



ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ  
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

# ДИПЛОМНА РАБОТА

на тема

Боеен робот с пулт за дистанционно управление  
по радио връзка

Дипломант:

**Александър Иванов**  
специалност Системно  
програмиране

Дипломен ръководител:

**инж. Владимир Гаристов**

28 февруари 2024 г.  
София





ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ  
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Дата на заданието: 15.11.2023 г.

Дата на предаване: 15.02.2024 г.

Утвърждавам:.....

/проф. д-р инж. П. Якимов/

## ЗАДАНИЕ

### за дипломна работа

ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ  
по професия код 481020 „Системен програмист“  
специалност код 4810201 „Системно програмиране“

на ученика Александър Евгениев Иванов от 12А клас

1. Тема: Боен робот с пулт за дистанционно управление по радио връзка
2. Изисквания:
  - 2.1. Радио връзка робот-управляващ модул
  - 2.2. Управление на посоката на движение посредством джойстици
  - 2.3. Управление чрез ШИМ на мощността на постояннотокови четкови мотори
  - 2.4. Управление на стъпков мотор
  - 2.5. Функции за безопасност - крайни изключватели и прекъсвач на захранването
3. Съдържание
  - 3.1. Теоретична част
  - 3.2. Практическа част
  - 3.3. Приложение

Дипломант :.....

/ Александър Иванов /

Ръководител:.....

/ инж. Владимир Гаристов /

ВРИД Директор:.....

/ ст. пр. д-р Веселка Христова /

# Съдържание

<b>Списък на фигурите</b>	<b>3</b>
<b>Списък на таблиците</b>	<b>4</b>
<b>1 Увод</b>	<b>5</b>
<b>2 ???? Методи, средства и методологии за изработване на бойни роботи ????</b>	<b>6</b>
2.1 Основни методи и технологии за задвижване на бойни роботи	6
2.2 Основни методи и технологии за дистанционно управление на бойни роботи . . . . .	8
2.3 Основни методи за защита на бойни роботи . . . . .	9
2.4 ?видове съществуващи бойни роботи? . . . . .	10
<b>3 Дизайн и блоковата схема на работа</b>	<b>11</b>
3.1 Функционални изисквания към работа . . . . .	11
3.2 Блокова схема на работа . . . . .	11
<b>4 Механика</b>	<b>13</b>
4.1 Цялостен модел на работа . . . . .	13
4.2 Задвижване на работа . . . . .	13
4.3 Задвижване на ръката . . . . .	14
4.4 Задвижване на оръжието . . . . .	15
<b>5 Проектиране на печатните платки</b>	<b>17</b>
5.1 Принципка електрическа схема на работа . . . . .	17
5.1.1 Радиочестотен модул . . . . .	17
5.1.2 Управление на моторите . . . . .	18
5.1.3 Сензори . . . . .	19
5.1.4 Захранване на печатната платка . . . . .	21
5.2 Опроводяване на печатната платка на работа . . . . .	23
5.3 Монтажна схема на работа . . . . .	23
5.4 Принципка електрическа схема на дистанционното . . . . .	23
5.4.1 Радиочестотен модул . . . . .	24
5.4.2 Сензори . . . . .	24

5.5	Опроводяване на печатната платка на дистанционното . .	24
5.6	Монтажна схема на дистанционното . . . . .	24
<b>6</b>	<b>Софтуерна реализация</b>	<b>25</b>
6.1	Логическа схема на проекта . . . . .	25
6.2	Софтуерна реализация на библиотеката на радиочестотния модул . . . . .	27
6.2.1	Инициализация на модула . . . . .	30
6.2.2	Изпращане на информация . . . . .	33
6.2.3	Получаване на информация . . . . .	34
6.3	Софтуерна реализация на дистанционното . . . . .	36
6.3.1	Инициализация на микроконтролера на дистанцион- ното . . . . .	36
6.3.2	Работен цикъл на дистанционното . . . . .	37
6.4	Софтуерна реализация на управлението на работа . . . . .	38
6.4.1	Инициализация на микроконтролера на работа . . . . .	38
6.4.2	Работен цикъл на работа . . . . .	39
6.4.3	Сензори за обратна връзка . . . . .	41
<b>7</b>	<b>Заключение</b>	<b>44</b>

# Списък на фигурите

2.1 Боеен робот, задвижван от вериги . . . . .	6
2.2 Боеен робот, задвижван от механични крака . . . . .	7
3.1 Блокова схема на дистанционното . . . . .	11
3.2 Блокова схема на работа . . . . .	12
5.1 Радиочестотен модул . . . . .	18
5.2 Оптрон . . . . .	18
5.3 Левел шифтър . . . . .	19
5.4 оптични сензори . . . . .	20
5.5 Крайни изключватели за ръката . . . . .	20
5.6 Защити на захранването . . . . .	21
5.7 Захранване на логическата част в работа. . . . .	22
6.1 Логическа схема на дистанционното . . . . .	26
6.2 Логическа схема на работа . . . . .	27

# Списък на таблиците

2.1 Сравнителна таблица за безжични технологии . . . . .	9
6.1 Пинове . . . . .	37
6.2 Пинове . . . . .	39

## Увод

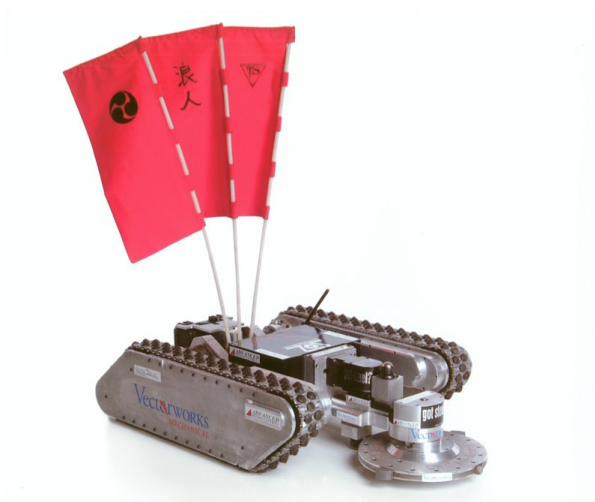
Още от зората си, човечеството търси разнообразни и вълнуващи начини да успее да избяга от еднообразното си ежедневиe. За целта много и различни методи за развлечение са възникнали, вариращи от приятелски игри до зрелищни спектакли. Едно от най-разпространените забавления още от тогава е също толкова популярно и днес, а именно – боевете. Зрелището на това два индивида да се бият помежду си завладява всеки зрител. Годишите са доказали, че колкото по-драматична е една битка, толкова по-силни са чувствата, които се пораждат у нейните наблюдатели. Но това води до един неизбежен проблем: участниците в такива интензивни битки винаги биват физически наранени. Поради тази причина се появява казусът как може да се постигнат тези зрелища, без участниците да пострадат. Решението на този проблем са битките с бойни роботи (или батълботи). Стандартна битка трае 3 минути и в тях роботизираните системи, контролирани с дистанционно управление, целят увреждането на опонента до такава степен, че вече да не може да извършва движения.



# ???? Методи, средства и методологии за изработване на бойни роботи ????

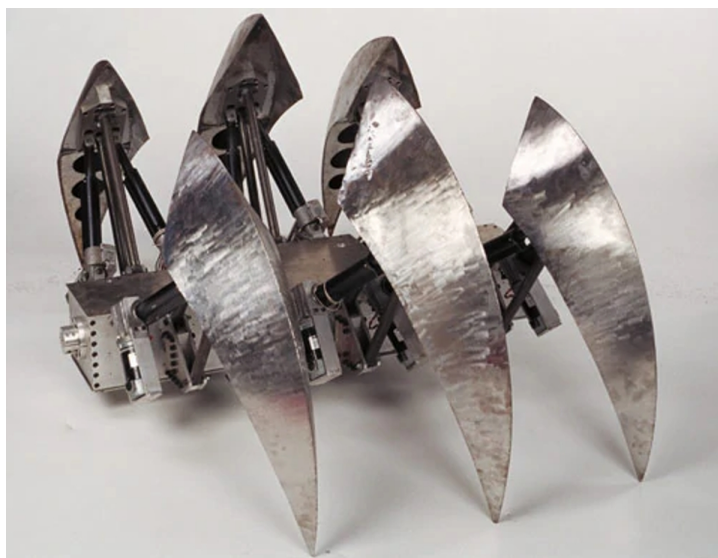
## 2.1 Основни методи и технологии за задвижване на бойни роботи

Бойните роботи традиционно могат да се задвижват по три начина чрез вериги, колела или механични крака. Има и други начини, но те не са ефективни в битка. Роботите, задвижвани с танкови вериги, имат отлично сцепление със земята и вървят много стабилно, но имат много недостатъци. Поради голямата площ на триене при завъртане те изразходват значително количество енергия. Освен това и самото движение се извършва сравнително бавно, което позволява на опонента да заобиколи робота и да го удари в гръб. Боен робот, задвижван от верига, може да се види на фиг. 2.1.



**Фигура 2.1:** Боен робот, задвижван от вериги

Механичните крака имат също много недостатъци. Някои от които са, че са много сложни за конструкция и контрол. Друг техен недостатък е, че те не са достатъчно здрави по време на битка, особено срещу посичащите спинер роботи. Повече за този тип бойни роботи може да се прочете в глава 1.4. Трета слабост е, че центъра на тежестта на роботи с такава система за задвижване е много високо над земята, което ги прави уязвими срещу атаки, целящи преобръщане. Батълбот, използващ механични крака може да бъде видян на фиг. 2.2.



