



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

КУРСОВ ПРОЕКТ

по предмет “Вградени микрокомпютърни системи”, 11 клас

Тема: Изграждане на противопожарен робот,
управляван с помощта на радио контролер

Изготвили:

Боян Жечев

Ростислав Ангелов

Преподавател:

маг. инж. Росен Витанов

**СОФИЯ
2023-2024**

УВОД

За по-малко от век бурното развитие на електротехниката и радиотехниката доведе до обособяването на нови науки, като полупроводникова електроника, телевизия, изчислителна техника, автоматика, радиоуправление, кибернетика и др., без които е немислим прогресът на човечеството.

Скоростното развитие на електрониката и микроелектрониката през последните десетилетия доведе до създаването на "интелигентни" интегрални схеми, наречени микроконтролери. Микроконтролерът е едночипова завършена компютърна система и представлява малък микрокомпютър, събран в един корпус, и с ограничени ресурси, в сравнение с така познатият ни персонален компютър [1].

Днес микроконтролерите се използват във всички сфери на живота. Почти всички съвременни електронни устройства използват микроконтролери, като печки, перални, хладилници, мрежови устройства и т.н.

Радиоелектрониката е ключова част от съвременните технологии. Тя включва изследването, разработката и прилагането на радио комуникации и свързани технологии. Радиоелектрониката е задължителна за функционирането на безжични мрежи, сателитни комуникации, съобщителните системи и т.н.

Микроелектрониката има голям напредък в пожарната безопасност и защита на населението. Позволява се създаването на изключително чувствителни детектори за дим и топлина, както и разработката на интелигентни контролни системи за управление на пожари. Не на последно място микроелектрониката съдейства за създаването на автономни роботи и машини, които могат да се използват за проникване в опасни зони и борба с пожарите. Тези системи могат да бъдат дистанционно управлявани и да предоставят важна информация на екипите за спасяване.

ГЛАВА 1. Проблемът и конкуренцията

1.1 Описание на проблема:

Пожарите са заплаха както за човешкия живот, така и за имуществото. Според националната противопожарна служба на САЩ 39% от всички пожари са строителни пожари. Този тип пожари водят до значителни загуби на живот, наранявания и унищожаване на имущество на стойност милиони долари. Държавата е предприела мерки за ограничаване на опустошително унищожаване на пожари чрез политики и агенции, посветени на реагирането на пожари като противопожарни служби.

Пожарникарите се опитват да реагират бързо на пожари и дори да изложат живота си на рисък. Те полагат усилия за спасяване на човешки живот и защита на имуществото от пожари.

Въпреки че мъжете и жените, работещи в противопожарната служба, са добре оборудвани и обучени, по-често, поради непредсказуемата среда, създадена от пожарите, пожарникарите могат да бъдат ранени или дори да умрат по време на изпълнение на задълженията си. Това е нежелателно в настоящото технологично пространство. Пожарникарите трябва да работят в по-добри условия за безопасност и ефективност.

Проектирането и производството на пожарогасителен робот трябва да изпълняват конкретни цели, например:

1. Да се проектира и внедри тестова среда;
2. Да се проектира и внедри алармена система;
3. Да се проектира и внедри система за пожароизвестяване;
4. Да се проектира и изработи рамката на робота;
5. Да се проектира и внедри система за водоснабдяване;
6. Да се проектира и реализира задвижващата система;
7. Да се проектира и внедри системата за управление.

Разбира се, трябва да се съобразят следните ограничения:

1. Работът не може да изкачва стълби;
2. Необходими са високи умения за изпълнение на някои раздели;
3. Някои части като гумените колела се стопяват при висока температура.

Ето и някои допълнителни функции, които могат да се имплементират:

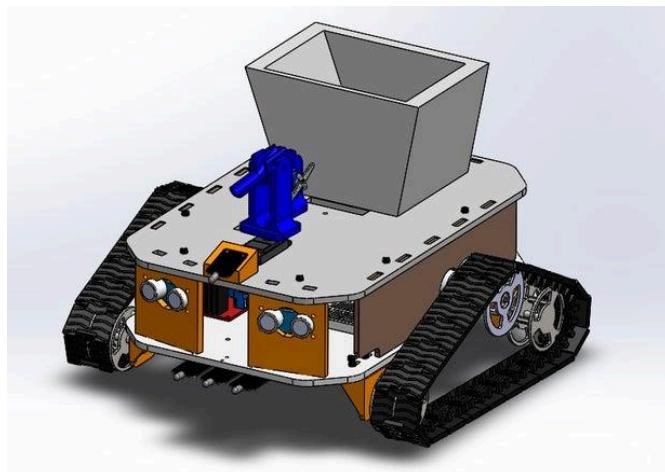
- да има робот, който може да бъде навигиран от дистанционно/радио контролер, като по този начин премахва човешкия елемент от суровите условия на горяща зона;

- да има камера, осигуряваща адекватно виждане на горящата сграда/структура, за да помогне при насочването роботът да гаси пожар на подходящото място;
- да има добре калибрирани сензори и точност и чувствителност за откриване на пожари с прецизност;
- да има ефективен механизъм за разпръскаване на вода с възможност за гасене на пожари от желаното разстояние.

1.2 Проучване на конкуренцията:

1.2.1 Firefighter Robot:

Отбор Firefighter_VUB, състоящ се от 4 члена, е автор на един-единствен проект в уебсайта Instructables, показва своето решение [2] за направата на пожарна машина. Проектът му представлява робот (Фиг. 1.1), направен да открива пожар, посредством сензори за пламък, отивайки към и потушавайки огъня с вода. Той също така може да избягва препятствия, докато върви към огъня с помощта на ултразвукови сензори. В допълнение, той изпраща имейл на автора, когато потуши огъня.



Фиг. 1.1 Проект “FireFighter Robot” на отбор Firefighter_VUB

Компонентите, които се използват, са както следва: Arduino Mega - 1 брой; 9V DC мотор - 2 броя; Микро серво 9g - 1 брой; Серво мотор 442h - 1 брой; Водна помпа - 1 брой; Ултразвуков звуков сензор - 2 броя; Сензор за пламък - 4 броя; H-bridge - 2 броя; Wi-Fi модул - 1 брой; Превключвател за включване/изключване - 1 брой; Мини breadboard - 1 брой; Arduino кабели; 9V батерия - 1 брой; 9V щепсел за батерия - 1 брой; LiPo 7.2Volt батерия - 1 брой; Комплект гумени вериги - 2 броя; Двигатели - 2 броя; Резервоар за вода (300 ml) - 1 брой; Маркуч за вода - 1 брой.

За механизма на водния пистолет отборът се нуждае от мотори, които могат да осигурят относително прецизно движение в определен диапазон. Във връзка с това

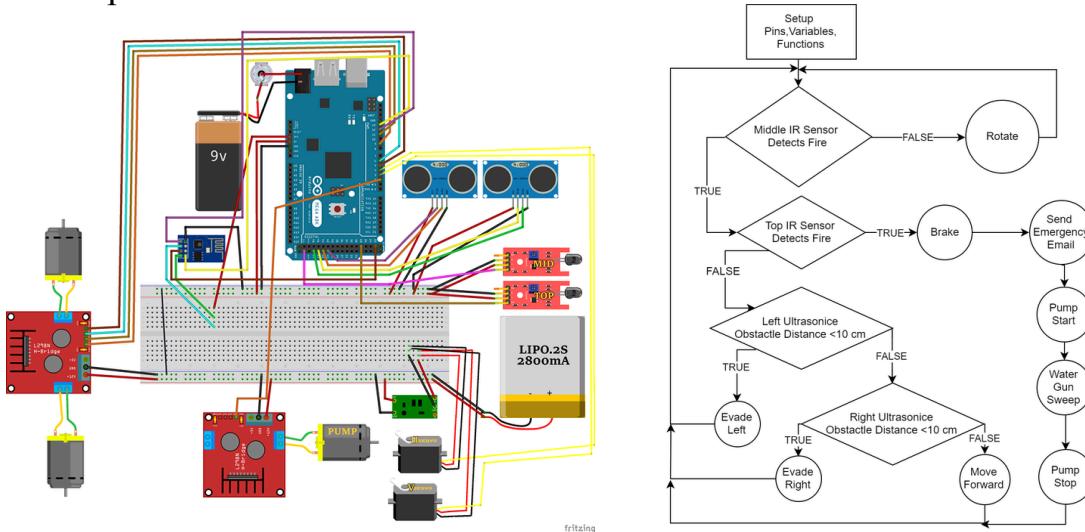
отборът се чуди между серво мотор или стъпков мотор. Избрано е да се използва батерия за захранване за всички двигатели, поради което в крайна сметка е взето решение да се използват серво мотори, най-вече защото те консумират по-малко енергия от стъпковите.

Отборът използва H-bridge, за да контролира посоката на въртене и на двета DC мотора (свързани към задвижващи колела). В допълнение, друг H-bridge се използва като обикновен превключвател за включване/изключване на водната помпа.

Двата ултразвукови сензора в проекта служат за избягване на препятствия. Също така отборът използва общо четири сензора за пламък. 3 сензора под шасито са свързани към аналогови и цифрови пинове на Arduino. Функцията на четвъртия сензор отгоре е да изпрати команда за спиране на автомобила на подходящо разстояние от пожара, така че в момента, в който сензорът отгоре открие пожара, той да изпрати команда за спиране на превозното средство и да се стартира водната помпа и съответно да се пусне водният пистолет, който да потуши огъня.

Главната причина за избора на микроконтролер Arduino Mega пред Arduino Uno е следната: Наличието на Wi-Fi модул увеличава драматично броя на редовете в кода и се нуждае от по-мощен процесор, за да се избегне възможен шанс за срив при изпълнение на кода. Гumenите вериги пък се използват за избягване на проблеми или подхлъзване в случай на хълзгав под или малки предмети по пътя на движение, както и за изкачване на стълби или други обекти.

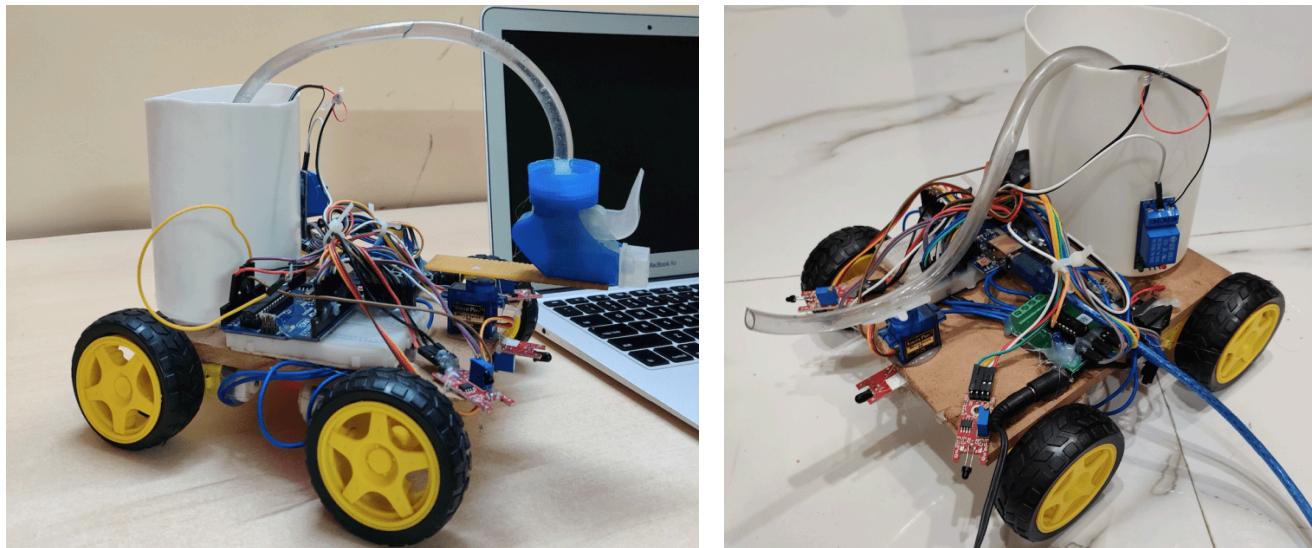
Лазерно изрязаните части са: Шаси (плексиглас 6mm) - 1 брой; Горна част (плексиглас 6mm) - 1 брой; Задна част (MDF 3mm) - 1 брой; Страницна част (MDF 3mm) - 2 броя. 3D печатаните части са: Ултразвуков държач - 2 броя; Държач за сензор за пламък - 1 брой; Държач за колесни лагери на колело - 4 броя; Настройка на воден пистолет - 1 брой. На Фиг. 1.2 можем да видим функционалната и блоковата схема на проекта:



Фиг. 1.2 Функционална и блокова схема на проекта

1.2.2 Fire fighting Robot:

Друг проект, свързан с изработка на пожарна машина, е този, публикуван в уебсайта Techatronic от неизвестен автор. Направена е машина [3], която работи автоматично, приближавайки се до огъня и изгасяйки го с помощта на воден механизъм. Пожарната машина (Фиг. 1.3) е много впечатляваща и може да се използва вместо превозно средство.



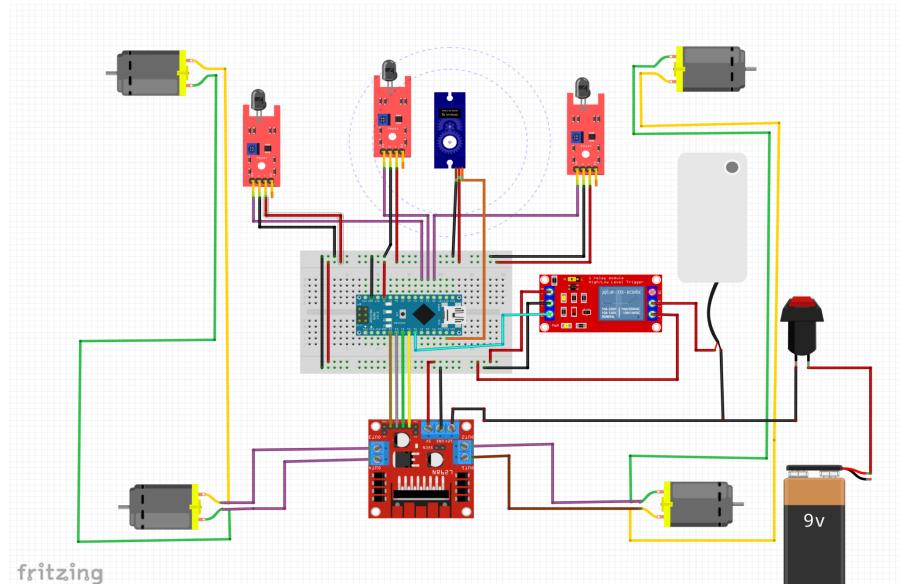
Фиг. 1.3 Проект “Fire fighting Robot”

Целта на този проект е да се проектира и разработи евтин противопожарен робот, който работи автоматично и гаси огън. Изработеният робот може сам да открива пожар и да контролира огъня, като хвърля вода. Той се състои от сензори, които откриват пожар и роботът се придвижа там, за да загаси пожара. Роботът има три сензора: един от предната страна, който следи дали има нещо пред робота и още два в двата предни ъгъла, които също търсят и засичат огън. Ако някой сензор открие пожар, на което и да е място, роботът ще го засече и ще се придвижи към него. Пожарната машина има 4 колела, 3 сензора, едно едноканално реле, един бутоон за пускане/спиране, една дюза за вода, една водна помпа (5V) и маркуч, един серво мотор и един микроконтролер Arduino, които му помогат да вземе решение според написания код. Захранва се с 9 - 12V батерия.

