ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός και υλοποίηση ενσωματωμένου συστήματος ελέγχου τροχοφόρου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του/της

Νικολή Νικόλαου

Επιβλέπων: Δρ. Δασυγένης Μηνάς

Αναπληρωτής Καθηγητής ΠΔΜ

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

Κοζάνη/Ιούνιος/2025

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA

FUCULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL &
COMPUTER ENGINEERING

Design and implementation of an integrated system for controlling a wheeled vehicle

THESIS

Nikoli Nikolaou

SUPERVISOR: Dr. Minas Dasygenis

Assistant Professor

Robotics, Embedded and Integrated Systems Laboratory

Kozani/June/2025

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Σχεδιασμός και υλοποίηση ενσωματωμένου συστήματος ελέγχου τροχοφόρου" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Δασυγένη Μηνά αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: Νικολής Νικόλαος

Περίληψη

Η ραγδαία ανάπτυξη του τομέα της ρομποτικής την τελευταία δεκαετία έχει οδηγήσει στην δημιουργία ρομποτικών συστημάτων ποικίλλων δυνατοτήτων. Ρομπότ με την ικανότητα αλληλεπίδρασης με το περιβάλλον πρωταγωνιστούν πλέον σε εργασίες των όποιον οι συνθήκες φέρουν κινδύνους για τον άνθρωπο.

Στην παρούσα διπλωματική αναλύεται η διαδικασία σχεδίασης και υλοποίησης ενός τέτοιου ρομποτικού συστήματος. Η κεντρική μονάδα των δυο διακριτών μερών του συστήματος είναι ο μικροελεγκτής ESP32 S3. Ως βάση του οχήματος επιλέχθηκε το τροχοφόρο σασί της εταιρείας Dagu. Το ρομπότ εξοπλίστηκε με έναν ρομποτικό βραχίονα για να είναι εφικτή η αλληλεπίδραση με το περιβάλλον. Για τον απομακρυσμένο έλεγχο αναπτύχθηκε ένα σύστημα επικοινωνίας κάνοντας χρήση του πρωτοκόλλου ESP-NOW και ένα PCB το οποίο παρέχει στον χρήστη δυνατότητα ελέγχου ρομπότ και ένδειξη σημαντικών πληροφοριών μέσω μιας οθόνης.

Η υλοποίηση συνδυάζει ενσωματωμένα συστήματα, μηχανολογικό και ηλεκτρολογικό σχεδιασμό, τεχνολογίες ρομποτικού βραχίονα και ασύρματης επικοινωνίας, αποδεικνύοντας την δυνατότητα κατασκευής ενός εύκολα προσαρμόσιμου οχήματος με την ικανότητα αλληλεπίδρασης με το περιβάλλον. Μελλοντικές έρευνες μπορούν να στοχεύσουν στην χρήση καλύτερου εξοπλισμού για την αύξηση της εμβέλειας χρήσης και ενδυνάμωση του ρομποτικού βραχίονα. Συνάμα μπορούν να ενσωματωθούν τεχνικές αντίστροφης κινηματικής με στόχο την βελτίωση χρήσης του βραχίονα.

Λέξεις Κλειδιά: ενσωματωμένο, ESP32, τροχοφόρο όχημα, ρομποτικός βραχίονας, PCB

Abstract

The rapid development of the robotics field over the past decade has led to the creation of robotic systems with a wide range of capabilities. Robots capable of interacting with their environment now play a key role in tasks where conditions pose potential risks to humans.

This thesis analyzes the design and implementation process of such a robotic system. The central unit of the two distinct parts of the system is the ESP32 S3 microcontroller. The wheeled chassis from the company Dagu was chosen as the base of the vehicle. The robot was equipped with a robotic arm to enable interaction with the environment. For remote control, a communication system was developed using the ESP-NOW protocol, along with a custom PCB that allows the user to control the robot and view important information through a display.

The implementation combines embedded systems, mechanical and electrical design, robotic arm technologies, and wireless communication, demonstrating the feasibility of constructing a highly adaptable vehicle capable of environmental interaction. Future research could focus on using improved hardware to extend operational range and strengthen the robotic arm. Additionally, inverse kinematics techniques could be integrated to enhance arm control and usability.

Keywords: embedded, ESP32, wheeled vehicle, robotic arm, PCB

Ευχαριστίες

Η παρούσα διπλωματική εκπονήθηκε στο πανεπιστήμιο Δυτικής Μακεδονίας και συγκεκριμένα στο τμήμα των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών. Θα ήθελα να εκφράζω την ευγνωμοσύνη μου προς όλους όσους με στήριξαν και με βοήθησαν να φέρω εις πέρας αυτή την εργασία.

Αρχικά θα ήθελα να ευχαριστήσω τον Δρ. Μηνά Δασυγένη ο οποίος με την αποφασιστικότητα και την καθοδήγηση του με βοήθησε να ολοκληρώσω την παρούσα διπλωματική εργασία.

Θα ήθελα επίσης να πω ένα μεγάλο ευχαριστώ στους φοιτητές και στους υποψήφιους διδάκτορες του εργαστηρίου ρομποτικής και ενσωματωμένων συστημάτων και συγκεκριμένα στον Αντώνη ο οποίος με καθοδήγησε με υπομονή και μου παρείχε μεγάλη βοήθεια στην εκτύπωση τρισδιάστατων μοντέλων.

Τέλος, να εκφράσω την ευγνωμοσύνη μου στους φίλους μου, στον συγκάτοικο μου Θοδωρή και στην οικογένεια μου. Η συνδρομή τους αποτέλεσε καθοριστικό παράγοντα στην εκπόνηση της παρούσας διπλωματικής εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1 – Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Η ταχεία εξέλιξη του κλάδου της ρομποτικής και της μηχατρονικής έχει οδηγήσει στην ανάπτυξη ρομποτικών συστημάτων, των οποίων η χρήση καλύπτει πληθώρα εφαρμογών. Από τομείς όπως την γεωργία, με την χρήση ρομπότ για γεωργία ακριβείας, έως τομείς της ιατρικής, με την χρήση ρομποτικών βραχιόνων για μικροχειρουργική, τα ρομπότ έχουν γίνει αναπόσπαστο κομμάτι της κοινωνίας μας. Στις μέρες μας, η χρήση των τροχοφόρων οχημάτων, αυτόματων ή ημιαυτόματων, καλύπτει ένα μεγάλο φάσμα εφαρμογών όπως την επιστημονική έρευνα, τον εντοπισμό και την διάσωση ατόμων σε δύσβατες και επικίνδυνες περιοχές, αλλά και στρατιωτικές εφαρμογές, όπως προσπέλαση εδάφους για την μεταφορά πολεμοφοδίων. Η προσθήκη ενός ρομποτικού βραχίονα επιτρέπει στα ρομπότ να αλληλοεπιδρούν με το περιβάλλον αυξάνοντας έτσι την αποτελεσματικότητα τους αλλά και τις δυνατότητες τους.

Η παρούσα διπλωματική εργασία αφορά την ανάλυση, σχεδίαση και ανάπτυξη ενός ενσωματωμένου συστήματος για τον έλεγχο ενός τροχοφόρου οχήματος εξοπλισμένο με έναν ρομποτικό βραχίονα. Στα πλαίσια του συστήματος κατασκευάστηκε ένα τηλεχειριστήριο για την ασύρματη επικοινωνία μεταξύ του χρήστη και του οχήματος, κάνοντας χρήση του μικροελεγκτή ESP32 S3, ενώ παράλληλα αναπτύχθηκε το κατάλληλο λογισμικό για τον έλεγχο του ρομποτικού βραχίονα. Επιπρόσθετα, έγινε χρήση αισθητήρων επιτάχυνσης, γυροσκοπίου, τάσης και ρεύματος, με στόχο την ανάπτυξη έξυπνου λογισμικού φιλικού προς τον χρήστη.

1.2 Παρόμοιο έργο και οι εφαρμογές του

Ο όγκος έργων με στόχο την ανάπτυξη τροχοφόρων οχημάτων με την ικανότητα αλληλοεπίδρασης με το περιβάλλον είναι τεράστιος. Παρόλα αυτά, μπορούμε να ξεχωρίσουμε 3 διακριτές κατηγορίες στις οποίες εντάσσονται τα περισσότερα από αυτά τα έργα. Η πρώτη κατηγορία είναι αυτή των ρομπότ ως χόμπι ή εκπαιδευτικής χρήσης. Αυτή η κατηγορία φέρνει σε επαφή τον καθημερινό άνθρωπο με την δύναμη των ρομποτικών συστημάτων μέσα από την κατασκευή προσιτών ρομπότ που προάγουν τον πειραματισμό και την δημιουργική σκέψη. Ένα από αυτά είναι το TurtleBot3[1], το οποίο κάνει χρήση ραντάρ LIDAR για αυτόνομη μετακίνηση, ενώ παράλληλα προσφέρεται η δυνατότητα προσθήκης του ρομποτικού βραχίονα OpenMANIPULATOR-X[2] από την εταιρεία ROBOTIS. Η δεύτερη κατηγορία είναι αυτή των βιομηχανικών ρομπότ. Χρησιμοποιούνται συνήθως σε βιομηχανικές εγκαταστάσεις με στόχο την αυτοματοποίηση διαδικασιών, όπως τη μεταφορά επικίνδυνων υλικών ή τη συλλογή δεδομένων. Ένα από αυτά τα ρομπότ είναι το Inspector V8[3] της εταιρείας Taurob με στόχο την χρήση σε πεδία πετρελαίου. Αυτό το ρομπότ είναι εξοπλισμένο με αισθητήρες τοξικών αερίων, GPS, LIDAR, κάμερα υψηλής ανάλυσης, έως 4K και έναν ρομποτικό βραχίονα για δειγματοληψία από το περιβάλλον. Μέσω του ιδιαίτερου σχεδιασμού των ροδών του το συγκεκριμένο ρομπότ έχει δυνατότητες διάβασης ανισόπεδου μονοπατιού ενώ ταυτόχρονα με την χρήση του LIDAR είναι ικανό να αποφύγει εμπόδια που μπλοκάρουν το πέρασμα του. Η τελευταία κατηγορία αποτελείται από ρομπότ που χρησιμοποιούνται για επιστημονική έρευνα. Είναι κατασκευασμένα για να αντέχουν και στις πιο απαιτητικές περιβαλλοντικές συνθήκες με εφαρμογές όπως την συλλογή δειγμάτων και την λήψη μετρήσεων. Παράδειγμα αυτής της κατηγορίας είναι το ρομπότ της εταιρείας ClearPath με όνομα Husky A300[4]. Το συγκεκριμένο ρομποτικό σύστημα μπορεί να εξοπλιστεί με 3 διαφορετικά πακέτα για την εξυπηρέτηση διαφόρων σκοπών. Τα πακέτα αυτά αποτελούνται από το πακέτο των αισθητήρων για την αυτοματοποίηση συλλογής δειγμάτων, LIDAR και κάμερες, το πακέτο ρομποτικού βραχίονα Franka Research 3[5] της ομώνυμης εταιρείας Franka και το πακέτο όρασης το οποίο

προσθέτει ένα σύστημα καμερών πάνω στο ρομπότ με στόχο την αυτόματη πλοήγηση. Η κατηγορία αυτή χαρακτηρίζεται από το υψηλό κόστος εξαιτίας της χρήσης ακριβού εξοπλισμού που παρέχει όμως μεγάλη εξειδίκευση ως προς τις δυνατότητες των ρομπότ. Στην διπλωματική αυτή κατασκευάσαμε ένα τροχοφόρο το οποίο θα συνδυάζει τα πλεονεκτήματα κάθε κατηγορίας τα οποία είναι, προσιτό κόστος χωρίς να θυσιάσουμε την ποιότητα κατασκευής του οχήματος, ευελιξία προσθήκης αισθητήρων, δυνατότητα προσπέλασης ανισόπεδου εδάφους με εμπόδια και μεταφορά υλικών και επικίνδυνων ουσιών με την χρήση του ρομποτικού βραχίονα.

Από κάτω παρατίθεται ένας πίνακας με τα κύρια κοινά χαρακτηριστικά αυτών των έργων σε σύγκριση με το τροχοφόρο όχημα της διπλωματικής εργασίας.

Πίνακας 1: Πίνακας σύγκρισης παρόμοιων κατασκευών

Χαρακτηριστικά	TurtleBot3 + OpenMANIPULATOR-X	Operator	Husky A300 + Franka Research 3	Όχημα Διπλωματικής
Τύπος ελέγχου	Αυτόματος και ημιαυτόματος	Αυτόματος και ημιαυτόματος	Αυτόματος και ημιαυτόματος	Χειροκίνητος
Τύπος εδάφους για προσπέλαση	Εσωτερικοί χώροι με λείο έδαφος	Τραχύ έδαφος και ανισόπεδες επιφάνειες	Τραχύ έδαφος και ανισόπεδες επιφάνειες	Τραχύ έδαφος και ανισόπεδες επιφάνειες
Μήκος βραχίονα	38 εκατοστά	164 εκατοστά	85.5 εκατοστά	51.4 εκατοστά
Μέγιστο βάρος ανύψωσης	0.5 κιλά	-	3 κιλά	0.5 κιλά
Μέγιστο βάρος μεταφοράς πάνω στο όχημα	30 κιλά	-	100 κιλά	5 κιλά
Βαθμοί ελευθερίας βραχίονα	5 (4 + 1 της δαγκάνας)	4	7	6
Βάρος οχήματος	1.8 κιλά + 0.7 κιλά	85 κιλά	80 κιλά + 17.8 κιλά	4.5 κιλά
Εκτιμώμενο Κόστος	3.000€	20.000-30.000€	70.000€	600€

Από τα παραπάνω στοιχεία παρατηρούμε ότι η κατασκευή της διπλωματικής αποτελεί την πιο προσιτή επιλογή χωρίς να θυσιάσουμε την ποιότητα κατασκευής. Ταυτόχρονα, ο ρομποτικός βραχίονας παρέχει τον δεύτερο μεγαλύτερο αριθμό βαθμών ελευθερίας και μεταφοράς φορτίου. Καθένα από τα παραπάνω οχήματα καλύπτει διαφορετικές απαιτήσεις με το όχημα που αναπτύξαμε να αποτελεί χρυσή τομή μεταξύ όλων από άποψης δυνατοτήτων και προσβασιμότητας.

1.3 Στόχος της παρούσας διπλωματικής εργασίας

Στόχος της παρούσας διπλωματικής εργασίας θα είναι η σχεδίαση ενός ολοκληρωμένου ενσωματωμένου συστήματος για τον έλεγχο του τροχοφόρου οχήματος Wild Thumper 6WD (6 wheel drive) της εταιρείας Dagu. Το όχημα αυτό είναι εξοπλισμένο με 6 DC κινητήρες, 3 σε κάθε πλευρά του οχήματος. Για τον έλεγχο αυτών των κινητήρων θα γίνει χρήση του ενσωματωμένου συστήματος T'REX robot controller της Dagu το οποίο παρέχει δύο κανάλια εξόδου για τον έλεγχο της τάσης των κινητήρων κάθε πλευράς του οχήματος. Το ενσωματωμένο αυτό θα ελέγχεται από έναν μικροελεγκτή ESP32 S3 της εταιρείας Espressif και συγκεκριμένα γίνεται χρήση της ολοκληρωμένης αναπτυξιακής πλακέτας ESP32 S3 DevKit M1 . Μέσω του πρωτοκόλλου επικοινωνίας I2C (Inter-Integrated Circuit) η πλακέτα στέλνει εντολές στο ενσωματωμένο σύστημα για τον έλεγχο της ταχύτητας των τροχών και για την λήψη πληροφοριών που παρέχει αυτό το σύστημα όπως την τάση της μπαταρίας, το ρεύμα των κινητήρων, την επιτάχυνση του οχήματος και την ύπαρξη σφαλμάτων. Κάνοντας χρήση του ίδιου πρωτοκόλλου η πλακέτα επικοινωνεί με έναν αισθητήρα IMU (Inertial Measurement Unit) [9] ο οποίος θα καταγράφει την περιστροφή του οχήματος ως προς τον κάθετο άξονα ενώ με τον ίδιο τρόπο η πλακέτα στέλνει εντολές σε μικροελεγκτή για σερβοκινητήρες ελέγχοντας έτσι τον ρομποτικό βραχίονα. Το όχημα θα εξοπλιστεί επίσης με ένα δευτερεύον κύκλωμα για την ασφαλή αποσύνδεση της μπαταρίας από το κύριο κύκλωμα με την χρήση ενός E-Stop (emergency stop) κουμπιού. Το ESP32 S3 DevKit M1[7] αποτελεί τον εγκέφαλο του οχήματος που κάνει τους απαραίτητους υπολογισμούς και έπειτα στέλνει τις κατάλληλες πληροφορίες στην κάθε πλακέτα. Η επικοινωνία του χρήστη με το όχημα θα γίνει μέσα από ένα διαφορετικό ενσωματωμένο σύστημα το οποίο θα έχει την μορφή τηλεχειριστηρίου. Μέσω αυτής ο χρήστης θα έχει την δυνατότητα να ελέγχει το όχημα και να χειρίζεται την ρομποτική δαγκάνα. Το ενσωματωμένο αυτό είναι εξοπλισμένο με ένα δεύτερο ESP32 S3 και συγκεκριμένα το ολοκληρωμένο ενσωματωμένο ESP32 S3 DevKit C1 [10] το οποίο μέσα από κουμπιά, για τον έλεγχο του βραχίονα, και ενός joystick (μοχλός αντίχειρα), για τον έλεγχο του οχήματος, λαμβάνει τις εντολές του χρήστη και έπειτα από επεξεργασία τις στέλνει στον εγκέφαλο του οχήματος. Η επικοινωνία γίνεται μέσω του πρωτοκόλλου ESP-NOW [11] της Espressif καθώς η χρήση του είναι απλή και γρήγορη. Το ενσωματωμένο τηλεχειριστήριο εμπεριέχει μια οθόνη η οποία θα εμφανίζει στον χρήστη διάφορες πληροφορίες σχετικά με το όχημα την κατάσταση της σύνδεσης την μπαταρία του οχήματος όπως επίσης και ένα μενού από το οποίο ο χρήστης θα μπορεί να αλλάξει τον τρόπο λειτουργίας και την ταχύτητα του οχήματος. Το σύστημα αυτό θα μπει μέσα σε ένα κουτί το οποίο δημιουργήθηκε με 3D εκτύπωση και θα αναγράφει πάνω πληροφορίες όπως την μπαταρία του τηλεχειριστηρίου και την αντιστοίχιση των κουμπιών με τα διάφορα μέρη του ρομποτικού βραχίονα. Ο προγραμματισμός του εγκεφάλου του οχήματος και του τηλεχειριστηρίου θα γίνει με την χρήση του προγράμματος Arduino IDE.

1.4 Η δομή της εργασίας

Στα επόμενα κεφάλαια γίνεται αναλυτική επεξήγηση του τρόπου λειτουργίας κάθε κομματιού της εργασίας.

Στο κεφάλαιο 2, με τίτλο «Θεωρητικό τεχνολογικό υπόβαθρο», θα καλύψουμε όλες τις τεχνολογίες και το θεωρητικό υπόβαθρο που χρησιμοποιήθηκαν για την δημιουργία των συστημάτων, την επικοινωνία και την λειτουργία του οχήματος και του τηλεχειριστηρίου. Θα αναλύσουμε το λογισμικό (software) αλλά και το υλικό (hardware) που αξιοποιήσαμε κατά την διάρκεια αυτής της εργασίας.

Στο κεφάλαιο 3, με τίτλο «Προσέγγιση και ανάπτυξη των συστημάτων», θα δούμε με περισσότερη λεπτομέρεια πως τα διάφορα κομμάτια της διπλωματικής εργασίας συνδέονται μεταξύ τους και πως επικοινωνούν. Αυτό περιλαμβάνει κυκλώματα και συνδεσμολογίες

(schematics) των ενσωματωμένων συστημάτων αλλά και πως δουλεύει η ασύρματη επικοινωνία μεταξύ του χρήστη και του οχήματος. Τέλος, θα δούμε πως το hardware συνεργάζεται με το software ώστε να έχουμε ένα ολοκληρωμένο αποτέλεσμα.

Στο κεφάλαιο 4, με τίτλο «Επεξήγηση κώδικα», θα εμβαθύνουμε στην λειτουργία του κώδικα του εγκεφάλου του οχήματος και του ασύρματου τηλεχειριστηρίου και το πως στήνεται η επικοινωνία μεταξύ τους μέσω κώδικα.

Στο κεφάλαιο 5, με τίτλο «Συμπεράσματα και μελλοντικές βελτιώσεις», αναφέρουμε όλες τις δυνατότητες που μας παρέχει η τελική κατασκευή και αναλύουμε πιθανές μελλοντικές βελτιώσεις που μπορούν να γίνουν πάνω της.

Κεφάλαιο 2 - Θεωρητικό και τεχνολογικό υπόβαθρο

Στο παρόν κεφάλαιο παρουσιάζουμε πληροφορίες σχετικά με την πλακέτα ανάπτυξης του λογισμικού ESP32 S3, τα προγράμματα για την ανάπτυξη του λογισμικού και του υλικού, το ενσωματωμένο σύστημα για τον έλεγχο του οχήματος, την δομή του ίδιου του οχήματος, την δομή του ρομποτικού βραχίονα και την μονάδα για τον έλεγχο του, τις μονάδες για την ανάγνωση εισόδων από τον χρήστη και την πλακέτα για τον έλεγχο περιστροφής.

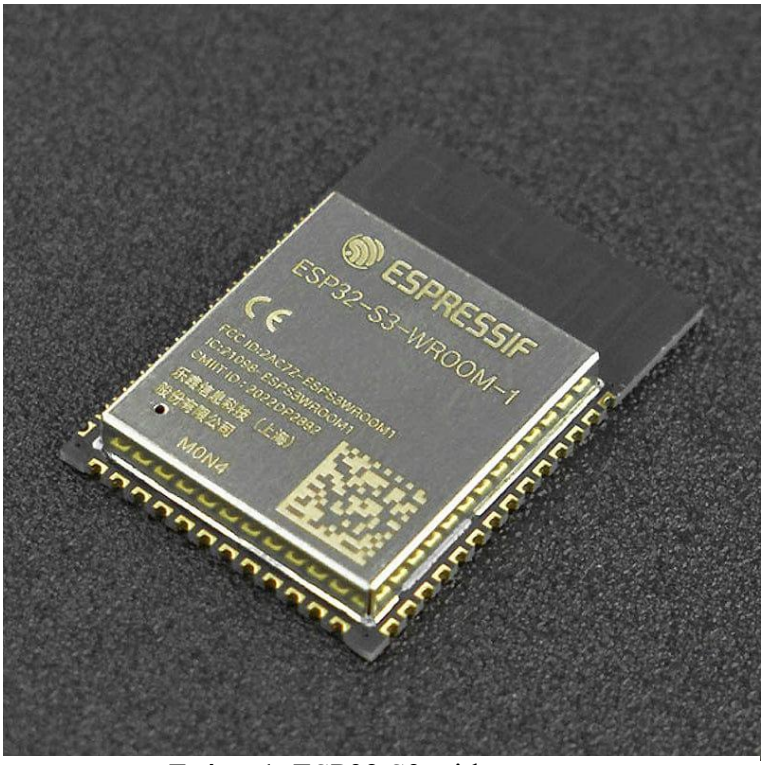
2.1 Οι μικροελεγκτές των συστημάτων που αναπτύχθηκαν

Αναλύοντας τις απαιτήσεις των συστημάτων καταλήγουμε ότι οι μικροελεγκτές που πρέπει να επιλέξουμε πρέπει να έχουν πολλές διεπαφές εισόδου εξόδου για την σύνδεση περιφερειακών, να υποστηρίζουν πολλά σειριακά πρωτόκολλα επικοινωνίας, να επιτρέπουν την ασύρματη επικοινωνία και να έχουν επαρκής υπολογιστική ισχύ. Με βάση αυτά τα κριτήρια οι επιλογές είναι πολυάριθμες αλλά θα αναφέρουμε 4. Το Arduino nano[6] της εταιρίας Arduino, το Raspberry Pi 4[7] της ομώνυμης εταιρείας Raspberry Pi, το ESP32 S3[8] της εταιρείας Espressif και το Blue Pill[9] της εταιρείας ST-Electronics. Όλες οι επιλογές παρέχουν διεπαφές για σειριακά πρωτόκολλα επικοινωνίας, παρόλα αυτά οι διεπαφές εισόδου εξόδου στις περισσότερες μονάδες δεν είναι επαρκείς. Επιπρόσθετα οι επιλογές Arduino nano και Blue Pill δεν παρέχουν τρόπο ασύρματης επικοινωνίας χωρίς την χρήση εξωτερικής μονάδας αυξάνοντας έτσι το κόστος και την πολυπλοκότητα κατασκευής. Το ESP32 S3 μας προσφέρει τις περισσότερες διεπαφές εισόδου εξόδου, συγκεκριμένα 45, ενώ το πρωτόκολλο ασύρματης εταιρείας ESP-NOW [10] της Espressif αποτελεί ιδανική επιλογή καθώς είναι ενσωματωμένο μέσα στην μονάδα ESP32 S3. Το Raspberry Pi 4 προσφέρει επίσης επικοινωνία μέσω Wi-Fi και έναν δυνατό τετραπύρηνο επεξεργαστή στα 1.8GHz όμως το κόστος του, 35€ για το φθηνότερο μοντέλο, είναι αυξημένο σε σχέση με αυτό του ESP32 S3, 10€ για το φθηνότερο μοντέλο, χωρίς να μας προσφέρει κάποιο ουσιαστικό πλεονέκτημα. Εξαιτίας των παραπάνω παραγόντων έγινε επιλογή του ESP32 S3 ως τον κύριο μικροελεγκτή καθώς το χαμηλό κόστος, η αυξημένη υπολογιστική ισχύ και η δυνατότητα ασύρματης επικοινωνίας μέσω του πρωτοκόλλου ESP-NOW καλύπτουν πλήρως τις απαιτήσεις μας.

2.1.1 Η μονάδα ESP32 S3 της Espressif

Η μονάδα ESP32 S3 (Σχήμα 1[1]) αποτελεί την κεντρική μονάδα των ενσωματωμένων συστημάτων μας. Το ESP32 S3 αποτελεί ένα System On Chip (SoC)[11] , δηλαδή είναι ένας μικροελεγκτής ο οποίος συνδυάζει πολλαπλά συστήματα μέσα σε ένα τσιπ. Εμπεριέχει έναν επεξεργαστή, προσωρινή μνήμη (RAM), μνήμη για ανάγνωση μόνο Read Only Memory (ROM), ενσωματωμένη δυνατότητα Wi-Fi και Bluetooth, περιφερειακά GPIO (General Purpose Input Output) που επιτρέπουν την σύνδεση του τσιπ με εξωτερικές μονάδες και συστήματα διαχείρισης ενέργειας εντός του τσιπ.

[1] Η πηγή της εικόνας: https://thepihut.com/products/esp32-s3-wroom-1-n4-module-pcb-antenna



Σχήμα 1: ESP32 S3 with antenna

Επεξεργαστής: Ο επεξεργαστής του ESP32 S3 είναι ο Xtensa® dual-core 32-bit LX7 microprocessor, ένας πολύ αποδοτικός διπύρηνος επεξεργαστής με ταχύτητα ρολογιού μέχρι 240 MHz, αυτό σημαίνει ότι σε 1 δευτερόλεπτο το εσωτερικό ρολόι του επεξεργαστή θα στείλει 240000000 παλμούς. Κάθε φορά που στέλνεται ένας παλμός ο επεξεργαστής εκτελεί τμήμα μιας εντολής ή ακόμα και ολόκληρη την εντολή αν είναι εφικτό. Ο επεξεργαστής συνεργάζεται στενά με την προσωρινή μνήμη RAM και την μνήμη μόνο ανάγνωσης ROM(read only memory). Οι εντολές αυτές καλύπτουν όλες τις μαθηματικές πράξεις, πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση αλλά και σύγκριση αριθμών, μεταφορά δεδομένων σε κάποια διεπαφή εξόδου, ανάγνωση και αποθήκευση δεδομένων από διεπαφές εισόδου.

Προσωρινή Μνήμη RAM: Η προσωρινή μνήμη αποτελεί τον χώρο στον οποίο ο επεξεργαστής αποθηκεύει δεδομένα τα οποία θέλει να κρατήσει κατά την διάρκεια εκτέλεσης του προγράμματος και πιθανώς να ξαναχρησιμοποιήσει αργότερα. Το συνολικό της μέγεθος είναι 512kB ενώ κατά την απενεργοποίηση του SoC η μνήμη αυτή καθαρίζει και τα δεδομένα χάνονται.

Μνήμη μόνο ανάγνωσης: Η μνήμη ROM αποτελεί το μέρος του τσιπ στο οποίο εμπεριέχεται λογισμικό για την φυσιολογική λειτουργία του τσιπ όπως την αρχικοποίηση της μνήμης και την φόρτωση του κώδικα μας. Δε μπορούμε να την αλλοιώσουμε και ενεργοποιείται αμέσως μόλις δώσουμε παροχή στον επεξεργαστή μας. Τέλος το μέγεθος της είναι 384kB.

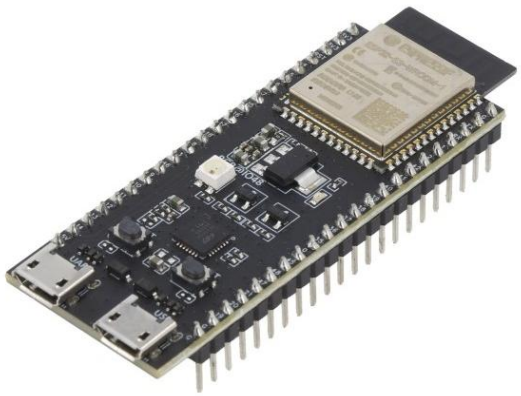
Wi-Fi: Το SoC εμπεριέχει μονάδα που του δίνει δυνατότητες χρήσης Wi-Fi και είναι ένα από τα χαρακτηριστικά που κάνουν το ESP32 S3 να ξεχωρίσει στην αγορά. Υποστηρίζει 3 πρωτόκολλα Wi-Fi τα 802.11b/g/n με μέγιστη ταχύτητα ανταλλαγής δεδομένων τα 150Mbps.

Bluetooth: Ένα ακόμα χαρακτηριστικό του τσιπ είναι η δυνατότητα χρήσης Bluetooth και συγκεκριμένα της τεχνολογίας Bluetooth LE (Low Energy) η οποία επιτρέπει στο ESP32 S3 να έχει μειωμένη κατανάλωση κατά την ενεργοποίηση και κατά την χρήση του.

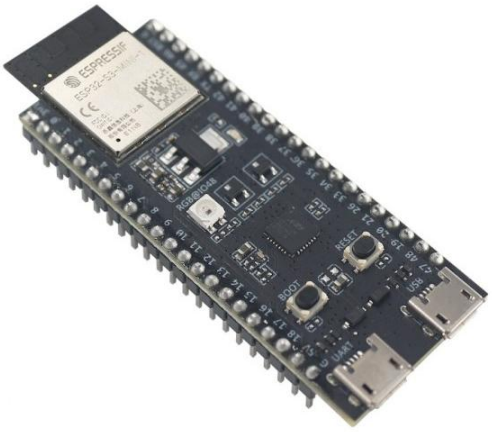
Περιφερειακά εξόδου και εισόδου GPIO: Το ESP32 S3 προσφέρει 45 διεπαφές που μπορεί να χρησιμοποιήσει ο χρήστης για να εισάγει δεδομένα στο SoC ή να πάρει πίσω από αυτό δεδομένα. Σε συνδυασμό με τον επεξεργαστή έχουμε δυνατότητες όπως την μετατροπή αναλογικής εισόδου σε ψηφιακή (ADC), την ανάγνωση ψηφιακής εισόδου, την ανάγνωση αισθητήρων αφής, την έξοδο ψηφιακών σημάτων, την χρήση πρωτοκόλλων επικοινωνίας τα οποία θα αναλύσουμε παρακάτω και πολλά ακόμα.

2.1.2 Τα ολοκληρωμένα ενσωματωμένα συστήματα με το ESP32 S3 SoC

Η μονάδα ESP32 S3 αποτελεί ένα πολύ ισχυρό SoC, για την διευκόλυνση των χρηστών όμως έχουν δημιουργηθεί κιτ τα οποία προσφέρουν ένα ολοκληρωμένο ενσωματωμένο σύστημα. Εμπεριέχουν εξωτερική μνήμη τύπου SPI Flash στην οποία αποθηκεύεται όλος ο κώδικας που θέλουμε να τρέξει το SoC, διεπαφές USB και UART για την σύνδεση του ενσωματωμένου με άλλες συσκευές, συνήθως υπολογιστές για τον προγραμματισμό του ενσωματωμένου ή την παροχή ρεύματος προς αυτό. Από μόνο του το SoC χρειάζεται συγκεκριμένη τάση εισόδου, από 3 έως 3.6 Volt, το ολοκληρωμένο ενσωματωμένο παρέχει την δυνατότητα στον χρήση να εισάγει τάση 5V στην ειδική διεπαφή 5V καθώς χρησιμοποιεί έναν ρυθμιστή τάσης ο οποίος μετατρέπει την είσοδο που δίνουμε σε 3.3 Volt. Με αυτό τον τρόπο το ESP32 S3 μπορεί να όταν τροφοδοτείται από ένα USB ή από μια εξωτερική πηγή με τάση εισόδου 5V. Σε αυτή την διπλωματική εργασία θα κάνουμε χρήση 2 διαδεδομένων ενσωματωμένων για το ESP32 S3 SoC τα οποία είναι το ESP32 S3 DevKit C1[12] (Σχήμα 2) και το ESP32 S3 DevKit M1[13] (Σχήμα 3).