**Фигура 2.2:** Боеен робот, задвижван от механични крака

Поради гореспоменатите причини, най-често бойните роботи се задвижват чрез колела. Разпространени са два начина на управление на задвижването на моторни средства с колела – Акерман управление и диференциално управление. Акерман управлението е възприето от повечето моторни-превозни средства. При него един голям мотор задвижва колелата и един по-малък отговаря за тяхното завъртане. То е ефективно при движение в права линия, но когато трябва да се завие се изискват определени маневри. Диференциалното управление е много по-често срещано в роботиката. При него лявата и дясната страна на работа се задвижват напълно индивидуално. Недостатъкът на този метод е фактът, че за да се движи моторното средство в права линия двете половини трябва да имат еднаква скорост, което е трудно за постигане. Голямото преимущество обаче е, че завиването става значително по-бързо.

Освен по начин на управление на задвижването роботите придвижващи се с колела се различават помежду си и чрез броя на задвижваните колела. При такива с две задвижващи колела и диференциално управление на задвижването, завиването се случва със сравнително ниски загуби на енергия. Проблемът е, че с две опорни точки, роботът най-вероятно ще има нужда от поне още една такава. Той се решава чрез добавянето на ball transfer units.

## 2.2 Основни методи и технологии за дистанционно управление на бойни роботи

Съгласно официалните изисквания за работа към бойните роботи, те трябва да бъдат контролирани безжично дистанционно. За целта могат да се използват много технологии за безжична комуникация, като едни от най-популярните са Wi-Fi, Bluetooth и nRF24L01.

Wi-Fi позволява много висока скорост на предаване на информацията, но има много недостатъци. Един от тях е, че има много високо потребление на енергия. Друг е, че не може директно да се предава информация от едно устройство на друго безжично, а първо трябва тази информация да се подаде на маршрутизатор и след това той да я изпрати до устройството получател. Отделно по време на боевете се очаква да има голяма публика и безплатен Wi-Fi което предполага, че каналите, които се използват за тази технология ще бъдат претоварени и съответно ще има по-лоша свързаност.

Bluetooth технологията осигурява средна скорост на предаване на информация, на цената на средно ниво на консумация на енергия. Недостатъкът е, че за да могат две устройства да се свържат и да общуват безжично и двете трябва предварително да се сдвоят, което е непрактично за системи, които не включват компютри или телефони.

Модулът nRF24L01 позволява безжична радиочестотна комуникация с други такива модули. От трите технологии за безжична комуникация този модул предлага тази с най-ниско потребление на енергия. За разлика от Wi-Fi, nRF24L01 може да се комуникира с друго устройство директно, без необходимостта от маршрутизатор. Предимството му

	Wi-fi	Bluetooth	nRF24L01+
Скорост	Висока	Средно	Средна
Обхват	10ки метри	10 метра	10-150 метра
Енергийна консумация	Висока	Средна	Ниска

**Таблица 2.1:** Сравнителна таблица за безжични технологии

пред Bluetooth е, че не е необходимо предварително двете устройства да се сдвояват.

## 2.3 Основни методи за защита на бойни роботи

За защита на бойните роботи винаги се монтира броня около неговата структура. Видовете броня са: традиционна, аблативна и реактивна.

Традиционната броня е изработена от много здрави и твърди материали, които се стремят да абсорбират и предадат енергията на удара без да се повреждат. Високата твърдост и здравина на този вид защита често се използва за чупене или изтъпяване на остриетата на вражеските оръжия и запазване целостта на робота при удари. Благодарение на здравината си тази броня по-рядко трябва да се сменя след битки, но нейният недостатък е, че при удар енергията на сблъсъка се предава до елементите вътре в робота, което може да доведе до тяхното повреждане.

Аблативната защита, от друга страна е проектирана да предпазва от щети робота, като самата тя бива повреждана чрез процеса аблация. Това е процеса на премахване на материал от повърхността на обект посредством изпарение или стружко отделяне. Материалите, от които е изградена са също твърди и здрави, но за разлика от традиционната броня имат по-ниска твърдост. Поради това тяхно свойство, когато тези брони трябва да предпазват от силни удари, вътрешните елементи биват изложени на риск. Материали като дървото са много ефикасни представители на този вид елементи, но друг техен недостатък е, че сблъсъците водят до много визуални следи, което често носи допълнителни точки на опонента за щети.

Третия вид броня е реактивната. По време на удар тя реагира по някакъв начин, за да предотврати щети. Има различни видове реактивна

броя и всяка има свой предимства и недостатъци. Пример за такъв вид защита е пласт гума между два пласта метал. Предимството ѝ е, че в случай на удар, пластът гума би абсорбирал енергията на удара. Този вид броя не е ефективна в боевете с роботи, поради причината, че много бързо бива повреждана и спира да пази.

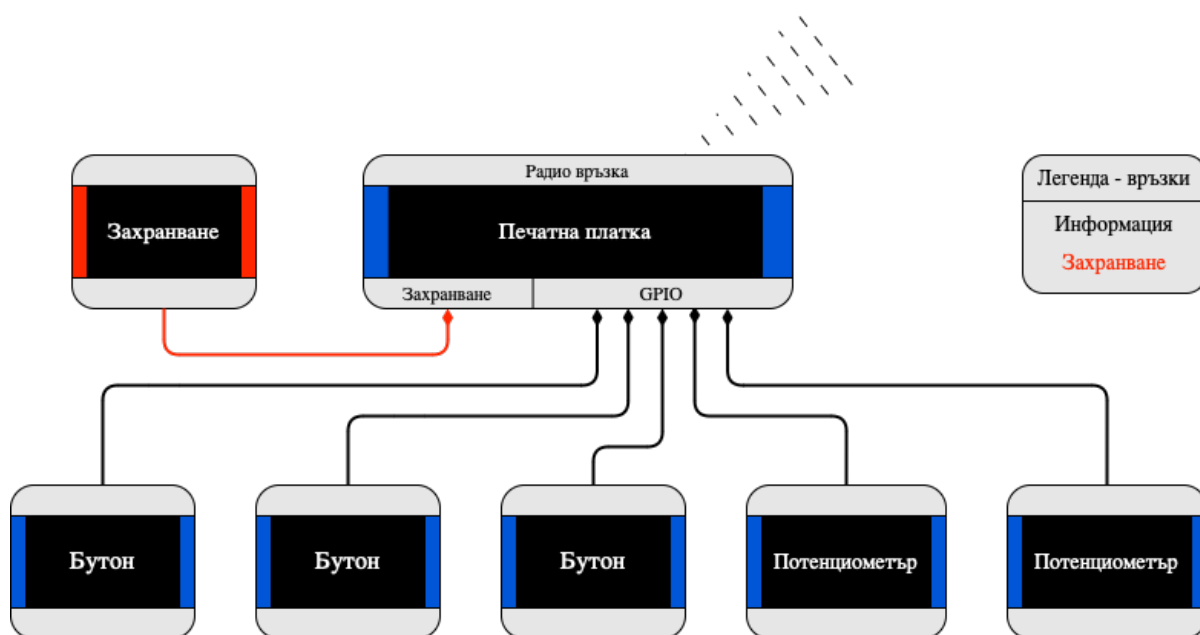
## 2.4 ?видове съществуващи бойни роботи?

Съществуват много различни видове батълботи. Основно изискване към всеки от тях е да имат поне едно оръжие, чрез което да могат да повредят опонента. В зависимост от своето оръжие роботите се разделят на 14 типа: „rammer“, „wedge“, „lifter“, „flipper“, „spearbots“, „horizontal spinner“, „sawbot“, „vertical spinner“, „drumbot“, „hammerbot“, „clamber“, „crusher“, „flamethrower“ и „multibot“. Има и други видове роботи, но те почти винаги могат да бъдат категоризирани в един от гореспоменатите видове. За ефективността на бойния робот в битка има голямо значение какво оръжие е избрано. Примерно със своите огромни дискове нанасящи разрушителни удари роботите с оръжие „vertical spinner“ са много ефективни срещу голяма част от роботите. Те имат обаче един фатален недостатък, който се изразява в това, че поради голямата скорост на въртене на съответното оръжие генерират жirosкопичен ефект, който намалява осезателно скоростта им на движение, което ги прави уязвими към удари отзад. С времето в роботските боеве категориите „flipper“, „horizontal spinner“, „vertical spinner“, „drumbot“, „hammerbot“ и „clamber“ са се открили като по-ефективни в сравнение с останалите.

