύος από το φορτίο. Επιπρόσθετα, έγινε εκτεταμένη χρήση καθορισμένων σταθεροποιητών (Fixed Voltage Regulators). Πρόκειται για τα LM7805, LM7806, LD1117V33, όπου έχουν τα ίδια χαρακτηριστικά με τον LM317. Απλώς, τα συγκεκριμένα παρέχουν προκαθορισμένη τάση στην έξοδό τους, όπως το LM7805 στα 5V, το LM7806 στα 6V και στα 3.3V το LD1117V33 (800mA και όχι 1.5A).

Τοποθετήθηκαν ψήκτρες, επικαλυπτόμενες με θερμοαγώγιμη πάστα (Thermal Paste) στη πίσω πλευρά των σταθεροποιητών, επιτυγχάνοντας καλύτερη απαγωγή θερμότητας. Έτσι, διατηρείται σε σχετικά χαμηλά επίπεδα η θερμοκρασία στο κέλυφος. Όλα τα αναφερθέντα ολοκληρωμένα κυκλώματα, συστήνουν τη τοποθέτηση πυκνωτή σε είσοδο-έξοδο. Στην είσοδο προτείνεται ένας πυκνωτής παράκαμψης, για να φιλτράρονται τα DC σήματα αφαιρώντας τον AC θόρυβο. Θεωρείται κρίσιμος ο πυκνωτής εξόδου για την βέλτιστη σταθεροποίηση του ρυθμιστή (Regulator) [9][10].

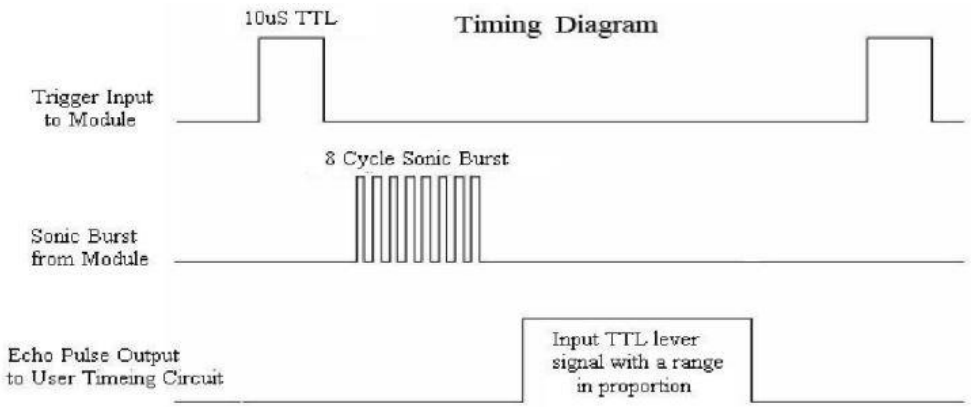
Η σειρά των L78XX, έχει μια πτώση τάση των 2V, τα LM317 μια πτώση των 1.25V και τα LD1117V33 πτώση 1V. Είναι υψίστης σημασίας αυτό, γιατί αν δοθεί μια τάση εισόδου με μικρότερη της ονομαστικής που έχουν ήδη, τότε θα μειωθεί η τάση στην έξοδο σύμφωνα με τις παραπάνω τιμές. Για παράδειγμα, στον LM317 από τους αντιστάτες ορίστηκε η τάση εξόδου στα 5V, όμως η είσοδος είναι στα 5.5V. Συνεπάγεται, ότι έξοδος θα καταλήξει να είναι 5.5V-1.25V=4.25V. Δηλαδή για να έχει 5V η έξοδος θα πρέπει η είσοδος να είναι τουλάχιστον 6.25V.




Σχήμα 4.7 Συνδεσμολογία Σταθεροποιητών LM317 και LD1117V33 [9][10]

## 4.6 Αισθητήρας Υπερήχων

Ο Αισθητήρας Υπερήχων είναι ένα ειδικό όργανο, για να μετράει την απόσταση από το σημείο που βρίσκεται, μέχρι ένα αντικείμενο σημαδεύοντάς το. Είναι ικανό να μετράει αποστάσεις που κυμαίνονται από τα 2 εκατοστά, μέχρι τα 4 μέτρα και με ακρίβεια έως 3 χιλιοστά. Η μονάδα HC-SR04 περιλαμβάνει




Σχήμα 4.8 Χρονικό Διάγραμμα Λειτουργίας Αισθητήρα HC-SR04 [11]

ένα κύκλωμα ελέγχου, πομπού και δέκτη υπερήχων. Ο τρόπος λειτουργίας του ορίζεται σε τρείς φάσης.

Στον ακροδέκτη σκανδαλισμού (Trigger), εισάγεται ένας θετικός τετραγωνικός παλμός 10μs. Αυτομάτως, η μονάδα εκπομπής στέλνει ένα σήμα οκτώ τετραγωνικών παλμών, συχνότητας 40KHz. Άμα συναντήσει κάποιο εμπόδιο το αρχικό σήμα, τότε θα ανακλαστεί προς τα πίσω. Ο αποδέκτης διαβάζει το σήμα επιστροφής, αντιλαμβάνοντας πως υπάρχει αντικείμενο μπροστά. Ανάλογος με την απόσταση αν βρίσκεται κοντά ή μακριά από το σημείο, ο ακροδέκτης ηχώ (Echo) εξάγει έναν παλμό υψηλής στάθμης πεπερασμένης διάρκειας [11].

Δέχεται τροφοδοσία τάσης 5V και απαιτεί ρεύμα 15mA. Οι διαστάσεις του είναι $45 \times 20 \times 15$mm. Είναι καθοριστική η ειδική φόρμουλα από τα φύλλα δεδομένων, καθότι γράφει πως μs/58.82 δίνει ακριβώς την απόσταση σε εκατοστά. Τα μs είναι η διάρκεια του παλμού που επιστρέφει το ποδαράκι Echo. Προτείνεται, ο κύκλος μέτρησης να πραγματοποιείται μετά τα 60ms, για να μην συγκρουστούνε χρονικά τα σήματα Trigger-Echo [11].

## 4.7 Πομπός και Δέκτης Ραδιοσυχνοτήτων

Ο Πομπός και Δέκτης Ραδιοσυχνοτήτων, είναι το τηλεπικοινωνιακό σύστημα που επιτρέπει την επικοινωνία μεταξύ ρομπότ-τηλεχειριστήριο, αποστέλλοντας-λαμβάνοντας ηλεκτρομαγνητικά κύματα.

Ο Πομπός (Transmitter) ορίζεται ως ένα ηλεκτρονικό εξάρτημα ασύρματης επικοινωνίας, καθότι μαζί με τη βοήθεια γραμμικής κεραίας, εκπέμπει ραδιοκύματα συχνότητας 315MHz. Ο ακροδέκτης VCC είναι για την τροφοδοσία, το GND για την γείωση και το DATA για τα δεδομένα που εκπέμπει. Το σύστημα του είναι σχετικά εύκολα δομημένο, γιατί το μόνο που πρέπει να γίνει είναι να δεχθεί λογικό "1" στο ποδαράκι DATA. Μόλις γίνει αυτό, ενεργοποιείται ο ταλαντωτής παράγοντας ένα φέρον σήμα συχνότητας 315MHz, ενώ με το λογικό "0" απενεργοποιείται ο ταλαντωτής [17].

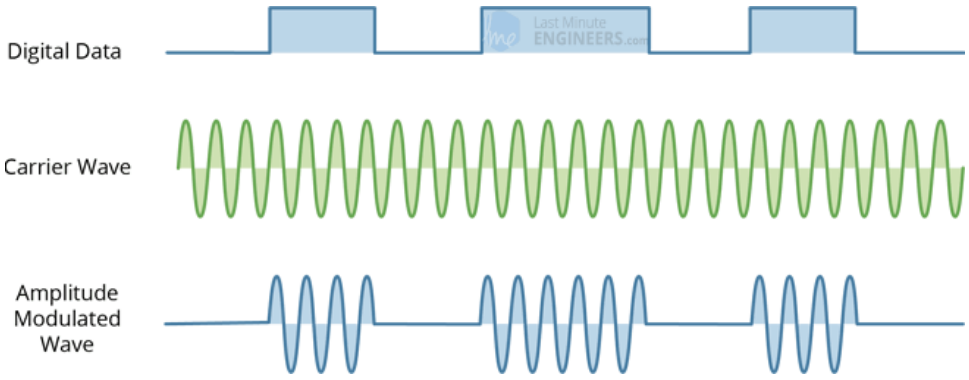
Από την άλλη πλευρά, ο Δέκτης (Receiver) διατυπώνεται ως μια πιο περίπλοκη ηλεκτρονική διάταξη, που αποτελείται εξίσου από μια γραμμική κεραία για να λαμβάνει ραδιοκύματα. Οι ακροδέκτες είναι οι ίδιοι με αυτούς του δέκτη, όμως διαθέτει δύο DATA (κοινό ποδαράκι) για να εξάγει τα δεδομένα που έλαβε. Εφόσον, λάβει κάποιο σήμα συχνότητας 315MHz, ενισχύεται από μερικούς τελεστικούς ενισχυτές (OP AMP). Στην συνέχεια, περνάει μέσα από ένα σύστημα Βρόγχου Κλειστής Φάσης, για να καταλήξει στον αποκωδικοποιητή (Decoder), επιλέγοντας μια ροή αποδιδόμενων bits. Δηλαδή bits με καθαρή αποκωδικοποίηση και χαμηλού θορύβου, οδηγώντας τα στην έξοδο DATA [17].

Εφαρμόζεται η τεχνική Διαμόρφωσης Μετατόπισης Πλάτους (Amplitude Shift Keying, ASK), για την ασύρματη μετάδοση της πληροφορίας. Είναι μια μορφή Διαμόρφωσης Πλάτους (AM), αντιπροσωπεύοντας ψηφιακά δεδομένα με την αλλαγή πλάτους του φέροντος σήματος, συχνότητας 315MHz (Carrier Signal). Το λογικό "0" σημαίνει αποκοπή του φέροντος, ενώ το λογικό "1" μέγιστη ισχύς πλάτους φέροντος. Ένα μειονέκτημα αυτής της μεθόδου, είναι πως είναι ευαίσθητη στις παρεμβολές (Interference) από άλλες συσκευές και τον θόρυβο (Noise). Όμως η μετάδοση δεδομένων γίνεται με μικρές ταχύτητες απόκρισης, αποτρέποντας την συνεχόμενη διακοπή των τηλεπικοινωνιών. Βέβαια, εάν βρεθεί σε περιβάλλον με αρκετά εμπόδια, είναι συνηθέστερο να χάνεται το σήμα πομπού [17].

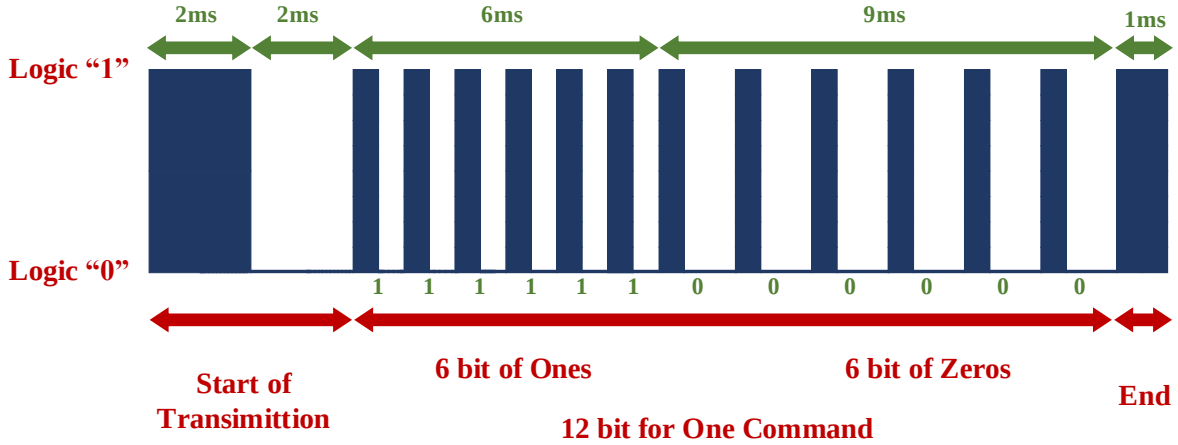
Δημιουργήθηκε το Crobarm Protocol (CP) για την μετάδοση κωδικοποιημένων εντολών, από το τηλεχειριστήριο προς το ρομπότ και αντιστρόφως. Μια κωδικοποιημένη εντολή χωρίζεται σε τρία μέρη.

Η εκκίνηση μετάδοσης γίνεται με έναν τετραγωνικό παλμό 2ms και ακολουθεί ένα κενό 2ms. Αμέσως μετά υπάρχει η πληροφορία των 12 bit, χωριζόμενη πάντα σε 6 άσσους (6ms) και 6 μηδενικά (9ms) για να διατηρείται σταθερός ο χρόνος απόκρισης. Το bit με τιμή 1, είναι παλμός 0.5ms και κενού 0.5ms, ενώ με τιμή 0 είναι bit παλμού 0.5ms με κενό 1ms. Τέλος, η εντολή κλείνει με έναν παλμό 1ms. Συνολικά η κάθε εντολή έχει διάρκεια χρόνου $2ms + 2ms + 6\text{ms} + 9\text{ms} + 1\text{ms} = 20$ ms. Η μορφή του πρωτοκόλλου προκαθορίστηκε έτσι, ώστε ανά 4 bit να λαμβάνεται μια από τις παρακάτω 6 τιμές, αποτελούμενες με δύο μηδενικά και δύο άσσους: 0011, 0101, 0110, 1001, 1010, 1100. Συνολικά καταγράφονται 216 μοναδικές εντολές. Ο υπολογισμός έφερε το παρακάτω αποτέλεσμα: $4 + 4 + 4 = 12$ bit άρα $6 \times 6 \times 6 = 216$.

Ο δέκτης είναι το μοντέλο XD-RF-5V, με διαστάσεις $30 \times 14 \times 7$mm, λειτουργεί σε συνεχή τάση 5V με ευαισθησία $S = -105$dB. Ο πομπός είναι το μοντέλο XD-FST, με διαστάσεις $19 \times 19$mm, λειτουργεί σε συνεχή τάση 3.5-12V με ισχύς μετάδοσής $P_T = 10$mW. Ορίζεται ως μέγιστη εμβέλεια τα 200m σε ανοιχτό χώρο, το οποίο εξαρτάται από την εφαρμοζόμενη τάση (μικρότερη τάση, μικρότερη απόσταση). Η πιο αποδοτική κεραία θα έχει το ίδιο μήκος με αυτό της κυματομορφής και δίνεται από τον παρακάτω τύπο $\lambda = v/f = (299792458\text{m/s})/433000000\text{Hz} = 69.24$cm. Το λ είναι το μήκος κύματος, το v η ταχύτητα του φωτός και το f η συχνότητα μετάδοσης. Επειδή είναι αρκετά μεγάλο το μήκος των 69.24cm, χρησιμοποιήθηκε ίσια κεραία καλωδίου (Straight Wire Antenna), σχεδόν το μισό από το αρχικό μέγεθος στα 30cm καλύπτοντας τις ανάγκες του Crobarm [17].




Σχήμα 4.9 Διαμόρφωση Σήματος κατά ASK [17]




Σχήμα 4.10 Θεωρητικό Παράδειγμα Κωδικοποιημένης Εντολής Πρωτοκόλλου CP

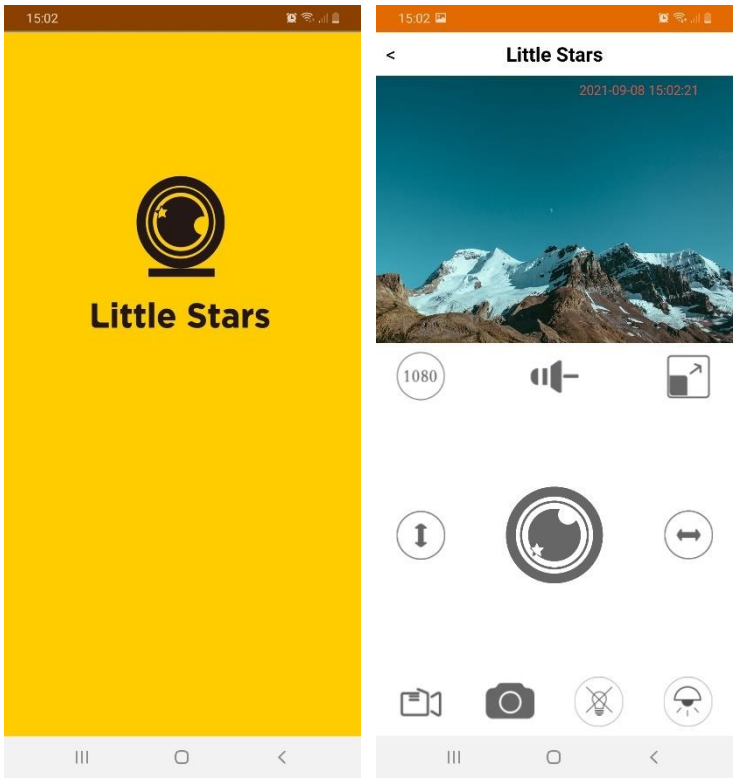
## 4.8 Ασύρματη Κάμερα Wi-Fi

Η ασύρματη ψηφιακή κάμερα A9 mini, είναι απαραίτητη για την καταγραφή και αποστολή ζωντανής μετάδοσης βίντεο, προς τον χειριστή του ρομπότ. Η κάμερα τοποθετείται στη πίσω βάση του τέταρτου κινητήρα σέρβο. Επομένως, προσφέρεται στο χρήστη, η δυνατότητά παρακολούθησης από το Crobarm στο τριγύρω περιβάλλον, ρυθμίζοντας μονάχα τον βραχίονα. Ειδικότερα, με αυτόν τον απομακρυσμένο τρόπο μπορούν να εντοπιστούν αντικείμενα, να παρθούν και να αποσταλούν σε άλλο μέρος. Συνεπάγεται, ότι αποτελεί σημαντικό πλεονέκτημα η όραση στο ρομπότ.

Η επικοινωνία μεταξύ ασύρματης IP κάμερας και smartphone, επιτυγχάνεται από το πρωτόκολλο Wi-Fi 2.4GHz (IEEE 802.11b/g/n). Το κινητό πρέπει να έχει εγκατεστημένη την εφαρμογή Little Stars (Android). Η Wi-Fi κάμερα μπορεί να συνδεθεί μέσω διαδίκτυο (Hotspot/Router) ή και χωρίς. Στην πρώτη περίπτωση παρέχεται μεγαλύτερη απόσταση σύνδεσης, από ένα τοπικό δίκτυο ή hot spot διευρύνοντας την σύνδεση μέχρι τα 60m σε ανοιχτό χώρο. Η δεύτερη περίπτωση μικραίνει την απόσταση στα 12m, όμως δεν απαιτείται internet.

Εφόσον η τροφοδοσία είναι συνεχόμενης τάσης 5V, εικοσιτέσσερις ώρες το εικοσιτετράωρο θα είναι διαθέσιμη για χρήση. Εμπεριέχει μπαταρία των 400mAh, με διάρκεια ζωής 2 ώρες. Η ποιότητα ανάλυσης είναι 1280x720p, η ταχύτητα στα 15FPS και η οπτική γωνία φακού στις 150ο. Μετά την ολοκλήρωση καταγραφής βίντεο, το αρχείο αποθηκεύεται στο κινητό. Τέλος, εμπεριέχεται ένα μικρόφωνο μέσα στο σύστημα.

Σχήμα 4.11 A9 Mini Camera Wi-Fi




Σχήμα 4.12 Είσοδος και Μενού Ασύρματης Κάμερας

# Κεφάλαιο 5ο: Προσομοίωση, Προγραμματισμός, Εφαρμογή

## 5.1 Σχεδίαση και Κατασκευή Τυπωμένων Κυκλωμάτων Πλακέτας

### 5.1.1 Θεωρητική Αναπαράσταση PCB

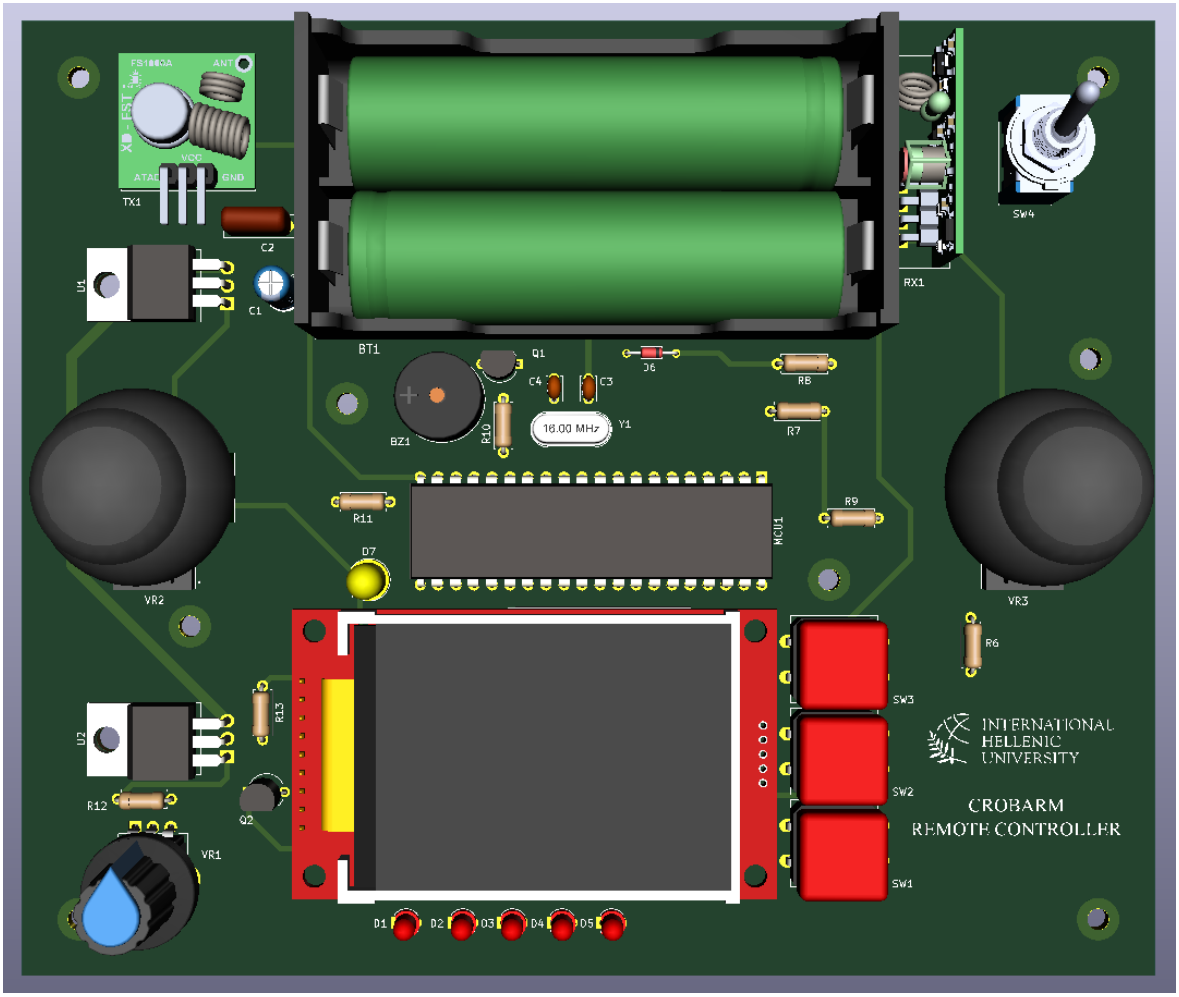
Η σχεδίαση των τυπωμένων κυκλωμάτων, πραγματοποιήθηκε στη σουίτα ανοικτού λογισμικού KiCad EDA έκδοσης 5.1.9. Αρχικά, πραγματοποιείται σχεδίαση των σχηματικών (Schematics) που αφορούν τα ηλεκτρονικά κυκλώματα. Επόμενο στάδιο, είναι η μετατροπή του σχηματικού σε Τυπωμένο Κύκλωμα Πλακέτας (PCB). Διαθέτει ένα ειδικό περιβάλλον τριών διαστάσεων για αξιολόγηση (3D Viewer). Δημιουργήθηκαν καινούργια σύμβολα (Symbols) και αποτυπώματα (Footprints), για να καλυφθεί η προσθήκη μερικών ηλεκτρονικών υλικών. Τα δύο τυπωμένα αποτελούνται από ένα επίπεδο γείωσης (Ground Plane), το οποίο είναι απαραίτητο για την μείωση του ηλεκτρικού θορύβου και των παρεμβολών. Επομένως, αποτρέπεται σε μεγάλο βαθμό, η αλλοίωση πληροφορίας στα ραδιοκύματα που στέλνονται ή λαμβάνονται. Κάποια 3D μοντέλα πάρθηκαν από τις ιστοσελίδες 3D Content Central, Snap EDA και Grab CAD.

Οι διαστάσεις της πλακέτας του Τηλεχειριστήριου Crobarm, είναι 13.5cm ύψους και 16cm πλάτους. Η έκταση της πλακέτας του Ηλεκτρονικού Συστήματος Crobarm, είναι μικρότερο με ύψος 10.6cm και 11cm πλάτους. Τα σχήματα 5.1, 5.2, 5.3, 5.4, αντικατοπτρίζουν το τρισδιάστατο σχήμα, που επρόκειτο να έχουν οι πλακέτες στην πραγματικότητα. Έπειτα, στα σχήματα 5.5, 5.6, ακολουθούν τα σχέδια σχηματικών, ώστε να γίνει η σύνδεση των ηλεκτρονικών υλικών μεταξύ τους.

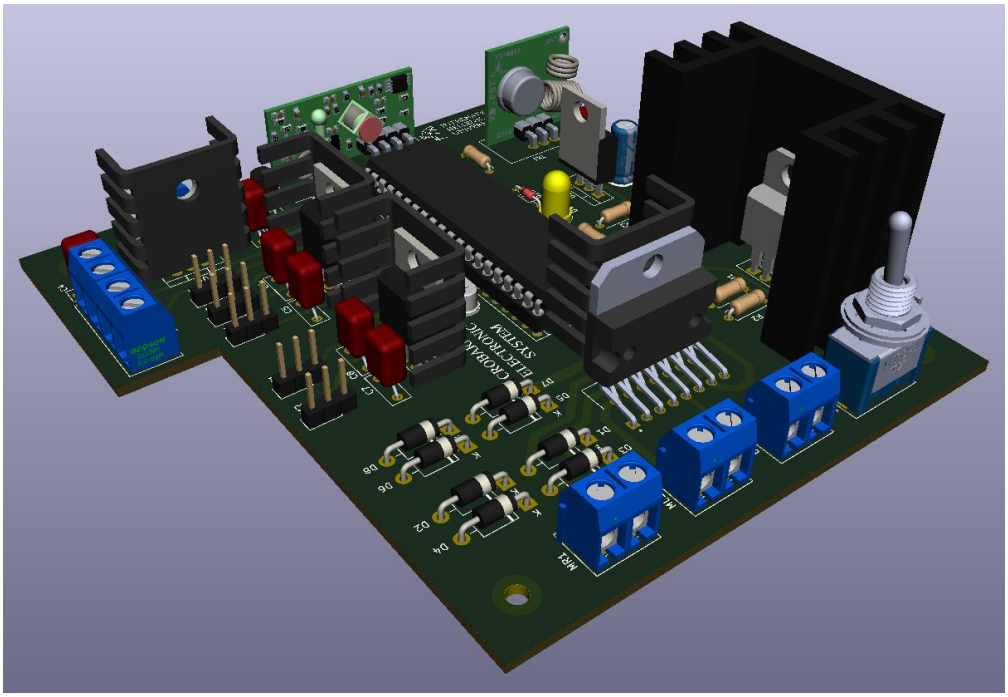
### 5.1.2 Πρακτική Εφαρμογή PCB

Στην συνέχεια τα σχήματα 5.7, 5.9, εκτυπώθηκαν από εκτυπωτή laser σε διαφάνειες μεγέθους A4, για να ακολουθήσει η διαδικασία κατασκευής των PCB. Αρχικά, κόπηκαν στα αντίστοιχα μεγέθη, δύο κομμάτια από φωτοευαίσθητη πλακέτα FR4, πάχους 1.57mm μονής όψης. Επόμενο βήμα, ήταν να εκτεθούν σε υπεριώδη ακτινοβολία UV. Η εκάστοτε πλακέτα ήταν ενωμένη, με τη διαφάνειά της για 10 λεπτά. Ύστερα, βουτήχτηκαν για 2 λεπτά περίπου σε tuboflo (περιέχει υδροξείδιο του νατρίου), με αποτέλεσμα να εμφανιστή το σχέδιο των πλακετών. Μετά, βουτήχτηκαν σε κεζάπ για 1 λεπτό περίπου (διάλυμα υδροχλωρικού οξέος), εμφανίζοντας το τελικό σχέδιο χαλκού-μόνωσης. Τέλος, με καθαρό οινόπνευμα αφαιρέθηκε η φωτοευαίσθητη μεμβράνη, ώστε να συγκολληθούν τα ηλεκτρονικά εξαρτήματα. Σημειώνεται, πως με τρυπάνι χειρός ανοίχτηκαν τρύπες βάσης διαμέτρου 3mm.

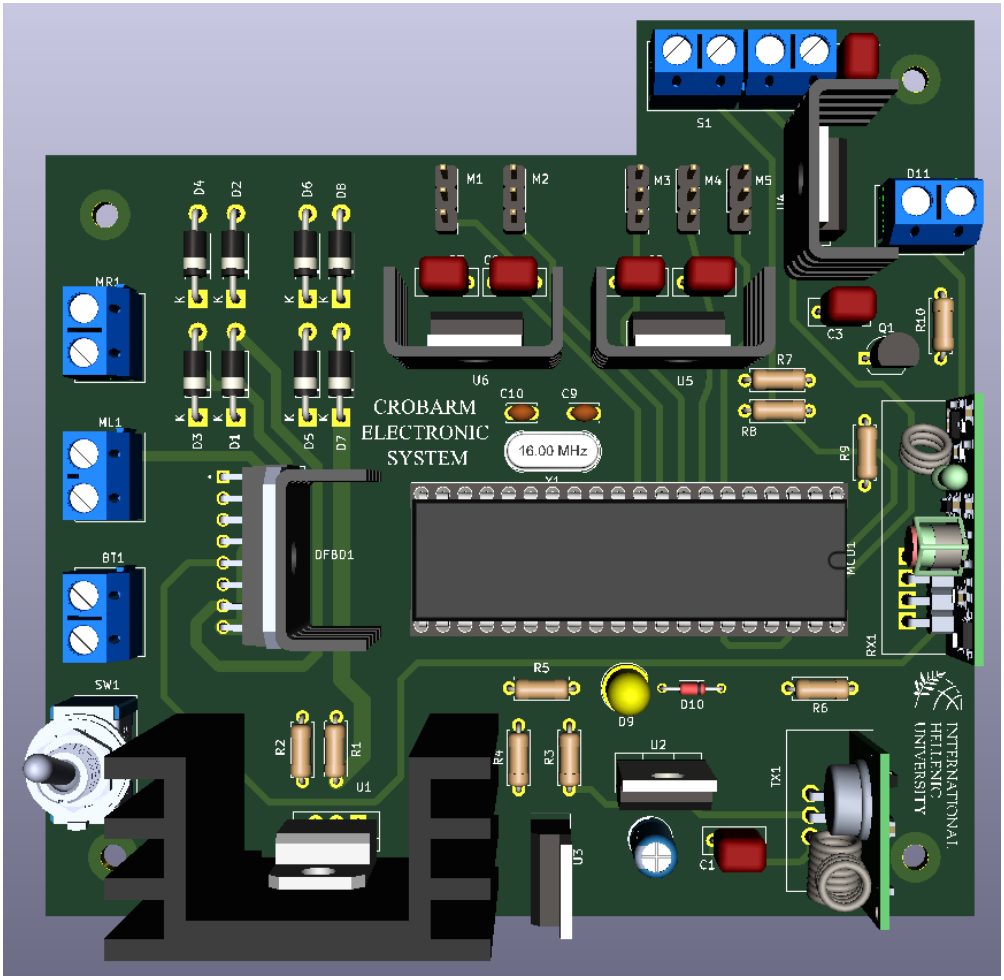
Από τα σχήματα 5.8, 5.10, του μπροστινού στρώματος χαλκού (Front Copper Layer) ,οι διαδρομές (Routes) χρησιμοποιήθηκαν ως χάλκινα καλώδια (Jumper Wires) και όχι σαν πλακέτα διπλής όψης. Όσον αφορά τα σχήματα 5.11, 5.12, υποδηλώνουν σε ποιο σημείο θα τοποθετηθεί το κάθε υλικό. Οι πίνακες 1, 2, 3, αναφέρουν όλα τα ηλεκτρονικά υλικά που χρησιμοποιήθηκαν. Στα σχήματα 5.13, 5.14, 5.15, φαίνεται το τελικό αποτέλεσμα από την διαδικασία κατασκευής, συγκόλλησης εξαρτημάτων και ολοκλήρωσης. Στο σχήμα 5.13 έγιναν αλλαγές, για να ταιριάζει με την καινούργια έκδοση από τα σχήματα 5.7, 5.9. Λόγο κοντινών αποστάσεων, ήταν δύσκολες οι συγκολλήσεις σε αρκετά σημεία.