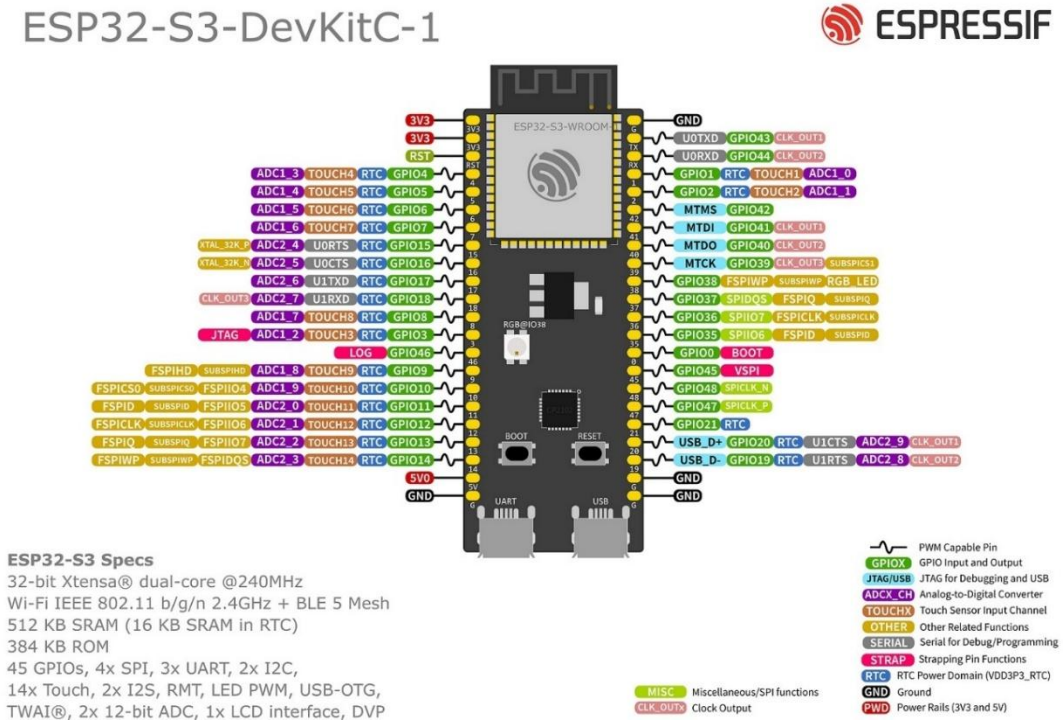


Σχήμα 2: ESP32 S3 DevKit C1

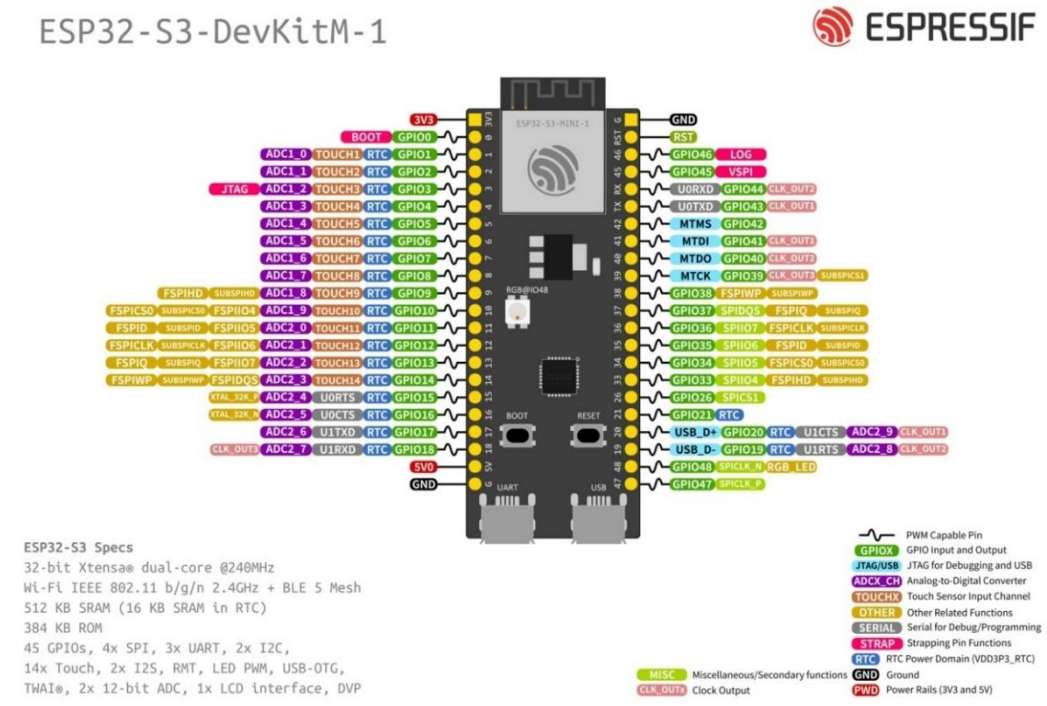


Σχήμα 3: ESP32 S3 DevKit M1

Από κάτω παρέχεται αναλυτική εικόνα όλων των διεπαφών των ολοκληρωμένων πλακετών ESP32 S3 DevKit C1 (Σχήμα 4[2]) και ESP32 S3 DevKit M1 (Σχήμα 5[3]).



Σχήμα 4: Οι διεπαφές του ESP32 S3 DevKit C1



Σχήμα 5: Οι διεπαφές του ESP32 S3 DevKit M1

[2] Η πηγή της εικόνας: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitc-1/user_guide_v1.0.html

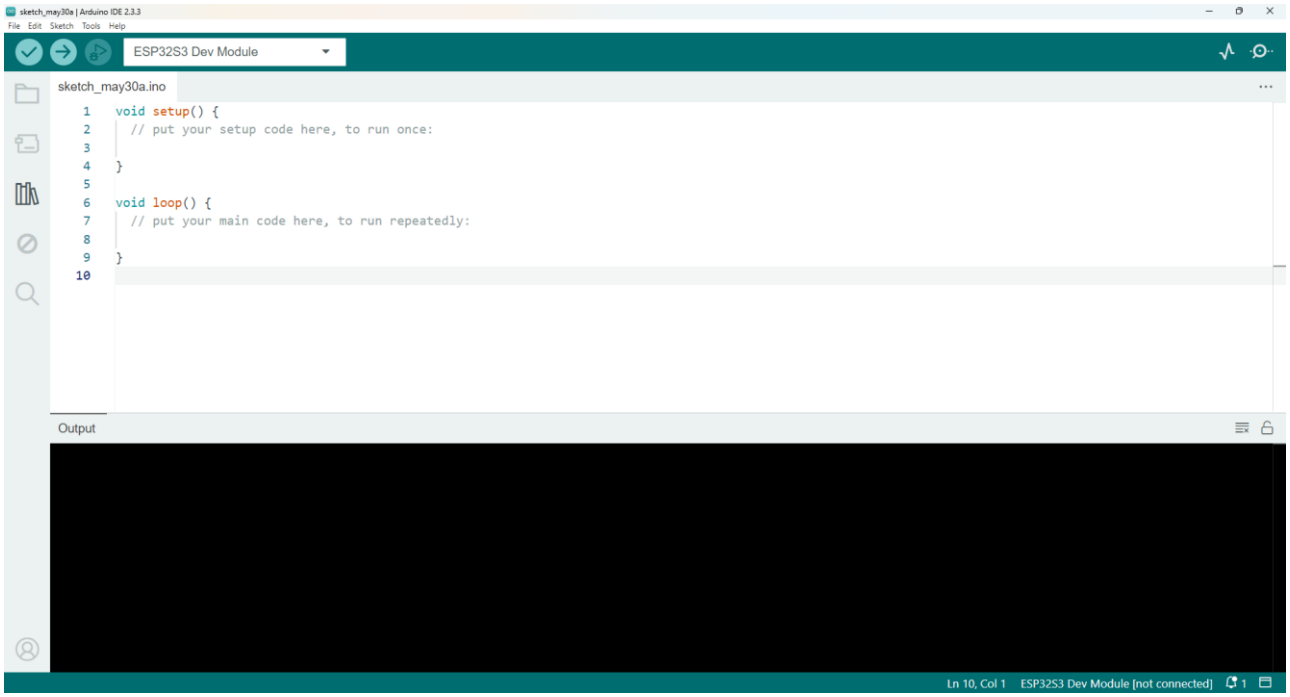
[3] Η πηγή της εικόνας: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitm-1/user_guide.html

Από τις εικόνες παρατηρούμε ότι το ESP32 S3 DevKit C1 παρέχει περισσότερες γειώσεις, 2 εξόδους τάσης στα 3.3V και συνολικά 36 διεπαφές γενικής εισόδου εξόδου σε αντίθεση με το ESP32 S3 Devkit M1 το οποίο παρέχει 2 γειώσεις, 1 έξοδο στα 3.3V, 1 έξοδο στα 5V και συνολικά 39 διεπαφές γενικής εισόδου εξόδου. Το ESP32 S3 DevKit C1 χρησιμοποιεί το ESP32 S3 WROOM1[14] με 8MB μνήμης Flash, ενώ το ESP32 S3 Devkit M1 χρησιμοποιεί το ESP32 S3 MINI1[15] με 4MB μνήμης Flash.

2.2 Εργαλεία ανάπτυξης του λογισμικού και του υλικού

2.2.1 Το λογισμικό προγραμματισμού Arduino IDE

Για τον προγραμματισμό αυτών των δύο ενσωματωμένων θα κάνουμε χρήση ενός πολύ διαδεδομένου προγράμματος, του Arduino IDE[16] (Σχήμα 6). Το Arduino IDE είναι ένα λογισμικό open source (ανοιχτού κώδικα) το οποίο χρησιμοποιείται για τον προγραμματισμό μικροελεγκτών Arduino. Το πρόγραμμα με το πέρασμα των ετών έχει εξελιχθεί και πλέον με την χρήση κατάλληλων βιβλιοθηκών που παρέχονται από την εταιρεία Espressif είναι εφικτός ο προγραμματισμός μικροελεγκτών ESP32 S3.Για να επιτύχουμε τον προγραμματισμό μίας συσκευής ESP32 S3 πρέπει να στήσουμε το περιβάλλον κατάλληλα. Το πρόγραμμα παρέχει στους χρήστες την δυνατότητα να επιλέξουν τον μικροελεγκτή που επιθυμούν να προγραμματίσουν προσφέροντας μια τεράστια γκάμα επιλογών. Στο παράρτημα Α παρέχονται περισσότερες πληροφορίες σχετικά με την προετοιμασία του περιβάλλοντος για τον προγραμματισμό των ESP32 S3

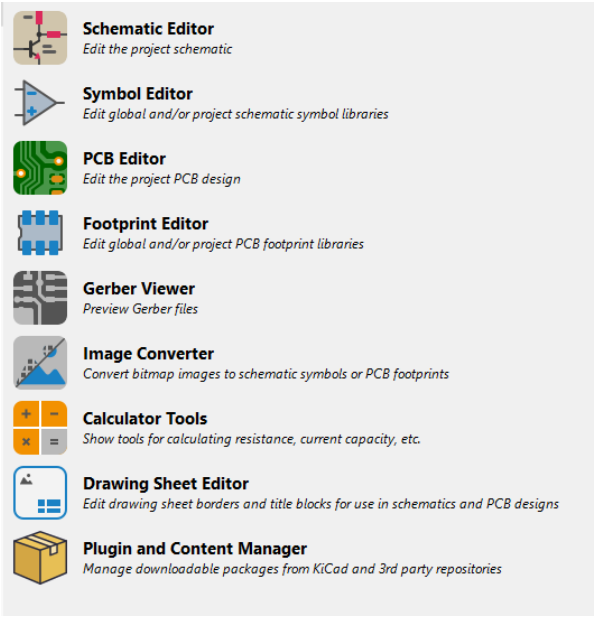


Σχήμα 6: Η επιφάνεια εργασίας του λογισμικού Arduino IDE

2.2.2 Το λογισμικό δημιουργίας τυπωμένης πλακέτας κυκλώματος

Ο εγκέφαλος του ενσωματωμένου συστήματος του ασύρματου χειριστηρίου είναι τοποθετημένος πάνω σε μια τυπωμένη πλακέτα PCB (Printed Circuit Board) μαζί με μια οθόνη και 12 απτικά κουμπιά SPST (Single Pole Single Throw, έχουν δηλαδή μόνο 2 καταστάσεις ON και OFF) ή αλλιώς tactile buttons. Για την δημιουργία του κυκλώματος και κατά επέκταση την δημιουργία του PCB έγινε χρήση του δωρεάν προγράμματος Kicad[17]. Το συγκεκριμένο λογισμικό επιτρέπει στους χρήστες να φτιάξουν ένα schematic (κύκλωμα) το οποίο μπορούν αργότερα να μετατρέψουν σε PCB. Έχουν την δυνατότητα να τοποθετήσουν ηλεκτρικά εξαρτήματα και έπειτα να δημιουργήσουν μονοπάτια για το ρεύμα ώστε να τα ενώσουν.

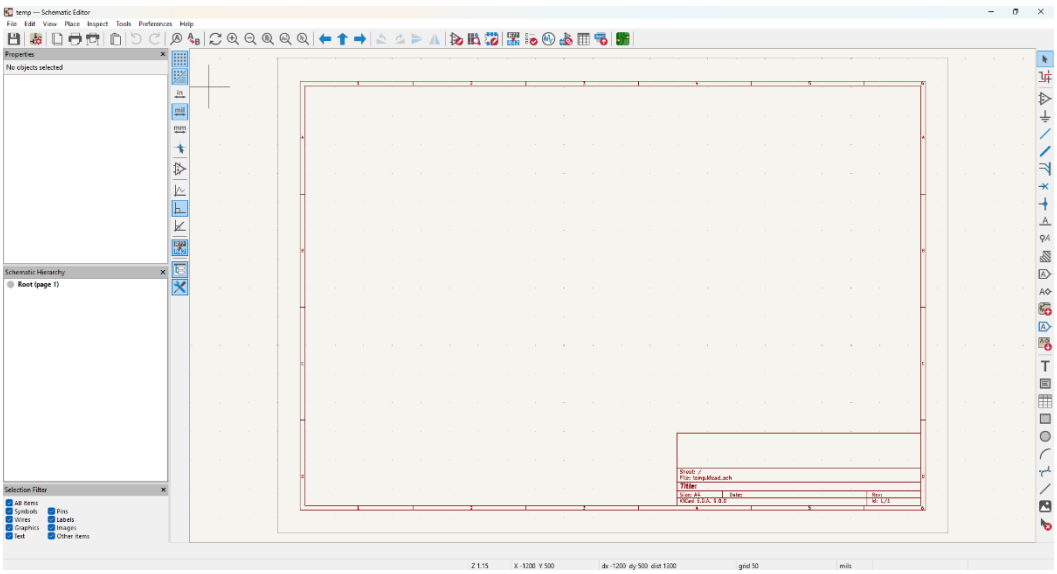
Σχήμα 7: Το λογότυπο του προγράμματος Kicad



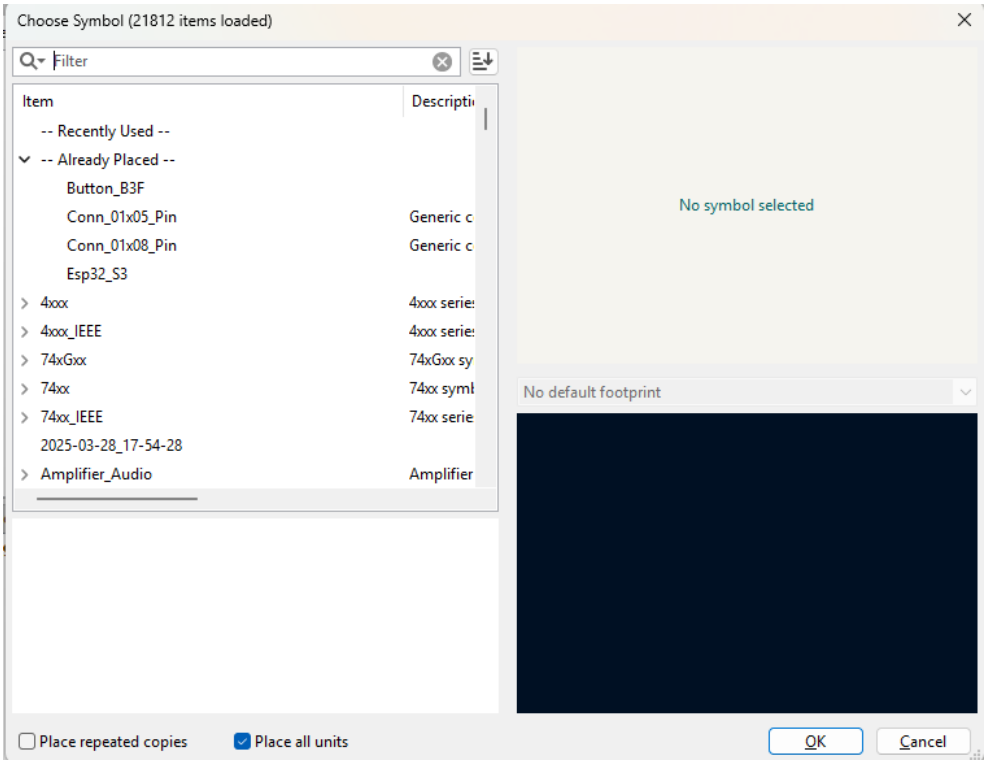
Σχήμα 8: Το αρχικό μενού του προγράμματος

Η λειτουργία του προγράμματος αποτελείται από διακριτά στάδια καθιστώντας την φιλική προς τους χρήστες. Αρχικά σχεδιάζουμε το κύκλωμα έπειτα κάνουμε όλες τις συνδέσεις μεταξύ των εξαρτημάτων και τέλος μετατρέπουμε το schematic σε PCB. Θα αναλύσουμε κάθε βήμα παρακάτω:

Φάση σχεδίασης (Schematic Editor): Αφού αναλύσουμε τις απαιτήσεις του συστήματος μας ξεκινάμε την σχεδίαση του κυκλώματος μας επιλέγοντας από το αρχικό μενού την επιλογή Schematic Editor (Σχήμα 8). Αυτό απαιτεί την επιλογή των κατάλληλων εξαρτημάτων και την σωστή σύνδεση όλων αυτών. Στο Kicad η επιλογή εξαρτημάτων γίνεται από το παράθυρο στο δεξιά μέρος της οθόνης με όνομα “Place Symbol”. Άμα το επιλέξουμε ανοίγει ένα μενού με διάφορες κατηγορίες συμβόλων (Σχήμα 10) . Τα σύμβολα αντιπροσωπεύουν εξαρτήματα τα οποία μπορούμε να βάλουμε στο κύκλωμα μας. Στην μπάρα αναζήτησης μπορούμε να επιλέξουμε το εξάρτημα και να το τοποθετήσουμε τον κύριο χώρο δημιουργίας κυκλώματος.

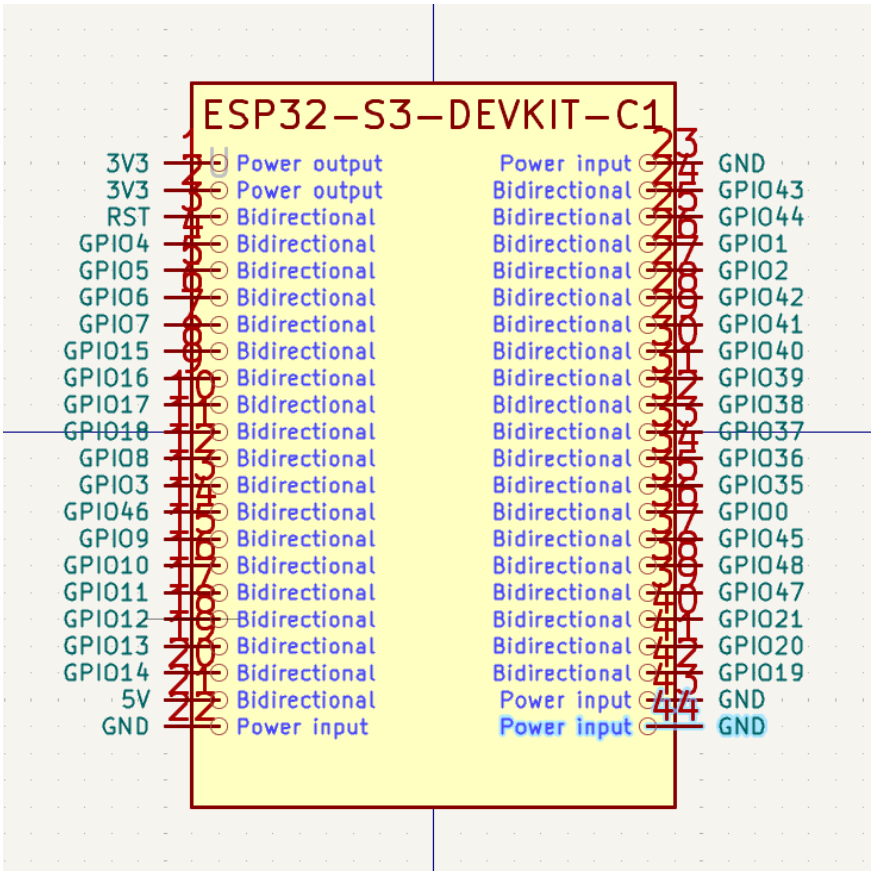


Σχήμα 9: Η επιφάνεια εργασίας του λογισμικού Kicad



Σχήμα 10: Μενού προσθήκης συμβόλων

Είναι πιθανό η αναζήτηση μας να μη γυρίσει το αποτέλεσμα που επιθυμούμε, για αυτό το λόγο το Kicad μας προσφέρει την δυνατότητα να φτιάξουμε δικά μας σύμβολα επιλέγοντας από το αρχικό μενού το “Symbol Editor”. Μέσα στο Symbol Editor μπορούμε να δημιουργήσουμε δικά μας σύμβολα, ορίζουμε εισόδους, εξόδους, γειώσεις, αριθμό επαφών και πολλά ακόμα. Καθώς το ESP32 S3 Devkit C1 δεν υπήρχε σαν έτοιμο σύμβολο κληθήκαμε να κατασκευάσουμε το δικό μας (Σχήμα 11). Στο παράρτημα Β δίνονται περαιτέρω οδηγίες για την δημιουργία συμβόλων.

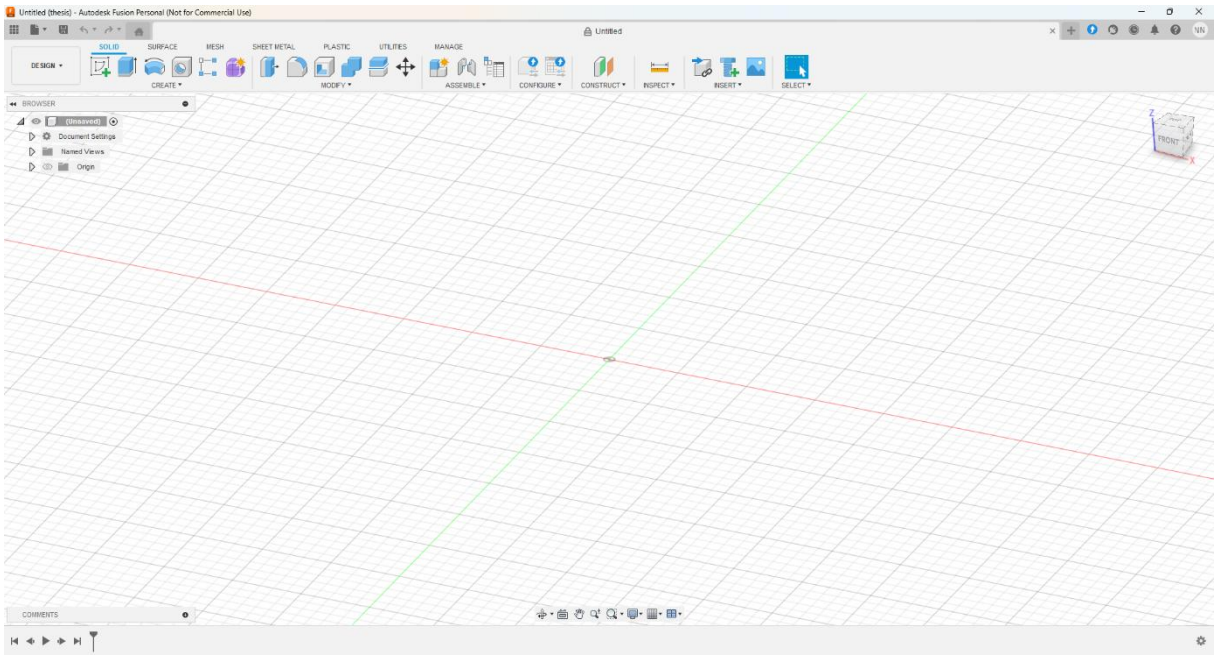


Σχήμα 11: Προσαρμοσμένο σύμβολο ESP32 S3 DevKit C1

Φάση Δημιουργίας PCB: Σε αυτό το στάδιο μπορούμε να τοποθετήσουμε τα εξαρτήματα που συνδέσαμε στο στάδιο σχεδίασης πάνω στην πλακέτα. Έπειτα μπορούμε να δημιουργήσουμε μονοπάτια χαλκού τα οποία συνδέουν 2 ή περισσότερα εξαρτήματα μεταξύ τους. Αφού είμαστε ικανοποιημένοι με την σχεδίαση της πλακέτας μας μπορούμε να κάνουμε εξαγωγή της πλακέτας σε αρχεία .Gerber τα οποία χρησιμοποιούν βιομηχανίες κατασκευής PCB όπως το JLCPCB για την εκτύπωση του κυκλώματος μας πάνω σε πλακέτες.

2.2.3 Το λογισμικό δημιουργίας μοντέλων για τρισδιάστατη εκτύπωση Fusion360

Το τελευταίο λογισμικό που χρησιμοποιήσαμε ονομάζεται Fusion 360 από την εταιρεία Autodesk. Το fusion 360 είναι ένα λογισμικό το οποίο προσφέρει πολλές δυνατότητες όπως την δημιουργία τρισδιάστατων μοντέλων για 3D εκτύπωση, τον έλεγχο αντοχής υλικών, την δημιουργία animation, την δυνατότητα παραγωγής φωτορεαλιστικών απεικονίσεων μοντέλων και άλλα. Εμείς θα κάνουμε χρήση μόνο της δημιουργίας τρισδιάστατων μοντέλων τα οποία αργότερα θα στείλουμε στο πρόγραμμα εκτύπωσης που έρχεται μαζί με τον 3D εκτυπωτή μας. Η διαδικασία ξεκινάει με την δημιουργία ενός δυσδιάστατου σκίτσου και στην συνέχεια με εντολές του λογισμικού όπως το extrude, fillet και chamber δίνουμε ύψος και σχήμα σε αυτό. Μας δίνεται επίσης η δυνατότητα να δημιουργήσουμε τρύπες για βίδες με την λειτουργία hole.



Σχήμα 12: Επιφάνεια εργασίας προγράμματος Fusion 360

2.3 Τα πρωτόκολλα επικοινωνίας των ESP32 S3

Το ESP32 S3 παρέχει πολλά πρωτόκολλα επικοινωνίας με περιφερειακές συσκευές σε αυτό το κεφάλαιο όμως θα αναλύσουμε αυτά τα οποία χρησιμοποιήσαμε στην παρούσα εργασία. Αυτά είναι το σειριακό πρωτόκολλο I2C, το σειριακό πρωτόκολλο επικοινωνίας SPI και το ασύρματο πρωτόκολλο ESP-NOW.

2.3.1 Το σειριακό πρωτόκολλο επικοινωνίας I2C (Inter Integrated Circuit)

Το πρωτόκολλο I2C αποτελεί ένα πρότυπο σειριακής επικοινωνίας[18]. Είναι σχεδιασμένο για επικοινωνία μεταξύ πολλαπλών συσκευών σε κοντινές αποστάσεις με χαμηλή ταχύτητα επικοινωνίας, 100KHz έως 400KHz. Το πρωτόκολλο κάνει χρήση 2 σημάτων:

- SDA (Serial Data): Το σήμα αυτό χρησιμοποιείται για την μεταφορά των δεδομένων από την μία συσκευή στην άλλη
- SCL (Serial Clock): Το σήμα αυτό κουβαλάει τον παλμό του ρολογιού που χρησιμοποιείται για τον συγχρονισμό των συσκευών.

Για να μπορέσουμε να περιγράψουμε την βασική αρχή λειτουργίας του πρωτοκόλλου αυτού πρέπει αρχικά να εξηγήσουμε τους όρους controller (Χειριστής) και peripheral (περιφερειακή μονάδα).

- Controller: Είναι η συσκευή υπεύθυνη για την έναρξη της επικοινωνίας. Στέλνει τους παλμούς του ρολογιού, τα δεδομένα που θέλει προς τα περιφερειακά που θέλει και έπειτα λήγει την επικοινωνία.
- Peripheral: Η συσκευή αυτή συνδέεται με τα σήματα που αναφέραμε παραπάνω με τον αφέντη. Έχει μια μοναδική διεύθυνση επικοινωνίας που της αντιστοιχίζεται και περιμένει μέχρι ο controller να ξεκινήσει να στέλνει δεδομένα προς αυτόν κάνοντας χρήση της μοναδικής του διεύθυνσης. Σε πολλές περιπτώσεις στέλνει πίσω στον controller δεδομένα αφού πολλές συσκευές peripheral είναι αισθητήρες.

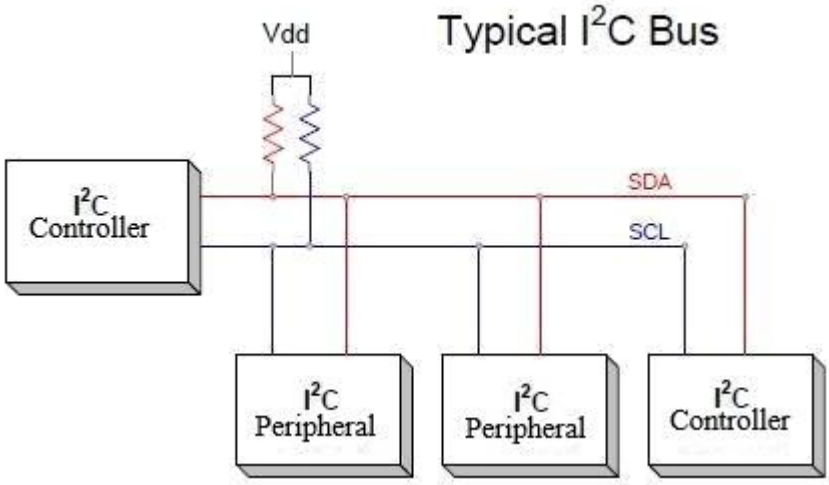
Για να ξεκινήσει η επικοινωνία ο controller μηδενίζει την τάση στην γραμμή SDA ενώ ταυτόχρονα ανεβάζει την τάση στην γραμμή SCL, με αυτό τον τρόπο ειδοποιούνται οι περιφερειακές μονάδες που είναι συνδεδεμένες με τον controller ότι θα ξεκινήσει η μετάδοση δεδομένων. Κάθε παλμός του ρολογιού αντιστοιχεί και σε ένα bit. Τα πακέτα επικοινωνίας χωρίζονται σε 2 μορφές:

- Πακέτο διεύθυνσης: Αποτελεί το πρώτο πακέτο που στέλνει ο controller με το που ξεκινήσει η επικοινωνία με τις περιφερειακές μονάδες. Περιέχει 7 bit που αντιστοιχούν στην διεύθυνση του περιφερειακού που θέλει να επικοινωνήσει και μετά 1 bit για να δηλώσει άμα θέλει να λάβει δεδομένα από τα περιφερειακά ή να του στείλει. Αν το περιφερειακό ανιχνεύσει σωστά την διεύθυνση του και το bit του τύπου επικοινωνίας, αποστολής ή λήψης δεδομένων, στον ένατο παλμό του ρολογιού ανεβάζει την τάση στην γραμμή SDA. Αν δεν γίνει αυτό τότε ο controller θεωρεί ότι η επικοινωνία με το περιφερειακό δεν ήταν επιτυχής και το bit αναγνώρισης γίνεται NACK (not acknowledged).
- Πακέτο δεδομένων: Αφού η επικοινωνία επιτευχθεί μεταξύ του controller και της περιφερειακής μονάδας ο υπεύθυνος για την αποστολή των δεδομένων, ο οποίος έχει οριστεί από το πακέτο διεύθυνσης, ξεκινάει να στέλνει πακέτα των 8 bit στην γραμμή SDA. Καθ' όλη την διάρκεια ο controller ελέγχει τους παλμούς ρολογιού ώστε να συγχρονίσει την αποστολή ή την ανάγνωση του κάθε bit. Στο τέλος το περιφερειακό μηδενίζει την τάση στην γραμμή SDA για να υποδηλώσει επιτυχή λήψη ή αποστολή των δεδομένων. Αυτή η διαδικασία μπορεί να γίνει πολλαπλές φορές μέχρι να αποσταλούν όλα τα δεδομένα. Τότε το περιφερειακό κρατάει την τάση στην γραμμή SDA σε λογικό 1 (δηλαδή έχει τάση) για να δηλώσει το τέλος της επικοινωνίας, δηλαδή NACK.

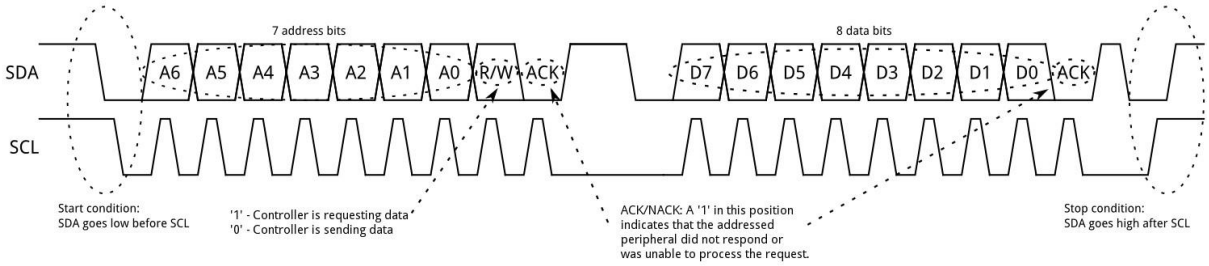
Αν ο controller λάβει NACK στο bit αναγνώρισης σε κάποιο από τα 2 πακέτα τότε η επικοινωνία διακόπτεται. Όταν γίνει λήψη όλων των πακέτων δεδομένων ο controller διακόπτει την επικοινωνία δίνοντας τάση στην γραμμή SDA ενώ η γραμμή SCL βρίσκεται ήδη στην κατάσταση του λογικού 1.

Οι παρακάτω εικόνες αναλύουν, την σύνδεση πολλαπλών χειριστών και περιφερειακών κάνοντας χρήση του πρωτοκόλλου I2C (Σχήμα 13[4]) και την δομή των πακέτων κατά την επικοινωνία ενός controller με μία περιφερειακή μονάδα (Σχήμα 14[5]).

[4] Η πηγή της εικόνας: https://docs.nanoframework.net/content/getting-started-guides/i2c-explained.html
[5] Η πηγή της εικόνας: https://learn.sparkfun.com/tutorials/i2c/all



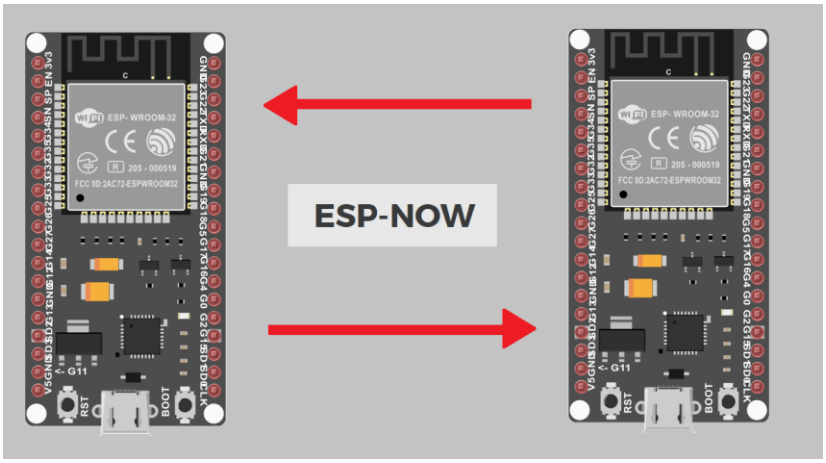
Σχήμα 13: Η δομή της επικοινωνίας στο πρωτόκολλο I2C



Σχήμα 14: Η δομή των πακέτων στο πρωτόκολλο I2C

2.3.2 Το πρωτόκολλο ασύρματης επικοινωνίας ESP-NOW

Το ESP-NOW [10] αποτελεί ένα πρωτόκολλο ασύρματης επικοινωνίας ειδικά φτιαγμένο για τα ESP32. Κάνει χρήση της τεχνολογίας του Wi-Fi και επιτρέπει σε πολλαπλά ESP32 να επικοινωνούν μεταξύ τους. Με βάση του μοναδικού κωδικού MAC που έχει κάθε συσκευή ESP32 γίνεται εφικτή η αποστολή προσωποποιημένων μηνυμάτων. Το ESP-NOW είναι ικανό να υποστηρίξει 10 ταυτόχρονες συνδέσεις οι οποίες μπορούν να φτάσουν τις 20 άμα ο χρήστης επιλέξει να μη κάνει κρυπτογράφηση των πακέτων. Τα πακέτα αυτά (payload) έχουν μέγεθος 250 Byte το καθένα και 247 Byte άμα επιλεχθεί να γίνει και κρυπτογράφηση τους. Η εμβέλεια της επικοινωνίας εξαρτάται από παράγοντες όπως εμπόδια, τοιχώματα, άλλες ηλεκτρονικές συσκευές κλπ. Σε κλειστούς χώρους με πολλά εμπόδια η εμβέλεια του πρωτοκόλλου είναι στα 30 με 50 μέτρα, ενώ σε εξωτερικούς χώρους χωρίς εμπόδια αυτή αυξάνεται στα 200 μέτρα με ελάχιστα πακέτα να χάνονται. Υπάρχει επίσης η δυνατότητα προσθήκης εξωτερικής κεραίας η οποία μπορεί να αυξήσει την εμβέλεια δραματικά.