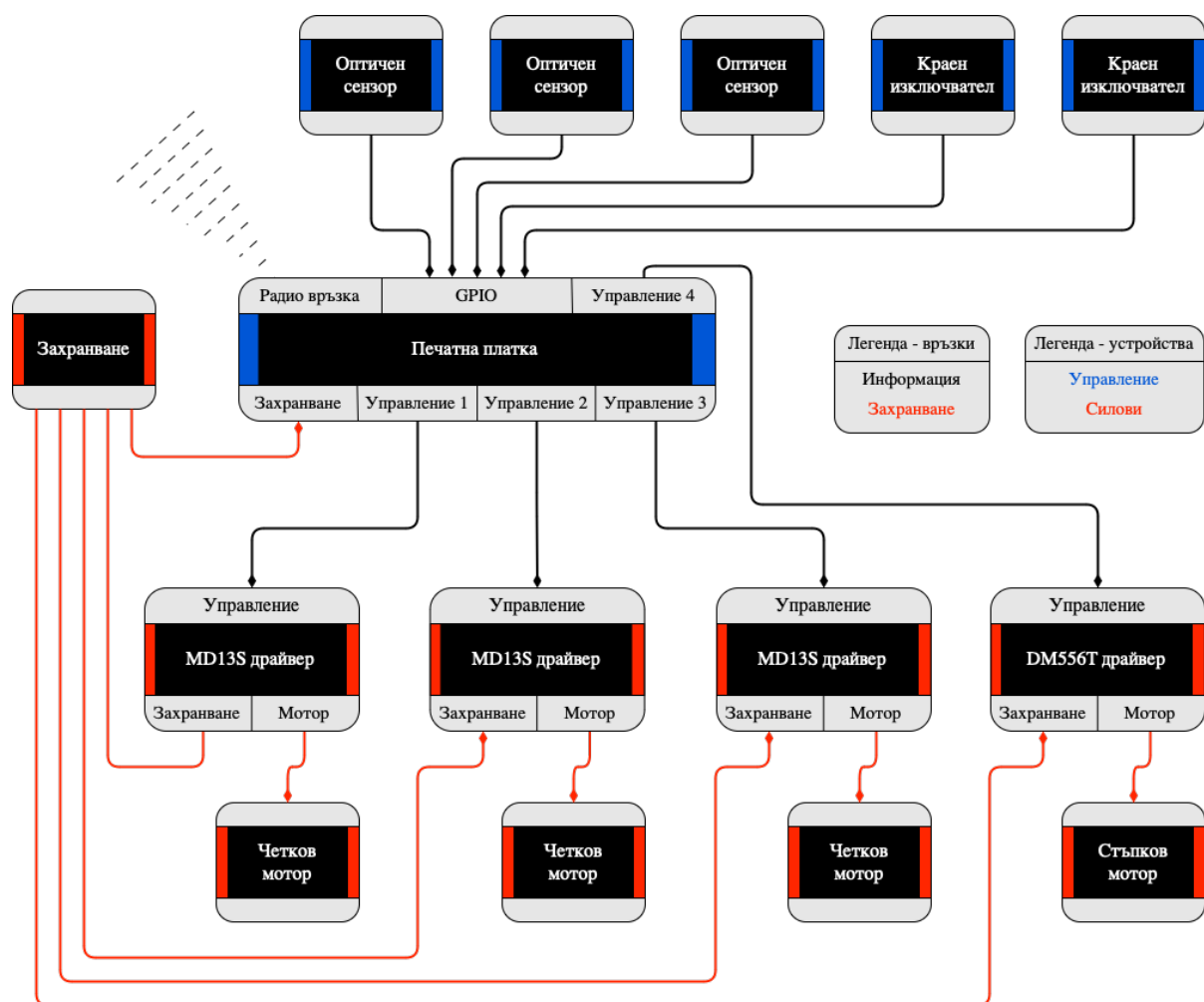
## Дизайн и блоковата схема на работа

### 3.1 Функционални изисквания към робота

### 3.2 Блокова схема на робота



**Фигура 3.1:** Блокова схема на дистанционното



**Фигура 3.2:** Блокова схема на робота

# Механика

## 4.1 Цялостен модел на работа

## 4.2 Задвижване на работа

По време на проектирането на механиката на боен робот първият проблем, който трябва да бъде разгледан е как той ще се придвижва по арената. След направеното проучване в ?? точка 1.1. ?? бе установено, че най-ефективният метод е диференциалното задвижване с колела. По дизайн роботът има две задвижващи колела от двете си страни и още две допълнителни ?? пасивни колела ??, които го балансират и му помагат да се движи по-добре. Всяко от задвижващите колела е с външен диаметър 80мм и ширина 12мм. Използваните мотори за тяхното задвижване са CHANCS 895 DC.

За да може батълбота, който тежи приблизително 18kg, да се задвижва от две колела трябва да бъде пресметнато какъв въртящ момент трябва да достига до всяко от тях <sup>1</sup>. За целта на изчисленията приемаме, че двете активни колела ще понасят цялото тегло на робота. Понеже на двете колела лежи тежестта от 18kg, означава, че всяко колело изпитва натиска от 9kg или еквивалентните им 88,29N. Коефициента на триене между колелата и пода на арената варира между 0,75 когато робота е в покой и 0,65 когато е в движение. Следователно най-голямата сила на съпротивление, която всяко колело може да генерира без самото колело да се върти е  $88,29\text{N} \times 0,75 = 66,2175\text{N}$ . След като диаметъра на колелото е 80мм следва, че радиусът му е 40мм или 0,04м. Следователно минималният въртящ момент, който позволява на колелата да се въртят е равен на  $66,2175\text{N} \times 0,04\text{m} = 2,6487\text{Nm}$ . Следователно минималният въртящ момент, който мотора трябва да достави на колелото е 2,6487Nm, но по време на въртенето си той има само 0,735Nm.

<sup>1</sup>Изчисленията са извършени като се приема, че роботът се движи на равна повърхност



Поради тази причина се налага да се направи редукция на скоростта на мотора за да се увеличи неговия въртящ момент. За да се изчисли каква трябва да бъде нужната редукция трябва да се намери отношението на необходимият въртящ момент и този, който може да бъде генериран. Редукцията, която се получава, че трябва да бъде реализирана между мотора и колелото е минимум  $2,6487\text{Nm} / 0,735\text{Nm} = 3,6$ .

Начинът, по който е реализирано диференциалното задвижване е като всяко от активните колела бива задвижвано от различен мотор. Връзката между тях е реализирана, чрез зъбчат ремък с ширина 10мм. За да се постигне необходимата редукция, отношението на зъбите на ремъчните ролки на осите на колелата и тези на моторите трябва да бъде равно на нея. Поради това използваните ролки на моторите са с по 10 зъба, а другите са с по 36 зъба. С цел намаляване на триенето между осите на колелата и стените, осите са захванати с лагери в специално изработените за целта лагерни опори, закрепени за стените.

## 4.3 Задвижване на ръката

Първата специфична функционалност на разработения боен робот е начина, по който механичната ръка се върти около оста си и се задържа на позицията, на която и бива зададено да седи. Тази задача бива изпълнена чрез употребата на стъпков мотор Nema 24.

За да може ръката да се върти около оста си първо трябва да бъде изчислено какъв въртящ момент ще бъде необходим за целта. Тежестта на ръката е приблизително 4kg, което означава, че силата която е нужно да бъде приложена за да може тя да се завърти е 39,24N. За да пресметнем колко е необходимият въртящ момент за нейното завъртане е необходимо получената сила да бъде умножена по дължината на ръката, която е 240мм(0,24м). По този начин получаваме, че трябва да бъдат приложени  $39,24\text{N} \times 0,24\text{m} = 9,4176\text{Nm}$ . Употребеният стъпков мотор може да задържа на една позиция само товари, които изискват въртящ момент 3,1Nm или по-малко, от което следва, че се налага да бъде приложена минимална редукция от 3,03 за да може да бъде реализирано въртенето на ръката.

Начинът, по който е осъществено завъртането на ръката е като тя бива захваната за своята ос така, че чрез завъртането на оста ръката да бъде придвижвана заедно с нея. Предаването на движението от стъпковия мотор към оста на ръката е постигнато посредством зъбни ремъци. С цел максималното увеличение на приложената редукция е поставена спомагателна ос между мотора и оста на ръката. По този начин се получават две редукции на движението на мотора. Първата бива от мотора към спомагателната ос, като поставената ролка на мотора има 12 зъба, а тази на спомагателната ос 36 зъба. Постиганата редукция чрез тази връзка е 3. Втората редукция бива от спомагателната ос към оста на ръката и използваните ролки са в същото отношение като тези в предишната предавка. Резултатът от двете връзки е, че общата редукция на скоростта на въртене на мотора се получава да бъде умножението на двете, от които е съставена, което я прави 9. За намаляване на съпротивлението по време на въртенето си са поставени лагерни опори в краищата на двете оси.

## 4.4 Задвижване на оръжието

Друг проблем, който трябва да бъде решен по време на проектирането на механиката на всеки боен робот е по какъв начин неговото оръжие ще се задвижва. Както е описано в точка **????** оръжието, което се използва в разработения проект е ??180-милиметров?? диск с ширина 8мм и специфична форма. Формата на диска може да бъде видяна на фигура **????**. Той е захванат за своята ос така, че чрез завъртането на оста и диска да се върти заедно с нея. Използваният мотор за движението на оръжието е CHANCS 895 DC. Начина, по който е реализирано предаването на движението от мотора до оста на диска е посредством плосък ремък. Избран е плосък ремък за тази цел, а не зъбчат поради причината, че при удар ремъка и мотора трябва отведнъж да спрат своето движение, което често може да доведе до прескачане на зъби на зъбчатия ремък и тяхното ронене. В тези ситуации плоския ремък няма да се амортизира по този начин, а само ще се приплъзне леко по своите ролки. Реализирана е и редукция от 5 на скоростта на тази предавка поради високата скорост, с която би се въртял диска. С цел намаляване на съпротивлението по време на въртенето си в краищата

си оста на диска е поставен в лагерни опори, които са застопорени за стените на ръката.

# Проектиране на печатните платки

За проектирането на печатните платки и монтажните схеми в проекта е използван софтуерния пакет KiCad. Той е избран за целта, защото е безплатен и има богата библиотека с компоненти, съдържащи символи и ?? очертания на корпуси ??.

## 5.1 Принципна електрическа схема на работа

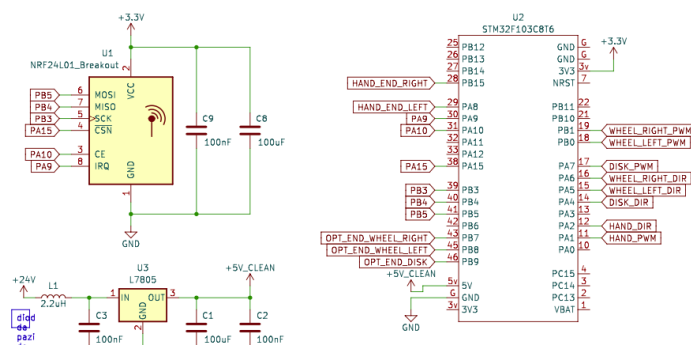
Принципната електрическа схема на печатната платка в работа може да бъде видяна на ?? приложение ?. Основният елемент на схемата е микроконтролерът STM32F103C8T6 (blue pill), който управлява останалите компоненти. Използваните компоненти в проектирането на тази печатната платка могат да бъдат видяни в таблица ???? приложение ????.

