

**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ
СИСТЕМИ към ТЕХНИЧЕСКИ
УНИВЕРСИТЕТ - СОФИЯ**

КУРСОВ ПРОЕКТ

по “Вградени микрокомпютърни системи”, 11 клас

Тема: Изграждане на Battle Bot

Изготвил:

Александър Мечкаров
Димитър Желев

Проверил:

Росен Витанов

СОФИЯ
2024

Съдържание

Съдържание.....	2
Увод.....	4
Проучване.....	5
1.1 Видове Battle Bots.....	5
1.1.1 Vertical/Horizontal Spinner Bots.....	5
1.1.2 Drum Bots.....	6
1.1.3 Crusher Bots.....	7
1.1.4 Flipper Bots.....	7
1.1.5 Wedge Bots.....	8
1.1.6 Hybrid Bots.....	9
1.2 Начин на управление.....	9
1.2.1 Radio Frequency (RF) control.....	9
Блокова схема.....	11
Електрическа схема.....	12
Избор на компоненти.....	13
1. ESP32.....	13
2. PlayStation 4 Controller.....	14
3. Cytron MDD20A.....	15
4. GB37 550 High Speed Carbon Brush DC Gear Motor.....	16
5. Акумулатор 12V/14Ah HIGH RATE LAVA.....	17
6. Реле.....	18
7. Оръжие.....	19
8. Мотор 100W 12V.....	19
9. Ъглошлайф приставка за бормашина Елтос.....	20
10. Зарядно за автомобил 12V.....	21
11. Kill Switch.....	21
12. Предпазител.....	22
Изработка на проекта.....	24
1. Изработка на корпуса.....	24
2. Поставяне на компонентите.....	26
Изработка на софтуер.....	29
1. Видове управление.....	29
2. Използвани библиотеки.....	29
3. Логика зад кода.....	30
3.1 Установяване на връзка между контролера и ESP32.....	30
3.2 Тестване на връзката.....	32
3.3 Управление на задвижването.....	32
3.4 Управление на оръжието.....	34
Целият код може да бъде намерен в GitHub хранилището на проекта. [4].....	35

Допълнителна литература.....	36
-------------------------------------	-----------

Увод

Много преди блясъкът на BattleBots™ на Comedy Central да донесе масова привлекателност на спорта, битката с роботи е имала много по-скромно начало. Началото на спорта датира през октомври 1989 г. в най-добрата среда за безумни неща: конгрес на научната фантастика. MileHiCon — конвенция, която се провежда от 1969 г. в Денвър — включва първия истински турнир, популяризиращ роботизирани битки, наречен Critter Crunch.

Според статии, намерени в интернет, малка група машинни инженери, известна като Denver Mad Scientist Club, е предвидила събитието, след като са изгледали видеоклипове от групата за роботизирано изкуство, известна като Survival Research Laboratory, и са научили за състезание в MIT, което изисква домашни роботи да се състезават в механични задачи като събиране на топки за пинг-понг. Чрез свързването на тези влияния с вече съществуващия Critter Crunch в MileHiCon, групата създава бойно събитие с роботи като никое друго до сега.

Проучване

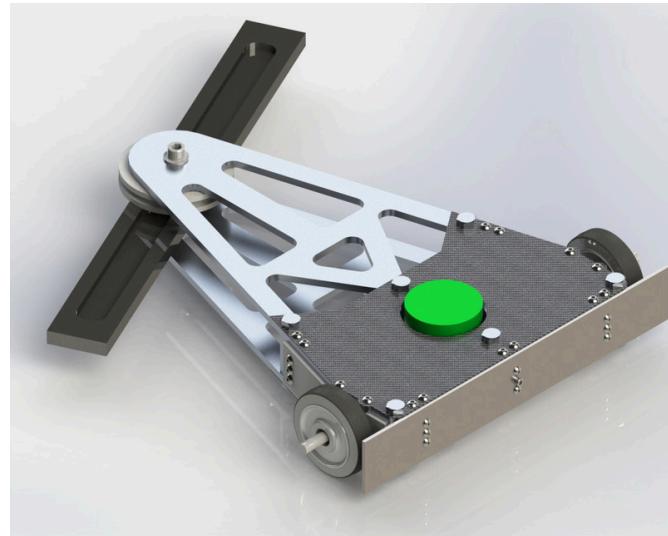
1.1 Видове Battle Bots

1.1.1 Vertical/Horizontal Spinner Bots

Spinner ботът е кралят на атаката и защитата, ако приемем, че остане жив. За да се класифицира боен бот като Spinner, външната страна на робота се върти или цялата горна въртене. Неговото нападение е неговата защита, което гарантира, че при нападение от опонента той също ще понесе щети. Моментната сила е ключова при нанасянето на големи щети и водачът трябва да умее правилно да маневрира през арената, изчаквайки Spinner-а да развие максимална скорост. Spinner ботовете са разрушителни както за конкуренцията, така и за самите себе си, затова се изисква добра поддръжка и много резервни части.



Фиг. 1.1.1a - Vertical Spinner



Фиг. 1.1.1b - Horizontal Spinner

1.1.2 Drum Bots

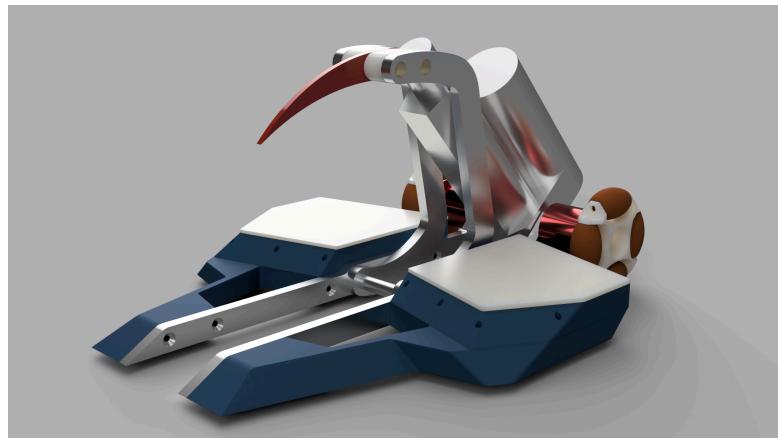
Drum ботовете са “heavy hitters” от типовете Battlebots. Начинът, по който работи Drum бота, е, че има огромна въртяща се маса отпред. Ако барабанът е проектиран правилно, той ще има способността да хвърля други Battle ботове във въздуха или отстрани на стената на арената. Въпреки това, поради тежката маса, въртяща се с високи обороти, тези роботи могат да бъдат трудни за управление. Причината за това е, че високият момент на инерция ще накара бота да иска да се преобрънне, докато завива.