Σχήμα 15: Το πρωτόκολλο επικοινωνίας ESP-NOW

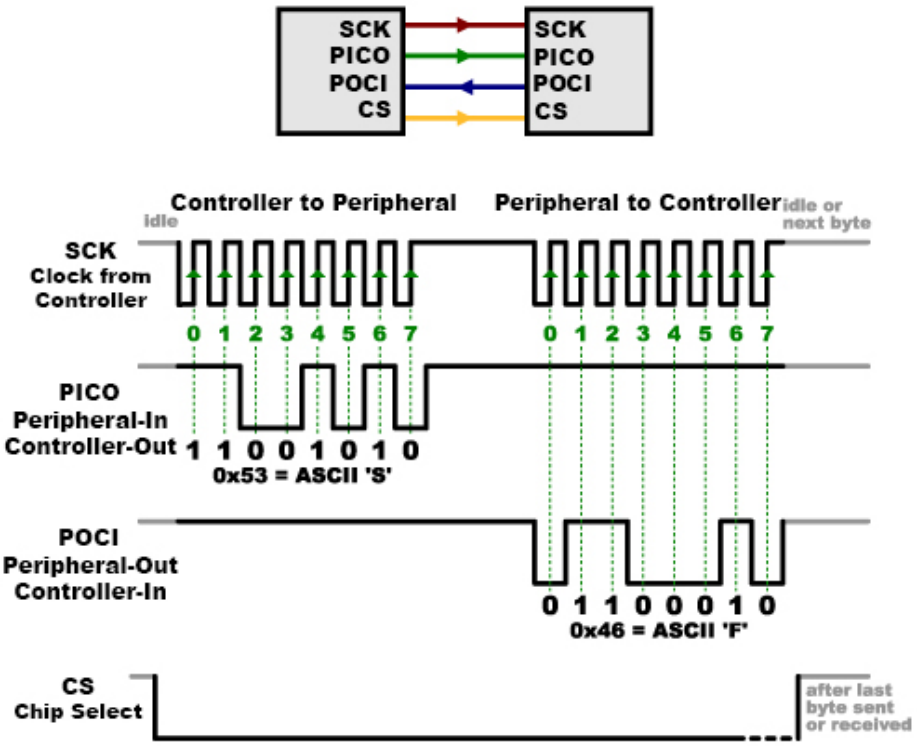
2.3.3 Το σειριακό πρωτόκολλο επικοινωνίας SPI (Serial Peripheral Interface)

Το τελευταίο πρωτόκολλο το οποίο θα χρησιμοποιήσουμε ονομάζεται SPI. Αποτελεί ένα πρωτόκολλο παρόμοιο με το I2C που αναλύσαμε παραπάνω με την λογική ότι υπάρχει ένας controller και ένα ή περισσότερα peripheral. Οι διαφορές έγκειται στην απαίτηση περισσότερων σημάτων, 4 για να είμαστε ακριβής (Σχήμα 16[6]). Αυτά είναι τα ακόλουθα:

- SCL (Serial Clock): Όπως και στο I2C έτσι και εδώ έχουμε έναν κοινό παλμό ο οποίος χρησιμοποιείται για τον συγχρονισμό του controller και των peripheral για την αποστολή και λήψη δεδομένων.
- POCI (Controller Out Peripheral In): Το σήμα αυτό κουβαλάει τα δεδομένα που θέλει να στείλει ο controller στο peripheral.
- PICO (Controller In Peripheral Out): Το σήμα αυτό κουβαλάει τα δεδομένα που θέλει να στείλει το peripheral προς τον controller, το αντίθετο του POSI.
- CS (Chip Select): Το σήμα αυτό ειδοποιεί το peripheral που θέλουμε να διαβάσει ή να στείλει δεδομένα όταν ο controller μηδενίσει την τάση.

Όπως παρατηρούμε το συγκεκριμένο πρωτόκολλο απαιτεί περισσότερα σήματα για να λειτουργήσει, παρόλα αυτά μας προσφέρει μεγαλύτερες ταχύτητες επικοινωνίας, για τους περισσότερους μικροελεγκτές είναι 16MHz, καθώς η ανταλλαγή δεδομένων μεταξύ controller και peripheral γίνεται ταυτόχρονα. Επίσης δεν έχουμε πακέτα όπως στο πρωτόκολλο I2C ούτε bit αναγνώρισης (ACK) καθιστώντας την δομή του πιο απλή και εύκολη. Η επικοινωνία λήγει μόλις ο controller επαναφέρει την τάση στο σήμα CS.

[6] Η πηγή της εικόνας: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi



Σχήμα 16: Το πρωτόκολλο επικοινωνίας SPI

2.4 Η δομή του οχήματος και το ενσωματωμένο σύστημα ελέγχου

Σε αυτή την ενότητα θα δούμε πιο αναλυτικά την δομή του οχήματος της διπλωματικής εργασίας και την πλακέτα την οποία επιλέξαμε για τον έλεγχο του.

2.4.1 Η δομή του οχήματος

Το σασί του οχήματος είναι πολύ σημαντικό για την κατασκευή μας. Το όχημα θέλουμε να έχει την ικανότητα προσπέλασης ανισόπεδου εδάφους, δυνατούς κινητήρες ώστε να ανεβαίνει δρόμους με κλίση και χαλίκια, στιβαρό σώμα το οποίο είναι ανθεκτικό σε κραδασμούς και χτυπήματα και τέλος να έχουμε την δυνατότητα να προσθέσουμε επεκτάσεις όπως τον ρομποτικό βραχίονα. Έπειτα από έρευνα αγοράς καταλήξαμε στην επιλογή του Wild Thumper 6WD (6 Wheel Drive)[19] της εταιρείας Dagu. Πρόκειται για ένα σασί φτιαγμένο από αλουμίνιο στο οποίο έχουν τοποθετηθεί 6 κινητήρες DC, 3 σε κάθε πλευρά (Σχήμα 17). Είναι εξοπλισμένο με ένα ανεξάρτητο σύστημα αναρτήσεων και ρόδες με μαλακά λαστιχένια καρφιά ειδικά σχεδιασμένα για ανώμαλο δρόμο (Σχήμα 18). Το σώμα του οχήματος είναι γεμάτο τρύπες 3 χιλιοστών για την τοποθέτηση εξαρτημάτων προσφέροντας δυνατότητες τροποποίησης. Οι κινητήρες είναι συνδεδεμένοι παράλληλα ανά πλευρά και ο καθένας έχει ρεύμα εκκίνησης τα 420mA στα 7.2V. Το μέγιστο δυνατό ρεύμα που μπορεί να τραβήξει ένας κινητήρας (stall current) είναι 6.6A στα 7.2V. Συνιστάται από τον κατασκευαστή μια πλακέτα

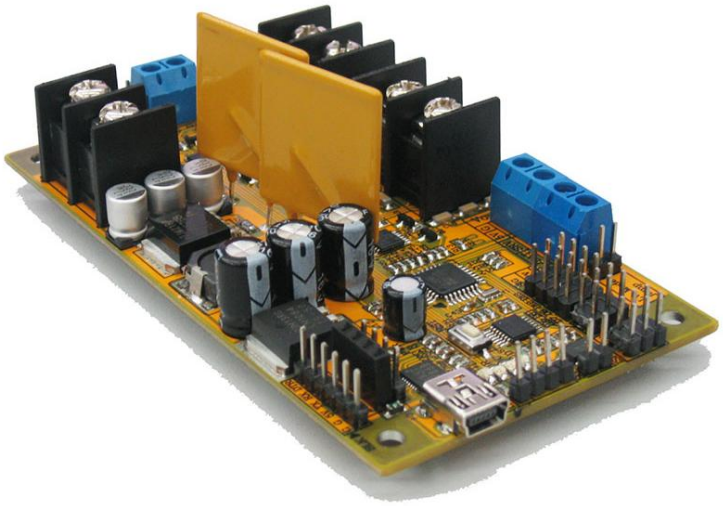
ελέγχου η οποία έχει 2 εξόδους τάσης (channel) και είναι ικανή να παρέχει έως και 20A ανά έξοδο.
Το βάρος του οχήματος ανέρχεται στα 2.7 κιλά ενώ οι διαστάσεις του είναι 420 x 300 x 130 χιλιοστά.

Σχήμα 17: Το όχημα Wild Thumper 6WD

Σχήμα 18: Ο τύπος ρόδας του οχήματος μαζί με
τον κινητήρα της

2.4.2 Η πλακέτα ελέγχου του οχήματος

Για τον ελέγχου του οχήματος ο κατασκευαστής προτείνει μία πλακέτα η οποία έχει 2 κανάλια εξόδου τάσης, μπορεί να παρέχει 20A στα 7.2V ανά κανάλι και ιδανικά έχει ενσωματωμένη προστασία από ηλεκτρική υπέρταση. Μία από τις προτάσεις του κατασκευαστή είναι η SMC G2 18v15, μια πλακέτα των 120 € η οποία καλύπτει όλες τις απαιτήσεις που αναφέραμε. Μετά από έρευνα αγοράς καταλήξαμε στην πλακέτα T’REX robot controller της εταιρείας Dagu (Σχήμα 19) η οποία επίσης καλύπτει τις ανάγκες μας αλλά στο μισό κόστος της SMC G2 18v15 (60€).



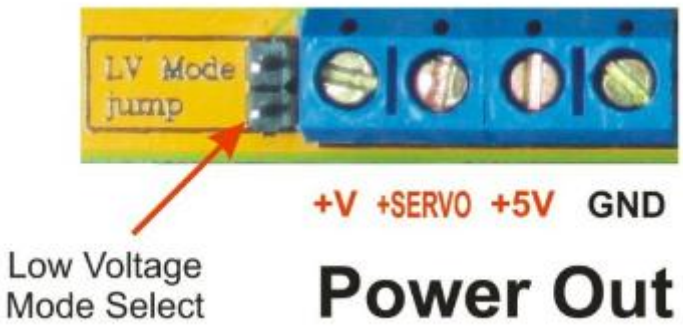
Σχήμα 19: Η πλακέτα T'REX Robot Controller της Dagu

Η συγκεκριμένη πλακέτα είναι εξοπλισμένη με έναν μικροελεγκτή Arduino και έχει κατασκευαστεί με στόχο τον έλεγχο κινητήρων. Προσφέρει 2 εξόδους τάσης (channel) οι οποίες μπορούν να παρέχουν έως και 40A ως μέγιστο ρεύμα (stall current), ενώ ταυτόχρονα οι επανασφαλιζόμενες (self-resetting) ασφάλειες προστατεύουν την πλακέτα από την υπερβολική άντληση ρεύματος από τους κινητήρες. Η τάση εισόδου της πλακέτας έχει 2 καταστάσεις:

- Κατάσταση χαμηλής τάσης: Σε αυτή την κατάσταση η πλακέτα περιμένει τάση εισόδου κάτω των 8V όπως για παράδειγμα μπαταρίες πολυμερών λιθίου (Li Po) στα 7.4V
- Κατάσταση υψηλής τάσης: Στην δεύτερη κατάσταση η πλακέτα δέχεται τάση εισόδου μεγαλύτερη των 8V έως και 30V.

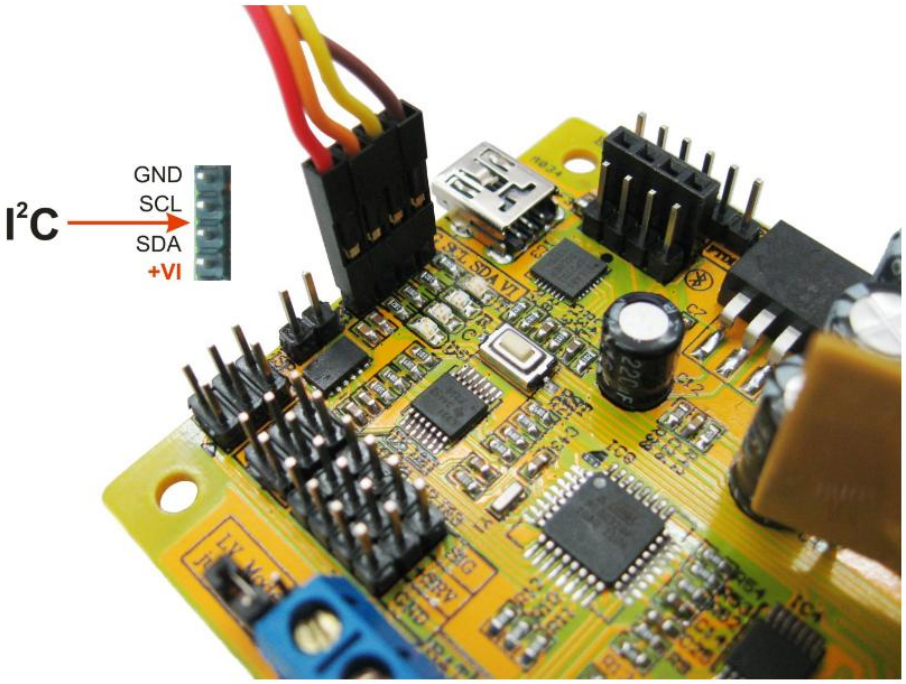
Η πλακέτα προσφέρει μια έξοδο σταθερής τάσης στα 5V για την τροφοδοσία εξωτερικού μικροελεγκτή ή αισθητήρων. Ανάλογα με την κατάσταση τάσης το μέγιστο ρεύμα που μπορεί να προσφέρει αυτή η έξοδος είναι 800mA στην χαμηλή τάση και 2A στην υψηλή τάση.

Για να επιλέξουμε ποια από τις 2 ρυθμίσεις θέλουμε το μόνο που έχουμε είναι να συνδέσουμε 2 επαφές πάνω στην πλακέτα με το όνομα "Low Voltage Mode" (Σχήμα 20).



Σχήμα 20: Επαφές ρύθμισης τάσης εισόδου

Ο μικροελεγκτής αυτής της πλακέτας είναι ένα Arduino Atmega328P, μέσα από τον οποίο μπορούμε να ελέγχουμε την ταχύτητα των κινητήρων αλλά και να λαμβάνουμε στοιχεία για την κατάσταση της πλακέτας. Η μονάδα μας δίνει 3 διαφορετικούς τρόπους με τους οποίος μπορούμε να επικοινωνήσουμε με τον μικροελεγκτή της, μέσω Bluetooth, μέσω ραδιοσυχνοτήτων (RC) με την χρήση εξωτερικής μονάδας με κεραία και τέλος μέσω του πρωτοκόλλου I2C (Σχήμα 21). Στην παρούσα εργασία κάναμε χρήση της τρίτης μεθόδου μετατρέποντας την πλακέτα σε περιφερειακό και το ESP32 S3 DevKit M1 σε χειριστή. Περισσότερες πληροφορίες για την επικοινωνία θα παρέχονται στο παράρτημα Γ. Τέλος η πλακέτα μας δίνει την δυνατότητα προσθήκης ενός εξωτερικού διακόπτη με την αποσύνδεση 2 επαφών.



Σχήμα 21: Η διεπαφή I2C της πλακέτας ελέγχου

Οι επαφές αυτές ελέγχουν ένα τρανζίστορ ειδικά φτιαγμένο για να αντέχει υψηλό ρεύμα, έως και 110A στιγμιαία. Η χρήση ενός διακόπτη ελέγχει άμα το τρανζίστορ άγει ή όχι ρεύμα ενεργοποιώντας και απενεργοποιώντας το περιφερειακό.

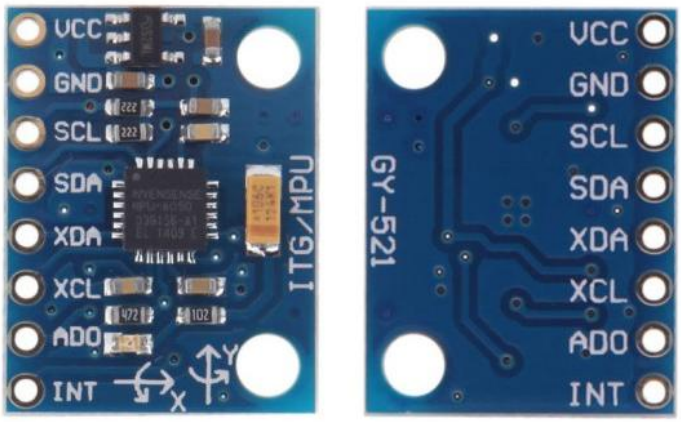
2.5 Οι περιφερειακές συσκευές του οχήματος και του τηλεχειριστηρίου

Σε αυτή την ενότητα θα αναλύσουμε την λειτουργία όλων των περιφερειακών συσκευών που θα προσθέσουμε στο όχημα αλλά και στο ασύρματο χειριστήριο.

2.5.1 Το επιταχυνσιόμετρο με γυροσκόπιο MPU6050

Στην διπλωματική αυτή κάνουμε χρήση ενός IMU (Inertial Measurement Unit) ονόματι MPU6050 (Σχήμα 22), για τον υπολογισμό της περιστροφής των ενσωματωμένων συστημάτων γύρω από τον κάθετο άξονα. Λειτουργεί μέσω του πρωτόκολλου I2C δίνοντας στον χρήστη την δυνατότητα να λαμβάνει τις μετρήσεις του για περαιτέρω χρήση. Η διεύθυνση

επικοινωνίας του είναι η 0x68 αλλά μας προσφέρεται η δυνατότητα να την αυξήσουμε κατά ένα και να την κάνουμε 0x69. Ένα από τα συχνότερα προβλήματα που αντιμετωπίζει αυτή η μονάδα είναι το φαινόμενου του Gimbal Lock[20], κατά το οποίο 2 άξονες περιστροφής κλειδώνουν μεταξύ τους χάνοντας έτσι έναν βαθμό ελευθερίας. Πολύ σημαντικό εργαλείο στην αντιμετώπιση αυτού είναι το DMP (Digital Motion Processor)[21] το οποίο παίρνει τα στοιχεία επιτάχυνσης και περιστροφής και έπειτα από επεξεργασία μας δίνει μετρήσεις πιο ακριβής χωρίς σφάλματα. Ένα μειονέκτημα αυτού του εργαλείου που πρέπει να αναφερθεί είναι ότι κατά την εκκίνηση θέλει λίγα δευτερόλεπτα για να σταθεροποιηθούν οι μετρήσεις του και έπειτα από εκτενής χρήση εμφανίζει κάποιο μικρό σφάλμα.

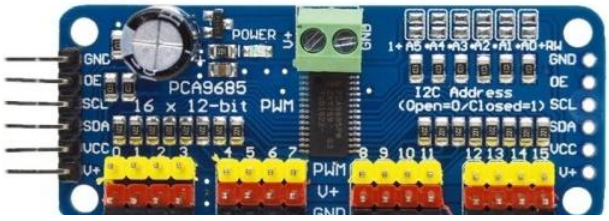


Σχήμα 22: Η πλακέτα GY-521 που χρησιμοποιεί το IMU

2.5.2 Η μονάδα διαχείρισης σερβοκινητήρων PCA9685 και οι σερβοκινητήρες του βραχίονα

Για τον έλεγχο των σερβοκινητήρων χρησιμοποιήσαμε την μονάδα PCA 9685[22] (Σχήμα 23). Η μονάδα αυτή παρέχει 16 κανάλια μεταβλητού πλάτους παλμού (PWM) με ανάλυση 12 bit, οι τιμές που στέλνει ξεκινάνε από το 0 και φτάνουν έως την τιμή 4096. Για την πλήρη κατανόηση αυτών των εννοιών θα γίνει μια συνοπτική αναφορά στον τρόπο λειτουργίας των σερβοκινητήρων στο παράρτημα Δ.

Η μονάδα αυτή δουλεύει λαμβάνοντας εντολές μέσω του πρωτοκόλλου I2C. Ο controller ενημερώνει το PCA9685 την τιμή PWM που θέλει να στείλει και το κανάλι με το οποίο θέλει να επικοινωνήσει. Ως αποτέλεσμα μπορούμε να ελέγχουμε ταυτόχρονα 16 διαφορετικούς σερβοκινητήρες με υψηλή ακρίβεια και πολύ μικρή απώλεια ενέργειας. Η διεύθυνση του είναι η 0x40 παρόλα αυτά μας δίνεται η δυνατότητα να την αλλάξουμε ενώνοντας ορισμένες επαφές πάνω στην μονάδα.



Σχήμα 23: Η μονάδα PCA9685

Τέλος, πρέπει να αναφέρουμε ότι το PCA9685 μετατρέπει απλώς τις εντολές που λαμβάνει σε PWM, δε δημιουργεί δικές του τάσεις οπότε χρειάζεται μια εξωτερική πηγή στα 5V η οποία είναι ικανή να προσφέρει όσο ρεύμα όσο χρειαστούν οι σερβοκινητήρες.

Για τον ρομποτικό βραχίονα θα κάνουμε χρήση 2 ειδών σερβοκινητήρων. Ο πρώτος είναι ο MG996R (Σχήμα 24), ένας σερβοκινητήρας ικανός να παράγει υψηλή ροπή με κόστος αυξημένες απαιτήσεις σε ρεύμα (πίνακας 2). Ο δεύτερος είναι ο MG90S (Σχήμα 25), ένας μικρότερος και πιο οικονομικός σερβοκινητήρας ο οποίος όμως είναι αρκετά πιο αδύναμος (πίνακας 3). Από κάτω παρέχονται εικόνες και τα τεχνικά τους χαρακτηριστικά.

Σχήμα 24: Ο σερβοκινητήρας MG996R

Σχήμα 25: Ο σερβοκινητήρας MG90S

Πίνακας 3: Τεχνικά χαρακτηριστικά του MG996R

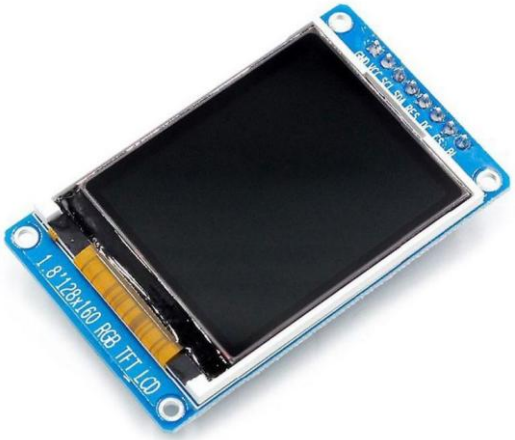
Stall Torque	11kg*cm @ 6V
Operating Voltage	4.8V – 7.2V
No load Current	170mA
Stall Current	1400mA
Max Speed	60 degrees in 0.15s
Weight	55g

Πίνακας 2: Τεχνικά χαρακτηριστικά του MG90S

Stall Torque	2.8kg*cm @ 6V
Operating Voltage	4.8V – 6V
No load Current	100mA
Stall Current	800mA
Max Speed	60 degrees in 0.08s
Weight	13.4g

2.5.3 Η οθόνη του ασύρματου τηλεχειριστηρίου

Το τηλεχειριστήριο που σχεδιάσαμε είναι εξοπλισμένο με μία οθόνη διαστάσεων 28 x 35 mm ή 1.8 ίντσες (Σχήμα 26). Κάνοντας χρήση του πρωτοκόλλου SPI εμφανίζουμε σε αυτή σημαντικές πληροφορίες για το όχημα, το είδος της κίνησης, την ρύθμιση ταχύτητας και την μπαταρία του οχήματος.



Σχήμα 26: Η οθόνη 1.8 ιντσών

2.6 Σύνοψη Κεφαλαίου

Στο παρόν κεφάλαιο μελετήσαμε όλο το απαραίτητο υλικό και λογισμικό το οποίο χρησιμοποιήθηκε για την εκπόνηση της παρούσας διπλωματικής εργασίας. Στο επόμενο κεφάλαιο θα αναλύσουμε πως το υλικό κομμάτι των 2 συστημάτων είναι δομημένο ώστε να έχουμε μια πλήρη εικόνα των κυκλωμάτων πριν αναλύσουμε το λογισμικό των μικροελεγκτών ESP32 S3 DevKit C1 και ESP32 S3 DevKit M1.

Κεφάλαιο 3. Το υλικό των συστημάτων

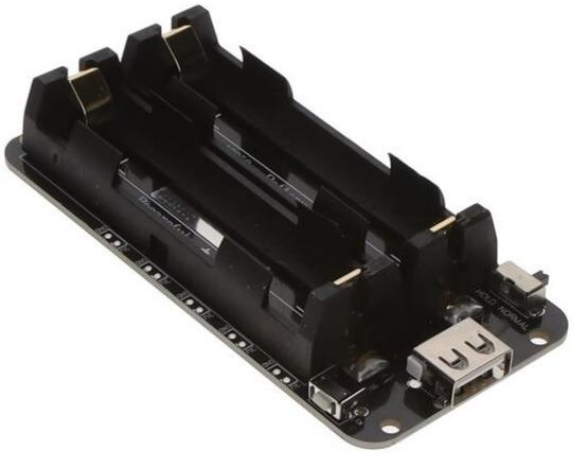
Σε αυτό το κεφάλαιο θα αναλύσουμε το hardware (υλικό) που αναπτύξαμε για τα 2 διακριτά συστήματα. Θα μελετήσουμε με λεπτομέρεια τις συνδεσμολογίες για την τροφοδοσία, την επικοινωνία των περιφερειακών, το PCB που αναπτύξαμε για το χειριστήριο, το σύστημα αποσύνδεσης της μπαταρίας από το όχημα για την προστασία των κυκλωμάτων όπως επίσης και τα μοντέλα που σχεδιάσαμε και εκτυπώσατε τρισδιάστατα.

3.1 Η δομή του ασύρματου χειριστηρίου

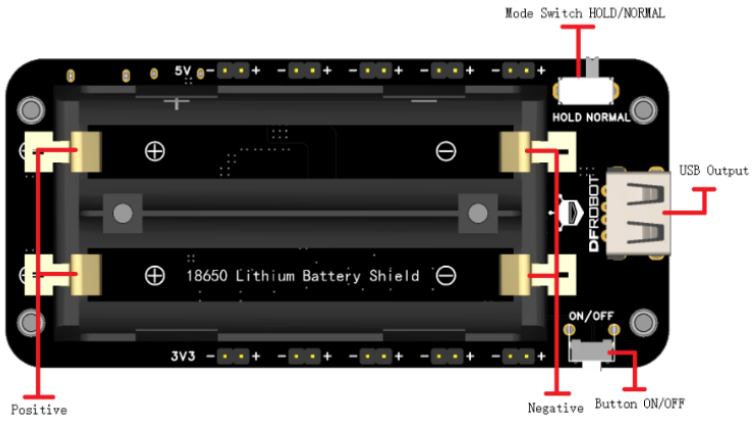
Το χειριστήριο απαρτίζεται από 2 μέρη, το PCB μαζί με τα ηλεκτρονικά που συνδέεται και το εξωτερικό κουτί που κρατάει τα πάντα στην θέση τους. Θα ξεκινήσουμε με την ανάλυση του κυκλώματος.

3.1.1 Το κύκλωμα του χειριστηρίου

Στόχος του κυκλώματος είναι ο συνδυασμός ενός Esp32 S3 Devkit C1, 12 κουμπιών για τον έλεγχο του ρομποτικού βραχίονα, μίας οθόνης για την εκτύπωση σημαντικών πληροφοριών, ενός MPU6050 και μιας μονάδας joystick για την πλοήγηση του οχήματος.

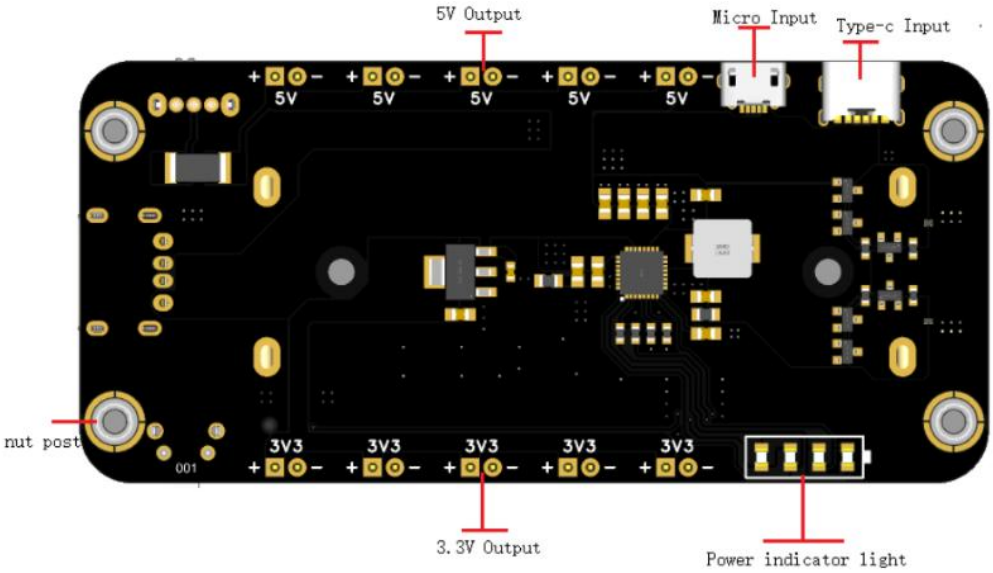


Σχήμα 27: Μονάδα μπαταριών



Σχήμα 28: Η μπροστά όψη της μονάδας μπαταριών

Όλο το κύκλωμα τροφοδοτείται από μια μονάδα (Σχήματα 27 & 28) με υποδοχές για 2 επαναφορτιζόμενες μπαταρίες τύπου 18650 η οποία παρέχει δυνατότητες επαναφόρτισης των μπαταριών μέσω καλωδίου USB. Προσφέρει εξόδους σε 2 τάσεις, 3.3V και 5V, ιδανικές για την λειτουργία όλων των ηλεκτρονικών εξαρτημάτων πάνω στο χειριστήριο. Η παροχή ξεκινάει όταν η μονάδα ενεργοποιηθεί, κάτι το οποίο επιτυγχάνεται με το πάτημα ενός διακόπτη πάνω της. Η μονάδα αυτή εμπεριέχει επίσης έναν μηχανισμό αυτόματης απενεργοποίησης όταν βρίσκεται στην λειτουργία Normal (Σχήμα 29) και το ρεύμα που παρέχει είναι οριακά μηδενικό. Στην λειτουργία Hold η συσκευή θα σβήσει μόνο με τον κανονικό τρόπο ο οποίος είναι με διπλό πάτημα του διακόπτη ενεργοποίησης. Για την φόρτιση του κυκλώματος χρησιμοποιούμε τις υποδοχές USB που βρίσκονται στο πίσω μέρος της μονάδας.



Σχήμα 29: Η πίσω όψη της μονάδας μπαταριών

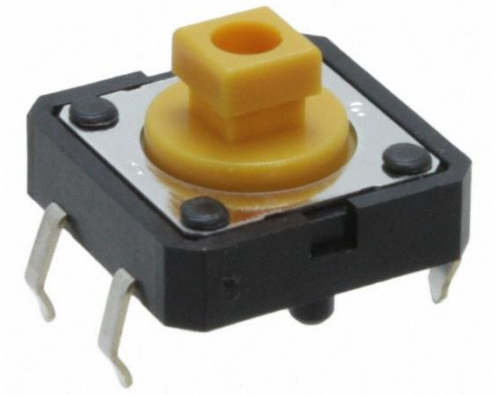
Για να κάνουμε προσβάσιμο το κουμπί ενεργοποίησης από το εξωτερικό του κουτιού κολλήσαμε στα άκρα του δύο καλώδια και στο άλλο άκρο έναν εξωτερικό διακόπτη (Σχήμα 30), με αυτό τον τρόπο μπορούμε να προσωμοιώσουμε το πάτημα του διακόπτη από το εξωτερικό της μονάδας.

Σχήμα 30: Ο εξωτερικός διακόπτης

Το ESP32 S3 Devkit C1 τροφοδοτείται με 2 τρόπους, ο πρώτος είναι μέσω των διεπαφών USB και UART που συναντάμε πάνω στο ενσωματωμένο ενώ ο δεύτερος είναι η

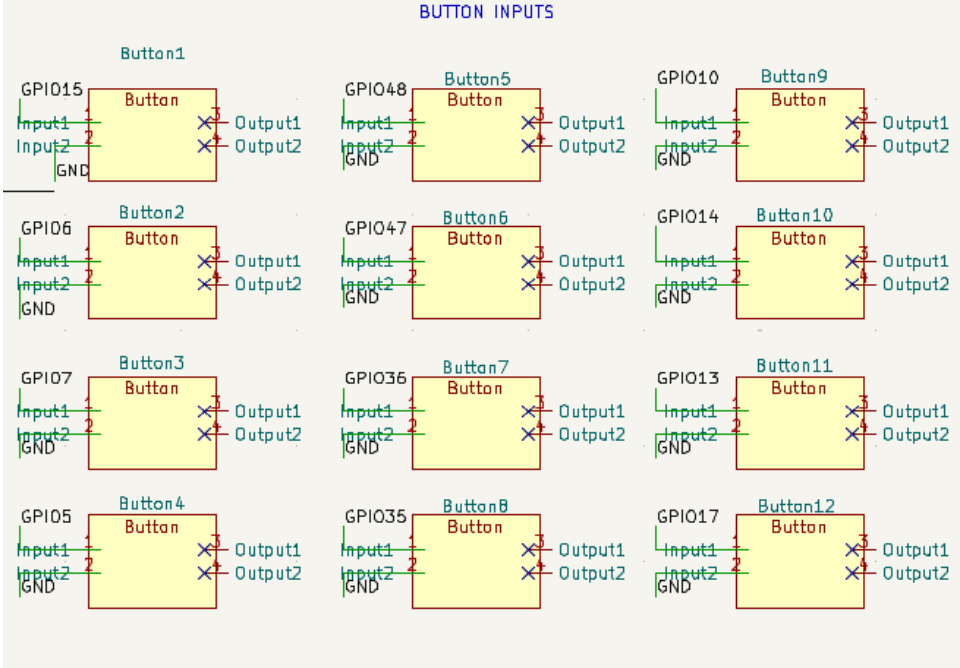
παροχή τάσης στην επαφή των 5V. Κάνοντας χρήση της εξόδου 5V της μονάδας των μπαταριών ενώνουμε με ένα καλώδιο την έξοδο 5V με την είσοδο του ESP32 S3 Devkit C1 και με ένα άλλο ενώνουμε τις γειώσεις μεταξύ τους. Επιτυγχάνουμε έτσι την τροφοδοσία του ενσωματωμένου χωρίς την χρήση των διεπαφών USB και UART. Το ESP32 S3 Devkit C1 βρίσκεται στερεομένο πάνω σε ένα PCB, το οποίο και αποτελεί το κύριο κύκλωμα του ασύρματου χειριστηρίου. Για τον σχεδιασμό και την δημιουργία του PCB χρησιμοποιήσαμε το πρόγραμμα Kicad που αναλύσαμε στο κεφάλαιο 2. Το κύκλωμα το οποίο κατασκευάστηκε αποτελείται από το ακόλουθα μέρη:

Απτικά Κουμπιά: Στο σύνολο το κύκλωμα μας περιέχει 12 απτικά κουμπιά (Σχήμα 31) SPST τα οποία θα χρησιμοποιήσουμε για τον έλεγχο του ρομποτικού βραχίονα. Το καθένα έχει το ένα άκρο του συνδεδεμένο σε μία έξοδο του ESP32 S3 Devkit C1 και το άλλο άκρο του συνδεδεμένο με μία γείωση. Όταν πατάμε τον διακόπτη συνδέουμε την έξοδο του ESP32 S3 με την γείωση, συνεπώς μπορούμε να ανιχνεύσουμε μέσω κώδικα ότι το κουμπί πιέστηκε. Στο πρόγραμμα Kicad έχουμε συνδέσει το σύμβολο του κάθε διακόπτη στο 1 άκρο με μια έξοδο του ESP32 (GPIOXX) και ένα άλλο με την γείωση (GND) (Σχήμα 33). Το πρόγραμμα μας δίνει την δυνατότητα να προσθέσουμε ετικέτες σε καλώδια που θέλουμε να θεωρηθεί ότι συνδέονται μεταξύ τους, χωρίς να πρέπει να τα ενώσουμε, επιτυγχάνοντας έτσι ένα πιο ευανάγνωστο κύκλωμα. Στο πραγματικό κύκλωμα χρησιμοποιήσαμε τους διακόπτες B3F-4055 με διαστάσεις 12 x 12 x 7.3mm και για λόγους αισθητικής προσθέσαμε χρωματιστά καπάκια (Σχήμα 32) καθώς σκοπεύουμε να χρησιμοποιήσουμε τους διακόπτες αυτούς σε ζευγάρια.