### 5.1.1 Радиочестотен модул

Връзката между дистанционното и работа се осъществява посредством nRF24L01+ PA/LNA радиочестотен модул. Комуникацията между него и използвания микроконтролер е посредством SPI интерфейс. Радиочестотният модул бива захранван с 3,3V от вградения регулатор на напрежението на blue pill. С цел повишаване на съотношението сигнал/шум са поставени 100nF керамичен кондензатор и 100uF електrolитен кондензатор, които да намалят съответно високочестотните и нискочестотните шумове.

предавател по далеч / мощни мотори = повече шум слаб сигнал

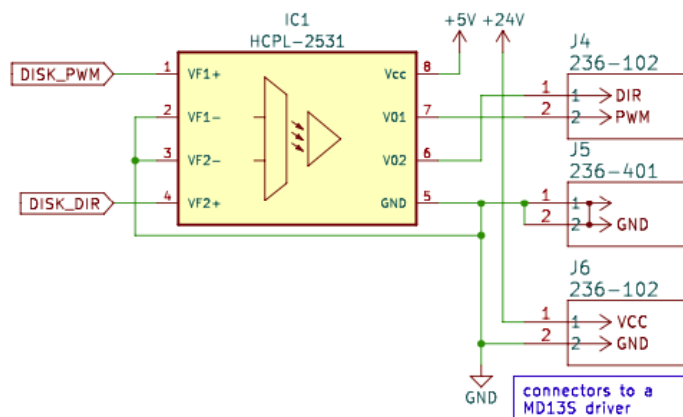
+ какво е ПА/ЛНА + режим на SPI интерфейс и неговата скорост



Фигура 5.1: Радиочестотен модул

### 5.1.2 Управление на моторите

Управлението на постоянно токовите четкови мотори се извършва посредством MD13S драйвери. Необходимите логически сигнали за да може да се реализира управлението са ШИМ сигнал, който упоменава скоростта, с която мотора трябва да се върти, и високо или ниско ниво, което да показва посоката на движение **????**<sup>1</sup>. С цел изолация между микроконтролера и драйверите са поставени оптрони. Начинът на свързване на всеки оптрон може да бъде видян на фиг. 5.2. Необходимото захранване на всеки мотор е 24V и то бива подадено посредством отделен конектор.

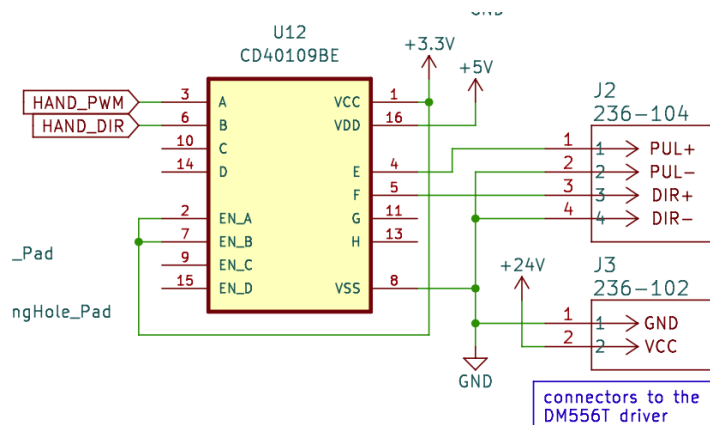


Фигура 5.2: Оптрон

За управлението на стъпковия мотор се използва драйвер DM556T. Сигналите за управление включват ШИМ сигнал, който определя стъпките за изпълнение и логическо ниво (високо или ниско), указващо

<sup>1</sup>Ужасно изказано

посоката на движение **????**<sup>2</sup>. Не е поставен допълнителен оптрон за изолация между драйвера и микроконтролера, поради причината, че на входовете си DM556T има вградени такива. Така направена схемата обаче няма да работи, защото логиката на blue pill-a е на 3,3V, докато драйвера използва 5V логика. Този проблем бива решен като се използва преобразувател на логически нива U12, който превежда логическото управление от 3,3V към 5V, съответстващо на логиката, използвана от драйвера.

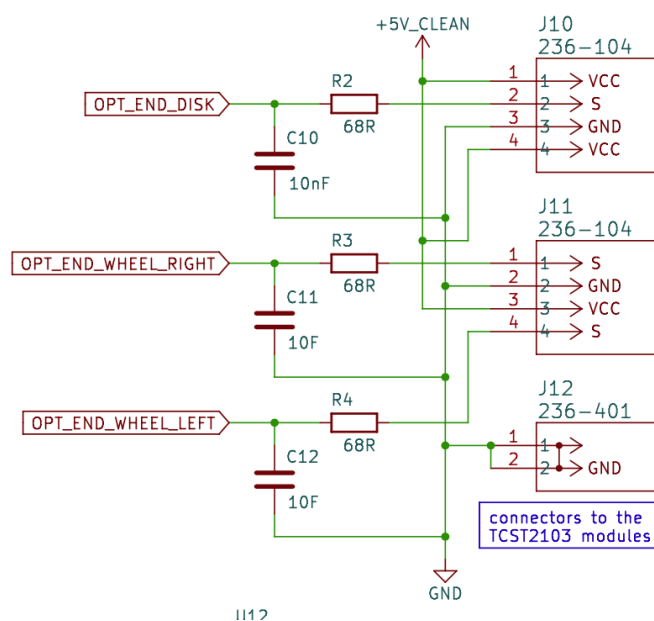


**Фигура 5.3:** Левел шифтър

### 5.1.3 Сензори

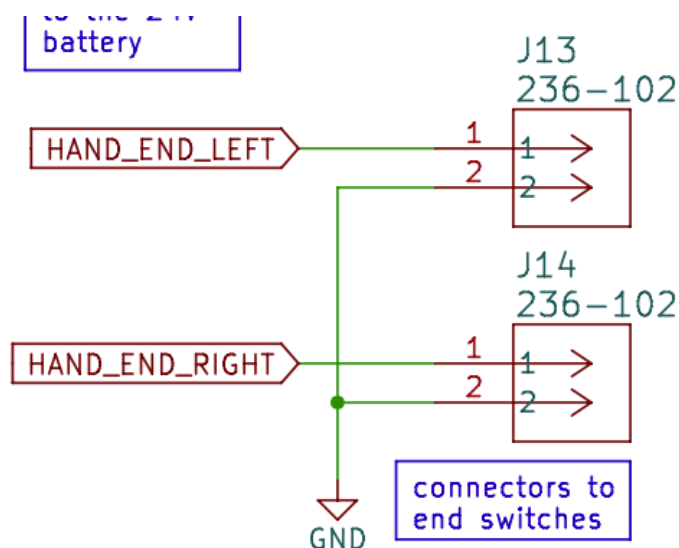
За следене на скоростта на работа са предвидени и инсталирани оптични сензори. Те биват свързвани към печатната плака чрез конекторите J10, J11 и J12. Всеки сензор има извод за данни и изводи за захранване.

<sup>2</sup>Ужасно изказано



**Фигура 5.4:** оптични сензори

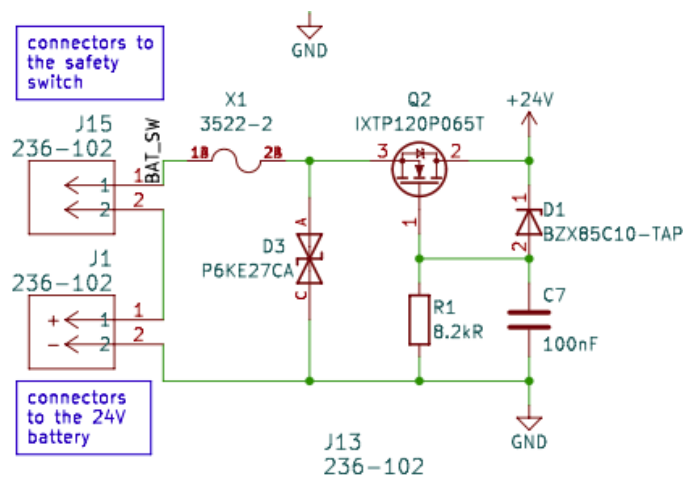
Съгласно изискванията за безопасност са предвидени и крайни изключватели, които да следят дали стъпковият мотор е достигнал някоя от крайните си позиции. Те са свързани, чрез конекторите



**Фигура 5.5:** Крайни изключватели за ръката

### 5.1.4 Захранване на печатната платка

Печатната платка се захранва директно от 24V постояннотокова батерия. Съгласно изискванията за безопасност изходите на конектор J15 са предвидени за ключ, който да спира цялото захранване на робота. Освен това има имплементирани хардуерни защиты против късо съединение, пренапрежение и обратен поляритет на захранването. Първата от трите е реализирана чрез поставянето на автомобилен бушон на държача X1 последователно свързан след главния ключ. Защитата против пренапрежение се осъществява посредством двупосочния трансил D3, който има номинално напрежение **????**. С цел застраховка против неправилен поляритет на напрежението е поставена P-MOS защита след трансила. Тя се реализира чрез транзистора Q2, ценеровия диод D1 и резистора R1. При правилното свързване на батерията през диода в Q1 протича ток. Поради образувания делител на напрежение D1 и R1, напрежението гейт-сорс на Q2 става равно на пробивното напрежение на D1 и транзисторът се отпушва. В случай, че батерията бъде свързана с обратен поляритет, диодът в Q1 бива свързан в обратна посока и транзисторът остава запушен. **????**<sup>3</sup>



Фигура 5.6: Защити на захранването

Захранването на логическата част на проекта е постигнато посредством линейни стабилизатори L7805. То е разделено на два канала за да може да бъдат намалени шумовете в захранването на микроконтролера и радиочестотния модул **????**<sup>4</sup>.