Ето малко повече детайли за работата на проекта: трите инфрачервени сензора за пламък непрекъснато търсят огън или пламък. Сензорът за пламък усеща огъня и изпраща информацията до Arduino, който е мозъкът на този робот. Микроконтролерът ще предприеме действия според състоянието и информацията, получена от сензора. Arduino ще даде команди на двигателите да започнат да вървят в желаната посока. Ако левият сензор даде информация за пожара, тогава Arduino ще задвижи двигателя в лява посока. Ако десният сензор даде информация за пожара, тогава Arduino ще задвижи двигателя в дясна посока. Ако средният сензор

даде информация за пожара, тогава Arduino ще задвижи и двата двигателя напред. Роботът ще спре близо до огъня и ще започне да хвърля вода, докато той бъде овладян и загасен напълно.

Функционалната схема на целия проект е представена на Фиг. 1.4:



Фиг. 1.4 Функционална схема на проекта

1.2.3 Little robot that fight fires:

Да си пожарникар - това наистина е една много опасна професия, но е възможно да се смекчи част от тази опасност. Когато пожарникарите влязат в горяща сграда, най-големият им страх е неизвестното. Те не знаят дали могат да се доверят на структурната цялост на сградата, и как да навигират във вътрешността, за да намерят жертви. Като част от проект [4], наречен “HelpResponder”, екип от изследователи от Universidad Rey Juan Carlos и Universidad Autónoma de Madrid създава робот, който може да влезе в сграда, за да събере данните, от които пожарникарите се нуждаят, за да вършат работата си безопасно.

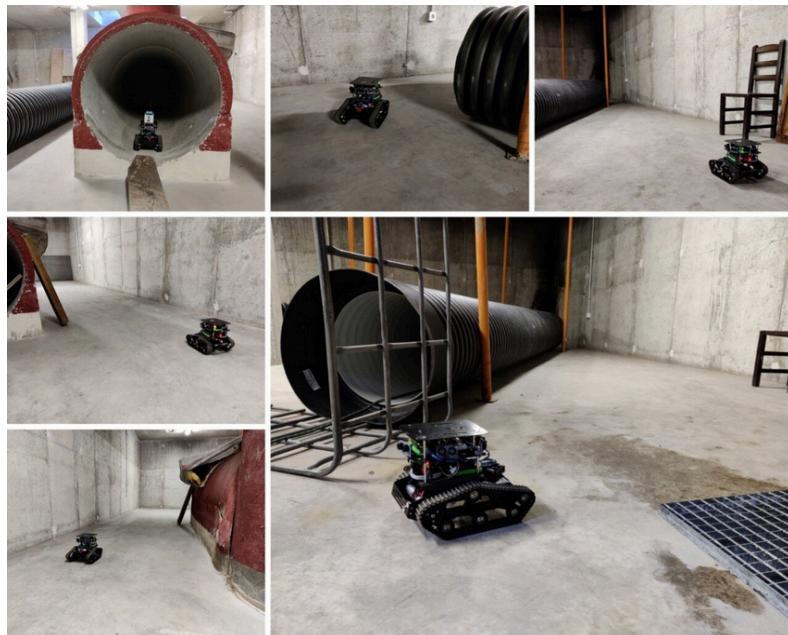
Този робот, който е средно голям, може да работи чрез ръчно управление или в автономен режим. И в двата случая работата му е да изследва сгради, по време на пожар или след бедствие, да картографира вътрешността и да открие опасности. Неговата система от камери позволява визуално откриване, но също така има множество интегрирани сензори за откриване на повишени температури и др. С тази информация пожарникарите могат след това да влязат в сградата и да спасят всеки, хванат в капан вътре, като избягват опасните зони или носят необходимото оборудване за справяне с тях. На Фиг. 1.5 можем да видим как изглежда създадената пожарна машина:



Фиг. 1.5 Проект “HelpResponder”

Контролът и наблюдението се осъществяват на две нива. На високо ниво, single-board компютър “Raspberry Pi 4 Model B” записва видео, обработва картографски операции и координира автономна навигация. На ниско ниво Arduino UNO WiFi Rev.2 събира входящи данни от сензори и управлява драйвера на двигателя. Вградените сензори включват сензор за температура/влажност, сензор за качеството на въздуха и ултразвукови сензори за навигация. Благодарение на модулния дизайн може да се добави допълнителен хардуер, за да отговаря на конкретни сценарии.

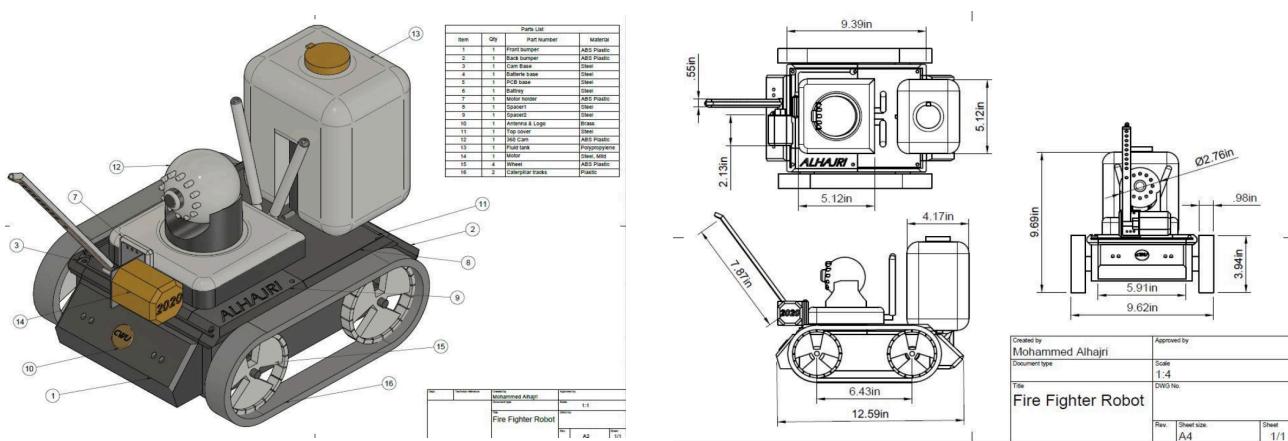
Екипът планира да продължи да подобрява робота, особено автономния му режим на работа. Но те вече са го тествали в симулации и в реалния свят (Фиг. 1.6) с положителни резултати. Ето няколко снимки от тестването на проекта им:



Фиг. 1.6 Тестване на проекта на отбор “HelpResponder”

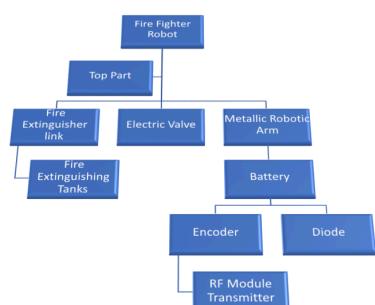
1.2.4 Fire Fighter Robot:

Мохамед Алхаджри е автор на проект, публикуван в уебсайта ScholarWorks, свързан с направата на пожарна машина. Авторът споделя, че в ерата на технологиите трябва да се полагат усилия, в които рисковите работни места трябва да се дават на машини. Проектът му [5] съдържа камера и ръка за хвърляне на вода, която е прикрепена към горната част на проекта. Движението на робота (Фиг. 1.7) зависи от правилното програмиране в софтуерната част. Проектът (Фиг. 1.8) трябва да се използва за малък огън. Пожарната машина усеща наличието на огън, локализира пожара чрез своята камера, след което водата се хвърля в огъня в рамките на 2 секунди след откриване на пожара, в разстояние повече от 2 метра.



Фиг. 1.7 Проект “Fire Fighter Robot”

На фигураната се вижда ръката, която хвърля водата в огъня. Моторизираната ръка води дюзата на тръбата за течност нагоре или надолу. Също така една камера е прикрепена в горната част. Тази 360-градусова камера служи като око на робота, която може да гледа във всички посоки и е с нощно виждане, може да се наблюдава и контролира безжично чрез смартфон. След засичането на огъня, сигналите се предават и количката следва огъня. В задната част на робота се намира резервоарът за пожарогасене. Той носи водата, която се хвърля над огъня. Клапанът на резервоарите за гасене се контролира от автора. Камерата, монтирана в горната част, е широкоъгълна камера. Цената на целия проект е около \$370 (666 лв.).



Фиг. 1.8 Блокова схема на проект “Fire Fighter Robot”

1.3 Нашето решение:

Нашето решение на проблема е да се изгради робот, като се вземе предвид мощността, която да захранва всичко - микроконтролер, двигателни драйвери и водна помпа. Роботът трябва да има огнеустойчиви материали, които също трябва да предлагат известно предимство в теглото, тъй като не трябва да бъде много тежък. Пример за такъв материал е алуминий, има висока издръжливост от удар или от падане. Задвижващият механизъм на робота трябва да има достатъчен въртящ момент и мощност. Освен това роботът може да има регулатор на скоростта, който да му помага за навигация и управление. Контролът на робота трябва да бъде улеснен и подпомогнат с добра сила на сигнала, за да се осигури подходящ обхват за използване от радио контролера.

В процеса на разработката и реализацията на дадения проект ще се правят непременно непрекъснати проучвания, за да се гарантира успеха на задачата.

Проектът ни “Противопожарен робот, управляван с помощта на радиоконтролер” би бил добър за пазара. Продуктът ще бъде достъпен както за ентузиасти, така и за обикновени любители. Неговата функционалност, макар и не толкова усъвършенствана, колкото на някои от конкурентите, би дошла на една много по-приемлива цена за малките и средните дружества или фирми, дори и за обикновените граждани, желаещи да го закупят. Решението се стреми да задоволи нуждите на точно тази част от обществото.

ГЛАВА 2. Блокова схема и концепция на проекта

Първоначалната идея на всяка една пожарна машина по света е да е в помощ на хората и да извършва задачи, които по своята същност са лесни, но пък от друга страна се повтарят многократно във времето. Оттук следва, че мисленето на хората трябва да се вземе под от съществено внимание при по-сложните действия и най-вече за вземането на решения. Подобна е и концепцията на нашия проект. Той цели да улесни работата в малки до средни предприятия и дружества, в които има подобен тип задачи. Моторите трябва да са здрави и бързи, за да могат достатъчно точно и мигновено да задвижват цялото тяло при команда. Бързината е ключова, защото стремежът е работата да се извърши възможно най-бързо.

Принципната блокова схема на проекта (фиг. 2.1) показва връзките, чието осъществяване е необходимо за работата на пожарната кола и радио контролера. За изпълнение на обикновени движения (напред, назад, наляво, надясно) проектът се нуждае от два постояннотокови мотора.

Системата се състои от три интегрирани сензора за пламък, които гарантират надеждно откриване на пламъци в различни направления. Тези сензори предоставят точна информация за настъпилия инцидент, като активират светодиодите за светлинни ефекти и пиезо-говорителя за звук. Всичко това предоставя ясно и бързо предупреждение за пожар и спомага за навременното приемане на мерки.

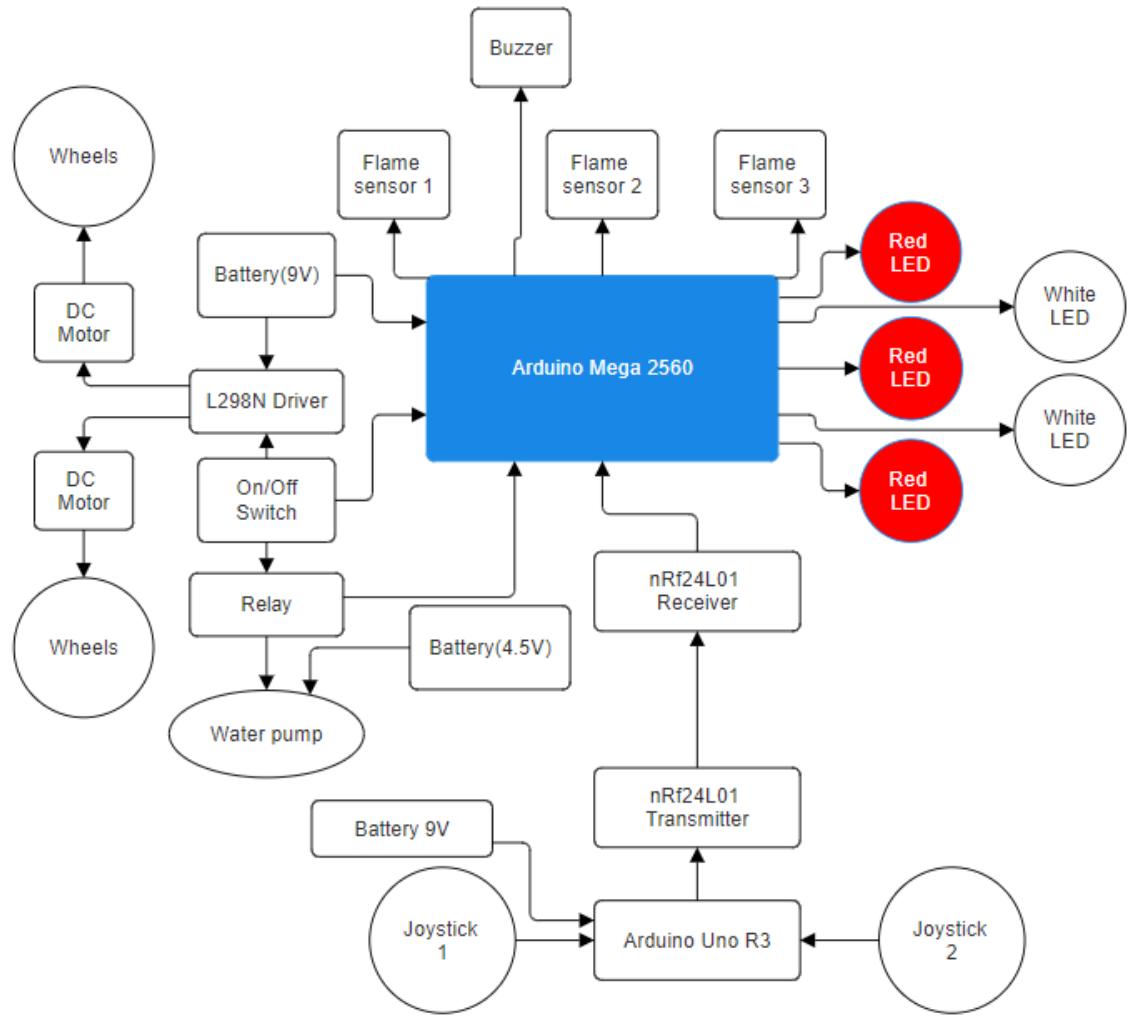
Захранването се осъществява или с помощта на батерия, или с някой друг източник, който да осигури достатъчно енергия за функционирането на пожарната кола.