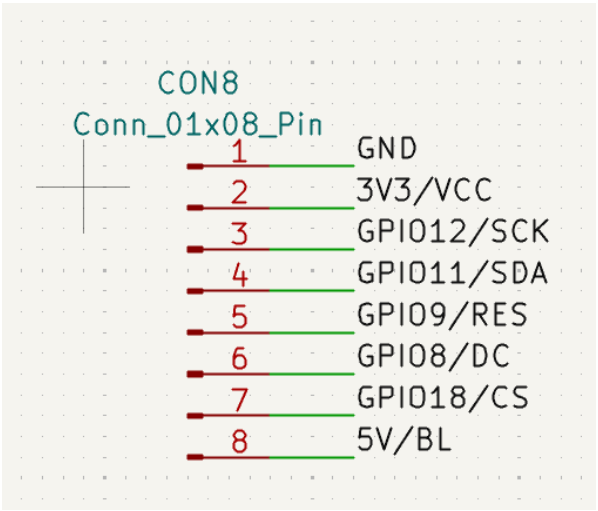
Σχήμα 31: Ο διακόπτης B3F-4055

Σχήμα 32: Καπάκι του διακόπτη σε κόκκινο



Σχήμα 33: Το σύνολο των διακοπτών στο πρόγραμμα Kicad

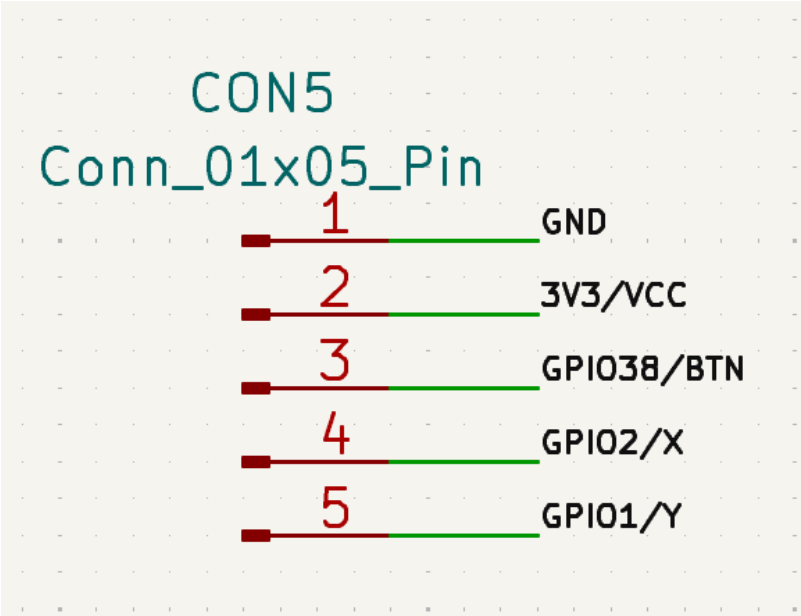
Οθόνη: Όπως αναφέραμε στο δεύτερο κεφάλαιο το χειριστήριο είναι εξοπλισμένο με μία οθόνη 1.8 ιντσών. Η σύνδεση της με το κύκλωμα μας απαιτεί την χρήση 8 καλωδίων, 1 για την παροχή τάσης (3V3/VCC), 1 για την γείωση (GND), 1 για τον φωτισμό της οθόνης (BL), 3 (SCK, SDA,CS) για το πρωτόκολλο SPI με το οποίο θα επικοινωνεί η οθόνη με το ESP32 S3, 1 για να μπορούμε να κάνουμε επανεκκίνηση την οθόνη (RES) και 1 έξτρα για να μπορούμε να ενημερώνουμε την οθόνη αν οι πληροφορίες που στέλνουμε αποτελούν εντολή ή δεδομένα προς εκτύπωση (DC). Το τελικό αποτέλεσμα είναι εμφανές στο σχήμα 34:



Σχήμα 34: Η συνδεσμολογία της οθόνης στο πρόγραμμα Kicad

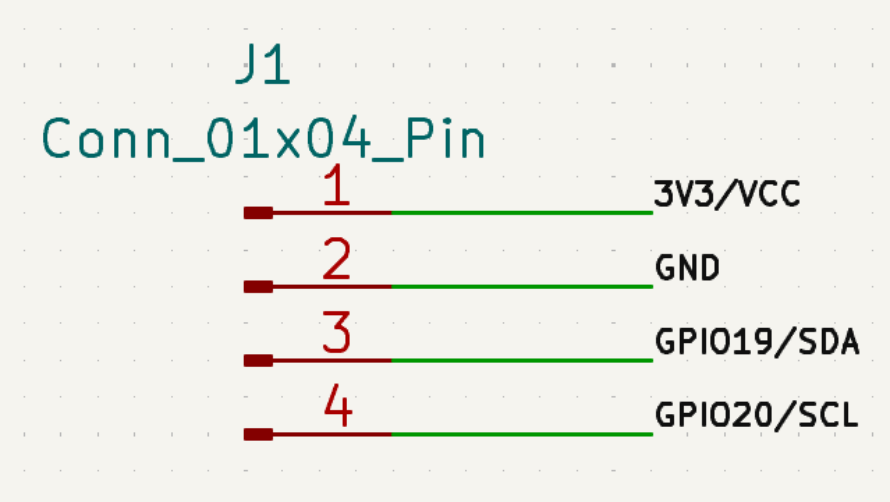
Μονάδα Joystick: Για την πλοήγηση του οχήματος θα χρησιμοποιήσουμε στο κύκλωμα μας μια μονάδα με 2 μεταβλητές αντιστάσεις γνωστή ως joystick ή μοχλός αντίχειρα. Επιλεξαμε να χρησιμοποιήσουμε την μονάδα joystick της εταιρείας Keyestudio (Σχήμα 35) λόγω της ποιότητας κατασκευής αλλά και του χαμηλού κόστους. Οι 2 μεταβλητές αντιστάσεις μας προσφέρουν μετρήσεις στους άξονες x και y τις οποίες μπορούμε να μετατρέψουμε σε κατεύθυνση προς την οποία θέλουμε να οδηγήσουμε το όχημα. Επίσης η συγκεκριμένη μονάδα παρέχει ενσωματωμένο κουμπί το οποίο μπορούμε να αξιοποιήσουμε μέσω κώδικα. Για την σύνδεση του απαιτούνται 5 καλώδια, 1 για την παροχή ρεύματος (3V3/VCC), 1 για την γείωση (GND), 1 για την ανίχνευση του κουμπιού (BTN) και 2 για τις μεταβλητές αντιστάσεις (X & Y) (Σχήμα 36). Πρέπει να προσέχούμε όταν συνδέσουμε τις μεταβλητές αντιστάσεις στο ESP32 S3 να επιλέξουμε εισόδους οι οποίες έχουν την δυνατότητα να μετατρέπουν αναλογική είσοδο σε ψηφιακή τιμή (ADC ή analog to digital converter).

Σχήμα 35: Η μονάδα joystick της Keyestudio



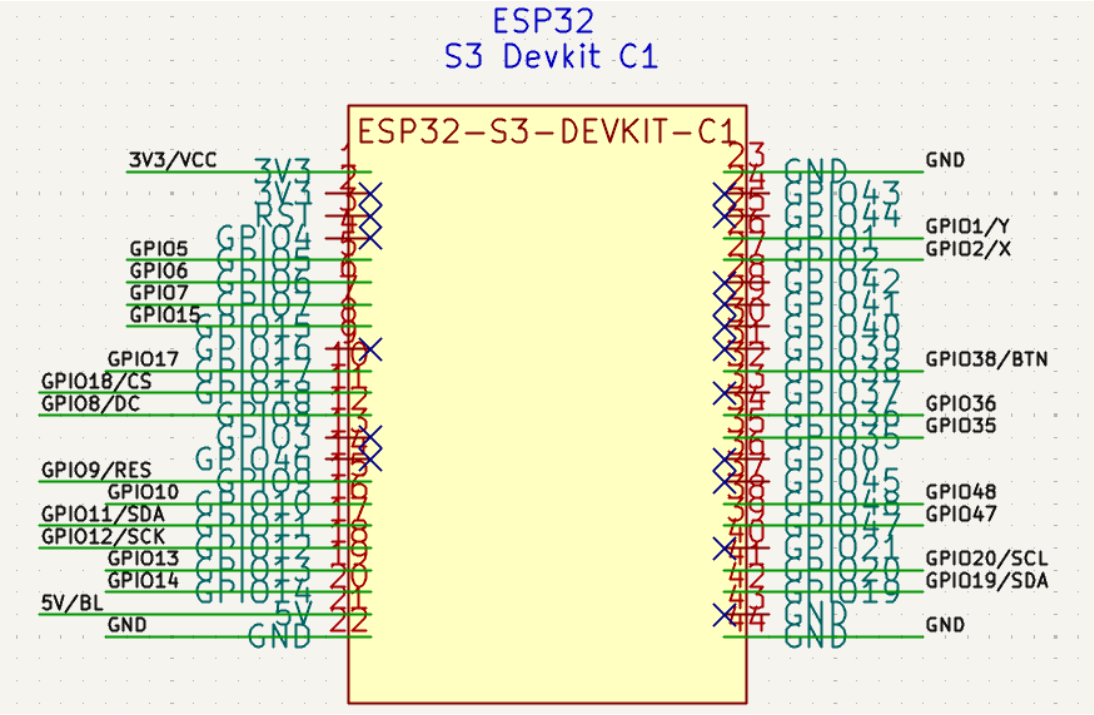
Σχήμα 36: Η συνδεσμολογία της μονάδας joystick

Μονάδα GY-521: Για τον υπολογισμό της περιστροφής του χειριστηριού σε σχέση με τον κάθετο άξονα όπως αναφέραμε στο κεφάλαιο 2 θα κάνουμε χρήση της μονάδας GY-521 η οποία εμπεριέχει το IMU MPU6050. Η σύνδεση αυτού με το ESP32 S3 απαιτεί 4 καλώδια καθώς οι 2 μονάδες επικοινωνούν μέσω του πρωτοκόλλου I2C. Το πρώτο καλώδιο θα χρησιμοποιηθεί για την παροχή τάσης (VCC), το δεύτερο για την γείωση (GND) ενώ θα χρειαστούμε 2 καλώδια ακόμα για τα σήματα SCL και SDA (Σχήμα 37).



Σχήμα 37: Η συνδεσμολογία του GY-521

Συνδέοντας όλες τις επαφές με το ESP32 S3 Devkit C1 και βάζοντας τις σωστές ετικέτες σε κάθε καλώδιο έχουμε την τελική συνδεσμολογία του χειριστηρίου η οποία απεικονίζεται παρακάτω στο σχήμα 38.

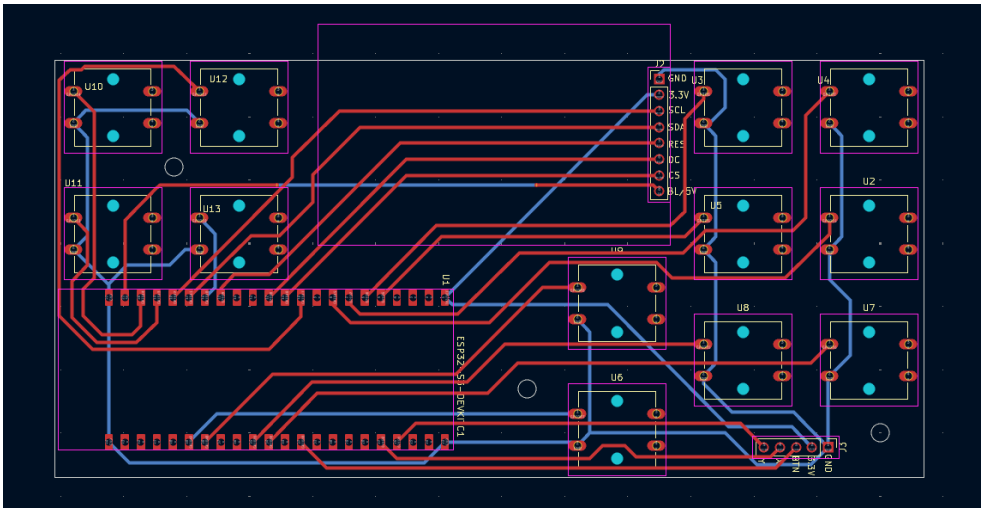


Σχήμα 38: Η τελική συνδεσμολογία ESP32 S3 DevKit C1

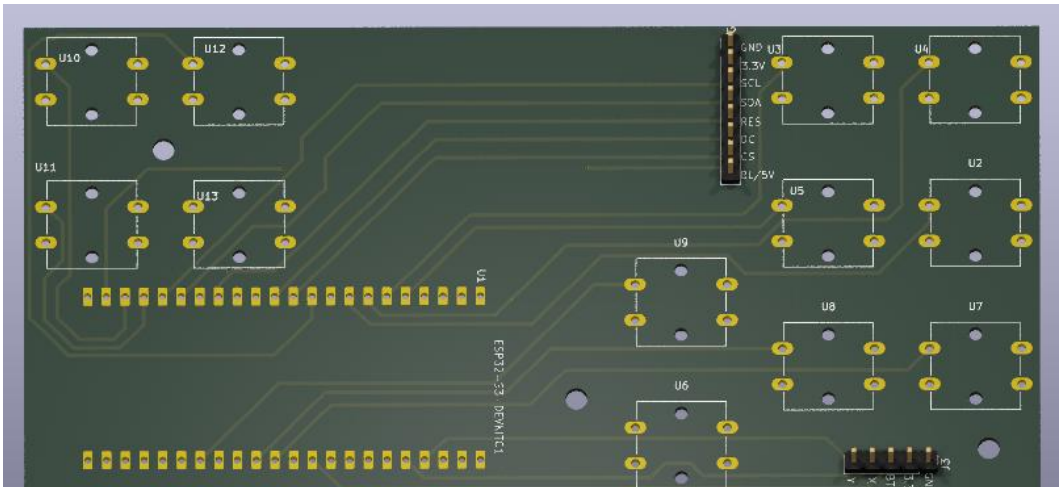
Στην δημιουργία του PCB επιλέξαμε για λόγους απλότητας αλλά και χώρου να βάλουμε μόνο την οθόνη τα κουμπιά και μια υποδοχή με βίδες για 5 καλώδια τα οποία θα χρησιμοποιηθούν για την σύνδεση του PCB με την μονάδα joystick (Σχήμα 39).

Σχήμα 39: Υποδοχή 5 καλωδίων με βίδες

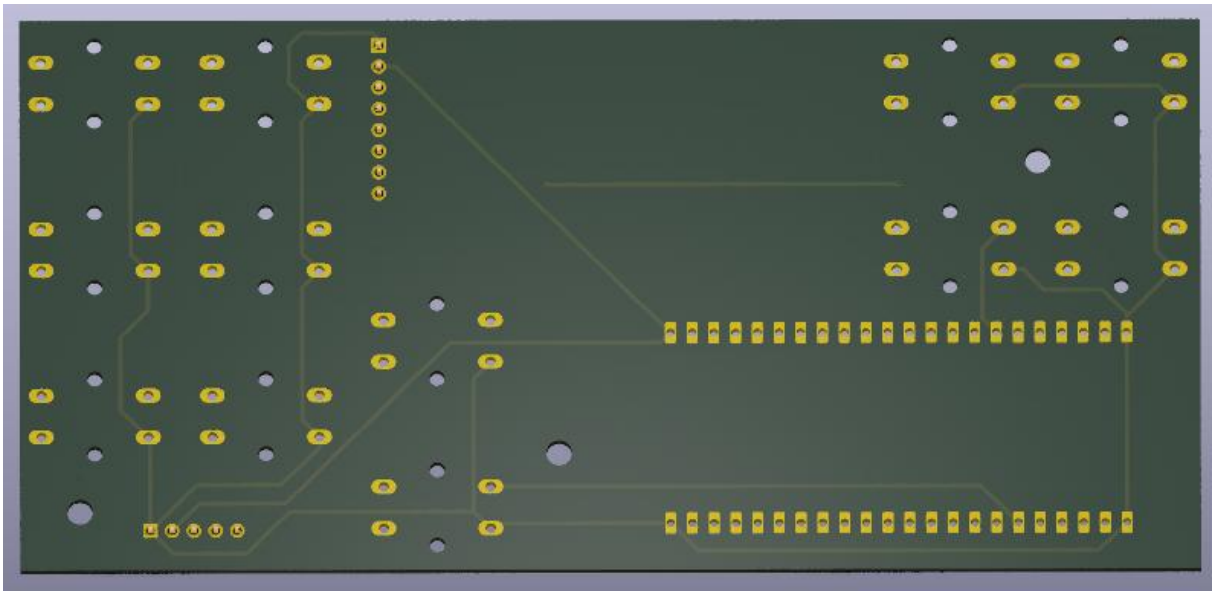
Οι υπόλοιπες συνδέσεις θα γίνουν με συγκόλληση των καλωδίων. Έτσι καταλήγουμε στο τελικό PCB το οποίο αποτελεί τον πυρήνα του ασύρματου χειριστηρίου (Σχήμα 40, 41 & 42).



Σχήμα 40: Το τελικό PCB του ασύρματου χειριστηρίου

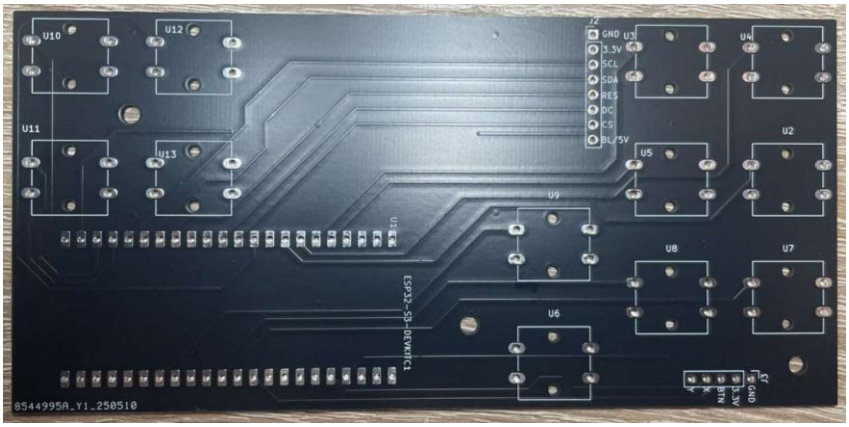


Σχήμα 41: 3D render του PCB του χειριστηρίου (μπροστά)

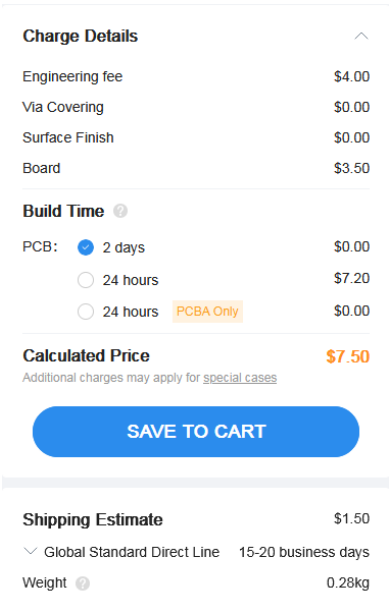


Σχήμα 42: 3D render του PCB του χειριστηρίου (πίσω)

Προσθέσαμε επίσης 3 τρύπες διαμέτρου 3mm πάνω στο PCB ώστε αργότερα να βιδωθεί μέσα στο κουτί το οποίο θα μπει. Έπειτα από την δημιουργία του τελικού PCB μπορούμε να εξάγουμε το αρχείο και να το στείλουμε σε κάποιον παραγωγό PCB ώστε να κατασκευαστεί. Υπάρχουν πολλές επιλογές, έπειτα από έρευνα όμως επιλέξαμε τον κατασκευαστή JLCPCB για την εγγυήση ποιότητας του αλλά και για τις προσιτές τιμές του καθώς το κόστος 5 αντίγραφων του PCB ανέρχεται στα 7.50€ (Σχήμα 43).



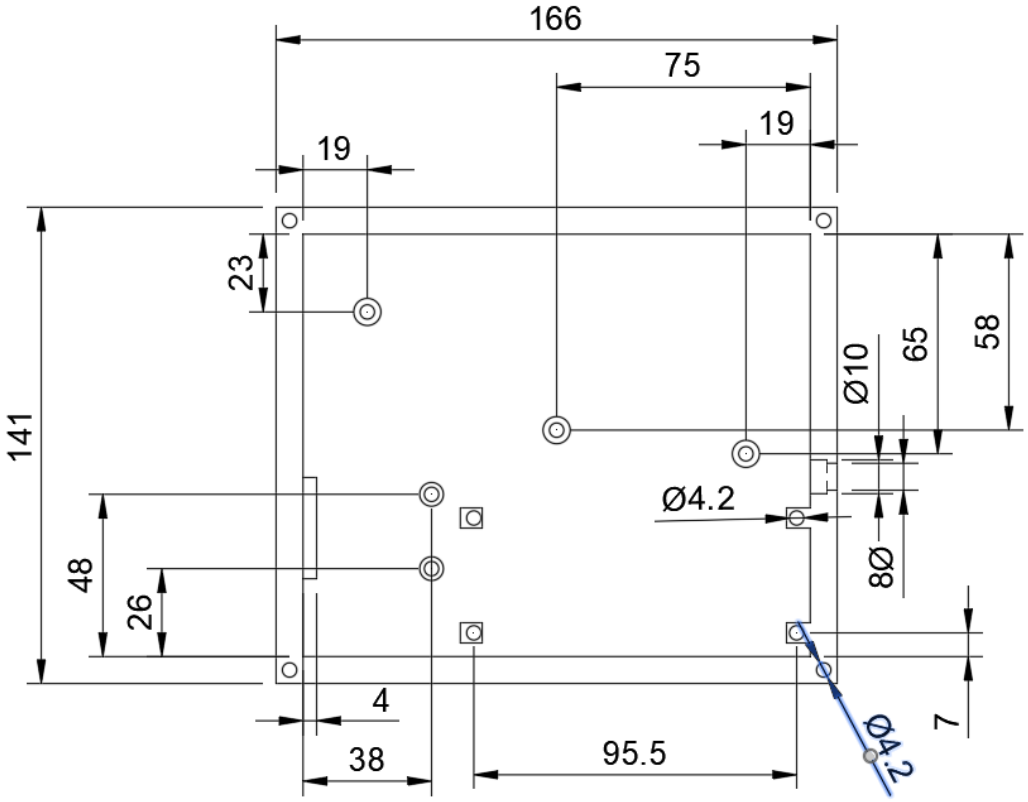
Σχήμα 43: Το PCB του χειριστηρίου



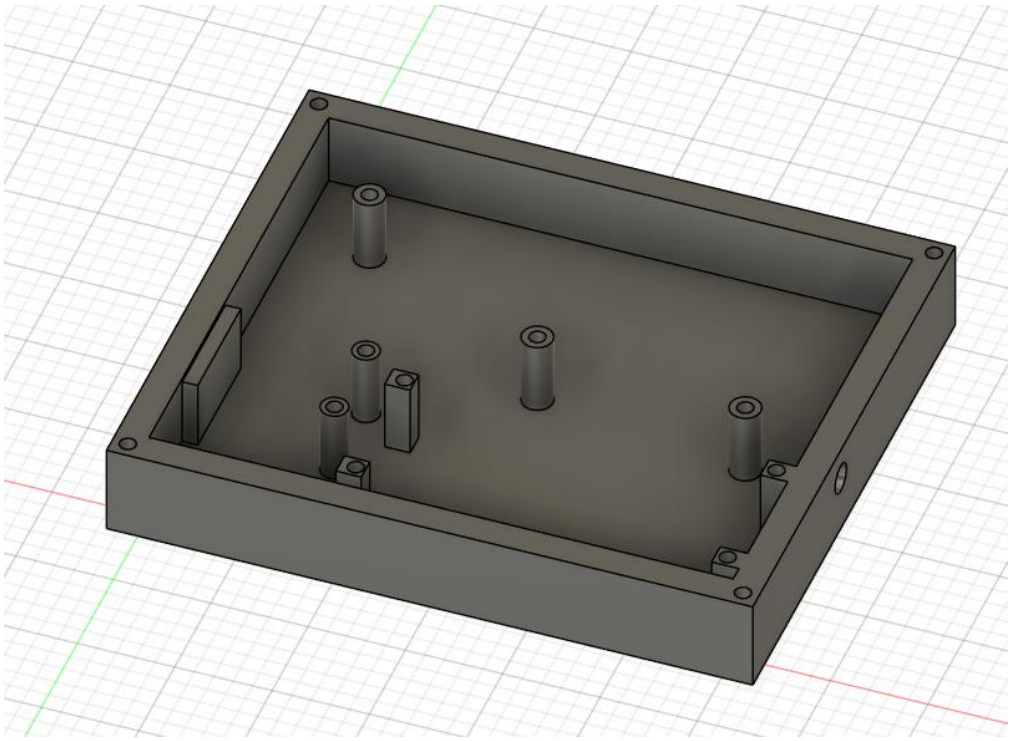
Σχήμα 44: Υπολογισμός κόστους παραγωγής του PCB

3.1.2 Το κουτί του ασύρματου χειριστηρίου

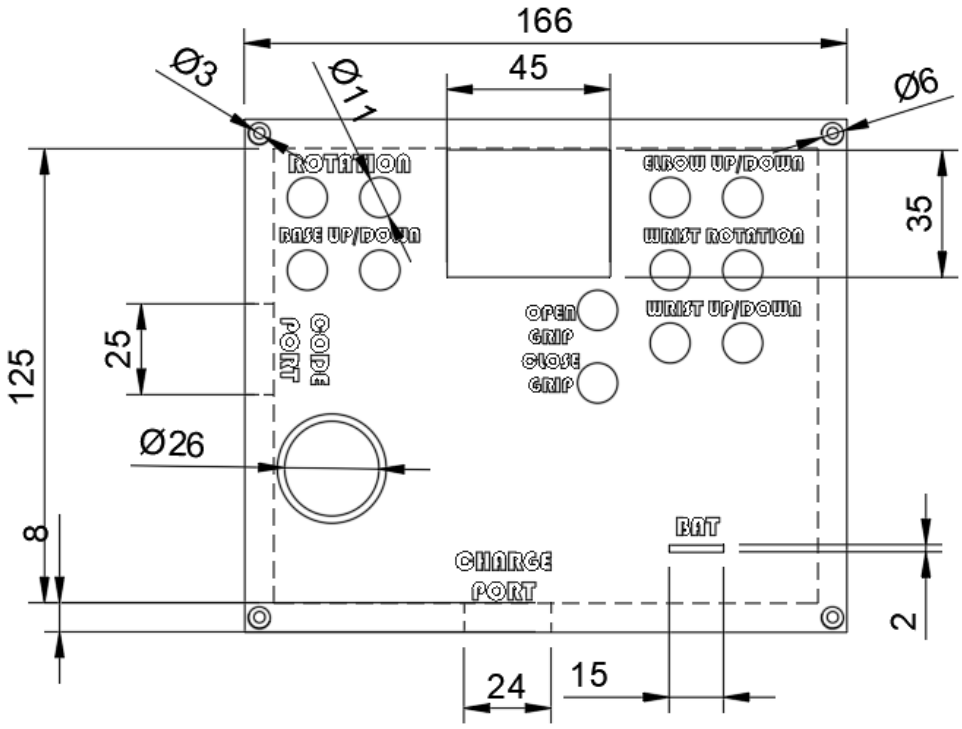
Με στόχο να προστατέψουμε το κύκλωμα θα σχεδιάζουμε ένα κουτί το οποίο θα εκθέτει μόνο τα απαραίτητα κομμάτια όπως τα κουμπιά, την οθόνη, το joystick τις θύρες φόρτισης και τον διακόπτη που ανάβει και κλείνει το κύκλωμα. Για την δημιουργία του επιλέξαμε το λογισμικό Fusion 360 της εταιρείας Autodesk το οποίο διανέμεται δωρεάν από το πανεπιστήμιο. Σχεδιάσαμε 2 κομμάτια τα οποία θα κλείνουν μεταξύ τους με βίδες 3mm. Στο κάτω κομμάτι σχεδιάσαμε υποδοχές για βίδες ώστε να κρατήσουμε όλα τα κομμάτια ακίνητα ενώ στο πάνω σχεδιάσαμε ειδικές τρύπες ώστε να προεξέχουν μόνο τα κομμάτια με τα οποία αλληλεπιδράει ο χρήστης. Από κάτω παρέχονται τα μηχανολογικά σχέδια των κομματιών και η τρισδιάστατη αναπαράσταση τους στα σχήματα 45, 46, 47, 48, 49 & 50.