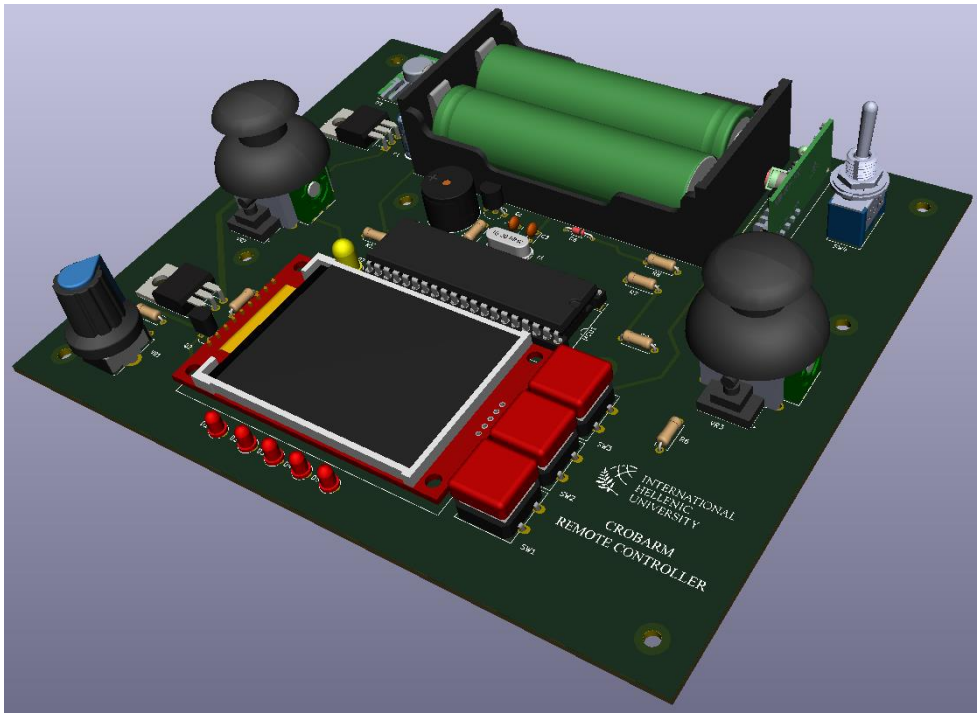

Σχήμα 5.1 Αποτύπωση Πρόσοψης Τηλεχειριστήριου Crobarm σε 3D Μορφή



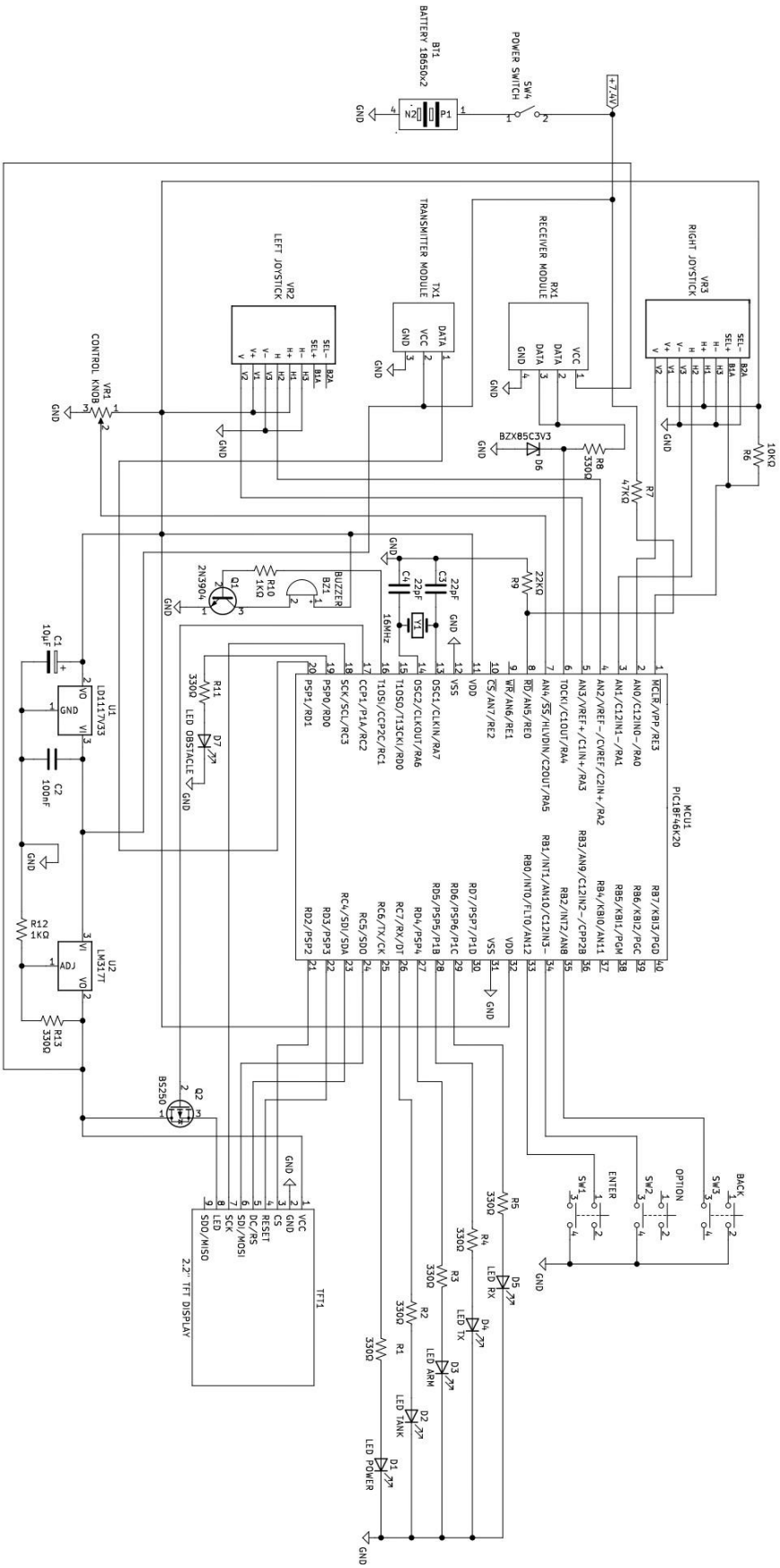
Σχήμα 5.2 Αποτύπωση Πλάγιας Όψης Ηλεκτρονικού Συστήματος Crobarm σε 3D



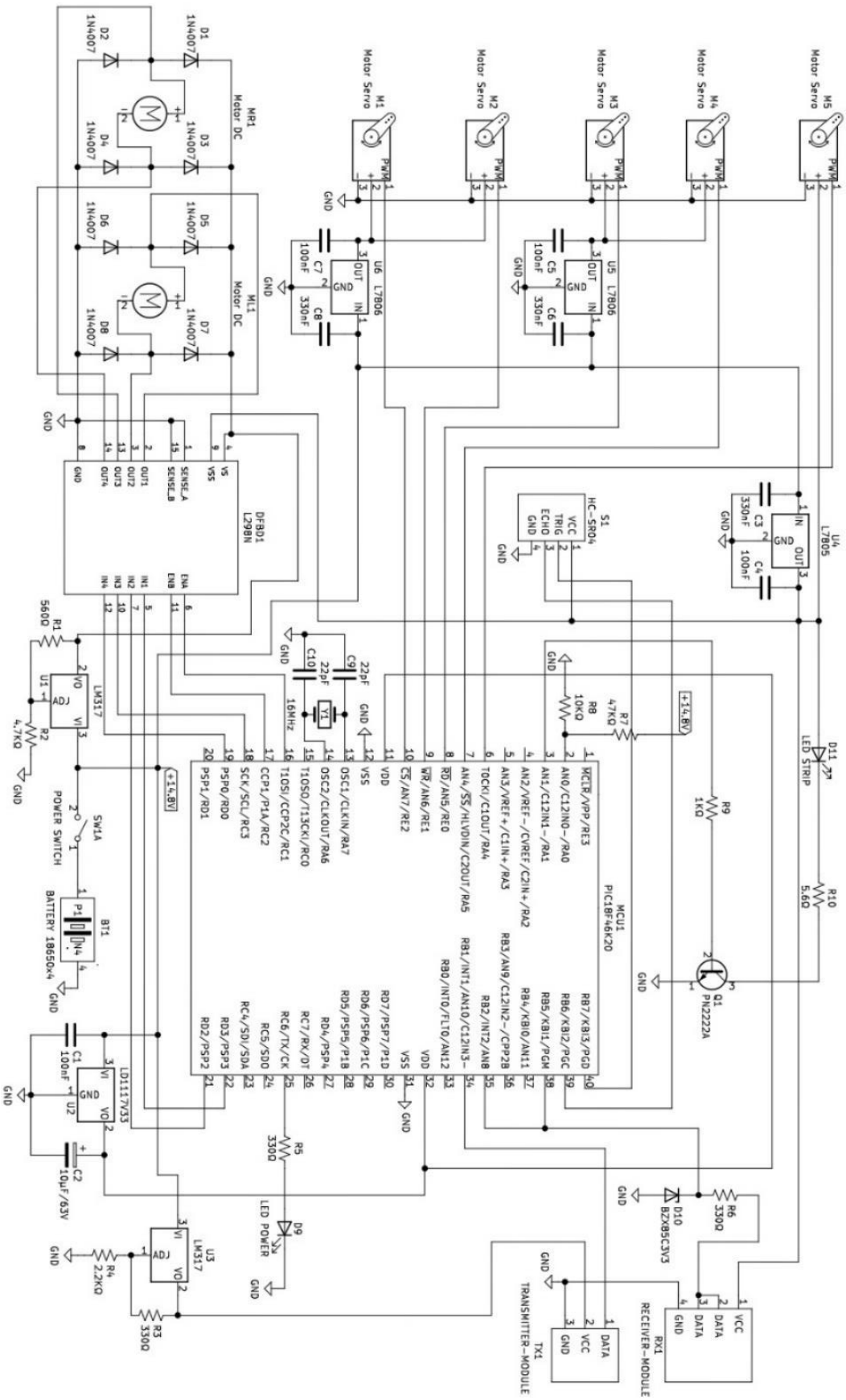

Σχήμα 5.3 Αποτύπωση Πρόσοψης Ηλεκτρονικού Συστήματος Crobarm σε 3D Μορφή



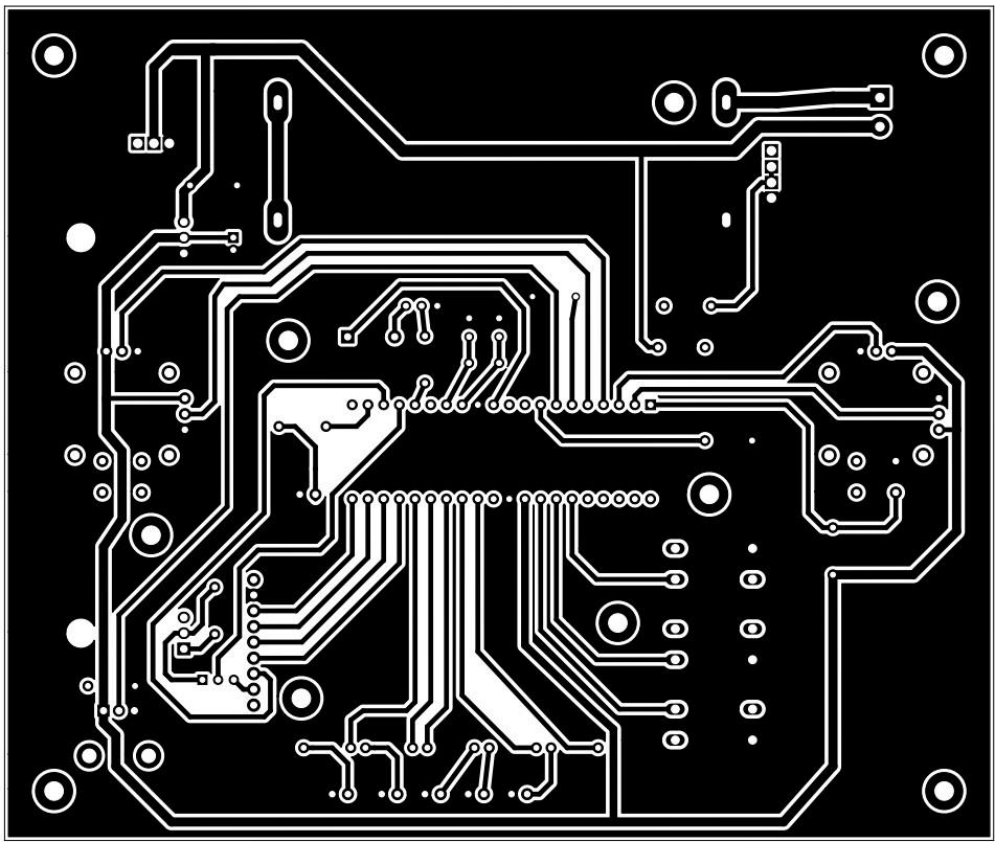

Σχήμα 5.4 Αποτύπωση Πλάγιας Όψης Τηλεχειριστήριου Crobarm σε 3D Μορφή



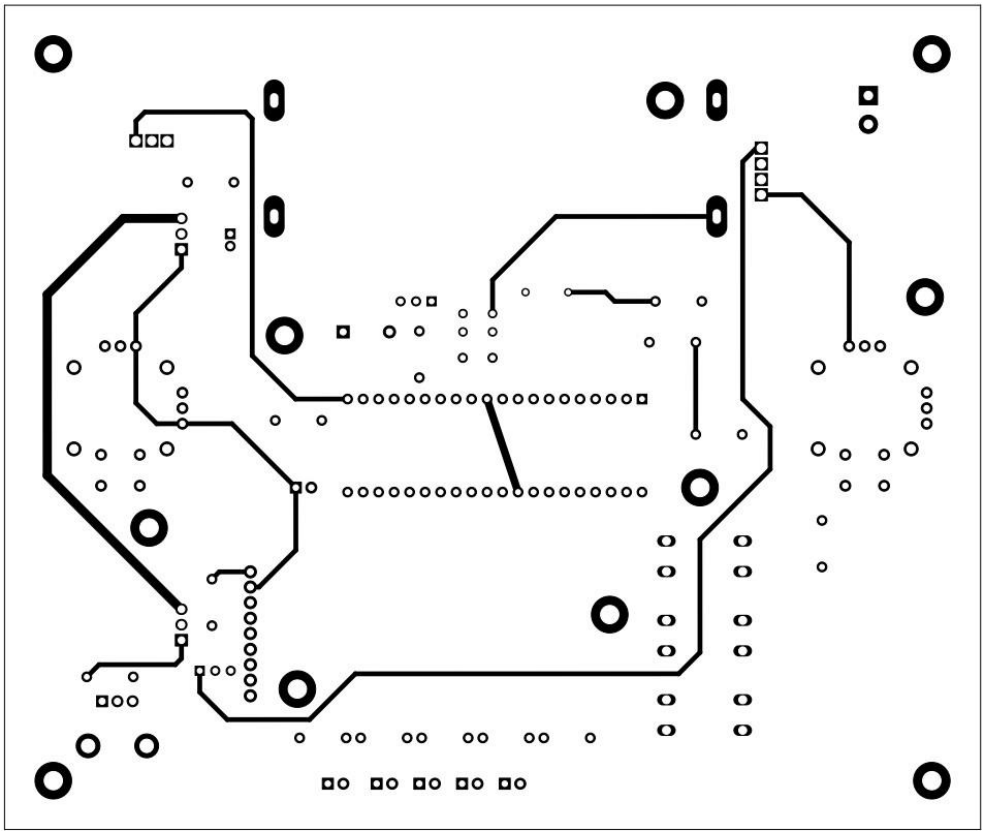

Σχήμα 5.5 Σχηματικό Τηλεχειριστήριου Crobarm



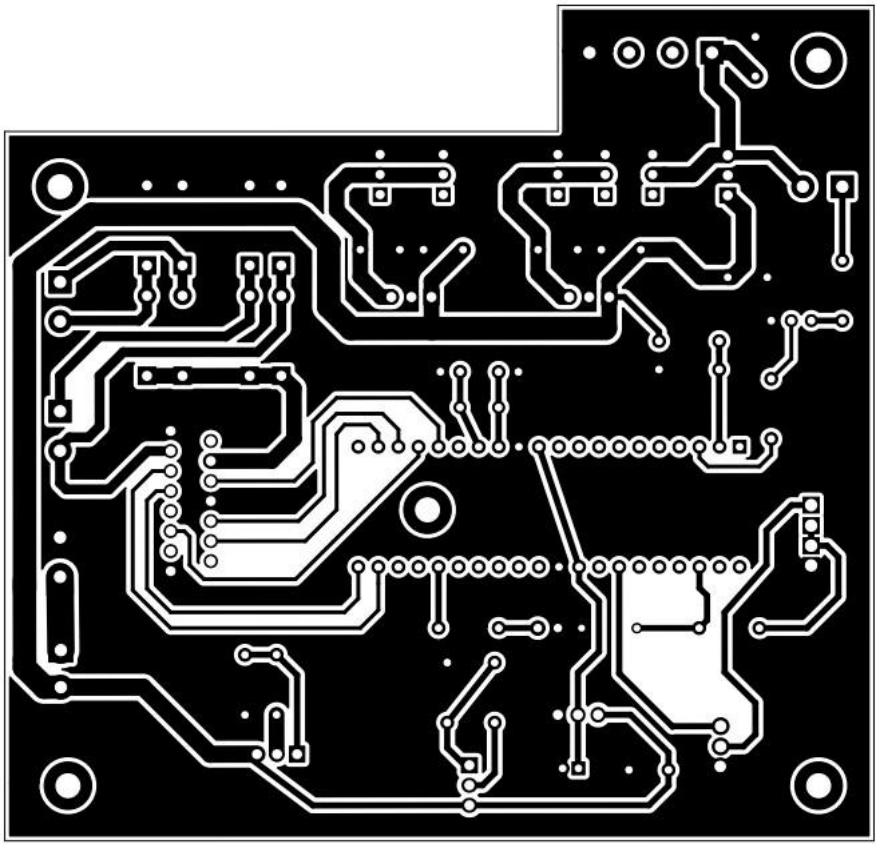

Σχήμα 5.6 Σχηματικό Ηλεκτρονικού Συστήματος Crobarm



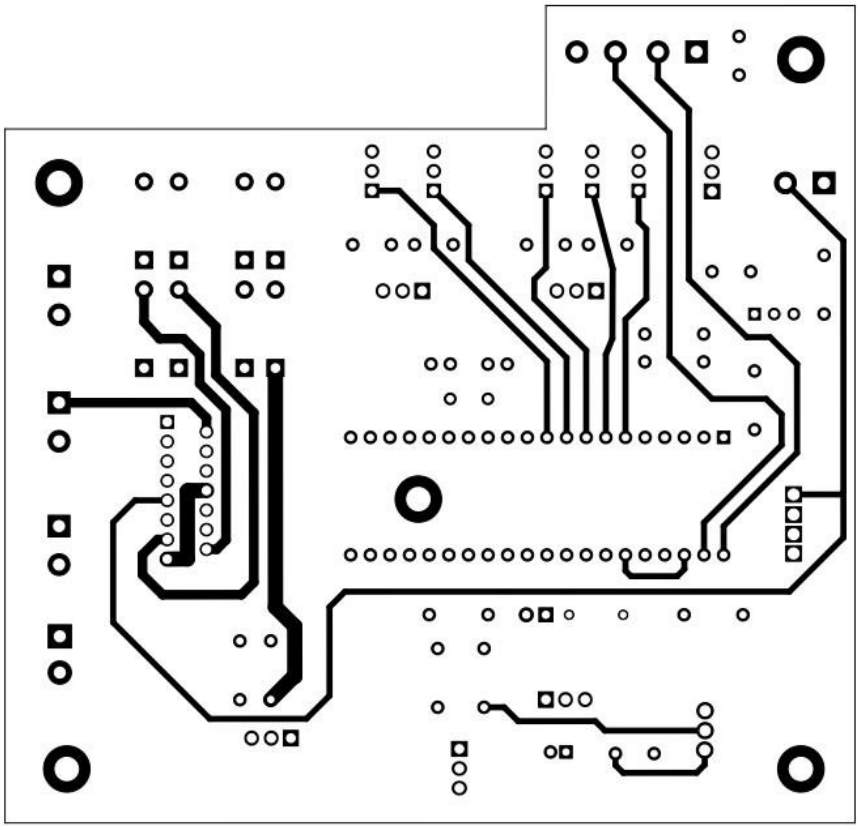
Σχήμα 5.7 Πίσω Στρώμα Χαλκού Τυπωμένου Τηλεχειριστήριου Crobarm



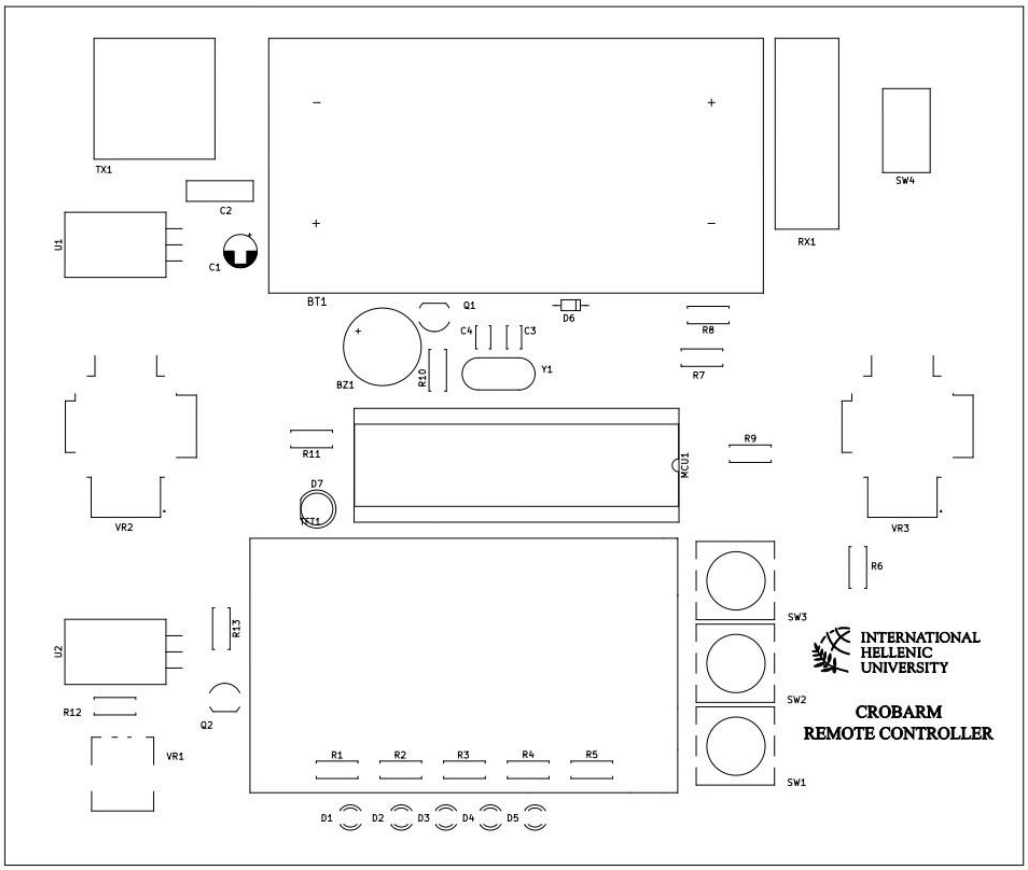
Σχήμα 5.8 Μπροστινό Στρώμα Χαλκού Τυπωμένου Τηλεχειριστήριου Crobarm



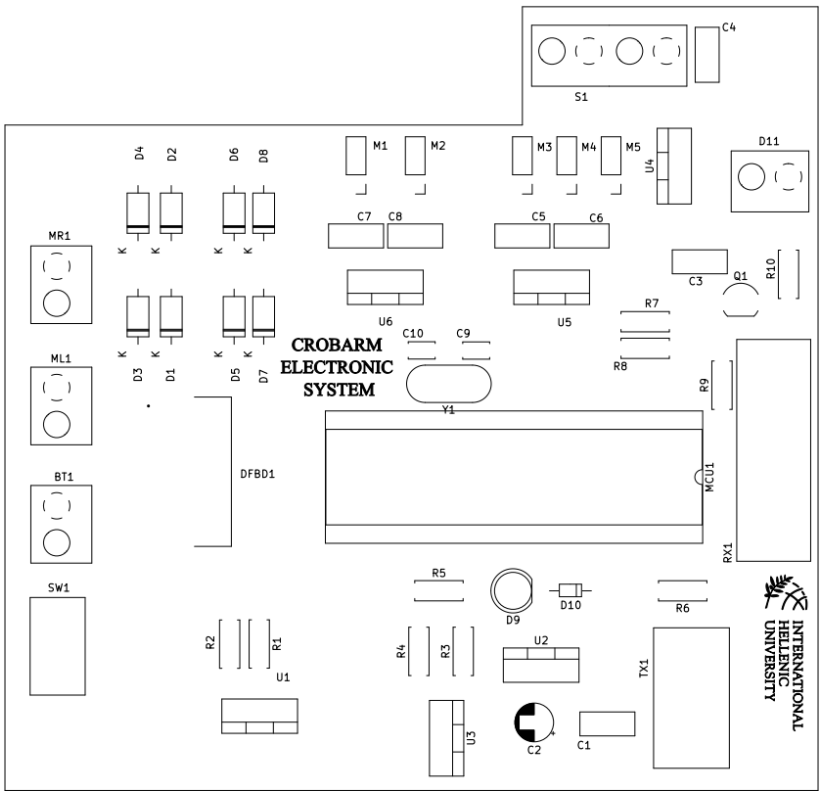
Σχήμα 5.9 Πίσω Στρώμα Χαλκού Τυπωμένου Ηλεκτρονικού Συστήματος Crobarm



Σχήμα 5.10 Μπροστινό Στρώμα Χαλκού Τυπωμένου Ηλεκτρονικού Συστήματος Crobarm




Σχήμα 5.11 Στρώμα Υλικών Τυπωμένου Τηλεχειριστήριου Crobarm




Σχήμα 5.12 Στρώμα Υλικών Τυπωμένου Ηλεκτρονικού Συστήματος Crobarm

| No | Symbol | Value |
|---|---|---|
| 1 | BT1 | BATTERY 18650 x 2 |
| 2 | BZ1 | BUZZER |
| 3 | C1 | 10μF |
| 4 | C2 | 100nF |
| 5 | C3 | 22pF |
| 6 | C4 | 22pF |
| 7 | D1 | LED POWER |
| 8 | D2 | LED TANK |
| 9 | D3 | LED ARM |
| 10 | D4 | LED TX |
| 11 | D5 | LED RX |
| 12 | D6 | BZX85C3V3 |
| 13 | D7 | LED OBSTACLE |
| 14 | MCU1 | PIC18F46K20 |
| 15 | Q1 | 2N3904 |
| 16 | Q2 | BS250 |
| 17 | R1 | 330Ω |
| 18 | R2 | 330Ω |
| 19 | R3 | 330Ω |
| 20 | R4 | 330Ω |
| 21 | R5 | 330Ω |
| 22 | R6 | 10KΩ |
| 23 | R7 | 47KΩ |
| 24 | R8 | 330Ω |
| 25 | R9 | 22KΩ |
| 26 | R10 | 1KΩ |
| 27 | R11 | 330Ω |
| 28 | R12 | 1KΩ |
| 29 | R13 | 330Ω |
| 30 | RX1 | RECEIVER MODULE |
| 31 | SW1 | ENTER |
| 32 | SW2 | OPTION |
| 33 | SW3 | BACK |
| 34 | SW4 | POWER SWITCH |
| 35 | TFT1 | 2.2" TFT DISPLAY |
| 36 | TX1 | TRANSMITTER MODULE |
| 37 | U1 | LD1117V33 |
| 38 | U2 | LM317T |
| 39 | VR1 | CONTROL KNOB |
| 40 | VR2 | LEFT JOYSTICK |
| 41 | VR3 | RIGHT JOYSTICK |
| 42 | Y1 | 16MHz |

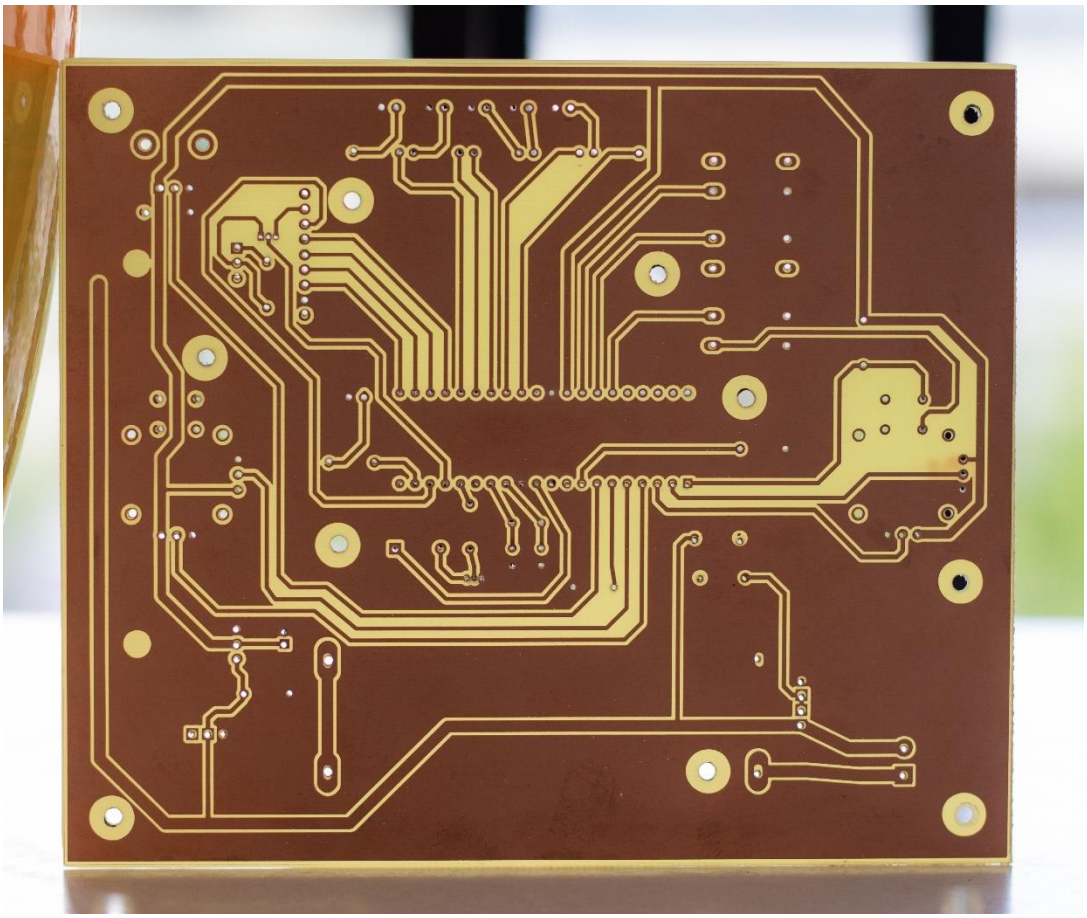
Πίνακας 2: Υλικά Τηλεχειριστήριου

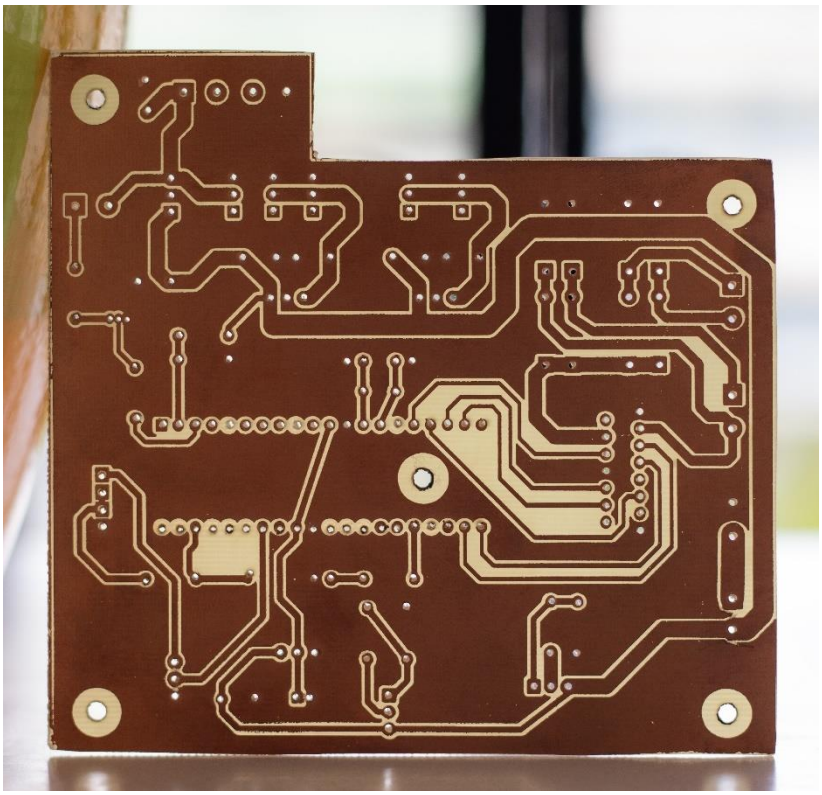
| |
|---|
| Το X συμβολίζει αριθμό. |
| CX: Το C είναι πυκνωτής. |
| DX: Το D είναι δίοδος ή LED. |
| RX: Το R είναι αντιστάτης 1/4W. |
| SW: Διακόπτης Tact ή Toggle. |
| YX: Κρύσταλλος. |
| MCU: Μικροελεγκτής. |
| QX: Τρανζίστορ BJT ή MOSFET. |
| UX: Σταθεροποιητής Τάσης. |

Πίνακας 1: Βοηθός Κατανόησης Υλικών

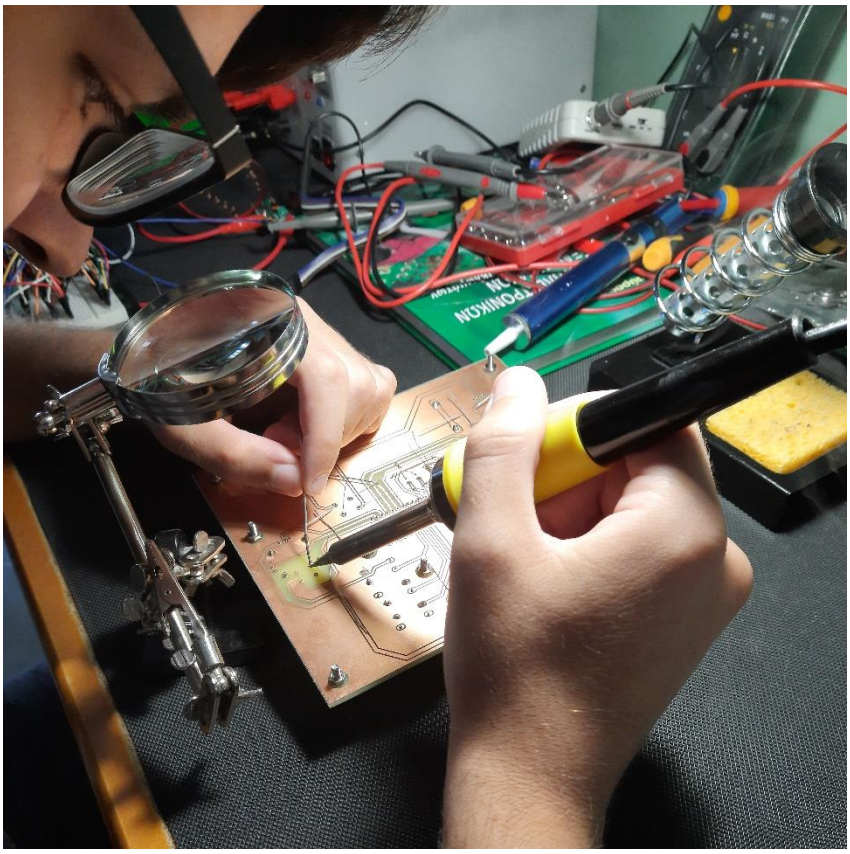
| No | Symbol | Value |
|---|---|---|
| 1 | BT1 | BATTERY 18650 x 4 |
| 2 | C1 | 100nF |
| 3 | C2 | 10μF/63V |
| 4 | C3 | 330nF |
| 5 | C4 | 100nF |
| 6 | C5 | 100nF |
| 7 | C6 | 330nF |
| 8 | C7 | 100nF |
| 9 | C8 | 330nF |
| 10 | C9 | 22pF |
| 11 | C10 | 22pF |
| 12 | D1 | 1N4007 |
| 13 | D2 | 1N4007 |
| 14 | D3 | 1N4007 |
| 15 | D4 | 1N4007 |
| 16 | D5 | 1N4007 |
| 17 | D6 | 1N4007 |
| 18 | D7 | 1N4007 |
| 19 | D8 | 1N4007 |
| 20 | D9 | LED POWER |
| 21 | D10 | BZX85C3V3 |
| 22 | D11 | LED STRIP |
| 23 | DFBD1 | L298N |
| 24 | M1 | MOTOR SERVO |
| 25 | M2 | MOTOR SERVO |
| 26 | M3 | MOTOR SERVO |
| 27 | M4 | MOTOR SERVO |
| 28 | M5 | MOTOR SERVO |
| 29 | MCU1 | PIC18F46K20 |
| 30 | ML1 | MOTOR DC |
| 31 | MR1 | MOTOR DC |
| 32 | Q1 | PN2222A |
| 33 | R1 | 560Ω |
| 34 | R2 | 4.7KΩ |
| 35 | R3 | 330Ω |
| 36 | R4 | 2.2KΩ |
| 37 | R5 | 330Ω |
| 38 | R6 | 330Ω |
| 39 | R7 | 51KΩ |
| 40 | R8 | 10KΩ |
| 41 | R9 | 1KΩ |
| 42 | R10 | 5.6Ω |
| 43 | RX1 | RECEIVER MODULE |
| 44 | S1 | HC-SR04 |
| 45 | SW1 | POWER SWITCH |
| 46 | TX1 | TRANSMITTER MODULE |
| 47 | U1 | LM317 |
| 48 | U2 | LD1117V33 |
| 49 | U3 | LM317 |
| 50 | U4 | L7805 |
| 51 | U5 | L7806 |
| 52 | U6 | L7806 |
| 53 | Y1 | 16MHz |