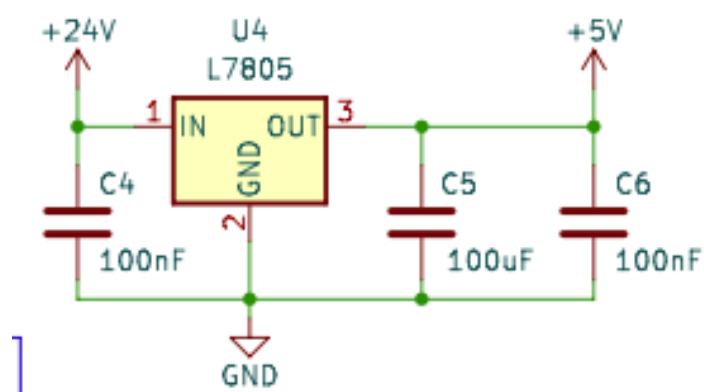
<sup>3</sup>Стабилизиране на напрежението чрез кондензатора C7

<sup>4</sup>Да кажа от къде идват шумовете

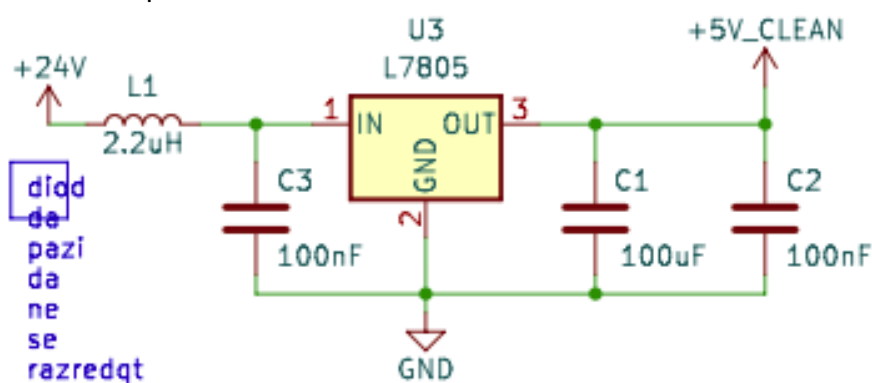


Първият канал използва стабилизатора U4 за да успее да свали напрежението до 5V, които да бъдат подадени после на оптроните на логическата част на четковите мотори и на логическия преобразовател на стъпковия мотор. Той е показан на фиг. 5.7а. За намаляване на шумовете в системата са поставени нискочестотния филтър C5 и височестотния филтър C6. <sup>5</sup>

Вторият канал на захранването е почти идентичен с първия. Той се използва за захранването на микроконтролера, радиочестотния модул и сензорите в работа и може да бъде видян на фиг. 5.7б. <sup>6</sup> + две захранвания не два канала



(a) Захранване на логиката на силовата електроника



(б) Захранване на микроконтролера, радиочестотния модул и сензорите

**Фигура 5.7:** Захранване на логическата част в работа.

<sup>5</sup>За допълнително стабилизиране на захранването е добавен кондензатора C4 - функция

<sup>6</sup>бобината L1

## 5.2 Опроводяване на печатната платка на работа

### 5.3 Монтажна схема на работа

Монтажната схема на работа може да бъде намерена в ?? приложение ?? . На нея може да се види основната част в работа, представляваща разработената печатна платка, която съдържа радиочестотния модул за комуникация, микроконтролера, управляващ работа, и хардуерните защиты на захранването. Нейното захранване идва от 24V батерия DeWalt DE0241 и съгласно изискванията за безопасност във всеки момент то може да бъде прекъснато чрез ключът SW1. Разработената платка захранва и управлява 3 постояннотокови четкови мотори с мощта на драйвери MD13S. С цел мониторинг на тяхната скорост е предвидено на всеки от тях да има монтиран по един оптичен сензор за скорост. Използваният нема 24 стъпков мотор се управлява посредством DM556T драйвер. Начинат на свързване на драйвера към печатната платка е конфигурация общ катод. Съгласно изискванията за безопасност чрез SW1 и SW2 се следи дали стъпковият мотор не е стигнал до крайните си състояния.

+ параметри на входовете на драйверите

### 5.4 Принципна електрическа схема на дистанционното

Принципната електрическа схема на печатната платка на дистанционното може да бъде видяна на ?? приложение ?? . Основният елемент на схемата е микроконтролерът STM32F103C8T6 (blue pill), който чете данните от сензорите, обработва ги и после ги праща към работа посредством радиочестотния модул. Използваните компоненти в проектирането на тази печатната платка могат да бъдат видяни в таблица ???? приложение ????

#### 5.4.1 Радиочестотен модул

#### 5.4.2 Сензори

### 5.5 Опроводяване на печатната платка на дистанционното

### 5.6 Монтажна схема на дистанционното

????<sup>7</sup>

---

<sup>7</sup>Трябва ли да има такова нещо изобщо

# Софтуерна реализация

Използваната среда за разработка на софтуерната част от дипломния проект е софтуерния пакет STM32CubeIDE на STMicroelectronics. Тази платформа е предназначена за изграждане на софтуерното управление на STM микроконтролери и микропроцесори. Чрез интеграцията на GNU GCC компилатор и GDB дебъгер, тя предоставя стабилна среда за разработка на C/C++ програми. В STM32CubeIDE е интегриран и графичния инструмент STM32CubeMX. Той улеснява конфигурацията и инициализацията на различни части и периферии на микроконтролерите, сред които са пинове, таймери, системни прекъсвания и DMA **????**<sup>1</sup>. Друга основна характеристика на избраната среда за разработка е възможността за comprehensive/advanced дебъгинг **????**<sup>2</sup> чрез мониторинг на процесорните ядра, регистрите на периферията, паметта и заделените променливи и други. Софтуерният пакет може да бъде свален безплатно от сайта на STMicroelectronics за 64-битовите версии на операционните системи Windows, Linux и macOS. Поради споменатите причини тази среда за разработка беше използвана за проекта.

## 6.1 Логическа схема на проекта

Съгласно заданието на дипломния проект разработеният боен робот бива управляван посредством специално изработено дистанционно. Комуникацията между тях бива реализирана посредством радиочестотните модули nRF24L01 на компанията Nordic. Те използват вградения си протокол за комуникация Enhanced ShockBurst, който е предназначен за лесна комуникация с ниска енергийна консумация.

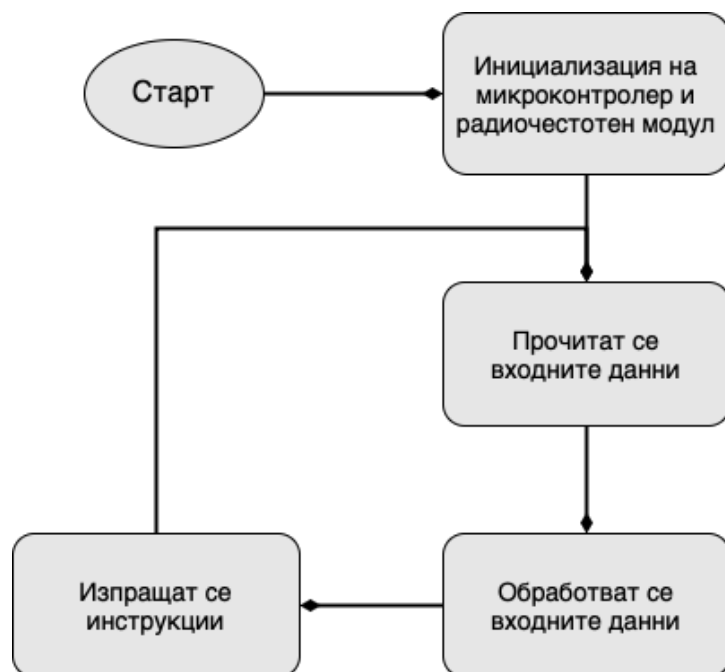
Като се получи захранване микроконтролера и радиочестотния модул в дистанционното биват инициализирани. След това започва непрекъснато повтарящият се на равни интервали цикъл на работа. Първо се прочитат входните данни от потенциометрите и бутоните. След

---

<sup>1</sup>Превод?

<sup>2</sup>Превод?

това те биват обработени и записани в подходящ вид. В края на този цикъл обработените потребителски команди биват записани в пакет с инструкции за контрол на бойния робот и той бива изпратен към батълбота за изпълнение. После цикъла на работа на дистанционното започва да се изпълнява отначало. Микроконтролерът е настроен да може да бъдат изпращани по 20 пакета с команди в секунда. Това скорост гарантира на пилота възможността да може да управлява своята машина с минимално системно забавяне.