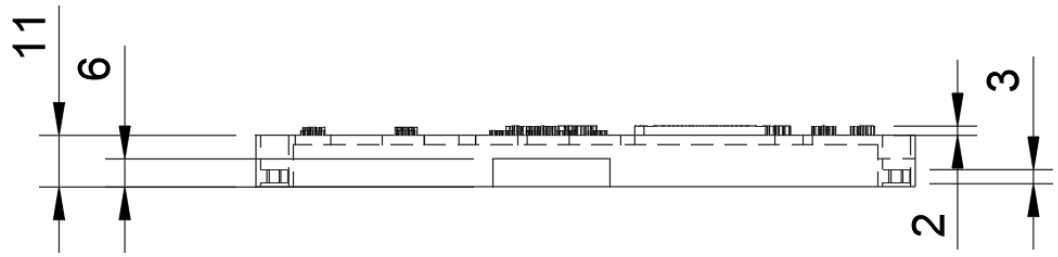
Σχήμα 45: Η κάτοψη του κάτω μέρους του κουτιού



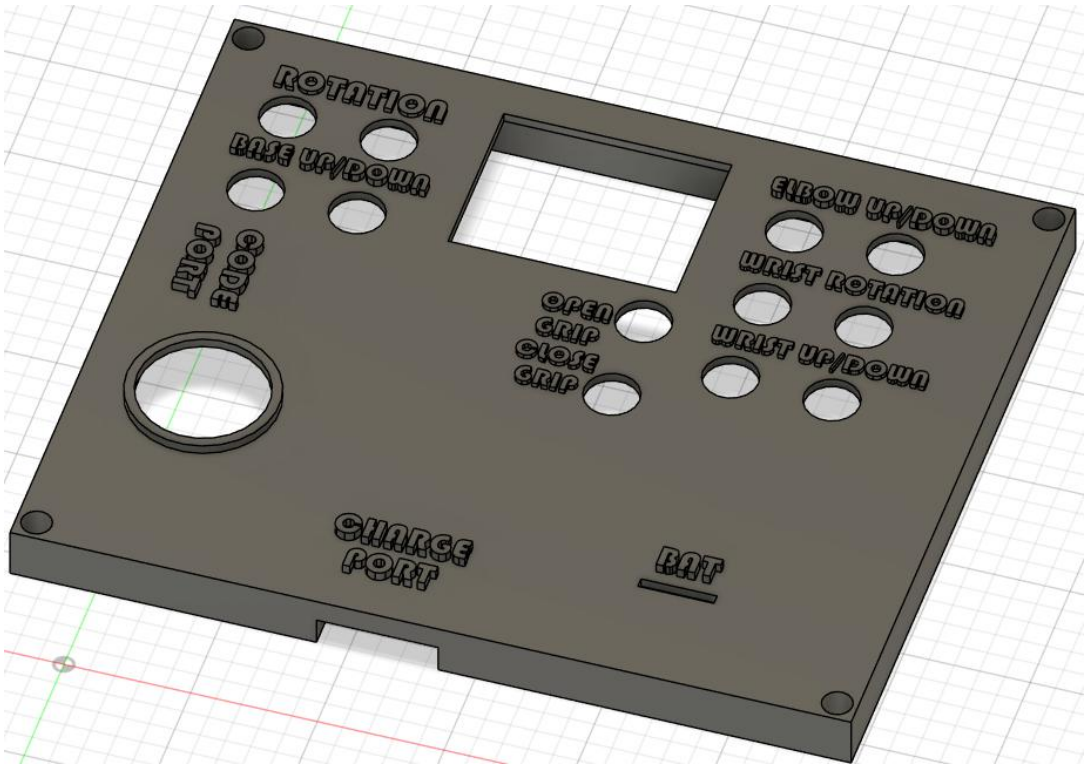
Σχήμα 46: Τρισδιάστατή Όψη του κάτω κουτιού



Σχήμα 48: Η κάτοψη του πάνω μέρους του κουτιού

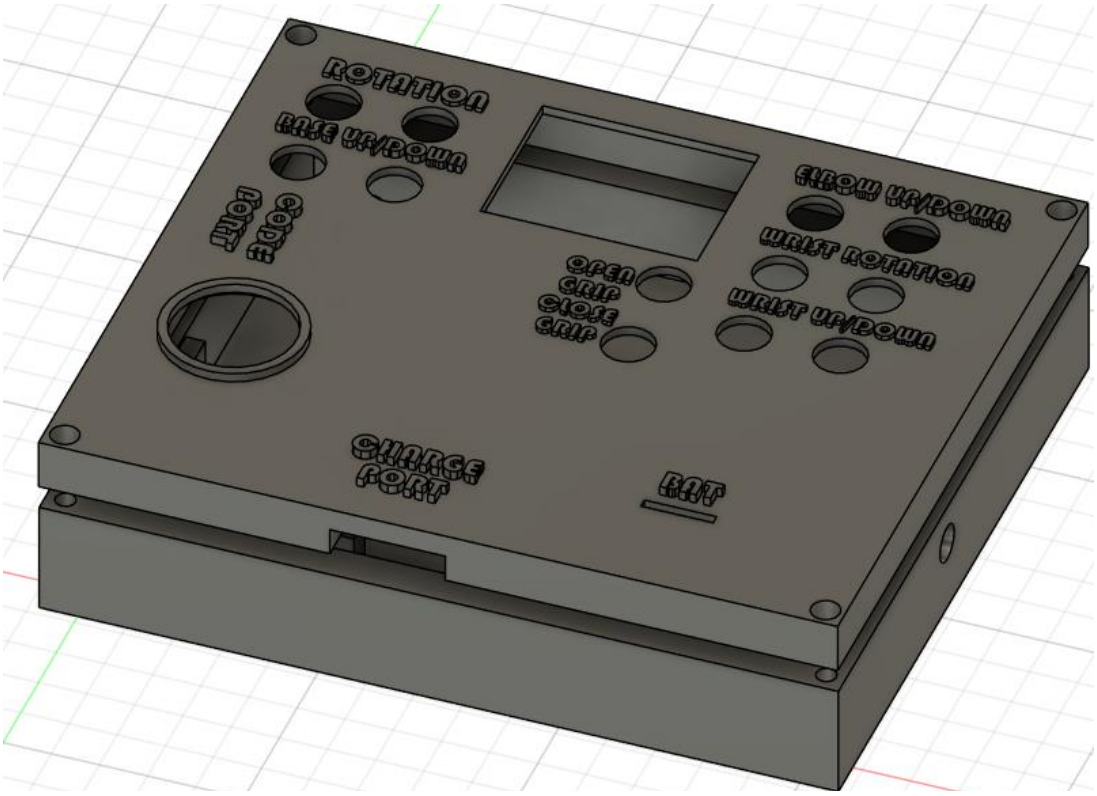


Σχήμα 50: Η πλάγια όψη του πάνω μέρους του κουτιού

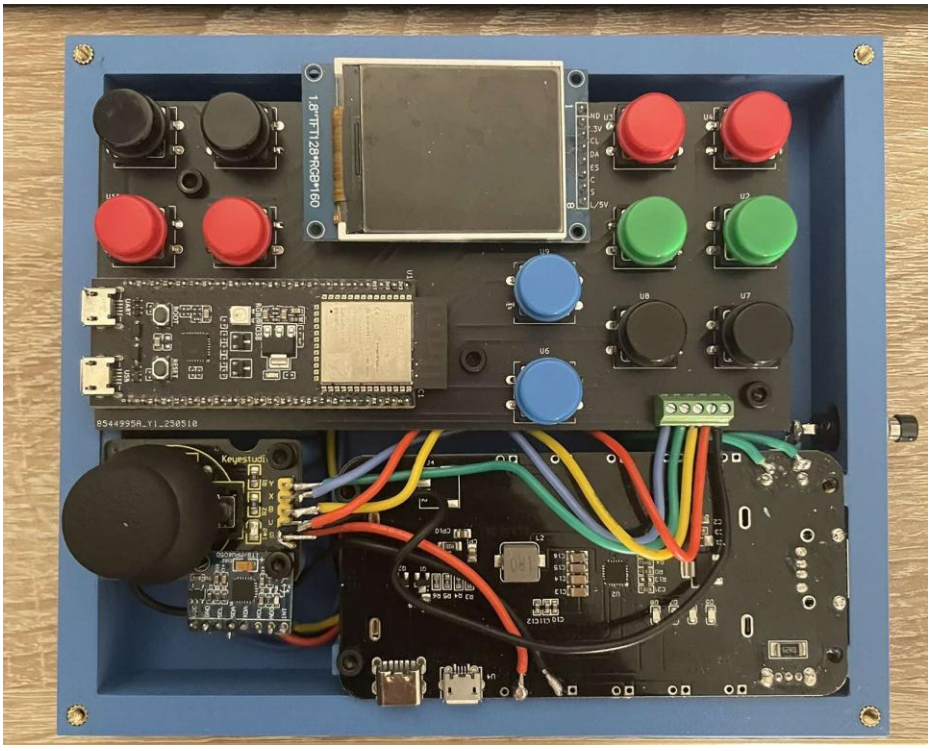


Σχήμα 49: Τρισδιάστατη αναπαράσταση του πάνω μέρους του κουτιού του χειριστηρίου

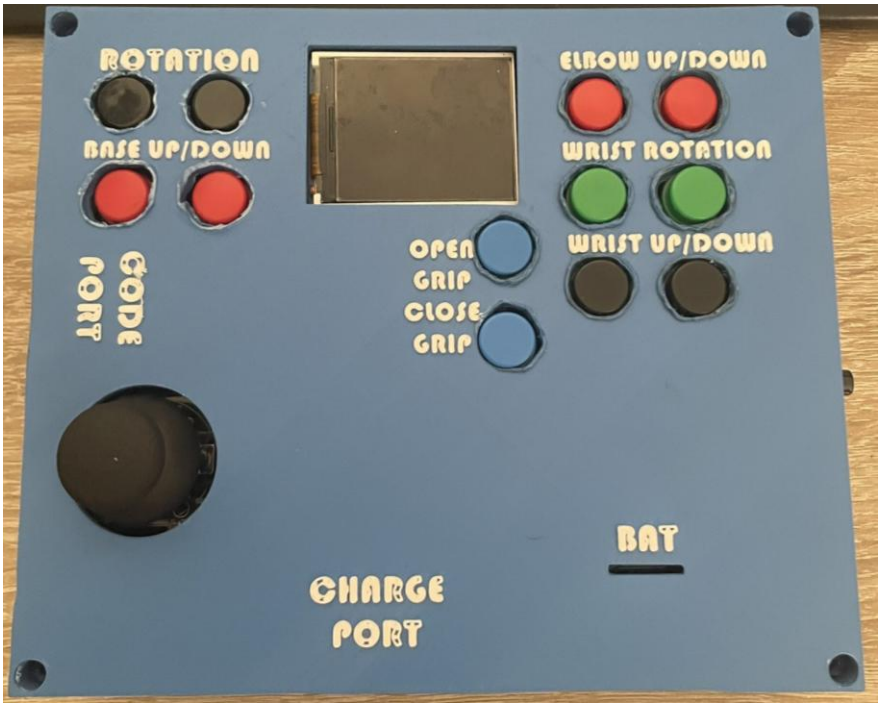
Ενώνοντας τα δύο κομμάτια δημιουργούμε ένα κουτί το οποίο προστατεύει τα ηλεκτρονικά εξαρτήματα χωρίς να περιορίζει καμία λειτουργία από τον χρήστη (Σχήμα 51).



Σχήμα 52: Τρισδιάστατη αναπαράσταση του κουτιού του χειριστηρίου



Σχήμα 51: Η κάτοψη της τελικής δομής του χειριστηρίου χωρίς το πάνω μέρος



Σχήμα 53: Η κάτοψη της τελικής δομής του χειριστηρίου με το πάνω μέρος

3.2 Το υλικό του οχήματος

Η δομή του οχήματος χωρίζεται σε 3 μέρη, τον ρομποτικό βραχίονα ο οποίος στην βάση του θα κρατάει τον εγκέφαλο του οχήματος, το όχημα με την πλακέτα ελέγχου και το σύστημα αποσύνδεσης της μπαταρίας μαζί με την μπαταρία.

3.2.1 Η μπαταρία του οχήματος και το κύκλωμα αποσύνδεσης της

Για την τροφοδοσία του οχήματος επιλέξαμε να χρησιμοποιήσουμε μια μπαταρία τύπου LiPo (Πολυμερούς Λιθίου) με μέση τάση εξόδου τα 14.8V (4S) (Σχήμα 54).
Το μέγεθος της μπαταρίας είναι 1800mAh και το C rating είναι 100, αυτό σημαίνει ότι το μέγιστο στιγμιαίο ρεύμα που μπορεί να παρέχει με ασφάλεια είναι 180A (1.8A * 100).

Σχήμα 54: Η μπαταρία του οχήματος

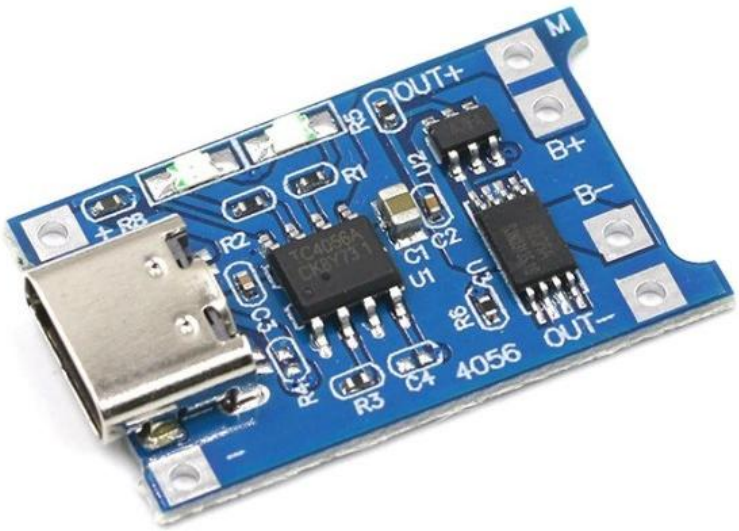
Για να μπορούμε να αποσυνδέσουμε την μπαταρία μας από το κύκλωμα με ασφάλεια θα κάνουμε χρήση ενός ρελέ σταθερής κατάστασης (SSR – Solid State Relay). Τα κριτήρια επιλογής του ρελέ είναι η δυνατότητα να αντέχει 14.8V συνεχούς τάσης και τουλάχιστον 40A όπου είναι το μέγιστο ρεύμα που μπορούν να τραβήξουν οι κινητήρες του οχήματος συνολικά.

Σχήμα 56: Το ρελέ του κυκλώματος ασφάλειας

Εμείς κάναμε χρήση του ρελέ SSR-40 DD της εταιρείας FOTEK[23] (Σχήμα 55). Σύμφωνα με τα χαρακτηριστικά που μας παρέχει η εταιρεία το συγκεκριμένο ρελέ αντέχει τάση εισόδου 5 έως 60V DC και 40A αποτελεί έτσι μια ιδανική επιλογή για το κύκλωμα μας. Το σήμα ελέγχου πρέπει να κυμαίνεται από τα 3V έως και τα 32V για αυτό κάναμε χρήση μιας δεύτερης μικρότερης μπαταρία τύπου Li ion στα 3.7V (Σχήμα 58). Για να μπορούμε να ελέγχουμε πότε θέλουμε το ρελέ να άγει ρεύμα και πότε όχι θα συνδέσουμε αυτή την μπαταρία με ένα μπουτόν τύπου μανιτάρι (Emergency Stop) (Σχήμα 56). Το μπουτόν αυτό έχει 2 εισόδους και 2 εξόδους τάσης. Η πρώτη ονομάζεται NO (Normally Open) και άγει ρεύμα μόνο όταν το κουμπί είναι πατημένο. Η δεύτερη ονομάζεται NC (Normally Closed) και άγει ρεύμα όταν το μπουτόν δεν

Σχήμα 55: Το μπουτόν τύπου μανιτάρι e-stop

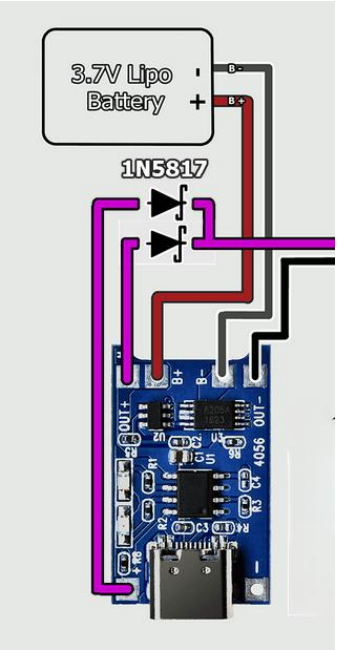
είναι πατημένο. Εφόσον θέλουμε το ρελέ να άγει ρεύμα όταν το κουμπί δεν είναι πατημένο,επιλέξαμε να συνδέσουμε την τάση της μικρής μπαταρίας στην NC είσοδο του μπουτόν. Συνεπώς όταν πατάμε το μπουτόν το σήμα ελέγχου στο ρελέ κόβεται και η μπαταρία αποσυνδέεται από το κύκλωμα. Μπορούμε να συνδέσουμε τον αρνητικό πόλο της μπαταρίας απευθείας πάνω στην πλακέτα ελέγχου του οχήματος, ενώ τον θετικό θα το συνδέσουμε πρώτα στην είσοδο υψηλής τάσης του ρελέ και έπειτα την έξοδο θα την συνδέσουμε στο θετικό πόλο της πλακέτας ελέγχου. Για την επαναφόρτιση της μικρής Li ion μπαταρίας θα κάνουμε χρήση της μονάδας TP4056[24] (Σχήμα 57) η οποία προσφέρει προστασία από υπερφόρτιση και υπεραποφόρτιση της μπαταρίας. Η φόρτιση γίνεται με ένα καλώδιο USB τύπου C ενώ οι φωτεινές ενδείξεις πάνω στην μονάδα μας δείχνουν πότε η μπαταρία έχει γεμίσει. Η μονάδα παρέχει 2 εισόδους, την B+ και την B- στην οποία συνδέουμε τον θετικό και τον αρνητικό πόλο της μπαταρίας μας, και τις OUT+ και OUT- όπου συνδέουμε το κύκλωμα που θέλουμε να τροφοδοτήσουμε. Στην προκειμένη περίπτωση θα συνδέσουμε το OUT+ με μία επαφή της NC εισόδου του μπουτόν ασφαλείας μας. Η έξοδος του μπουτόν οδηγείται στον θετικό πόλο του σήματος ελέγχου του ρελέ και το σήμα από τον αρνητικό πόλο επιστρέφει πίσω στην μονάδα φόρτισης, στην επαφή OUT-.



Σχήμα 57: Η μονάδα φόρτισης TP4056

Σχήμα 58: Η επαναφορτιζόμενη μπαταρία 18650 των 3.7V

Σύμφωνα με τις οδηγίες του κατασκευαστή η φόρτιση της μπαταρίας θα σταματήσει όταν η τάση της φτάσει κοντά στα 4.2V και το ρεύμα που τραβάει είναι κάτω των 100mA. Η χρήση της μπαταρίας κατά την φόρτιση δεν συνιστάται, για αυτό τον λόγο θα κάνουμε χρήση της πηγής για την παροχή ρεύματος στο κουμπί. Για να το επιτύγχουμε αυτό έγινε χρήση 2 διόδων schottky (χαμηλής πτώσης τάσης) με στόχο τον διαχωρισμό της εισόδου της μονάδας φόρτισης (IN+ και IN-) και της εξόδου της (OUT+ και OUT-). Όταν υπάρχει τάση στην είσοδο της μονάδας η δίοδος εισόδου άγει ρεύμα προς το μπουτόν ενώ έχουμε ανάστροφη τάση στην δίοδο της εξόδου.

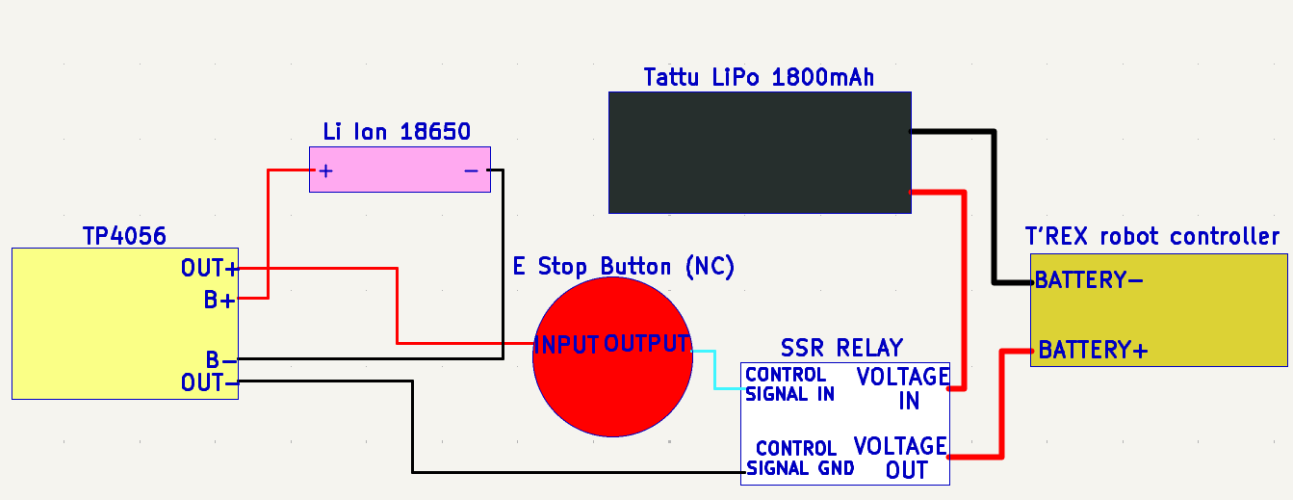


Σχήμα 59: Η συνδεσμολογία του κυκλώματος φόρτισης

Σχήμα 60: Η δίοδος Schottky 1N5817

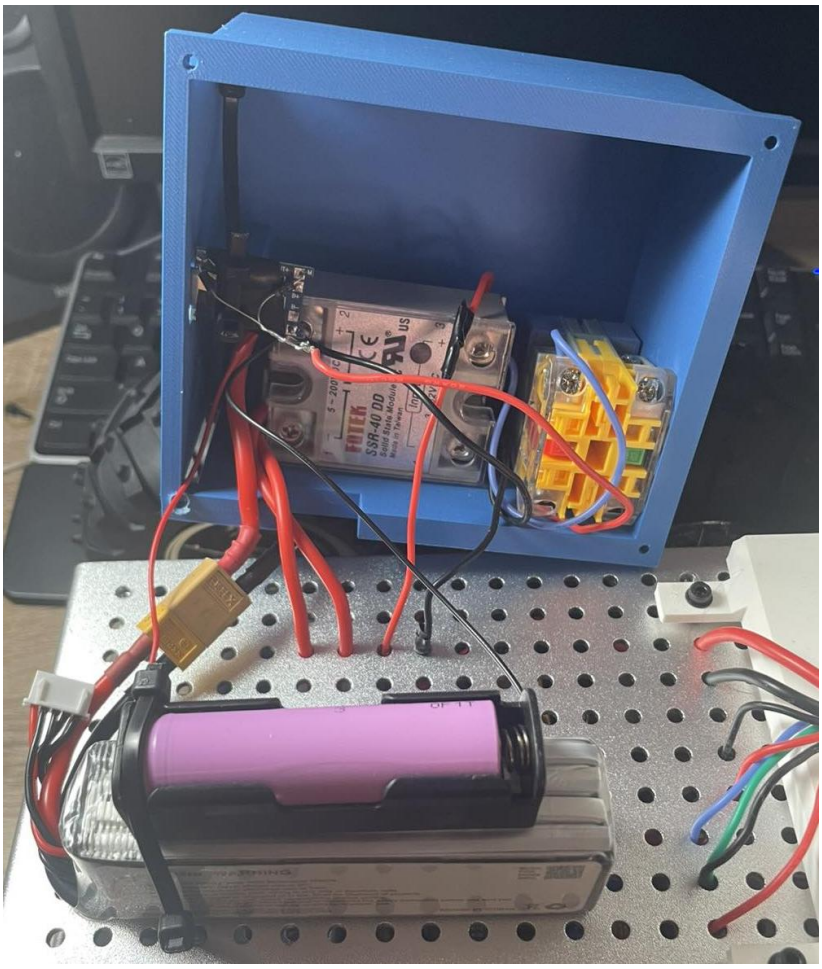
Ως αποτέλεσμα η πηγή παρέχει την τάση στο φορτίο μας. Από την άλλη όταν αποσυνδεθεί η πηγή τάσης από την είσοδο τότε η δίοδος εξόδου της μονάδας είναι αυτή που άγει ρεύμα και λειτουργεί το μπουτόν μας. Η δίοδος που επιλέξαμε είναι η 1N5817[25] καθώς καλύπτει τις ανάγκες μας και αντέχει την τάση που θέλουμε να εισάγουμε, παρόλα αυτά και δίοδοι όπως η 1N5818 και 1N5819 θα μπορούσαν να είχαν χρησιμοποιηθεί.

Παρατηρώντας τα χαρακτηριστικά της διόδου γνωρίζουμε ότι η πτώση τάσης που προκαλεί στην έξοδο είναι 0.45V ενώ η μέση τάση της μπαταρίας μας είναι 3.7V. Εφόσον το ρελέ μας χρειάζεται το ελάχιστο 3V στην είσοδο του σήματος ελέγχου συμπεραίνουμε ότι αυτή η πτώση δε θα επηρεάσει την λειτουργία του. Το τελικό κύκλωμα για την αποσύνδεση της μπαταρίας είναι το ακόλουθο (Σχήμα 61):



Σχήμα 61: Το κύκλωμα αποσύνδεσης της μπαταρίας

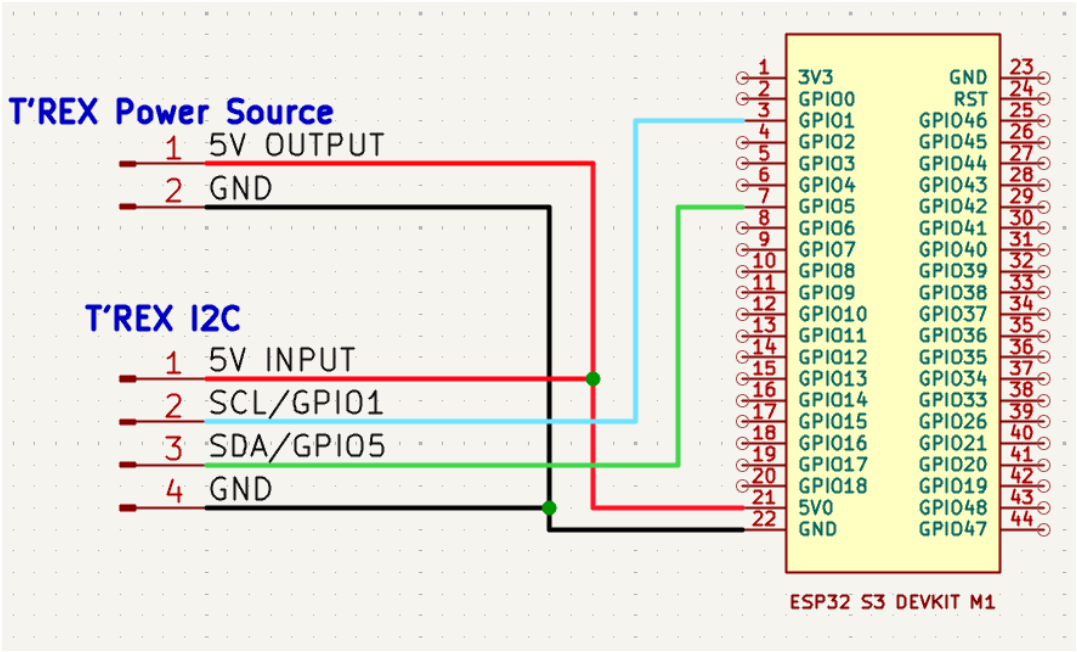
Για την προστασία του κυκλώματος εκτυπώσαμε τρισδιάστατα ένα τετράγωνο κάλυμμα (Σχήμα 62) ενώ ταυτόχρονα εκμεταλλευτήκαμε τις τρύπες του καλύμματος του οχήματος για την στήριξη του κουτιού και την διαχείριση των καλωδίων.



Σχήμα 62: Η εφαρμογή του κυκλώματος πάνω στο όχημα

3.2.2 Η συνδεσμολογία του ESP32 S3 DevKit M1 με την πλακέτα ελέγχου του οχήματος

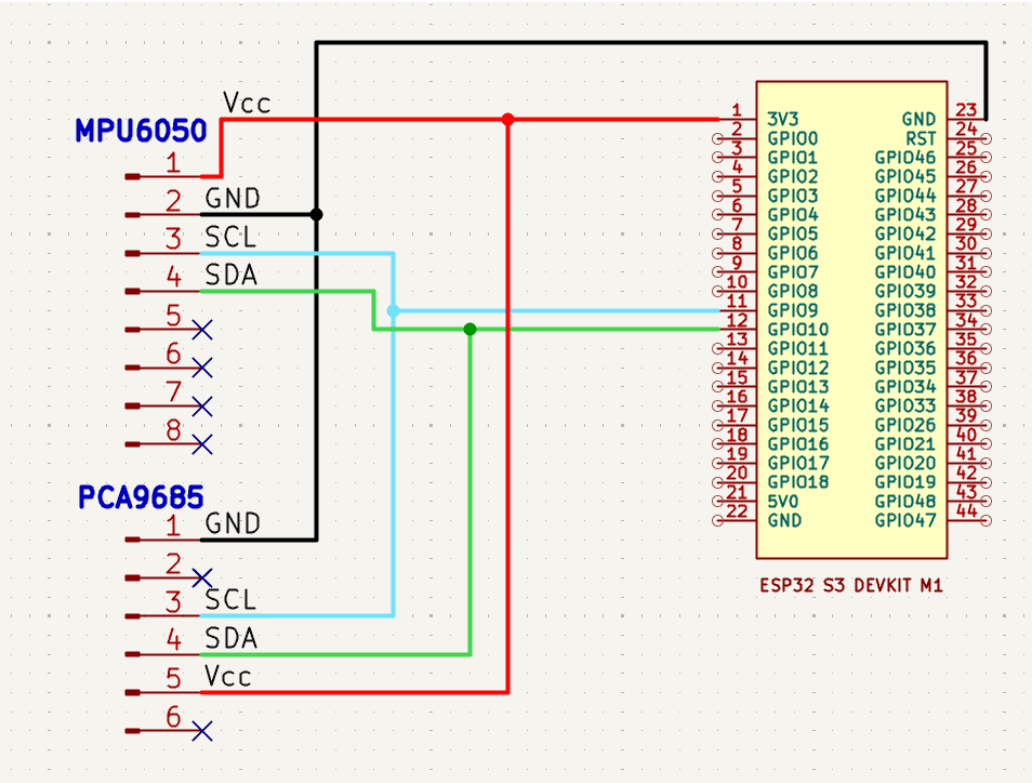
Αφού παρέχουμε στην πλακέτα ελέγχου τάση με ασφαλή τρόπο μπορούμε να την προδέσουμε πάνω στο όχημα. Κάναμε χρήση της εξόδου σταθερής τάσης 5V με στόχο την τροφοδοσία του Esp32 S3 DevKit M1 που τοποθετήσαμε πάνω στο τροχοφόρο. Θα συνδέσουμε την έξοδο 5V με την επαφή 5V του ESP32 S3 και την έξοδο GND με μία από τις γειώσεις που παρέχει το ESP32 S3 (Σχήμα 63). Όπως αναφέραμε και στο κεφάλαιο 2 η επικοινωνία μεταξύ των μονάδων γίνεται μέσω του πρωτοκόλλου I2C. Συνδέσαμε την είσοδο 5V Input με την επαφή 5V του ESP32 S3, τις γειώσεις μεταξύ τους και για τα σήματα ελέγχου, SDA και SCL, κάναμε χρήση των επαφών GPIO 5 και GPIO 1 του ESP32 S3 αντίστοιχα.



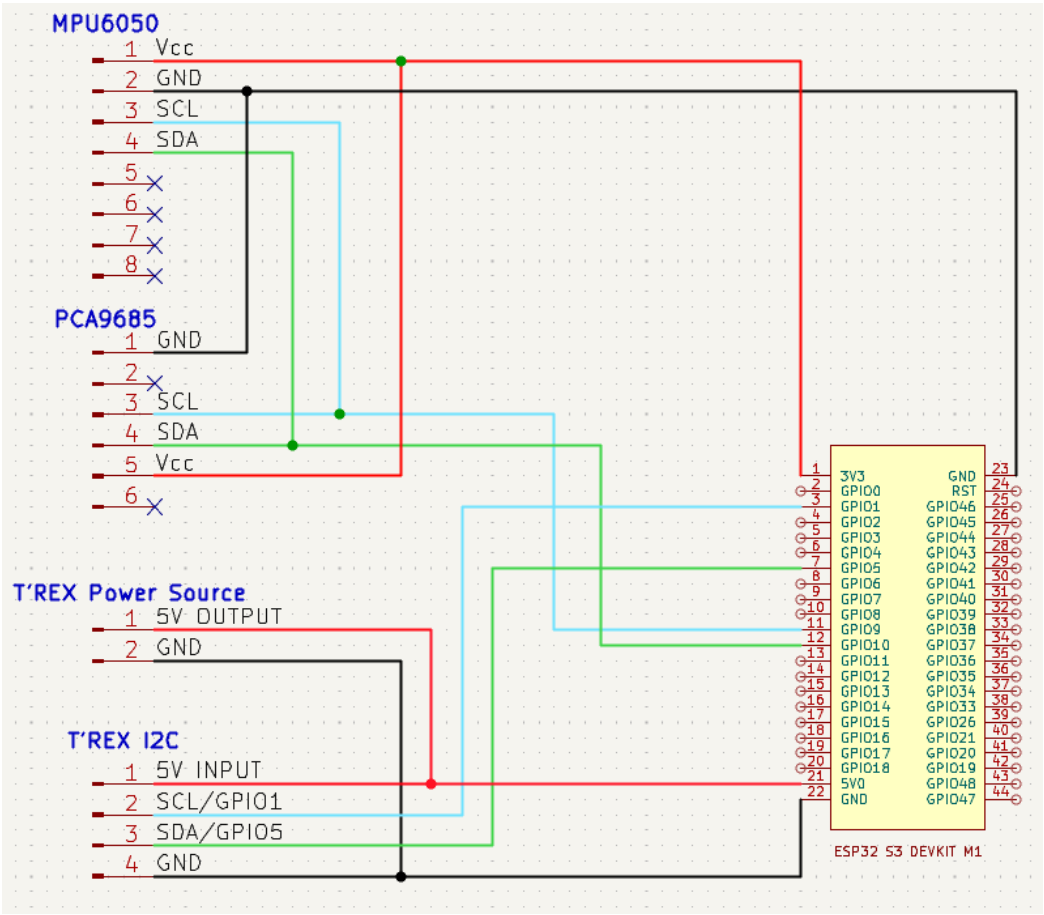
Σχήμα 63: Η συνδεσμολογία μεταξύ του εγκεφάλου και της πλακέτας ελέγχου οχήματος

3.2.3 Το κύκλωμα και η δομή του ρομποτικού βραχίονα

Στην βάση του ρομποτικού βραχίονα θα τοποθετηθεί ο εγκέφαλος του οχήματος μαζί με μία μονάδα MPU6050 και την πλακέτα οδήγησης σερβοκινητήρων PCA9685. Οι μονάδες θα συνδεθούν με τον εγκέφαλο και θα επικοινωνούν μέσω του πρωτοκόλλου I2C με αυτόν. Η παροχή τάσης από τον εγκέγαλο θα γίνει μέσω της επαφής 3.3V και οι γειώσεις θα συνδεθούν μεταξύ τους καθώς θέλουμε να είναι κοινές και στις 3 πλακέτες. Για τα σήματα SDA και SCL θα επιλέξουμε τις επαφές GPIO10 και GPIO9 αντίστοιχα (Σχήμα 64). Όπως είδαμε στο κεφάλαιο 2 το πρωτόκολλο I2C μας δίνει την δυνατότητα να έχουμε πολλαπλές περιφερειακές συσκευές συνδεδεμένες με τις ίδιες εξόδους για τα σήματα SDA και SCL. Επιλέξαμε να συνδέσουμε την πλακέτα ελέγχου του οχήματος σε διαφορετική σύνδεση I2C είναι γιατί θέλουμε ο δίαυλος επικοινωνίας να είναι πάντα διαθέσιμος για την ανταλλαγή δεδομένων μεταξύ του ESP32 S3 DevKit M1 και αυτής. Έτσι το σύστημα μας θα έχει καλύτερη απόκριση στις εντολές που θα λαμβάνει από το ασύρματο χειριστήριο.



Σχήμα 64: Η συνδεσμολογία του εγκεφάλου με τις μονάδες



Σχήμα 65: Η τελική συνδεσμολογία του ESP32 S3 DevKit M1 με τα περιφερειακά

Όπως αναφέραμε και στο κεφάλαιο 2 η μονάδα PCA9685 χρειάζεται εξωτερική παροχή τάσης στα 5V η οποία θα τροφοδοτεί τους σερβοκινητήρες. Καθώς η τάση της μπαταρίας είναι πολύ μεγαλύτερη, από αυτή που μπορεί να αντέξει η μονάδα, πρέπει πρώτα να την μειώσουμε στα 5V. Για να το πετύχουμε αυτό χρησιμοποιήσαμε έναν DC/DC μετατροπέα τάσης. Επιλέξαμε τον 12-24V/5V 15 A [26] μετατροπέα της εταιρείας TOBSUN. Σύμφωνα με τις προδιαγραφές του μετατροπέα μπορούμε να εισάγουμε τάση από 12 έως 24V και να πάρουμε στην έξοδο σταθερά 5V και έως 15A. Η παροχή ρεύματος υπερκαλύπτει τις ανάγκες του ρομποτικού βραχίονα μας, καθώς η μέγιστη κατανάλωση ρεύματος είναι 6.15A, πολύ λιγότερο από όσο μπορεί να παρέχει ο μετατροπέας μας.

Σχήμα 66: Ο μετατροπέας τάσης

Συνδέοντας την μπαταρία απευθείας με τον μετατροπέα και έπειτα την έξοδο του με την είσοδο της μονάδας PCA9685 μπορούμε να παρέχουμε στους σερβοκινητήρες της ρομποτικής δαγκάνας το απαιτούμενο ρεύμα με ασφάλεια. Στην περίπτωση που θέλουμε να αποσυνδέσουμε την μπαταρία με το κύκλωμα ασφαλείας η τροφοδοσία από το T'REX robot controller προς το ESP32 S3 διακόπτεται. Συνεπώς η παροχή της μονάδας PCA9685 κόβεται μηδενίζοντας έτσι το ρεύμα που τραβάει ο ρομποτικός βραχίονας από τον μετατροπέα κρατώντας το κύκλωμα ασφαλές.