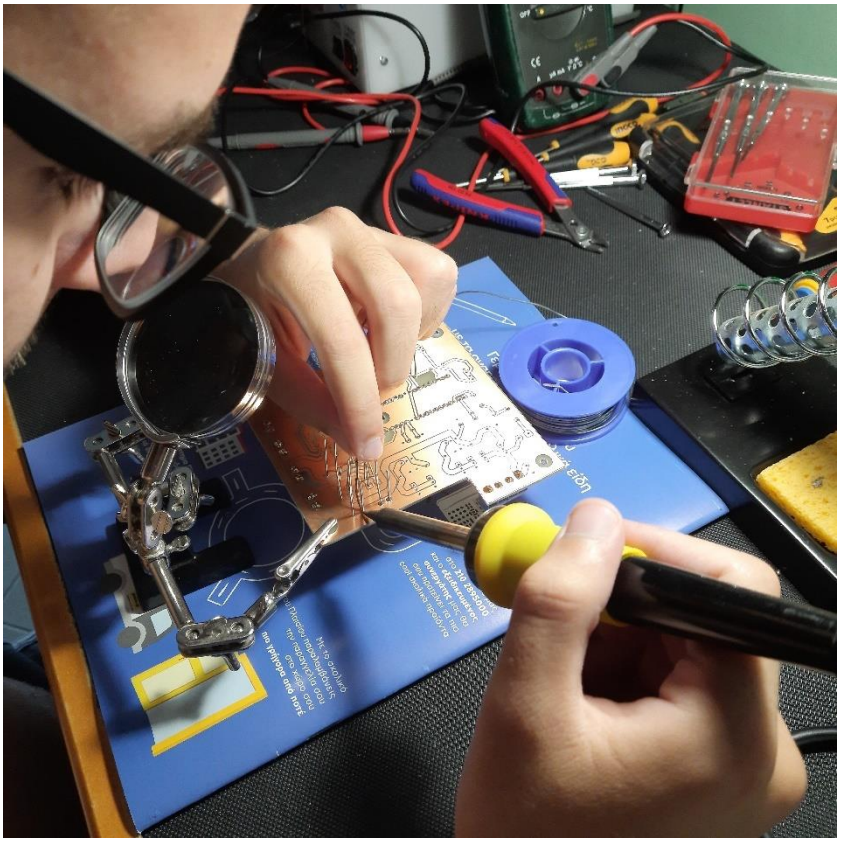
Πίνακας 2: Υλικά Ρομπότ



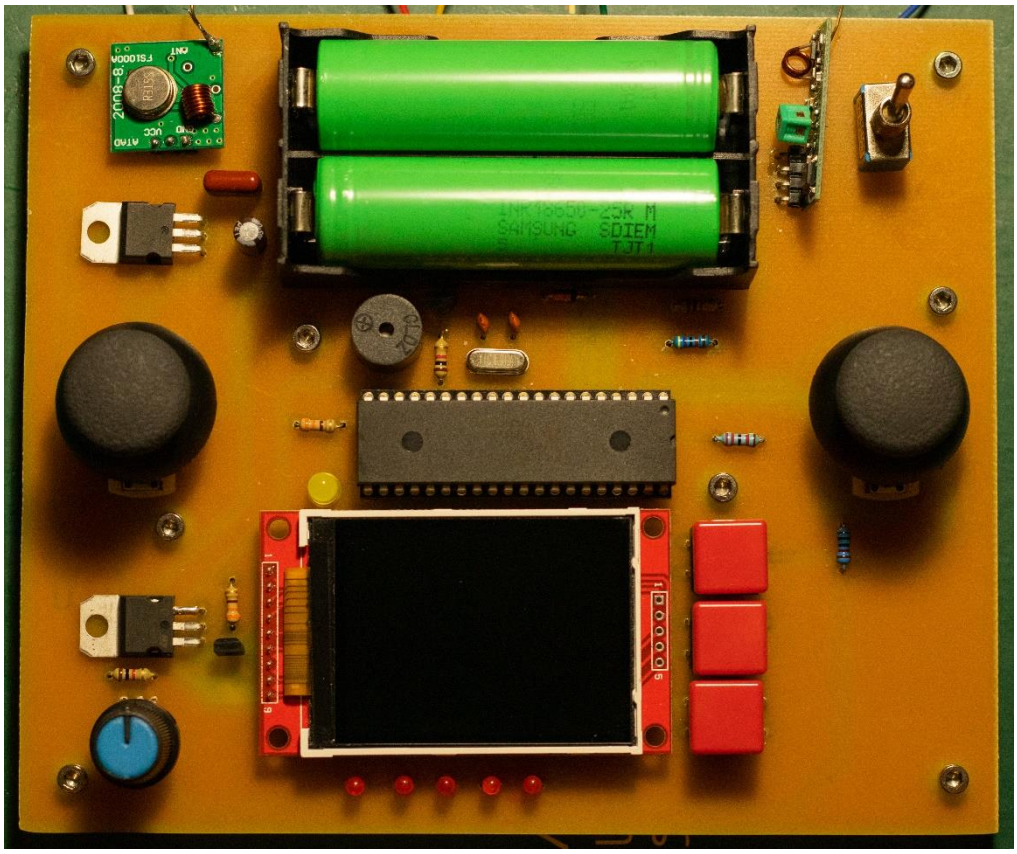


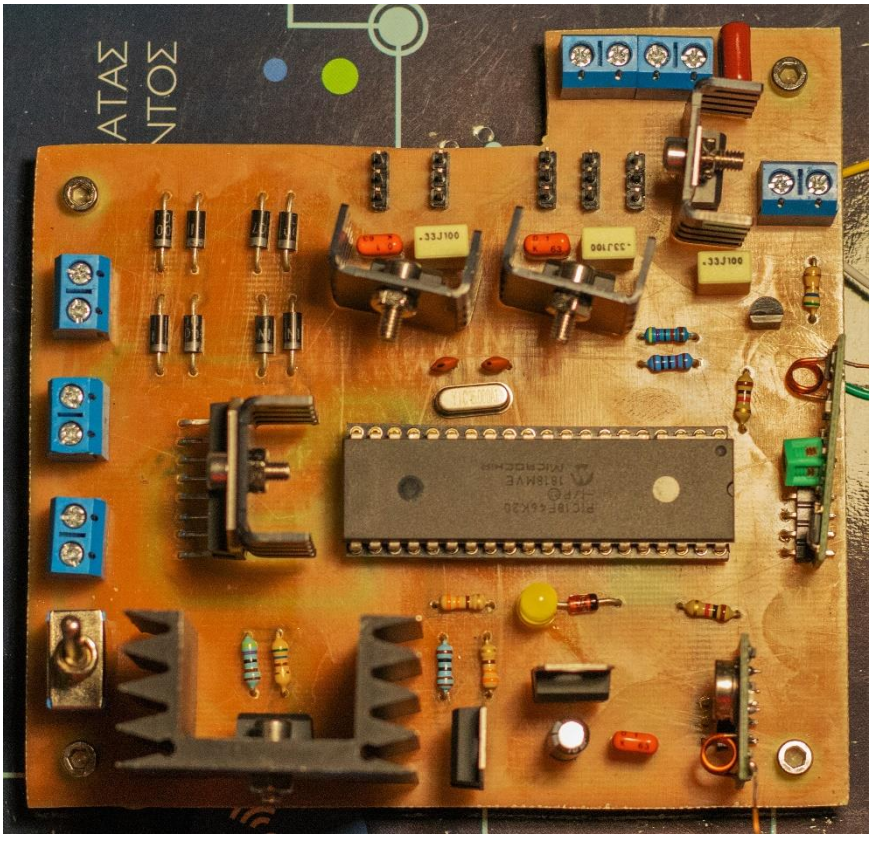
Σχήμα 5.13 Πλακέτες PCB για Τηλεχειριστήριο και Ρομπότ





Σχήμα 5.14 Συγκόλληση Υλικών στις Πλακέτες PCB για Τηλεχειριστήριο και Ρομπότ





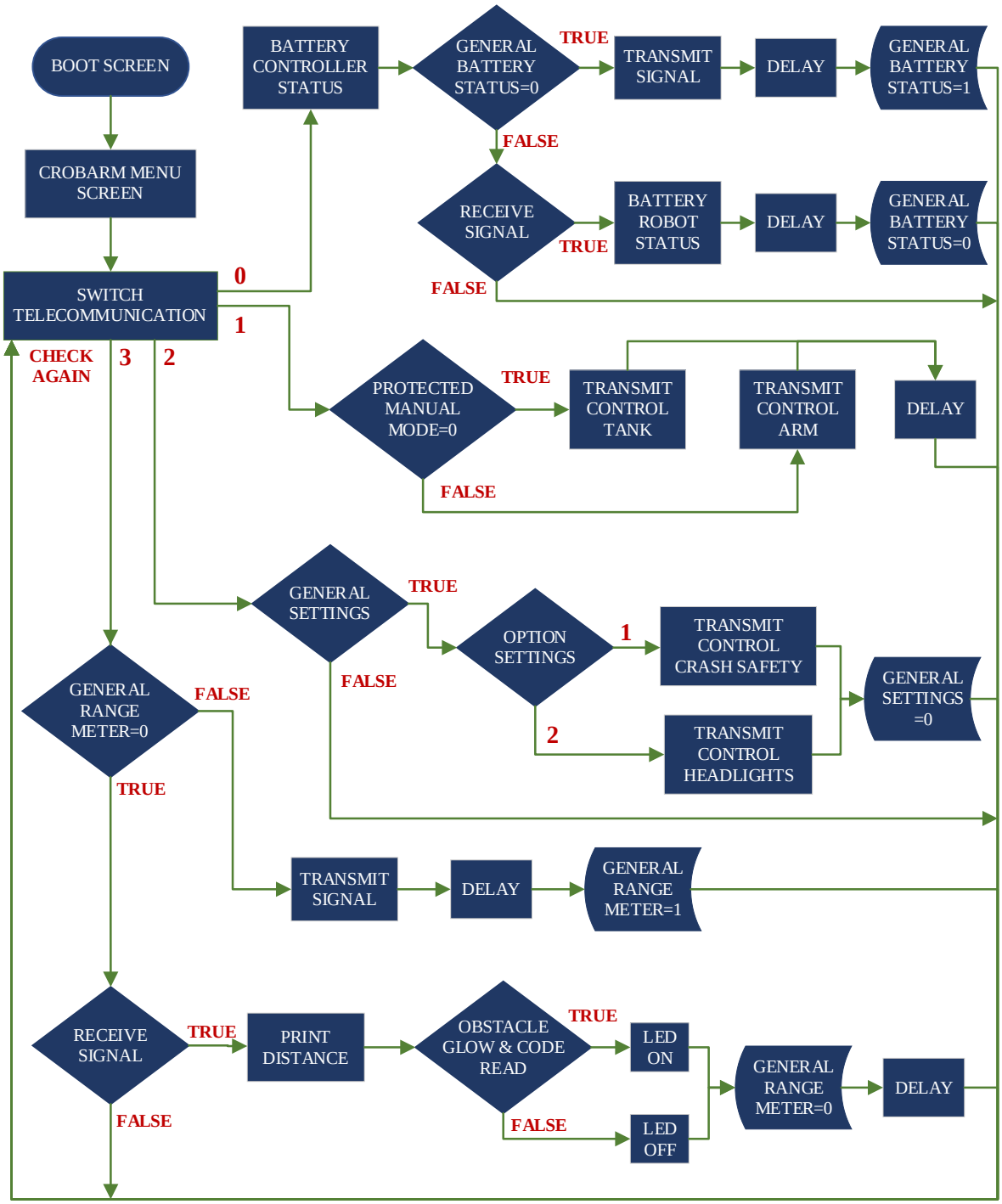
Σχήμα 5.15 Τελική Μορφή PCB για Τηλεχειριστήριο και Ρομπότ

## 5.2 Προγραμματισμός Ρομπότ και Τηλεχειριστηρίου

### 5.2.1 Επεξήγηση Λειτουργικότητας Κώδικα

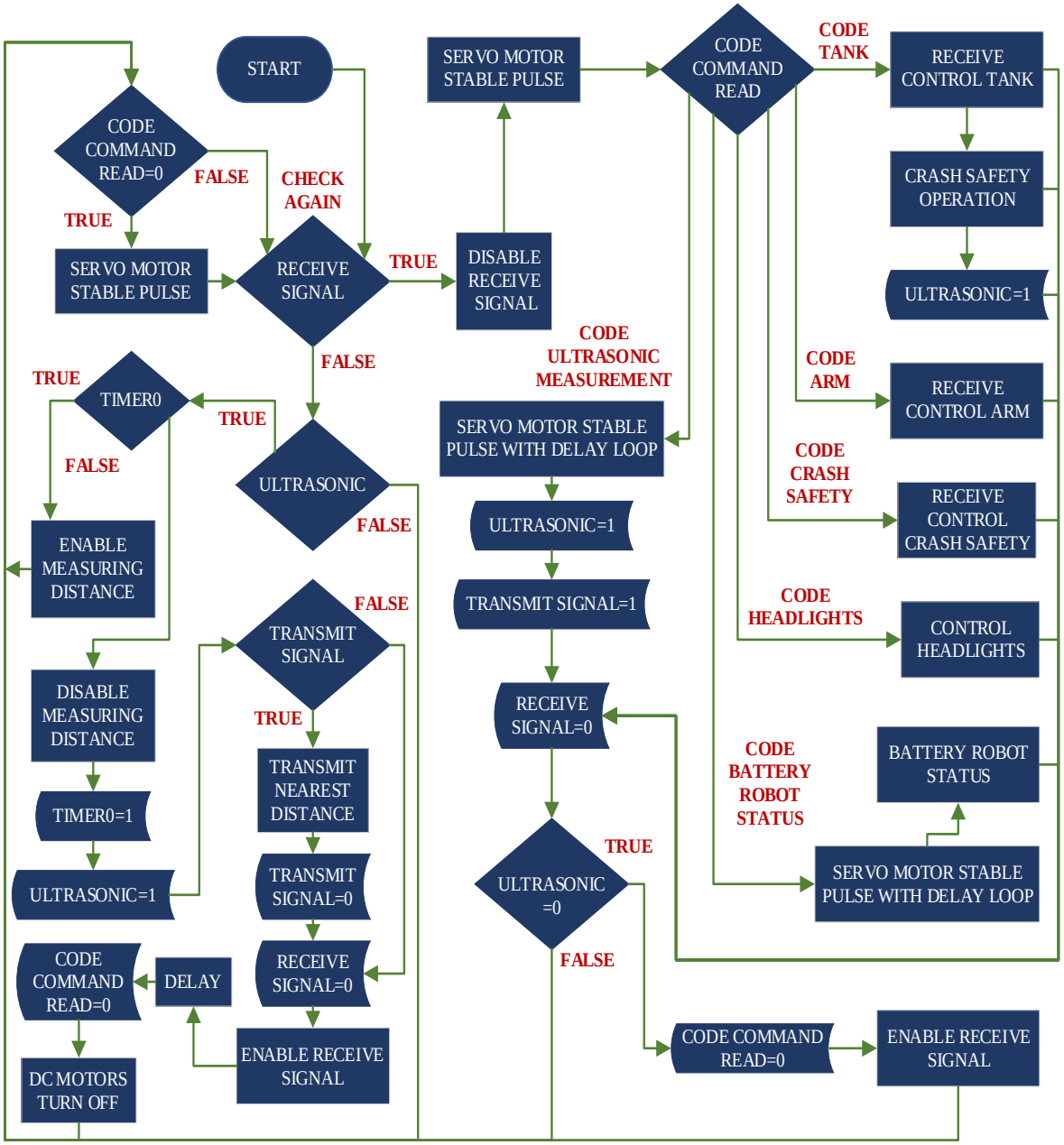
Ο κώδικας των μικροελεγκτών PIC, αναπτύχθηκε στο ολοκληρωμένου περιβάλλοντος ανάπτυξης MPLABX, έκδοσης 5.45, το οποίο δημιουργήθηκε από την εταιρία Microchip Technology. Κύριο εργαλείο είναι ο μεταγλωττιστής CCS C Compiler, έκδοσης 5.07, γράφοντας σε γλώσσα προγραμματισμού C αποτελούμενο με συναρτήσεις (Functions), εντολές προεπεξεργαστή (Preprocessor) και βιβλιοθήκες μικροελεγκτών (Head Files). Ο μεταγλωττιστής είναι απαραίτητος, για την μετατροπή εντολών από συμβολική γλώσσα (C στην προκειμένη περίπτωση), σε γλώσσα μηχανής (Machine Language) δηλαδή στο δυαδικό σύστημα 0 ή 1. Οι ολοκληρωμένοι κώδικες δεκαεξαδικής μορφής, εγγράφτηκαν στους δύο μικροελεγκτές, μέσω ενός προγραμματιστικού συστήματος (Programmer) ονόματι PICkit 3, με τη βοήθεια δικού του λογισμικού έκδοσης 3.1.


Σχήμα 5.16 Προγραμματιστής PICkit3 Φορτώνει το Κώδικα στον Μικροελεγκτή




Σχήμα 5.17 Διάγραμμα Ροής Κώδικα Τηλεχειριστήριου

Στα σχήματα 5.14, 5.15, εμφανίζονται τα διαγράμματα ροής, για την επανάληψη βρόγχου (While Loop) από τους δύο κώδικες. Συνοπτικά, περιγράφεται από τα ροογραφήματα η σειρά με την οποία γίνεται εκτέλεση των εντολών. Ασχέτως από την συνεχόμενη διενέργεια προγράμματος, ο κώδικας μικροελεγκτή τηλεχειριστήριου, με τη χρήση διακοπών διαχειρίζεται το μενού. Διαπιστώνεται από τα διαγράμματα πως δεν υπάρχει αμφίδρομη τηλεπικοινωνία σε κάθε διεργασία, παρά μόνο σε συγκεκριμένες περιπτώσεις, όπως είναι η μπαταρία του ρομπότ και η μέτρηση απόστασης.




Σχήμα 5.18 Διάγραμμα Ροής Κώδικα Ρομπότ

Στις επόμενες σελίδες, παρουσιάζονται ολόκληροι οι πηγαίοι κώδικες (Source Code) από τους δύο μικροελεγκτές. Είναι αρκετά μεγάλο το πλήθος των γραμμών τους, λόγω των πολυποίκιλων εντολών που εκτελούν αλλά και της πολυπλοκότητας που το διακατέχει, για την προσωρινή αποθήκευση κάθε ενέργειας. Όσον αφορά το πρόγραμμα για το τηλεχειριστήριο, γράφτηκαν 1462 γραμμές κώδικα και για το ρομπότ 916. Τα παραρτήματα περιλαμβάνουν τρείς πολύτιμες βιβλιοθήκες: δύο για την αρχικοποίηση ρυθμίσεων στο σύστημα των μικροελεγκτών και μια με συναρτήσεις, μεταβλητές που αφορούν την οθόνη. Επίσης, πρέπει να σημειωθεί πως οι βιβλιοθήκες ILI9341.h, GFX_Library.h, πάρθηκαν έτοιμες από το διαδίκτυο για την οδήγηση και τα γραφικά της οθόνης. Αξίζει να σημειωθεί ότι η βιβλιοθήκη TFT.h, αποτελείται από 1068 γραμμές κώδικα. Οι βιβλιοθήκες PIC18F46K20.h έχουν 37 γραμμές κώδικα με εντολές προεπεξεργαστή.