Фиг.1.1.2 - Drum Bot

1.1.3 Crusher Bots

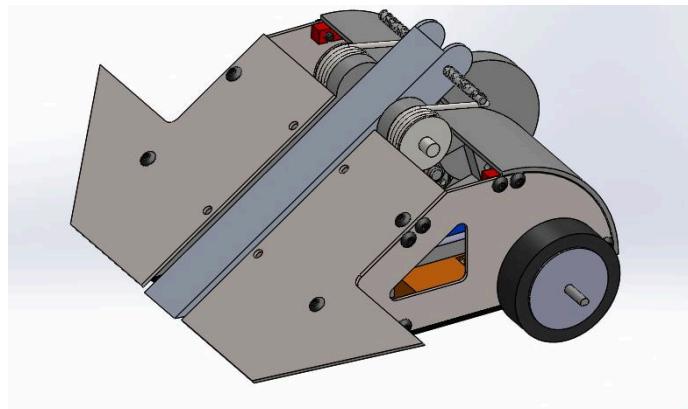
Crusher ботовете имат менгеме, подобно на трошачка, проектирано да държи и евентуално да огъва рамката на опонента. Необходимо е да се отдели много време за упражняване на шофирането, тъй като не само ще трябва да се управлява робота, но и да се добие усет за атакуването в правилния момент, за да хванете другия бот. Освен това водачът трябва да е добре запознат с правилата на мача, относно ограничението на времето, през което можете да задържате опонента си, преди да трябва да бъде пуснат.



Фиг. 1.1.3 - Crusher Bot

1.1.4 Flipper Bots

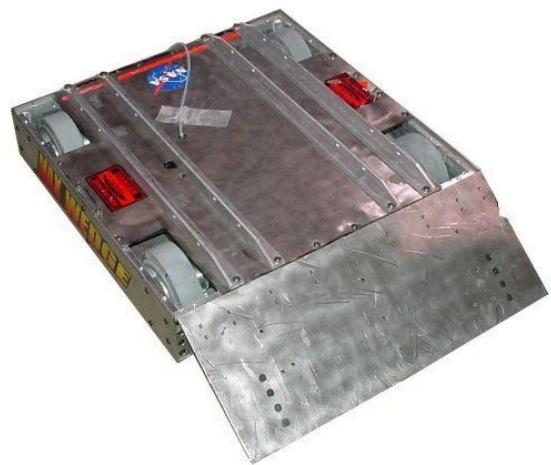
Flipper ботът има ръка, обикновено свързана с мощна пневматична система, която се използва за обръщане на опонента. Подобно на трошачката, нужен е добър шофьор, който има добри рефлекси, за да работи с този бот. Ако флиперът работи от пневматична система, тогава има ограничен период от време, в който можете да се използва рамото на флипера, преди въздухът в резервоарите да свърши.



Фиг. 1.1.4 - Flipper Bot

1.1.5 Wedge Bots

Wedge ботът е проектиран с наклонена равнина отпред. Целият смисъл на това е да даде възможност на Battle бота да вдигне опонента си и да го забие в стената на арената или в някаква друга опасност. Този робот обикновено е много здрав структурно, което го прави много труден за унищожаване. Въпреки това, поради липсата на оръжие, те не винаги нанасят много щети на опонента си, те също изискват опитен шофьор, който да ги управлява.



Фиг. 1.1.5 - Wedge Bot

1.1.6 Hybrid Bots

Хибридите комбинират изброените по-горе стилове, позволявайки допълнителни дизайнерски характеристики, които могат да се използват като вторично оръжие. Най-големият проблем при това е, че ще увеличи времето за процес на проектиране и ще трябва да се обърне повече внимание на теглото. Освен това е трудно да се поставят вторични функции на спинер.

1.2 Начин на управление

Дистанционното управление е най-разпространеният метод за управление, при който човешки оператор използва ръчно устройство, за да изпраща команди до робота, насочвайки неговите движения и действия. Ръчното устройство обикновено включва джойстици или други контроли, които позволяват на оператора да движи робота и да управлява оръжията му.