Για τον σχεδιασμό του ρομποτικού βραχίονα κάναμε εκτενή έρευνα με στόχο να βρούμε έναν ο οποίος παρέχει πλήρη ελευθερία κινήσεων στον χρήστη, είναι ελαφρύς και προσφέρει ικανοποιητικό workspace (έκταση στην οποία φτάνει ο ρομποτικός βραχίονας). Η έρευνα μας οδήγησε στην πλατφόρμα Cults3D και συγκεκριμένα στον ρομποτικό βραχίονα του κατασκευαστή FABRI_CREATOR [7] ο οποίος σχεδίασε έναν βραχίονα με 6 βαθμούς ελευθερίας, συνολικό μήκος πάνω από 50 εκατοστά (51.4 εκατοστά) και συμβατότητα με Arduino. Έπειτα από αγορά των σχεδίων, στα 20€, ανακατασκευάσαμε την βάση του βραχίονα ώστε να είναι συμβατή με το ESP32 S3 DevKit M1 και το PCA9685 καθώς σκοπεύουμε να τα ενσωματώσουμε μέσα σε αυτή. Επίσης αλλάξαμε το σχήμα της βάσης του βραχίονα σε τετραγωνικό με σκοπό να τον στερεώσουμε πάνω στο σασί του οχήματος. Ο βραχίονας

[7] https://cults3d.com/en/3d-model/gadget/brazo-robotico-con-arduino-fusio360-file-robotic-arm-guardar-reproducir-exp

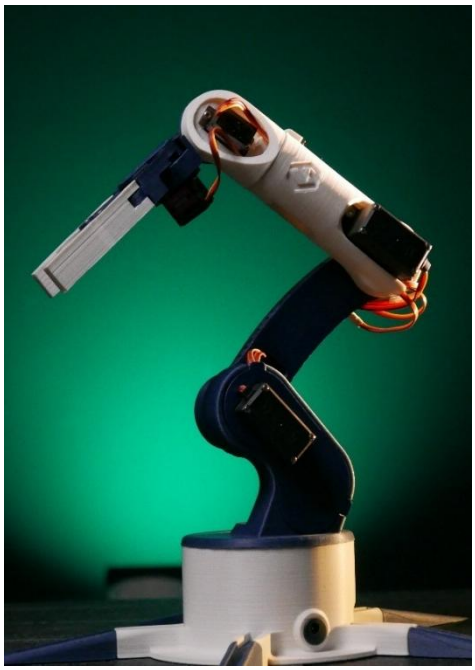
απαρτίζεται από 4 διακριτά μέρη, την βάση, το χέρι, τον καρπό και την δαγκάνα. Αναλυτικότερα:

Βάση: Κάνει χρήση ενός MG996R σερβοκινητήρα και είναι υπεύθυνη για την περιστροφή του βραχίονα και κατά επέκταση της δαγκάνας στον κάθετο άξονα.

Χέρι: Χωρίζεται σε 2 κλειδώσεις, τον ώμο και τον αγκώνα. Ο ώμος ενώνεται με τον αγκώνα μέσω ενός συνδέσμου και με τον ίδιο τρόπο ενώνεται ο αγκώνας με τον καρπό. Σε αυτά τα άκρα έχουμε εγκαταστήσει άλλους 2 σερβοκινητήρες MG996R οι οποίοι είναι υπεύθυνοι για το βάθος και το ύψος της δαγκάνας.

Καρπός: Ο καρπός αποτελείται από 2 μικρούς MG90S σερβοκινητήρες οι οποίοι είναι υπεύθυνοι για την περιστροφή του καρπού και για την κάμψη του προς τα πάνω και προς τα κάτω.

Δαγκάνα: Η δαγκάνα αποτελεί το τελευταίο κομμάτι του ρομποτικού βραχίονα και συγκεκριμένα αυτό το οποίο αλληλοεπιδράει με το περιβάλλον. Είναι προσκολλημένη πάνω στον καρπό και ελέγχεται από έναν μικρό σερβοκινητήρα MG90S. Για αυξημένη πρόσφυση με το αντικείμενο που θέλουμε να πιάσουμε, κολλήσαμε στις άκρες της δαγκάνας κομμάτια από λάστιχο.



Σχήμα 68: Ο ρομποτικός βραχίονας που αγοράσαμε

Σχήμα 67: Το τελικό αποτέλεσμα εκτύπωσης του ρομποτικού βραχίονα

3.3 Σύνοψη κεφαλαίου

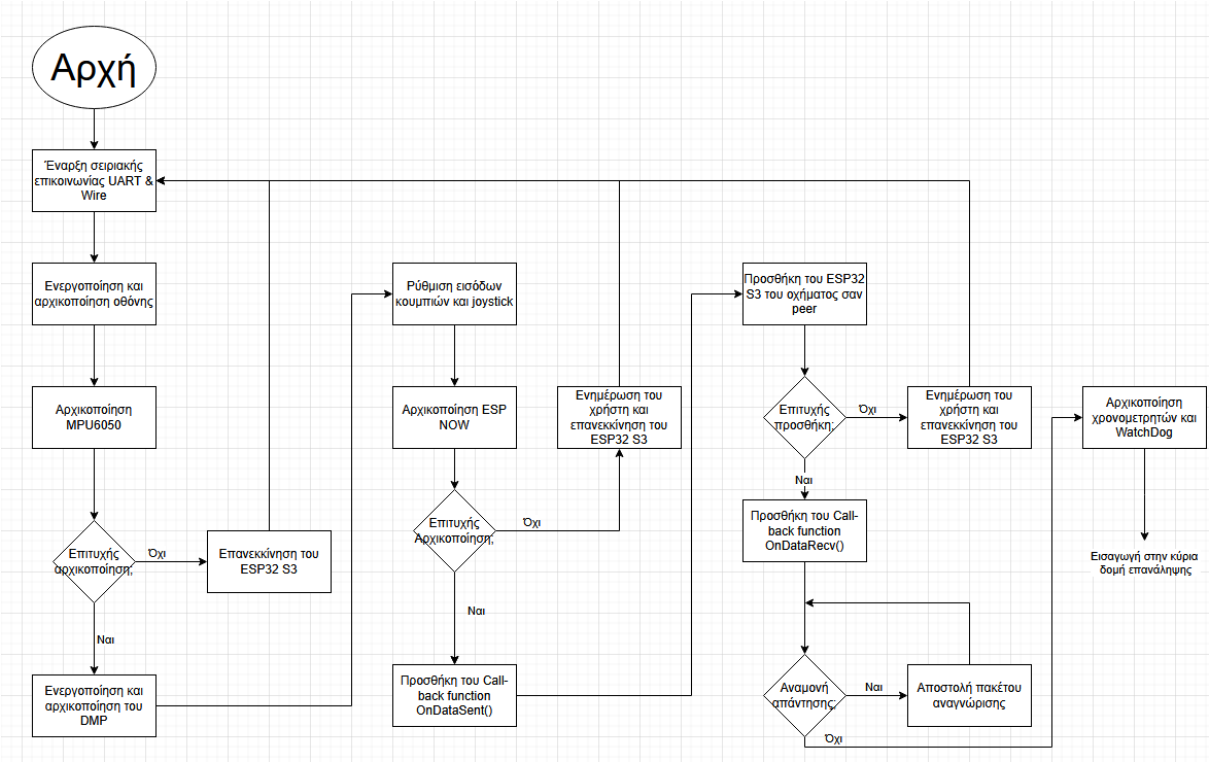
Στο παρόν κεφάλαιο αναλύσαμε το υλικό των 2 διακριτών συστημάτων, την δομή του ρομποτικού βραχίονα και τις συνδεσμολογίες που χρησιμοποιήσαμε για την επικοινωνία των περιφερειακών μονάδων με τις κεντρικές μονάδες του κάθε συστήματος. Στο επόμενο κεφάλαιο γίνεται μελέτη των σημαντικότερων συναρτήσεων του κώδικα των 2 αυτών συστημάτων.

Κεφάλαιο 4. Το λογισμικό των μονάδων ESP32 S3 DevKit C1 & ESP32 S3 DevKit M1

Σε αυτή την ενότητα θα αναλύσουμε τον κώδικα ο οποίος εκτελείται στα ESP32 S3 με στόχο να μελετήσουμε την λειτουργία των συστημάτων. Θα χωρίσουμε αυτό το κεφάλαιο σε 2 ενότητες, στον κώδικα του ασύρματου χειριστηρίου και στον κώδικα του τροχοφόρου οχήματος. Θα μελετήσουμε την ροή επικοινωνίας των δύο αυτών συστημάτων όπως επίσης και τις πρακτικές που εφαρμόσαμε για να κατασκευάσουμε ένα ολοκληρωμένο πακέτο λογισμικού. Η δομή ενός προγράμματος στο λογισμικό του Arduino IDE χωρίζεται σε 2 μέρη, το κομμάτι της αρχικοποίησης συστημάτων (setup) και την κύρια δομή επανάληψης η οποία εκτελείται όσο η μονάδα είναι ενεργοποιημένη. Για την καλύτερη κατανόηση της ροής του κώδικα παρατίθενται διαγράμματα ροής για το λογισμικό που αναπτύχθηκε για τα 2 διακριτά συστήματα.

4.1 Το λογισμικό του ασύρματου χειριστηρίου

Ξεκινώντας παρατίθεται το διάγραμμα ροής αρχικοποίησης του κώδικα του χειριστηρίου (Σχήμα 69).



Σχήμα 69: Διάγραμμα ροής αρχικοποίησης του χειριστηρίου

Συνεχίζοντας την ανάλυση του κώδικα, παρουσιάζονται παρακάτω οι βιβλιοθήκες που χρησιμοποιήσαμε όπως επίσης και οι σημαντικότερες συναρτήσεις οι οποίες αποτελούν την βάση λειτουργίας του χειριστηρίου.

```
#include <WiFi.h> //Βιβλιοθήκη για την ενεργοποίηση του WiFi στο ESP32 S3
#include <esp_now.h> // Βιβλιοθήκη για χρήση του πρωτοκόλλου ESP NOW
#include <SPI.h> //Βιβλιοθήκη για την χρήση του πρωτοκόλλου SPI
#include <Adafruit_GFX.h> //Βιβλιοθήκη για την χρήση γραφικών στην οθόνη
#include <Adafruit_ST7735.h> //Βιβλιοθήκη για την χρήση της οθόνης
#include <bitmaps.h> //Βιβλιοθήκη για την αποθήκευση εικόνων
#include <Wire.h> //Βιβλιοθήκη για την χρήση του πρωτοκόλλου I2C
#include "I2Cdev.h" //Επιπλέον βιβλιοθήκη για την χρήση του πρωτοκόλλου I2C,
// η βιβλιοθήκη αυτή χρησιμοποιείται από την βιβλιοθήκη MPU6050_6Axis_MotionApps20.h
#include "MPU6050_6Axis_MotionApps20.h" //Βιβλιοθήκη για την χρήση της μονάδας MPU6050
#include "esp_task_wdt.h" //Βιβλιοθήκη για την χρήση του μετρητή watchdog
//για επανεκκίνηση της μονάδας σε περίπτωση σφάλματος
```

Στην αρχικοποίηση του κώδικα εισάγουμε όλες τις βιβλιοθήκες που θα χρησιμοποιήσουμε. Η βιβλιοθήκη “bitmaps.h” περιέχει 4 bitmaps τα οποία κατασκευάσαμε από μία εικόνα βέλους κατεύθυνσης την οποία περιστρέψαμε στις 90, 180 και 270 μοίρες με στόχο να δείξουμε την κατεύθυνση προς την οποία έχει περιστραφεί το όχημα. Τα bitmap αποτελούν μορφές απεικόνισης εικόνων μέσω ενός πίνακα pixel (εικονοστοιχείο). Κάθε pixel έχει μια χρωματική τιμή και μέσω του bitmap αντιστοιχίζεται σε ένα συγκεκριμένο σημείο της εικόνας. Οι βιβλιοθήκες “I2Cdev.h” και “MPU6050_6Axis_MotionApps20.h” έγιναν εγκατάσταση από τον φάκελο “i2cdevlib/Arduino/MPU6050” του χρήστη “jrowberg” από την σελίδα GitHub. Αφού ολοκληρώθηκε η εισαγωγή των βιβλιοθηκών ενεργοποιούμε όλες τις περιφερειακές μονάδες οι οποίες επικοινωνούν με το ESP32 S3 DevKit C1 οι οποίες είναι η οθόνη, το MPU6050, τα κουμπιά και το joystick. Επίσης ορίζουμε έναν ενσωματωμένο μετρητή επαγρίπνισης στο ESP32 S3 ο οποίος ονομάζεται watchdog και είναι υπεύθυνος για την επανεκκίνηση της συσκευής σε περίπτωση που διακοπεί η φυσιολογική ροή του κώδικα. Στην συνέχεια ενεργοποιούμε το πρωτόκολλο ESP-NOW και προσθέτουμε το ESP32 S3 του οχήματος ως peer, ώστε να ολοκληρωθεί η επικοινωνία μεταξύ των συστημάτων. Κατά την αρχικοποίηση της επικοινωνίας μεταξύ των ESP32 S3 ορίζουμε και 2 ασύγχρονες συναρτήσεις οι οποίες εκτελούνται όταν ένα ESP32 S3 στείλει ένα πακέτο επικοινωνίας και όταν λάβει ένα. Οι ασύγχρονες εντολές εκτελούνται σε οποιοδήποτε σημείο του προγράμματος χωρίς να εμποδίζουν την φυσιολογική ροή του. Οι δύο που μας παρέχει το πρωτόκολλο ESP NOW είναι η onDataSent() η οποία εκτελείται κατά την αποστολή ενός πακέτου και η onDataRecv() που εκτελείται κατά την λήψη ενός πακέτου. Εμείς αξιοποιήσαμε την δεύτερη συνάρτηση με στόχο την αυτόματη αποθήκευση δεδομένων, αλλά και την ενημέρωση στοιχείων πάνω στην οθόνη. Πριν δούμε με περισσότερες λεπτομέρειες αυτή τη συνάρτηση θα κάνουμε μία αναφορά στα πακέτα επικοινωνίας που φτιάξαμε για την αποστολή και την λήψη δεδομένων. Συνολικά η επικοινωνία αποτελείται από 3 πακέτα, το πρώτο είναι το πακέτο που στέλνουμε προς το ESP32 S3 του τροχοφόρου οχήματος (Σχήμα 67). Το πακέτο αυτό εμπεριέχει μια δομή που αποτελείται από έναν πίνακα με την τιμή των 12 κουμπιών πάνω στο χειριστήριο και έναν πίνακα που αποθηκεύουμε τις τιμές της ταχύτητας περιστροφής των κινητήρων στην αριστερή και στην δεξιά πλευρά του οχήματος. Το δεύτερο πακέτο είναι αυτό που λαμβάνουμε από το ESP32 S3 του ρομπότ και εμπεριέχει δεδομένα όπως την τάση της μπαταρίας, την περιστροφή του οχήματος ως προς τον κάθετο άξονα, το ρεύμα που αντλεί το κάθε channel της πλακέτας ελέγχου του οχήματος, έναν δείκτη πιθανών σφαλμάτων στα πακέτα επικοινωνίας μεταξύ του

ESP32 S3 του οχήματος και της πλακέτας ελέγχου και τέλος, μία διχοτομική μεταβλητή που ενεργοποιείται αν υπάρχει σύγκρουση του οχήματος (Σχήμα 68).

```
27  struct controller_payload{
28    bool buttons[12] = {0,0,0,0,0,0,
29    0,0,0,0,0,0};
30    short wheel_speed[2] = {0,0};
31  };
32  struct controller_payload contr_payload;
```

Σχήμα 70: Το πακέτο επικοινωνίας που στέλνει το χειριστήριο

```
31  struct receiver_payload{
32    byte errorFlag;
33    byte batteryVoltageHighByte;
34    byte batteryVoltageLowByte;
35    bool impactDetected;
36    float carYaw;
37    int leftCurrent;
38    int rightCurrent;
39  };
40  struct receiver_payload rec_payload;
```

Σχήμα 71: Το πακέτο επικοινωνίας που λαμβάνει το χειριστήριο

Το τελευταίο πακέτο χρησιμοποιείται για την δημιουργία ενός συστήματος heartbeat, δηλαδή παλμού το οποίο εξασφαλίζει ότι το χειριστήριο γνωρίζει αν χάθηκε η επικοινωνία με το όχημα και να πράξει αναλόγως. Αποτελείται από μία διχοτομική μεταβλητή η οποία είναι αληθές αν η επικοινωνία έχει χαθεί και ψευδές αν είναι σταθερή. Η αποστολή αυτού του πακέτου γίνεται μόνο αν το χειριστήριο ανιχνεύσει ότι η επικοινωνία με το όχημα έχει χαθεί. Κάθε φορά που γίνεται λήψη του δεύτερου πακέτου γίνεται επαναφορά ενός μετρητή. Αν ο μετρητής δεν αλλάξει τιμή εντός μισού δευτερολέπτου το σύστημα θεωρεί ότι η επικοινωνία χάθηκε και ξεκινάει η αποστολή του τρίτου πακέτου μέχρι το όχημα να στείλει πίσω μία απάντηση. Η απάντηση του οχήματος στο πακέτου του τρίτου τύπου είναι με ένα πακέτο ακριβώς ίδιας μορφής. Έπειτα η επικοινωνία επανέρχεται στην φυσιολογική κατάσταση, δηλαδή γίνεται χρήση του πρώτου και το δεύτερου πακέτου. Όλα αυτά εκτελούνται μέσα στην ασύγχρονη εντολή onDataRecv() (Σχήμα 72). Αν το πακέτο που λήφθηκε είναι του δεύτερου τύπου, αντιγράφουμε όλα τα δεδομένα σε μια μεταβλητή που φτιάξαμε με όνομα rec_payload (Σχήμα 71). Στην συνέχεια ελέγχουμε αν το errorFlag είναι μηδενικό (Σχήμα 73). Αν είναι συμπεραίνουμε, ότι υπήρξε θέμα μεταξύ της επικοινωνίας της πλακέτας ελέγχου και του ESP32 S3 του οχήματος και διακόπτουμε την λειτουργία του χειριστηρίου για να αποφύγουμε να δημιουργήσουμε βλάβες στο όχημα.

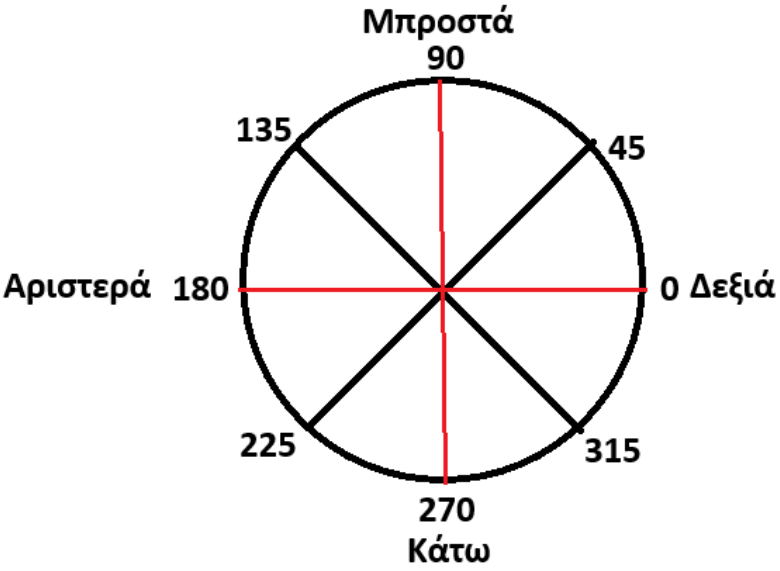
```
103 //Η εντολή OnDataRecv εκτελείται όταν το ESP32 S3 λάβει ένα πακέτο μέσω του πρωτόκολλου ESP-NOW
104 void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
105   //Ενημέρωση του heartbeat timer
106   heartbeat_timer = millis();
107   //Αναγνώριση του πακέτου σύνδεσης του ESP32 S3 του τροχοφόρου οχήματος
108   if(len == sizeof(bool)){
109     //Ενημέρωση της κατάστασης της επικοινωνίας
110     waiting_response = false;
111     Serial.println("Received Heartbeat");
112     return;
113   }
114   //Αντιγραφή όλων των δεδομένων του πακέτου του οχήματος
115   memcpy(&rec_payload, incomingData, len);
```

Σχήμα 72: Η δομή της συνάρτησης onDataRecv του χειριστηρίου

```
116   if(rec_payload.errorFlag != 0){
117     display.fillScreen(ST77XX_BLACK);
118     display.setCursor(0, 0);
119     display.print("Error Code:");
120     display.print(rec_payload.errorFlag);
121     while(1);
122   }
```

Σχήμα 73: Αναγνώριση σφάλματος από το πακέτο επικοινωνίας

Αν η τιμή του errorFlag είναι μηδενική μπορούμε να χρησιμοποιήσουμε τα υπόλοιπα δεδομένα του πακέτου. Ένα από τα δεδομένα που χρησιμοποιούμε είναι η περιστροφή του οχήματος ώστε να εκτυπώσουμε στην οθόνη ένα βέλος που δείχνει την κατεύθυνση προς την οποία κοιτάει το όχημα σε σχέση με την περιστροφή του χειριστηρίου. Για τον υπολογισμό της σωστής κατεύθυνσης χωρίζουμε το εύρος των γωνιών, από 0º έως 359º, σε 4 ίσα τεταρτημόρια κάνοντας τις εξής αντιστοιχίες (Σχήμα 74):



Σχήμα 74: Τα τεταρτημόρια κατεύθυνσης του οχήματος

Αφού υπολογίσουμε την διαφορά μεταξύ της γωνίας του οχήματος και του χειριστηρίου, προσθέτουμε 45º ώστε να μετατοπίσουμε την τιμή από τα τεταρτημόρια των κόκκινων γραμμών σε αυτά των μαύρων. Έπειτα διαιρούμε τον αριθμό αυτό με 90º για να υπολογίσουμε σε ποιο τεταρτημόριο ανήκει η γωνία που υπολογίσαμε. Το ακέραιο αποτέλεσμα της διαίρεσης αντιστοιχεί και σε μία κατεύθυνση περιστροφής ως εξής: 0 για δεξιά, 1 για μπροστά, 2 για αριστερά, 3 για κάτω και 4 πάλι για δεξιά. Ο λόγος που υπάρχουν 2 τιμές για την δεξιά κατεύθυνση είναι επειδή η μέγιστη τιμή της διαφοράς μεταξύ των γωνιών είναι 359º και σε αυτό το νούμερο προσθέτουμε 45º. Συνεπώς η μέγιστη τιμή της γωνίας είναι 404º και το ακέραιο μέρος της διαίρεσης με 90 επιστρέφει 4. Αυτός ο δείκτης τεταρτημόριου αντιστοιχεί σε ένα βέλος κατεύθυνσης της βιβλιοθήκης bitmap.h το οποίο μπορούμε να εκτυπώσουμε πάνω στην οθόνη (Σχήμα 75).

```
123   //Αποθηκεύουμε την διαφορά περιστροφής του οχήματος και του χειριστηρίου
124   int new_car_yaw_index = rec_payload.carYaw - current_yaw + 45;
125   //Μετατρέπουμε την γωνία σε ένα εύρος από 0 έως 360 μοίρες
126   new_car_yaw_index += (new_car_yaw_index < 0) * 360;
127   //Διαιρούμε με 90 ώστε να υπολογίσουμε τον δείκτη κατεύθυνσης
128   //Ο δείκτης αντιστοιχεί σε μία από τις 4 κατευθύνσεις
129   //0 = δεξιά
130   //1 = μπροστά
131   //2 = αριστερά
132   //3 = πίσω
133   //4 = δεξιά
134   new_car_yaw_index = new_car_yaw_index/90;
135   //Αν η νέα κατεύθυνση είναι διαφορετική από την προηγούμενη, ζωγραφίζουμε το βέλος της νέας κατεύθυνσης
136   if(current_yaw_index != (new_car_yaw_index % 4)){
137     //Χρωματίζουμε μαύρο όλο το πλαίσιο που τοποθετούμε το βέλος
138     display.fillRect(80,60,62,62,ST77XX_BLACK);
139     //Ζωγραφίζουμε το νέο βέλος με βάση τον δείκτη κατεύθυνσης
140     display.drawBitmap(80,60,arrow_bitmaps[new_car_yaw_index % 4],BITMAP_WIDTH,BITMAP_HEIGHT,ST77XX_WHITE);
141     current_yaw_index = new_car_yaw_index;
142   }
```

Σχήμα 75: Ο υπολογισμός του βέλους κατεύθυνσης

Αφού υπολογίσουμε την γωνία περιστροφής και ενημερώσουμε το βέλος κατεύθυνσης αναλόγως υπολογίζουμε την τάση της μπαταρίας ώστε να εκτυπώσουμε στην οθόνη το ποσοστό μπαταρίας του οχήματος. Για να το πετύχουμε αυτό ορίσαμε έναν δισδιάστατο πίνακα ο οποίος συνδυάζει 3 στοιχεία (Σχήμα 76), μια τιμή τάσης, έναν χρωματικό κωδικό, ένα ποσοστό στα 100. Μία μπαταρία LiPo 14.8V όταν είναι πλήρως γεμάτη έχει τάση εξόδου 16.80V ενώ όσο αδειάζει η τάση της μπορεί να πέσει και στα 13.2V.

```
19  const int batteryVoltageTable[][3] = {{1680,100,ST77XX_GREEN},{1660,95,ST77XX_GREEN},{1644,90,ST77XX_GREEN},
20  {1632,85,ST77XX_GREEN},{1608,80,ST77XX_GREEN},{1592,75,ST77XX_GREEN},
21  {1580,70,ST77XX_GREEN},{1564,65,ST77XX_GREEN},{1548,60,ST77XX_GREEN},{1540,55,ST77XX_GREEN},
22  {1520,50,ST77XX_GREEN},{1508,45,ST77XX_ORANGE},{1496,40,ST77XX_ORANGE},{1484,35,ST77XX_ORANGE},
23  {1472,30,ST77XX_ORANGE},{1460,25,ST77XX_RED},{1444,20,ST77XX_RED},{1432,15,ST77XX_RED},
24  {1400,10,ST77XX_RED},{1370,5,ST77XX_RED},{1320,0,ST77XX_RED}};
25  int battery_index = 0;
```

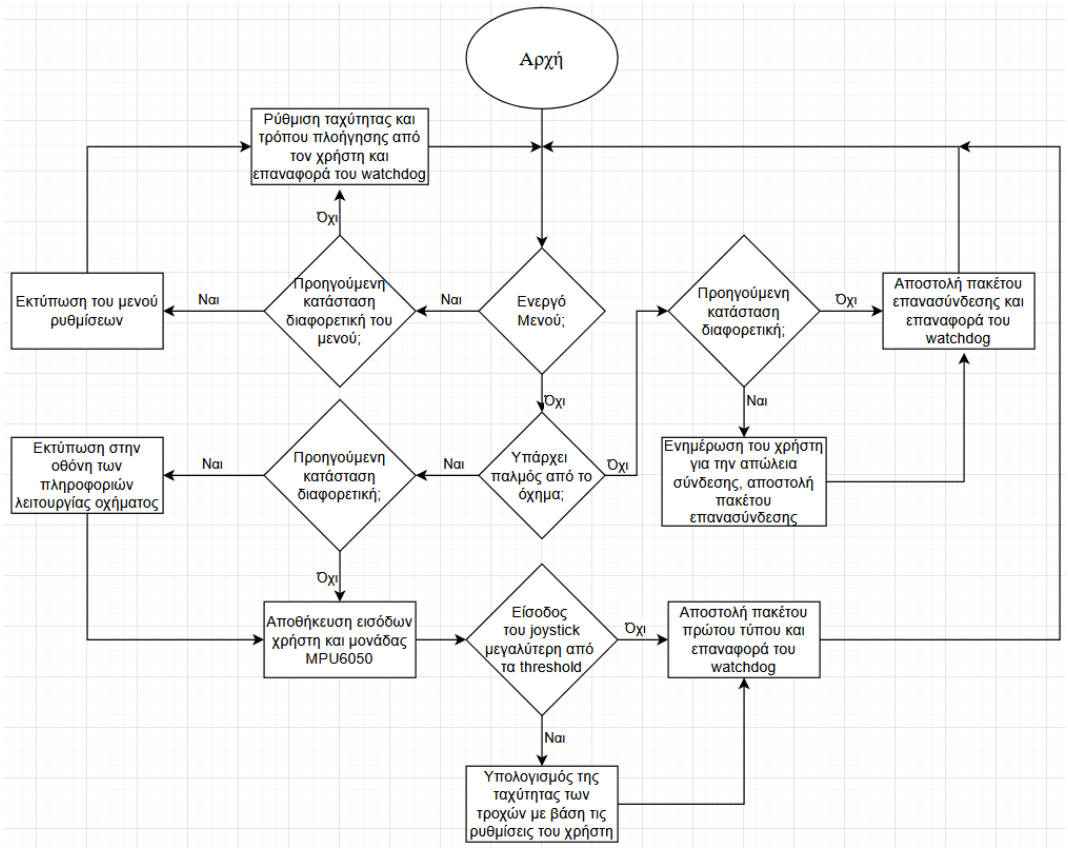
Σχήμα 76: Πίνακας αντιστοίχισης τάσης εξόδου μπαταρίας με το ποσοστό φόρτισης

Οι τιμές δηλώνουν την τάση πολλαπλασιασμένη με 100 οπότε η τιμή 1680 συμβολίζει τα 16.8V. Καθώς η άντληση υψηλού ρεύματος από τους κινητήρες μειώνει την τάση εξόδου της μπαταρίας υπολογίζουμε την πραγματική τάση της μπαταρίας προσθέτοντας την τωρινή μέτρηση τάσης εξόδου με την εσωτερική αντίσταση της μπαταρίας, την οποία μετρήσαμε και υπολογίσαμε στα 0.009 Ohm, πολλαπλασιασμένη με το συνολικό ρεύμα που αντλούν οι κινητήρες. Έπειτα μπορούμε να ελέγξουμε αν η πραγματική τάση εξόδου της μπαταρίας είναι πιο χαμηλή από την τιμή της τάσης του πίνακα τιμών της μπαταρίας. Στην περίπτωση που είναι ενημερώνουμε την οθόνη ώστε να εκτυπώνονται τα σωστά δεδομένα. Ο χρωματικός κωδικός χρησιμοποιείται, ώστε να δίνουμε χρώμα στην μπάρα ένδειξης της μπαταρίας. Η μπάρα είναι πράσινη άμα το ποσοστό της μπαταρίας είναι υψηλό, πορτοκαλί άμα βρίσκεται κοντά στην μέση και κόκκινη αν η τάση είναι κάτω από το 25%. Κάνοντας χρήση του ποσοστού τις εκατό μπορούμε να αλλάξουμε αναλόγως και το μέγεθος της μπάρας που εκτυπώνουμε στην οθόνη. Οπότε αν η μπαταρία είναι πλήρης η μπάρα είναι μήκους 100 pixel και αναλόγως το ποσοστό μικραίνει. Στο τέλος εκτυπώνουμε στην σειριακή θύρα, αν είναι συνδεδεμένη, την τάση της μπαταρίας και το ρεύμα των channel της πλακέτας ελέγχου για debug και σαφήνεια των μετρήσεων (Σχήμα 77).

```
155   //Αποθήκευση της τάσης της μπαταρίας
156   int battery_voltage = (rec_payload.batteryVoltageHighByte<<8) | rec_payload.batteryVoltageLowByte;
157   //Καθώς κατά την χρήση του οχήματος η τάση εξόδου της μπαταρίας πέφτει
158   //υπολογίζουμε την πραγματική της τιμή προσθέτοντας την μετρούμενη τάση
159   //συν την εσωτερική αντίσταση της μπαταρίας (0.009 Ohm) επί το συνολικό ρεύμα που χρειάζονται οι κινητήρες
160   if(abs(contr_payload.wheel_speed[0]) > 60 || abs(contr_payload.wheel_speed[1]) > 60){
161     Serial.println("Balanced Battery Voltage");
162     battery_voltage += 0.009 * (rec_payload.leftCurrent + rec_payload.rightCurrent);
163   }
164   //Προσπέλαση του πίνακα τιμών της μπαταρίας μέχρι να βρεθεί η σωστή τιμή ποσοστού της μπαταρίας
165   if(batteryVoltageTable[battery_index][0] > battery_voltage){
166     do{
167       battery_index++;
168     }
169     while(batteryVoltageTable[battery_index][0] > battery_voltage);
170     display.fillRect(60,0,100,8,ST77XX_BLACK);
171     display.fillRect(60,1,batteryVoltageTable[battery_index][1],6,batteryVoltageTable[battery_index][2]);
172   }
```

Σχήμα 77: Υπολογισμός και εκτύπωση του ποσοστού της μπαταρίας

Εφόσον αναλύσαμε την ασύγχρονη συνάρτηση onDataRecv(), μπορούμε τώρα να δούμε την κύρια δομή επανάληψης του προγράμματος η οποία είναι υπεύθυνη για την ανάγνωση της εισόδου του χρήστη, του υπολογισμού ταχύτητας των τροχών του οχήματος, την χρήση του κύριου μενού ρυθμίσεων και τον έλεγχο της κατάστασης της επικοινωνίας. Από κάτω παρέχουμε το διάγραμμα ροής της συνάρτησης για συνάφεια (Σχήμα 78).