### 5.2.2 Κώδικας Μικροελεγκτή Τηλεχειριστήριου

```
#include <PIC18F46K20.h>
#include <TFT.h>

#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#use fast_io(D)
#use fast_io(E)

#byte PORTA=0xF80
#byte PORTB=0xF81
#byte PORTC=0xF82
#byte PORTD=0xF83
#byte PORTE=0xF84

#define VDC 3.345

//Set a coded command which is going to be transmitted wireless
void transmitSignal(unsigned long codedCommandSend);

//Transmits a coded command containing control for the Tank Operation.
//Movements: Forward, Backwards, Right, Left.
//Speed: Slow, Normal, Fast
void transmitControlTank(void);

//Transmits a coded command containing control for the Arm Operation
//Speed: Slow, Normal.
void transmitControlArm(void);

//Transmits a coded command containing control for the Headlights
//LEd strip is tuned ON or OFF
void transmitControlHeadlights(void);

//Transmits a coded command containing control for the Crash Safety
//Crash Safety is turned ON or OFF
void transmitControlCrashSafety(void);

//A signal is received containing a coded command which is stored temporary
short receiveSignal(void);

//The temporary coded command is checked for matching with an array value in cm
int receiveNearestDistance(void);

//Buzzer produces a noise in three different tones according the 2 variables
void buzzerSystemSound(long setSemiPeriodMicroseconds,long maxSemiPeriodValue);

//Control the brightness of the screen: 20%, 40%, 60%, 80%, 100%
void controlBrightnessDisplay(void);

//Remote control's battery life is being displayed on the screen in percentage
void batteryRemoteControlStatus(void);
```

```
//Robot's battery life is being displayed on the screen in percentage
void batteryRobotStatus(void);

//Once Option is pressed in Main Menu, the cursor is changing position between the
//four Sub Menus that are available
void optionMainMenu(void);

//Once Option is pressed in Manual Mode, the cursor is changing position between the
//four mechanisms that are available
void optionManualMode(void);

//Once Back is pressed in Manual Mode, the sub menu of Manual Mode is deleted
//returning to the Main Menu
void backManualMode(void);

//Once Enter is pressed in Manual Mode, the sub menu of Manual Mode is displayed on screen
void enterManualMode(void);

//Once Option is pressed in Settings, the cursor is changing position between the
//four mechanisms that are available
void optionSettings(void);

//Once Back is pressed in Settings, the sub menu of Settings is deleted
//returning to the Main Menu
void backSettings(void);

//Once Enter is pressed in Settings, the sub menu of Settings is displayed on screen
void enterSettings(void);

//Once Enter is pressed in Range Meter, the mechanism Obstacle Glow is changing
//status ON or OFF
void optionRangeMeter(void);

//Once Back is pressed in RangeMeter, the sub menu of Range Meter is deleted
//returning to the Main Menu
void backRangeMeter(void);

//Once Enter is pressed in RangeMeter, the sub menu of Range Meter is displayed on screen
void enterRangeMeter(void);

//Displays in cm the distance between robot and obstacle
void printDistance(void);

//Once Enter is pressed in About Thesis, a text about thesis is displayed on the screen
void enterAboutThesis(void);

short flagEnableButtons=1,flagButtonEnter=0,flagButtonBack=0;
short flagTransmitStop=0,flagReceiveSignal=0,generalBatteryStatus=0;
unsigned long adjustBrightness=0,switchSubMenu=0;
int flagOptionMainMenu=1,switchOptionMainMenu=1,lastOptionMainMenu=1;
int lastOptionManualMode=1,keepLastOptionManualMode=1,switchOptionManualMode=1;
short flagManualModeEnter=0,flagLastManualMode=1,flagProtectedManualMode=0;
short flagSettingsEnter=0,flagLastSettings=1,wrongValue=0;
int lastOptionSettings=1,switchOptionSettings=1,switchTelecommunication=0;
```

```
short flagDistance=0,flagRangeMeterEnter=0,generalRangeMeter=0;

void main(void)
{
   set_tris_a(0xFF);
   set_tris_b(0x07);
   set_tris_c(0x00);
   set_tris_d(0x00);
   set_tris_e(0x01);

   port_b_pullups(0x07);

   setup_adc(ADC_CLOCK_DIV_64|ADC_TAD_MUL_20);
   setup_adc_ports(sAN0|VSS_VDD);
   setup_adc_ports(sAN1|VSS_VDD);
   setup_adc_ports(sAN2|VSS_VDD);
   setup_adc_ports(sAN3|VSS_VDD);
   setup_adc_ports(sAN4|VSS_VDD);
   setup_adc_ports(sAN5|VSS_VDD);

   setup_timer_2(T2_DIV_BY_16,249,1);
   setup_ccp1(CCP_PWM);
   set_pwm1_duty(adjustBrightness);

   tft_begin();
   bootScreen();
   menuScreen();

   output_high(PIN_C6);
   output_high(PIN_C7);

   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);

   ext_int_edge(0,L_TO_H);
   ext_int_edge(1,L_TO_H);
   ext_int_edge(2,L_TO_H);

   enable_interrupts(GLOBAL);

   while(TRUE)
   {
      switch(switchTelecommunication)
      {
         case 0:
            batteryRemoteControlStatus();
            if(!generalBatteryStatus)
            {
               transmitSignal(0xAAA);
               delay_ms(800);
               generalBatteryStatus=1;
             }
            else
```

```
    {
      if(receiveSignal())
      {
        batteryRobotStatus();
        delay_ms(1000);
        generalBatteryStatus=0;
      }
    }
    break;

  case 1:
    if(!flagProtectedManualMode)
    {
      transmitControlTank();
    }
    else
    {
      transmitControlArm();
    }
    delay_ms(2);
    break;

  case 2:
    if(generalSettings)
    {
      if(lastOptionSettings==1)
      {
        transmitControlCrashSafety();
      }
      else if(lastOptionSettings==2)
      {
        transmitControlHeadlights();
      }
      generalSettings=0;
    }
    break;

  case 3:
    if(!generalRangeMeter)
    {
      transmitSignal(0xAAC);
      delay_ms(800);
      generalRangeMeter=1;
    }
    else
    {
      if(receiveSignal())
      {
        printDistance();

        if((!lastObstacleGlow)&&(codedCommandRead!=0xA96))
        {
          output_high(PIN_D0);
        }
```

```
                else
                {
                   output_low(PIN_D0);
                }
                generalRangeMeter=0;
                delay_ms(1000);
             }
          }
          break;
      }
   }
}

#INT_EXT2
void pressButtonBack(void)
{
   output_low(PIN_D5);
   output_low(PIN_D6);
   output_low(PIN_D1);
   flagTransmitStop=1;
   generalBatteryStatus=0;
   generalRangeMeter=0;
   buzzerSystemSound(179,1117);

   if(flagButtonBack)
   {
      switch(lastOptionMainMenu)
      {
         case 1:
            backManualMode();
            break;
         case 2:
            backSettings();
            break;
         case 3:
            backRangeMeter();
            break;
         case 4:
            deleteAboutThesis();
            flagEnterAboutThesis=1;
            break;
      }
      switchSubMenu=1;
      flagButtonBack=0;
      printMainMenu();
      optionMainMenu();
      switchTelecommunication=0;
   }
}

#INT_EXT1
void pressButtonOption(void)
{
```

```
   output_low(PIN_D5);
   output_low(PIN_D6);
   flagTransmitStop=1;
   generalBatteryStatus=0;
   generalRangeMeter=0;
   buzzerSystemSound(160,1250);
   display_setTextSize(2);

   switch(switchSubMenu)
   {
      case 1:
         optionMainMenu();
         break;
      case 2:
         optionManualMode();
         break;
      case 3:
         optionSettings();
      default:
         break;
   }

   if(!switchSubMenu)
   {
      switchSubMenu=1;
      optionMainMenu();
   }
}

#INT_EXT
void pressButtonEnter(void)
{
   output_low(PIN_D5);
   output_low(PIN_D6);
   flagTransmitStop=1;
   wrongValue=0;
   keepRobotBattery=1;
   generalBatteryStatus=0;
   generalRangeMeter=0;
   buzzerSystemSound(125,1600);

   flagButtonEnter=1;
   flagEnterAboutThesis=flagEnterAboutThesis;

   if(!flagEnableButtons)
   {
      switch(lastOptionMainMenu)
      {
         case 1:
            enterManualMode();
            break;

         case 2:
```

```
            enterSettings();
            break;
         case 3:
            enterRangeMeter();
            break;
         case 4:
            enterAboutThesis();
            break;
      }
      flagButtonBack=1;

   }
   flagEnableButtons=0;
}

void transmitSignal(unsigned long codedCommandSend)
{
   unsigned int bitCode=0;

   output_high(PIN_D5);

   output_high(PIN_D1);
   delay_ms(2);
   output_low(PIN_D1);
   delay_ms(2);

   for(bitCode=0;bitCode<=11;bitCode++)
   {
      if(bit_test(codedCommandSend,(11-bitCode)))
      {
         output_high(PIN_D1);
         delay_us(500);
         output_low(PIN_D1);
         delay_us(500);
      }
      else
      {
         output_high(PIN_D1);
         delay_us(500);
         output_low(PIN_D1);
         delay_ms(1);
      }
   }
   output_high(PIN_D1);
   delay_ms(1);
   output_low(PIN_D1);

   output_low(PIN_D5);
}

void transmitControlTank(void)
{
   float inputAnalogVoltage=0;
```

```
int movementTank=3,speedTank=0,flagInaction=0;
unsigned long codedCommandTank[6][6]=
{
   {0x35A,0x35C,0x363,0x365,0x366,0x369},
   {0x36A,0x36C,0x393,0x395,0x396,0x399},
   {0x39A,0x39C,0x3A3,0x3A5,0x3A6,0x3A9},
   {0x3AA,0x3AC,0x3C3,0x3C5,0x3C6,0x3C9},
   {0x3CA,0x3CC,0x533,0x535,0x536,0x539},
   {0x53A,0x53C,0x553,0x555,0x556,0x559}
};

  switch(lastTankSpeed)
  {
     case 1:
        speedTank=5;
        break;
     case 2:
        speedTank=3;
        break;
     case 3:
        speedTank=1;
        break;
  }

  do{
     set_adc_channel(movementTank);
     inputAnalogVoltage=100*((float)read_adc()*VDC)/1023.0;

     (movementTank==3)?(movementTank=5):(movementTank=6);

     if(18>=inputAnalogVoltage)
     {
        transmitSignal(codedCommandTank[movementTank-speedTank][0]);
        break;
     }
     else if((inputAnalogVoltage>=19)&&(inputAnalogVoltage<=89))
     {
        transmitSignal(codedCommandTank[movementTank-speedTank][1]);
        break;
     }
     else if((inputAnalogVoltage>=90)&&(inputAnalogVoltage<=161))
     {
        transmitSignal(codedCommandTank[movementTank-speedTank][2]);
        break;
     }
     else if((inputAnalogVoltage>=162)&&(inputAnalogVoltage<=174))
     {
        ++flagInaction;
        if(flagInaction==2)

        {
           transmitSignal(0x55A);
           break;
```

```
            }
        }
        if((inputAnalogVoltage>=175)&&(inputAnalogVoltage<=250))
        {
            transmitSignal(codedCommandTank[movementTank-speedTank][3]);
            break;
        }
        else if((inputAnalogVoltage>=251)&&(inputAnalogVoltage<=324))
        {
            transmitSignal(codedCommandTank[movementTank-speedTank][4]);
            break;
        }
        else if((inputAnalogVoltage>=325)&&(inputAnalogVoltage<=340))
        {
            transmitSignal(codedCommandTank[movementTank-speedTank][5]);
            break;
        }
        movementTank=1;

    }while(movementTank<7);
}

void transmitControlArm(void)
{
    float inputAnalogVoltage=0;
    int flagInaction=0,rotationArm=0,speedArm=0;
    unsigned long codedCommandArm[10][6]=
    {
        {0x55C,0x563,0x565,0x566,0x569,0x56A},
        {0x56C,0x593,0x595,0x596,0x599,0x59A},
        {0x59C,0x5A3,0x5A5,0x5A6,0x5A9,0x5AA},
        {0x5AC,0x5C3,0x5C5,0x5C6,0x5C9,0x5CA},
        {0x5CC,0x633,0x635,0x636,0x639,0x63A},
        {0x63C,0x653,0x655,0x656,0x659,0x65A},
        {0x65C,0x663,0x665,0x666,0x669,0x66A},
        {0x66C,0x693,0x695,0x696,0x699,0x69A},
        {0x69C,0x6A3,0x6A5,0x6A6,0x6A9,0x6AA},
        {0x6AC,0x6C3,0x6C5,0x6C6,0x6C9,0x6CA}
    };

    (!lastArmSpeed)?(speedArm=0):(speedArm=5);

    while(rotationArm<5)
    {
        set_adc_channel(rotationArm);
        inputAnalogVoltage=100*((float)read_adc()*VDC)/1023.0;


        if(33>inputAnalogVoltage)
        {
            transmitSignal(codedCommandArm[rotationArm+speedArm][0]);
            break;
        }
```

```
      else if(92>inputAnalogVoltage)
      {
        transmitSignal(codedCommandArm[rotationArm+speedArm][1]);
        break;
      }
      else if(152>=inputAnalogVoltage)
      {
        transmitSignal(codedCommandArm[rotationArm+speedArm][2]);
        break;
      }
      else if((178>inputAnalogVoltage)&&(inputAnalogVoltage>152))
      {
        ++flagInaction;

        if(flagInaction==5)
        {
          transmitSignal(0x6CC);
          break;
        }
      }
      else if(238>inputAnalogVoltage)
      {
        transmitSignal(codedCommandArm[rotationArm+speedArm][3]);
        break;
      }
      else if(298>inputAnalogVoltage)
      {
        transmitSignal(codedCommandArm[rotationArm+speedArm][4]);
        break;
      }
      else if(340>=inputAnalogVoltage)
      {
        transmitSignal(codedCommandArm[rotationArm+speedArm][5]);
        break;
      }
      rotationArm++;
    }
}

void transmitControlHeadlights(void)
{
   switch(lastHeadlights)
   {
      case 0:
        transmitSignal(0x333);
        break;
      case 1:
        transmitSignal(0x335);

        break;
   }
}
```

```
void transmitControlCrashSafety(void)
{
   switch(lastCrashSafety)
   {
      case 0:
         transmitSignal(0x339);
         break;
      case 1:
         transmitSignal(0x33A);
         break;
   }
}

short receiveSignal(void)
{
   unsigned int bitCode=0;
   checkProtocolPart=1;
   countBinaryCode=0;
   output_high(PIN_D6);

   switch(checkProtocolPart)
   {
      case 1:
         codedCommandRead=0;

         while((input(PIN_A4))&&(countBinaryCode<=16))
         {
            delay_us(200);
            countBinaryCode++;
         }

         if((countBinaryCode>=13)&&(countBinaryCode<=7))
         {
            countBinaryCode=0;
            return FALSE;
         }
         else
         {
            countBinaryCode=0;
         }

         while((!input(PIN_A4))&&(countBinaryCode<=16))
         {
            delay_us(200);
            countBinaryCode++;
            if(countBinaryCode==10)
            {
               break;
            }

         }

         if((countBinaryCode>=13)&&(countBinaryCode<=7))
```

```
    {
      countBinaryCode=0;
      return FALSE;
    }
    else
    {
      countBinaryCode=0;
      checkProtocolPart=2;
    }
    continue;

  case 2:
    for(bitCode=0;bitCode<=11;bitCode++)
    {
      while((input(PIN_A4))&&(countBinaryCode<=10))
      {
        countBinaryCode++;
        delay_us(100);
      }

      if((countBinaryCode>=8))
      {
        countBinaryCode=0;
        return FALSE;
      }
      else
      {
        countBinaryCode=0;
      }

      while((!input(PIN_A4))&&(countBinaryCode<=20))
      {
        countBinaryCode++;
        delay_us(100);
      }

      if((countBinaryCode>=18))
      {
        countBinaryCode=0;
        return FALSE;
      }

      if(8>=countBinaryCode)
      {
        countBinaryCode=0;
        bit_set(codedCommandRead,(11-bitCode));
      }
      else
      {
        countBinaryCode=0;

        bit_clear(codedCommandRead,(11-bitCode));
      }
```

```
}
        checkProtocolPart=3;
        continue;

     case 3:
        do{
           delay_us(100);
           countBinaryCode++;

        }while((input(PIN_A4))&&(countBinaryCode<=16));

        if(countBinaryCode>=14)
        {
           return FALSE;
        }
        else
        {
           output_low(PIN_D6);
           delay_ms(5);
           return TRUE;
        }
   }
}

int receiveNearestDistance(void)
{
   int distanceCentimeters[5][12]=
   {
      {15,16,17,18,19,20,21,22,23,24,25,26},
      {27,28,29,30,31,32,33,34,35,36,37,38},
      {39,40,41,42,43,44,45,46,47,48,49,50},
      {51,52,53,54,55,56,57,58,59,60,61,62},
      {63,64,65,66,67,68,69,70}
   };

   switch(codedCommandRead)
   {
      case 0x933:
         return distanceCentimeters[0][0];
         break;
      case 0x935:
         return distanceCentimeters[0][1];
         break;
      case 0x936:
         return distanceCentimeters[0][2];
         break;
      case 0x939:
         return distanceCentimeters[0][3];
         break;
      case 0x93A:
         return distanceCentimeters[0][4];
         break;
      case 0x93C:
```

```
    return distanceCentimeters[0][5];
    break;
case 0x953:
    return distanceCentimeters[0][6];
    break;
case 0x955:
    return distanceCentimeters[0][7];
    break;
case 0x956:
    return distanceCentimeters[0][8];
    break;
case 0x959:
    return distanceCentimeters[0][9];
    break;
case 0x95A:
    return distanceCentimeters[0][10];
    break;
case 0x95C:
    return distanceCentimeters[0][11];
    break;
case 0x963:
    return distanceCentimeters[1][0];
    break;
case 0x965:
    return distanceCentimeters[1][1];
    break;
case 0x966:
    return distanceCentimeters[1][2];
    break;
case 0x969:
    return distanceCentimeters[1][3];
    break;
case 0x96A:
    return distanceCentimeters[1][4];
    break;
case 0x96C:
    return distanceCentimeters[1][5];
    break;
case 0x993:
    return distanceCentimeters[1][6];
    break;
case 0x995:
    return distanceCentimeters[1][7];
    break;
case 0x996:
    return distanceCentimeters[1][8];
    break;
case 0x999:
    return distanceCentimeters[1][9];
    break;
case 0x99A:
    return distanceCentimeters[1][10];
    break;
```

```
case 0x99C:
   return distanceCentimeters[1][11];
   break;
case 0x9A3:
   return distanceCentimeters[2][0];
   break;
case 0x9A5:
   return distanceCentimeters[2][1];
   break;
case 0x9A6:
   return distanceCentimeters[2][2];
   break;
case 0x9A9:
   return distanceCentimeters[2][3];
   break;
case 0x9AA:
   return distanceCentimeters[2][4];
   break;
case 0x9AC:
   return distanceCentimeters[2][5];
   break;
case 0x9C3:
   return distanceCentimeters[2][6];
   break;
case 0x9C5:
   return distanceCentimeters[2][7];
   break;
case 0x9C6:
   return distanceCentimeters[2][8];
   break;
case 0x9C9:
   return distanceCentimeters[2][9];
   break;
case 0x9CA:
   return distanceCentimeters[2][10];
   break;
case 0x9CC:
   return distanceCentimeters[2][11];
   break;
case 0xA33:
   return distanceCentimeters[3][0];
   break;
case 0xA35:
   return distanceCentimeters[3][1];
   break;
case 0xA36:
   return distanceCentimeters[3][2];
   break;
case 0xA39:
   return distanceCentimeters[3][3];
   break;
case 0xA3A:
   return distanceCentimeters[3][4];
```

```
        break;
    case 0xA3C:
        return distanceCentimeters[3][5];
        break;
    case 0xA53:
        return distanceCentimeters[3][6];
        break;
    case 0xA55:
        return distanceCentimeters[3][7];
        break;
    case 0xA56:
        return distanceCentimeters[3][8];
        break;
    case 0xA59:
        return distanceCentimeters[3][9];
        break;
    case 0xA5A:
        return distanceCentimeters[3][10];
        break;
    case 0xA5C:
        return distanceCentimeters[3][11];
        break;
    case 0xA63:
        return distanceCentimeters[4][0];
        break;
    case 0xA65:
        return distanceCentimeters[4][1];
        break;
    case 0xA66:
        return distanceCentimeters[4][2];
        break;
    case 0xA69:
        return distanceCentimeters[4][3];
        break;
    case 0xA6A:
        return distanceCentimeters[4][4];
        break;
    case 0xA6C:
        return distanceCentimeters[4][5];
        break;
    case 0xA93:
        return distanceCentimeters[4][6];
        break;
    case 0xA95:
        return distanceCentimeters[4][7];
        break;
    case 0xA96:
        return distanceCentimeters[4][7];
        break;
    default:
        return FALSE;

}
```

```
}

void buzzerSystemSound(long setSemiPeriodMicroseconds,long maxSemiPeriodValue)
{
   unsigned long countSemiPeriod;

   if(!lastSystemSound)
   {
      for(countSemiPeriod=1;countSemiPeriod<=maxSemiPeriodValue;countSemiPeriod++)
      {
         output_toggle(PIN_C1);
         delay_us(setSemiPeriodMicroseconds);
      }
      output_low(PIN_C1);
   }
   else
   {
      delay_ms(200);
   }
}

void controlBrightnessDisplay(void)
{
   switch(switchBrightnessDisplay)
   {
      case 1:
         adjustBrightness=999;
         break;
      case 2:
         adjustBrightness=750;
         break;
      case 3:
         adjustBrightness=500;
         break;
      case 4:
         adjustBrightness=250;
         break;
      case 5:
         adjustBrightness=0;
         break;
   }
   set_pwm1_duty(adjustBrightness);
}

void batteryRemoteControlStatus(void)
{
   float inputAnalogVoltage=0;

   set_adc_channel(5);
   inputAnalogVoltage=100*((float)read_adc()*VDC)/1023.0;

   if((inputAnalogVoltage>=255)&&(flagControllerBattery100))
   {
```

```
    display_setTextSize(2);
    display_setTextColor(ILI9341_WHITE);
    display_setCursor(178,82);
    display_print("100%\r\n");

    flagControllerBattery100=0;
    flagControllerBattery80=1;
}
else if((255>inputAnalogVoltage)&&(inputAnalogVoltage>=242)&&(flagControllerBattery80))
{
    display_setTextSize(2);
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(178,82);
    display_print("100%\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setCursor(185,82);
    display_print("80%\r\n");

    flagControllerBattery80=0;
    flagControllerBattery60=1;
}
else if((242>inputAnalogVoltage)&&(inputAnalogVoltage>=229)&&(flagControllerBattery60))
{
    display_setTextSize(2);
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(185,82);
    display_print("80%\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setCursor(185,82);
    display_print("60%\r\n");

    flagControllerBattery60=0;
    flagControllerBattery40=1;
}
else if((229>inputAnalogVoltage)&&(inputAnalogVoltage>=216)&&(flagControllerBattery40))
{
    display_setTextSize(2);
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(185,82);
    display_print("60%\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setCursor(185,82);
    display_print("40%\r\n");

    flagControllerBattery40=0;
    flagControllerBattery20=1;
}
else if((216>inputAnalogVoltage)&&(inputAnalogVoltage>=204)&&(flagControllerBattery20))
{
    display_setTextSize(2);
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(185,82);
    display_print("40%\r\n");
```

```
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,82);
      display_print("20%\r\n");

      flagControllerBattery20=0;
      flagControllerBatteryLow=1;
   }
   else if((203>inputAnalogVoltage)&&(flagControllerBatteryLow)&&(wrongValue))
   {
      display_setTextSize(2);
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(185,82);
      display_print("40%\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,82);
      display_print("20%\r\n");

      flagControllerBatteryLow=0;
   }
   wrongValue=1;
}

void batteryRobotStatus(void)
{
   if(((codedCommandRead==0xA99)||(flagRobotBattery==100))&&(keepRobotBattery))
   {
      display_setTextSize(2);
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(178,139);
      display_print("100%\r\n");

      flagRobotBattery=80;
      keepRobotBattery=0;
   }
   else if(((codedCommandRead==0xA9A)||(flagRobotBattery==80))&&(keepRobotBattery))
   {
      display_setTextSize(2);
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(178,139);
      display_print("100%\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,139);
      display_print("80%\r\n");

      flagRobotBattery=60;
      keepRobotBattery=0;
   }
   else if(((codedCommandRead==0xA9C)||(flagRobotBattery==60))&&(keepRobotBattery))
    {
      display_setTextSize(2);
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(185,139);
      display_print("80%\r\n");
```

```
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,139);
      display_print("60%\r\n");

     flagRobotBattery=40;
     keepRobotBattery=0;
   }
   else if(((codedCommandRead==0xAA3)||(flagRobotBattery==40))&&(keepRobotBattery))
   {
      display_setTextSize(2);
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(185,139);
      display_print("60%\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,139);
      display_print("40%\r\n");

     flagRobotBattery=20;
     keepRobotBattery=0;
   }
   else if(((codedCommandRead==0xAA5)||(flagRobotBattery==20))&&(keepRobotBattery))
   {
      display_setTextSize(2);
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(185,139);
      display_print("40%\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,139);
      display_print("20%\r\n");

      flagRobotBattery=0;
      keepRobotBattery=0;
   }
   else if(((codedCommandRead==0xAC5)||(flagRobotBattery==0))&&(keepRobotBattery))
   {
      display_setTextSize(2);
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(185,139);
      display_print("20%\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(185,139);
      display_print("LOW\r\n");
   }
}

void optionMainMenu(void)
{
   if(!flagOptionMainMenu)
   {
      if((switchOptionMainMenu>=1)&&(switchOptionMainMenu<=3))
      {
         ++switchOptionMainMenu
      }
```

```
      else
      {
        switchOptionMainMenu=1;
      }
    }
    else
    {
      flagOptionMainMenu=0;
      switchOptionMainMenu=lastOptionMainMenu;
    }
    display_setTextSize(2);

    switch(switchOptionMainMenu)
    {
      case 1:
        display_setTextColor(ILI9341_WHITE);
        display_setCursor(15,151);
        display_print("About Thesis\r\n");
        display_setTextColor(ILI9341_DARKYELLOW);
        display_setCursor(21,61);
        display_print("Manual Mode\r\n");
        break;

      case 2:
        display_setTextColor(ILI9341_WHITE);
        display_setCursor(21,61);
        display_print("Manual Mode\r\n");
        display_setTextColor(ILI9341_DARKYELLOW);
        display_setCursor(39,91);
        display_print("Settings\r\n");
        break;

      case 3:
        display_setTextColor(ILI9341_WHITE);
        display_setCursor(39,91);
        display_print("Settings\r\n");
        display_setTextColor(ILI9341_DARKYELLOW);
        display_setCursor(21,121);
        display_print("Range Meter\r\n");
        break;

      case 4:
        display_setTextColor(ILI9341_WHITE);
        display_setCursor(21,121);
        display_print("Range Meter\r\n");
        display_setTextColor(ILI9341_DARKYELLOW);
        display_setCursor(15,151);
        display_print("About Thesis\r\n");
        break;
    }
    lastOptionMainMenu=switchOptionMainMenu;
}
```

```
void optionManualMode(void)
{
  if(!flagLastManualMode)
  {
    if((switchOptionManualMode>=0)&&(switchOptionManualMode<=3))
    {
      switchOptionManualMode++;
    }
    else
    {
      switchOptionManualMode=1;
    }
  }
  else
  {
    flagLastManualMode=0;
    switchOptionManualMode=lastOptionManualMode;
  }

  switch(switchOptionManualMode)
  {
    case 1:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,158);
      display_print("Arm Speed:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,68);
      display_print("Tank Operation:\r\n");
      break;

    case 2:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,68);
      display_print("Tank Operation:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,98);
      display_print("Tank Speed:\r\n");
      break;
    case 3:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,98);
      display_print("Tank Speed:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,128);
      display_print("Arm Operation:\r\n");
      break;
    case 4:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,128);
      display_print("Arm Operation:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,158);
      display_print("Arm Speed:\r\n");
```

```
break;
    }
    lastOptionManualMode=switchOptionManualMode;
}

void backManualMode(void)
{
    deleteManualMode();
    deleteTankOnArmOff();
    deleteTankOffArmOn();
    deleteTankSpeed();
    deleteArmSpeed();
    flagManualModeEnter=0;
    flagLastManualMode=1;
    flagTankSpeed=1;
    flagArmSpeed=1;
    flagOptionMainMenu=1;
}

void enterManualMode(void)
{
    if(!flagManualModeEnter)
    {
        switchTelecommunication=1;
        deleteMainMenu();
        printManualMode();
        flagManualModeEnter=1;
        flagButtonEnter=0;

        if(keepLastOptionManualMode==1)
        {
            printArmSpeed();
            printTankSpeed();
            printTankOnArmOff();
        }
        else if(keepLastOptionManualMode==3)
        {
            printArmSpeed();
            printTankSpeed();
            printTankOffArmOn();
        }
        if(keepLastOptionManualMode==2)
        {
            printArmSpeed();
            printTankSpeed();
            if(!flagProtectedManualMode)
            {
                printTankOnArmOff();
            }
            else
            {
                printTankOffArmOn();
            }
```

```
        }
        else if(keepLastOptionManualMode==4)
        {
          printArmSpeed();
          printTankSpeed();
          if(!flagProtectedManualMode)
          {
            printTankOnArmOff();
          }
          else
          {
            printTankOffArmOn();
          }
        }
        optionManualMode();
    }

    if((lastOptionManualMode==1)&&(flagButtonEnter))
    {
      printTankOnArmOff();
      flagProtectedManualMode=0;
      keepLastOptionManualMode=1;
    }
    else if((lastOptionManualMode==3)&&(flagButtonEnter))
    {
      printTankOffArmOn();
      flagProtectedManualMode=1;
      keepLastOptionManualMode=3;
    }
    if((lastOptionManualMode==2)&&(flagButtonEnter))
    {
      printTankSpeed();
      keepLastOptionManualMode=2;
    }
    else if((lastOptionManualMode==4)&&(flagButtonEnter))
    {
      printArmSpeed();
      keepLastOptionManualMode=4;
    }

    switchSubMenu=2;
}

void backSettings()
{
    deleteSettings();
    flagSettingsEnter=0;
    flagLastSettings=1;
    flagCrashSafety=1;
    flagSystemSound=1;
    flagHeadlights=1;
    flagBrightnessDisplay=1;
    flagOptionMainMenu=1;
```

```
}

void optionSettings(void)
{
  if(!flagLastSettings)
  {
    if((switchOptionSettings>=1)&&(switchOptionSettings<=3))
    {
      switchOptionSettings++;
    }
    else
    {
      switchOptionSettings=1;
    }
  }
  else
  {
    flagLastSettings=0;
    switchOptionSettings=lastOptionSettings;
  }

  switch(switchOptionSettings)
  {
    case 1:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,158);
      display_print("Brightness:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,68);
      display_print("Crash Safety:\r\n");
      break;
    case 2:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,68);
      display_print("Crash Safety:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,98);
      display_print("Headlights:\r\n");
      break;
    case 3:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,98);
      display_print("Headlights:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,128);
      display_print("System Sound:\r\n");
      break;
    case 4:
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(15,128);
      display_print("System Sound:\r\n");
      display_setTextColor(ILI9341_DARKYELLOW);
      display_setCursor(15,158);
```

```
            display_print("Brightness:\r\n");
            break;

    }
    lastOptionSettings=switchOptionSettings;
}

void enterSettings(void)
{
    if(!flagSettingsEnter)
    {
        switchTelecommunication=2;
        deleteMainMenu();
        printSettings();
        flagSettingsEnter=1;
        flagButtonEnter=0;

        printCrashSafety();
        printHeadlights();
        printSystemSound();
        printBrightnessDisplay();

        optionSettings();
    }

    if((lastOptionSettings==1)&&(flagButtonEnter))
    {
        printCrashSafety();
    }
    else if((lastOptionSettings==2)&&(flagButtonEnter))
    {
        printHeadlights();
    }
    else if((lastOptionSettings==3)&&(flagButtonEnter))
    {
        printSystemSound();
    }
    else if((lastOptionSettings==4)&&(flagButtonEnter))
    {
        printBrightnessDisplay();
        controlBrightnessDisplay();
    }
    switchSubMenu=3;
}

void optionRangeMeter(void)
{
    if(!flagObstacleGlow)
    {
        (!switchObstacleGlow)?(switchObstacleGlow=1):(switchObstacleGlow=0);
    }
    else
    {
```

```
      flagObstacleGlow=0;
      switchObstacleGlow=lastObstacleGlow;
   }

   switch(switchObstacleGlow)
   {
      case 0:
         display_setTextColor(ILI9341_NAVY);
         display_setCursor(190,128);
         display_print("OFF\r\n");
         display_setTextColor(ILI9341_WHITE);
         display_setCursor(190,128);
         display_print("ON\r\n");
         break;
      case 1:
         display_setTextColor(ILI9341_NAVY);
         display_setCursor(190,128);
         display_print("ON\r\n");
         display_setTextColor(ILI9341_WHITE);
         display_setCursor(190,128);
         display_print("OFF\r\n");
         break;
   }
   lastObstacleGlow=switchObstacleGlow;
}

void backRangeMeter(void)
{
   flagDistance=0;
   flagCrashSafety=1;
   flagRangeMeterEnter=0;
   flagDanger=1;
   flagCaution=1;
   flagSafe=1;
   flagObstacleGlow=1;
   deleteRangeMeter();
   flagOptionMainMenu=1;
   output_low(PIN_D0);
   transmitSignal(0xAC3);
   delay_ms(50);
}

void enterRangeMeter(void)
{
   if(!flagRangeMeterEnter)
   {
      switchTelecommunication=3;
      flagDistance=1;
      flagRangeMeterEnter=1;
      deleteMainMenu();
      printRangeMeter();
   }
   optionRangeMeter();
```

```
    switchSubMenu=10;
}

void printDistance(void)

{
   newDistance=receiveNearestDistance();

   if(newDistance!=lastDistance)
   {
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(129,98);
      printf(display_print,"%d\r\n",lastDistance);
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(129,98);
      printf(display_print,"%d\r\n",newDistance);
   }
   else if(newDistance==lastDistance)
   {
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(129,98);
      printf(display_print,"%d\r\n",newDistance);
   }
   printZoneStatus();
   lastDistance=newDistance;
}

void enterAboutThesis(void)
{
   if(flagEnterAboutThesis)
   {
      switchTelecommunication=10;
      deleteMainMenu();
      printAboutThesis();
      flagEnterAboutThesis=0;
      switchSubMenu=10;
      flagOptionMainMenu=1;
   }
}
```

### 5.2.3 Κώδικας Μικροελεγκτή Ρομπότ

```
#include <PIC18F46K20.h>

#use fast_io(A)
#use fast_io(B)
#use fast_io(C)
#use fast_io(D)
#use fast_io(E)

#byte PORTA=0xF80
```

```
#byte PORTB=0xF81
#byte PORTC=0xF82
#byte PORTD=0xF83
#byte PORTE=0xF84

#define leftMotorIn1 PIN_D2
#define leftMotorIn2 PIN_D3
#define rightMotorIn3 PIN_C3
#define rightMotorIn4 PIN_D0

#define VDC 3.345

//Awaiting a receiving signal to decode its command
void procedureReceivedSignal(void);

//The distance measurement procedure is performed
void measuringDistance(void);

//A signal is received containing a coded command which is stored temporary
short receiveSignal(void);

//The temporary coded command is checked for matching with an array value
//If the command is matched, the Tank performs movement
void receiveControlTank(void);

//The temporary coded command is checked for matching with an array value
//If the command is matched, the Arm performs an action
void receiveControlArm(void);

//The temporary coded command is checked for matching
//If the command is matched, Headlights will be turned ON or OFF
void receiveControlHeadlights(void);

//The temporary coded command is checked for matching
//If the command is matched, Crash Safety will be turned ON or OFF
void receiveControlCrashSafety(void);

//Set a coded command which is going to be transmitted wireless
void transmitSignal(unsigned long codedCommandSend);

//Transmits a coded command containing the distance which was measured before
void trasmitNearestDistance(void);

//Function for the control of 2 DC Motors. Set Speed and Direction
void directCurrentMotor(unsigned long motorDutyCycle,short motorDirection,int motorMovement);

//Function for the control speed of 5 Servo Motors rotating to the tight
void servoRotationRight(int speedRotationRight,int selectServoRight);

//Function for the control speed of 5 Servo Motors rotating to the left
void servoRotationLeft(int speedRotationLeft,int selectServoLeft);

//Control Pulse in microseconds for Servo Motor 1
void servoMotor1(void);
```

```
//Control Pulse in microseconds for Servo Motor 2
void servoMotor2(void);

//Control Pulse in microseconds for Servo Motor 3
void servoMotor3(void);

//Control Pulse in microseconds for Servo Motor 4
void servoMotor4(void);

//Control Pulse in microseconds for Servo Motor 5
void servoMotor5(void);

//Holds the servo shafts in place
void servoMotorStablePulse(void);

//If crash safety is ON and the robot is going to be crashed, DC Motors are stopped immediately.
//Then, the robot starts going backwards up to 40cm-60cm (according to terrain))
void crashSafetyOperation(void);

//Transmits a coded command containing the battery life of the Robot
void batteryRobotStatus(void);

float inputAnalogVoltage=0,distance=0;
short flagReceiveSignal=0,flagCrashSafety=1,flagTimer0=1,flagUltrasonic=0,flagTransmitSignal=0;
short flagBattery100=1,flagBattery80=1,flagBattery60=1,flagBattery40=1,flagBattery20=1;
short keepFlagUltrasonic=0,flagServoMotorStablePulse=0,flagDistanceControlTank=0;
unsigned int numberCentimeter=0,checkProtocolPart=0,countBinaryCode=0,bitCode=0;
unsigned long echoPulse=0,codedCommandRead=0,countTimeDelay=0;
unsigned long servoPulse1=1380,servoPulse2=1000,servoPulse3=2150,servoPulse4=1380;
unsigned long lastServoPulse1=0,lastServoPulse2=0,lastServoPulse3=0,lastServoPulse4=0,;
unsigned long servoPulse5=750, lastServoPulse5=0;
short flagStablePulse=1;

void main(void)
{
   set_tris_a(0x09);
   set_tris_b(0x74);
   set_tris_c(0x00);
   set_tris_d(0x00);
   set_tris_e(0x00);

   output_high(PIN_C6);

   ext_int_edge(2);
   enable_interrupts(INT_EXT2);
   disable_interrupts(INT_TIMER0);
   enable_interrupts(GLOBAL);

   setup_adc(ADC_CLOCK_INTERNAL);
   setup_adc_ports(sAN0|VSS_VDD);
   set_adc_channel(0);

   setup_timer_0(T0_INTERNAL|T0_DIV_16);
   setup_timer_2(T2_DIV_BY_4,249,1);
```

```
while(TRUE)
{
   if(flagReceiveSignal)
   {
      disable_interrupts(INT_EXT2);
      servoMotorStablePulse();

      if((codedCommandRead>=0x35A)&&(codedCommandRead<=0x55A))
      {
         receiveControlTank();
         crashSafetyOperation();
         flagUltrasonic=1;
      }
      else if((codedCommandRead>=0x55C)&&(codedCommandRead<=0x6CC))
      {
         receiveControlArm();
      }
      else if((codedCommandRead==0x339)||(codedCommandRead==0x33A))
      {
         receiveControlCrashSafety();
      }
      else if((codedCommandRead==0x333)||(codedCommandRead==0x335))
      {
         receiveControlHeadlights();
      }
      else if(codedCommandRead==0xAAA)
      {
         for(countTimeDelay=0;countTimeDelay<12;countTimeDelay++)
         {
            delay_ms(100);
            servoMotorStablePulse();
         }
         batteryRobotStatus();
      }
      else if(codedCommandRead==0xAAC)
      {
         for(countTimeDelay=0;countTimeDelay<12;countTimeDelay++)
         {
            delay_ms(100);
            servoMotorStablePulse();
         }

         flagUltrasonic=1;
         flagTransmitSignal=1;
      }
      flagReceiveSignal=0;


   if(!flagUltrasonic)
      {
         codedCommandRead=0;
         enable_interrupts(INT_EXT2);
      }
   }
```

```
      else if(flagUltrasonic)
      {
         if(flagTimer0)
         {
            enable_interrupts(INT_TIMER0);
         }
         else
         {
            disable_interrupts(INT_TIMER0);
            flagTimer0=1;
            flagUltrasonic=0;

            if(flagTransmitSignal)
            {
               trasmitNearestDistance();
               flagTransmitSignal=0;
            }
            flagReceiveSignal=0;
            enable_interrupts(INT_EXT2);
            delay_ms(30);
            codedCommandRead=0;
            setup_ccp1(CCP_OFF);
            setup_ccp2(CCP_OFF);
          }
       }

      if(codedCommandRead==0)
      {
         servoMotorStablePulse();

      }
   }
}

#INT_EXT2
void procedureReceivedSignal(void)
{
   flagReceiveSignal=receiveSignal();
}

#INT_TIMER0

void measuringDistance(void)
{
   output_high(PIN_B7);
   delay_us(10);
   output_low(PIN_B7);


   echoPulse=0;
   set_timer0(0);

   while(!input(PIN_B6))
   {
```

```
      set_timer0(0);
   }
   while((input(PIN_B6))&&(get_timer0()<=5882))
   {
      echoPulse=get_timer0();
   }
   distance=(echoPulse/58.82);
   delay_ms(20);

   if(flagDistanceControlTank)
   {
      (15>=distance)?(numberCentimeter=15):(numberCentimeter=0);
      flagDistanceControlTank=0;
   }
   flagTimer0=0;
}

short receiveSignal(void)
{
   bitCode=0;
   checkProtocolPart=0;
   codedCommandRead=0;
   countBinaryCode=0;

   switch(checkProtocolPart)
   {
      case 0:
         while((input(PIN_B5))&&(countBinaryCode<=16))
         {
            delay_us(200);
            countBinaryCode++;
         }

         if((countBinaryCode>=13)&&(countBinaryCode<=9))
         {
            return FALSE;
         }
         else
         {
            countBinaryCode=0;
         }

         while((!input(PIN_B5))&&(countBinaryCode<=16))
         {
            delay_us(200);
            countBinaryCode++;
         }


         if((countBinaryCode>=13)&&(countBinaryCode<=9))
         {
            return FALSE;
         }
         else
```

```
    {
      countBinaryCode=0;
      checkProtocolPart=1;
    }
    continue;

  case 1:
    for(bitCode=0;bitCode<=11;bitCode++)
    {

      while((input(PIN_B5))&&(countBinaryCode<=10))
      {
        countBinaryCode++;

        delay_us(100);
      }

      if((countBinaryCode>=9))
      {
        return FALSE;
      }
      else
      {
        countBinaryCode=0;
      }

      while((!input(PIN_B5))&&(countBinaryCode<=21))
      {
        countBinaryCode++;
        delay_us(100);
      }

      if((countBinaryCode>=20))
      {
        return FALSE;
      }

      if(8>=countBinaryCode)
      {
        countBinaryCode=0;
        bit_set(codedCommandRead,(11-bitCode));

      }
      else
      {
        countBinaryCode=0;
        bit_clear(codedCommandRead,(11-bitCode));
      }
    }

    checkProtocolPart=2;
    continue;

  case 2:
```

```
        while((input(PIN_B5))&&(countBinaryCode<=16))
        {
          countBinaryCode++;
          delay_us(100);
        }

        if(countBinaryCode>=15)
        {
          return FALSE;
        }
        else
        {
          return TRUE;
        }
    }
}

void receiveControlTank(void)
{
    int movementTank=0,motorDirection=0;
    unsigned long speedTank[6][6]=
    {
      {925,920,900,880,920,925},
      {926,923,908,908,930,940},
      {929,922,910,918,922,928},
      {930,926,921,930,933,945},
      {970,935,920,920,935,970},
      {970,932,921,935,942,970}
    };
    unsigned long codedCommandTank[6][6]=
    {
      {0x35A,0x35C,0x363,0x365,0x366,0x369},
      {0x36A,0x36C,0x393,0x395,0x396,0x399},
      {0x39A,0x39C,0x3A3,0x3A5,0x3A6,0x3A9},
      {0x3AA,0x3AC,0x3C3,0x3C5,0x3C6,0x3C9},
      {0x3CA,0x3CC,0x533,0x535,0x536,0x539},
      {0x53A,0x53C,0x553,0x555,0x556,0x559}
    };

    flagDistanceControlTank=1;


    if(codedCommandRead!=0x55A)
    {
      if(codedCommandRead<=codedCommandTank[1][5])
      {
        movementTank=0;
      }
      else if(codedCommandRead<=codedCommandTank[3][5])
      {
        movementTank=2;
      }
      else
      {
```

```
            movementTank=4;
        }

        while(movementTank<6)
        {
            if(codedCommandRead==codedCommandTank[movementTank][0])
            {
                directCurrentMotor(speedTank[movementTank][0],1,motorDirection);
                break;
            }
            else if(codedCommandRead==codedCommandTank[movementTank][1])
            {
                directCurrentMotor(speedTank[movementTank][1],1,motorDirection);
                break;
            }
            else if(codedCommandRead==codedCommandTank[movementTank][2])
            {

                directCurrentMotor(speedTank[movementTank][2],1,motorDirection);
                break;
            }
            else if(codedCommandRead==codedCommandTank[movementTank][3])
            {
                directCurrentMotor(speedTank[movementTank][3],0,motorDirection);
                break;
            }
            else if(codedCommandRead==codedCommandTank[movementTank][4])
            {
                directCurrentMotor(speedTank[movementTank][4],0,motorDirection);
                break;
            }
            else if(codedCommandRead==codedCommandTank[movementTank][5])
            {
                directCurrentMotor(speedTank[movementTank][5],0,motorDirection);
                break;
            }
            motorDirection++;
            movementTank++;
        }
    }
    else

    {
        setup_ccp1(CCP_OFF);
        setup_ccp2(CCP_OFF);
    }
}

void receiveControlArm(void)
{
    int servoMotor=0,servoSpeed=0,finalServoSpeed=0;

    unsigned long codedCommandArm[10][6]=
    {
```

```
    {0x55C,0x563,0x565,0x566,0x569,0x56A},
    {0x56C,0x593,0x595,0x596,0x599,0x59A},
    {0x59C,0x5A3,0x5A5,0x5A6,0x5A9,0x5AA},
    {0x5AC,0x5C3,0x5C5,0x5C6,0x5C9,0x5CA},
    {0x5CC,0x633,0x635,0x636,0x639,0x63A},
    {0x63C,0x653,0x655,0x656,0x659,0x65A},
    {0x65C,0x663,0x665,0x666,0x669,0x66A},
    {0x66C,0x693,0x695,0x696,0x699,0x69A},
    {0x69C,0x6A3,0x6A5,0x6A6,0x6A9,0x6AA},
    {0x6AC,0x6C3,0x6C5,0x6C6,0x6C9,0x6CA}
};

(codedCommandRead<=codedCommandArm[4][5])?(servoSpeed=0):(servoSpeed=5);

if(codedCommandRead!=0x6CC)
{
   while(servoMotor<5)
   {
      if(codedCommandRead==codedCommandArm[servoMotor+servoSpeed][0])

      {
         (servoSpeed==0)?(finalServoSpeed=6):(finalServoSpeed=12);
         servoRotationRight(finalServoSpeed,servoMotor);
         break;
      }
      else if(codedCommandRead==codedCommandArm[servoMotor+servoSpeed][1])
      {
         (servoSpeed==0)?(finalServoSpeed=4):(finalServoSpeed=8);
         servoRotationRight(finalServoSpeed,servoMotor);
         break;
      }
      else if(codedCommandRead==codedCommandArm[servoMotor+servoSpeed][2])
      {
         (servoSpeed==0)?(finalServoSpeed=2):(finalServoSpeed=6);
         servoRotationRight(finalServoSpeed,servoMotor);
         break;
      }
      else if(codedCommandRead==codedCommandArm[servoMotor+servoSpeed][3])
      {
         (servoSpeed==0)?(finalServoSpeed=2):(finalServoSpeed=6);
         servoRotationLeft(finalServoSpeed,servoMotor);

         break;
      }
      else if(codedCommandRead==codedCommandArm[servoMotor+servoSpeed][4])
      {
         (servoSpeed==0)?(finalServoSpeed=4):(finalServoSpeed=8);
         servoRotationLeft(finalServoSpeed,servoMotor);
         break;
      }
      else if(codedCommandRead==codedCommandArm[servoMotor+servoSpeed][5])
      {

         (servoSpeed==0)?(finalServoSpeed=6):(finalServoSpeed=12);
```

```
            servoRotationLeft(finalServoSpeed,servoMotor);
            break;
          }
          ++servoMotor;
        }
    }
}

void receiveControlHeadlights(void)
{
   switch(codedCommandRead)
   {
      case 0x333:
         output_low(PIN_A1);
         break;
      case 0x335:
         output_high(PIN_A1);
         break;
   }
}

void receiveControlCrashSafety(void)
{
   switch(codedCommandRead)
   {
      case 0x339:
         flagCrashSafety=1;
         break;
      case 0x33A:
         flagCrashSafety=0;
         break;
   }
}

void transmitSignal(unsigned long codedCommandSend)
{
   output_high(PIN_B1);
   delay_ms(2);
   output_low(PIN_B1);
   delay_ms(2);


   for(bitCode=0;bitCode<=11;bitCode++)
   {
      if(bit_test(codedCommandSend,(11-bitCode)))
      {
         output_high(PIN_B1);
         delay_us(500);
         output_low(PIN_B1);
         delay_us(500);
      }
      else
      {
         output_high(PIN_B1);
```

```
            delay_us(500);
            output_low(PIN_B1);
            delay_ms(1);
        }
    }
    output_high(PIN_B1);
    delay_ms(1);
    output_low(PIN_B1);
}

void trasmitNearestDistance(void)
{
    unsigned long codedCommandDistance[5][12]=
    {
        {0x933,0x935,0x936,0x939,0x93A,0x93C,0x953,0x955,0x956,0x959,0x95A,0x95C},
        {0x963,0x965,0x966,0x969,0x96A,0x96C,0x993,0x995,0x996,0x999,0x99A,0x99C},
        {0x9A3,0x9A5,0x9A6,0x9A9,0x9AA,0x9AC,0x9C3,0x9C5,0x9C6,0x9C9,0x9CA,0x9CC},
        {0xA33,0xA35,0xA36,0xA39,0xA3A,0xA3C,0xA53,0xA55,0xA56,0xA59,0xA5A,0xA5C},
        {0xA63,0xA65,0xA66,0xA69,0xA6A,0xA6C,0xA93,0xA95,0xA96}
    };

    if(26>=distance)
    {
        for(numberCentimeter=15;numberCentimeter<=26;numberCentimeter++)
        {
            if(numberCentimeter>=distance)
            {
                transmitSignal(codedCommandDistance[0][numberCentimeter-15]);
                break;
            }
        }
    }
    else if(38>=distance)
    {
        for(numberCentimeter=27;numberCentimeter<=38;numberCentimeter++)
        {
            if(numberCentimeter>=distance)
            {
                transmitSignal(codedCommandDistance[1][numberCentimeter-27]);
                break;

            }
        }
    }
    else if(50>=distance)
    {
        for(numberCentimeter=39;numberCentimeter<=50;numberCentimeter++)
        {
            if(numberCentimeter>=distance)
            {
                transmitSignal(codedCommandDistance[2][numberCentimeter-39]);
                break;
            }
        }
```

```
    }
    else if(62>=distance)
    {
      for(numberCentimeter=51;numberCentimeter<=62;numberCentimeter++)
      {
        if(numberCentimeter>=distance)
        {
          transmitSignal(codedCommandDistance[3][numberCentimeter-51]);
          break;
        }
      }
    }
    else if(70>=distance)
    {
      for(numberCentimeter=63;numberCentimeter<=70;numberCentimeter++)
      {
        if(numberCentimeter>=distance)
        {
          transmitSignal(codedCommandDistance[4][numberCentimeter-63]);
          break;
        }
      }
    }
    else
    {
      transmitSignal(codedCommandDistance[4][8]);
    }
}

void directCurrentMotor(unsigned long motorDutyCycle,short motorDirection,int motorMovement)
{
    if(!motorMovement)
    {
      switch(motorDirection)
      {
        case 0:
          output_high(leftMotorIn1);
          output_low(leftMotorIn2);
          output_low(rightMotorIn3);

          output_high(rightMotorIn4);
          break;

        case 1:
          output_low(leftMotorIn1);
          output_high(leftMotorIn2);
          output_high(rightMotorIn3);
          output_low(rightMotorIn4);
          break;
      }
    }
    else if(motorMovement)
    {
```

```
      switch(motorDirection)
      {
         case 0:
            output_high(leftMotorIn1);
            output_low(leftMotorIn2);
            output_high(rightMotorIn3);
            output_low(rightMotorIn4);
            break;

         case 1:
            output_low(leftMotorIn1);
            output_high(leftMotorIn2);
            output_low(rightMotorIn3);
            output_high(rightMotorIn4);
            break;
      }
   }
   setup_ccp1(CCP_PWM);
   setup_ccp2(CCP_PWM);

   set_pwm1_duty(motorDutyCycle);
   set_pwm2_duty(motorDutyCycle);
}

void servoRotationRight(int speedRotationRight,int servoRight)
{
   int countRight=0;

   switch(servoRight)
   {
      case 0:
         for(lastServoPulse2=servoPulse2;lastServoPulse2<2300;lastServoPulse2+=2)
         {
            countRight++;

            if(countRight==speedRotationRight)
            {
               servoPulse2=lastServoPulse2;
               servoMotor2();

               break;
            }
         }
         break;

      case 1:
         for(lastServoPulse1=servoPulse1;lastServoPulse1<2400;lastServoPulse1+=2)
         {
            countRight++;

            if(countRight==speedRotationRight)
            {
               servoPulse1=lastServoPulse1;
               servoMotor1();
```

```
                break;
            }
        }

    case 2:
        for(lastServoPulse3=servoPulse3;lastServoPulse3<2000;lastServoPulse3+=2)
        {
            countRight++;

            if(countRight==speedRotationRight)
            {
                servoPulse3=lastServoPulse3;
                servoMotor3();
                break;
            }
        }
        break;

    case 3:
        for(lastServoPulse4=servoPulse4;lastServoPulse4<2400;lastServoPulse4+=2)
        {
            countRight++;

            if(countRight==speedRotationRight)
            {
                servoPulse4=lastServoPulse4;
                servoMotor4();
                break;
            }
        }
        break;

    case 4:
        for(lastServoPulse5=servoPulse5;lastServoPulse5<1200;lastServoPulse5+=2)
        {
            countRight++;

            if(countRight==speedRotationRight)

            {
                servoPulse5=lastServoPulse5;
                servoMotor5();
                break;
            }
        }
        break;
  }
}

void servoRotationLeft(int speedRotationLeft,int servoLeft)
{
  int countLeft=0;
```

```
switch(servoLeft)

{
  case 0:
    for(lastServoPulse2=servoPulse2;lastServoPulse2>1000;lastServoPulse2-=2)
    {
      countLeft++;

      if(countLeft==speedRotationLeft)
      {
        servoPulse2=lastServoPulse2;
        servoMotor2();
        break;
      }
    }
    break;

  case 1:
    for(lastServoPulse1=servoPulse1;lastServoPulse1>400;lastServoPulse1-=2)
    {
      countLeft++;

      if(countLeft==speedRotationLeft)
      {
        servoPulse1=lastServoPulse1;
        servoMotor1();
        break;
      }
    }
    break;

  case 2:
    for(lastServoPulse3=servoPulse3;lastServoPulse3>1000;lastServoPulse3-=2)
    {
      countLeft++;

      if(countLeft==speedRotationLeft)
      {
        servoPulse3=lastServoPulse3;

        servoMotor3();
        break;
      }
    }
    break;

  case 3:
    for(lastServoPulse4=servoPulse4;lastServoPulse4>400;lastServoPulse4-=2)
    {
      countLeft++;

      if(countLeft==speedRotationLeft)
      {
        servoPulse4=lastServoPulse4;
```

```
                servoMotor4();
                break;

            }
        }
        break;

    case 4:
        for(lastServoPulse5=servoPulse5;lastServoPulse5>750;lastServoPulse5-=2)
        {
            countLeft++;

            if(countLeft==speedRotationLeft)
            {
                servoPulse5=lastServoPulse5;
                servoMotor5();
                break;
            }
        }
        break;
  }
}

void servoMotor1(void)
{
  output_high(PIN_E2);
  delay_us(servoPulse1);

  output_low(PIN_E2);
}

void servoMotor2(void)
{
  output_high(PIN_E1);
  delay_us(servoPulse2);
  output_low(PIN_E1);
}

void servoMotor3(void)

{
  output_high(PIN_A5);
  delay_us(servoPulse3);
  output_low(PIN_A5);
}

void servoMotor4(void)
{
  output_high(PIN_E0);
  delay_us(servoPulse4);
  output_low(PIN_E0);
}
```

```
void servoMotor5(void)
{
   output_high(PIN_A4);
   delay_us(servoPulse5);

   output_low(PIN_A4);
}

void servoMotorStablePulse(void)
{
   servoMotor1();
   servoMotor2();
   servoMotor3();
   servoMotor4();
   servoMotor5();
}

void crashSafetyOperation(void)
{
   if((flagCrashSafety)&&(numberCentimeter==15))
   {
      directCurrentMotor(930,1,0);
      for(countTimeDelay=0;countTimeDelay<5;countTimeDelay++)
      {
         delay_ms(100);
         servoMotorStablePulse();
      }
      setup_ccp1(CCP_OFF);
      setup_ccp2(CCP_OFF);
   }
}
void batteryRobotStatus(void)
{
   inputAnalogVoltage=100*((float)read_adc()*VDC)/1023.0;

   if(inputAnalogVoltage>=280)
   {
      flagBattery100=0;
      flagBattery80=1;
      transmitSignal(0xA99);

   }
   else if((280>inputAnalogVoltage)&&(inputAnalogVoltage>=266))
   {
      flagBattery80=0;
      flagBattery60=1;
      transmitSignal(0xA9A);
   }
   else if((266>inputAnalogVoltage)&&(inputAnalogVoltage>=252))
   {
      flagBattery60=0;
      flagBattery40=1;
      transmitSignal(0xA9C);
   }
```

```
  else if((252>inputAnalogVoltage)&&(inputAnalogVoltage>=238))
  {
    flagBattery40=0;
    flagBattery20=1;
    transmitSignal(0xAA3);

  }
  else if((238>inputAnalogVoltage)&&(inputAnalogVoltage>=224))
  {
    flagBattery20=0;
    transmitSignal(0xAA5);
  }
  else
  {
    transmitSignal(0xAC5);
  }
}
```