1.2.1 Radio Frequency (RF) control

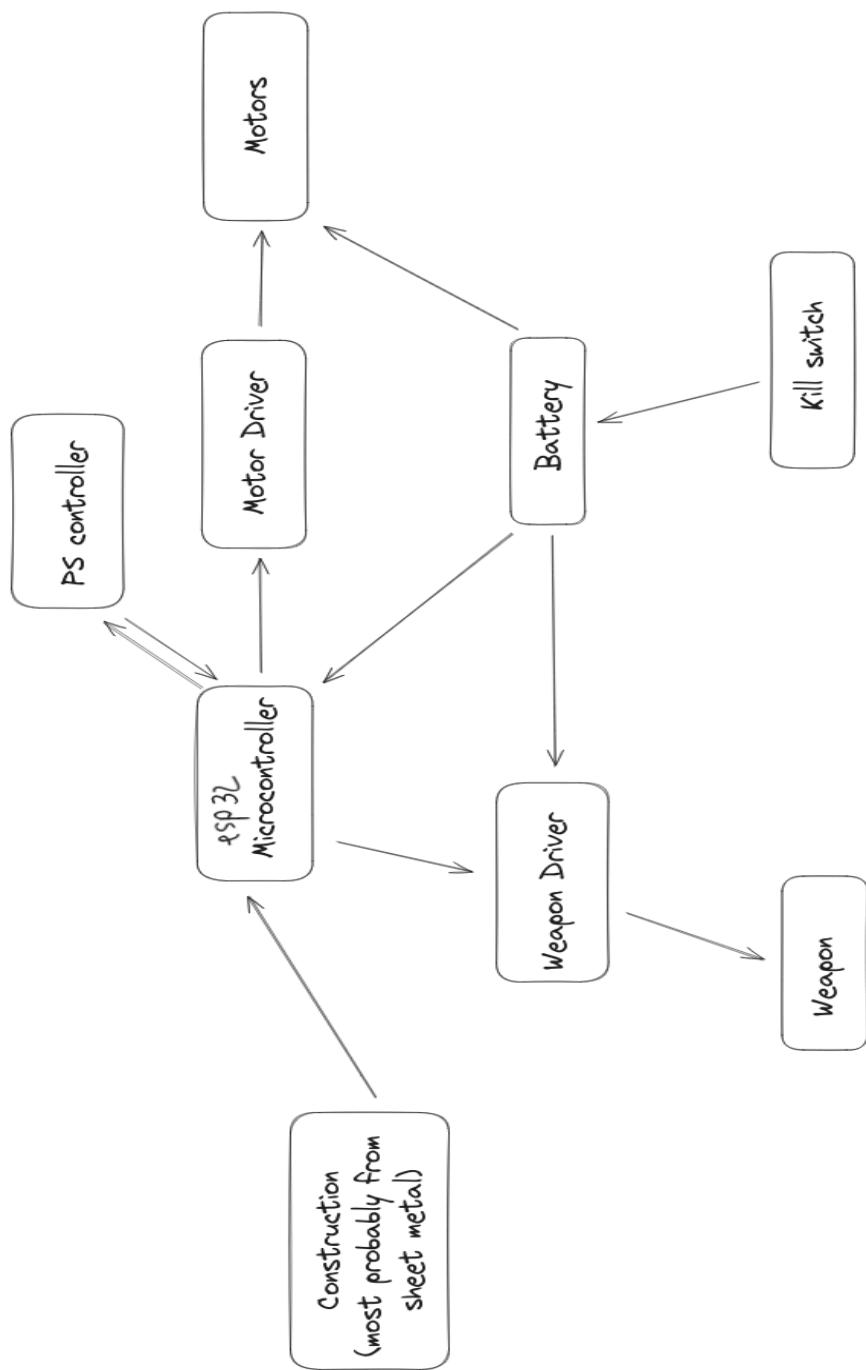
За приемане на радиосигнали трябва да се използва антена. Въпреки това, тъй като антената ще улавя хиляди радиосигнали наведнъж, е необходим радиотунер, за да се настрои на определена честота (или честотен диапазон). Това обикновено се прави чрез резонатор – в най-простата му форма, верига с кондензатор и индуктор образуват настроена верига. Резонаторът усилва трептенията в определена честотна лента, като същевременно намалява трептенията на други честоти извън тази лента.

Frequency	Wavelength	Designation
3 – 30 Hz	10^5 – 10^4 km	Extremely low frequency
30 – 300 Hz	10^4 – 10^3 km	Super low frequency
300 – 3000 Hz	10^3 – 100 km	Ultra low frequency
3 – 30 kHz	100 – 10 km	Very low frequency
30 – 300 kHz	10 – 1 km	Low frequency
300 kHz – 3 MHz	1 km – 100 m	Medium frequency
3 – 30 MHz	100 – 10 m	High frequency
30 – 300 MHz	10 – 1 m	Very high frequency
300 MHz – 3 GHz	1 m – 10 cm	Ultra high frequency
3 – 30 GHz	10 – 1 cm	Super high frequency
30 – 300 GHz	1 cm – 1 mm	Extremely high frequency
300 GHz - 3000 GHz	1 mm - 0.1 mm	Tremendously high frequency

Фиг. 1.2.1 - Radio Communication Range

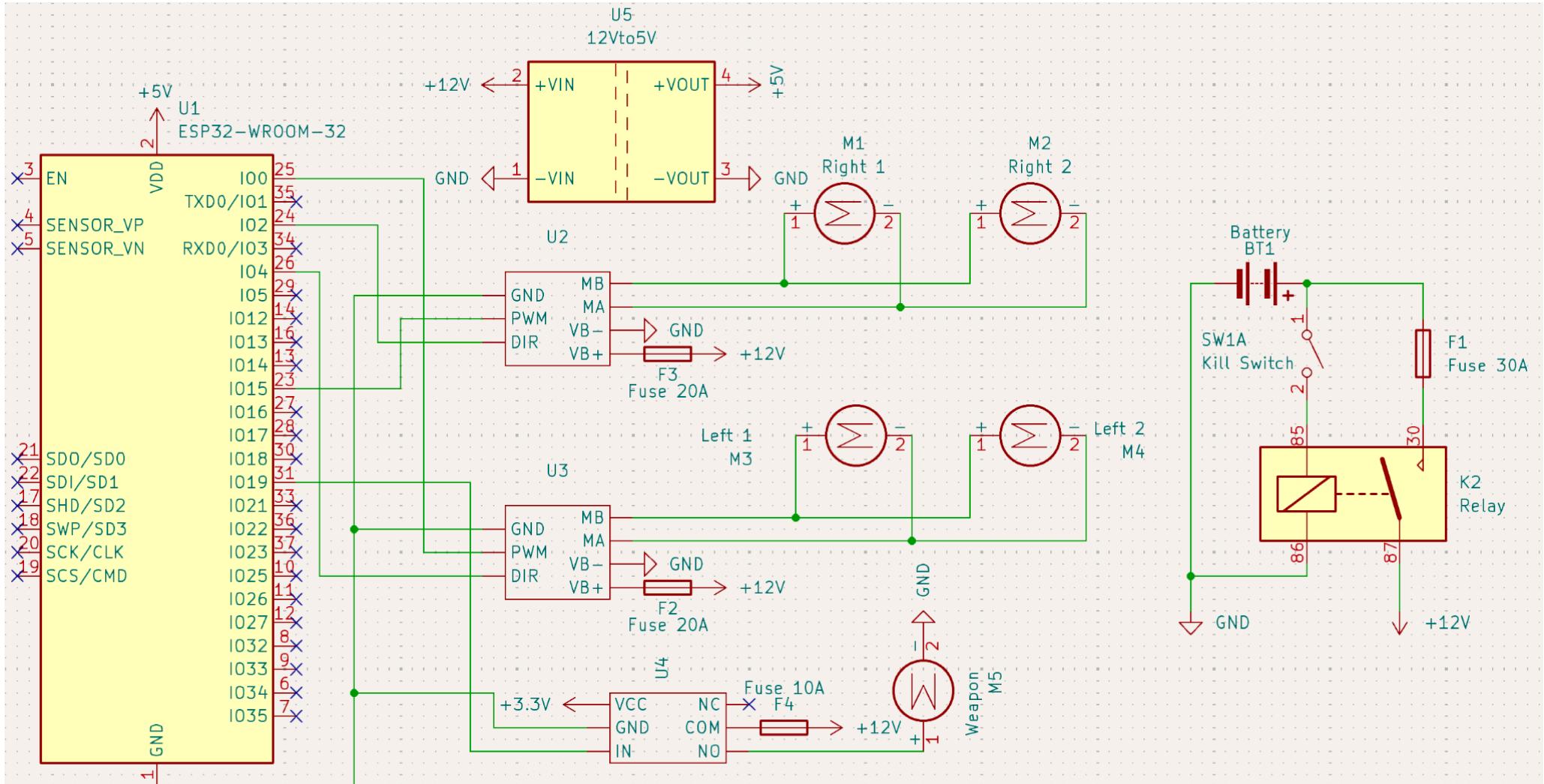
Разстоянието, на което радиокомуникациите са полезни, зависи значително от неща, като мощност на предавателя, качество на приемника, тип, размер и височина на антената, начин на предаване, шум и смущаващи сигнали.