Σχήμα 78: Διάγραμμα ροής κύριας δομής επανάληψης χειριστηρίου

Το ασύρματο χειριστήριο είναι υπεύθυνο για τον υπολογισμό της ταχύτητας των τροχών του οχήματος. Η μέση και η μέγιστη ταχύτητα του οχήματος ορίζεται από το μενού ρυθμίσεων το οποίο έχει 4 επιλογές ταχύτητας, Slow, Normal, Fast, Mad Max και επιτρέπουν το 56%, το 70%, το 80% και το 100% της δύναμης του οχήματος αντίστοιχα. Ο υπολογισμός εξαρτάται από την ρύθμιση πλοήγησης που έχει επιλεχθεί από τον χρήστη. Η ρύθμιση πλοήγησης αποτελείται από 2 επιλογές το tank mode και το directional mode. Το tank mode είναι η προεπιλογή του τρόπου πλοήγησης και λειτουργεί εξαρτημένη μόνο από την είσοδο του joystick. Με βάση το X και το Y που διαβάζει το ESP32 S3 υπολογίζουμε την γωνία που σχηματίζεται με τον X άξονα με εύρος τιμών από -180º έως 180º. Έτσι αναλόγως την γωνία αλλάζουμε και την ταχύτητα στους τροχούς. Αν η γωνία είναι μεταξύ 45º και 135º η ταχύτητα υπολογίζεται από το μήκος του διανύσματος του σημείου X,Y από το 0,0 πολλαπλασιασμένο με την μέγιστη επιτρεπτή ταχύτητα (Σχήμα 78). Το ίδιο ισχύει και αν η γωνία είναι από -45º έως -135º με την διαφορά ότι η ταχύτητα είναι αρνητική. Τέλος αν η απόλυτη γωνία είναι μεγαλύτερη των 135º ή μικρότερη των 45º τότε περιστρέφουμε το όχημα αριστερά ή δεξιά αντιστοίχως θέτοντας την ταχύτητα περιστροφής των τροχών ίση σε τιμή αλλά αντίθετη σε φορά με μέγιστη τιμή το 160 (Σχήμα 81). Για την επιλογή της μέγιστης ταχύτητας περιστροφής δοκιμάστηκαν διάφορες τιμές και παρατηρήθηκε ότι σε αυτή το όχημα έχει αρκετή δύναμη ώστε να περιστρέφεται χωρίς όμως να το παρασέρνει η φόρα όταν θέλουμε να σταματήσει. Για την περιστροφή βασιστήκαμε το φαινόμενο του differential steering το οποίο επιτρέπει στο όχημα να περιστρέφεται γύρω από τον κάθετο άξονα χωρίς να κουνηθεί προς τα μπροστά ή προς τα πίσω.

```
537  void TankControlMovement(int joystickX,int joystickY){
538    //Υπολογισμός μέτρου του διανύσματος του σημείου X,Y από το 0,0
539    int vector_length = (joystickX * joystickX) + (joystickY * joystickY);
540    float movement_vector_length = sqrt(vector_length);
541    //Αν η ταχύτητα είναι πολύ χαμηλή το όχημα δε κινείται οπότε ορίζουμε ένα ελάχιστο threshold
542    float movement_vector_magnitude = constrain(movement_vector_length/512,0.4,1);
543    float threshold = 0.5;
544    //Υπολογισμός γωνίας με τον άξονα X
545    int desired_angle = int(atan2((double)joystickY,(double)joystickX) * 180/M_PI);
```

Σχήμα 79: Υπολογισμός μεταβλητών στην συνάρτηση TankControlMovement()

```
555    else{
556      int target_speed = speed_modes[speed_mode_index]* movement_vector_magnitude;
557      if(desired_angle > 0){
558        //left side wheel speed
559        contr_payload.wheel_speed[0] += increment_per_step[speed_mode_index];
560        //right side wheel speed
561        contr_payload.wheel_speed[1] += increment_per_step[speed_mode_index];
562        if(contr_payload.wheel_speed[0] > target_speed){
563          contr_payload.wheel_speed[0] = target_speed;
564          contr_payload.wheel_speed[1] = target_speed;
565        }
566      }
567      else{
568        //left side wheel speed
569        contr_payload.wheel_speed[0] -= increment_per_step[speed_mode_index];
570        //right side wheel speed
571        contr_payload.wheel_speed[1] -= increment_per_step[speed_mode_index];
572        if(contr_payload.wheel_speed[0] < -target_speed){
573          contr_payload.wheel_speed[0] = -target_speed;
574          contr_payload.wheel_speed[1] = -target_speed;
575        }
```

Σχήμα 80: Ευθύγραμμη κίνηση μέσω της συνάρτησης TankControlMovement()

Σύμφωνα με τον κατασκευαστή του οχήματος η απότομη αλλαγή στην ταχύτητα περιστροφής των τροχών απαιτεί πολύ ρεύμα οδηγώντας σε πτώση της τάσης της εξόδου της μπαταρίας. Για να αποφύγουμε αυτό το πρόβλημα προγραμματίσαμε ένα σύστημα επιτάχυνσης ώστε το όχημα να φτάνει την ταχύτητα στόχου σταδιακά. Ως αποτέλεσμα η πτώση της τάσης εξόδου της μπαταρίας μειώνεται και έπειτα από δοκιμές μετρήθηκε κοντά στα 500mV.

```
545   int desired_angle = int(atan2((double)joystickY,(double)joystickX) * 180/M_PI);
546   if(abs(desired_angle) > 135 && movement_vector_magnitude > threshold){
547     //Η μέγιστη ταχύτητα που θέλουμε να φτάσουν οι τροχοί
548     int target_speed = 160;
549     //Η ταχύτητα των αριστερών τροχών
550     contr_payload.wheel_speed[0] -= increment_per_step[speed_mode_index];
551     if(contr_payload.wheel_speed[0] < -target_speed){
552       contr_payload.wheel_speed[0] = -target_speed;
553     }
554     //Η ταχύτητα των δεξιών τροχών
555     contr_payload.wheel_speed[1] += increment_per_step[speed_mode_index];
556     if(contr_payload.wheel_speed[1] > target_speed){
557       contr_payload.wheel_speed[1] = target_speed;
558     }
559     //Εκτύπωση αριθμών
560     Serial.print("Target Speed:");
561     Serial.println(target_speed);
562     Serial.print("Left motor Speed:");
563     Serial.println(contr_payload.wheel_speed[0]);
564     Serial.print("Right motor Speed:");
565     Serial.println(contr_payload.wheel_speed[1]);
566   }
```

Σχήμα 81: Αριστερόστροφη περιστροφή μέσω της συνάρτησης TankControlMovement()

Η δεύτερη ρύθμιση πλοήγησης είναι η directional η οποία εξαρτάται από την είσοδο του joystick αλλά και από την σχετική περιστροφή μεταξύ του χρήστη και του οχήματος. Η περιστροφή του χειριστηρίου ορίζει τον άξονα περιστροφής για το όχημα και αναλόγως την κατεύθυνση που δίνει ο χρήστης με το joystick το όχημα ακολουθεί αυτή. Το όχημα περιστρέφεται όπως και στην συνάρτηση TankControlMode() μέχρι η περιστροφή του να αντιστοιχεί σε αυτή που δίνει ο χρήστης, έπειτα προχωράει προς εκείνη την κατεύθυνση με ταχύτητα που υπολογίζεται με τον ίδιο τρόπο που υπολογίζεται στο tank mode. Η κατεύθυνση περιστροφής είναι ίδια με αυτή της μικρότερης γωνίας μεταξύ της τωρινής περιστροφής του

```
589 void DirectionalMovement(int joystickX,int joystickY){
590   //Υπολογισμός γωνίας με τον άξονα X
591   int desired_angle = int(atan2((double)joystickY,(double)joystickX)* 180/M_PI);
592   //Μετατροπή γωνίας σε εύρος από 0 έως 360 μοίρες
593   desired_angle += (desired_angle < 0) * 360;
594   //Υπολογισμός μέτρου του διανύσματος του σημείου X,Y από το 0,0
595   int vector_length = (joystickX * joystickX) + (joystickY * joystickY);
596   float movement_vector_length = sqrt(vector_length);
597   float movement_vector_magnitude = constrain(movement_vector_length/512,0,1);
598   //Υπολογισμός διαφοράς γωνίας μεταξύ οχήματος και γωνίας περιστροφής που ορίζει ο χρήστης
599   int angle = int(rec_payload.carYaw - (desired_angle + current_yaw));
600   angle = angle%360;
601   //Μετατροπή γωνίας σε εύρος -180 έως 180 μοίρες
602   if(angle > 180){angle -= 360;}
603   if(angle < -180){angle += 360;}
604   int target_speed = speed_modes[speed_mode_index] * movement_vector_magnitude;
605   if(movement_vector_magnitude > 0.5){
606     if(abs(angle) > 10){
607       //Το 0 αντιστοιχεί σε φορά περιστροφής αντίστροφη του ρολογιού ενώ το 1 ίδια με αυτού
608       bool rotate_direction = (angle < 0);
609       float temp_rotation_speed = abs(angle)/180.0;
610       float rotation speed = constrain(temp rotation speed,0.6,1);
```

Σχήμα 82: Υπολογισμός δεδομένων για την χρήση της πλοήγησης Directional Movement

οχήματος και της επιθυμητής περιστροφής. Αν η γωνία περιστροφής απέχει πάνω από 15º τότε το όχημα περιστρέφεται (Σχήμα 83). Στην περίπτωση που το όχημα ευθυγραμμιστεί ή είναι ήδη ευθυγραμμισμένο με την επιθυμητή γωνία, προχωράει ευθεία. Αν στην προηγούμενη επανάληψη το όχημα περιστρεφόταν τότε μηδενίζουμε πρώτα την ταχύτητα των τροχών καθώς η απότομη αλλαγή της από την περιστροφή στην ευθεία πορεία θα αυξήσει απότομα τις απαιτήσεις του συστήματος σε ρεύμα, κάτι το οποίο αναφέραμε παραπάνω θέλουμε να αποφύγουμε.

```
639   int target_speed = speed_modes[speed_mode_index] * movement_vector_magnitude;
640   if(movement_vector_magnitude > 0.5){
641     if(abs(angle) > 15){
642       //Το θ αντιστοιχεί σε φορά περιστροφής αντίστροφη του ρολογιού ενώ το 1 ίδια με αυτού
643       bool rotate_direction = (angle < 0);
644       float temp_rotation_speed = abs(angle)/180.0;
645       float rotation_speed = constrain(temp_rotation_speed,0.6,1);
646       if(rotate_direction){
647         world_pos_rotating = true;
648         target_speed = 160;
649         //Η ταχύτητα των αριστερών τροχών
650         contr_payload.wheel_speed[0] -= increment_per_step[speed_mode_index];
651         if(contr_payload.wheel_speed[0] < -target_speed){
652           contr_payload.wheel_speed[0] = -target_speed;
653         }
654         //Η ταχύτητα των δεξιών τροχών
655         contr_payload.wheel_speed[1] += increment_per_step[speed_mode_index];
656         if(contr_payload.wheel_speed[1] > target_speed){
657           contr_payload.wheel_speed[1] = target_speed;
658         }
659         //Εκτύπωση αριθμών
660         Serial.print("Left motor Speed:");
661         Serial.println(contr_payload.wheel_speed[0]);
662         Serial.print("Right motor Speed:");
663         Serial.println(contr_payload.wheel_speed[1]);
664       }
```

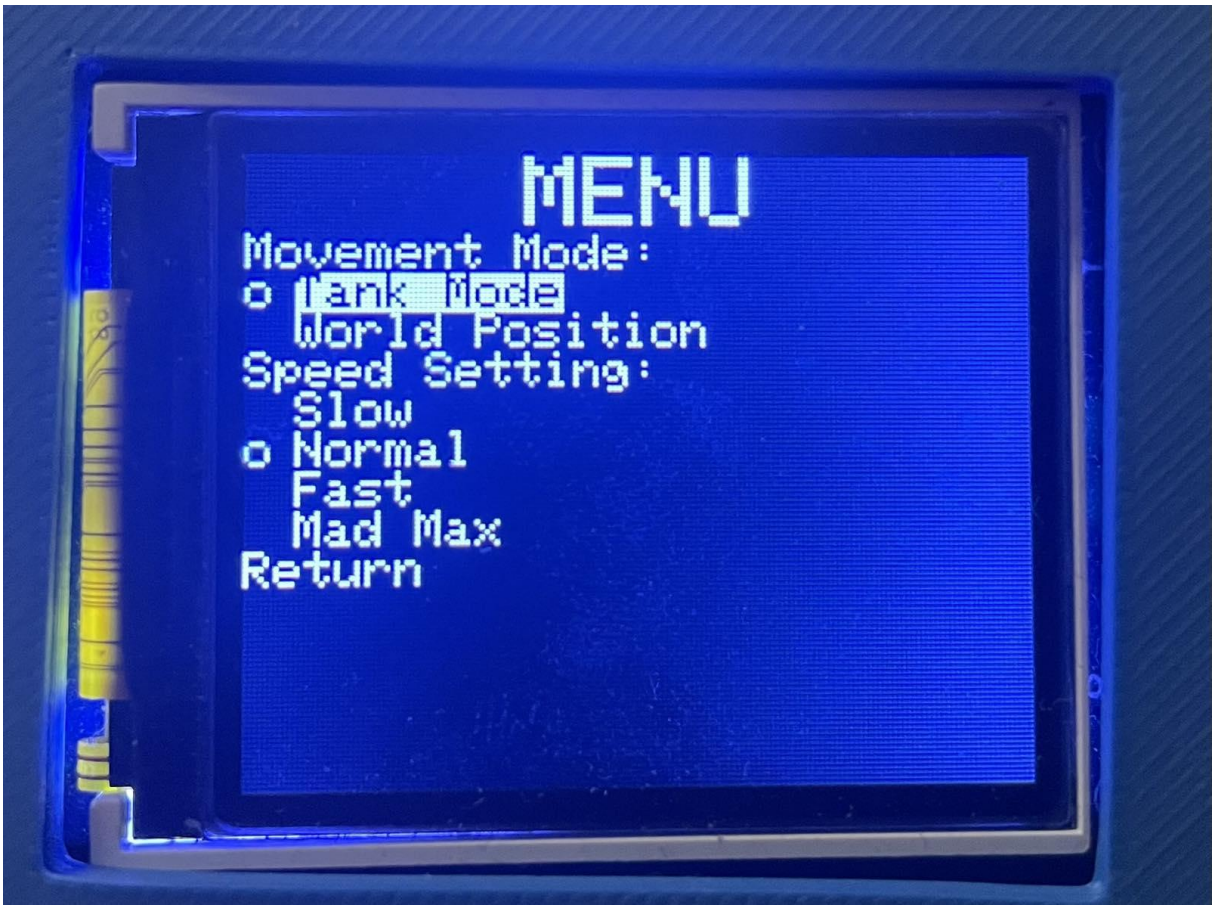
Σχήμα 83: Αριστερόστροφης περιστροφή μέσω της συνάρτησης DirectionalMovement()

```
685       else{
686         if(world_pos_rotating){
687           world_pos_rotating = false;
688           contr_payload.wheel_speed[0] = 0;
689           contr_payload.wheel_speed[1] = 0;
690           Serial.print("Left motor Speed:");
691           Serial.println(contr_payload.wheel_speed[0]);
692           Serial.print("Right motor Speed:");
693           Serial.println(contr_payload.wheel_speed[1]);
694           return;
695         }
696         //Η ταχύτητα των αριστερών τροχών
697         contr_payload.wheel_speed[0] += increment_per_step[speed_mode_index];
698         //Η ταχύτητα των δεξιών τροχών
699         contr_payload.wheel_speed[1] += increment_per_step[speed_mode_index];
700         if(contr_payload.wheel_speed[0] > target_speed){
701           contr_payload.wheel_speed[0] = target_speed;
702           contr_payload.wheel_speed[1] = target_speed;
703         }
```

Σχήμα 84: Ευθύγραμμη κίνηση μέσω της συνάρτησης DirectionalMovement()

Τέλος, να σημειωθεί πως για την χρήση του Directional Movement κατά την αρχικοποίηση και το χειριστήριο και το όχημα πρέπει να έχουν τον ίδιο προσανατολισμό καθώς μέσα στον κώδικα μετράμε την μεταβολή της περιστροφής και όχι την περιστροφή καθαυτή. Αυτό συμβαίνει καθώς το MPU6050 από μόνο του δε μπορεί να προσφέρει μετρήσεις πραγματικής περιστροφής χωρίς ένα μαγνητόμετρο, το οποίο μετράει τα μαγνητικά πεδία της γης, παρέχοντας έναν απόλυτο άξονα περιστροφής.

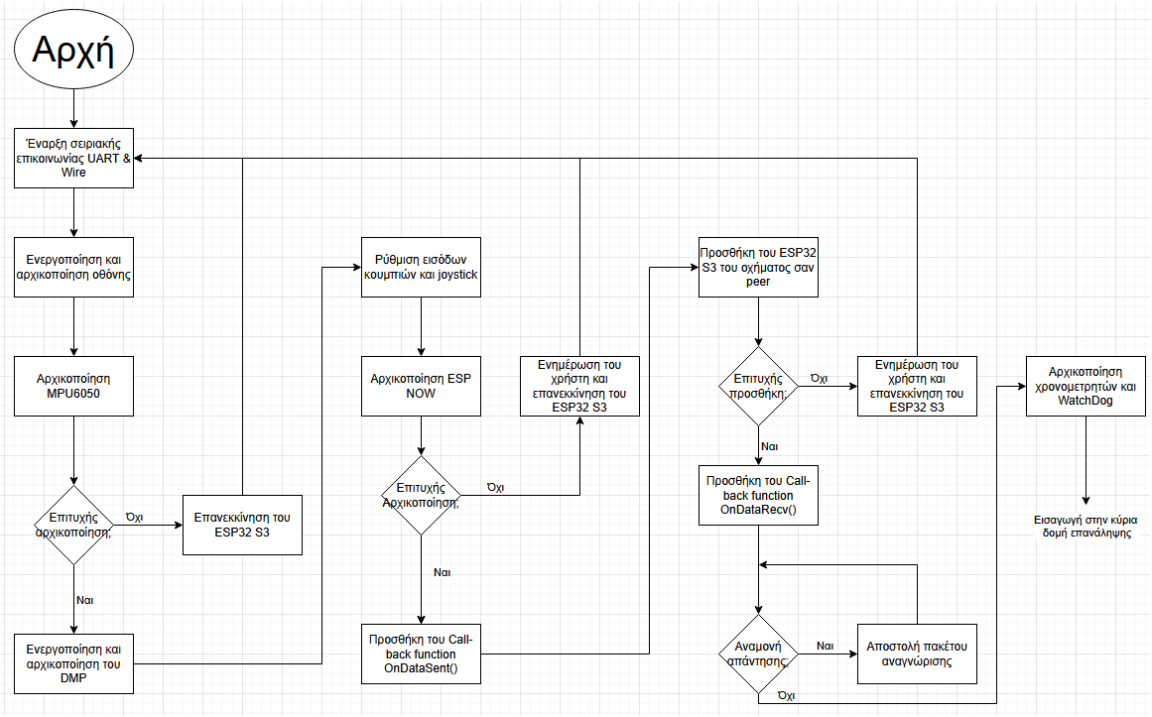
Ολοκληρώνοντας τον κώδικα του χειριστηρίου πρέπει να αναφέρουμε το μενού ρυθμίσεων το οποίο προστέθηκε και μπορεί να ενεργοποιηθεί πατώντας το κουμπί του joystick. Μέσα από αυτό το μενού ρυθμίσεων (Σχήμα 85) ο χρήστης έχει την δυνατότητα να επιλέξει αν θέλει ο υπολογισμός της ταχύτητας να γίνεται μέσω της συνάρτησης TankControlMovement (Tank Mode) ή μέσω της συνάρτησης DirectionalMovement (World Position). Επίσης προσφέρεται η δυνατότητα επιλογής ανάμεσα σε 4 ρυθμίσεις ταχύτητας που όπως αναφέραμε παραπάνω χωρίζονται στις επιλογές Slow, Normal Fast, Mad Max. Οι ρυθμίσεις αυτές επηρεάζουν την μέγιστη ταχύτητα του οχήματος μόνο όταν προχωράει ευθεία και όχι στην περιστροφή του. Η επιλογή οποιασδήποτε ρύθμισης γίνεται πάλι μέσω του κουμπιού του joystick ενώ για την επιστροφή στην αρχική οθόνη ο χρήστης πρέπει να πάει στην επιλογή Return.



Σχήμα 85: Το μενού ρυθμίσεων του χειριστηρίου

4.2 Το λογισμικό του τροχοφόρου οχήματος

Παραπάνω μελετήσαμε πως το χειριστήριο λαμβάνει τις εισόδους από τον χρήστη και τις μετατρέπει σε πακέτα επικοινωνίας. Σε αυτή την ενότητα θα αναλύσουμε πως το ESP32 S3 DevKit M1 που βρίσκεται πάνω στο ρομπότ λαμβάνει και διαχειρίζεται αυτά τα δεδομένα. Αρχικά παρουσιάζουμε το διάγραμμα ροής της αρχικοποίησης του κώδικα του τροχοφόρου (Σχήμα 86).



Σχήμα 86: Διάγραμμα ροής αρχικοποίησης του κώδικα οχήματος

Στον κώδικα εισάγουμε βιβλιοθήκες για την χρήση του πρωτοκόλλου ESP-NOW, την χρήση του πρωτοκόλλου I2C, την χρήση του MPU6050, την χρήση της μονάδας PCA9685 και την χρήση του ενσωματωμένου ασύγχρονου μετρητή επαγρύπνησης watchdog (Σχήμα 87).

```
1   #include <WiFi.h>
2   #include <esp_now.h>
3   #include <Wire.h>
4   #include <Adafruit_PWMServoDriver.h>
5   #include "I2Cdev.h"
6   #include "MPU6050_6Axis_MotionApps20.h"
7   #include <Adafruit_NeoPixel.h>
8   #include "esp_system.h"
9   #include "esp_task_wdt.h"
10
11  #define WDT_TIMEOUT 5
```

Σχήμα 87: Οι βιβλιοθήκες του κώδικα του οχήματος

Σε αυτό τον κώδικα κάνουμε χρήση της ασύγχρονης συνάρτησης onDataRecv(), η οποία όπως στον κώδικα του χειριστηρίου έτσι και εδώ εκτελείται όταν το ESP32 S3 λαμβάνει ένα πακέτο.

Κατά την λήψη του πακέτου το ESP32 S3 ενημερώνει, με βάση την τιμή των κουμπιών του χειριστηρίου, την περιστροφή των σερβοκινητήρων του ρομποτικού βραχίονα. Η μεταβολή της τιμής PWM των σερβοκινητήρων είναι σταθερή και την ορίζουμε ως μια μεταβλητή με όνομα SERVOSTEP. Για τον πλήρη έλεγχο της μέγιστης και της ελάχιστης γωνίας των σερβοκινητήρων ορίσαμε έναν πίνακα ο οποίος ενώνει 3 στοιχεία (Σχήμα 88), την ελάχιστη PWM τιμή του σερβοκινητήρα (άρα κατά επέκταση την ελάχιστη γωνία), την μέγιστη τιμή PWM και την τωρινή τιμή PWM. Επιπρόσθετα ορίσαμε 2 σταθερές την SERVOMIN και την SERVOMAX οι οποίες αντιστοιχούν στις τιμές 150 και 600 αντιστοίχως.

```
78  //Συνάρτηση μετατροπής γωνίας σε PWM
79  //Με βάση την ελάχιστη και μέγιστη τιμή PWM
80  double angleToPwm(double angle,double min, double max){
81    Serial.print("Turning Angle:");
82    Serial.println(angle);
83    return map(angle,0,180,min,max);
84  }
85  //Μεταβλητές PCA9685
86  Adafruit_PWMServoDriver pca = Adafruit_PWMServoDriver();
87  int servo_pwm[][3] = {{SERVOMIN,SERVOMAX,angleToPwm(90,SERVOMIN,SERVOMAX)},
88  {120,SERVOMAX,angleToPwm(100,120,SERVOMAX)},
89  {SERVOMIN,490,angleToPwm(90,SERVOMIN,490)},
90  {SERVOMIN,SERVOMAX,SERVOMIN},
91  {SERVOMIN,SERVOMAX,SERVOMIN},
92  {270,370,270}};
```

Σχήμα 88: Μεταβλητές PCA9685

Αφού κάνουμε λήψη του πακέτου, αντιγράψουμε τα δεδομένα του (Σχήμα 89):

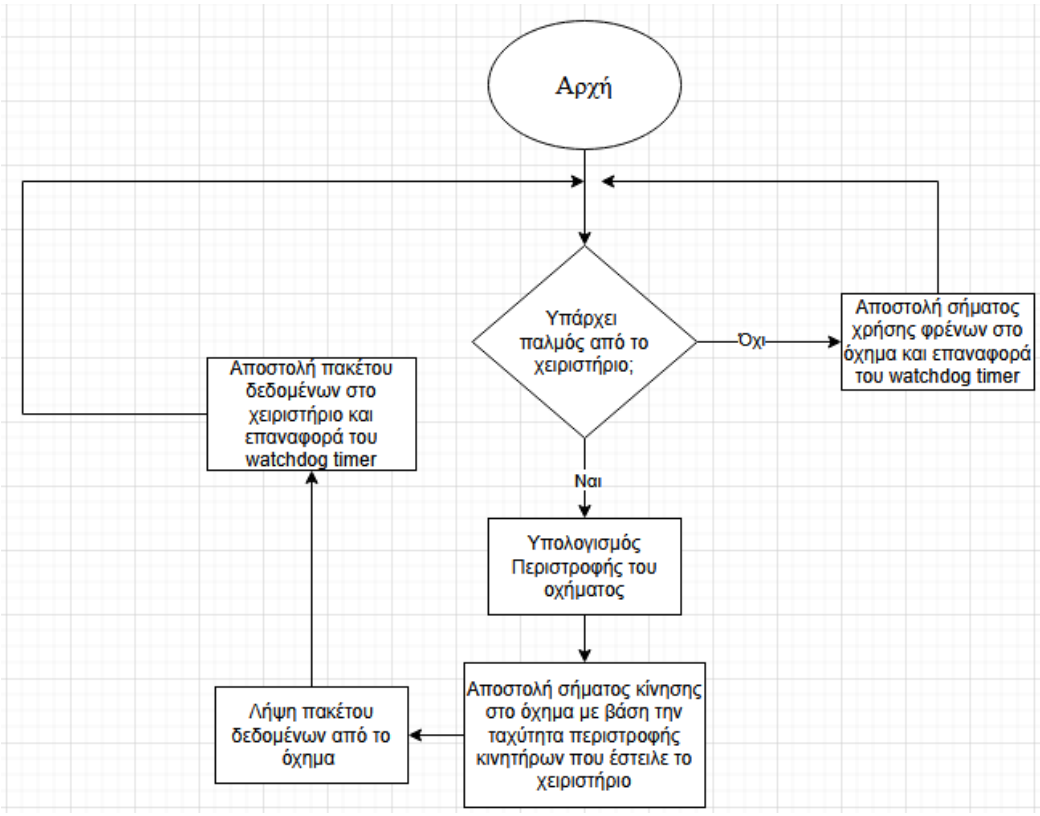
```
108   memcpy(&received_payload, incomingData, sizeof(received_payload));
109   int input1 = received_payload.buttons[1] - received_payload.buttons[0];
110   int input2 = received_payload.buttons[3] - received_payload.buttons[2];
111   int input3 = received_payload.buttons[5] - received_payload.buttons[4];
112   int input4 = received_payload.buttons[7] - received_payload.buttons[6];
113   int input5 = received_payload.buttons[9] - received_payload.buttons[8];
114   int input6 = received_payload.buttons[11] - received_payload.buttons[10];
```

Σχήμα 89: Αντιγραφή δεδομένων του πακέτου που λήφθηκε

Τα κουμπιά έχουν χωριστεί σε 6 ζευγάρια με κάθε ζευγάρι να αντιστοιχεί σε έναν σερβοκινητήρα. Μέσα στις τιμές input1-6 έχουμε αποθηκεύσει την τιμή του κάθε ζευγαριού. Η μεταβολή της τιμής PWM του κάθε σερβοκινητήρα εξαρτάται από την τιμή του αντίστοιχου input. Αρχικά ελέγχουμε αν η τιμή του input είναι διαφορετική του μηδενός (Σχήμα 90). Στην περίπτωση που είναι, πολλαπλασιάζουμε την τιμή του με την τιμή SERVOSTEP που έχουμε ορίσει ως 10 και προσθέτουμε το αποτέλεσμα στην τωρινή τιμή PWM του σερβοκινητήρα. Τέλος, περιορίζουμε την τιμή PWM ανάμεσα στα όρια που ορίσαμε στον πίνακα servo_pwm παραπάνω (Σχήμα 91).

```
127   if(input1 != 0){
128     servo_pwm[0][2] = constrain(servo_pwm[0][2] + input1 * SERVOSTEP, servo_pwm[0][0], servo_pwm[0][1]);
129     pca.setPWM(0,0,servo_pwm[0][2]);
130   }
131   if(input2 != 0){
132     servo_pwm[1][2] = constrain(servo_pwm[1][2] + input2 * SERVOSTEP, servo_pwm[1][0], servo_pwm[1][1]);
133     pca.setPWM(1,0,servo_pwm[1][2]);
134   }
135   if(input3 != 0){
136     servo_pwm[2][2] = constrain(servo_pwm[2][2] + input3 * SERVOSTEP, servo_pwm[2][0], servo_pwm[2][1]);
137     pca.setPWM(2,0,servo_pwm[2][2]);
138   }
139   if(input4 != 0){
140     servo_pwm[3][2] = constrain(servo_pwm[3][2] + input4 * SERVOSTEP, servo_pwm[3][0], servo_pwm[3][1]);
141     pca.setPWM(3,0,servo_pwm[3][2]);
142   }
143   if(input5 != 0){
144     servo_pwm[4][2] = constrain(servo_pwm[4][2] + input5 * SERVOSTEP, servo_pwm[4][0], servo_pwm[4][1]);
145     pca.setPWM(4,0,servo_pwm[4][2]);
146   }
147   if(input6 != 0){
148     servo_pwm[5][2] = constrain(servo_pwm[5][2] + input6 * SERVOSTEP, servo_pwm[5][0], servo_pwm[5][1]);
149     pca.setPWM(5,0,servo_pwm[5][2]);
150   }
```

Σχήμα 90: Υπολογισμός νέας τιμής PWM των σερβοκινητήρων



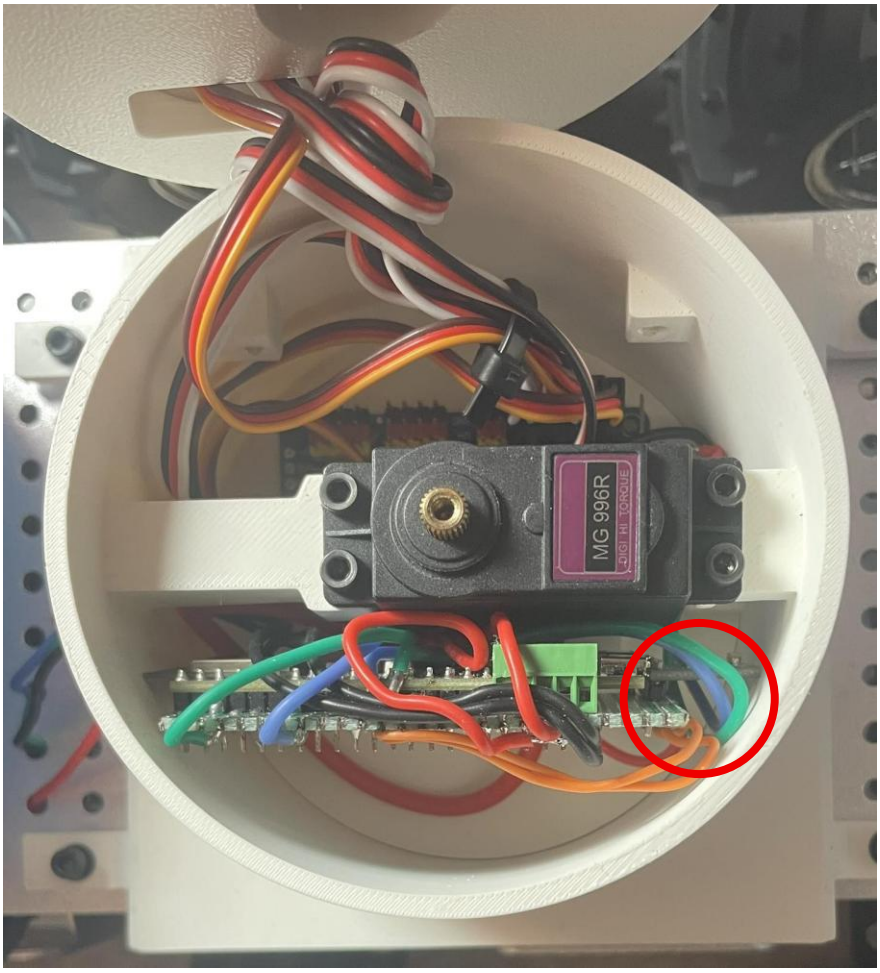
Στην κύρια δομή επανάληψης το ESP32 S3, υπολογίζουμε την περιστροφή του οχήματος, στέλνουμε πακέτα επικοινωνίας προς αυτό, κάνουμε λήψη πακέτων δεδομένων από αυτό και τέλος στέλνουμε κομμάτι αυτών των δεδομένων πίσω στο χειριστήριο. Παρακάτω (Σχήμα 85) παρέχεται το διάγραμμα ροής της κύριας δομής επανάληψης.

Σχήμα 91: Διάγραμμα ροής της κύριας δομής επανάληψης του ESP32 S3 του οχήματος

Στο παράρτημα Β παρέχονται περισσότερες πληροφορίες σχετικά με τις εντολές αποστολής και λήψης πακέτων από την πλακέτα ελέγχου του οχήματος. Για τον υπολογισμό της γωνίας περιστροφής χρησιμοποιήσαμε τις εντολές που μας παρέχει η βιβλιοθήκη για την χρήση του MPU6050 (Σχήμα 94). Η διαδικασία αποτελείται από τα ακόλουθα βήματα:

- Λήψη πακέτου περιστροφικών δεδομένων από το MPU6050
- Περιστροφή των δεδομένων στον X άξονα κατά 90º
- Λήψη δεδομένων βαρυτικής επιτάχυνσης και καθαρισμός των περιστροφικών δεδομένων
- Μετατροπή του τελικού αποτελέσματος σε γωνίες υπολογισμένες σε rad

Η περιστροφή των δεδομένων γύρω από τον άξονα X γίνεται καθώς η μονάδα MPU6050 του οχήματος έχει τοποθετηθεί κάθετα μέσα στην βάση του ρομποτικού βραχίονα όπως φαίνεται εντός του κόκκινου κύκλου στο σχήμα 92.



Σχήμα 92: Η θέση του MPU6050 του οχήματος

Τα περιστροφικά δεδομένα (quaternion) χρησιμοποιούμε για την περιγραφή γωνιών ενός διανύσματος στον τρισδιάστατο χώρο. Πριν μπορέσουμε να κάνουμε χρήση αυτών των τιμών πρέπει να φτιάξουμε τον προσανατολισμό τους. Αυτό επιτεύχθηκε πολλαπλασιάζοντας τα δεδομένα που έχουμε με ένα quaternion που περιγράφει περιστροφή γύρω από τον άξονα X

κατά 90º. Ο πολλαπλασιασμός γίνεται μέσα από την συνάρτηση του σχήματος 93 [8] η οποία επιστρέφει τα νέα περιστροφικά δεδομένα τα οποία έχουν προσανατολιστεί σωστά.

```
96   Quaternion Multiply_Quaternion(Quaternion q1, Quaternion q2){
97     return Quaternion(q1.w * q2.w - q1.x * q2.x - q1.y * q2.y - q1.z * q2.z,
98                       q1.w * q2.x + q1.x * q2.w + q1.y * q2.z - q1.z * q2.y,
99                       q1.w * q2.y - q1.x * q2.z + q1.y * q2.w + q1.z * q2.x,
100                      q1.w * q2.z + q1.x * q2.y - q1.y * q2.x + q1.z * q2.w
101    );
```

Σχήμα 93: Η συνάρτηση πολλαπλασιασμού περιστροφικών δεδομένων

Με αυτό τον τρόπο υπολογίζουμε την περιστροφή γύρω από τον κάθετο άξονα ή αλλιώς yaw. Η ίδια μεθοδολογία χρησιμοποιείται και στο χειριστήριο, χωρίς την διαδικασία προσανατολισμού, για τον υπολογισμό περιστροφής του. Καθώς οι μετρήσεις του DMP απαιτούν λίγα δευτερόλεπτα ώστε να σταθεροποιηθούν ορίσαμε ένα χρονικό περιθώριο στο οποίο λαμβάνουμε την μεταβολή της περιστροφής και την προσθέτουμε στην αρχική περιστροφή που υπολογίσαμε πολλαπλασιασμένη με έναν δείκτη ο οποίος έχει τεθεί στο 0.95 (Σχήμα 94). Η τιμή του δείκτη επιλέχθηκε έπειτα από πειραματισμό και επέφερε ικανοποιητική ακρίβεια στην αρχική τιμή περιστροφής.

```
250    if(mpu.dmpGetCurrentFIFOPacket(buffer)){
251      mpu.dmpGetQuaternion(&quaternion, buffer);
252      quaternion = Multiply_Quaternion(quaternion, Quaternion(cos(PI/4), sin(PI/4), 0, 0));
253      mpu.dmpGetGravity(&gravity, &quaternion);
254      mpu.dmpGetYawPitchRoll(ypr, &quaternion, &gravity);
255      current_yaw = -ypr[0] * 180/ M_PI;
256      if(millis() - start_timer < stabilization_time){
257        starting_yaw += 0.95 * (current_yaw - prev_yaw);
258        prev_yaw = current_yaw;
259      }
260      current_yaw = current_yaw - starting_yaw + 90;
261      current_yaw += ((current_yaw < 0) * 360);
262      current_yaw = int(current_yaw) % 360;
263      Serial.print("Car Yaw:");
264      Serial.println(current_yaw);
265      ack_payload.carYaw = current_yaw;
266    }
```

Σχήμα 94: Λήψη περιστροφικών δεδομένων και υπολογισμός yaw

4.3 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό είδαμε τις σημαντικότερες συναρτήσεις του κώδικα κάθε συστήματος όπως επίσης και τον τρόπο επικοινωνίας των 2 ESP32 S3. Στο επόμενο κεφάλαιο θα δώσουμε μετρήσεις σχετικά με τις δυνατότητες του οχήματος που προέκυψαν από πειράματα και προτεινόμενες βελτιώσεις των συστημάτων.