## 5.3 Προσομοίωση με Πειραματισμούς και Τελική Εφαρμογή

### 5.3.1 Προσομοίωση Crobarm

Η αρχική αναπαράσταση λειτουργίας για το πραγματικό ρομποτικό σύστημα, διεξάχθηκε σε ένα περιβάλλον Προσομοίωσης Αυτοματοποίησης Ηλεκτρονικού Σχεδίου (EDA Simulatior). Το επαγγελματικό λογισμικό Proteus Design Suite έκδοσης 8.9, της εταιρίας Labcenter Electronics Ltd, βοήθησε στην διαδικασία μεταχείρισης ολόκληρου του μοντέλου. Η δημιουργία μίμησης ήταν ένα σημαντικό μέρος της πτυχιακής, ώστε να γίνεται μια συνεχής αξιολόγηση για τον τρόπο που δουλεύει. Επομένως, αυτό είχε σαν αποτέλεσμα την βελτιστοποίηση, αλλαγή και πρόσθεση ηλεκτρονικών υλικών ή συνδέσεις καλωδίων. Το πόρισμα που προκύπτει κατά την προσομοίωση των σχημάτων 5.19, 5.20, δείχνει μεγάλη ακρίβεια με υψηλά επίπεδα ρεαλιστικότητας και λειτουργικότητας. Λόγο του σχεδιαστικού φόρτου και των εκτελέσιμων εντολών από τους μικροελεγκτές, το πρόγραμμα δεν τρέχει σε πραγματικό χρόνο (Real Time) αλλά με καθυστέρηση, χωρίς να επηρεάζεται η δραστηριότητά του.

Στο σχήμα 5.21, χρησιμοποιείται παλμογράφος (Oscilloscope) για την εμφάνιση αποστολής/λήψης πακέτων μεταξύ των δύο συσκευών. Το κίτρινο χρώμα είναι ένα κωδικοποιημένο σήμα περιέχοντας μια εντολή, η οποία εκπέμπεται από το μικροελεγκτή MCU1 προς τον MCU2. Όταν διαβάσει την πληροφορία ο MCU2, εκτελείται μια συγκεκριμένη ενέργεια στο ηλεκτρονικό σύστημα Crobarm. Ύστερα, με μπλε χρώμα στέλνεται ένα αντίστοιχο σήμα, από τον μικροελεγκτή MCU2 προς τον MCU1. Αφού ο MCU1 αναγνωρίσει την εντολή, θα προβεί σε μια άμεση αντίδραση του τηλεχειριστήριου. Η τηλεπικοινωνία χωρίζεται σε τρία στάδια: συνεχόμενη, αμφίδρομη, επιλεκτική. Η αμφίδρομη (Amphidromous Telecommunication), είναι ένας κύκλος επαναλαμβανόμενης επικοινωνίας χωρίς κάποια διακοπή. Εάν υπάρξει διακοπή, τότε μόλις εφαρμοστή η ανάκαμψη ξεκινάει από την αρχή πάλι η διεξαγωγή αποστολής/λήψης πακέτων για το ποσοστό μπαταρίας ή μέτρηση απόστασης. Η συνεχόμενη (Continuous Telecommunication), διεξάγει συνεχόμενη αποστολή εντολών για τον έλεγχο του ρομπότ. Προβλέπεται από την προσομοίωση, η απόκριση στο τανκ να είναι 110ms και στον

βραχίονα 50ms. Τέλος, η επιλεκτική (Selective Telecommunication) αποστέλλει μονάχα μια εντολή προς το ρομπότ, η οποία προκαλεί αλλαγή κατάστασης στα φώτα LED ή την προστασία σύγκρουσης.

Το σχήμα 5.22, είναι μια αναπαράσταση ελέγχου της θέση του άξονα, για τους πέντε κινητήρες σέρβο, όπου αποτελούν το ρομποτικό βραχίονα. Τα PWM σήματα δεν εκτελούνται ταυτόχρονα, αλλά το ένα μετά το άλλο σε χρονοθυρίδες. Επίσης, διαφέρουν μεταξύ τους καθότι σύμφωνα με την γωνία που έχει διαγράψει το καθένα, ο παλμός μπορεί να είναι μεγαλύτερης διάρκειας ή μικρότερης. Αυτό σημαίνει ότι μεγαλώνοντας το κύκλο εργασίας, θα αυξηθούν οι μοίρες, ενώ όταν μικραίνει θα μειώνονται οι μοίρες. Ο χρόνος ενός τέτοιου τετραγωνικού παλμού, κυμαίνεται από τα 400μs μέχρι τα 2400μs, καλύπτοντας ένα εύρη φάσμα από 0 έως 180 μοίρες. Ανά 50ms,100ms,110ms και συνεχόμενα μπορεί να ανανεωθεί το σήμα παλμού στους κινητήρες, διατηρώντας σταθερή γωνία σε ένα καθοριστικό σημείο χωρίς να μεταβληθεί. Αλλάζει μονάχα αν δεχτεί παλμική εντολή, η οποία θα διαφέρει από την προηγούμενη. Όσον αφορά το κενό διάστημα, διαφέρουν οι χρόνοι ανάλογα με τις κρίσιμες διεργασίες, που εκτελεί ο μικροελεγκτής. Ο καταλύτης, είναι πως αναλόγως τις κινήσεις των ρυθμιζόμενων αντιστάσεων (Variable Resistor), πραγματοποιείται μετάδοση εντολών προς το ρομπότ έτσι ώστε να περιστρέφονται οι σέρβο. Το ίδιο συμβαίνει στην περίπτωση των DC κινητήρων, όμως για την κατεύθυνση και την ταχύτητά τους.

Παρουσιάζεται από το σχήμα 5.23, ο έλεγχος στροφών για δύο κινητήρες συνεχούς ρεύματος, στα αριστερά και δεξιά οι οποίοι κινούνε το ρομπότ. Δύο συνεχόμενα PWM σήματα συχνότητας 2KHz, έχοντας πάντα τον ίδιο κύκλο εργασίας, καθορίζουν την ταχύτητα και το σταμάτημα του οχήματος. Τα σήματα διακόπτονται μόνο εάν η θέση των μοχλών, βρίσκεται τοποθετημένη σε ουδέτερο σημείο. Με άλλα λόγια, η περιστρεφόμενη επαφή των ποτενσιόμετρων πρέπει να είναι στη μέση (50%), ακινητοποιώντας με αυτόν το τρόπο το ερπυστριοφόρο. Μεγαλύτερος κύκλος εργασίας παλμών, σημαίνει αυξημένη ταχύτητα στροφών για τους κινητήρες, διαφορετικά θα είναι μικρότερη. Τρία προαναφερόμενα επίπεδα επιτάχυνσης, είναι υπεύθυνα για αυτή την ειδική ρύθμιση. Είτε για την κίνηση προς τα μπροστά, πίσω, δεξιά ή αριστερά. Η ταχύτητα τανκ είναι στο αργό (Slow) στην παρούσα φάση. Οι ταχύτητες κανονική (Normal) και γρήγορη (Fast) έχουν την ίδια ακριβώς λογική, απλά με μεγαλύτερο παλμό, σύμφωνα με το αντίστοιχο επίπεδο.

### 5.3.2 Πειραματικό Μέρος Crobarm

Μετά από κάθε εκτέλεση προσομοίωσης, εκπληρωνόταν η διενέργεια ενός μικρού πειράματος. Σκοπός ήταν να εξεταστεί αν ενεργή σωστά ή λάθος, το εκάστοτε μοντέλο σύμφωνα με την αναπαράσταση που σχεδιάζονταν. Η πειραματική μέθοδος ήταν αναγκαία, για την έμπρακτη δοκιμή της θεωρητικής εργασίας και μελέτης που επισπεύσθηκε. Κατά την διάρκεια εκτέλεσης του φαινομένου γινόταν παράλληλα έλεγχος. Μετά από πολυάριθμα πειράματα, το σύστημα κατέληξε να ανταποκρίνεται με τον ίδιο ακριβώς τρόπο και στην πραγματικότητα. Τα ηλεκτρονικά κυκλώματα, στηνόντουσαν επάνω στη Δοκιμαστική Πλακέτα Bread Board μοντέλου ZY-203, όπως φαίνεται από το σχήμα 5.24. Περιλαμβάνονται μπόρνες τροφοδοσίας γείωσης και 1260 τρύπες.

Στο σχήμα 5.25, εμφανίζονται τα RF modules με δύο κεραίες, για την αποστολή και λήψη κωδικοποιημένων εντολών. Στα αριστερά είναι ο πομπός, ενώ στα δεξιά ο δέκτης. Εξαιτίας του θορύβου στην δοκιμαστική πλακέτα, ο αποδέκτης καταλήγει να διαβάζει αλλοιωμένη πληροφορία. Χρειάστηκε να τοποθετηθούν εξωτερικά, συνδέοντάς τους ακροδέκτες με καλώδια από κλιπ κροκόδειλου. Φάνηκε

να δουλεύει χωρίς παρενέργειες.

Φωτογραφίες υποδεικνύουν το πειραματικό μέρος της πτυχιακής εργασίας, καθότι εξετελέσθη με επιτυχία ανταποκρινόμενο στην πραγματικότητα. Η πηγή συνεχούς τάσης των 5V, καλύφθηκε από ένα DIY τροφοδοτικό σταθερού υπολογιστή. Ο μικρός παλμογράφος DSI FNIRSI-150 15001K, διαθέτει ένα κανάλι (1 Channel) και χρησιμοποιήθηκε για την καταγραφή των σημάτων, επαληθεύοντας έτσι ότι δουλεύει το σύστημα. Επίσης, το πολύμετρο Mastech MS8217 εμφανίζει στην οθόνη του, την μετρούμενη συνεχή τάση από το ποτενσιόμετρο, που ορίζεται ως είσοδο για τους αναλογικούς ακροδέκτες A3 και A1.

Προκύπτει από το σχήμα 5.27, η επιβεβαίωση της τηλεπικοινωνίας μεταξύ των μικροελεγκτών PIC, με κατεύθυνση μετάδοσης εντολών από τον MCU1 προς τον MCU2. Από την στιγμή που στέλνει συνέχεια κωδικοποιημένα σήματα ο MCU1, αυτό σημαίνει πως και ο MCU2 κάνει το ίδιο κατά την αμφίδρομη. Από το σχήμα 5.28 διακρίνεται στην οθόνη του παλμογράφου, ο ίδιος χρόνος απόκρισης με αυτόν της προσομοίωσης, στα 110ms και 50ms.

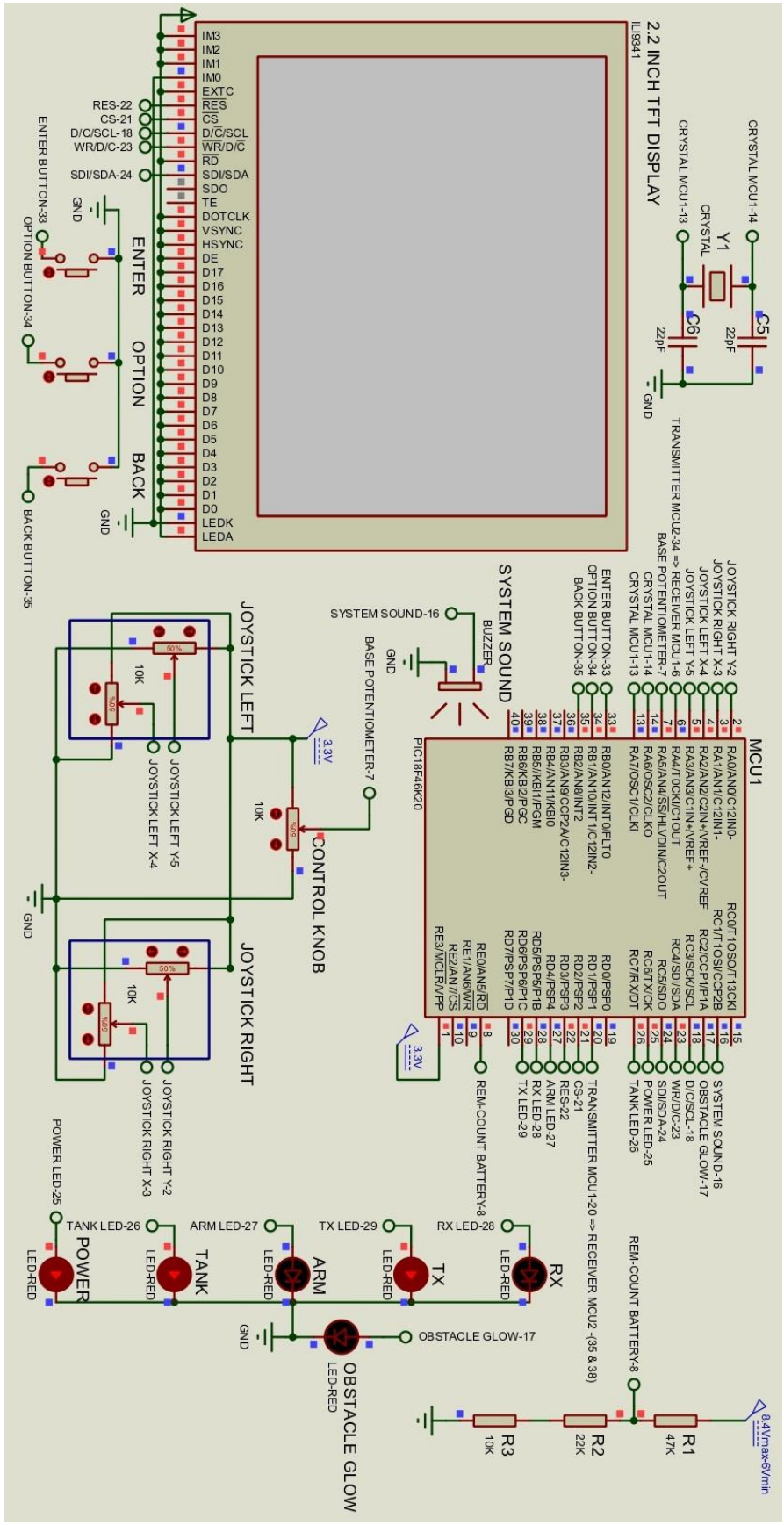
Επαληθεύεται στο σχήμα 5.29, η μέθοδος ελέγχου για την επιτάχυνση των DC κινητήρων, σε Αργή Ταχύτητα Τανκ. Είναι εμφανές από τα διαγράμματα, πως τα σήματα PWM είναι ίδιας μορφής με αυτά τις προσομοίωσης, έχοντας μια μικρή ανοχή. Συμπεριλαμβάνονται και τα υπόλοιπα επίπεδα επιτάχυνσης στα σχήματα 5.30, 5.31. Το τελευταίο μέρος του πειράματος, είναι η διατύπωση του μηχανισμού με τον οποίον ελέγχονται οι κινητήρες Σέρβο, στο σχήμα 5.32. Η δοκιμή σέρβο πραγματοποιήθηκε, στέλνοντας ρυθμιζόμενα παλμικά σήματα, όπως ακριβώς ήταν η προσομοίωση, καταλήγοντας ξανά να ανταποκριθούν με την ίδια ακριβώς λογική.

### 5.3.3 Τελική Εφαρμογή Crobarm

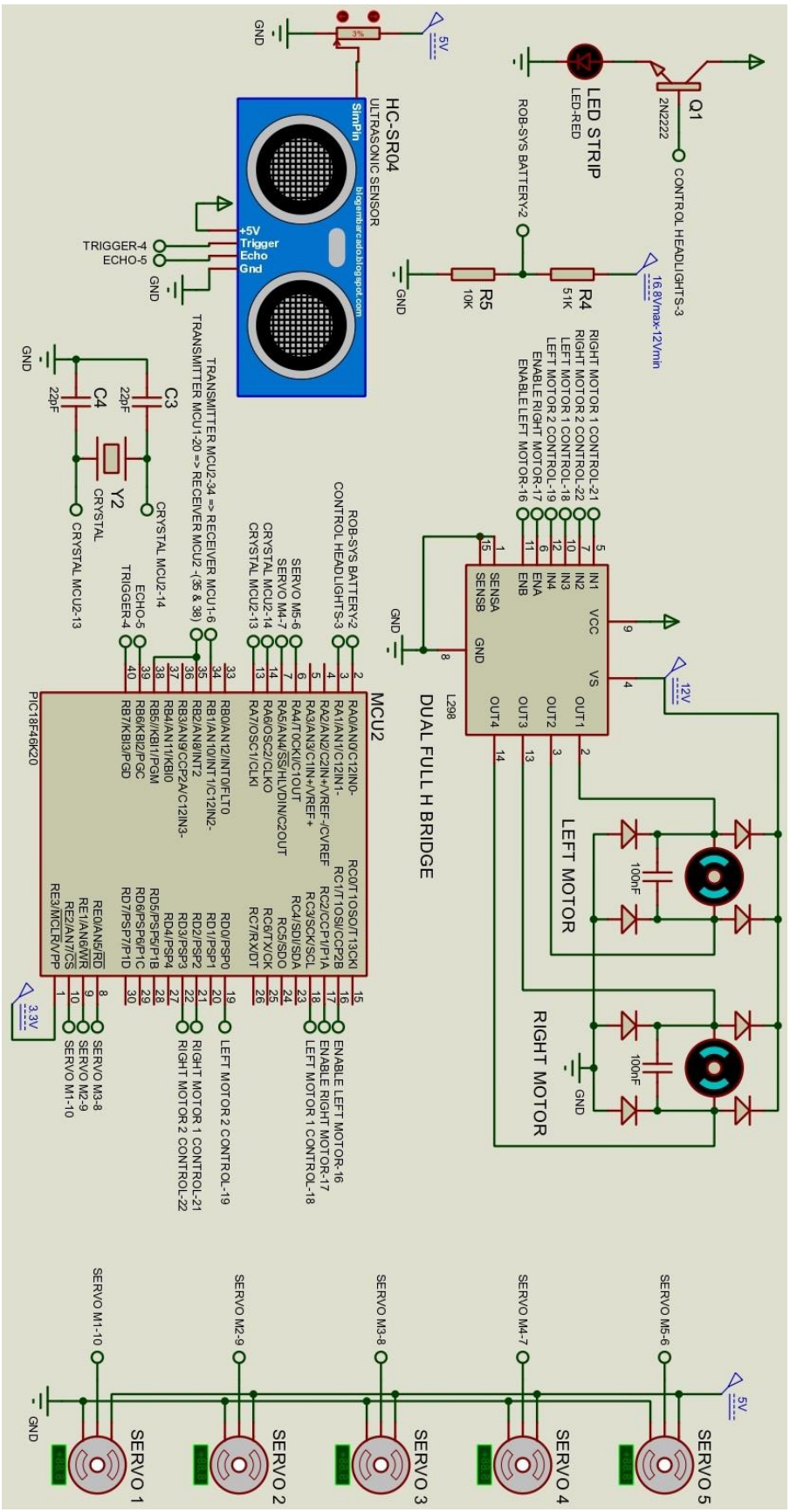
Απομένει το πρακτικό μέρος, για την υλοποίηση μιας πλήρους μοντελοποίησης με κατασκευή. Αρχικά, συναρμολογήθηκαν οι σκελετοί του βραχίονα και του τανκ τοποθετώντας τους κινητήρες. Έπειτα, σχεδιάστηκε από ανεξίτηλο μαρκαδόρο επάνω σε πλεξιγκλάς, τα κομμάτια που έπρεπε να κοπούν, για την βάση του ρομπότ και το εξωτερικό σώμα του τηλεχειριστήριου. Όσα σημεία ήταν σημαδεμένα με μελάνι, τρυπήθηκαν από δράπανο. Ακολούθως, τοποθετήθηκαν βίδες με παξιμάδια για το στήσιμο των τυπωμένων κυκλωμάτων πλακέτας. Τα διάφορα κομμάτια από το πλεξιγκλάς, μαζί με το πλαστικό κάλυμμα των κεραιών, κολλήθηκαν κατάλληλα με κόλα LOGO.

Διεκπεραιώνοντας το κατασκευαστικό μέρος των δύο συσκευών και με τις μπαταρίες πλήρως φορτισμένες, έγινε η πρώτη δοκιμή του Crobarm. Υπήρχαν κάποιες παράμετροι από τους κώδικες, με τιμές οι οποίες έπρεπε να ρυθμιστούν κατάλληλα, για την πιο ομαλή και επιθυμητή εμπειρία χρήστη. Κυρίως, έγινε καλιμπράρισμα στον κύκλο εργασίας από τα PWM σήματα. Περιλαμβάνονται φωτογραφίες από τα μέρη και την ολοκλήρωση του συστήματος Crobarm. Οι φωτογραφίες παρουσιάζονται από το σχήμα 5.33 μέχρι το 5.52.

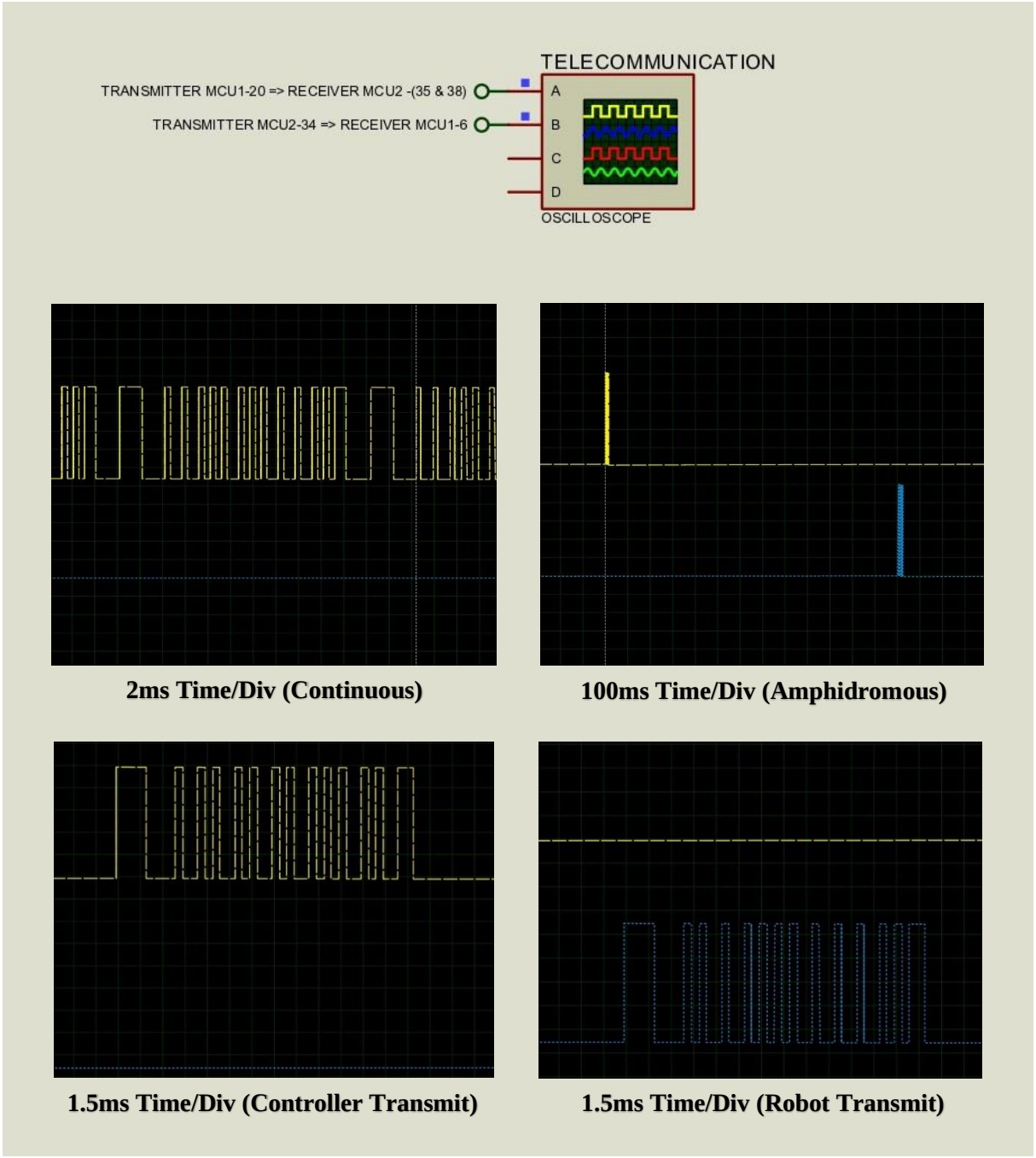
Πρέπει να αναφερθεί πως το συνολικό κόστος για το σασί του ρομπότ, βραχίονα, πλακέτες, πλεξιγκλάς, τα ηλεκτρονικά υλικά, τα εξαρτήματα, τις μπαταρίες και τη κάμερα που αγοράστηκαν ανέρχεται στα 180 ευρώ.