Комуникационният модул nRF24L01 е интегриран в системата, предоставяйки надеждна безжична връзка между контролера и пожарната кола. Този модул позволява пренос на данни с висока стабилност и обхват, което е от съществено значение за управлението на автономни устройства.

За да може да се контролира лесно и от всеки, системата се състои и от радио контролер, осигуряващ бърз и ефективен безжичен достъп до устройството, което най-общо включва конфигурация и подаване на команди.

2.1 Блокова схема на проекта:

Принципната блокова схема на проекта (Фиг. 2.1) показва как на най-базово ниво изглежда свързаността на електрическите компоненти от проекта. Изградена е с помощта на уеб-базирания инструмент за диаграми SmartDraw.



Фиг. 2.1 Принципна блокова схема на проекта

2.2 Микроконтролер - Arduino:

Микроконтролерът е “мозъкът” на една система. Целта му е да преглежда данните от всички сензори и на тяхна база да взема решения за движението и управлението на гумите на тялото. Най-важните критерии, по които се избира микроконтролер, са: цена, наличност в Интернет, опит на отбора с него.

Arduino [6] е една от най-известните електронни платформи в света с отворен код, въз основа на хардуера и софтуера, които са лесни за използване. Електронните платки „Arduino“ могат да се закупят или готови, или като „направи си сам“ комплекти, като схемите им са свободно достъпни за всеки, който би искал да ги сглоби сам. Arduino е и страхотна търговска марка, която гарантира качеството и надеждността на продуктите, екипът от професионалисти от компанията задълбочено проверява новите платки, преди да бъдат пуснати в търговските мрежи. Неслучайно Arduino печели огромна популярност в областта на проектите, изградени на базата на електроника. Микроконтролерът в платката се програмира с

Arduino език за програмиране (базиран на Wiring и подобен на C++) в развойна среда Arduino (базирана на Processing).

2.2.1 Arduino Uno:

Това е една микроконтролерна платка, която е базирана на Microchip ATmega328P. Съдържа 6 аналогови пина, 14 цифрови входно/изходни пина, от които 6 могат да се използват като ШИМ, PWM (Pulse Width Modulation) изхода, 16 MHz кварцов резонатор, USB връзка, жак за захранване, ICSP конектор и бутон за нулиране. Няма много големи поводи за притеснение, ако се направи на нещо нередно при работа с Arduino Uno [7], в най-лошия случай може да се наложи да се замени чип за няколко долара и да започнем отначало работата си. Това е най-разпространеният микроконтролер за момента и оригиналната му версия струва 40 лв. Платката Uno е първата от серия USB Arduino платки и референтният модел за платформата Arduino. На Фиг. 2.2 можем да видим как изглежда една Arduino Uno платка:



Фиг. 2.2 Arduino Uno R3

2.2.2 Arduino Mega:

Arduino Mega 2560 [8] е микроконтролерна платка, базирана на ATmega2560. Има 54 цифрови входно/изходни пина (от които 15 могат да се използват като PWM изходи), 16 аналогови входа, 4 UART (хардуерни сериен порта), 16 MHz кристален осцилатор, USB връзка, жак за захранване, ICSP конектор, и Reset бутон. Съдържа всичко необходимо за поддръжка на микроконтролера; просто го свързваме към компютър с USB кабел или го захранваме с AC-към-DC адаптер или батерия. Цената му е 82.22 лв. Arduino Mega е дълъг 101.52mm, широк е 53.3mm, а теглото му е 37 g. На Фиг. 2.3 можем да видим как изглежда една Arduino Mega платка:



Фиг. 2.3 Arduino Mega 2560

2.3 Мотор:

Моторите са определящият компонент в проекта. От това какви мотори се изберат зависят до голяма степен бързината, функционалността и цената на пожарната кола. Критериите за избор на мотори са: цена, сила и тегло.

2.3.1 DC мотор:

Моторите (двигателите) преобразуват електрическата енергия в механична енергия. DC моторът (Фиг. 2.4) е електрически мотор, който работи с постоянен ток (DC). Във всеки електродвигател работата се основава на прост електромагнетизъм. Тоководещ проводник генерира магнитно поле; когато полученото нещо след това се постави във външно магнитно поле, то ще изпита сила, пропорционална на тока в проводника и на силата на външното магнитно поле. Както добре знаем, противоположните (северна и южна) полярности се привличат, докато подобни полярности (север и север, юг и юг) се отблъскват. Вътрешната конфигурация на DC мотор [9] е предназначена да използва магнитното взаимодействие между проводник с ток и външно магнитно поле за генериране на въртеливо движение. Двигателите с постоянен ток (DC) се използват широко за генериране на движение в редица продукти и проекти. DC моторите с постоянен магнит се радват на нарастваща популярност в приложения, изискващи компактен размер, висок въртящ момент, висока ефективност и ниска консумация на енергия. Цената на един DC мотор е около 25 лв.



Фиг. 2.4 DC мотор

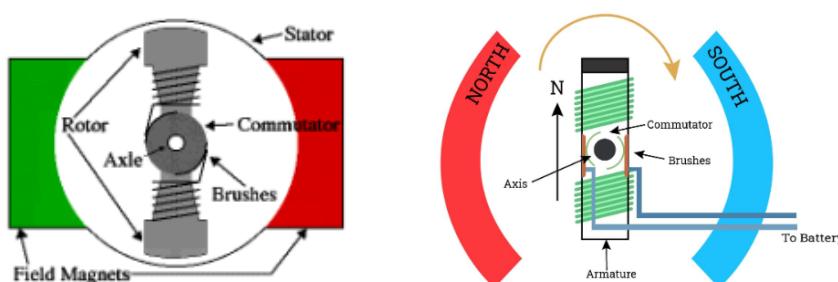
В мотор за постоянен ток с четка (brushed DC motor) четките осъществяват механичен контакт с набор от електрически контакти, осигурени на комутатор, закрепен към арматура, образувайки електрическа верига между източника на постоянен ток и намотките на бобината на арматурата (Фиг. 2.5). Като арматурата се върти на ос, неподвижните четки влизат в контакт с различни секции на въртящия се комутатор.

DC двигателите с постоянен магнит използват две или повече четки, контактуващи с комутатор, който осигурява постоянен ток към намотките на ротора, който осигурява желаното магнитно отблъскване/привличане с постоянните магнити, разположен около периферията на мотора.

Четките са конвенционално разположени в кутии за четки и използват U-образна форма пружина, която накланя четката в контакт с комутатора. Безчетковите постояннотокови двигатели с постоянен магнит се използват широко в различни приложения поради тяхната простота в дизайн, висока ефективност и нисък шум. Тези мотори работят чрез електронна комутация на намотките на статора, а не чрез конвенционалната механична комутация, постигната от притискащото зацепване на четките срещу въртящ се комутатор.

Безчетковият постояннотоков двигател основно се състои от вал, оборудван с ротор с един или повече постоянни магнити, разположени върху вала, и статор, който включва статорен компонент и фазови намотки. Образуват се въртящи се магнитни полета от токовете, приложени към намотките.

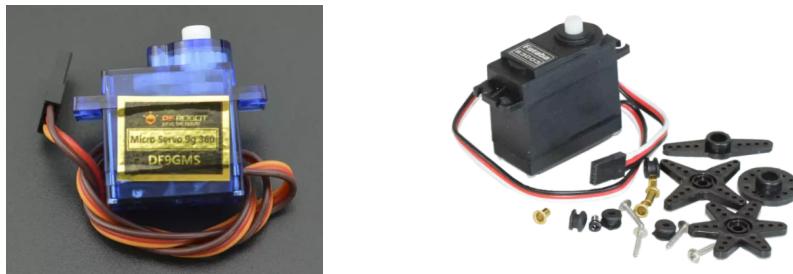
Роторът се състои от поне един постоянен магнит, заобиколен от статора, при което роторът се върти в статора. Два лагера са монтирани с аксиално разстояние един до друг на вала за поддържане на роторния възел и статорния възел един спрямо друг. За постигане на електронна комутация, безчеткови двигатели за постоянноен ток обикновено включват електронен контролер за управление на възбудждането на статора намотки.



Фиг. 2.5 Вътрешна архитектура на DC мотор

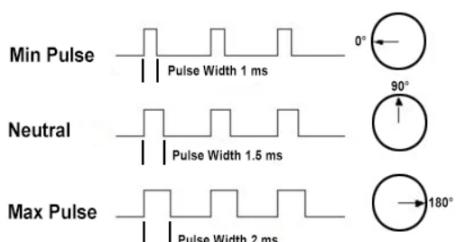
2.3.2 Servo мотор:

Серво моторът [10] е малък постояннотоков двигател с добавени следните компоненти: редуктор на зъбно колело, сензор за положение на вала на мотора и електронна верига, която управлява работата на мотора. Цената на един серво мотор може да варира, но като цяло една нормална и стандартна цена на серво мотор е между 10 и 15 лв. На Фиг. 2.6 можем да видим два примера за серво мотори:



Фиг. 2.6 Микро серво мотор (отляво)
и серво мотор (отдясно)

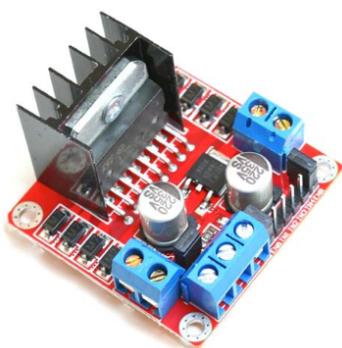
Редукторът на зъбното колело, предоставен в серво мотора, е голям; стандартното серво има 180:1 предавателно отношение. Това означава, че валът на мотора с постоянен ток трябва да направи 180 оборота, за да произведе 1 оборот на серво вала. Това голямо предавателно отношение намалява скоростта на сервото и пропорционално увеличава своя въртящ момент. Серво моторите обикновено се използват за ъглово позициониране, като например в самолети с радиоуправление. Те имат диапазон на движение от 0 до 180 градуса. Обикновено потенциометър измерва позицията на изходящия вал през цялото време, така че контролерът може точно да постави и поддържа позицията си. Сервото очаква да вижда импулс на всеки 20 милисекунди (0,02 секунди). Дължината на импулса ще определи колко върти мотора. Импулс от 1,5 милисекунди (Фиг. 2.7) ще накара мотора да се завърти на 90-градусова позиция (неутрална позиция). Ако импулсът е по-къс от 1,5 ms, тогава моторът ще завърти вала до по-близо до 0 градуса. Ако импулсът е по-дълъг от 1,5 ms, валът се завърта по-близо до 180 градуса. Количество мощност, приложена към мотора, е пропорционална на разстоянието, което трябва да измине. Така че, ако валът трябва да се завърти на голямо разстояние, моторът ще работи на пълна скорост. Ако трябва да се завърти само малко, моторът ще работи с по-ниска скорост.



Фиг. 2.7 Позиция на серво мотор за
управление на променлива ширина на импулса

2.4 Драйвер:

Драйверът за мотори е модул за двигатели, който ни позволява да контролираме работната скорост и посока на два мотора едновременно. Пример за двойно двупосочен моторен драйвер е L298N Dual H-Bridge Motor Driver [11], базиран на много популярната интегрална схема L298 Dual H-Bridge Motor Driver. Схемата ни позволява лесно и независимо да управлявате два постояннотокови мотора с напрежение от 7 до 12V с консумация до 2A на всеки в двете посоки. Той е идеален за роботизирани приложения и много подходящ за свързване към микроконтролер, изискващ само няколко контролни линии на мотор. Може също така да се свързва с прости ръчни превключватели, TTL логически портове, релета и др. Тази платка е оборудвана с LED индикатори за захранване, вграден +5V регулятор и защитни диоди. L298N (Фиг. 2.8) е проектиран да работи с входно напрежение от 3.2V до 40V постоянен ток. Той предоставя захранванващо напрежение в диапазона от 5V до 35V DC. Максималната консумация за логиката е от 0 до 36mA. Модулът има възможност за контрол на входното напрежение за управление и активация в два обхвата: ниско и високо. За управление, ниският обхват е между -0.3V и 1.5V, а високият обхват е между 2.3V и Vss. За активация, ниският обхват е от -0.3V до 1.5V (когато управляващият сигнал е невалиден), а високият обхват е от 2.3V до Vss (когато управляващият сигнал е активен). Максималната консумация на енергия при температура $T = 75^{\circ}\text{C}$ е 20W, а температурният обхват за съхранение варира от -25°C до $+130^{\circ}\text{C}$. Модулът предлага регулиран изход от +5V, предназначен за захранване на контролерни платки като Arduino. Физическите размери на модула са 3.4cm x 4.3cm x 2.7cm. Цената му е около 10 лв.