[8] Οι τύποι για τον υπολογισμό του πολλαπλασιασμού προήλθαν από την ιστοσελίδα του math works: https://www.mathworks.com/help/aeroblks/quaternionmultiplication.html

Κεφάλαιο 5. Συμπεράσματα και μελλοντικές βελτιώσεις

Στο παρόν σημείο, μελετάμε τις δυνατότητες του ρομποτικού μας συστήματος μέσα από πειραματισμούς. Περαιτέρω γίνεται μια σύντομη ανάλυση πιθανών βελτιώσεων πάνω σε αυτό αλλά και πιθανών μελλοντικών επεκτάσεων.

5.1 Μέτρηση δυνατοτήτων του συστήματος μέσω πειραμάτων

Για την δοκιμή του οχήματος επιλέξαμε να διεξάγουμε 2 πειράματα, το πρώτο αποτελείται από την ανύψωση ενός πλαστικού μπουκαλιού σε διάφορα βάρη. Κατά το δεύτερο πείραμα το όχημα κλήθηκε να συλλέξει και να μεταφέρει ένα αντικείμενο κάνοντας χρήση του ρομποτικού βραχίονα.. Στόχος μας μέσω αυτών των πειραμάτων είναι ο υπολογισμός της μέγιστης εμβέλειας ελέγχου του οχήματος, σε ανοιχτό πεδίο, ο υπολογισμός του μέγιστου βάρους ανύψωσης μέσω του βραχίονα, όπως επίσης και ο υπολογισμός του χρόνου συνεχούς λειτουργίας.

5.1.1 Πείραμα ανύψωσης

Κατά το πρώτο πείραμα ο ρομποτικός βραχίονας κλήθηκε να ανυψώσει 1 μπουκάλι νερό το οποίο γεμίσαμε με νερό ποικίλλων βαρών, στα 100 γραμμάρια, στα 200 γραμμάρια, στα 300 γραμμάρια, στα 400 γραμμάρια και στα 500. Για την αρχική ανύψωση του μπουκαλιού από το έδαφος ο βραχίονας έπρεπε αρχικά να εκταθεί πλήρως και αφού συλλέξει το αντικείμενο, να το μεταφέρει αριστερά ή δεξιά από το όχημα. Τα αποτελέσματα του πειράματος δείξαμε ότι ο βραχίονας ήταν ικανός να ανυψώσει και να μεταφέρει με μεγάλη ευκολία τα μπουκάλια βάρους 100 και 200 γραμμαρίων. Στα 300 και στα 400 γραμμάρια παρατηρήθηκε ότι ενώ ο βραχίονας ήταν ικανός να σηκώσει το βάρος από το έδαφος, δεν ήταν ικανός να ξεπεράσει το επίπεδο στο οποίο ήταν παράλληλος με το έδαφος όπου και η ροπή που ασκεί το βάρος είναι μεγαλύτερη. Το φαινόμενο αυτό παρατηρήθηκε ακόμα και όταν ξεπερνούσαμε αυτό το επίπεδο δίνοντας το μπουκάλι στον βραχίονα από μεγαλύτερο ύψος (Σχήμα 95). Ως αποτέλεσμα το αντικείμενο δε μπορούμε να ανυψωθεί παραπάνω, παρόλα αυτά ήταν εφικτή η περιστροφή του αντικειμένου αριστερά και δεξιά του οχήματος. Τέλος, στα 500 γραμμάρια η δαγκάνα δεν ήταν ικανή να κρατήσει το αντικείμενο οπότε δεν υπήρχε ανύψωση του. Παρακάτω παρέχεται ένας πίνακας μετρήσεων μέγιστου ύψους και βάρους ανύψωσης (Πίνακας 4)

Πίνακας 4: Πίνακας αποτελεσμάτων πειράματος ανύψωσης

Βάρος Μπουκαλιού σε γραμμάρια	Μέγιστη ανύψωση μπουκαλιού από το έδαφος σε εκατοστά	Ικανότητα μεταφοράς αντικειμένου αριστερά και δεξιά από το όχημα
100	60	Ναι
200	60	Ναι
300	20	Ναι
400	20	Ναι
500	0	Όχι

Σχήμα 95: Μέγιστη ανύψωση μεγάλου βάρους από τον βραχίονα

5.1.2 Πείραμα μεταφοράς αντικειμένου

Μέσα από αυτό το πείραμα θέλαμε να μελετήσουμε την ικανότητα του οχήματος να μεταφέρει αντικείμενα και να υπολογίσουμε την μέγιστη εμβέλεια της επικοινωνίας μεταξύ του χειριστηρίου και του οχήματος. Για τον υπολογισμό της απόστασης χρησιμοποιήσαμε την σελίδα του Google Maps, ορίζοντας ως αφετηρία την θέση του χρήστη και ως προορισμό την τελική θέση του ρομπότ. Για τα πρώτα 100 μέτρα η σύνδεση ήταν σταθερή με 0 διακοπές, έπειτα από τα 100 έως τα 200 μέτρα μετρήθηκαν συνολικά 2 διακοπές στην επικοινωνία, η οποία όμως αποκαταστάθηκε γρήγορα. Από τα 200 έως 220 μέτρα η επικοινωνία εμφάνισε τακτικές διακοπές και στα 220 μέτρα η επικοινωνία κόπηκε χωρίς αποκατάσταση. Καθ' όλη την διάρκεια του πειράματος το όχημα κουβαλούσε με τον ρομποτικό βραχίονα 1 μπουκάλι νερό βάρους 250 γραμμαρίων το οποίο κατάφερε και κράτησε μέχρι το τέλος. Επιπροσθέτως, μέσα από αυτό το πείραμα εκτιμήσαμε την διάρκεια ζωής της μπαταρίας. Η μπαταρία φορτίστηκε πλήρως πριν διεξαχθούν τα πειράματα ενώ κατά την ολοκλήρωση τους η τάση εξόδου ήταν 15.11V. Με βάση τον πίνακα χωρητικότητας που χρησιμοποιούμε, η μπαταρία έφτασε στο 50% της χωρητικότητας της ενώ η διάρκεια συνεχούς χρήσης ήταν 15 λεπτά. Ο συνολικός χρόνος εκτιμάται στα 20 με 25 λεπτά καθώς η πτώση της τάσης εξόδου της μπαταρίας θα ενεργοποιήσει τον κώδικα προστασίας της πλακέτας T'REX robot controller με αποτέλεσμα το όχημα να ακινητοποιηθεί ενώ η μπαταρία έχει ακόμα αρκετό φορτίο.

5.2 Συμπεράσματα

Τα ρομποτικά τροχοφόρα συστήματα εξοπλισμένα με ρομποτικό βραχίονα, για την αλληλοεπίδραση με το περιβάλλον, έχουν εισέλθει σε διάφορους κλάδους της ζωής μας από την ψυχαγωγία έως και τα επαγγέλματα μας, καθιστώντας τα προσβάσιμα από όλους.

Μέσα από αυτή την διπλωματική εργασία κατασκευάσαμε ένα ενσωματωμένο σύστημα για τον χειρισμό ενός τροχοφόρου οχήματος το οποίο έχει και δυνατότητες χειρισμού ρομποτικού βραχίονα. Το σύστημα είναι ικανό να μεταφέρει αντικείμενα χαμηλού βάρους σε ανισόπεδο έδαφος με εμβέλεια χειρισμού πάνω από 200 μέτρα.

Το τελικό αποτέλεσμα αυτής της εργασίας αποτελεί μια οικονομική εναλλακτική ενός ρομποτικού συστήματος με ρομποτικό βραχίονα παρέχοντας ικανότητες μελλοντικής βελτίωσης αναλόγως τις απαιτήσεις του χρήστη.

5.3 Πιθανές βελτιώσεις και μελλοντικές επεκτάσεις του συστήματος

Σε αυτή την ενότητα αναφέρουμε κομμάτια του συστήματος τα οποία μπορούν να βελτιωθούν καθώς και πιθανές επεκτάσεις των δυνατοτήτων του. Κάποιες από αυτές είναι:

1. Αύξηση της εμβέλειας: Η εμβέλεια επικοινωνίας μεταξύ του χειριστηρίου και του οχήματος μπορεί να αυξηθεί, με την χρήση κεραίας ή συστήματος ασύρματης επικοινωνίας υψηλής εμβέλειας FPV (First Person View), βελτιώνοντας την χρήση του συστήματος.
2. Προσθήκη κάμερας: Με την προσθήκη μιας κάμερας στο εμπρόσθιο μέρος του οχήματος το σύστημα αποκτάει δυνατότητες καταγραφής βίντεο ακόμα και αποστολή ζωντανής αναμετάδοσης στο χειριστήριο. Προϋπόθεση αυτού είναι η βελτίωση της οθόνης ώστε να υποστηρίζει την ποιότητα της αναμετάδοσης.
3. Προσθήκη αυτόματης μετακίνησης στον χώρο μέσω τεχνητής νοημοσύνης: Το ESP32 S3 προσφέρει δυνατότητες χρήσης τεχνητής νοημοσύνης. Οι χρήστες έχουν την δυνατότητα να κατασκευάσουν ένα σύστημα το οποίο θα καθοδηγεί το όχημα αυτόματα από το ένα σημείο στο άλλο, ενώ παράλληλα γίνεται χρήση του ρομποτικού βραχίονα για την μεταφορά υλικών.
4. Βελτίωση του ρομποτικού βραχίονα: Ο ρομποτικός βραχίονας κάνει χρήση μικρών σερβοκινητήρων για την αλληλοεπίδραση με το περιβάλλον. Προτείνεται χρήση μεγαλύτερων και πιο ισχυρών σερβοκινητήρων για την μεταφορά μεγαλύτερου φορτίου. Μπορούν επίσης οι χρήστες, μέσα από κώδικα, να προσθέσουν εξισώσεις αντίστροφης κινηματικής καθιστώντας ευκολότερη την χρήση του βραχίονα.
5. Προσθήκη μεγαλύτερης μπαταρίας: Για την αύξηση της αυτονομίας του οχήματος συνιστάται η προσθήκη μπαταρίας ίδιας ή μεγαλύτερης χωρητικότητας με την ίδια τάση. Προτείνεται επίσης η χρήση μίας δεύτερης πανομοιότυπης μπαταρίας συνδεδεμένης παράλληλα με την μπαταρία του οχήματος ώστε να μειωθεί το φαινόμενο πτώσης της τάσης εξόδου.
6. Χρήση μαγνητόμετρου: Η μονάδα MPU6050 έπειτα από εκτεταμένη χρήση εμφανίζει σφάλματα στις μετρήσεις του. Για τον μηδενισμό του σφάλματος συνιστάται η χρήση μίας μονάδας μαγνητόμετρου. Σε συνδυασμό με τις μετρήσεις του MPU6050 μπορούμε να υπολογίζουμε την περιστροφή του χειριστηρίου και του οχήματος χωρίς σφάλματα βελτιώνοντας την χρηστικότητα του συστήματος.

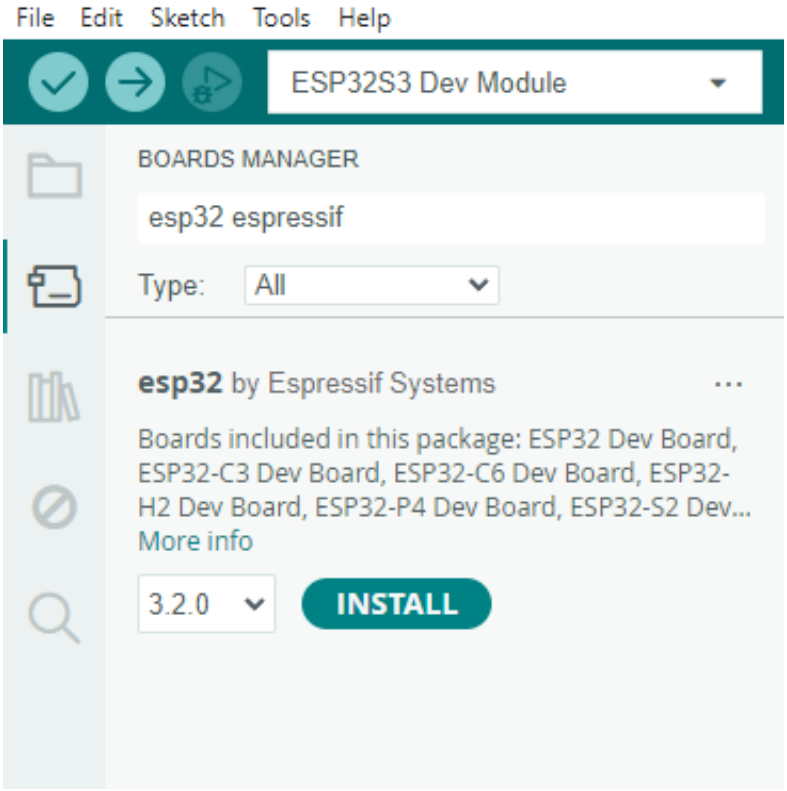
Οι παραπάνω βελτιώσεις θα αναβαθμίσουν σημαντικά το σύστημα μας αυξάνοντας την απόδοση, την χρηστικότητα, τις δυνατότητες και την αυτονομία του καθιστώντας το

μία ακόμα πιο ανταγωνιστική επιλογή στην αγορά των ρομποτικών συστημάτων με ρομποτικό βραχίονα.

Παράρτημα

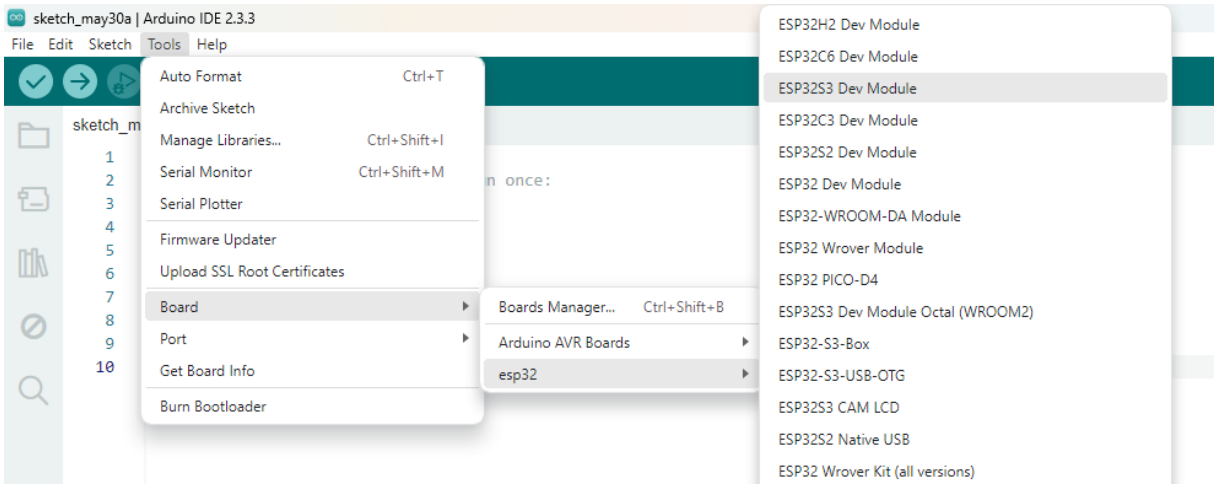
Παράρτημα Α – Προετοιμασία του περιβάλλοντος Arduino IDE για τον προγραμματισμό των ESP32 S3

Μέσα στο πρόγραμμα του Arduino IDE μπορούμε να επιλέξουμε ποιον μικροελεγκτή θα προγραμματίσουμε. Στο αριστερό μέρος του παραθύρου του προγράμματος υπάρχει μια μπάρα επιλογών, μία από αυτές τις επιλογές ονομάζεται board manager και μας δίνει την δυνατότητα να κατεβάσουμε βιβλιοθήκες έτοιμες για κάθε μικροελεγκτή. Στην μπάρα αναζήτησης θα γράψουμε “esp32 espressif” και θα κάνουμε εγκατάσταση το πακέτο το οποίο έχει προγραμματίσει η Espressif για τους μικροελεγκτές ESP32 (Σχήμα 96).



Σχήμα 96: Επιλογή του σωστού πακέτου μικροελεγκτών

Αφού κάνουμε την εγκατάσταση του σωστού πακέτου μικροελεγκτών μπορούμε να πάμε στις ρυθμίσεις και να επιλέξουμε την πλακέτα ESP32 που θέλουμε να προγραμματίσουμε, στην συγκεκριμένη περίπτωση θα επιλέξουμε την πλακέτα ESP32 S3 Dev Module (Σχήμα 97).



Σχήμα 97: Επιλογή της σωστής πλακέτας

Ο προγραμματισμός γίνεται στις γλώσσες προγραμματισμού C και C++ οι οποίες είναι και οι πιο διαδεδομένες όσο αναφορά τον προγραμματισμό μικροελεγκτών. Σε αυτό το σημείο είναι σημαντικό να κάνουμε μια αναφορά στις βιβλιοθήκες του Arduino IDE και συγκεκριμένα πως μπορούμε να προσθέσουμε δικές μας βιβλιοθήκες στην λίστα με τις προ εγκατεστημένες βιβλιοθήκες του προγράμματος. Κατά την εγκατάσταση του προγράμματος δημιουργείται στον υπολογιστή στην τοποθεσία των εγγράφων ένας φάκελος με το όνομα Arduino. Μέσα στον συγκεκριμένο φάκελο αποθηκεύονται όλα τα project που φτιάχνει ο χρήστης αλλά και οι βιβλιοθήκες που ο χρήστης κάνει εγκατάσταση στον φάκελο με όνομα libraries. Συνεπώς άμα επιθυμούμε να χρησιμοποιήσουμε δικές μας βιβλιοθήκες που δεν συμπεριλαμβάνονται στο λογισμικό Arduino IDE πρέπει να βάλουμε τα αρχεία μέσα σε εκείνο τον φάκελο, έπειτα μπορούμε να καλέσουμε την βιβλιοθήκη κανονικά όπως θα καλούσαμε κάθε άλλη βιβλιοθήκη “ #include<το όνομα της βιβλιοθήκης.h>”.

Παράρτημα Β – Δημιουργία συμβόλων στο λογισμικό Kicad

Για την δημιουργία συμβόλων στο πρόγραμμα Kicad ο χρήστης πρέπει να ανοίξει από το αρχικό μενού την επιλογή Symbol Editor. Έπειτα κάνοντας χρήση του εργαλείου σχεδίασης παραλληλογράμμων σχεδιάζει το περίγραμμα του συμβόλου. Στην συνέχεια με το εργαλείο τοποθέτησης επαφών, ο χρήστης μπορεί να τοποθετήσει επαφές πάνω στο σύμβολο, να τις ονομάσει και να δηλώσει τι τύπος επαφής είναι, είσοδος ή έξοδος τάσης, είσοδος ή έξοδος ή αμφίδρομή επαφή δεδομένων ή γείωση. Έπειτα πρέπει να δώσει ένα όνομα στο σύμβολο του και να το αποθηκεύσει. Έτσι το σύμβολο θα εμφανίζεται στην αναζήτηση συμβόλων του προγράμματος Kicad

Παράρτημα Γ – Επικοινωνία ESP32 S3 με το T'REX robot controller

Σύμφωνα με το εγχειρίδιο χρήσης όταν η πλακέτα λειτουργεί ως περιφερειακή συσκευή η διεύθυνση επικοινωνίας είναι η 0x07. Τα πακέτα επικοινωνίας πρέπει να έχουν ειδική μορφή μέσα στον κώδικα, συγκεκριμένα είναι 27 byte (Σχήμα 98).

1. Start byte – must be 0x0F (15 decimal)
2. PWM frequency – a number from 1 to 7 to select motor PWM frequency
3. Left motor speed high byte
4. Left motor speed low byte
5. Left motor brake – 0=brake off 1=brake on
6. Right motor speed high byte
7. Right motor speed low byte
8. Right motor brake – 0=brake off 1=brake on
9. Servo 0 position high byte
10. Servo 0 position low byte
11. Servo 1 position high byte
12. Servo 1 position low byte
13. Servo 2 position high byte
14. Servo 2 position low byte
15. Servo 3 position high byte
16. Servo 3 position low byte
17. Servo 4 position high byte
18. Servo 4 position low byte
19. Servo 5 position high byte
20. Servo 5 position low byte
21. Accelerometer de-vibrate 0-255 default=50
22. Impact sensitivity high byte
23. impact sensitivity low byte
24. lowbat high byte
25. lowbat low byte
26. I²C address 0-127
27. clock frequency – 0=100kHz 1=400kHz

Σχήμα 98: Τα πακέτα που λαμβάνει το T'REX robot controller

Τα byte 3-4 και 6-7 μας επιτρέπουν να ελέγχουμε την ταχύτητα περιστροφής των κινητήρων μας, οι τιμές που δέχονται είναι από -255 έως 255 με τις αρνητικές τιμές να συμβολίζουν αντίθετη φορά περιστροφής. Τα byte 5 και 8 χρησιμοποιούνται για την ενεργοποίηση ηλεκτρικών φρένων, αν η τιμή τους είναι 1 ενεργοποιούνται. Τα byte 9 έως 20 δε μας ενδιαφέρουν εφόσον δεν χρησιμοποιούμε σερβοκινητήρες αλλά κινητήρες από τις εξόδους τάσης (channels). Το σύστημα έχει έλεγχο για κρούσεις, σε περίπτωση μίας επειδή το σασί μπορεί ακόμα να δονείται μπορούμε να ορίσουμε έναν χρόνο στον οποίο η ανίχνευση κρούσεων απενεργοποιείται για να μην έχουμε ψευδής ανίχνευση. Ο χρόνος ορίζεται ως την τιμή που εισάγουμε στο byte 21 επί 2mS (millisecond). Η αρχική τιμή είναι 50 άρα συνολικά 100mS. Για την προστασία της μπαταρίας η πλακέτα μας προσφέρει 2 byte για τον έλεγχο της τάσης εισόδου, αν η τάση της μπαταρίας είναι χαμηλότερη από αυτή που ορίζουμε τότε το σύστημα δεν αφήνει τους κινητήρες να κουνηθούν. Τα byte 26 και 27 χρησιμοποιούνται από το πρωτόκολλο I2C και είναι η διεύθυνση επικοινωνίας και η ταχύτητα του ρολογιού, εμείς θα κάνουμε χρήση των τυπικών τιμών που είναι 0x07 για την διεύθυνση επικοινωνίας και 100kHz για την συχνότητα των παλμών του ρολογιού.

Η πλακέτα μας προσφέρει την δυνατότητα λήψης πακέτων με χρήσιμες πληροφορίες μέσω του πρωτόκολλου I2C. Τα πακέτα αυτά έχουν μέγεθος 24 byte και περιέχουν έχουν την ακόλουθη μορφή (Σχήμα 99):

1. Start byte – will be 0xF0 (240 decimal)
2. Error flag – 0 indicates the last command packet was received ok
3. Battery voltage high byte
4. Battery voltage low byte
5. Left motor current high byte
6. Left motor current low byte
7. Left encoder count high byte
8. Left encoder count low byte
9. Right motor current high byte
10. Right motor current low byte
11. Right motor encoder count high byte
12. Right motor encoder count low byte
13. Accelerometer X-axis high byte
14. Accelerometer X-axis low byte
15. Accelerometer Y-axis high byte
16. Accelerometer Y-axis low byte
17. Accelerometer Z-axis high byte
18. Accelerometer Z-axis low byte
19. Impact X-axis high byte
20. Impact X-axis low byte
21. Impact Y-axis high byte
22. Impact Y-axis low byte
23. Impact Z-axis high byte
24. Impact Z-axis low byte

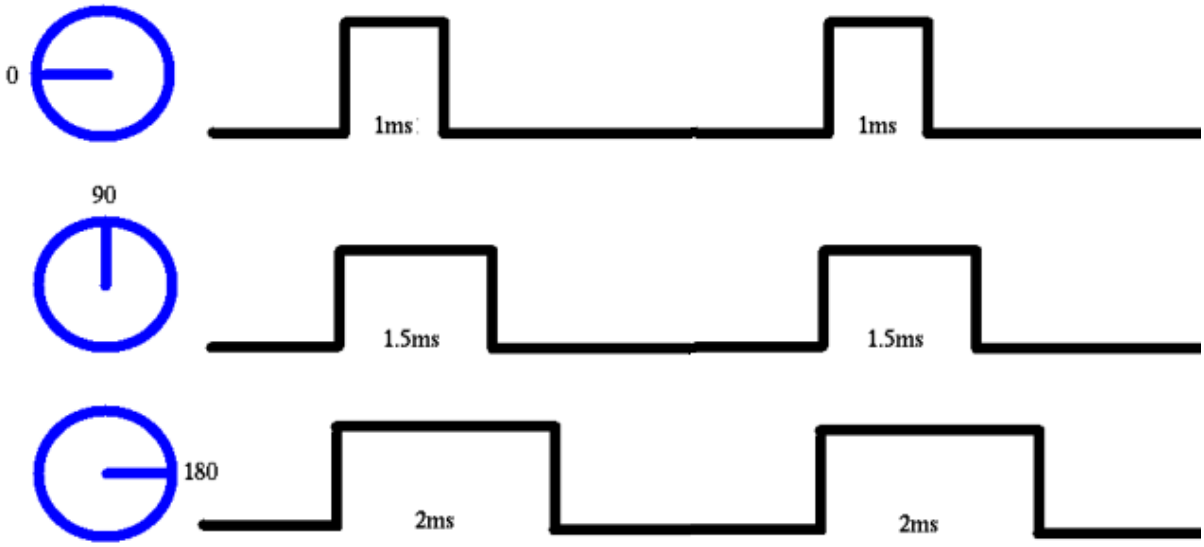
Σχήμα 99: Η δομή του πακέτου δεδομένων από το T'REX robot controller

Τα byte 2 και 3 μας δίνουν την τάση της μπαταρίας, καθώς οι μπαταρίες όσο αποφορτίζουν μειώνουν την τάση που παρέχουν, εδώ πρέπει να σημειωθεί πως η τάση της μπαταρίας μειώνεται προσωρινά περισσότερο από το πραγματικό και όταν τραβάμε πολύ ρεύμα από αυτή. Τα byte 5 6 και 9 10 μας δείχνουν πόσο είναι το συνολικό ρεύμα που τραβάει η κάθε έξοδος τάσης (channel). Τα byte 13 έως 18 είναι μετρήσεις ενός επιταχυνσιόμετρου ενσωματωμένου στην πλακέτα. Τέλος, τα byte 19 έως 24 χρησιμοποιούν τις μετρήσεις του επιταχυνσιόμετρου και στην περίπτωση που ανιχνευτή σύγκρουση του οχήματος καταγράφουν την αλλαγή στην επιτάχυνση πριν και μετά την κρούση.

Η πλακέτα μας προσφέρει 2 εντολές για την αποστολή και λήψη αυτών των πακέτων. Η εντολή για την αποστολή δεδομένων ονομάζεται MasterSend() και δέχεται τα στοιχεία που αναλύσαμε στο σχήμα 98. Η δεύτερη εντολή για την λήψη δεδομένων ονομάζεται MasterReceive() και μας επιστρέφει τα δεδομένα τα οποία αναλύσαμε στο σχήμα 99.

Παράρτημα Δ – Η λειτουργία σερβοκινητήρων

Οι σερβοκινητήρες έχουν μια συχνότητα παλμού, συνήθως 50Hz, κατά την οποία διαβάζουν την τάση εισόδου. Όταν θέλουμε να κουνήσουμε έναν σερβοκινητήρα πρέπει να αλλάξουμε την τάση στα άκρα του, αυτή η διαδικασία γίνεται συνήθως με την χρήση μεταβλητών αντιστάσεων, δημιουργώντας έτσι απώλειες. Η τεχνολογία PWM[27] μας επιτρέπει να ελέγχουμε τους σερβοκινητήρες πιο αποτελεσματικά και με μεγαλύτερη ακρίβεια μέσω παλμών. Οι παλμοί αυτοί έχουν ως ελάχιστη τιμή το 0 και ως μέγιστη την τιμή που επιλέγουμε να παρέχουμε εμείς, συνήθως 3.3V ή 5V. Έπειτα μπορούμε να αλλάξουμε το πλάτος του παλμού σε σχέση με τον παλμό του σερβοκινητήρα. Μας δίνεται έτσι η δυνατότητα να επηρεάσουμε την μέση τάση που ο σερβοκινητήρας βλέπει στην είσοδο του και κατά επέκταση την περιστροφή του.



Σχήμα 100: Η επιρροή του PWM στην γωνία του σερβοκινητήρα

Οι περισσότεροι σερβοκινητήρες είναι προγραμματισμένοι έτσι ώστε ένας παλμός διάρκειας 1ms να ισούται με 0^o, 1.5ms να ισούται με γωνία 90^o και 2ms με 180^o (Σχήμα 100[9]). Η ανάλυση των 12 bit σημαίνει ότι η μονάδα μας προσφέρει 4096 τιμές μεταξύ του 1ms και των 2ms ώστε να έχουμε μεγάλη ακρίβεια στην θέση περιστροφής του σερβοκινητήρα μας.

[9] Η εικόνα προήλθε από: https://electronics.stackexchange.com/questions/346603/driving-servo-motor-with-pwm-signal

Παράρτημα Ε - Κατάλογος υλικών

Παρακάτω παρέχεται αναλυτικός πίνακας με όλα τα υλικά που χρησιμοποιήθηκαν για την κατασκευή του οχήματος και του χειριστηρίου.

Πίνακας 5: Πίνακας υλικών των 2 συστημάτων

Υλικό	Ποσότητα	Εκτιμώμενο Κόστος Μονάδας
Σασί της εταιρείας Dagu	1	360€
T'REX robot controller	1	60€
ESP32 S3 DevKit C1	1	20€
ESP32 S3 DevKit M1	1	20€
Μπαταρία Tattu 14.8V 1800mAh	1	29€
SSR-40DD	1	8€
Emergency Stop Button 12V	1	7€
Wire 24 AWG	3 μέτρα	4€
Wire 16-18 AWG	1	2€
Σχέδια ρομποτικού βραχίονα	1	20€
Σερβοκινητήρας MG996R	3	5€
Σερβοκινητήρας MG90S	3	4€
MPU6050	2	3.50€
Απτικό Κουμπί	12	0.15€
Joystick Module	1	4€
1.8 Inch TFT Display	1	8€
Μονάδα φόρτισης μπαταριών 18650	1	12€
Κουμπί στιγμιαίας χρήσης	1	0.50€
DC/DC Step Down Converter	1	10
PCB χειριστηρίου	1	7.50€

Βιβλιογραφία

[1] Clear Path Robotics, 'TurtleBot3'. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/features/#specifications

[2] Robotis, 'Robotis Manipulator'. [Online]. Available: https://emanual.robotis.com/docs/en/platform/openmanipulator_x/specification/#hardware-specification

[3] Taurob, 'Inspector V8'. [Online]. Available: https://www.taurob.com/wp-content/uploads/products/Inspector-V8_folder.pdf

[4] Clear Path Robotics, 'Husky A300'. [Online]. Available: https://clearpathrobotics.com/husky-a300-unmanned-ground-vehicle-robot/

[5] Franka, 'Franka Research 3'. [Online]. Available: https://franka.de/franka-research-3

[6] Arduino, 'Arduino Nano'. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf

[7] Raspberry Pi, 'Raspberry Pi 4'. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[8] Espressif, 'ESP32 S3'. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

[9] ST Electronics, 'Blue Pill'. [Online]. Available: https://stm32-base.org/boards/STM32F103C8T6-Blue-Pill.html

[10] Espressif, 'ESP-NOW'. [Online]. Available: https://www.espressif.com/en/solutions/low-power-solutions/esp-now

[11] Akanksha Soni, 'System on a Chip: How Smaller, Faster Devices are Made'. [Online]. Available: https://www.ansys.com/blog/what-is-system-on-a-chip

[12] Espressif, 'ESP32 S3 DevKit C1'. [Online]. Available: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitc-1/index.html

[13] Espressif, 'ESP32 S3 DevKit M1'. [Online]. Available: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitm-1/index.html

[14] Espressif, 'ESP32 S3 WROOM1'. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[15] Espressif, 'ESP32 S3 MINI1'. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-mini-1_mini-1u_datasheet_en.pdf

[16] Arduino, 'Arduino IDE'. [Online]. Available: https://www.arduino.cc/en/about/#what-is-arduino

[17] Kicad, 'Kicad'. [Online]. Available: https://www.kicad.org/

[18] Γιάννης Πλευριτάκης, ‘Το πρωτόκολλο I2C’. [Online]. Available: https://learnelectronics.gr/to-πρωτόκολλο-επικοινωνίας-i2c/
[19] Dagu, ‘Wild Thumper 6WD’. [Online]. Available: https://www.pololu.com/product/1561
[20] XSENS, ‘Gimbal Lock’. [Online]. Available: https://base.movella.com/s/article/Understanding-Gimbal-Lock-and-how-to-prevent-it?language=en_US
[21] GeekMomProject, ‘Χρήση DMP, τα πλεονεκτήματα και τα μειονεκτήματα’. [Online]. Available: https://www.geekmomprojects.com/mpu-6050-dmp-data-from-i2cdevlib/
[22] Adafruit, ‘PCA9685’. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf
[23] Fotek, ‘Fotek SSR 40DD Datasheet’. [Online]. Available: https://www.alldatasheet.net/datasheet-pdf/view/1132471/FOTEK/SSR-40DD-H.html
[24] Addicore, ‘TP4056 Datasheet’. [Online]. Available: https://grobotronics.com/images/companies/1/datasheets/TP4056%20Datasheet.pdf?1530184261538
[25] Mouser, ‘1N5817’. [Online]. Available: https://gr.mouser.com/datasheet/2/389/1n5817-1848842.pdf
[26] Tobsun, ‘12/24V – 5V 15A Stepdown Converter’. [Online]. Available: https://www.hellasdigital.gr/electronics/boost-and-buck-converters/dc-dc-converter-12v-24v-to-5v-15a/
[27] Sparkfun, ‘Pulse Width Modulation (PWM)’. [Online]. Available: https://learn.sparkfun.com/tutorials/pulse-width-modulation/all

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

PCB	Printed Circuit Board
SPST	Single Poll Single Throw
I2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
IDE	Integrated Development Environment
SoC	System On a Chip
IMU	Inertial Measurement Unit
PWM	Pulse-Width Modulation
DC	Direct Current
SCL	Serial Clock
SDA	Serial Data
POCI	Peripheral Out Controller In
PICO	Peripheral In Controller Out
GND	Ground
VCC	Voltage Common Collector
RAM	Random Access Memory
ROM	Read Only Memory
LiPo	Lithium Polymer
Li Ion	Lithium Ion
TFT	Thin-Film Transistor
LCD	Liquid-Crystal Display
GPIO	General Purpose Input Output
DMP	Digital Motion Process

Απόδοση Ξενόγλωσσων Όρων

Απτικό	Tactile
Αλγόριθμός	Algorithm
Λογισμικό	Software
Υλικό	Hardware
Πακέτο	Payload
Σύστημα σε τσιπ	SoC
Δομή επανάληψης	Loop
Δομή αρχικοποίησης	Setup
Διάγραμμα ροής	Flow chart
Μνήμη τυχαίας προσπέλασης	RAM
Μνήμη μόνο για ανάγνωση	ROM
Μοχλός αντίχειρα	Joystick
Τρανζίστορ λεπτού στρώματος	TFT
Οθόνη υγρών κρυστάλλων	LCD
Πλακέτα τυπωμένου κυκλώματος	PCB
Γενική χρήσης είσοδος έξοδος	GPIO