Блокова схема



Фиг. 2.1 - Блокова схема

Електрическа схема



Фиг. 2.2 - Електрическа схема

Избор на компоненти

1. ESP32

ESP32 е мощен микроконтролер, създаден от Espressif Systems, който се използва за разработка на различни типове IoT (Интернет на нещата) проекти. ESP32 предлага широк набор от възможности и функции, като безжична комуникация (Wi-Fi и Bluetooth), вградени аналогови и цифрови входове/изходи, процесор с две ядра, различни периферни устройства (SPI, I2C, UART, GPIO и други), възможност за криптиране и сигурност на данните, и голям избор от библиотеки и SDK (Софтуерни развойни комплекти) за разработка на софтуер. ESP32 се използва в много различни приложения, като умни домове, индустриални автоматизации, сензорни мрежи, проследяване на данни, мобилни устройства и други. Той е популярен сред хобистите и професионалните разработчици заради своята възможност да комуникира с различни устройства по безжичен начин и да бъде програмиран за разнообразни задачи. В нашия случай микроконтролерът ще бъде напълно достатъчен за нуждите на проекта и ще бъде главния “мозък” в него.



Фиг. 3.1- ESP32

2. PlayStation 4 Controller

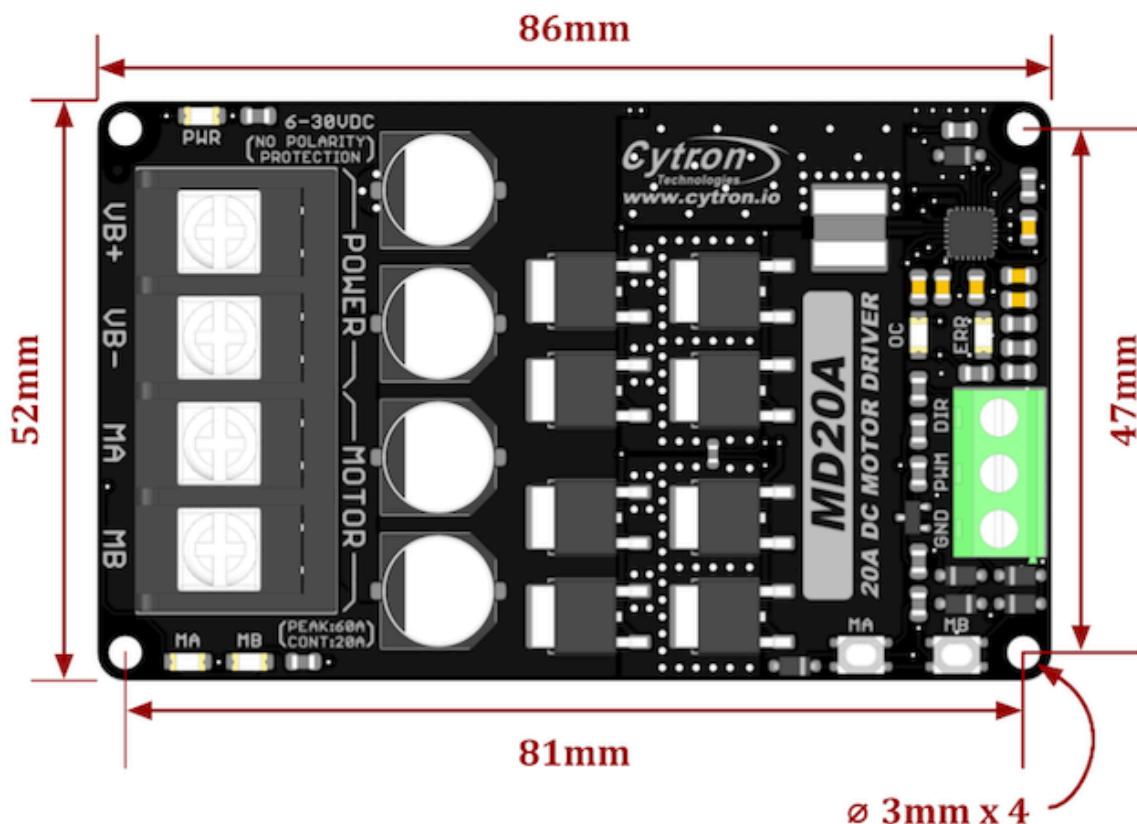
Контролерът за PlayStation 4 (PS4) е периферно устройство, предназначено за управление на игрите на PlayStation 4 конзолата. Този контролер се нарича DualShock 4. Използва Bluetooth комуникация, което го прави изключително полезен за нашия проект. С негова помощ ще можем лесно да управляваме нашият батълбот без нуждата от създаването на нов контролер. Контролерът идва с множество бутони и два “стика” за управление. С тях ще можем да управляваме движението на бота, а с помощта на бутоните ще можем да включваме и изключваме неговото оръжие.