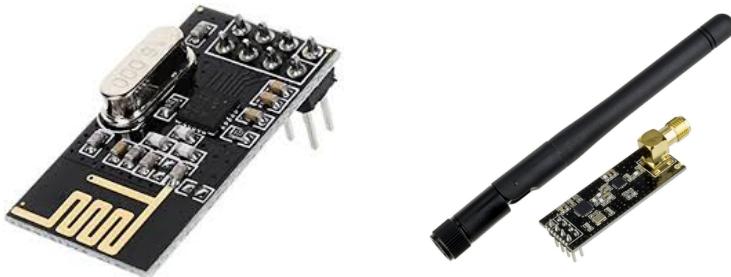


Фиг. 2.8 L298N драйвер за мотори

2.5 Безжичен RF модул:

В съвременната комуникация безжичното предаване на данни е за предпочитане от страна на много компании и организации. nRF24L01 [12] е един такъв безжичен приемо-предавателен модул, което означава, че всеки модул може

както да изпраща, така и да получава данни. Те работят на честота от 2,4 GHz. Модулите, когато се управляват ефективно, могат да покрият разстояние от 100 метра, което го прави чудесен избор за всички проекти с безжично дистанционно управление. Модулът работи на 3,3 V. nRF24L01 [13] работи с помощта на SPI комуникация. Той съдържа в себе си 8 пина: 1) GND (Земя); 2) Vcc (Захранване) – захранване на модула с 3.3V; 3) CE (Chip Enable) – използва се за активиране на SPI комуникация; 4) CSN (Chip Select Not) – този пин трябва винаги да се поддържа на високо ниво, в противен случай ще деактивира SPI; 5) SCK (Serial clock) – осигурява тактовия импулс, чрез който работи SPI комуникацията; 6) MOSI (Master Out Slave In) – свързан към MOSI пина на MCU, за да може модулът да получава данни от MCU; 7) MISO (Master In Slave Out) – свързан към MISO пин на MCU, за да може модулът да изпраща данни от MCU; 8) IRQ (Interrupt) – това е активен нисък пин и се използва само ако е необходимо прекъсване. Цената на един nRF24L01 приемник е 4.50 лв., а на предавател - 11 лв. На Фиг. 2.9 е представена снимка на nRF24L01 модул:



Фиг. 2.9 nRF24L01 модул

2.6 Сензор за пламък:

Сензорът за пламък [14] се състои от IR приемник, резистор, кондензатор, потенциометър и LM393 компаратор в интегрална схема. Модулът е изграден на базата на инфрачервен фотодиод. Може да открива инфрачервена светлина с дължина на вълната, варираща от 700nm до 1000nm. Чувствителността се регулира чрез променлив резистор с ъгъл на откриване 60 градуса. Работното напрежение е между 3.3V и 5.3V DC, с цифров изход за индикация на присъствие на сигнал. Отчитането се определя от LM393 компаратор. Платката е с размери 27mm x 15,5mm, с 2 монтажни отвора и 4-пинов интерфейсен конектор, с разстояние между изводите 2,54mm. Платката (Фиг. 2.10) разполага с четири изводна рейка. Към два от тези изводи се подава захранване, а на другите два се подава аналогов или цифров изходен сигнал. Цифровият се подава чрез логически нива, а аналоговият - под формата на напрежение, което се променя с приближаване или отдалечаване на пламъка. Цената за този сензор е 3 лв.



Фиг. 2.10 Сензор за пламък

2.7 Водна помпа:

Водната помпа [15] (Фиг. 2.11) има за цел да предостави автоматизиран механизъм за управление на водоснабдяването, подходящ за гасене на пожари, където точното и програмируемо подаване на вода е от съществено значение. Тя съчетава гъвкавост, ефективност и възможност за персонализация в зависимост от конкретните нужди на приложението. Използвайки водна помпа, работеща на 5V (5 лв.) или 12V (20 лв.), тя може да бъде свързана с Arduino с помощта на реле или на драйвер.



Фиг. 2.11 Водни помпи, работещи на 5V (отляво)
и на 12V (отдясно)

2.8 Зумер:

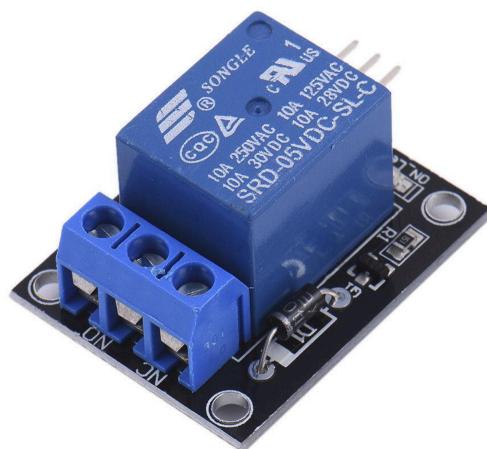
Зумерът (buzzer) [16] е електронен компонент, който генерира звуков сигнал, обикновено честотен и краткотраен. Този сигнал се използва за предоставяне на аудио аларми или звукови сигнали в различни електронни устройства. Обикновено зумерите (Фиг. 2.12) са изградени от пиезоелектричен кристал, който може да вибрира, когато му се приложи електрическо напрежение. Този вибриращ елемент създава аудиовълни във въздуха и генерира характерен звук. Цената му е 4 лв.



Фиг. 2.12 Зумер

2.9 Реле:

Електрически управляван превключвател, като реле, се използва за включване/изключване на товар, като позволява протичането на ток през него. Това реле просто се управлява от ниско напрежение (5V), което се генерира от изводите на Arduino. Така че управлението на реле модул с платката Arduino е много просто. Обикновено релетата са много полезни, когато искаме да управляваме електрическа верига със сигнал с ниска мощност. Има различни видове релета, използвани в различни приложения. Ще бъде разгледан едноканален реле модул [26] за Arduino: Размерите му са: дължина - 34mm; ширина - 26mm; височина - 18mm. Теглото му е 14g и се захранва с 5V, което е подходящо за използване с Arduino. На Фиг. 2.13 е представена снимка на едноканален реле модул:



Фиг. 2.13 Едноканален реле модул

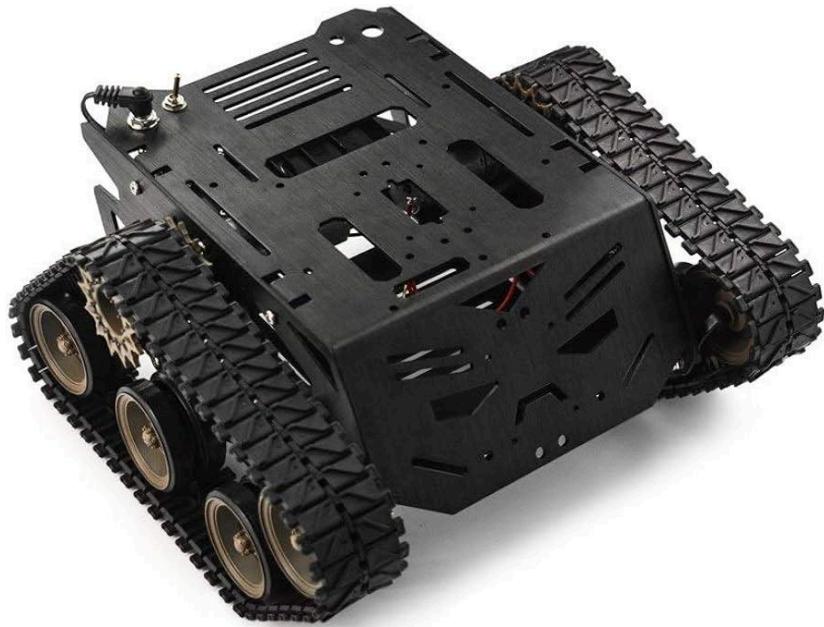
Външността на релето е превключвател, който се управлява електрически чрез електромагнит. Този електромагнит просто се задейства чрез ниско напрежение като 5V от микроконтролер и издърпва релеен контакт, за да свързва или изключва верига, базирана на високо напрежение. Релето, представено на фигурата включва различни изводи, които са:

Извод 1 - сигнален извод (задействане на реле): този входен извод се използва за активиране на релето; Извод 2 (земя): Това е извод за земя; Извод 3 (VCC): Този входен захранващ извод се използва за захранване на бобината на релето; Извод 4 (нормално отворен): Това е NO (нормално отворен) терминал на релето; Извод 5 (общ): Това е общият терминал на релето; Извод 6 (нормално затворен): Това е нормално затворен (NC) извод на релето.

Релейните модули могат да се справят с товари до 10 ампера. Те са идеални за различни устройства като пасивни инфрачервени детектори и други сензори. Тези модули се използват с Arduino и други микроконтролери.

2.10 Тяло на пожарната машина:

Devastator Tank платформата за мобилен робот (Devastator Tank Mobile Robot Platform) [17] е изработена от високоякостна алуминиева сплав, което я прави изключително здрава и издръжлива. Танкът е напълно съвместим с популярните микроконтролери на пазара, например Arduino, Raspberry Pi, Lattepanda и т.н. Високоскоростните двигатели и веригите с първокласно качество също му позволяват да се движи бързо по повърхности и високопроизводителното му окачване и има изключителна мобилност дори през най-трудните терени. Самата платформа на робота (Фиг. 2.14) има множество монтажни отвори, които позволяват на потребителите да добавят различни сензори, серво мотори и микроконтролери (напр. Romeo All-in-one, Raspberry Pi Model B+ и др.) Този комплект е идеален за любители, преподаватели, състезания с роботи и изследователски проекти. Ето и някои спецификации: работното му напрежение е 3 - 8V DC, диаметър на задвижващото колело: 43mm; максимална скорост - 36cm/s; тегло - 780g. Размерите на платформата са: дължина - 225mm; ширина - 220mm; височина - 108mm. Цената на избора ни за тяло на пожарната кола е 250 лв.



Фиг. 2.14 Devastator Tank Mobile Robot Platform

2.11 Съд с вода и маркуч:

Съдът с вода, или водният резервоар, служи за съхранение и транспортиране на вода, предназначена за бързо и ефективно гасене на пожари. Това е ключов компонент за системата на проекта, който доставя вода до различни огнегасителни системи, с помощта на водната помпа, която се намира в него.

Маркучът в проекта също е важен компонент за доставка на вода към мястото на пожара. Той се свързва с водната помпа, която се намира в съда, транспортиращ вода, и може да бъде управляван в коя посока да бъде завъртян, за да изгаси цялостно и ефективно възникналия пожар.

Цената на един обикновен съд за вода е 1 лв., а на маркуча - 2 лв. На Фиг. 2.15 можем да видим как изглеждат примерен съд за вода и маркуч, които могат да се използват в такъв контекст:



Фиг. 2.15 Съд за вода и маркуч

2.12 Джойстик:

Много роботизирани проекти се нуждаят от джойстик. Модулът KY-023 [18] предлага едно достъпно решение към това. Джойстик модулът е подобен на аналоговите джойстици, намиращи се в геймпадовете. Цената му е 4.50 лв. Създаването му се състои в монтиране на два потенциометъра под ъгъл от 90 градуса. Преместването на джойстика ще доведе до промяна на изхода от 0V до 5V, в зависимост от посоката му. Когато движим джойстика, стойностите се променят от 0 до 1023, в зависимост от позицията му. Модулът се състои от 5 пина: 1) GND: земя; 2) +5V: 5V DC; 3) VRX: напрежение, пропорционално на позиция x; 4) VRY: напрежение, пропорционално на позиция y; 5) SW: бутон за превключване. На Фиг. 2.16 можем да видим как изглежда описания джойстик модул KY-023:



Фиг. 2.16 KY-023 джойстик модул

2.13 Захранване:

Захранването на проекта може да се осъществи по два начина: чрез AA батерии, които се препоръчват от каталожните данни за Devastator Tank платформата за мобилен робот, или чрез LiPo батерия.

2.13.1 AA батерия:

AA е най-разпространеният тип захранващи галванични елементи и акумулатори. Номиналното напрежение е 1,5 V при галваничните елементи, 1,2 V – при Ni-Cd и Ni-MH акумулатори, 1,6 V при Ni-Zn акумулатори. Елементът AA [19] представлява цилиндър с диаметър 13,5 – 14,5mm. Дължината на елемента заедно с контактния накрайник на положителния полюс е 50,5mm. Цилиндричната част е покрита с изолираща обвивка. Изводите са разположени на противоположните краища на цилиндъра. Цената на 4 броя алкални AA батерии от марка Duracell е 6 лв. Предимствата на AA батериите включват:

- 1)** AA батерии са универсални и съвместими с широка гама от устройства. Почти всеки дом и офис разполага с устройства, които използват този тип батерии.
- 2)** AA батерии се предлагат повсеместно в магазините и са лесни за намиране. Те са идеални за преносими устройства.

Недостатъците включват:

- 1)** В сравнение с по-големите батерии, AA батерии обикновено имат по-ограничен живот, особено при интензивно използване.
- 2)** AA батерии обикновено не са презареждаеми.

2.13.2 LiPo батерия:

Литиево-полимерните (LiPo) батерии [20] са вид акумулаторна батерия, която има редица предимства и недостатъци в сравнение с други видове батерии. Цената на една 7.4V/1800mAh LiPo батерия е 35 лв. Ето някои от основните предимства и недостатъци на използването на LiPo батерии в проект с Arduino:

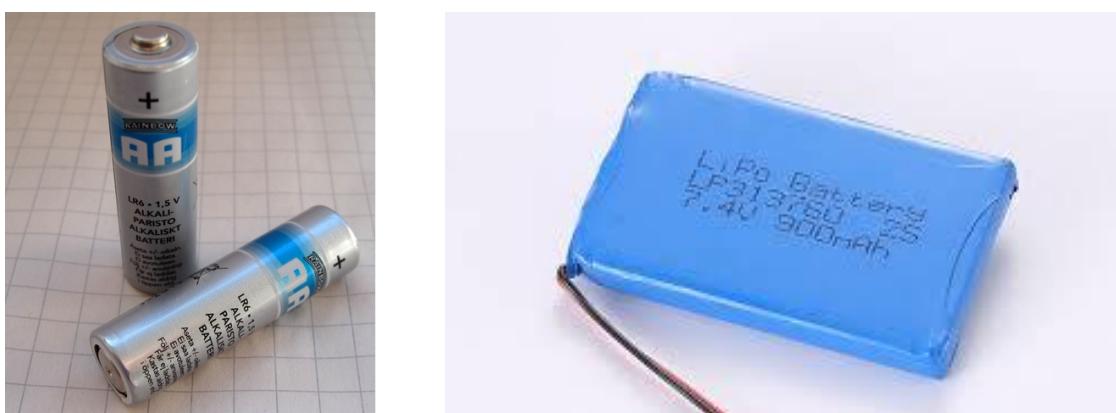
- 1)** LiPo батерии имат висока енергийна плътност, което означава, че могат да съхраняват много енергия в малка опаковка. Това ги прави идеални за използване в преносими устройства и проекти, където пространството е на първо място;
- 2)** LiPo батерии имат сравнително ниска степен на саморазреждане, което означава, че могат да задържат заряда си дълго време, когато не се използват. Това ги прави удобни за проекти, които изискват дълги периоди на бездействие;
- 3)** LiPo батерии са относително лесни за зареждане и разреждане, което ги прави удобни за използване. Те могат да се зареждат с обикновено зарядно устройство за батерии, а напрежението им може лесно да се следи с помощта на сензор за напрежение.