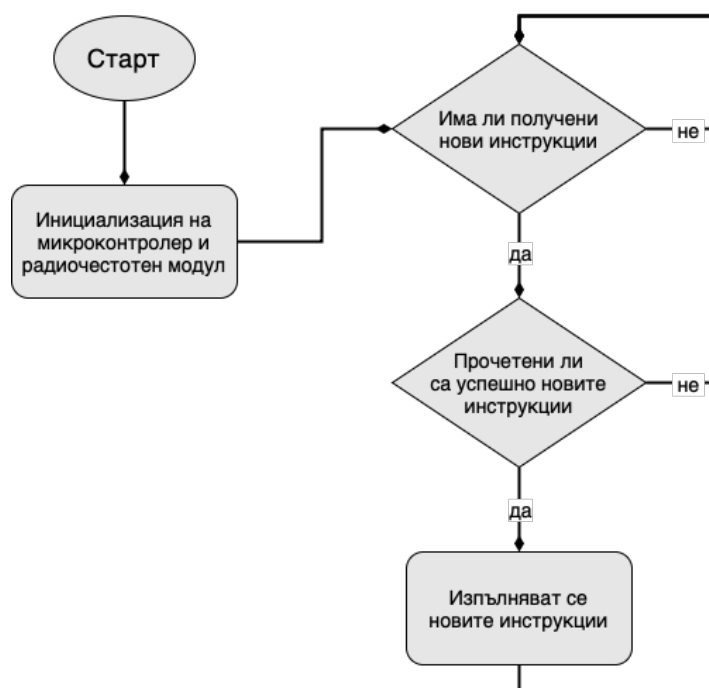
**Фигура 6.1:** Логическа схема на дистанционното

Полетата, които се съдържат в изпратения пакет са инструкция за позицията, на която трябва да застане механичната ръка на робота, инструкции за скоростта и посоката на въртене на двете колела и инструкция за това дали оръжието трябва да се върти. Структурата на такъв пакет може да бъде видяна на Програмен код 6.1.

```
11 typedef struct
12 {
13     int8_t hand_position; // hand position (-100 ~ +100)
14     int8_t left_speed;    // left wheel speed  (-100 ~ +100)
15     int8_t right_speed;   // right wheel speed (-100 ~ +100)
16     uint8_t weapon;      // weapon on/off (0 / 1)
17 } Payload;
```

**Програмен код 6.1:** Пакет инструкции

Подобно на дистанционното първата работа на печатната платка, когато получи захранване е да се инициализират микроконтролера и радиочестотния модул. След това започва непрекъснат цикъл на проверяване дали има получен пакет инструкции и ако има се прави опит той да бъде достъпен. При успешното им прочитане те биват заредени в паметта на платката и биват изпълнени. След тази стъпка следва повторно започване на работния цикъл на работа. В случаите, в които няма получени инструкции или не бъде успешно тяхното прочитане, цикъла на работа започва отначало и се спира движението на робота.



**Фигура 6.2:** Логическа схема на работа

## 6.2 Софтуерна реализация на библиотеката на радиочестотния модул

Безжичната комуникация между дистанционното и бойния робот се осъществява посредством радиочестотния модул nRF24L01 на компанията Nordic. За целите на дипломния проект не беше употребявана готова библиотека за управление на избрания модул, а беше разработена собствена такава. За нейната реализация са употребявани функционалностите на вградената HAL библиотека. С цел улеснение на

написването на библиотеката, в хедър файла **????**<sup>3</sup> са дефинирани като макроси адресите в паметта на регистрите и SPI командните думи. За да може библиотеката да работи, предварително трябва да бъде конфигуриран SPI интерфейса и пиновете, които се използват nRF модула. С цел улеснение на работата с библиотеката е предвидено предварително изводите на радиочестотния модул да имат зададени потребителските етикети NRF24L01\_CSN, NRF24L01\_CE и NRF24L01\_IRQ на съответните им пинове на микроконтролера.

Основните функционалности, които трябва да бъдат реализирани, за да може библиотеката да бъде разработена са писане и четене на регистрите на радиочестотния модул. За да се пише в регистър се подготвя масив с дължина 2 байта. В първия се поставя номера на регистъра, а във втория стойността, която трябва да бъде записана. За да може nRF модула да разбере, че трябва да запише получената стойност, трябва да бъде поставена единица на 5-та позиция от първия байт. Съгласно секция 8.3. "SPI комуникация" от документацията на nRF24L01 всяка SPI операция трябва да бъде започната с падащ фронт на CSN извода на радиочестотния модул. След това се извиква HAL функцията за предаване на информация по SPI интерфейса.

```
37 // write single byte to the particular register
38 void nrf24_write_reg (const uint8_t reg, const uint8_t data)
39 {
40     uint8_t buffer[2];
41     buffer[0] = reg | (1 << 5);
42     buffer[1] = data;
43
44     // Pull CSN pin LOW to select the device
45     csn_select();
46
47     HAL_SPI_Transmit(NRF24_SPI, buffer, 2, 1000);
48
49     // Pull CSN pin HIGH to unselect the device
50     csn_unselect();
51 }
```

### Програмен код 6.2: Писане в регистър

Четенето на регистър се реализира аналогично. При него не се подготвя масив, в който да се постави адреса на регистъра и той не бива редактиран. Вместо това първо се използва HAL функцията за преда-

---

<sup>3</sup>Превод?

ване на номера на регистъра, който трябва да бъде прочетен, и след това се използва HAL функцията за получаване на данни по SPI.

```
72 // read single byte from the particular register
73 uint8_t nrf24_read_reg (uint8_t reg)
74 {
75     uint8_t data = 0;
76
77     // Pull CSN pin LOW to select the device
78     csn_select();
79
80     HAL_SPI_Transmit(NRF24_SPI, &reg, 1, 100);
81     HAL_SPI_Receive(NRF24_SPI, &data, 1, 100);
82
83     // Pull CSN pin HIGH to unselect the device
84     csn_unselect();
85
86     return data;
87 }
```

### Програмен код 6.3: Четене на регистър

Основната част от регистрите в паметта на използвания nRF модул са с размер 1 байт, но тези, които съхраняват адреси за каналите за комуникация 0 и 1 имат 5 пъти по-голяма дължина. Поради тази причина са написани функциите `nrf24_write_reg_multi()` и `nrf24_read_reg_multi()`, които съответно могат да бъдат видени на Програмен код 6.4 и Програмен код 6.5. Те са реализирани аналогично на по-малките им подобни функции.

```
52 // write multiple bytes to the particular register
53 void nrf24_write_reg_multi (const uint8_t reg, const uint8_t *data, const size_t
    size)
54 {
55     uint8_t buffer[6];
56     buffer[0] = reg | (1 << 5);
57     buffer[1] = data[0];
58     buffer[2] = data[1];
59     buffer[3] = data[2];
60     buffer[4] = data[3];
61     buffer[5] = data[4];
62
63     // Pull CSN pin LOW to select the device
64     csn_select();
65
66     HAL_SPI_Transmit(NRF24_SPI, buffer, size + 1, 1000);
67
68     // Pull CSN pin HIGH to unselect the device
69     csn_unselect();
```



```
70 }
```

#### Програмен код 6.4: Писане на големи регистри

```
88 // read multiple bytes from the particular register
89 void nrf24_read_reg_multi (uint8_t reg, uint8_t *data, const size_t size)
90 {
91     // Pull CSN pin LOW to select the device
92     csn_select();
93
94     HAL_SPI_Transmit(NRF24_SPI, &reg, 1, 100);
95     HAL_SPI_Receive(NRF24_SPI, data, size, 1000);
96
97     // Pull CSN pin HIGH to unselect the device
98     csn_unselect();
99 }
```

#### Програмен код 6.5: Четене на големи регистри

### 6.2.1 Инициализация на модула

При получаване на захранване първата задача на микроконтролера и радио модула е да бъдат конфигурирани. Инициализацията на първия може да бъде видяна в раздел 6.3.1 и раздел 6.4.1. Привеждането на радиочестотния модул в състояние, което позволява безжично предаване на информация е осъществено като се извикват една след друга първо функцията за инициализация на модула и след това и допълнителна такава, с която се пояснява дали същия е предавател или получател на информацията.

При началото на инициализацията на CE пина се подава ниско ниво за да се гарантира, че няма да се активира механизма за изпращане на данни. След това се подават зануляват стойностите на CONFIG и RF\_CH регистрите. Те следва да бъдат редактирани при задаването на роля на модула **????**<sup>4</sup>. Чрез записване на 0 в EN\_AA регистъра се изключват автоматичните потвърждения по време на безжичното предаване. В регистъра EN\_RXADDR стойността 0 спира употребата на 6-те информационни канала **????**<sup>5</sup>. За да се дефинира дължината на адреса да бъде 5 байта в регистъра SETUP\_AW са поставени единици на битове 0 и 1.

---

<sup>4</sup>????

<sup>5</sup>data pipe

Последния регистър, който се конфигурира в рамките на тази функция е RF\_SETUP. Чрез поставяне на 0 на 3 бит в него се задава скоростта на предаване на информация да бъде 1Mbps, а посредством единици на битове 1 и 2, мощността на предаване на изхода на радиото е 0dBm.

```
145 // initialize the RF module
146 void NRF24_init (void)
147 {
148     // disable the device before initializing
149     ce_disable();
150
151     nrf24_write_reg(CONFIG, 0); // to be configured later
152     nrf24_write_reg(EN_AA, 0); // No auto ACK
153     nrf24_write_reg(EN_RXADDR, 0); // disabling the data pipes
154     nrf24_write_reg(SETUP_AW, 0x03); // TX/RX address length = 5 bytes
155     nrf24_write_reg(SETUP_RETR, 0); // No retransmissions
156     nrf24_write_reg(RF_CH, 0); // will be set up during TX or RX configuration
157     nrf24_write_reg(RF_SETUP, 0x06); // Output power = 0db, data rate = 1 Mbps
158 }
```

#### Програмен код 6.6: Инициализация на радио модула

След функцията за инициализиране на nRF модула се извиква функция, която да пояснява дали модула е предавател или получател на данни. В Програмен код 6.7 може да бъде видяна процедурата за конфигуриране предавател. Първо се задава в регистър RF\_CH честотата (канала), на която ще се предава информацията. След това се записва 5 байтовия адрес на получателя в TX\_ADDR регистъра. После без да се редактира останалата част от регистъра в CONFIG се поставят единица на бит 1 и се оставя нула на бит 0 съответно за да се стартира дейността на радиочестотния модул и за да се влезе в режим на изпращач на информация.

```
164 void NRF24_TX_mode (const uint8_t *address, const uint16_t channel)
165 {
166     nrf24_write_reg(RF_CH, channel); // select the frequency channel
167     nrf24_write_reg_multi(TX_ADDR, address, 5); // set up the TX address
168
169     // power up the device
170     uint8_t config = nrf24_read_reg(CONFIG);
171     config |= (1 << 1); // Power up; PTX
172     nrf24_write_reg(CONFIG, config);
173 }
```