Фиг. 3.2- DualShock 4

3.Cytron MDD20A

Cytron MDD20A е мощен драйвер за управление на електромотори. Модулът е предназначен за двупосочен контрол на два постояннотоков електромотор, работещ с напрежение 6V – 30V. Осигурява до 20A изходен ток на мотора, без допълнително охлаждане и издържа до 60A за няколко секунди. Логическата част се захранва от напрежението за моторите и не се нуждае от допълнително захранване. Управлява се с опростен интерфейс, като от микроконтролер са необходими само по два сигнала на мотор: високо или ниско ниво, или PWM до 20KHz. Предвидени са два бутона за бързо тестване или ръчен контрол на моторите.



Фиг. 3.3- Cytron MDD20A

4. GB37 550 High Speed Carbon Brush DC Gear Motor

GB37 550 е мощен четков електромотор, работещ на 12V. Моторът има максимална мощност от 40W. Може да издържи тежест около 32 килограма. Електромоторът прави 90 оборота в секунда. Идва в комплект със 130mm диаметрова висококачествена гума. В проектът са използвани 4 комплекта (4 гуми и 4 мотора).



Фиг. 3.4- GB37 550 motor & wheel

5. Акумулатор 12V/14Ah HIGH RATE LAVA

Батерията, която е нужна за захранването на робота е 12V и е с капацитет от 14Ah. Ще бъде достатъчна за захранването на всички компоненти в робота и се очаква да издържи за 5 битки, които имат продължителност от 3 минути. Предимството, което притежават батериите от серия High Rate – е изработката на решетката, която позволява 20% увеличение на мощността. Това ни подсигурява, че пиковия ток ,нужен на робота да функционира в битка, ще му бъде предоставен.



Фиг. 3.5- Акумулаторна батерия LAVA 12V/14Ah

6. Реле

Релето е електрически задвижван ключ. Те обикновено използват електромагнит (намотка), за да управляват вътрешния си механичен превключващ механизъм (контакти). Когато релейният контакт е отворен, това ще включи захранването за верига, когато бобината е активирана.

За правилната работа на нашия робот ще използваме две релета. Едното (фиг. 3.6а) ще се използва за контролирането на цялата верига от ключа (Kill Switch), а другото (фиг 3.6б) ще се използва за включване и изключване на оръжието.



Фиг. 3.6а- Кит K2013 Модул с реле (активно ниско ниво)



Фиг. 3.6б- Автомобилно реле 4120-1C-12V 30A

7. Оръжие

За оръжие на робота е дисков нож за фреза. То ще има за цел да нанася вреда на противника. Дискът е с диаметър от 160мм и дебелина 15мм.



Фиг. 3.7 - Дисков нож за фреза

8. Мотор 100W 12V

За завъртане на оръжието ще използваме мотор от винтоверт. Той има мощност от 100W и работи на 12V. Освен моторът ще се използва и приставката на винтоверта, която го хваща с редуктора и ги държи заедно, за да може да не се отделят при по-силен сблъсък с противника.



Фиг. 3.8 - Мотор 100W 12V

9. Ъглошлайф приставка за бормашина Елтос

Ъглошлайф приставката за бормашина от марката Елтос е устройство, което се монтира на стандартна бормашина, за да я превърне в ъглошлайф. Тази приставка позволява използването на бормашината за рязане, шлайфане и полиране на различни материали като метал, дърво или пластмаса. Преобразувателят обикновено се състои от защитен кожух, държач за диска и механизъм за прехвърляне на въртящия момент от бормашината към диска.



Фиг. 3.9 - Ъглошлайф приставка за бормашина Елтос

10. Зарядно за автомобил 12V

Зарядното устройство за кола е електрическо устройство, което преобразува 12-волтовото напрежение от автомобилната батерия в 5 волта, подходящи за захранване на микроконтролер ESP32. Устройството използва DC-DC преобразувател, който осигурява стабилизиран изход и защита срещу пренапрежение и късо съединение. За целта на захранването на нашия проект, ще се възползваме от едно такова зарядно, което ефективно намалява напрежението и осигурява безопасно и надеждно захранване на микроконтролера директно от автомобилната система.



Фиг. 3.10 - Зарядно за автомобил 12V

11. Kill Switch

Kill Switch е специализиран електрически ключ, предназначен да прекъсва напълно захранването към робота в случай на нужда. Този ключ функционира като ултимативно средство за сигурност, позволявайки незабавно и цялостно изключване на робота за предотвратяване на неочеквани аварии или защита на операторите и оборудването. Kill Switch

обикновено се поставя на достъпно място на робота и може да се активира ръчно или автоматично, в зависимост от проектните изисквания. Използването на такъв ключ е критично в ситуации, където контролът над роботизирани системи трябва да се прекъсне моментално, за да се осигури безопасността на системата и околната среда.



Фиг. 3.11 - Switch ASW16-101

12. Предпазител

Бушонът с топящ се проводник е елемент от електрическите защитни устройства, използван широко в електрониката за защита на електрически вериги от прекомерен ток, който може да предизвика повреди или пожар. Този тип бушон се състои от топящ се проводник, обикновено направен от олово или друг метал с ниска точка на топене, който е монтиран във въздушонепроницаема туба, често направена от стъкло или керамика. Работата на бушона е базирана на принципа, че при протичането на ток над предварително зададеното защитно ниво, топящият се проводник се нагрява и се прекъсва (топи), което

спира тока и предотвратява по-нататъшни повреди във веригата. Това прекъсване ефективно изолираувредената част от останалата част на системата, предотвратявайки допълнителни щети.



Фиг. 3.12 - Предпазител с топящ се проводник

Изработка на проекта

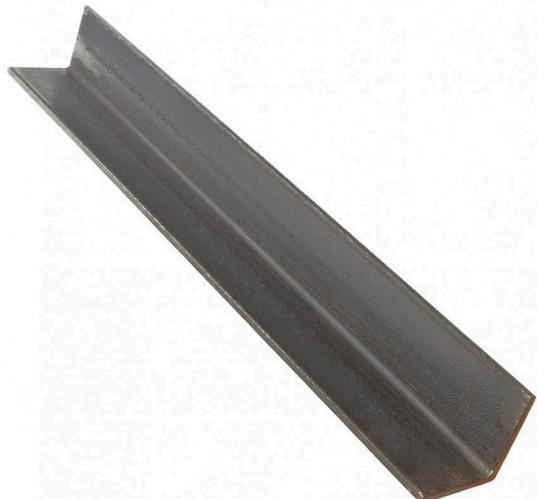
1. Изработка на корпуса

Корпусът е изработен от 1.5мм ламарина, подобна на тази на автомобил. Първата задача е ламарината да се нареже на няколко парчета с определен размер - 4 за стените на корпуса и 1 за тавана му. Долната част на робота е направена от 3см OSB плоскост (талашит), за да може лесно да се монтират компоненти върху него. След като парчетата са нарязани трябва да се изшкурят от нежеланата ръжда и да се намажат с киселина, за да може останалата ръжда да се махне.