Недостатъците на LiPo батериите включват:

1) LiPo батерии могат да бъдат опасни, ако не се боравят правилно. Те могат да се запалят или експлодират, ако са презаредени, късо съединение или пробити. Важно е да се използва подходящо зарядно устройство за батерии и да се работи внимателно с LiPo батерии;

2) LiPo батерии могат да бъдат по-скъпи от другите видове батерии. Това може да ги направи по-малко привлекателни за проекти, при които цената е основно съображение;

На Фиг. 2.17 можем да видим как изглеждат AA и LiPo батерии:



Фиг. 2.17 AA батерия и LiPo батерия

2.14 Други електронни елементи:

2.14.1 Резистор:

Един от основните градивни елементи в съвременната електроника и електротехниката е резисторът. От гледна точка на електричеството материалите се делят на две категории: проводници и изолатори. През проводниците може да протича електрически ток, а през изолаторите – не. Резисторът [21] е основен електронен елемент, с точно определена, постоянна стойност на електрическото съпротивление. Номиналното съпротивление се измерва в омове (Ω). Резисторите са пасивни елементи, което означава, че не могат да произвеждат енергия, а само да я консумират. Цената на един резистор е около 30 - 50 ст. Ето как изглежда един 10K резистор, Фиг. 2.18:



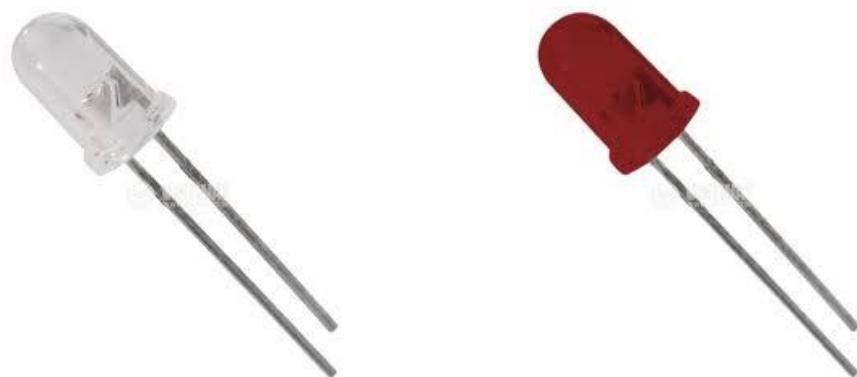
Фиг. 2.18 10K резистор

2.14.2 Светодиод:

Светодиодите [22] са от типа диоди, които превръщат електрическата енергия в светлина. Накратко, светодиодите са като малки електрически крушки, които изискват много по-малко мощност, за да светят. Също така са по-енергийно ефективни, така че не са склонни да се нагряват, както правят обикновените крушки. Това ги прави идеални за устройства с ниска мощност. Светодиодите, също както и обикновените изправителни диоди, пропускат ток само в едната посока и когато няма ток, няма светлина. Това показва, че светодиодът не може да бъде повреден ако объркаме поляритета, просто няма да свети.

Положителната страна на светодиода се нарича „анод“, а отрицателната - „катод“. Токът тече от анода към катода и никога в обратна посока. От това колко ярко ще свети един светодиод зависи тока, който минава през него. Ако свържем светодиод директно към източник на ток е възможно да дефектира. Ето защо е важно да се ограничи количеството ток, преминаващо през диода. Затова използваме последователно свързан резистор, който да ограничава тока, който преминава през светодиода. Цената на един светодиод е между 35 - 80 ст.

На Фиг. 2.19 можем да видим два светодиода - бял и червен.

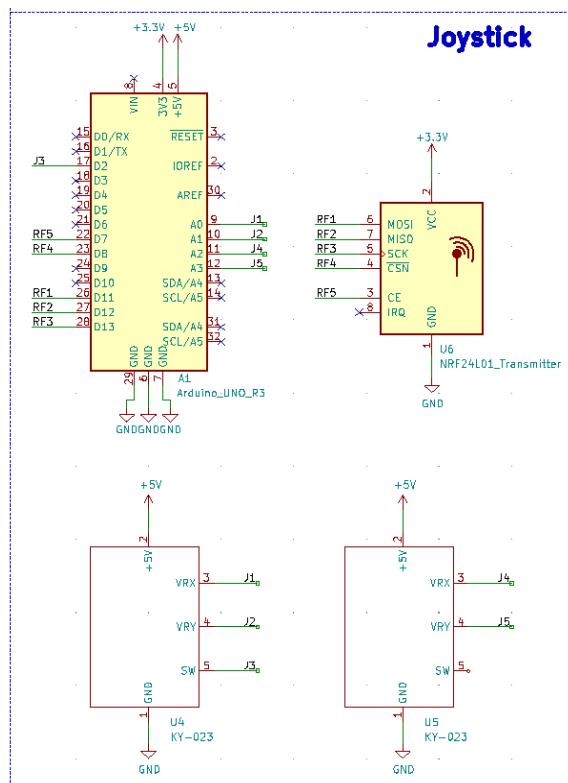


Фиг. 2.19 Бял и червен светодиод

ГЛАВА 3. Принципна електрическа схема

Електрическите схеми на радиоконтролера за управление на пожарната машина и на самия робот са проектирани с помощта на KiCad. Това е безплатен софтуер за автоматизация в проектирането на електронния дизайн. Улеснява проектирането и симулацията на електронен хардуер. Нека видим как изглеждат двете електрически схеми на проекта:

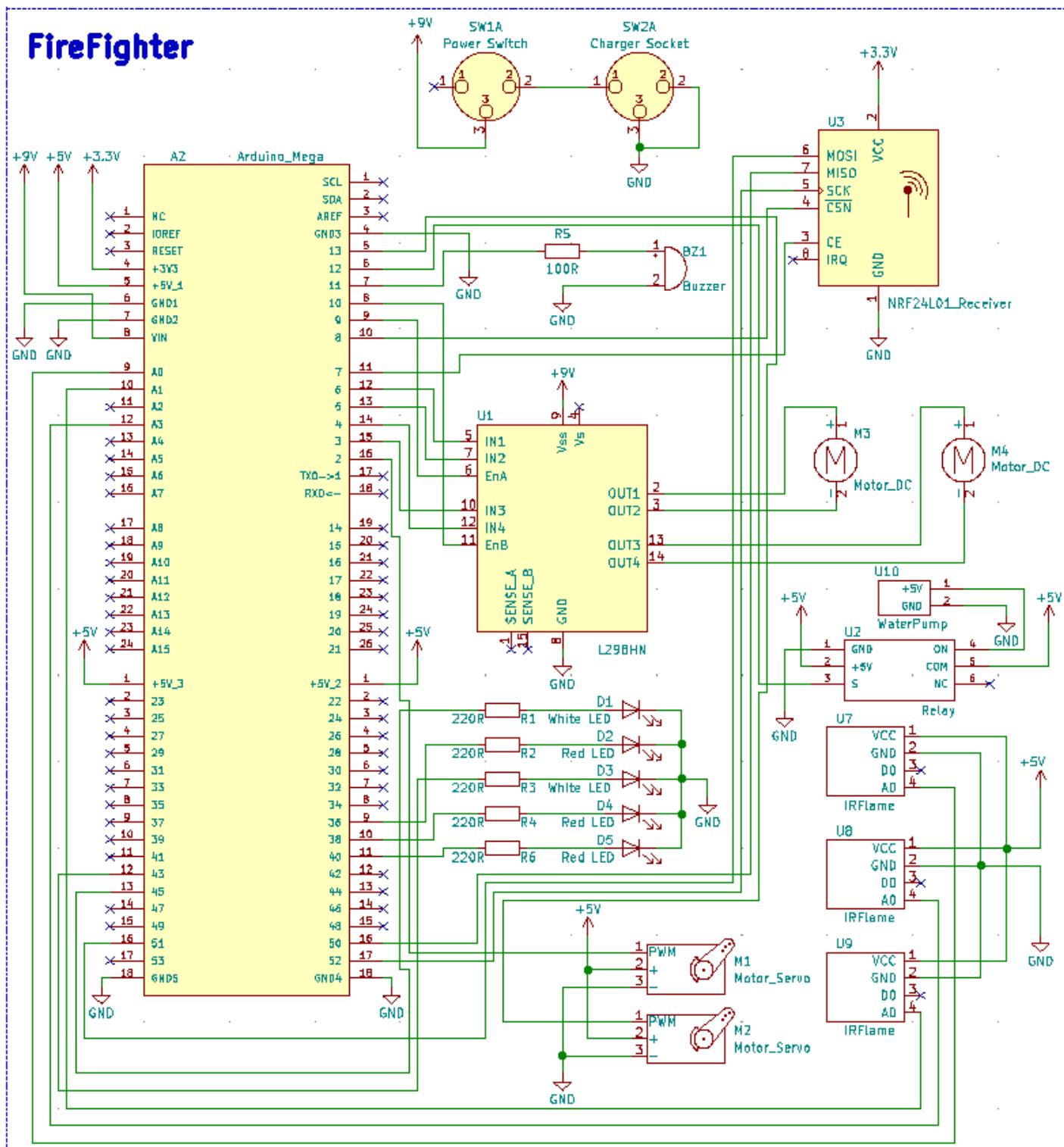
3.1 Електрическа схема на радиоконтролера за управление:



За реализирането на тази част от проекта се използват Arduino Uno, nRF24L01 безжичен модул и два джойстика KY-023. Двата джойстика имат следните функции: 1) При натискане на първия джойстик водата започва да тече от съда през маркуча. След това можем отново да натиснем бутона, за да спрем този процес. Възможно е и да движим джойстика, който има функциите: Forward (напред), Backward (назад), Left (наляво) и Right (надясно). Захранва се с 5V от микроконтролера. Когато движим джойстика, стойностите се променят от 0 до 1023, в зависимост от позицията му, поради което VRX и VRY пиновете на джойстика са свързани към аналогови пинове на ардуиното. SW пинът отговаря за това дали е натиснат джойстикът, или не, т.е. има две състояния, поради което е свързан към цифров пин на ардуиното; 2) Вторият джойстик има функциите: Up (нагоре), Down (надолу), Turn Left (завъртане наляво) и Turn Right (Завъртане надясно). Свързването на пиновете съответства на правилата, казани за първия джойстик. След това имаме Arduino Uno, за което са свързани джойстиците и безжичният модул nRF24L01. Последният елемент от електрическата схема е единият nRF24L01 модул [12][13] от проекта. Той използва SPI интерфейс и се управлява с помощта на команди, изпращани чрез този интерфейс. За използване на модула, изводите му са свързани към микроконтролера, за да се инициализира SPI интерфейса и да се изпратят нужните команди. За да се предават данни с модула, трябва да се настройт двата модула - един като предавател и един като получател. В случая nRF24L01 модулът е настроен като предавател. Той изпраща данни на определен адрес, който получателят е настроен да слуша.

или не, т.е. има две състояния, поради което е свързан към цифров пин на ардуиното; 2) Вторият джойстик има функциите: Up (нагоре), Down (надолу), Turn Left (завъртане наляво) и Turn Right (Завъртане надясно). Свързването на пиновете съответства на правилата, казани за първия джойстик. След това имаме Arduino Uno, за което са свързани джойстиците и безжичният модул nRF24L01. Последният елемент от електрическата схема е единият nRF24L01 модул [12][13] от проекта. Той използва SPI интерфейс и се управлява с помощта на команди, изпращани чрез този интерфейс. За използване на модула, изводите му са свързани към микроконтролера, за да се инициализира SPI интерфейса и да се изпратят нужните команди. За да се предават данни с модула, трябва да се настройт двата модула - един като предавател и един като получател. В случая nRF24L01 модулът е настроен като предавател. Той изпраща данни на определен адрес, който получателят е настроен да слуша.

3.2 Електрическа схема на пожарната машина:

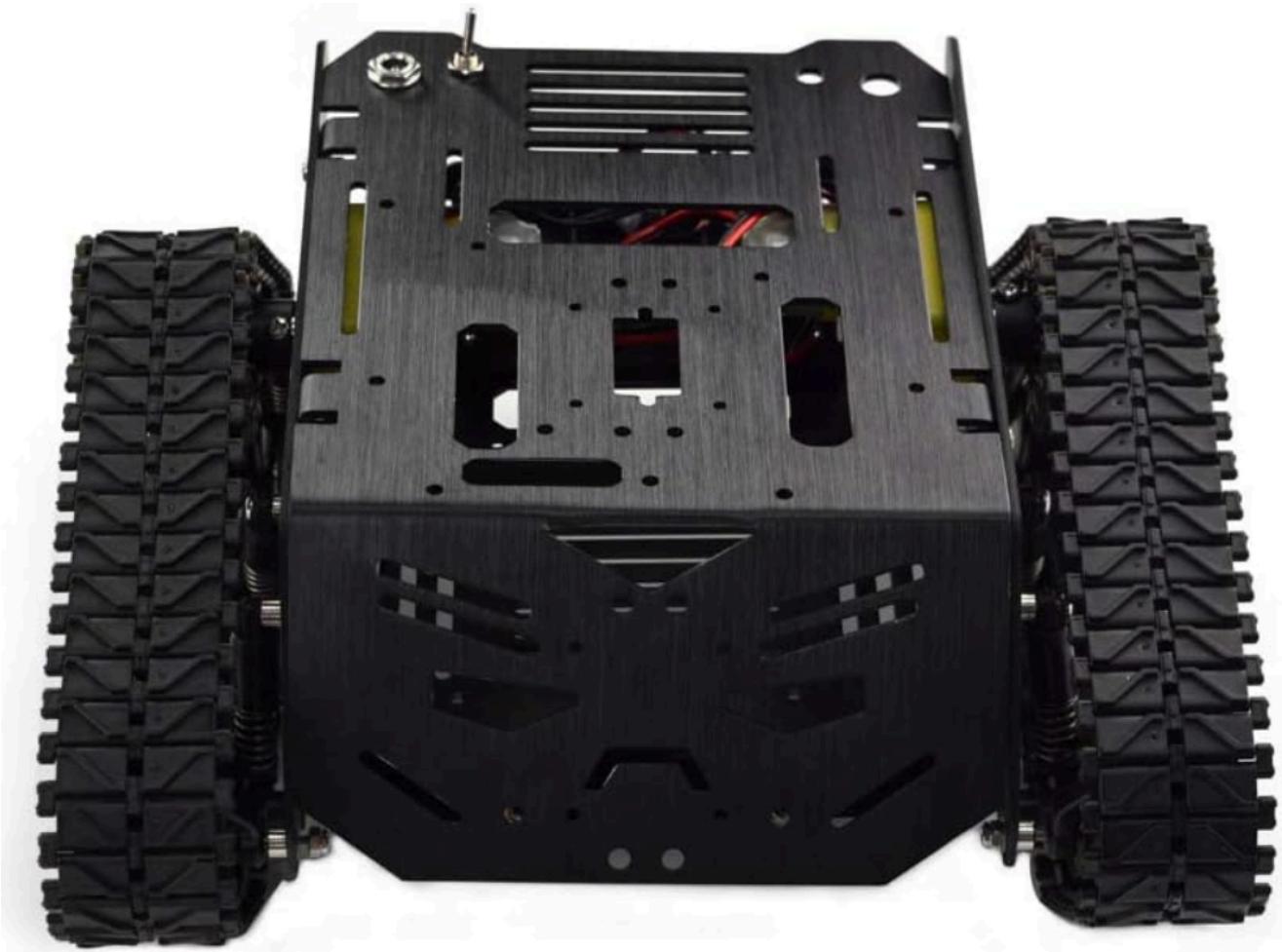


ограничават тока, който преминава през светодиодите. Превключвателят има за цел да пуска и да спира цялата система, докато гнездото за зарядно устройство помага за по-лесното подаване на захранване на цялата пожарна машина. Второто гнездо за батерии служи за захранване на водната помпа, работеща на 5V. NRF24L01 безжичният модул от тази част на проекта е настроен като приемник. Той използва SPI интерфейс и се управлява с помощта на команди, изпращани чрез този интерфейс. Той получава данни на определен адрес, на който е настроен да слуша. Чрез двата DC мотора, работещи на 9-12V, се задвижва тялото на пожарната машина с определена скорост. Използван е H-bridge (L298N), който контролира посоката на въртене и на двата DC мотора (свързани към задвижващи колела). В допълнение, едно реле (KY-019) служи за включване/изключване на водната помпа. Главният мозък на цялата пожарна машина е микроконтролерът Arduino Mega. В сравнение с други видове Arduino (Uno, Nano, Leonardo и т.н.), Arduino Mega се отличава с по-голям брой пинове, в случай на интерес за разширяване на дизайна и добавяне на още функции на даден проект, както е и в нашия случай. Използват се общо три сензора за пламък. Те са свързани към аналогови пинове на главния микроконтролер. Функцията им е да засекат пламъци и да изпратят команда за спиране на автомобила на подходящо разстояние от пламъка/огъня, така че в същия момент да се стартира и водната помпа, която да потуши огъня. Когато сензорите засекат информация, тогава те изпращат сигнал чрез зумер, който издава писклив звук и известява потребителя за наличие на пламък или на пожар. За целия механизъм на водния пистолет се използват един серво и един микро серво мотор. Тези мотори могат да осигурят относително прецизно движение в определен диапазон, както и консумират малко енергия. Серво моторът служи за задвижване на водния пистолет наляво или надясно, докато микро серво моторът мести маркуча с вода нагоре или надолу.