#### Програмен код 6.7: Конфигуриране на изпращач на данни

Подобно на последно разглежданата функция NRF24\_RX\_mode() започва със задаване на честотата на предаване на информация. След това без да се редактира останалата част от регистъра EN\_RXADDR се поставя единица на бит 1 поради за да се позволи употребата на информационен канал 1. После се записва 5 байтовия адрес на изпращача на информация в RX\_ADDR\_P1 регистъра. За да се дефинира дължината на пакета данни, който ще се изпраща безжично, се записва големината на структурата Payload в регистъра RX\_PW\_P1. След това без да се реагира останалата му част в CONFIG регистъра се поставят единици на битове 0 и 1 съответно за да се стартира дейността на радиочестотния модул и за да се влезе в режим на получател на данни. В края на тази функция CE пина бива поставен във високо състояние и се държи в това през целия период на работа. В случай, че CE мине в ниско ниво, nRF модула ще излезе от режима на получател.

```
234 void NRF24_RX_mode (const uint8_t *address, const uint16_t channel)
235 {
236     nrf24_write_reg(RF_CH, channel); // select the frequency channel
237
238     // select data pipe 1
239     uint8_t en_rxaddr = nrf24_read_reg(EN_RXADDR);
240     en_rxaddr |= (1 << 1);
241     nrf24_write_reg(EN_RXADDR, en_rxaddr);
242
243     nrf24_write_reg_multi(RX_ADDR_P1, address, 5); // set up the TX address
244
245     nrf24_write_reg(RX_PW_P1, sizeof(Payload)); // payload lenght of pipe 1
246
247     // power up the device
248     uint8_t config = nrf24_read_reg(CONFIG);
249     config |= (1 << 1) | (1 << 0); // Power up; PRX
250     nrf24_write_reg(CONFIG, config);
251
252     // enable the device after configuring
253     ce_enable();
254 }
255
256 // check if data is received on specific pipeline
```

### Програмен код 6.8: Конфигуриране на получател на данни

Стойностите, които са избрани за скорост на безжичното предаване, мощността на предаване на изхода на nRF модула и честотата на излъчване на сигнали са подбрани такива, че да осигуряват максимален обхват на безжичната комуникация.

## 6.2.2 Изпращане на информация

Изпращането на пакет инструкции от дистанционното към робота се извършва посредством функцията NRF24\_transmit(). Първата стъпка от нейното изпълнение е пакета с инструкции да бъде зареден в паметта на радио модула, посредством функцията, която може да бъде видяна в Програмен код 6.10. След това се генерира импулс на СЕ извода по-дълъг от 10us. Съгласно секция 6.1.5. "режим предаване"от документацията на nRF24L01 при такъв импулс в режима за изпращане на данни, записаните в паметта на модула пакети инструкции започват да се излъчват. Останалата част на функцията проверява дали пакетите са били изпратени успешно като проверява съдържанието на FIFO\_STATUS регистъра. Ако четвъртият бит в него е вдигнат означава, че пакета е бил успешно изпратен.

```
204 // transmit the data
205 int NRF24_transmit (const Payload *payload)
206 {
207     push_payload(payload);
208
209
210     ce_enable();
211     HAL_Delay(1); // data send
212     ce_disable();
213
214
215     csn_select();
216
217     uint8_t fifostatus = nrf24_read_reg(FIFO_STATUS);
218     if ((fifostatus & (1 << 4)) && !(fifostatus & (1 << 3)))
219     {
220         nrf24_write_cmd(FLUSH_TX);
221
222         csn_unselect();
223         return 0; // success
224     }
225
226     csn_unselect();
227     return 1; // failure
228 }
```

### Програмен код 6.9: Изпращане на пакет инструкции

Когато се зарежда пакет инструкции в паметта на радиочестотния модул, първо се проверява дали пакета не е твърде голям. В случай, че е опита за безжично изпращане на инструкциите бива прекратен.

Останалата част от зареждането на пакета наподобява значително функциите за писане в регистри. Първо се заделя масив от еднобайтови променливи. След това на първа позиция в масива се записва SPI командата за записване на пакета в паметта на модула и чрез функцията `memcpy()` се записва съдържанието на пакета в масива от втория елемент нататък. Както беше споменато в предишната секция следва да се направи падащ фронт на пин CSN. Когато буфера е вече готов, той бива зареден в паметта, чрез HAL функцията за предаване на информация по SPI интерфейса. Накрая функцията завършва като CSN пина ве върне в свойто предишно състояние.

```
175 int push_payload (const Payload *payload)
176 {
177     if (32 < sizeof(Payload))
178     {
179         return 1; // too big size
180     }
181
182     uint8_t *buffer = (uint8_t*) calloc(sizeof(Payload) + 1, sizeof(uint8_t));
183     if (NULL == buffer)
184     {
185         return 2; // calloc error
186     }
187
188     buffer[0] = W_TX_PAYLOAD;
189     memcpy(buffer + 1, payload, sizeof(Payload));
190
191     // Pull CSN pin HIGH to LOW
192     csn_unselect();
193     csn_select();
194
195     HAL_SPI_Transmit(NRF24_SPI, buffer, sizeof(Payload) + 1, 1000);
196
197     free(buffer);
198
199     // Pull CSN pin HIGH to unselect the device
200     csn_unselect();
201     return 0; // success
202 }
```

**Програмен код 6.10:** Зареждане на пакет инструкции

### 6.2.3 Получаване на информация

Получаването на пакети с инструкции се реализира чрез две функции. Първата проверява дали има нова получена информация в радио моду-

ла, а втората е същинското и извличане в паметта на микроконтролера. Проверката се осъществява като се прочита STATUS регистъра. Ако бит 6 е единица следва, че има получени нови команди, а в битове 1, 2 и 3 е запазен номера на информационния канал, от който е получената информация. След това стойността на същия регистър се рестартира и функцията връща резултат, показващ, че има получени данни.

```
256 // check if data is received on specific pipeline
257 int is_data_received (const int pipenum)
258 {
259     csn_select();
260
261     uint8_t status = nrf24_read_reg(STATUS);
262
263     if ((status & (1 << 6)) && (status & (pipenum << 1)))
264     {
265         nrf24_write_reg(STATUS, (1 << 6));
266
267         csn_unselect();
268         return 0; // data received; success
269     }
270
271     csn_unselect();
272     return 1; // data not received; success
273 }
```

#### **Програмен код 6.11:** Проверка дали има получен пакет инструкции

Във функцията за извличане на данни първата първо се дефинират двата помощни масива. В първия се записва SPI командата R\_RX\_PAYLOAD, която се използва за да се извлече получения нов пакет информация, а във втория следва тя да се бъде записана. След това посредством HAL функцията за предаване и получаване на информация по SPI интерфейса командата се изпраща до радиочестотния модул и се получава резултата във втория помощен масив. После чрез функцията memcpy() информацията се премества от помощния регистър в желаната променлива.

```
275 // receive data
276 int NRF24_receive (Payload *payload)
277 {
278     // select the device
279     csn_select();
280
281     // payload command
282     uint8_t TX_buffer[sizeof(Payload) + 1];
283     TX_buffer[0] = R_RX_PAYLOAD;
```

```

284
285     uint8_t RX_buffer[sizeof(Payload) + 1];
286
287     // receive the payload
288     HAL_SPI_TransmitReceive(NRF24_SPI, TX_buffer, RX_buffer, sizeof(Payload) + 1,
        1000);
289
290     memcpy(payload, RX_buffer + 1, sizeof(Payload) + 1);
291
292
293     HAL_Delay(1);
294
295     // unselect the device
296     csn_unselect();
297
298     return 0; // success
299 }

```

**Програмен код 6.12:** Получаване на пакет инструкции

## 6.3 Софтуерна реализация на дистанционното

Както е описано в раздел 3.2 компонентите, които се използват в дистанционното са 2 потенциометъра и 3 бутона.

### 6.3.1 Инициализация на микроконтролера на дистанционното

След като получи захранването си микроконтролера преминава през серия от процеси, които трябва да конфигурират използваните периферии, изводи и променливи. Посредством функцията HAL\_Init() се рестартират всички периферии и се инициализират HAL библиотеката и всички нейни функционалности. След това чрез SystemClock\_Config() се конфигурира системния таймер. После следва конфигурацията на изводите на микроконтролера. Една от тях може да бъде видяна в Програмен код 6.13. За това кой пин как е конфигуриран и за какво се използва може да се разбере от таблица табл. 6.1. След това се инициализира SPI връзката с nRF модула. Чрез MX\_TIM4\_Init() се инициализира таймер 4, с помощта на когото се изпращат пакетите с инструкции на равни периоди. След това се инициализират и двата

аналогово-цифрови преобразувателя, които се използват за четене на състоянието на потенциометрите. Последната инициализация е тази на радиочестотния модул и се задава неговия режим на работа. Преди да започне повтарящият се цикъл на работа таймера се стартира и се пуска калибрация на аналогово-цифровите преобразуватели.

```

72  /*Configure GPIO pin : PtPin */
73  GPIO_InitStruct.Pin = WEAPON_Pin;
74  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
75  GPIO_InitStruct.Pull = GPIO_PULLUP;
76  HAL_GPIO_Init(WEAPON_GPIO_Port, &GPIO_InitStruct);