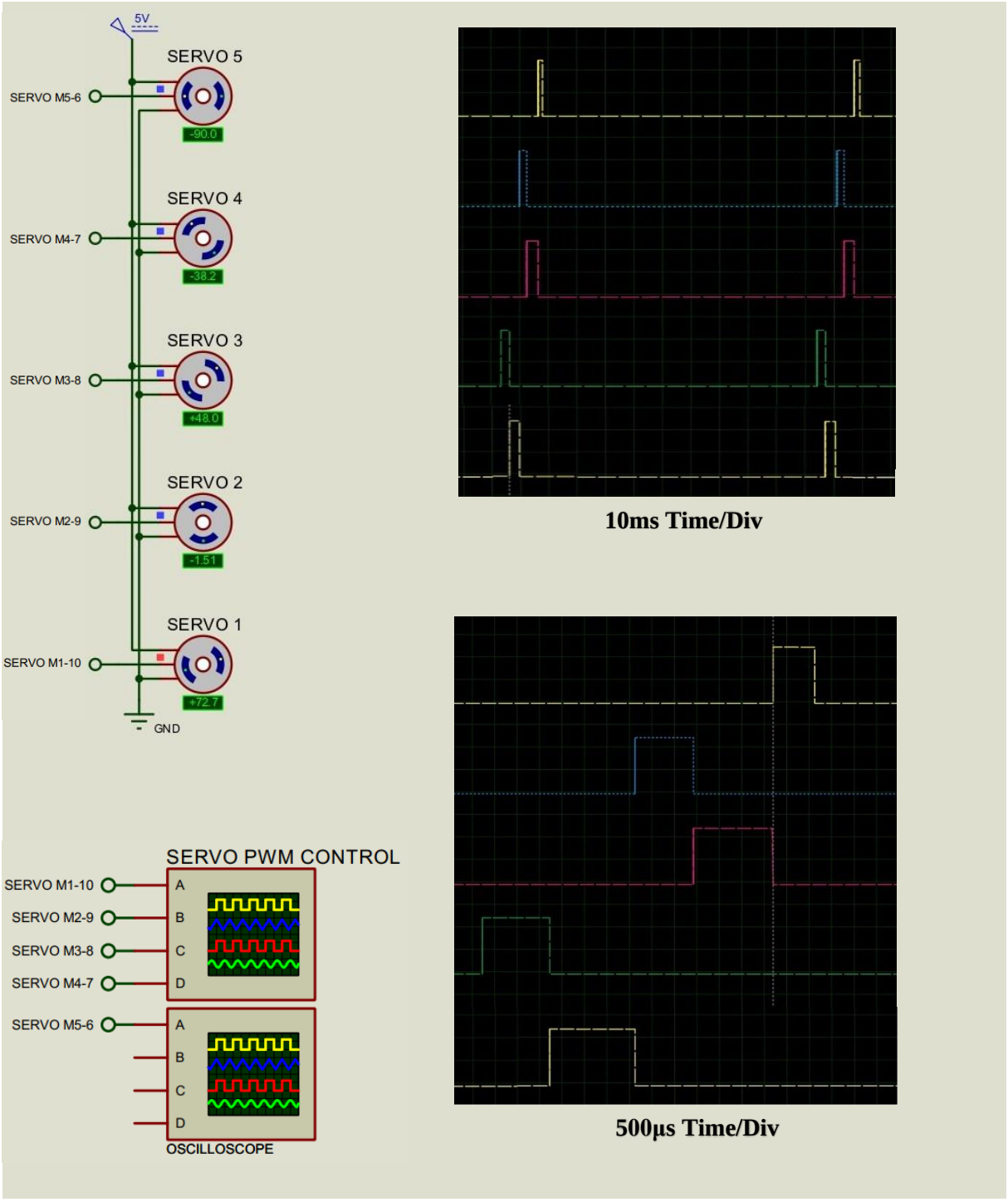

Σχήμα 5.19 Μοντέλο Προσομοίωσης Τηλεχειριστήριου Crobarm



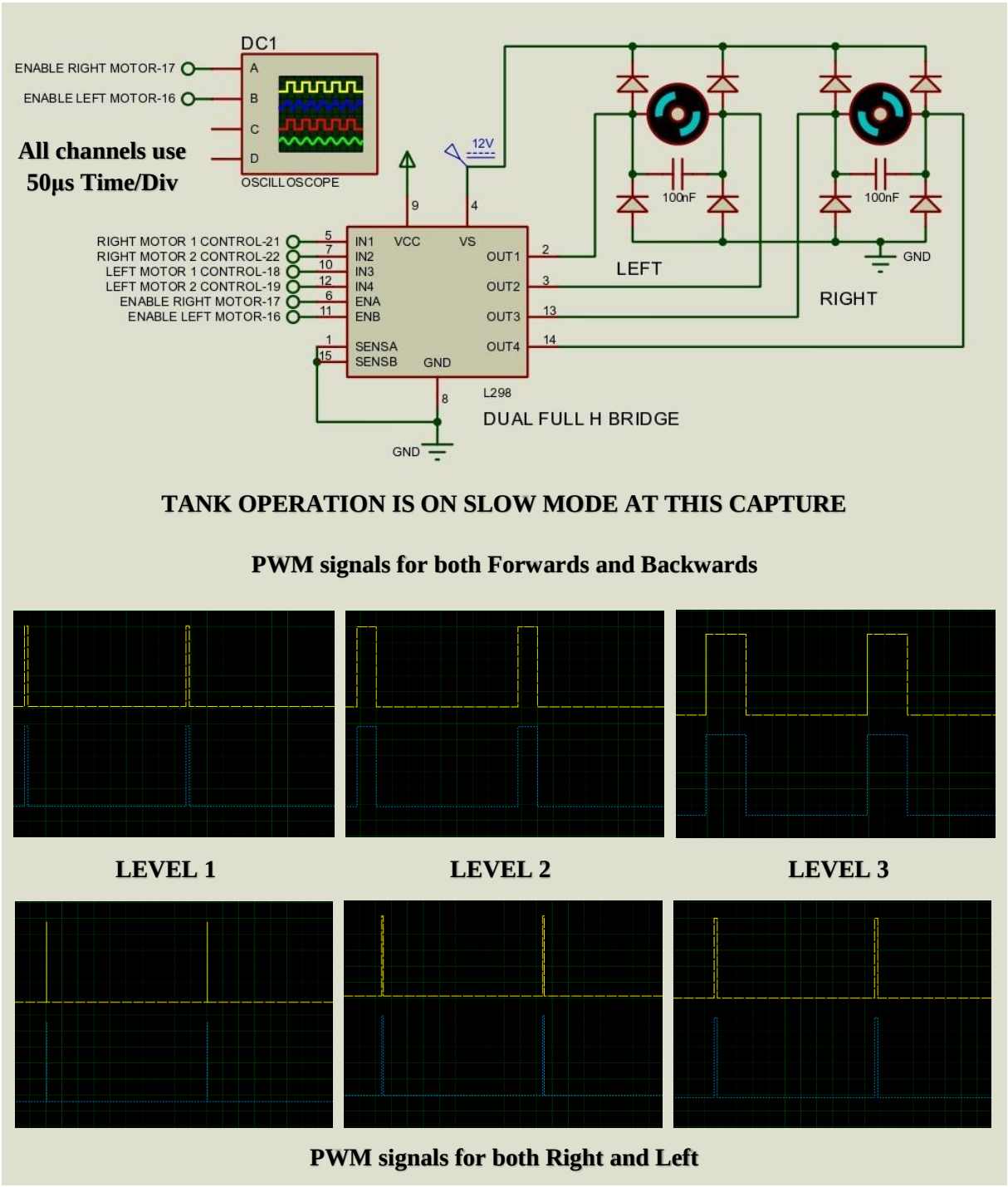

Σχήμα 5.20 Μοντέλο Προσομοίωσης Ηλεκτρονικού Συστήματος Crobarm



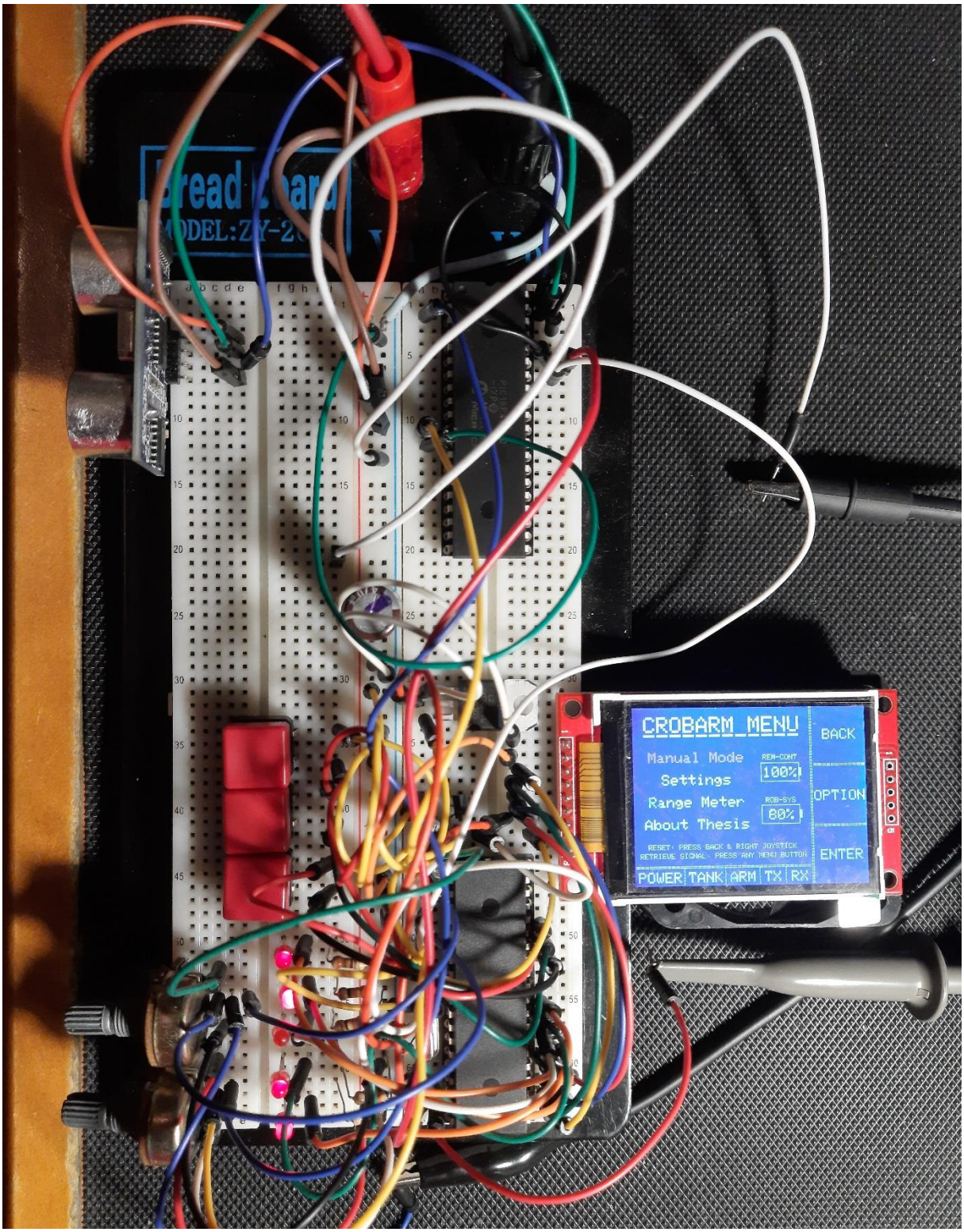

Σχήμα 5.21 Προσομοίωση Παλμογράφου για Αποστολή/Λήψη Κωδικοποιημένων Σημάτων



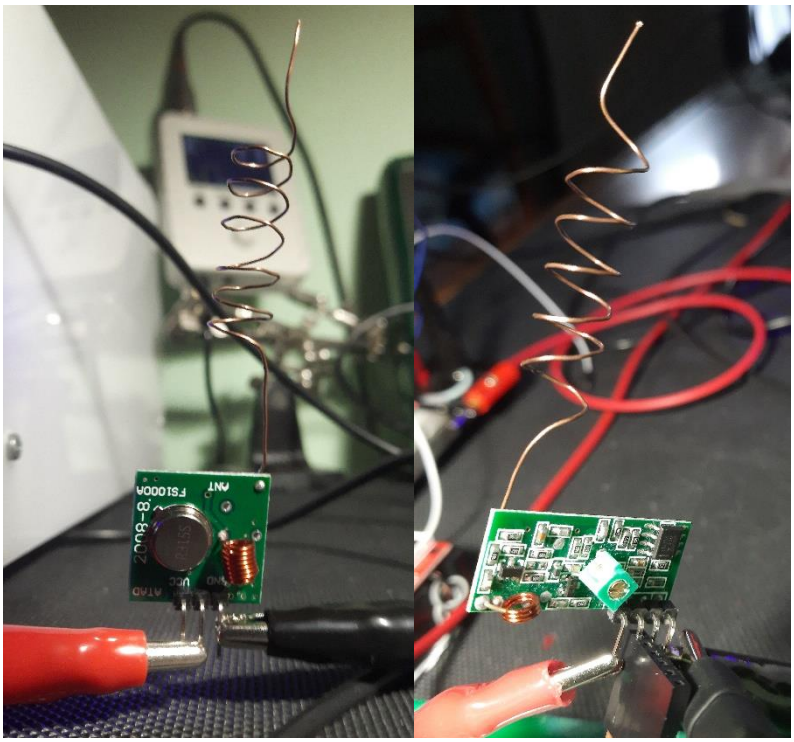

Σχήμα 5.22 Προσομοίωση Παλμογράφου για Έλεγχο Κινητήρων Σέρβο με Σήματα PWM




Σχήμα 5.23 Προσομοίωση Παλμογράφου για Έλεγχο Στροφών DC Κινητήρων με Σήματα PWM



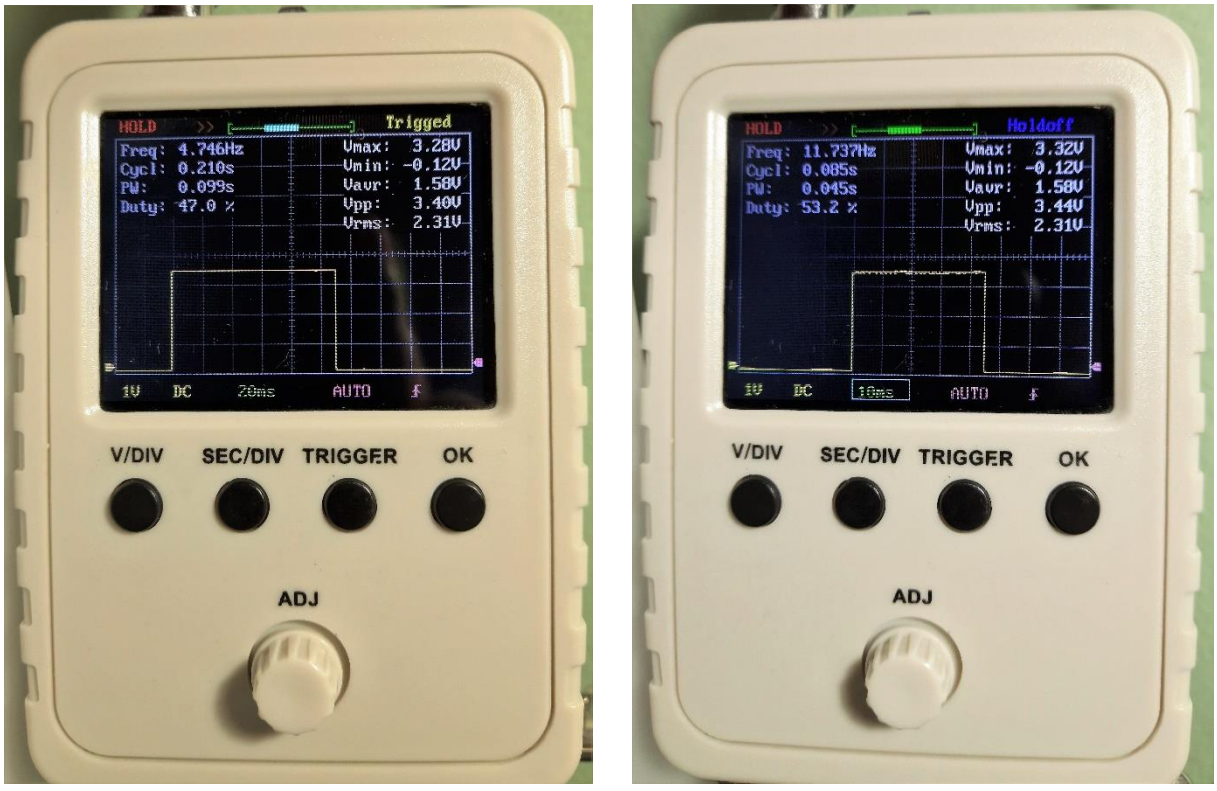

Σχήμα 5.24 Πείραμα Ρομποτικού Συστήματος Crobarm se Δοκιμαστική Πλακέτα



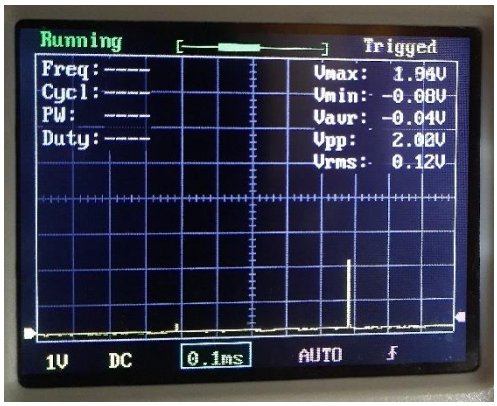
Σχήμα 5.25 Πομπός και Δέκτης Ραδιοσυχνοτήτων στα 315MHz


Σχήμα 5.26 Μονάδα Τροφοδοσίας Πειράματος


Σχήμα 5.27 Πειραματική Καταγραφή Κωδικοποιημένων Σημάτων από MCU1 προς MCU2

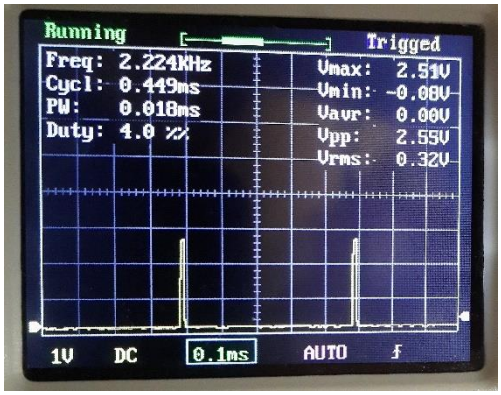



Σχήμα 5.28 Πειραματική Καταγραφή Ανταπόκρισης Τανκ και Βραχίονα



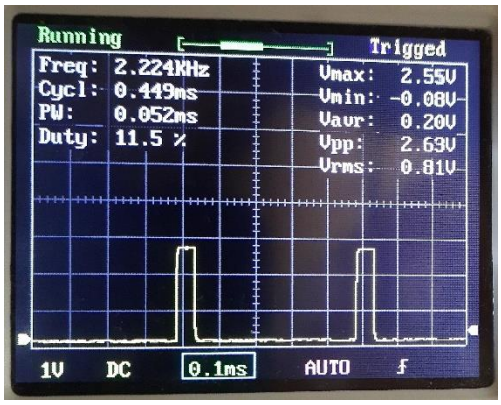

**NEUTRAL**


**All these analog voltages were read with multimeter and input pin A3 of MCU1**



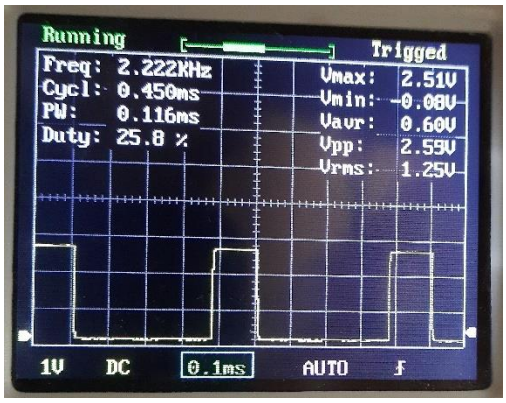

**LEVEL 1**


**Each value was captured randomly for the Backward motion of the tank**




**LEVEL 2**

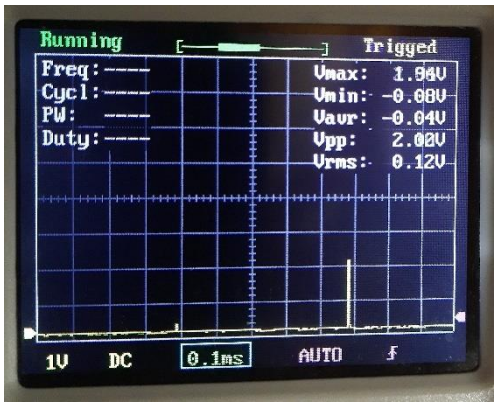

**All the values are between the permissible limit according to code**




**LEVEL 3**

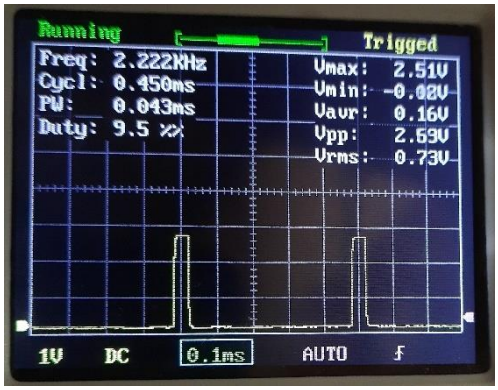

**Acceleration levels are switched depending on the voltage changes**

Σχήμα 5.29 Πειραματική Καταγραφή Σημάτων PWM για Αργή Ταχύτητα Τανκ



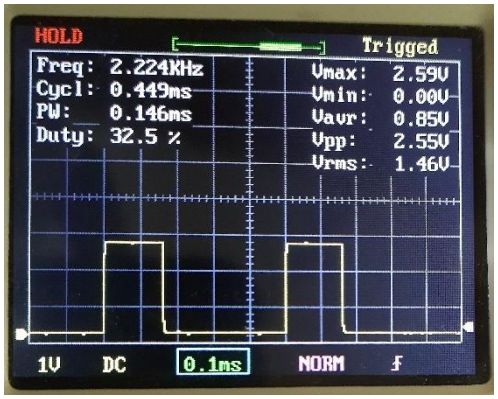

**NEUTRAL**


**All these analog voltages were read with multimeter and input pin A3 of MCU1**



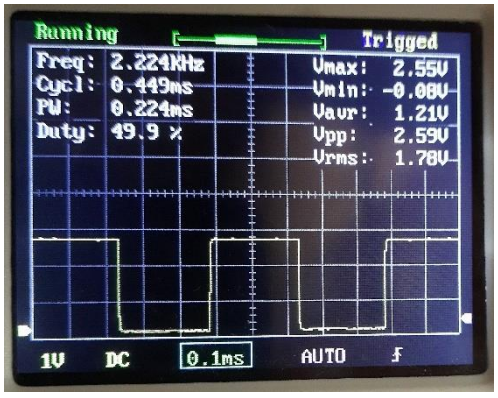

**LEVEL 1**


**Each value was captured randomly for the Backward motion of the tank**




**LEVEL 2**

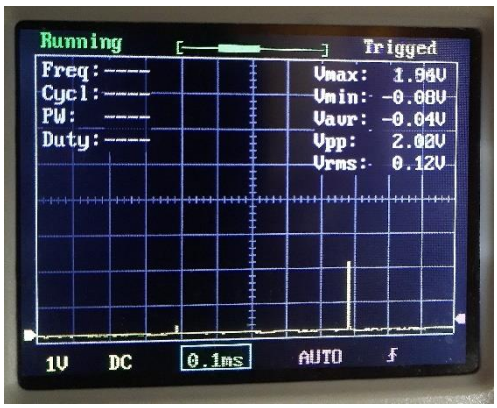

**All the values are between the permissible limit according to code**




**LEVEL 3**

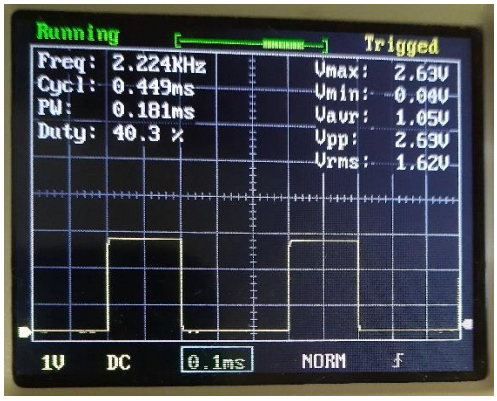

**Acceleration levels are switched depending on the voltage changes**

Σχήμα 5.30 Πειραματική Καταγραφή Σημάτων PWM για Κανονική Ταχύτητα Τανκ



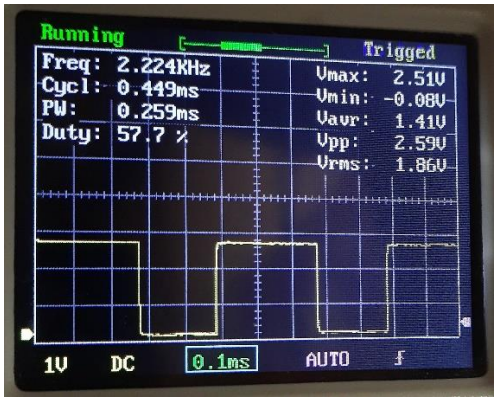

**NEUTRAL**


**All these analog voltages were read with multimeter and input pin A3 of MCU1**



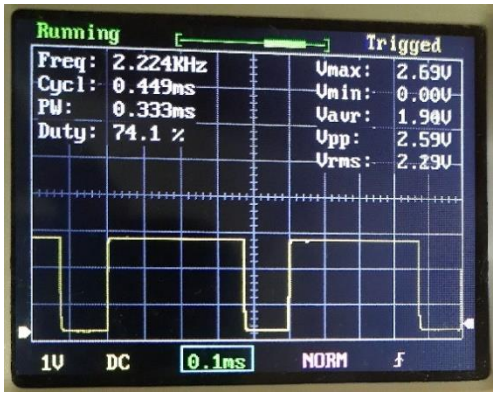

**LEVEL 1**


**Each value was captured randomly for the Backward motion of the tank**




**LEVEL 2**

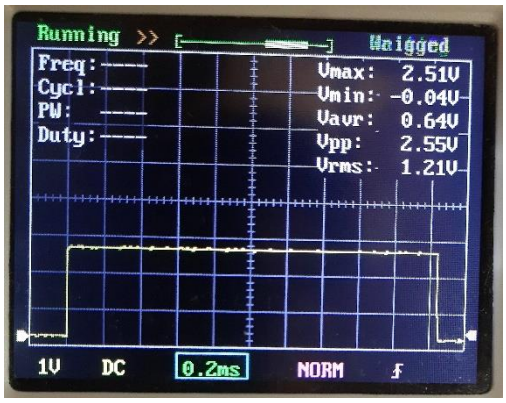

**All the values are between the permissible limit according to code**




**LEVEL 3**


**Acceleration levels are switched depending on the voltage changes**

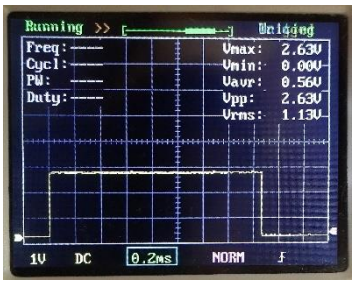
Σχήμα 5.31 Πειραματική Καταγραφή Σημάτων PWM για Γρήγορη Ταχύτητα Τανκ

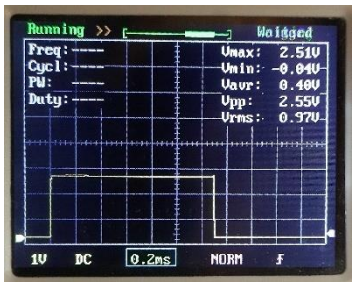


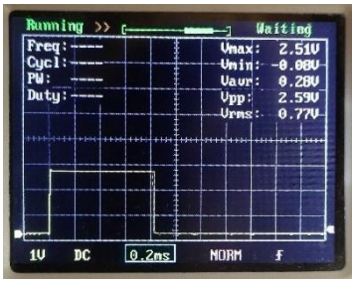

**NEUTRAL**

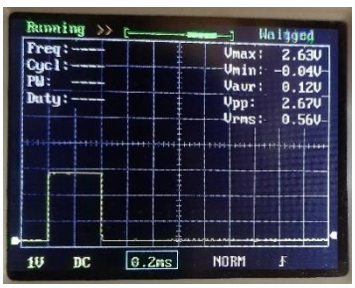

**The oscilloscope shows in different time periods how the pulse is changing. Falling Pulse means that the Servo Motor is rotating left. While rising pulse provides rotation to right**