ГЛАВА 4. Проектиране и изграждане на хардуера

4.1 Реализация на корпуса на робота на проекта:

Корпусът на проекта представлява Devastator Tank платформа за мобилен робот (Devastator Tank Mobile Robot Platform), изработен от високоякостна алуминиева сплав, която го прави изключително здрав и издръжлив. Високоскоростните мотори и веригите с първокласно качество също му позволяват да се движи бързо по повърхности и има изключителна мобилност дори през най-трудните терени. Самата платформа на робота (Фиг. 4.1) има множество монтажни отвори, които позволяват добавяне на различни сензори, серво мотори и други компоненти или микроконтролери. Пълните инструкции за сглобяването на платформата могат да бъдат намерени в “Използвана литература”, секция [25].



Фиг. 4.1 Платформа на робота

На Фиг. 4.2 - 4.5 са представени снимки от процеса на сглобяване на основата на корпуса на проекта:



Фиг. 4.2 Сглобяване на механизма за амортизори на робота



Фиг. 4.3 и 4.4 Сглобени части на механизма за амортизори на робота и поставени колела за задвижване на проекта



Фиг. 4.5 Съставни части и изглед на напълно сглобения робот

Трябва да се отбележи, че така изглежда прототипът на робота (без електрониката по него). Тук предстои развитие на проекта с добавяне на различни елементи и компоненти по него, което означава и нов дизайн на корпуса.

4.2 Реализация на корпуса на контролера на проекта:

Контролерът за управление на робота е “сърцето” на устройството, което координира и контролира всички функции и действия на робота. Той се състои от модул за комуникация със сензорите и моторите на робота, като осигурява възможност за извличане на данни от околната среда и предаване на команди за управление и действие. Контролерът е изработен от дърво и е облечен със самозалепващо се фолио, за да придобие по-красив вид.

На Фиг. 4.6 - 4.11 могат да се видят снимки от процеса на изработката му.



Фиг. 4.6 Процес на изработка на контролера за управление на робота

За изработката на контролера са използвани различни триони, винтове, пили, както и дрелка със свредла.



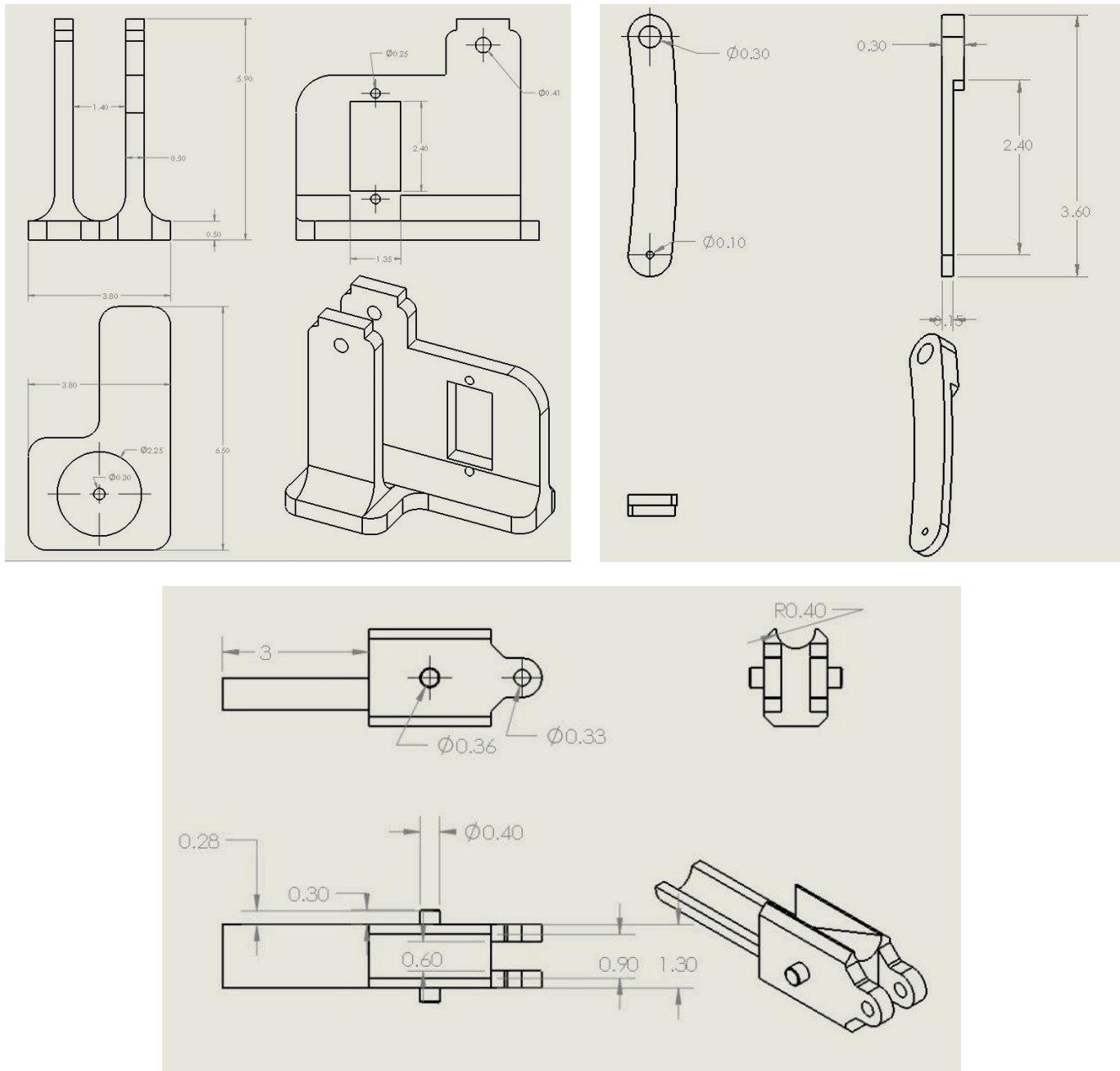
Фиг. 4.7 и 4.8 Сглобен (вляво) и боядисан (вдясно) контролер за управление на робота



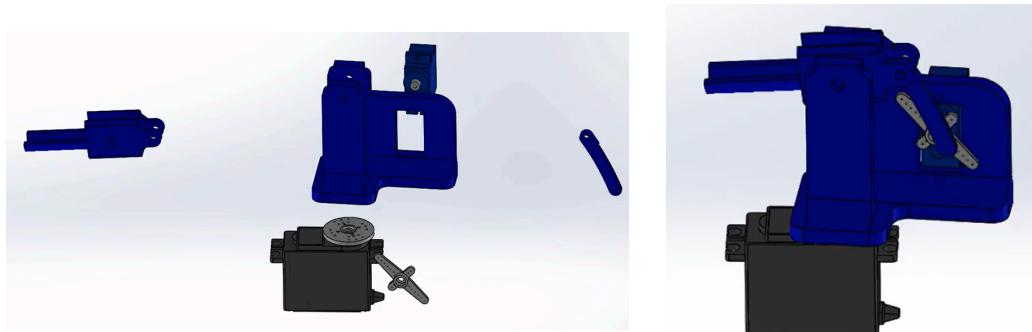
Фиг. 4.8 и 4.9 Облепен (вляво) контролер и изгледът му отвътре (вдясно)

4.3 Реализация на механизма за въртене на маркуча:

Механизмът за управление на въртене на маркуча служи за ефективното и цялостно гасене на огън и борба с пожари, което се осъществява с помощта на два серво мотора. Механизмът позволява бързо и точно регулиране на посоката, в която се изхвърля водата от помпата. На Фиг. 4.10 и 4.11 са приложени снимки на частите и размерите им за реализация на механизма за управление на маркуча на водната помпа от робота.



Фиг. 4.10 Размери на 3D частите за механизма за въртене на маркуча



Фиг. 4.11 Изглед на монтиране на 3D частите за механизма за въртене на маркуча

4.4 Равносметка и себестойност на проекта:

До началото на март 2024 разработката на този курсов проект е коствало на отбора \approx 490 лв. и общо около 100 - 120 часа работа. Към 14 април 2024 г., средствата за осъществяването на прототипа на робота възлизат на около \approx 550 лв., а часовете, прекарани в изработването, са около 150, като вече може да се каже, че проектът е във финалната си фаза на функционалност.

ГЛАВА 5. Логика на проекта и свързване на отделните хардуерни компоненти

5.1 Изграждане на контролера за управление на робота:

След оформянето и изработката на основата на контролера, който служи за управление на пожарникарския робот, в тази глава се разглежда как се монтират останалите компоненти върху него. В момента той изглежда така (Фиг. 5.1):



Фиг. 5.1 Контролер без електроника

Следват да бъдат монтирани електронните елементи върху него. За “мозък” на системата е избран микроконтролер Arduino Uno, поради размерите му и функционалността му, която е повече от достатъчна в случая. Към него са свързани два KY-023 джойстик модула. На Фиг. 5.2 може да се види снимка от процеса на монтажа им:



Фиг. 5.2 Монтаж на джойстиците върху контролера

Двата джойстика имат следните функции:

1) Първият джойстик служи за управление на робота. Възможно е да го движим, както следва: Forward (напред), Backward (назад), Left (наляво) и Right (надясно). Захранва се с 5V от микроконтролера. Когато движим джойстика, стойностите се променят от 0 до 1023, в зависимост от позицията му, поради което VRX и VRY пиновете на джойстика са свързани към аналогови пинове на ардуиното. При първия джойстик SW пинът, който отговаря за това дали е натиснат джойстикът, или не, се използва, за да се пусне/спре водната помпа, която да потуши огъня и пламъците.

2) Вторият джойстик има функциите: Up (нагоре), Down (надолу), Turn Left (завъртане наляво) и Turn Right (Завъртане надясно). Отново се захранва с 5V от Arduino Uno. Когато движим джойстика, стойностите се променят от 0 до 1023, в зависимост от позицията му. При втория джойстик SW пинът, който отговаря за това дали е натиснат джойстикът, или не, не се използва. Като цяло, вторият джойстик е отговорен за механизма на маркуча, състоящ се от два серво мотора, който подпомага за ефективното и пълноценно прекратяване на горенето.

Другият компонент, свързан към Arduino Uno, е безжичният модул nRF24L01. Той използва SPI интерфейс и се управлява с помощта на команди, изпращани чрез този интерфейс. За използване на модула, пиновете му са свързани към микроконтролера, за да се инициализира SPI комуникация и да се изпратят нужните команди и данни. В случая nRF24L01 модулът е настроен като предавател. Той изпраща данни на определен адрес, който получателят е настроен да слуша.

Не на последно място, захранването на контролера се осъществява чрез 9V батерия, свързана към захранващия жак (Power Jack) на Arduino Uno.

На Фиг. 5.3 може да се види как изглежда целият монтиран контролер:



Фиг. 5.3 Монтиран контролер за управление на робота

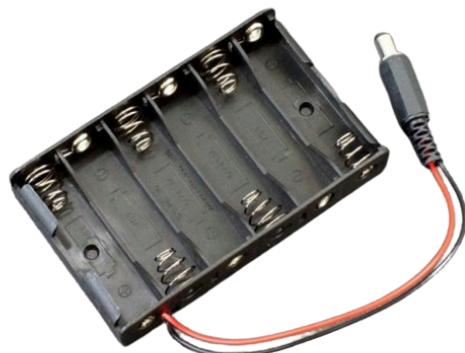
5.2 Изграждане на пожарникарския робот:

На Фиг. 5.4 е представен корпусът на пожарникарския робот, на който предстои да бъде имплементирана електроника:



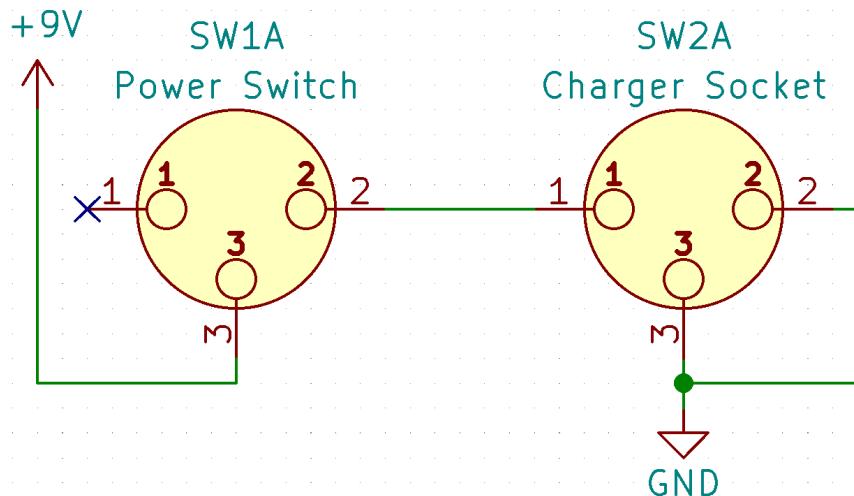
Фиг. 5.4 Корпус на робота без електроника

Първото нещо, което трябва да се реши, е по какъв начин да се захранва пожарникарският робот. Най-удобният вариант е чрез батерии. Избрано е да се захранва с 6 броя AA батерии, защото те осигуряват 9V захранване (всяка една от произвежда 1.5V напрежение), което е достатъчно за изискванията ни. Друга причина е, че повече от 6 броя батерии не биха били се събрали във вътрешността на робота. На Фиг. 5.5 е представен държачът за батериите:



Фиг. 5.5 Държач за 6 AA батерии

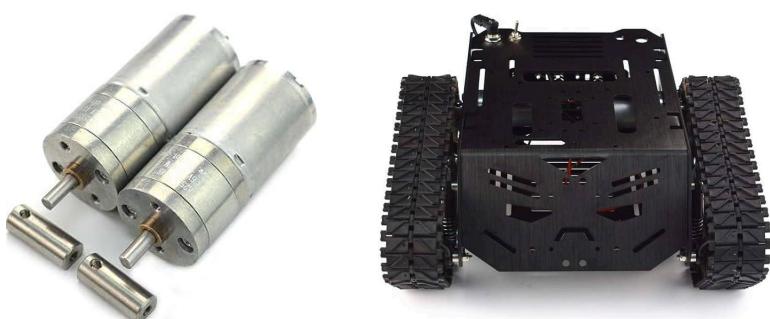
Адаптерът на държача за батерии се свързва към специално гнездо за него, монтирано на робота. То, от своя страна, се свързва към ключ за пускане и спиране на целия робот. На Фиг. 5.6 може да се види как се осъществява свързването между двата компонента:



Фиг. 5.6 Свързаност между гнездото за адаптер и ключът за пускане/спиране на пожарникарския робот

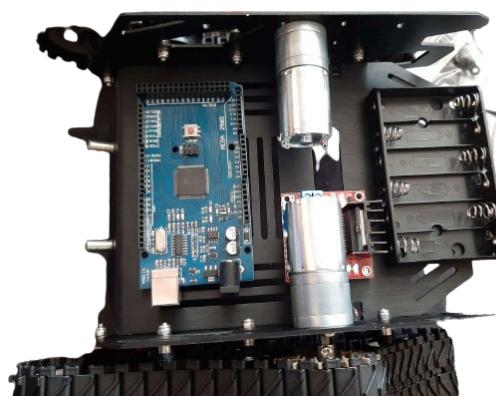
След като е осигурено захранване, трябва да се помисли за микроконтролер, който съдържа много пинове в себе си. Едно такова решение е Arduino Mega. То има цели 54 цифрови входно/изходни пина (от които 15 могат да се използват като PWM изходи), 16 аналогови входа, 4 UART (хардуерни серийни порта), 16 MHz кристален осцилатор, USB връзка, жак за захранване, Reset бутон и др. Просто го свързваме към компютър с USB кабел или го захранваме с батерия. Arduino Mega е дълъг 101.52mm, широк е 53.3mm, а теглото му е 37 g. Това означава, че спокойно може да се побере в робота, който има следните размери: дължина - 225mm; ширина - 220mm; височина - 108mm.

Следва монтирането на DC моторите върху платформата. С помощта на винтове, гайки и отвертка, двигателите се прикрепят към робота. Към тях след това се сглобяват гумите и се слага верига, която му помага да бъде стабилен и да преминава през трудни терени. Ето го и роботът след монтирането им (Фиг. 5.7):



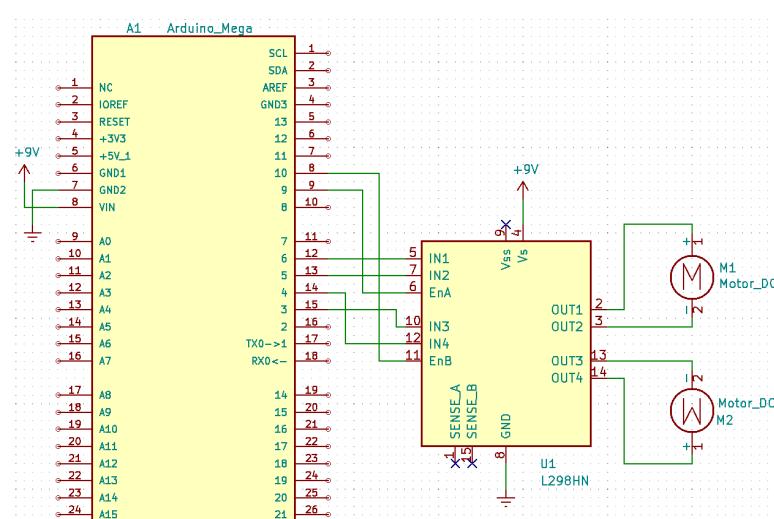
Фиг. 5.7 Робот с монтирани DC мотори и колела

Разбира се, тези DC мотори трябва да бъдат управлявани по някакъв начин. Това може да се извърши чрез т. нар. “Драйвер” (Driver). Драйверът ни позволява да контролираме работната скорост и посока на два мотора едновременно. Пример за двойно двупосочен моторен драйвер е L298N Dual H-Bridge Motor Driver. Схемата ни позволява лесно и независимо да управлявате два постояннотокови мотора с напрежение от 7 до 12V с консумация до 2A на всеки в двете посоки. Той е идеален за роботизирани приложения и много подходящ за свързване към микроконтролер, изискващ само няколко контролни линии на мотор. Физическите размери на модула са 3.4cm x 4.3cm x 2.7cm, т.е. той се побира идеално в робота. Когато се монтира (Фиг. 5.8) към основата на робота, то той вече има следния изглед:



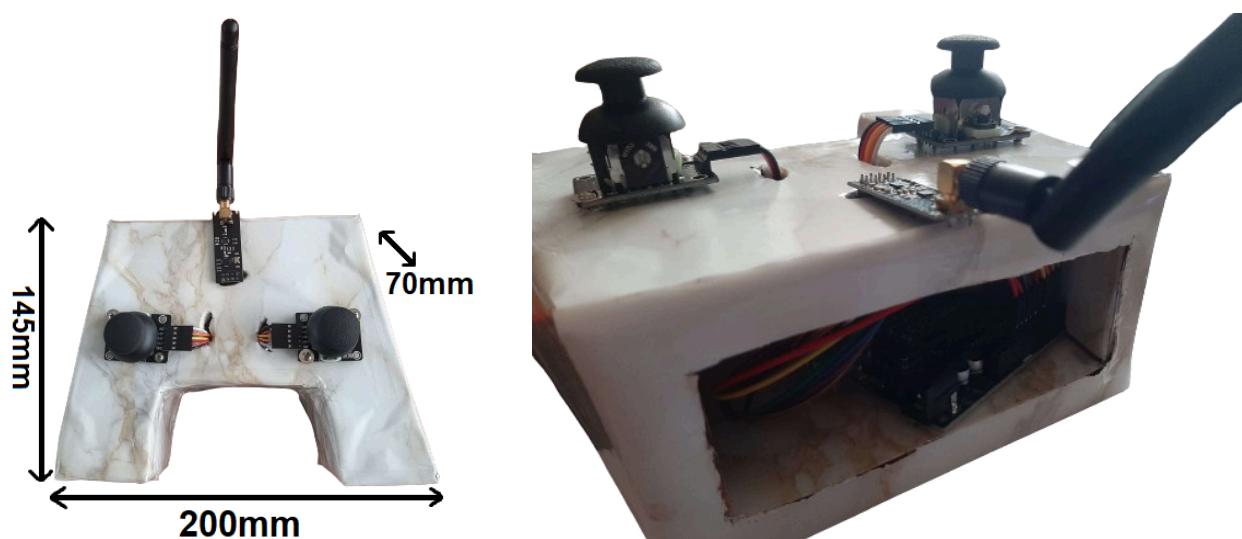
Фиг. 5.8 Монтирани DC мотори, Arduino Mega и L298N драйвер на основата на робота

Следващата стъпка е да се свържат по някакъв начин тези три компонента един с друг. Това се осъществява с помощта на проводници, като за + и - изводите на DC моторите е хубаво проводниците да бъдат запоени, за да се избегне максимално опасността да се откачат от моторите и те да не функционират правилно. На Фиг. 5.9 е представена електрическа схема на свързване на компонентите:



Фиг. 5.9 Електрическа схема на свързаните DC мотори, Arduino Mega и L298N драйвер

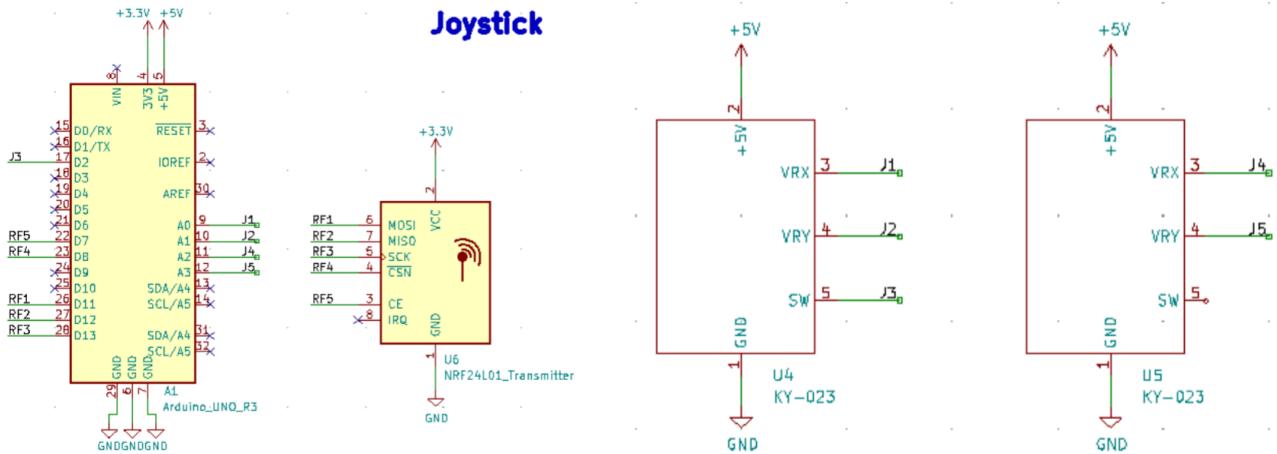
С примерен код за управление на посока на движение на моторите може да се тества дали те функционират правилно. Ако това е така, следващата стъпка е да можем да управляваме робота от разстояние, с помощта на джойстици. Разбира се, трябва данните да се предават от джойстиците към робота, поради което има нуждата от някакъв модул за комуникация. Едно такова решение е nRF24L01 модул, чиято работоспособност на пръв поглед не е много сложна. Като за начало се монтира основата на контролера и прикрепването на двата джойстика, микроконтролера (Arduino Uno R3) и единия nRF24L01 безжичен модул (предавател), описано по-подробно в “Глава 4. Проектиране и изграждане на хардуера”. На фиг. 5.10 са показани снимки на готовия за предаване на данни и информация контролер:



Фиг. 5.10 Монтиран контролер с електроника

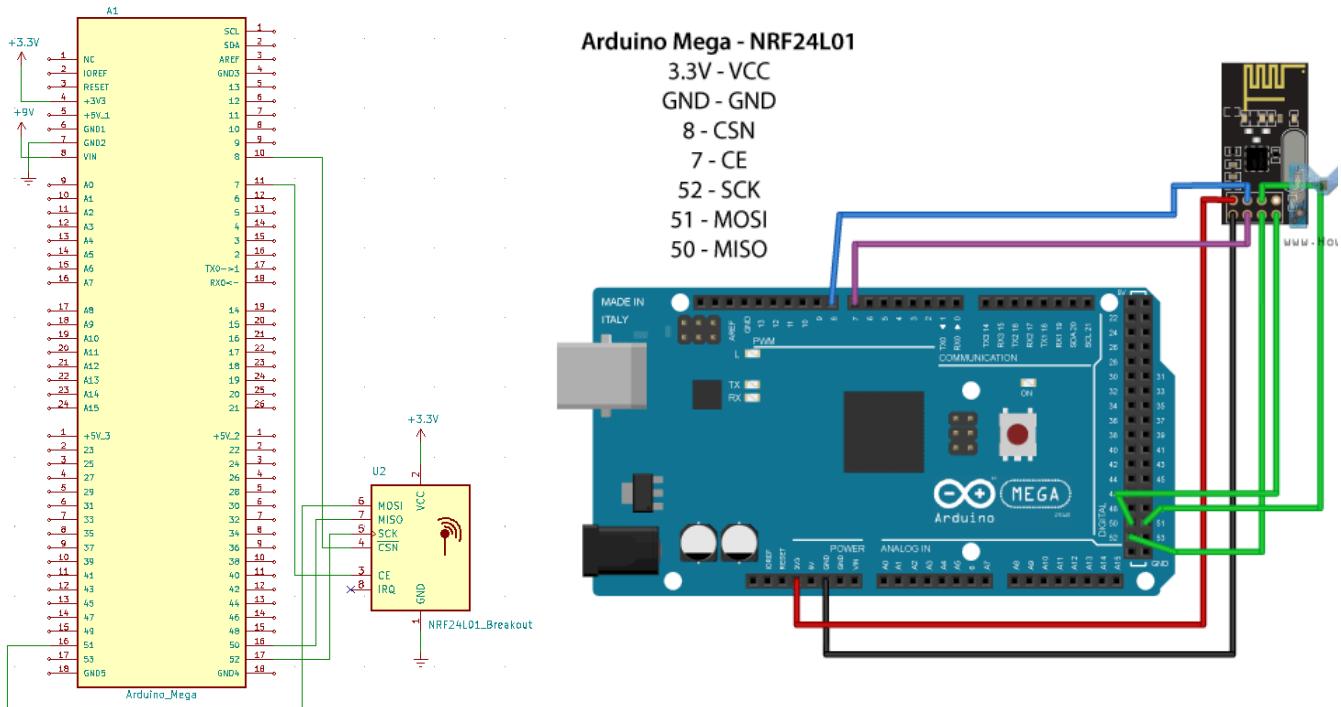
Поради ограниченото място вътре в контролера, а и нуждата от малко изводи, се използва микроконтролер Arduino Uno R3, който е с размери: дължина - 68,6mm; ширина - 53,4mm; тегло - 25 гр. NRF24L01 е модул, който е наличен в почти всички магазини за електроника, а и начинът му на работа не е сложен. Ето защо е избран за употреба в този проект, а размерите му са 30.3mm (дължина) x 14.5mm (ширина). Трябва да се съобрази начина на свързването му към микроконтролера, тъй като за осъществяването на SPI комуникацията помежду им, трябва да се използват конкретни изводи на Arduino Uno модула, а именно: 13 (SCK), 12 (MISO), 11 (MOSI). Двата джойстика служат за управление на робота в различни посоки (напред - назад - наляво - надясно), както и за управление на посоката на маркуча (наляво - надясно - нагоре - надолу). Размерите му са: ширина - 27mm, височина - 33mm; дължина - 45mm. Не на последно място, това са размерите на целия контролер: ширина - 145mm, височина - 70mm; дължина - 200mm. Теглото му е 300g, но това е главно заради материала, от който е изработен, а именно - дърво.