Фиг. 4.1a - Нарязани плоскости от ламарина

След като плоскостите са нарязани трябва да ги монтираме заедно. За тази цел ще използваме винкели, с чиято помощ нашият робот ще е по-здрав. Те ще направят го направят не само по-здрав, но и ще улеснят монтирането на плоскостите и ще се избегне нуждата от всякакви заварки.



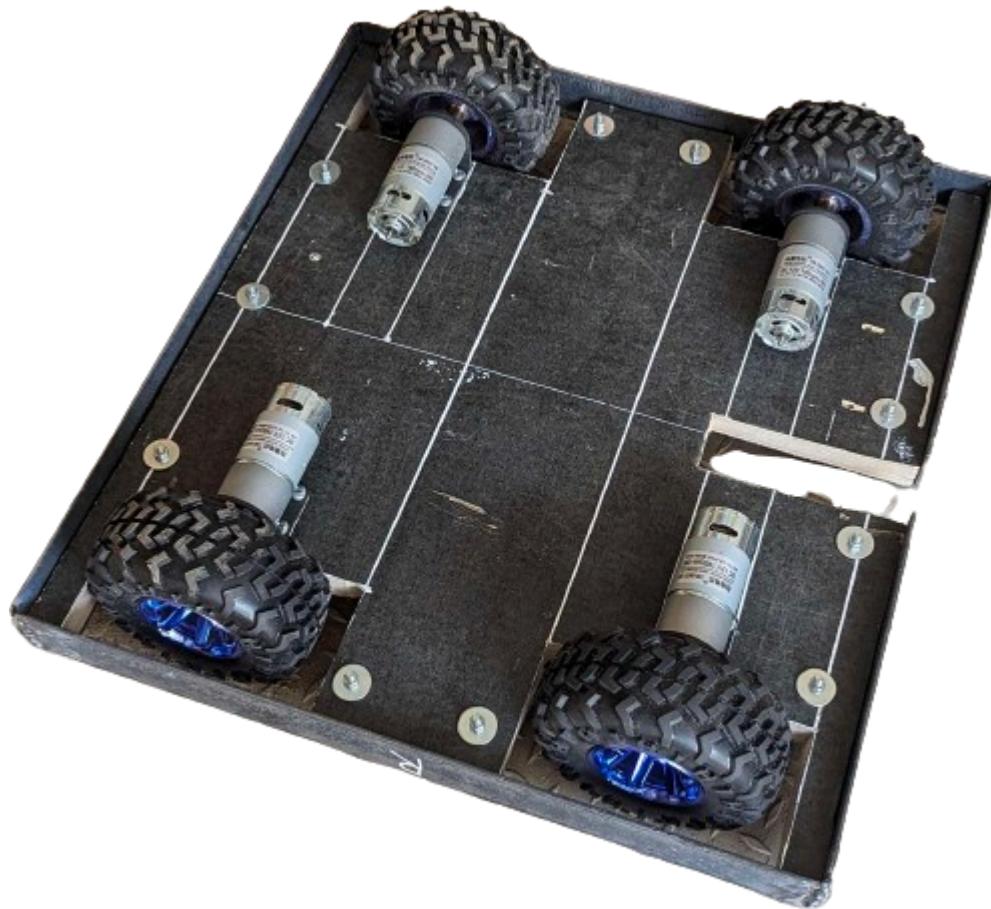
Фиг. 4.1б - Винкел



Фиг. 4.1в - Стените на конструкцията с винкел

2. Поставяне на компонентите

След сглобяване на конструкцията следва да се поставят компонентите върху долната част от корпуса. Първа стъпка е да се поставят гумите и техните мотори. Те идват в комплект с винтове и приставки за монтаж.



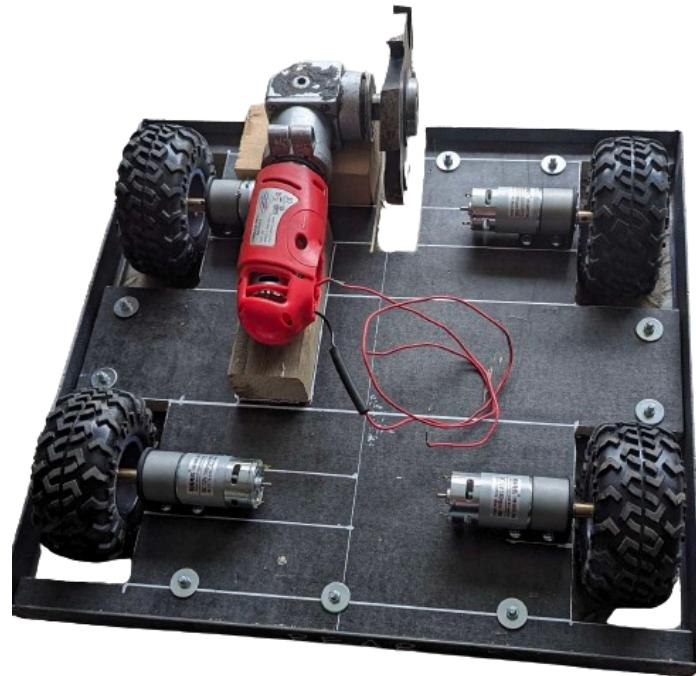
\Фиг. 4.1г - Монтирани гуми

Преди да започнем да монтираме всички мотори, трябва да свържем автомобилното реле към батерията и ключа на робота (Kill Switch). Положителният потенциал на батерията трябва да отиде на 30 пин на релето, а ключът да се свърже към плюса на батерията и 85 пин на релето. Отрицателния потенциал на батерията трябва да се свърже с 86 пин на релето. Така от 87 пин на автомобилното реле можем да вземем положителния

потенциал на батерията и да го използваме за моторите и захранване на микроконтролера.