**Above 1.55V the pulse is falling**

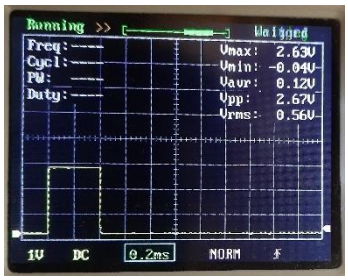


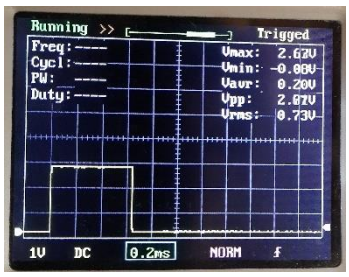



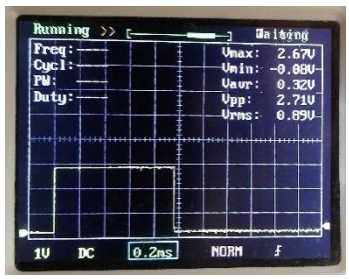



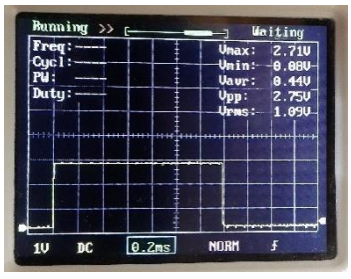




**Below 1.32V the pulse is rising**

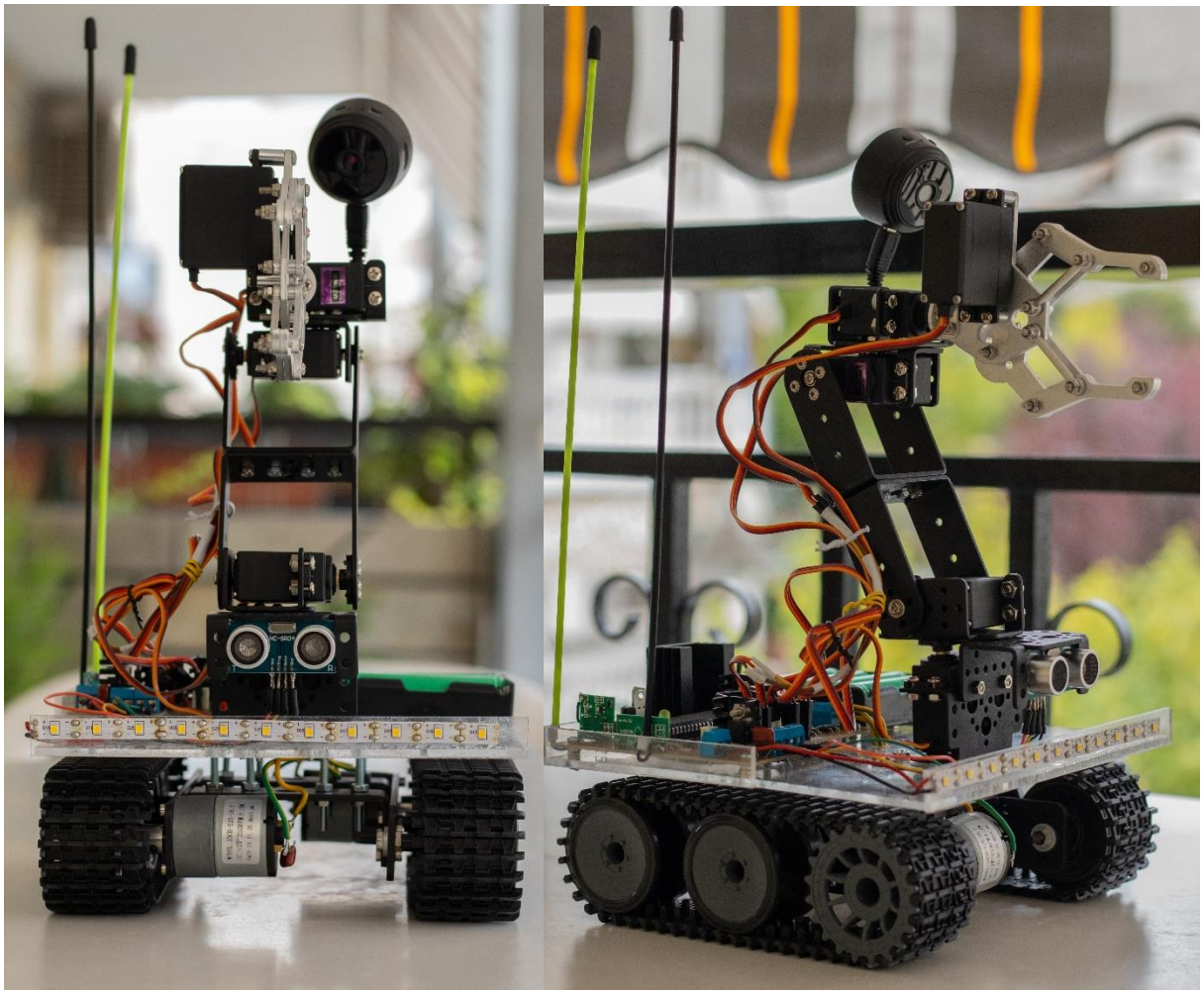


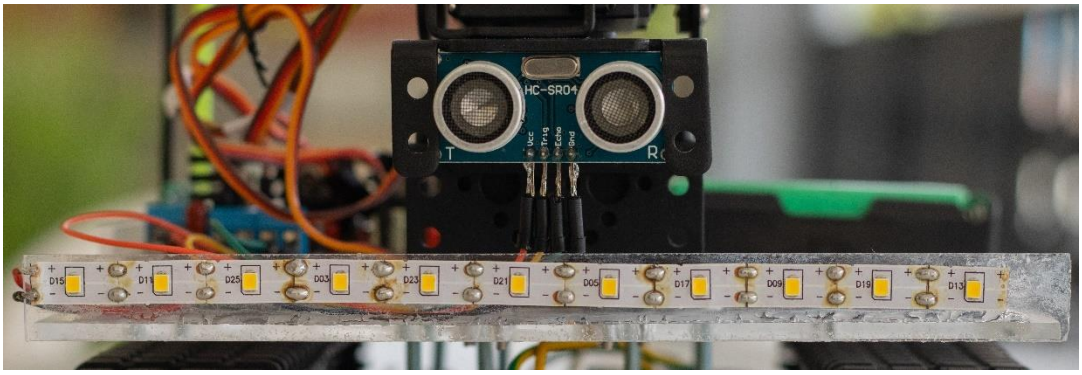




Σχήμα 5.32 Πειραματική Καταγραφή Σήματος PWM για Έλεγχο Κινητήρα Σέρβο

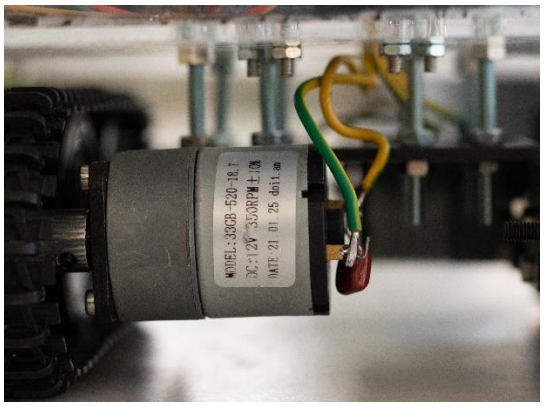


Σχήμα 5.33 Πρόσοψη και Πλάγια Όψη Ρομποτικού Συστήματος Crobarm

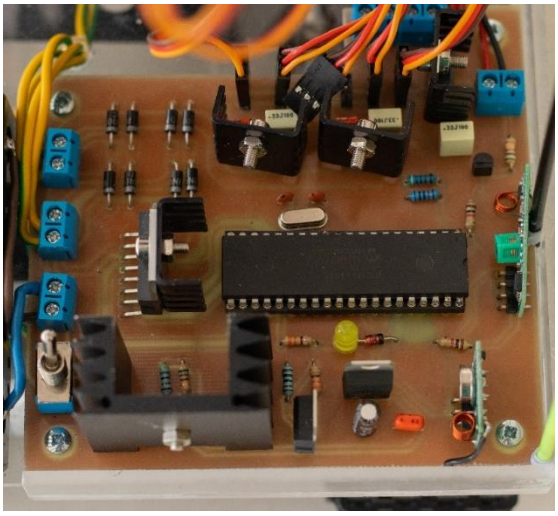



Σχήμα 5.34 Αισθητήρας Υπερήχων HC-SR04 και Ταινία με 11 LED

Σχήμα 5.35 Σύστημα Τροφοδοσίας Crobarm από Μπαταρίες

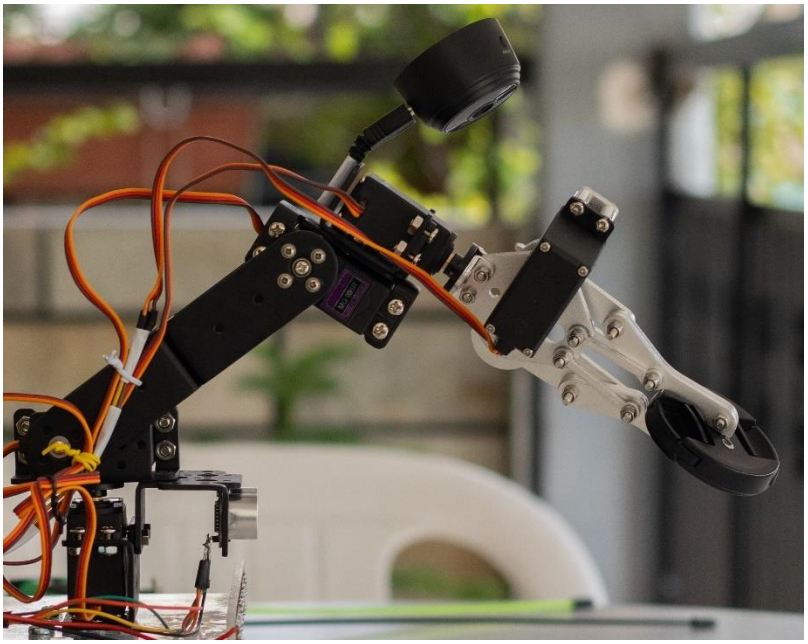


Σχήμα 5.36 DC Κινητήρες με Πυκνωτή 100nF

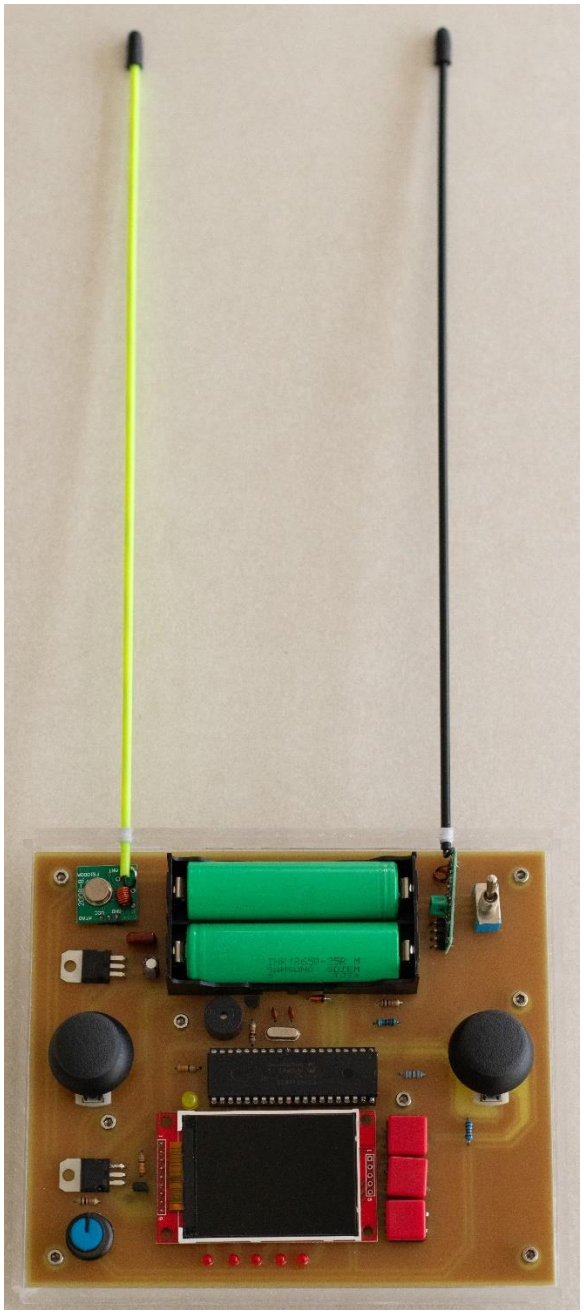


Σχήμα 5.38 Ηλεκτρονικό Σύστημα Crobarm

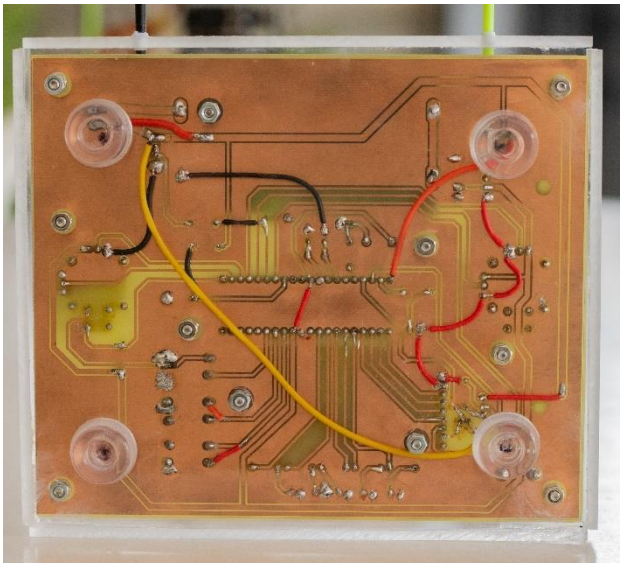
Σχήμα 5.37 Κεραίες Crobarm



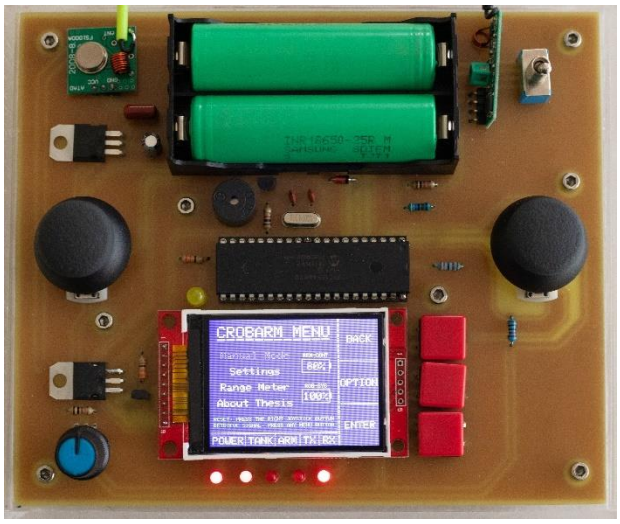
Σχήμα 5.39 Ρομποτικός Βραχίονας και Ασύρματη Κάμερα



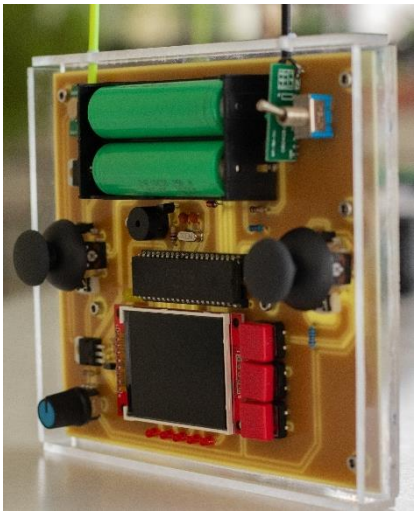
Σχήμα 5.40 Πρόσοψη Τηλεχειριστήριου με Σώμα



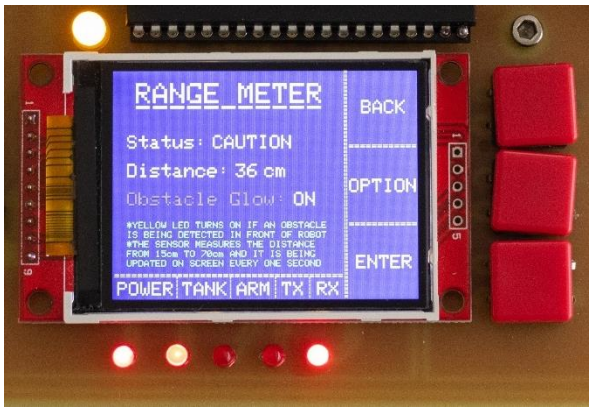
Σχήμα 5.41 Πίσω Όψη Τηλεχειριστήριου με Ποδαράκια Βάσης



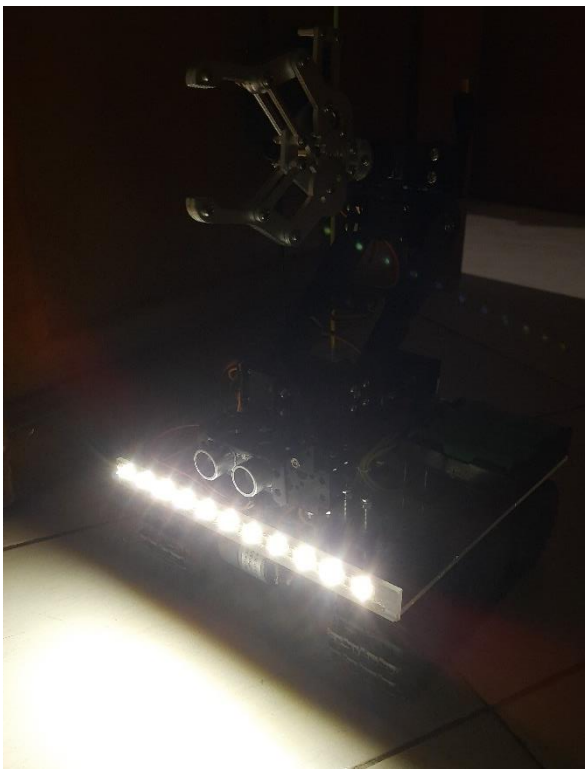

Σχήμα 5.42 Πρόσοψη Τηλεχειριστήριου στο Μενού



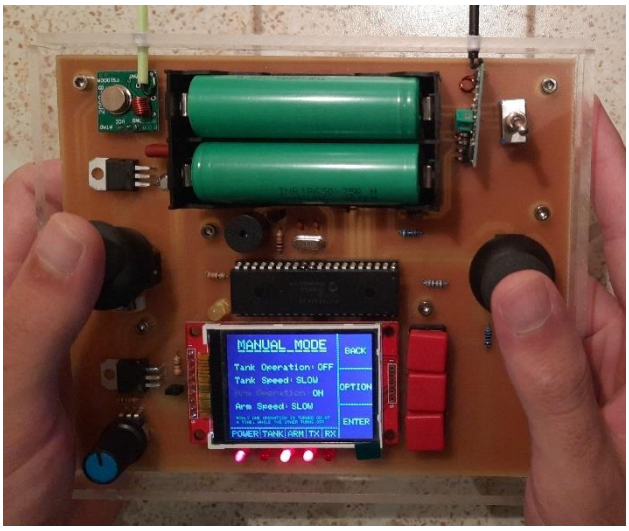
Σχήμα 5.43 Πλάγια Όψη Τηλεχειριστήριου με Σώμα




Σχήμα 5.44 Μέτρηση Απόστασης Σχήματος 5.48 Φάση 1

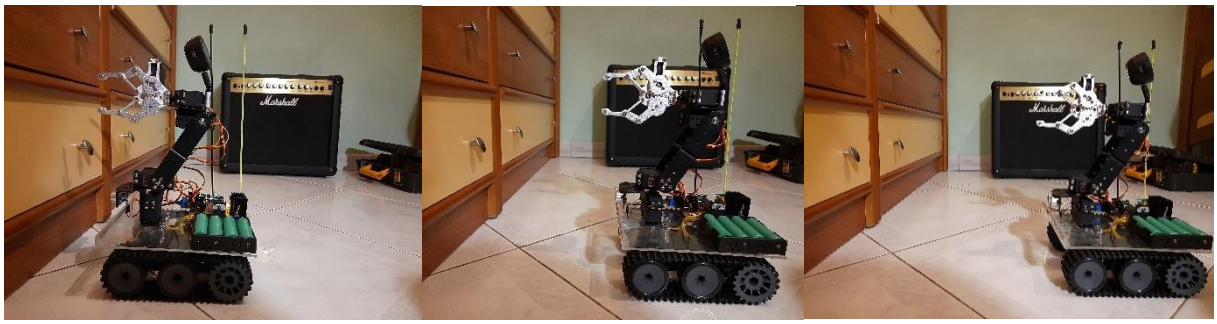


Σχήμα 5.46 Ενεργοποιημένα Φώτα LED




Σχήμα 5.45 Ενεργοποιημένη η Εργασία Βραχίονας

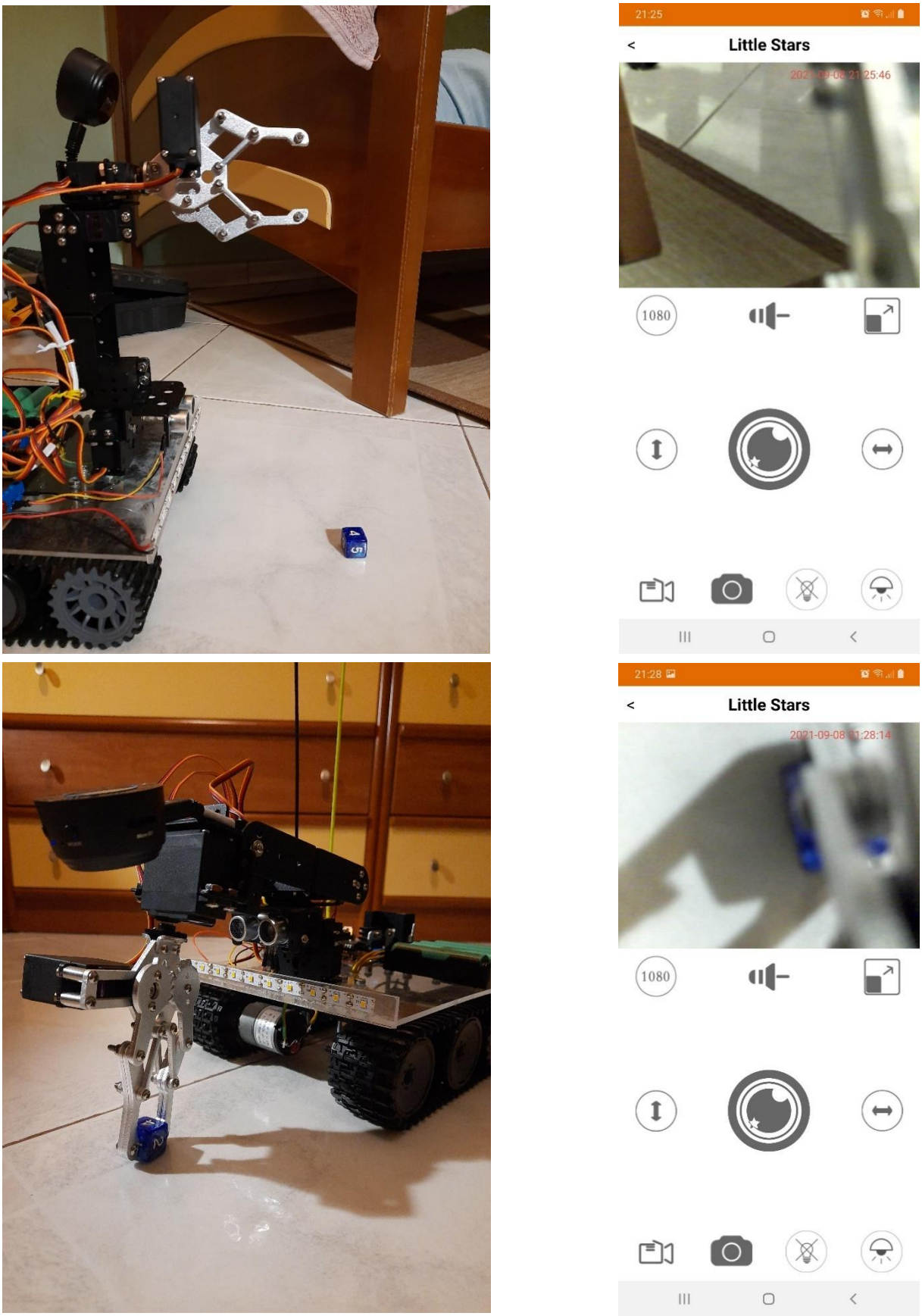
Φάση 3 Φάση 2 Φάση 1



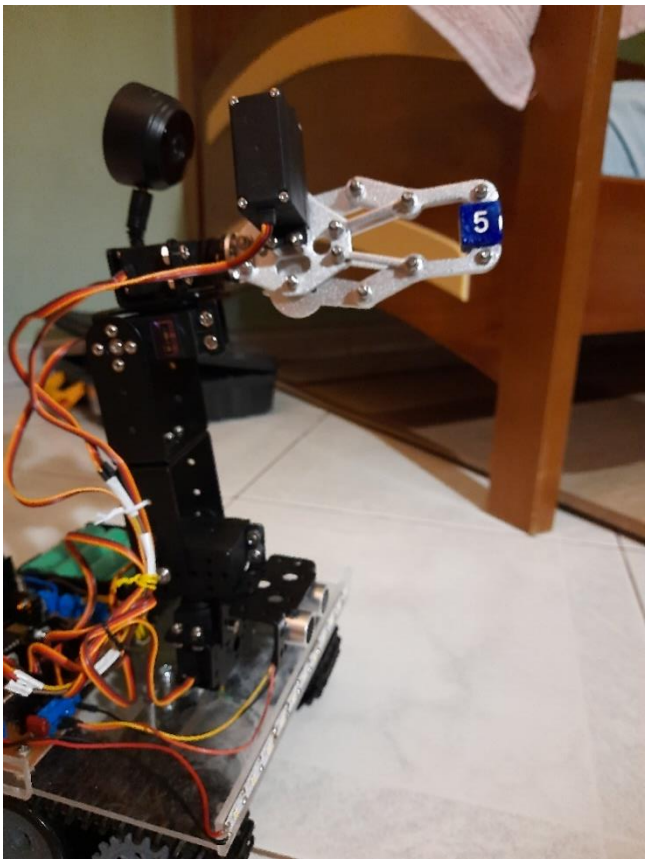
Σχήμα 5.47 Πλησιάζει το Ρομπότ στο Εμπόδιο Σιγά-Σιγά

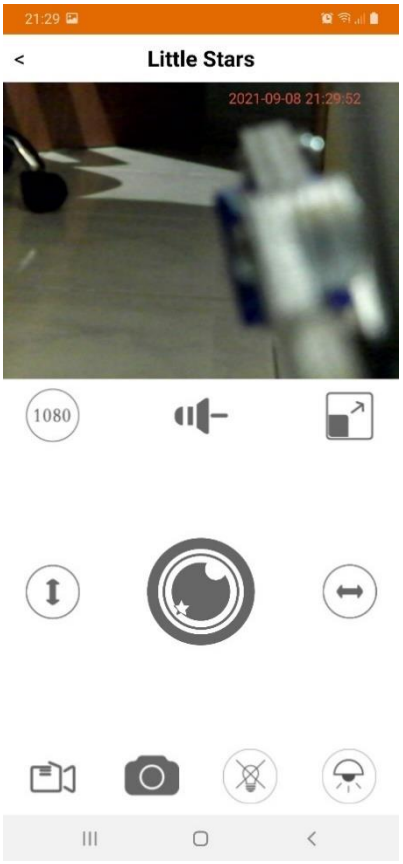
Φάση 4

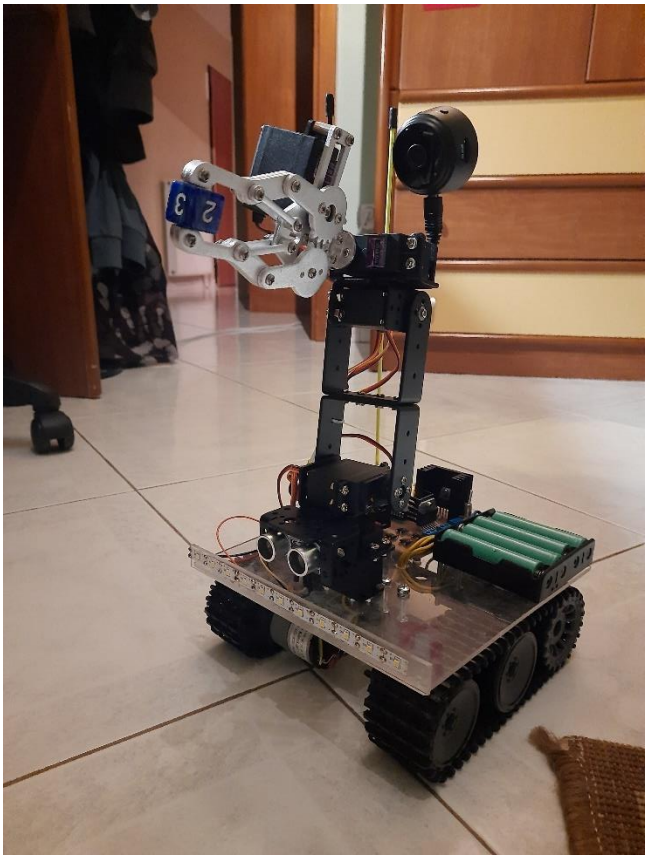
Σχήμα 5.48 Αποφυγή Σύγκρουσης με Ενεργοποιημένη την Ασφάλεια Κρούσης

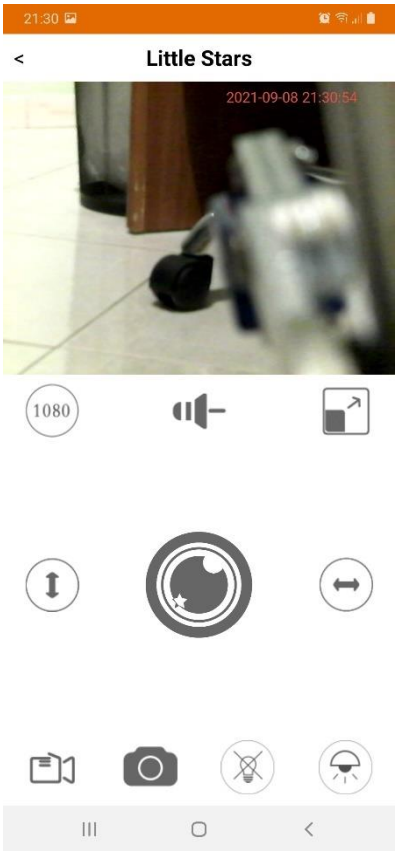



Σχήμα 5.49 Λειτουργία Βραχίονα/Τανκ για Μεταφορά ενός Ζαριού από Σημείο Α σε Σημείο Β – 1

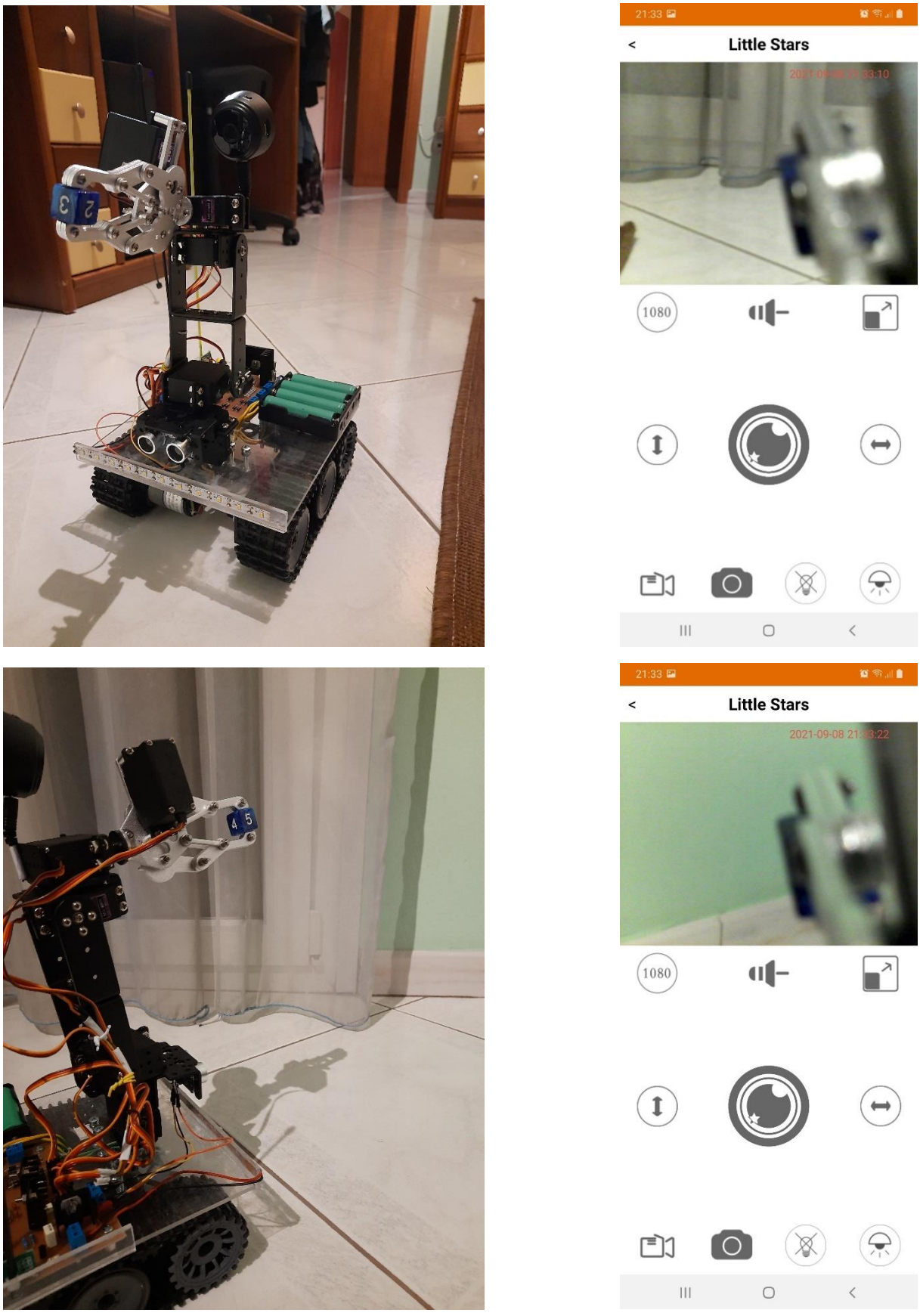




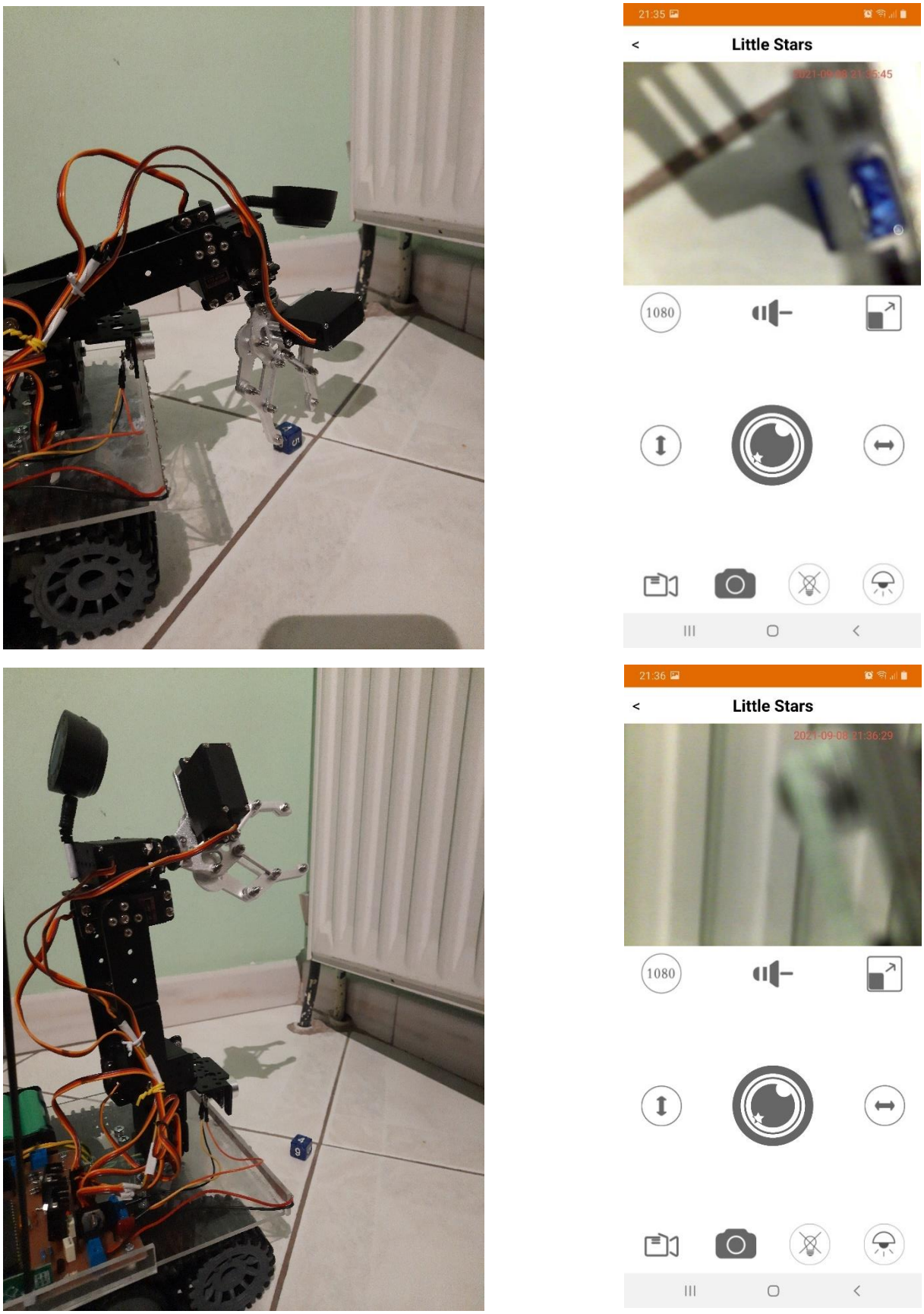






Σχήμα 5.50 Λειτουργία Βραχίονα/Τανκ για Μεταφορά ενός Ζαριού από Σημείο Α σε Σημείο Β – 2




Σχήμα 5.51 Λειτουργία Βραχίονα/Τανκ για Μεταφορά ενός Ζαριού από Σημείο Α σε Σημείο Β – 3




Σχήμα 5.52 Λειτουργία Βραχίονα/Τανκ για Μεταφορά ενός Ζαριού από Σημείο Α σε Σημείο Β – 4

# Κεφάλαιο 6ο: Συμπεράσματα και Προτάσεις Βελτίωσης

Ανακεφαλαιώνοντας, με την ολοκλήρωση της παρούσας πτυχιακής εργασίας, διαπιστώθηκε πως με τη χρήση δύο μικροελεγκτών PIC18F46K20, το σχεδιασμό των PCB και τη τελική κατασκευή, μπόρεσε να δημιουργηθεί ένα αξιοπρεπές Μη Επανδρωμένο Όχημα, διαθέτοντας τηλεχειρισμό. Παρέχεται μεγάλη ακρίβεια στις κινήσεις του οχήματος/βραχίονα, για την περάτωση λεπτομερών εργασιών, παρακολούθηση, μεταφορά αντικειμένων, εκτέλεση σοβαρών αποστολών και καταγραφή δεδομένων. Ειδικότερα, αποδείχθηκε σπουδαία προσθήκη, η ασύρματη ψηφιακή κάμερα για τον απομακρυσμένο έλεγχο. Έτσι, διευκολύνεται μέσω οθόνης ο χειριστής, για τις κινήσεις που πρέπει να πράξει το ρομπότ, με μέγιστη απόσταση επικοινωνίας τα 60 μέτρα σε ανοιχτό χώρο.

Σημαντική βελτίωση θα μπορούσε να θεωρηθεί, η αύξηση του τηλεπικοινωνιακού εύρους για πομπούς και δέκτες. Η απόσταση ελέγχου θα αυξηθεί σημαντικά, στέλνοντας το Crobarm σε πιο απομακρυσμένα σημεία. Επομένως, θα πρέπει να μην χάνεται εύκολα το σήμα, αν βρεθεί κοντά σε τοίχο ή άλλα εξωτερικά εμπόδια μειώνοντας την ισχύ μεταδιδόμενου σήματος. Επίσης, η αλλαγή κάμερας με καλύτερη ανάλυση και μεγαλύτερη διάρκεια ζωής, μαζί με την δυνατότητα μεγαλύτερης εμβέλειας.

# Βιβλιογραφία


**Βιβλία**

[1] Κ. Πεκμεστζή, Καθηγητής Ε.Μ.Π, "Συστήματα Μικροϋπολογιστών ΙΙ, Μικροελεγκτές AVR και PIC", Αθήνα 2009.

[2] Γιακουμής Άγγελος, Λέκτορας ΔΙ.ΠΑ.Ε, "Εργαστήριο στα Ενσωματωμένα Συστήματα", Σημειώσεις, Θεσσαλονίκη

[3] Δημητριάδης Παναγιώτης, Καθηγητής ΔΙ.ΠΑ.Ε, "Ηλεκτροκίνηση και Ευφυή Δίκτυα", Σημειώσεις, Θεσσαλονίκη

[4] Ζωή Δουλγέρη, Καθηγήτρια Α.Π.Θ, "Ρομποτική – Κινηματική, δυναμική και έλεγχος αρθρωτών βραχιόνων", Ιανουάριος 2007.