```

### Програмен код 6.13: Конфигуриране на пин

	Wi-fi	Bluetooth	nRF24L01+
Скорост	Висока	Средно	Средна
Обхват	10ки метри	10 метра	10-150 метра
Енергийна консумация	Висока	Средна	Ниска

Таблица 6.1: Пинове

## 6.3.2 Работен цикъл на дистанционното

Както е описано в раздел 6.1 цикъла на работа се повтаря на равни интервали. Това е постигнато посредством прекъсването при преливане на таймер 4. Той е конфигуриран така, че прекъсването да бъде генерирано на всяка десета от секундата. След извикването на прекъсването се вдига флага `send_flag`. При това действие се прочитат състоянията на потенциометрите и се записват на полетата за скорост на лявото и дясното колело. Това е последвано от извикването на функцията за калибриране на данните. В края на този цикъл се инициира изпращане на пакета с инструкциите и при успех се сменя състоянието на вградения светодиод.

```

113  /* Infinite loop */
114  /* USER CODE BEGIN WHILE */
115  while (1)
116  {
117      if (send_flag)
118      {

```



```

119     send_flag = 0;
120
121     HAL_ADC_Start_IT(&hadc1);
122     HAL_ADC_Start_IT(&hadc2);
123     payload_manipulation();
124
125     if (0 == NRF24_transmit(&payload))
126     {
127     HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
128     }
129 }
130
131 /* USER CODE END WHILE */
132
133 /* USER CODE BEGIN 3 */
134 }

```

#### Програмен код 6.14: Работен цикъл

Успоредно на това има конфигурирани и външни прекъсвания. Те имат функцията да редактират позицията на ръката и да пускат или спират движението на оръжието.

## 6.4 Софтуерна реализация на управлението на работата

### 6.4.1 Инициализация на микроконтролера на работата

След като получи захранването си микроконтролера преминава през серия от процеси, които трябва да конфигурират използваните периферии, изводи и променливи. Посредством функцията HAL\_Init() се рестартират всички периферии и се инициализират HAL библиотеката и всички нейни функционалности. След това чрез SystemClock\_Config() се конфигурира системния таймер. После следва конфигурацията на изводите на микроконтролера. Една от тях може да бъде видяна в Програмен код 6.15. За това кой пин как е конфигуриран и за какво се използва може да се разбере от таблица табл. 6.2. След това се инициализира SPI връзката с nRF модула. Чрез функциите MX\_TIM3\_Init(), MX\_TIM2\_Init() и MX\_TIM4\_Init() се инициализират таймери 2, 3 и 4. Таймер 3 се използва за генериране широчинно импулсен сигнал за контролиране на скоростта на въртене на четковите мотори, а 4-тия се

използва за измерване на тяхната скорост. Таймер 2 се използва за генерирането на импулси за движението на робота. След това се инициализира радиочестотния модул и се задава неговия режим на работа. Чрез функцията HAL\_TIM\_PWM\_Start() се стартира генерирането на широчинно импулсния сигнал. Преди да започне повтарящият се цикъл на работа и таймери 2 и 4 се стартират.

```
72  /*Configure GPIO pin : PtPin */
73  GPIO_InitStruct.Pin = WEAPON_Pin;
74  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
75  GPIO_InitStruct.Pull = GPIO_PULLUP;
76  HAL_GPIO_Init(WEAPON_GPIO_Port, &GPIO_InitStruct);
```

**Програмен код 6.15:** Конфигуриране на пин

	Wi-fi	Bluetooth	nRF24L01+
Скорост	Висока	Средно	Средна
Обхват	10ки метри	10 метра	10-150 метра
Енергийна консумация	Висока	Средна	Ниска

**Таблица 6.2:** Пинове

## 6.4.2 Работен цикъл на робота

Както е описано в раздел 6.1 цикъла на работа започва с проверка на това дали има получен нов пакет инструкции. Ако има получен такъв се извиква функцията за извличане на данните от радиочестотния модул и ако е успешна се извиква функцията, която прилага инструкциите.

```
122  while (1)
123  {
124      // TODO: replace the pooling
125      if (0 == is_data_received(1))
126      {
127          if (0 == NRF24_receive(&payload))
128          {
129              unload_payload();
130          }
131      }
132      /* USER CODE END WHILE */
133
134      /* USER CODE BEGIN 3 */
```

135 }

### Програмен код 6.16: Работен цикъл

Функцията за прилагането на инструкциите е съставена от помощни функции. Първо чрез функцията `get_brushed_DIR()` се проверява в коя посока кой от четковите мотори трябва да се върти и се задава съответната стойност на съответния извод. След това посредством `get_brushed_DIR()` се задава какъв трябва да бъде коефициента на запълване на широчинно-импулсния сигнал, който се използва за дава каква трябва да бъде скоростта на въртене на четковите мотори.

```
249 void get_brushed_DIR(void)
250 {
251     HAL_GPIO_WritePin(DISK_DIR_GPIO_Port, DISK_DIR_Pin, SET);
252
253     if (payload.left_speed < 0)
254     {
255         payload.left_speed *= -1;
256         HAL_GPIO_WritePin(WHEEL_LEFT_DIR_GPIO_Port, WHEEL_LEFT_DIR_Pin, RESET);
257     }
258     else
259     {
260         HAL_GPIO_WritePin(WHEEL_LEFT_DIR_GPIO_Port, WHEEL_LEFT_DIR_Pin, SET);
261     }
262
263     if (payload.right_speed < 0)
264     {
265         payload.right_speed *= -1;
266         HAL_GPIO_WritePin(WHEEL_RIGHT_DIR_GPIO_Port, WHEEL_RIGHT_DIR_Pin, RESET);
267     }
268     else
269     {
270         HAL_GPIO_WritePin(WHEEL_RIGHT_DIR_GPIO_Port, WHEEL_RIGHT_DIR_Pin, SET);
271     }
272 }
273
274 void get_brushed_PWM(void)
275 {
276     // ARR => TOP
277     TIM3->CCR2 = enable[0] * payload.weapon * TIM3->ARR;
278     TIM3->CCR3 = enable[2] * payload.left_speed * TIM3->ARR;
279     TIM3->CCR4 = enable[1] * payload.right_speed * TIM3->ARR;
280 }
```

### Програмен код 6.17: Управление на четковите мотори

Последната функция, която се извиква във функцията `unload_payload()` се казва `get_hand_DIR()`. Тя се използва за да се определя посоката на

движение на стъпковия мотор. Импулсите за колко стъпки да бъдат направени се генерират чрез прекъсването при преливане на таймер 2. Функцията за обработка на прекъсването може да бъде видяна в прогн. код 6.20.

```
282 void get_hand_DIR(void)
283 {
284     if (hand_enable[0] && payload.hand_position < 0)
285     {
286         payload.hand_position *= -1;
287         HAL_GPIO_WritePin(HAND_DIR_GPIO_Port, HAND_DIR_Pin, RESET);
288     }
289     else if (hand_enable[1])
290     {
291         HAL_GPIO_WritePin(HAND_DIR_GPIO_Port, HAND_DIR_Pin, SET);
292     }
293     else
294     {
295         payload.hand_position = 0; // stop
296     }
297 }
```

**Програмен код 6.18:** Управление на стъпковия мотор

### 6.4.3 Сензори за обратна връзка

Съгласно изискванията за безопасност описани в раздел 3.1 са имплементирани и двата крайни изключвателя за ръката. Те са свързани към изводи на микроконтролера, за които има конфигурирани външни прекъсвания, които се генерират при падащ и нарастващ фронт на вълната. Когато такова прекъсване се появи се вдига флаг, който спира работата на движението на ръката докато ръката не отвори отново крайния изключвател.

```
184 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
185 {
186     if (GPIO_Pin == OPT_END_DISK_Pin)
187     {
188         cycles[0]++;
189     }
190     if (GPIO_Pin == OPT_END_WHEEL_RIGHT_Pin)
191     {
192         cycles[1]++;
193     }
194     if (GPIO_Pin == OPT_END_WHEEL_LEFT_Pin)
```

```

195 {
196     cycles[2]++;
197 }
198
199
200
201 if (GPIO_Pin == HAND_END_RIGHT_Pin)
202 {
203     hand_enable[0] = !hand_enable[0];
204 }
205 if (GPIO_Pin == HAND_END_LEFT_Pin)
206 {
207     hand_enable[1] = !hand_enable[1];
208 }
209 }

```

### Програмен код 6.19: Външни прекъсвания

В проекта има монтирани и оптични сензори, чрез които се следи приблизителната скорост на четковите мотори. Тази функционалност е реализирана посредством външни прекъсвания и прекъсване при преливане. При генерирането на първото се увеличава променлива, която съдържа броя завъртания на съответния мотор. Чрез прекъсването при преливане на таймер 4, на равни интервали от време, се проверява дали има направени завъртания за текущия период. Ако няма направени следва да бъде забранено на съответния мотор да се върти известно време.

```

211 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
212 {
213     if (&htim2 == htim)
214     {
215         if (2 * payload.hand_position != hand_counter)
216         {
217             hand_counter++;
218             HAL_GPIO_TogglePin(HAND_PWM_GPIO_Port, HAND_PWM_Pin);
219         }
220         else
221         {
222             payload.hand_position = 0;
223             hand_counter = 0;
224         }
225     }
226
227     if (&htim4 == htim)
228     {
229         for (int i = 0; i < 3; i++)
230         {
231             if (enable[i] && 0 == cycles[i])

```

```
232     {
233     enable[i] = 0;
234     cycles[i] = 0;
235     }
236     else if (!enable[i] && 4 > cycles[i])
237     {
238     cycles[i]++;
239     }
240     else
241     {
242     enable[i] = 1;
243     cycles[i] = 0;
244     }
245     }
246 }
247 }
```

**Програмен код 6.20:** Прекъсвания при преливане

## Заключение