След като гумите са поставени трябва да се монтира мотора от винтоверта и оръжието. За да може оръжието да не допира пода трябва да се сложи дървена летвичка, която ще повдигне мотора, заедно с оръжието с 5см. След като летвичката е сложена трябва да сложим мотора с приставката за ъглошлайф и да се монтира върху летвичката. За да може да е стабилна цялата конструкция моторът е закрепен с допълнителни свински опашки. Ъгловата предавка е закрепена с 7см винт, за да може да захвате летвата и долната част заедно. Оръжието е закрепено със специално направена гайка.

Следващата стъпка е да се устави връзката между микроконтролера и оръжието. За тази цел ще използваме Кит K2013 Модул с реле (активно ниско ниво) (*Фиг 3.6a*). Тъй като релето работи с активно ниско ниво, свързваме NO (Normally Open) изхода му с положителния потенциал на мотора. На COM (Common) изхода на релето свързваме положителния потенциал на батерията, който може да се вземе от 87 пин на автомобилното реле, но преди това трябва да се постави 10A предпазител, за да подсигурим, че токът, който ще мине от там няма да е толкова висок, че да изгори мотора. От другата страна на контролираното реле, IN (Input) входа му трябва да се свърже с 9 пин на микроконтролера, за да може да се подава сигнал и да се включва и изключва оръжието от него. На VCC и GND входовете релето трябва да се свърже със съответните за това пинове от микроконтролера (GND и 3.3V).



Фиг. 4.1д - Монтирани гуми и оръжие

След поставянето на оръжието и гумите тряба да се поставят и останалите компоненти. Драйвърите трябва да се монтират. Тъй като разполагаме с 2 драйвера и 4 мотора, трябва да сложим на един драйвър по 2 мотора. Като левите мотори са вързани към единия и десните мотори към другия драйвер. Макар и да е предназначен за 1 мотор, драйверът ще издържи тока и на двета мотора. Драйвърите трябва да се свържат с 20А бушони преди да се свържат с батерията, за да сме сигурни, че няма да мине прекалено голям ток и няма да ни изгурят драйверите. След това връзваме другата страна на бушоните с главното ни реле, което е вързано към батерията и нашия Kill Switch. За да се вземе положителен потенциал от батерията, то драйверите трябва да се свържат към 87 изход на автомобилното реле. Определения пин на драйверите за контролиране на моторите трябва да се свърже с ESP 32 микроконтроллера, за да може да им подаваме сигнал.

ESP 32 микроконтролерът трябва да се захрани с зарядното за автомобил. За да направим това, трябва зарядното да се свърже с 87 изход на автомобилното реле и микроконтролерът да се включи с Micro USB към USB кабел.

След като всичко това е направено се има пълна

функционалност. Трябва да се затвори и да се монтира изключващия ключ с помощта на гайката, която идва в комплект заедно с него.

Изработка на софтуер

1. Видове управление

Има два главни вида управление - управление базирано на танк (Tank-based controls) и управление базирано на конзола (Console-based controls). Можем да използваме сигналите подадени от Playstation 4 контролера и по двата начина.

Console-based контролите позволяват по-голямо удобство за повечето хора, тъй като са широко разпространени и лесни за научаване.

2. Използвани библиотеки

За програмиране на ESP32 най-често се използва Arduino IDE, но тъй като този микроконтролер е базиран на различен чип от този на Arduino, трябва да се изтеглят допълнителни библиотеки. Библиотеките са Open-Source в GitHub и трябва да се добавят в Boards Manager-а на Arduino IDE, за да бъдат достъпни.

ArduinoIDE → Settings... → Additional boards manager URLs
В полето се добавят линкове [1] и [2] от допълнителна литература.

Библиотеката се изтегля в Boards Manager-а на компилатора:

Tools → Boards → Boards Manager

И в полето се търси: **esp32 by Espressif Systems**

За Bluetooth връзката се използва библиотеката **Bluepad32**, която е Open-Source в GitHub и трябва да се добави в Boards Manager-а на Arduino IDE, за да бъде достъпна.

ArduinoIDE → Settings... → Additional boards manager URLs
В полето се добавя линк [3] от допълнителна литература.

Библиотеката се изтегля в Boards Manager-а на компилатора:

Tools → Boards → Boards Manager

И в полето се търси: **esp32_bluepad32 by Ricardo Quesada**

Cytron предлага и библиотека, за по-лесно управление на драйверите, която се изтегля в Library Manager-а на компилатора:

Tools → Manage Libraries...

И в полето се търси: **Cytron Motor Drivers Library by Cytron**

3. Логика зад кода

За да функционира правилно, Battle Bot-а трябва да се движи напред, назад, да завива наляво, надясно и да включва и изключва оръжието.

3.1 Установяване на връзка между контролера и ESP32

Bluepad32 предоставя готов шаблон за разработката на връзката между двете устройства, от която се използват следните функции, по които можем да работим.

```
> void onConnectedController(ControllerPtr ctl) { ...  
}  
  
> void onDisconnectedController(ControllerPtr ctl) { ...  
}  
  
> void dumpGamepad(ControllerPtr ctl) { ...  
}  
  
> void processGamepad(ControllerPtr ctl) { ...  
}
```

Фиг. 5.1 - Функции за установяване на връзка между контролера и ESP32

Всяко устройство с Bluetooth има свой собствен уникален Bluetooth адрес. Във функцията **onConnectedController** се случва опитът за свързване на устройството към микроконтролера. За да

позволим връзка само и единствено с нашият Playstation 4 контролер, ESP 32 трябва да сравни устройството, което иска да се свърже с него и уникалният Bluetooth адрес на нашият контролер - ако не съвпаднат връзката се отменя и продължава да търси.

```
void onConnectedController(ControllerPtr ctl) {
    bool foundEmptySlot = false;
    for (int i = 0; i < BP32_MAX_GAMEPADS; i++) {
        if (myControllers[i] == nullptr) {
            Serial.printf("CALLBACK: Controller is connected, index=%d\n", i);
            // Additionally, you can get certain gamepad properties like:
            // Model, VID, PID, BTAddr, flags, etc.
            ControllerProperties properties = ctl->getProperties();

            uint8_t desiredBtAddr[] = {0xa4, 0xae, 0x12, 0xeb, 0xf9, 0x73};

            // Compare the Bluetooth address with the desired address
            if (memcmp(properties.btaddr, desiredBtAddr, 6) == 0) {
                Serial.println("Desired Bluetooth address matched!");
                myControllers[i] = ctl;
                foundEmptySlot = true;
                break;
            } else {
                Serial.println("Desired Bluetooth address not matched!");
            }
        }
    }
    if (!foundEmptySlot) {
        Serial.println("CALLBACK: Controller connected, but could not find empty slot");
    }
}
```