[5] Nebojsa Matic, CEO of Mikroelectronika, "The PIC microcontroller", Book 1, Serbia

[6] Milan Verle, "PIC mikrokontroleri", 2008

[7] Δημήτριος Β. Νικολός, Καθηγητής Πανεπιστήμιου Πατρών, "Αρχιτεκτονική Υπολογιστών", 1η Έκδοση, Ιούλιος 2012

**Data Sheet**

[8] Servomotor, "MG996R High toque Metal Fear Dual Ball Bearing Servo", datasheet

[9] STMicroelectronics, "LOW DROP FIXED AND ADJUSTABLE POSITIVE VOLTAGE REGULATORS", LD1117 SERIES datasheet

[10] STMicroelectronics, "1.2V to 37V adjustable voltage regulators", LM217, LM317 datasheet

[11] ELEC Freaks, "Ultrasonic Ranging Module HC – SR04", datasheet

[12] Microchip Technology Inc, "28/40/44-Pin Flash Microcontrollers with XLP Technology", PIC18F2XK20/4XK20 datasheet, 2010-2015

[13] STMicroelectronics, "Dual Full-Bridge Driver", L298 datasheet, January 2000

**Internet Site**

[14] Schmidt Ocean Institute, ROV FAQS, https://schmidtocean.org/education/rov-faqs/

[15] Servos Explained, Sparkfun, https://www.sparkfun.com/servos

[16] Last Minute ENGINEERS, Interface L298N DC Motor Driver Module with Arduino, https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

[17] Last Minute ENGINEERS, How 433MHz RF Work, https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/

[18] Wikipedia, Electric Motor, https://en.wikipedia.org/wiki/Electric_motor

[19] RS Components, Robotic Arm, https://uk.rs-online.com/web/generalDisplay.html?id=ideas-

and-advice/robotic-arms-guide

[20] Wikipedia, UGV, https://en.wikipedia.org/wiki/Unmanned_ground_vehicle

[21] Science online, UGV, https://www.online-sciences.com/robotics/unmanned-ground-vehicle-ugv-advantages-and-disadvantages/

[22] Wikipedia, UAV, https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[23] Deep Trekker, Underwater ROVs, https://www.deeptrekker.com/resources/underwater-rovs

[24] GeeksforGeeks, Difference between Von Neumann and Harvard Architecture, https://www.geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/

[25] International Journal of Applied Science – Research and Review, Humanoid Robot: Application and Influence, https://www.imedpub.com/articles/humanoid-robot--application-and-influence.php?aid=23790

[26] Science online, UAV, https://www.online-sciences.com/robotics/unmanned-aerial-vehicle-uses-advantages-and-disadvantages/

**Εικόνες Διαδικτύου**

[27] https://www.researchgate.net/figure/Common-robot-arm-configurations_fig1_244270445

[28] https://www.electronicsweekly.com/blogs/mannerisms/dilemmas/year-of-the-humanoids-2018-01/

[29] https://online.jcu.edu.au/blog/robot-complements-patient-care-in-australian-first

[30] https://en.wikipedia.org/wiki/General_Atomics_MQ-9_Reaper#/media/File:MQ-9_Reaper_UAV_(cropped).jpg

[31] https://www.dji.com/gr/phantom-4-pro

[32] https://www.ecagroup.com/en/solutions/h1000-rov-remotely-operated-vehicle

[33] https://twitter.com/SchmidtOcean/status/754213369733844992/photo/1

[34] https://www.microchip.com/en-us/product/PIC18F46K20

[35] https://www.aliexpress.com/item/32323810883.html?spm=a2g0o.productlist.0.0.72bd4ad7Dyaqvq&algo_pvid=ba897304-69b4-47c3-8cc0-13875fd43f1a&algo_exp_id=ba897304-69b4-47c3-8cc0-13875fd43f1a-1

[36] https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

[37] https://megaoffers.gr/product/ip-camera-wifi-mini-%ce%b19-1080p-883570/

[38] https://www.army-technology.com/projects/centaur-unmanned-ground-vehicle/

[39] https://rheinmetall-defence.com/en/rheinmetall_defence/systems_and_products/unbemannte_fahrzeuge/mission_master/index.php

[40] https://elements.envato.com/robotic-arm-modern-industrial-technology-automated-Z4QG2TL

# Παραρτήματα

## A. Βιβλιοθήκη PIC18F46K20.h για το Τηλεχειριστήριο

```
#include <18F46K20.h>
#device ADC=10

#FUSES NOWDT
#FUSES WDT128
#FUSES NOPROTECT
#FUSES BROWNOUT
#FUSES PUT
#FUSES H4
#FUSES NOCPD
#FUSES STVREN
#FUSES NODEBUG
#FUSES NOLVP
#FUSES NOWRT
#FUSES NOWRTD
#FUSES NOIESO
#FUSES NOFCMEN
#FUSES NOPBADEN
#FUSES NOWRTC
#FUSES NOWRTB
#FUSES NOEBTR
#FUSES NOEBTRB
#FUSES NOCPB
#FUSES MCLR
#FUSES NOLPT1OSC
#FUSES NOXINST
#FUSES WRTB
#FUSES NOLPT1OSC
#FUSES CCP2C1
#FUSES NOCPB
#FUSES NOCPD
#FUSES NOWRTC
#FUSES NOWRTD
#FUSES NOEBTR
#FUSES NOEBTRB

#use delay(clock=64M, RESTART_WDT)
```

## B. Βιβλιοθήκη PIC18F46K20.h για το Ρομπότ

```
#include <18F46K20.h>
#device ADC=10

#FUSES NOWDT
#FUSES WDT128
```

```
#FUSES NOPROTECT
#FUSES BROWNOUT
#FUSES PUT
#FUSES H4
#FUSES NOCPD
#FUSES STVREN
#FUSES NODEBUG
#FUSES NOLVP
#FUSES NOWRT
#FUSES NOWRTD
#FUSES NOIESO
#FUSES NOFCMEN
#FUSES NOPBADEN
#FUSES NOWRTC
#FUSES NOWRTB
#FUSES NOEBTR
#FUSES NOEBTRB
#FUSES NOCPB
#FUSES MCLR
#FUSES NOLPT1OSC
#FUSES NOXINST
#FUSES WRTB
#FUSES NOLPT1OSC
#FUSES CCP2C1
#FUSES NOCPB
#FUSES NOCPD
#FUSES NOWRTC
#FUSES NOWRTD
#FUSES NOEBTR
#FUSES NOEBTRB

#use delay(clock=64M, RESTART_WDT)
```

## Γ. Βιβλιοθήκη TFT.h για το Τηλεχειριστήριο

```
#define TFT_CS  PIN_D2
#define TFT_RST PIN_D3
#define TFT_DC  PIN_C4

#use SPI(SPI1,MODE=0,BITS=8,STREAM=ILI9341,BAUD=5000000)
#include <ILI9341.h>
#include <GFX_Library.h>

void bootScreen(void);
void menuScreen(void);
void printExtraMenu(void);
void printMainMenu(void);
void deleteMainMenu(void);
void printManualMode(void);
void deleteManualMode(void);
void printTankOnArmOff(void);
```

```
void deleteTankOnArmOff(void);
void printTankOffArmOn(void);
void deleteTankOffArmOn(void);
void printTankSpeed(void);
void deleteTankSpeed(void);
void printArmSpeed(void);
void deleteArmSpeed(void);
void printSettings(void);
void deleteSettings(void);
void printCrashSafety(void);
void printHeadlights(void);
void printSystemSound(void);
void printBrightnessDisplay();
void printRangeMeter(void);
void deleteRangeMeter(void);
void printStatus(void);
void printAboutThesis(void);
void deleteAboutThesis(void);

unsigned long checkProtocolPart=0,countBinaryCode=0,codedCommandRead=0;
short flagControllerBattery100=1,flagControllerBattery80=1,flagControllerBattery60=1;
short flagControllerBattery40=1,flagControllerBattery20=1,flagControllerBatteryLow=1;
int flagTankSpeed=1,switchTankSpeed=1,lastTankSpeed=1;
int flagArmSpeed=1,switchArmSpeed=0,lastArmSpeed=0,flagRobotBattery=0;
short flagObstacleGlow=1,switchObstacleGlow=0,lastObstacleGlow=0;
short flagSystemSound=1,switchSystemSound=0,lastSystemSound=0;
int flagHeadlights=1,switchHeadlights=0,lastHeadlights=0;
int flagBrightnessDisplay=1,switchBrightnessDisplay=5,lastBrightnessDisplay=5;
short flagDanger=1,flagCaution=1,flagSafe=1,generalSettings=1;
int lastDistance=0,newDistance=0;
short flagCrashSafety=1,switchCrashSafety=0,lastCrashSafety=0;
short flagEnterAboutThesis=1,keepRobotBattery=1;

void bootScreen(void)
{
   unsigned int i,w;

   display_setRotation(3);
   display_fillScreen(ILI9341_BLACK);
   display_setTextColor(ILI9341_WHITE);

   display_setTextSize(2);
   display_setCursor(7,10);
   display_print("INTERNATIONAL\r\n");
   display_setCursor(9,30);
   display_print("HELLENIC\r\n");
   display_setCursor(9,50);
   display_print("UNIVERSITY\r\n");

   display_setCursor(182,10);
   display_print("SCHOOL\r\n");
   display_setCursor(182,30);
   display_print("OF\r\n");
   display_setCursor(182,50);
```

```
    display_print("ENGINEERING\r\n");

    display_setTextSize(1);
    for(i=5;i<65;i+=8)
    {
       display_setCursor(169,i);
       display_print("|\r\n");
    }

    display_setTextSize(2);
    display_setCursor(11,85);
    display_print("DEPARTMENT OF INFORMATION\r\n");
    display_setCursor(5,105);
    display_print("AND ELECTRONIC ENGINEERING\r\n");

    display_setTextSize(1);
    display_setCursor(140,137);
    display_print("THESIS\r\n");
    display_setTextSize(3);
    display_setCursor(80,160);
    display_print("<CROBARM>\r\n");

    display_setTextSize(1);
    display_setCursor(150,220);
    display_print("SUPERVISOR AGGELOS GIAKOUMIS\r\n");
    display_setCursor(96,230); //144
    display_print("UNDERGRADUATE STUDENT STEFANOS LIAKAS\r\n");

    display_setCursor(2,231);
    printf(display_print,"BOOTING\r\n");

    for(w=0;w<5;w++)
    {
       for(i=45;i<60;i+=5)
       {
          display_setCursor(i,231);
          display_setTextColor(ILI9341_WHITE);
          printf(display_print,".");
          delay_ms(500);
       }
       for(i=45;i<60;i+=5)
       {
          display_setCursor(i,231);
          display_setTextColor(ILI9341_BLACK);
          printf(display_print,".");
       }
    }
    display_setTextColor(ILI9341_BLACK);
    display_setCursor(2,231);
    printf(display_print,"BOOTING\r\n");
    display_setCursor(2,231);
    display_setTextColor(ILI9341_WHITE);
    printf(display_print,"READY\r\n");
}
```

```
void printExtraMenu(void)
{
   unsigned int i;

   //CONTROLLER BATTERY
   display_setTextSize(1);
   for(i=175;i<=225;i+=3)
   {
      display_setCursor(i,70);
      display_print("_\r\n");
   }
   for(i=175;i<=225;i+=3)
   {
      display_setCursor(i,95);
      display_print("_\r\n");
   }
   for(i=76;i<=94;i+=3)
   {
      display_setCursor(173,i);
      display_print("|\r\n");
   }
   for(i=76;i<=94;i+=3)
   {
      display_setCursor(225,i);
      display_print("|\r\n");
   }
   for(i=83;i<=87;i+=2)
   {
      display_setCursor(226,i);
      display_print("|\r\n");
      display_setCursor(227,i);
      display_print("|\r\n");
      display_setCursor(228,i);
      display_print("|\r\n");
   }
   display_setCursor(178,65);
   display_print("REM-CONT\r\n");

   //ROBOT BATTERY
   for(i=175;i<=225;i+=3)
   {
      display_setCursor(i,127);
      display_print("_\r\n");
   }
   for(i=175;i<=225;i+=3)
   {
      display_setCursor(i,152);
      display_print("_\r\n");
   }

   for(i=133;i<=153;i+=3)
   {
      display_setCursor(173,i);
      display_print("|\r\n");
```

```
}
   for(i=133;i<=153;i+=3)
   {
      display_setCursor(225,i);
      display_print("|\r\n");
   }

   for(i=140;i<=144;i+=2)
   {
      display_setCursor(226,i);
      display_print("|\r\n");
      display_setCursor(227,i);
      display_print("|\r\n");
      display_setCursor(228,i);
      display_print("|\r\n");
   }
   display_setCursor(181,123);
   display_print("ROB-SYS\r\n");

   display_setTextColor(ILI9341_GREENYELLOW);
   display_setTextSize(1);
   display_setCursor(8,185);
   display_print("RESET: PRESS THE RIGHT JOYSTICK BUTTON\r\n");
   display_setCursor(8,197);
   display_print("RETRIEVE SIGNAL: PRESS ANY MENU BUTTON\r\n");
}

void menuScreen(void)
{
   unsigned int i;

   display_setRotation(3);
   display_setTextColor(ILI9341_WHITE);
   display_fillScreen(ILI9341_NAVY);

   display_setTextSize(1);
   for(i=0;i<240;i+=8)
   {
      display_setCursor(243,i);
      display_print("|\r\n");
      display_setCursor(241,i);
      display_print("|\r\n");
   }
   for(i=216;i<239;i+=8)
   {
      display_setCursor(65,i);
      display_print("|\r\n");
   }
   for(i=216;i<239;i+=8)
   {
      display_setCursor(121,i);
      display_print("|\r\n");
   }
   for(i=216;i<239;i+=8)
```

```
    {
      display_setCursor(168,i);
      display_print("|\r\n");
    }
    for(i=216;i<239;i+=8)
    {
      display_setCursor(204,i);
      display_print("|\r\n");
    }
    display_setCursor(248,76);
    display_print("============");
    display_setCursor(248,158);
    display_print("============");
    display_setCursor(2,212);
    display_print("========================================");

    display_setTextSize(2);
    display_setCursor(261,31);
    display_print("BACK");
    display_setCursor(248,114);
    display_print("OPTION");
    display_setCursor(255,194);
    display_print("ENTER");
    display_setCursor(2,219);

    display_setCursor(5,222);
    display_print("POWER\r\n");
    display_setCursor(73,222);
    display_print("TANK\r\n");
    display_setCursor(130,222);
    display_print("ARM\r\n");
    display_setCursor(177,222);
    display_print("TX\r\n");
    display_setCursor(214,222);
    display_print("RX\r\n");

    display_setTextSize(3);
    display_setCursor(15,13);
    display_print("CROBARM MENU\r\n");
    display_setCursor(15,19);
    display_print("_____________");

    display_setTextSize(2);
    display_setCursor(21,61);
    display_print("Manual Mode\r\n");
    display_setCursor(39,91);
    display_print("Settings\r\n");
    display_setCursor(21,121);
    display_print("Range Meter\r\n");
    display_setCursor(15,151);
    display_print("About Thesis\r\n");

    printExtraMenu();
}
```

```
void printMainMenu(void)
{
  unsigned int i;

  display_setTextColor(ILI9341_WHITE);
  display_setTextSize(3);
  display_setCursor(15,13);
  display_print("CROBARM MENU\r\n");
  display_setCursor(15,19);
  display_print("_____________");

  display_setTextSize(2);
  display_setCursor(21,61);
  display_print("Manual Mode\r\n");
  display_setCursor(39,91);
  display_print("Settings\r\n");
  display_setCursor(21,121);
  display_print("Range Meter\r\n");
  display_setCursor(15,151);
  display_print("About Thesis\r\n");

  printExtraMenu();
}

void deleteMainMenu(void)
{
  unsigned int i;

  display_setTextColor(ILI9341_NAVY);
  display_setTextSize(3);
  display_setCursor(15,13);
  display_print("CROBARM MENU\r\n");
  display_setCursor(15,19);
  display_print("_____________");

  display_setTextSize(2);
  display_setCursor(21,61);
  display_print("Manual Mode\r\n");
  display_setCursor(39,91);
  display_print("Settings\r\n");
  display_setCursor(21,121);
  display_print("Range Meter\r\n");
  display_setCursor(15,151);
  display_print("About Thesis\r\n");

  printExtraMenu();
  display_setTextColor(ILI9341_NAVY);
  display_setCursor(8,185);
  display_print("RESET: PRESS THE RIGHT JOYSTICK BUTTON\r\n");
  display_setCursor(8,197);
  display_print("RETRIEVE SIGNAL: PRESS ANY MENU BUTTON\r\n");

  display_setTextSize(2);
  display_setCursor(178,82);
```

```
    display_print("100%\r\n");
    display_setCursor(185,82);
    display_print("80%\r\n");
    display_setCursor(185,82);
    display_print("60%\r\n");
    display_setCursor(185,82);
    display_print("40%\r\n");
    display_setCursor(185,82);
    display_print("20%\r\n");
    display_setCursor(185,82);
    display_print("LOW\r\n");

    display_setCursor(178,139);
    display_print("100%\r\n");
    display_setCursor(185,139);
    display_print("80%\r\n");
    display_setCursor(185,139);
    display_print("60%\r\n");
    display_setCursor(185,139);
    display_print("40%\r\n");
    display_setCursor(185,139);
    display_print("20%\r\n");
    display_setCursor(185,139);
    display_print("LOW\r\n");

    flagControllerBattery100=1;
    flagControllerBattery80=1;
    flagControllerBattery60=1;
    flagControllerBattery40=1;
    flagControllerBattery20=1;
}

void printManualMode(void)
{
    display_setTextColor(ILI9341_WHITE);
    display_setTextSize(3);
    display_setCursor(24,13);
    display_print("MANUAL MODE\r\n");
    display_setCursor(24,19);
    display_print("___________");

    display_setTextSize(2);
    display_setCursor(15,68);
    display_print("Tank Operation:\r\n");
    display_setCursor(15,98);
    display_print("Tank Speed:\r\n");
    display_setCursor(15,128);
    display_print("Arm Operation:\r\n");
    display_setCursor(15,158);
    display_print("Arm Speed:\r\n");

    display_setTextSize(1);
    display_setCursor(15,189);
    display_setTextColor(ILI9341_GREENYELLOW);
```

```
    display_print("*ONLY ONE OPERATION IS TURNED ON AT\r\n");
    display_setCursor(15,199);
    display_print("A TIME, WHILE THE OTHER TURNS OFF\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setTextSize(2);
}

void deleteManualMode(void)
{
    display_setTextColor(ILI9341_NAVY);
    display_setTextSize(3);
    display_setCursor(24,13);
    display_print("MANUAL MODE\r\n");
    display_setCursor(24,19);
    display_print("_____________");

    display_setTextSize(2);
    display_setCursor(15,68);
    display_print("Tank Operation:\r\n");
    display_setCursor(15,98);
    display_print("Tank Speed:\r\n");
    display_setCursor(15,128);
    display_print("Arm Operation:\r\n");
    display_setCursor(15,158);
    display_print("Arm Speed:\r\n");

    display_setTextSize(1);
    display_setCursor(15,189);
    display_print("*ONLY ONE OPERATION IS TURNED ON AT\r\n");
    display_setCursor(15,199);
    display_print("A TIME, WHILE THE OTHER TURNS OFF\r\n");
    display_setTextSize(2);
}

void printTankOnArmOff(void)
{
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(200,68);
    display_print("OFF\r\n");
    display_setCursor(188,128);
    display_print("ON\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setCursor(200,68);
    display_print("ON\r\n");
    display_setCursor(188,128);
    display_print("OFF\r\n");

    output_high(PIN_C7);
    output_low(PIN_D4);
}

void deleteTankOnArmOff(void)
{
    display_setTextColor(ILI9341_NAVY);
```

```
   display_setCursor(200,68);
   display_print("ON\r\n");
   display_setCursor(188,128);
   display_print("OFF\r\n");
}

void printTankOffArmOn(void)
{
   display_setTextColor(ILI9341_NAVY);
   display_setCursor(200,68);
   display_print("ON\r\n");
   display_setCursor(188,128);
   display_print("OFF\r\n");
   display_setTextColor(ILI9341_WHITE);
   display_setCursor(200,68);
   display_print("OFF\r\n");
   display_setCursor(188,128);
   display_print("ON\r\n");

   output_low(PIN_C7);
   output_high(PIN_D4);
}

void deleteTankOffArmOn(void)
{
   display_setTextColor(ILI9341_NAVY);
   display_setCursor(200,68);
   display_print("OFF\r\n");
   display_setCursor(188,128);
   display_print("ON\r\n");
}

void printTankSpeed(void)
{
   if(!flagTankSpeed)
   {
      if((switchTankSpeed>=0)&&(switchTankSpeed<=2))
      {
         switchTankSpeed++;
      }
      else
      {
         switchTankSpeed=1;
      }
   }
   else
   {
      flagTankSpeed=0;
      switchTankSpeed=lastTankSpeed;
   }

   switch(switchTankSpeed)
   {
      case 1:
```

```
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(152, 98);
            display_print("FAST\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(152, 98);
            display_print("SLOW\r\n");
            break;
        case 2:
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(152, 98);
            display_print("SLOW\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(152, 98);
            display_print("NORMAL\r\n");
            break;
        case 3:
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(152, 98);
            display_print("NORMAL\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(152, 98);
            display_print("FAST\r\n");
            break;

    }
    lastTankSpeed=switchTankSpeed;
}

void deleteTankSpeed(void)
{
    display_setCursor(152, 98);
    display_setTextColor(ILI9341_NAVY);
    display_print("SLOW\r\n");
    display_setCursor(152, 98);
    display_print("NORMAL\r\n");
    display_setCursor(152, 98);
    display_print("FAST\r\n");
}

void printArmSpeed(void)
{
    if(!flagArmSpeed)
    {
        if(!switchArmSpeed)
        {
            switchArmSpeed++;
        }
        else
        {
            switchArmSpeed=0;
        }
    }
    else
    {
```

```
      flagArmSpeed=0;
      switchArmSpeed=lastArmSpeed;
   }


   switch(switchArmSpeed)
   {
      case 0:
         display_setTextColor(ILI9341_NAVY);
         display_setCursor(140, 158);
         display_print("NORMAL\r\n");
         display_setTextColor(ILI9341_WHITE);
         display_setCursor(140, 158);
         display_print("SLOW\r\n");
         break;
      case 1:
         display_setTextColor(ILI9341_NAVY);
         display_setCursor(140, 158);
         display_print("SLOW\r\n");
         display_setTextColor(ILI9341_WHITE);
         display_setCursor(140, 158);
         display_print("NORMAL\r\n");
         break;
   }
   lastArmSpeed=switchArmSpeed;
}

void deleteArmSpeed(void)
{
   display_setTextColor(ILI9341_NAVY);
   display_setCursor(140, 158);
   display_print("SLOW\r\n");
   display_setCursor(140, 158);
   display_print("NORMAL\r\n");
   display_setCursor(140, 158);
   display_print("FAST\r\n");
}

void printSettings(void)
{
   display_setTextColor(ILI9341_WHITE);
   display_setTextSize(3);
   display_setCursor(49,13);
   display_print("SETTINGS\r\n");
   display_setCursor(49,19);
   display_print("________");
   display_setTextSize(2);
   display_setCursor(15,68);
   display_print("Crash Safety:\r\n");
   display_setCursor(15,98);
   display_print("Headlights:\r\n");
   display_setCursor(15,128);
   display_print("System Sound:\r\n");
   display_setCursor(15,158);
```

```
    display_print("Brightness:\r\n");

    display_setTextSize(1);
    display_setCursor(15,189);
    display_setTextColor(ILI9341_GREENYELLOW);

    display_print("*CRASH SAFETY PROTECTS THE ROBOT FROM\n");
    display_setCursor(15,199);
    display_print("CRASHING WHEN THE DISTANCE IS 15cm<=\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setTextSize(2);
}

void deleteSettings(void)
{
    display_setTextColor(ILI9341_NAVY);
    display_setTextSize(3);
    display_setCursor(49,13);
    display_print("SETTINGS\r\n");
    display_setCursor(49,19);
    display_print("________");
    display_setTextSize(2);
    display_setCursor(15,68);
    display_print("Crash Safety:\r\n");
    display_setCursor(15,98);
    display_print("Headlights:\r\n");
    display_setCursor(15,128);
    display_print("System Sound:\r\n");
    display_setCursor(15,158);
    display_print("Brightness:\r\n");
    display_setTextSize(1);
    display_setCursor(15,189);
    display_print("*CRASH SAFETY PROTECTS THE ROBOT FROM\n");
    display_setCursor(15,199);
    display_print("CRASHING WHEN THE DISTANCE IS 15cm<=\r\n");
    display_setTextSize(2);

    display_setCursor(178,68);
    display_print("OFF\r\n");
    display_setCursor(178,68);
    display_print("ON\r\n");

    display_setCursor(154,98);
    display_print("OFF\r\n");
    display_setCursor(154,98);
    display_print("ON\r\n");

    display_setCursor(178,128);
    display_print("OFF\r\n");
    display_setCursor(178,128);
    display_print("ON\r\n");

    display_setCursor(154,158);
    display_print("20%\r\n");
```

```
    display_setCursor(154,158);
    display_print("40%\r\n");
    display_setCursor(154,158);
    display_print("60%\r\n");
    display_setCursor(154,158);
    display_print("80%\r\n");

    display_setCursor(154,158);
    display_print("100%\r\n");
}

void printCrashSafety(void)
{
    if(!flagCrashSafety)
    {
        (!lastCrashSafety)?(switchCrashSafety=1):(switchCrashSafety=0);
    }
    else
    {
        flagCrashSafety=0;
        switchCrashSafety=lastCrashSafety;
    }

    switch(switchCrashSafety)
    {
        case 0:
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(178,68);
            display_print("OFF\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(178,68);
            display_print("ON\r\n");
            break;
        case 1:
            display_setCursor(178,68);
            display_setTextColor(ILI9341_NAVY);
            display_print("ON\r\n");
            display_setCursor(178,68);
            display_setTextColor(ILI9341_WHITE);
            display_print("OFF\r\n");
            break;
    }
    lastCrashSafety=switchCrashSafety;
    generalSettings=1;
}

void printSystemSound(void)
{
    if(!flagSystemSound)
    {
        (!switchSystemSound)?(switchSystemSound=1):(switchSystemSound=0);
    }
    else
    {
```

```
    flagSystemSound=0;
    switchSystemSound=lastSystemSound;
  }

  switch(switchSystemSound)
  {
    case 0:
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(178,128);
      display_print("OFF\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(178,128);
      display_print("ON\r\n");
      break;
    case 1:
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(178,128);
      display_print("ON\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(178,128);
      display_print("OFF\r\n");
      break;
  }
  lastSystemSound=switchSystemSound;
}

void printHeadlights(void)
{
  if(!flagHeadlights)
  {
    (!lastHeadlights)?(switchHeadlights=1):(switchHeadlights=0);
  }
  else
  {
    flagHeadlights=0;
    switchHeadlights=lastHeadlights;
  }

  switch(switchHeadlights)
  {
    case 0:
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(154,98);
      display_print("ON\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(154,98);
      display_print("OFF\r\n");
      break;
    case 1:
      display_setTextColor(ILI9341_NAVY);
      display_setCursor(154,98);
      display_print("OFF\r\n");
      display_setTextColor(ILI9341_WHITE);
      display_setCursor(154,98);
```

```
            display_print("ON\r\n");
            break;
    }
    lastHeadlights=switchHeadlights;
    generalSettings=1;
}



void printBrightnessDisplay(void)
{
    if(!flagBrightnessDisplay)
    {
        if((switchBrightnessDisplay>0)&&(switchBrightnessDisplay<=4))
        {
            switchBrightnessDisplay++;
        }
        else
        {
            switchBrightnessDisplay=1;
        }
    }
    else
    {
        flagBrightnessDisplay=0;
        switchBrightnessDisplay=lastBrightnessDisplay;
    }

    switch(switchBrightnessDisplay)
    {
        case 1:
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(154,158);
            display_print("100%\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(154,158);
            display_print("20%\r\n");
            break;
        case 2:
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(154,158);
            display_print("20%\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(154,158);
            display_print("40%\r\n");
            break;
        case 3:
            display_setTextColor(ILI9341_NAVY);
            display_setCursor(154,158);
            display_print("40%\r\n");
            display_setTextColor(ILI9341_WHITE);
            display_setCursor(154,158);
            display_print("60%\r\n");
            break;
```

```
        case 4:
          display_setTextColor(ILI9341_NAVY);
          display_setCursor(154,158);
          display_print("60%\r\n");
          display_setTextColor(ILI9341_WHITE);
          display_setCursor(154,158);
          display_print("80%\r\n");
          break;
        case 5:
          display_setTextColor(ILI9341_NAVY);
          display_setCursor(154,158);
          display_print("80%\r\n");
          display_setTextColor(ILI9341_WHITE);
          display_setCursor(154,158);
          display_print("100%\r\n");
          break;
    }
    lastBrightnessDisplay=switchBrightnessDisplay;
}

void printRangeMeter(void)
{
    display_setTextColor(ILI9341_WHITE);
    display_setTextSize(3);
    display_setCursor(24,13);
    display_print("RANGE METER\r\n");
    display_setCursor(24,19);
    display_print("____________");

    display_setTextSize(2);
    display_setCursor(15,68);
    display_print("Status:\r\n");
    display_setCursor(15,98);
    display_print("Distance:   cm\r\n");
    display_setCursor(15,128);
    display_setTextColor(ILI9341_DARKYELLOW);
    display_print("Obstacle Glow:\r\n");

    display_setTextSize(1);
    display_setTextColor(ILI9341_GREENYELLOW);
    display_setCursor(15,159);
    display_print("*YELLOW LED TURNS ON IF AN OBSTACLE\n");
    display_setCursor(15,169);
    display_print("IS BEING DETECTED IN FRONT OF ROBOT\r\n");
    display_setCursor(15,179);
    display_print("*THE SENSOR MEASURES THE DISTANCE\r\n");
    display_setCursor(15,189);
    display_print("FROM 15cm TO 70cm AND IT IS BEING\n");
    display_setCursor(15,199);
    display_print("UPDATED ON SCREEN EVERY ONE SECOND\r\n");
    display_setTextColor(ILI9341_WHITE);
    display_setTextSize(2);
}
```

```
void deleteRangeMeter(void)
{
   display_setTextColor(ILI9341_NAVY);
   display_setTextSize(3);
   display_setCursor(24,13);
   display_print("RANGE METER\r\n");
   display_setCursor(24,19);
   display_print("____________");

   display_setTextSize(2);
   display_setCursor(15,68);
   display_print("Status:\r\n");
   display_setCursor(15,98);
   display_print("Distance:   cm\r\n");
   display_setCursor(15,128);
   display_print("Obstacle Glow:\r\n");
   display_setCursor(192,98);
   display_print("<\r\n");
   display_setCursor(192,98);
   display_print(">\r\n");
   display_setCursor(190,128);
   display_print("ON\r\n");
   display_setCursor(190,128);
   display_print("OFF\r\n");

   display_setCursor(105,68);
   display_print("DANGER\r\n");
   display_setCursor(105,68);
   display_print("CAUTION\r\n");
   display_setCursor(105,68);
   display_print("SAFE\r\n");

   display_setTextColor(ILI9341_NAVY);
   display_setCursor(129,98);
   printf(display_print,"%d\r\n",lastDistance);

   display_setTextSize(1);
   display_setCursor(15,159);
   display_print("*YELLOW LED TURNS ON IF AN OBSTACLE\n");
   display_setCursor(15,169);
   display_print("IS BEING DETECTED IN FRONT OF ROBOT\r\n");
   display_setCursor(15,179);
   display_print("*THE SENSOR MEASURES THE DISTANCE\r\n");
   display_setCursor(15,189);
   display_print("FROM 15cm TO 70cm AND IT IS BEING\n");
   display_setCursor(15,199);
   display_print("UPDATED ON SCREEN EVERY ONE SECOND\r\n");
   display_setTextSize(2);
}

void printZoneStatus(void)
{
   if((0x933==codedCommandRead)&&(flagDanger))
   {
```

```
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(105,68);
    display_print("CAUTION\r\n");
    display_setCursor(105,68);
    display_print("SAFE\r\n");
    display_setCursor(192,98);
    display_print(">\r\n");

    display_setTextColor(ILI9341_WHITE);
    display_setCursor(105,68);
    display_print("DANGER\r\n");
    display_setCursor(192,98);
    display_print("<\r\n");

    flagDanger=0;
    flagCaution=1;
    flagSafe=1;
}
else if((0xA96>codedCommandRead)&&(codedCommandRead>=0x935)&&(flagCaution))
{
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(105,68);
    display_print("SAFE\r\n");
    display_setCursor(105,68);
    display_print("DANGER\r\n");
    display_setCursor(192,98);
    display_print("<\r\n");
    display_setCursor(192,98);
    display_print(">\r\n");

    display_setTextColor(ILI9341_WHITE);
    display_setCursor(105,68);
    display_print("CAUTION\r\n");

    flagDanger=1;
    flagCaution=0;
    flagSafe=1;
}
else if((0xA96==codedCommandRead)&&(flagSafe))
{
    display_setTextColor(ILI9341_NAVY);
    display_setCursor(105,68);
    display_print("CAUTION\r\n");
    display_setCursor(105,68);
    display_print("DANGER\r\n");
    display_setCursor(192,98);
    display_print("<\r\n");

    display_setTextColor(ILI9341_WHITE);
    display_setCursor(105,68);
    display_print("SAFE\r\n");
    display_setCursor(192,98);
    display_print(">\r\n");
```

```
        flagDanger=1;
        flagCaution=1;
        flagSafe=0;
    }
}

void printAboutThesis(void)
{
    display_setTextColor(ILI9341_WHITE);
    display_setTextSize(3);
    display_setCursor(15,13);
    display_print("ABOUT THESIS\r\n");
    display_setCursor(15,19);
    display_print("_____________");

    display_setTextSize(1);
    display_setCursor(14,54);
    display_print("THE OFFICIAL TITLE OF THIS THESIS IS\r\n");
    display_setCursor(7,66);
    display_print("<RESEARCH CONSTRUCTION AND APPLICATION\r\n");
    display_setCursor(12,78);
    display_print("OF INTELLIGENT CRAWLER ROBOT WITH ARM\r\n");
    display_setCursor(8,90);
    display_print("CONTROLLED REMOTELY THROW RF> WHICH IS\r\n");
    display_setCursor(14,102);
    display_print("ABBREVIATED AS CROBARM PROJECT. THIS\r\n");
    display_setCursor(16,114);
    display_print("WORK WAS PREPARED PERSONALLY BY THE\r\n");
    display_setCursor(9,126);
    display_print("UNDERGRADUATE STUDENT STEFANOS LIAKAS\r\n");
    display_setCursor(2,138);
    display_print("AT THE INTERNATIONAL HELLENIC UNIVERSITY\r\n");
    display_setCursor(8,150);
    display_print(", SCHOOL OF ENGINEERING, DEPARTMENT OF\r\n");
    display_setCursor(5,162);
    display_print("INFORMATION AND ELECTONIC ENGINEERING.\r\n");
    display_setCursor(7,174);
    display_print("THE MAIN USE OF THIS ROBOTIC SYSTEM IS\r\n");
    display_setCursor(1,186);
    display_print("FOR RESEARCH, TASKS, AMBIENT OBSERVATION\r\n");
    display_setCursor(50,198);
    display_print("AND INFORMATION TRANSFER.\n");
    display_setTextSize(2);
}

void deleteAboutThesis(void)
{
    display_setTextColor(ILI9341_NAVY);
    display_setTextSize(3);
    display_setCursor(15,13);
    display_print("ABOUT THESIS\r\n");
    display_setCursor(15,19);
    display_print("_____________");
```

```
    display_setTextSize(1);
    display_setCursor(14,54);
    display_print("THE OFFICIAL TITLE OF THIS THESIS IS\r\n");
    display_setCursor(7,66);
    display_print("<RESEARCH CONSTRUCTION AND APPLICATION\r\n");
    display_setCursor(12,78);
    display_print("OF INTELLIGENT CRAWLER ROBOT WITH ARM\r\n");
    display_setCursor(8,90);
    display_print("CONTROLLED REMOTELY THROW RF> WHICH IS\r\n");
    display_setCursor(14,102);
    display_print("ABBREVIATED AS CROBARM PROJECT. THIS\r\n");
    display_setCursor(16,114);
    display_print("WORK WAS PREPARED PERSONALLY BY THE\r\n");
    display_setCursor(9,126);
    display_print("UNDERGRADUATE STUDENT STEFANOS LIAKAS\r\n");
    display_setCursor(2,138);
    display_print("AT THE INTERNATIONAL HELLENIC UNIVERSITY\r\n");
    display_setCursor(8,150);
    display_print(", SCHOOL OF ENGINEERING, DEPARTMENT OF\r\n");
    display_setCursor(5,162);
    display_print("INFORMATION AND ELECTONIC ENGINEERING.\r\n");
    display_setCursor(7,174);
    display_print("THE MAIN USE OF THIS ROBOTIC SYSTEM IS\r\n");
    display_setCursor(1,186);
    display_print("FOR RESEARCH, TASKS, AMBIENT OBSERVATION\r\n");
    display_setCursor(50,198);
    display_print("AND INFORMATION TRANSFER.\n");
    display_setTextSize(2);
}
```