Фиг. 5.2 - Функция *onConnectedController*

Функцията **onDisconnectedController** е готова от шаблона - нищо не е модифицирано по нея, тя изглежда така:

```

void onDisconnectedController(ControllerPtr ctl) {
    bool foundController = false;

    for (int i = 0; i < BP32_MAX_GAMEPADS; i++) {
        if (myControllers[i] == ctl) {
            Serial.printf("CALLBACK: Controller disconnected from index=%d\n", i);
            myControllers[i] = nullptr;
            foundController = true;
            break;
        }
    }

    if (!foundController) {
        Serial.println("CALLBACK: Controller disconnected, but not found in myControllers");
    }
}

```

Фиг. 5.3 - Функция onDisconnectedController

3.2 Тестване на връзката

Функцията **dumpGamepad** е базова функция, която принтира сигналите, които ESP32 получава от контролера, в Serial монитора.

```

void dumpGamepad(ControllerPtr ctl) {
    Serial.printf(
        "idx=%d, dpad: 0x%02x, buttons: 0x%04x, axis L: %4d, %4d, axis R: %4d, %4d, brake: %4d, throttle: %4d, "
        "misc: 0x%02x, gyro x:%6d y:%6d z:%6d, accel x:%6d y:%6d z:%6d\n",
        ctl->index(),           // Controller Index
        ctl->dpad(),            // DPAD
        ctl->buttons(),         // bitmask of pressed buttons
        ctl->axisX(),           // (-511 - 512) left X axis
        ctl->axisY(),           // (-511 - 512) left Y axis
        ctl->axisRX(),          // (-511 - 512) right X axis
        ctl->axisRY(),          // (-511 - 512) right Y axis
        ctl->brake(),            // (0 - 1023): brake button
        ctl->throttle(),         // (0 - 1023): throttle (AKA gas) button
        ctl->miscButtons(),      // bitmak of pressed "misc" buttons
        ctl->gyroX(),            // Gyro X
        ctl->gyroY(),            // Gyro Y
        ctl->gyroZ(),            // Gyro Z
        ctl->accelX(),           // Accelerometer X
        ctl->accelY(),           // Accelerometer Y
        ctl->accelZ()            // Accelerometer Z
    );
}

```

Фиг. 5.4 - Функция за тестване на връзката

3.3 Управление на задвижването

Главната функция **processGamePad** се извиква в цикъла на програмата със задачата да обработва данните, получени от контролера и да ги подава на компонентите в бойния робот.

За целта се проверява дали пилотът е подал сигнал за акселерация/завиване/оръжие и изпълняваме дадената команда.

```
if ((acc - accSpeed) * -1 <= ctl->brake() && ctl->throttle() == 0){
    acc -= accSpeed;
}
else if (acc * -1 > ctl->brake() && ctl->throttle() == 0){
    acc += accSpeed;
    if (acc * -1 < deadzone){acc = 0;}
}

else if (acc + accSpeed <= ctl->throttle() && ctl->brake() == 0){
    acc += accSpeed;
}
else if (acc > ctl->throttle() && ctl->brake() == 0){
    acc -= accSpeed;
    if (acc < deadzone){acc = 0;}
}
```

Фиг. 5.5 - Код за управление на задвижването на батълбота

За да предотвратим големи пикове на тока, използваме метод за акселерация на моторите, спрямо аналоговия сигнал, който е подаден.

```
int speed = map(acc, -1020, 1020, -220, 220);
speed *= -1;
```

Фиг. 5.6 - Код за предотвратяване на големи пикове на тока

Сигналите на бутоните L2 & R2, които контролера праща варират от 0 до ~ 1020 , но готовата библиотека на Cytron управлява драйвърите със стойности от -255 до 255.

За да коригираме сигналите, обръщаме стойността на **acc** при задвижване назад да бъде от 0 до -1020. Така може да се map-не променливата, така че да бъде от -220 до 220.

```
if(ctl->axisX() < 0 - deadzone){  
    motorR.setSpeed(speed * -1);  
    motorL.setSpeed(speed);  
    Serial.println("steer left");  
}  
else if(ctl->axisX() > 0 + deadzone){  
    motorR.setSpeed(speed);  
    motorL.setSpeed(speed * -1);  
    Serial.println("steer right");  
}  
else{  
    motorR.setSpeed(speed);  
    motorL.setSpeed(speed);  
}
```

Фиг. 5.6 - Код за проверка на посоката на задвижване

Преди подаването на скоростта към моторите проверяваме дали пилотът иска да завие в някоя от посоките, за да се предприемат нужните действия за задвиждане.

3.4 Управление на оръжието

Оръжието трябва да има способността да се включва и изключва по време на битка, задачата на кода е да отпуши релето към оръжието при подаден сигнал от пилотът.

```
if (ctl->a()) {  
    if (millis() - prev >= interval) {  
        prev = millis();  
        if(on == 0){  
            digitalWrite(19, HIGH);  
            on = 1;  
        }  
        else{  
            digitalWrite(19, LOW);  
            on = 0;  
        }  
        Serial.println(on);  
    }  
    else{  
        Serial.println("time has NOT passed");  
    }  
}  
}
```

Фиг. 5.6 - Код за включване/изключване на оръжието

Интегрираме и интервал (cooldown) за включване/изключване на релето, за да преодотвратим обработката на сигнала през всеки тик от секундата, което би довело до бързото включване и изключване на оръжите при нормално натискане на бутона.

Целият код може да бъде намерен в GitHub хранилището на проекта. [4]

Допълнителна литература

1. https://arduino.esp8266.com/stable/package_esp8266com_index.json
2. https://espressif.github.io/arduino-esp32/package_esp32_index.json
3. https://raw.githubusercontent.com/ricardoquesada/esp32-arduino-lib-builder/master/bluepad32_files/package_esp32_bluepad32_index.json
4. <https://github.com/MechkarovTUES/BattleBot>