На фиг. 5.11 е показана цялата електрическа схема на контролера, с всички съставни електронни компоненти:



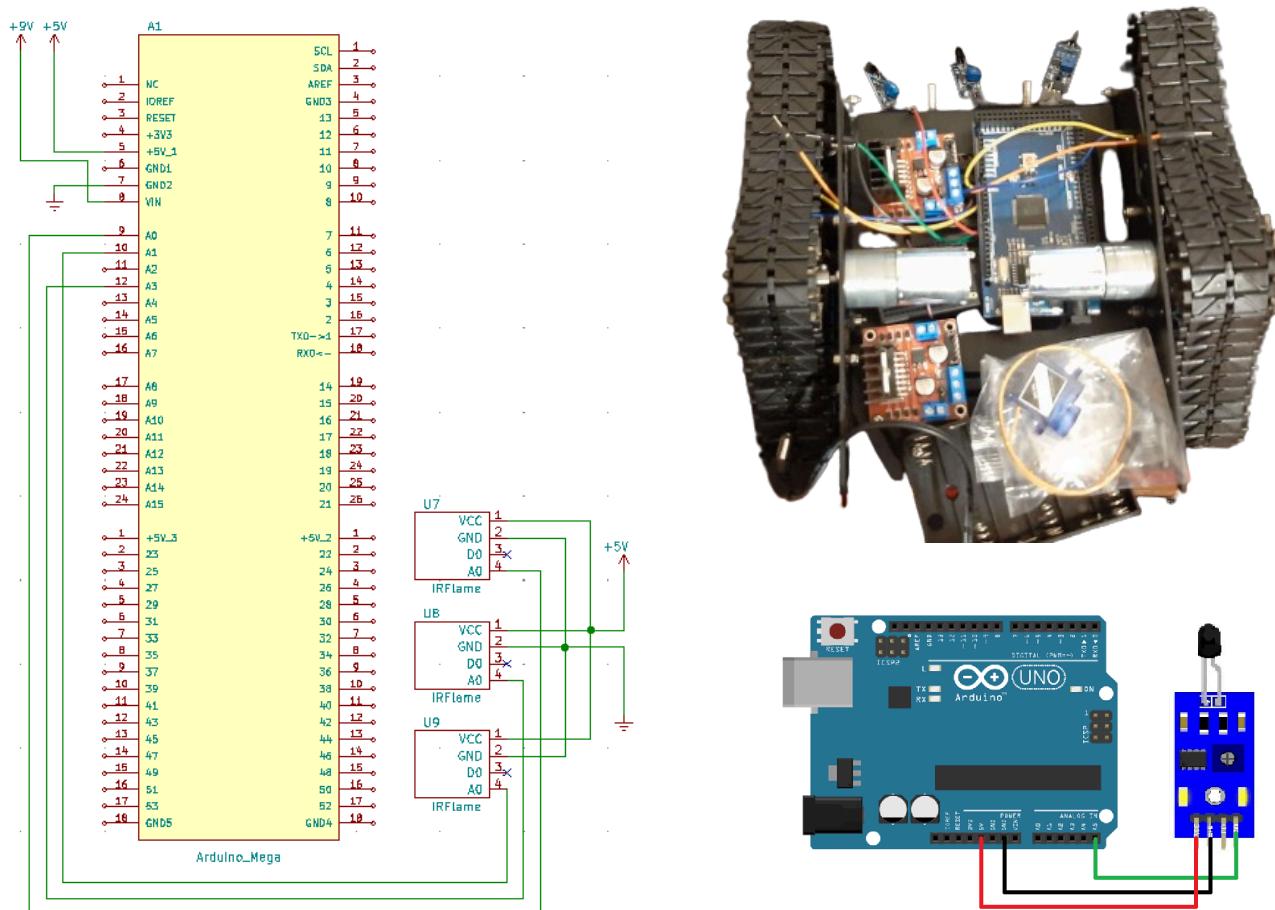
Фиг. 5.11 Електрическа схема на контролера

Следващата стъпка е свързването на втория nRF24L01 безжичен модул (приемник) към основата на робота. Трябва да се съобрази начина на свързването му към микроконтролера, тъй като за осъществяването на SPI комуникацията помежду им, трябва да се използват конкретни изводи на Arduino Mega модула, а именно изводи: 52 (SCK), 50 (MISO), 51 (MOSI). На фиг. 5.12 можем да видим примерно свързване на безжичния модул към Arduino Mega:



Фиг. 5.12 Електрическа схема на свързване на безжичен nRF24L01 модул към Arduino Mega

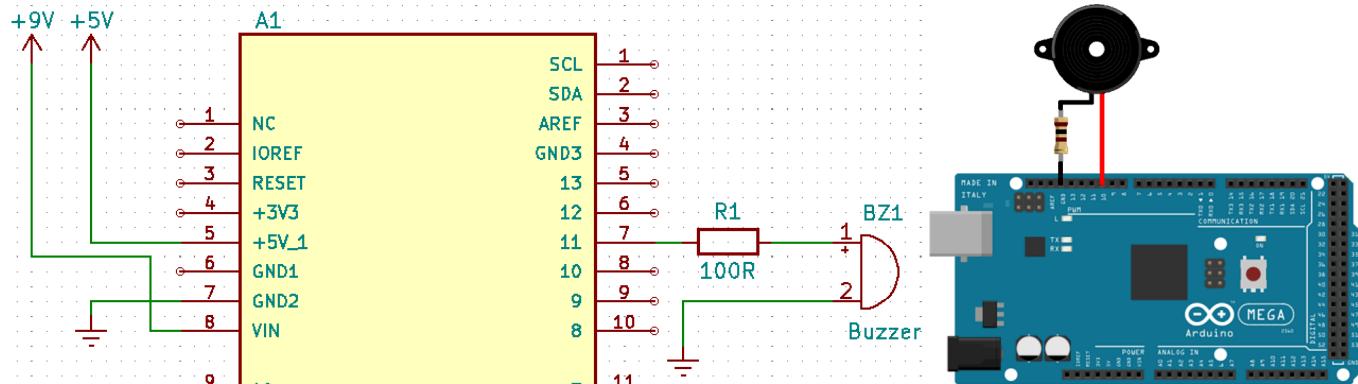
След написване на програма за комуникация между двата безжични nRF24L01 модула (предавател и приемник) и съответно такава за управление на посока на движение на моторите от разстояние, може да се тества дали те функционират правилно. Ако робота се движи напред, назад, наляво и надясно, управляван безжично, то всичко до този момент е наред. Оттук нататък трябва да се насочим към електронни компоненти, нужни на една пожарна кола. На първо място - това са сензорите за пламък. За този проект ще се използват 3 такива сензора, но в действителност те трябва да са много повече. Сензорът за пламък се състои от IR приемник, резистор, кондензатор, потенциометър и LM393 компаратор в интегрална схема. Модулът е изграден на базата на инфрачервен фотодиод. Работното напрежение е между 3.3V и 5.3V, с цифров изход за индикация на присъствие на сигнал. Отчитането се определя от LM393 компаратор. Платката е с размери 27mm x 15,5mm. Цифровият изходен сигнал се подава чрез логически нива, а аналоговият - под формата на напрежение, което се променя с приближаване или отдалечаване на пламъка. На фиг. 5.12 можем да видим електрическата схема на свързване на трите сензора за пламък към Arduino Mega, както и как изглежда роботът до този момент:



Фиг. 5.13 Електрическа схема на три сензора за пламък към Arduino Mega

Когато един от тези три сензора за пламък засече пламък/огън, то ще трябва потребителят да бъде уведомен по някакъв начин, че в околността е наличен огън. Едно от решенията е да се използва зумер, който да издаде звук с определена честота, когато някой от сензорите засече пожар. Обикновено зумерите са изградени от пиезоелектричен кристал, който може да вибрира, когато му се приложи електрическо напрежение. Този вибриращ елемент, с диаметър 12mm и работещ на 5V, създава аудиовълни във въздуха и генерира характерен звук. Поради капацитетния характер на пиеzo зумера, трябва да има начин той да се разреди, когато захранването е изключено. За да се подобри неговата производителност, лесното решение е да се постави резистор през пиеzo проводниците, за да се помогне за разреждането му. Например резистор от 100Ω може да свърши работа.

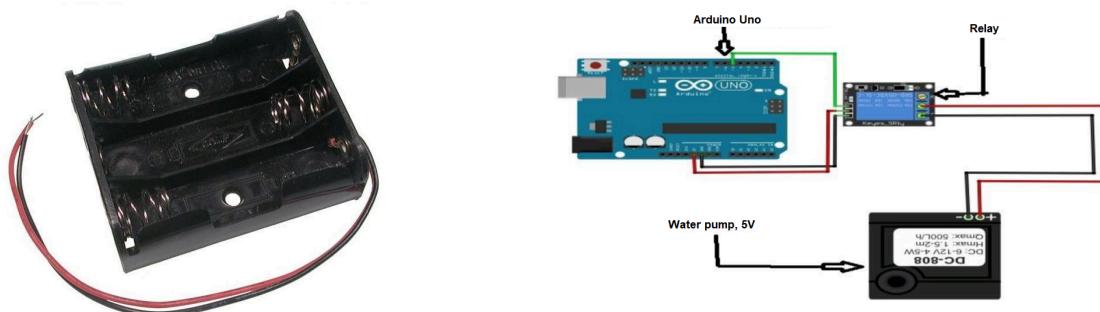
На фиг. 5.14 са представени схеми на свързване на зумер към Arduino Mega микроконтролер.



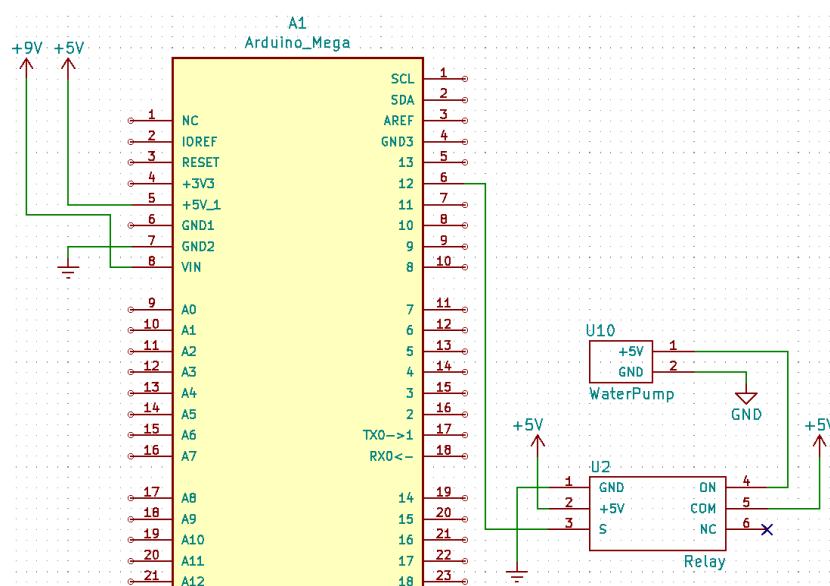
Фиг. 5.14 Електрическа схема на свързване на зумер към Arduino Mega

С правилен и работещ код вече може да се тества проекта и да се забележи, че когато един от сензорите за пламък засече огън, то зумерът се активира и издава звук, информирайки потребителя за наличие на пожар. Следващата, а може би и най-важна стъпка, е да се измисли начин за гасене на пожара. Най-логично е да се използва съд с вода, в който е монтирана водна помпа. Когато един или повече от сензорите засекат огън и съответно зумерът издаде писклив звук, то тогава потребителят веднага трябва да може да реагира, натискайки единия джойстик от контролера и активирайки водната помпа. Тя, от своя страна, ще засмуче вода от съда и ще я насочи към огнената среда чрез маркуч, който ще подпомогне да потуши пожара ефективно.

В този проект е избрана водна помпа, работеща на 5V, а тя се свързва към Arduino Mega микроконтролера с помощта на реле, което позволява на Arduino платката да контролира водната помпа. Размерите на водната помпа са: височина - 45mm, диаметър - 23mm; а тези на релето са: дължина - 35mm; ширина - 25mm; височина - 12mm. Преди да се свържат компонентите трябва да се реши друг проблем - по какъв начин ще се захранва водната помпа. Както е отбелязано, тя работи на 5V, следователно ще трябва или от 9V захранването на робота да се вземат 4.5V, с помощта на три AA батерии, или отделно ново захранване с гнездо за три AA батерии. Първият вариант е по-удобен, но батерийте ще се изхабят много по-бързо. Ето защо е избран вторият, където се използва отделно захранване от 4.5V за водната помпа, подаващо се към СОМ извода на релето. За да бъде всичко изяснено, на фиг. 5.15 е показан държачът за три AA батерии и примерно свързване на водна помпа към Arduino Uno, а на фиг. 5.16 е показана електрическа схема на водната помпа и релето, свързани към Arduino Mega микроконтролера:



Фиг. 5.15 Държач за 3 AA батерии и примерно свързване на водна помпа към Arduino Uno



Фиг. 5.16 Електрическа схема на водна помпа и реле към Arduino Mega

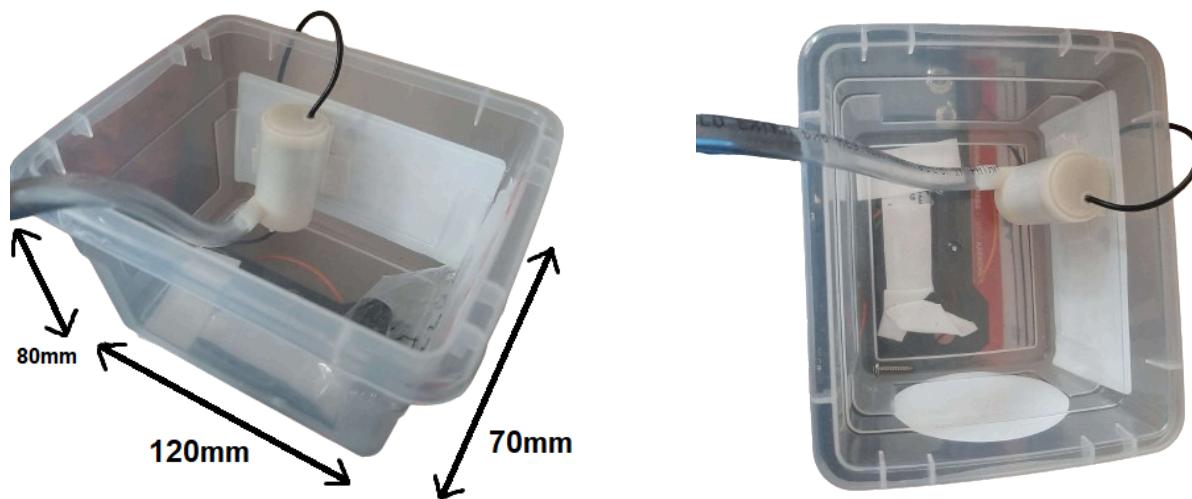
За проекта е избрана пластмасова кутия, която да играе ролята на съд с вода, в който е монтирана водната помпа. Размерите на кутията са: дължина - 120mm; ширина - 70mm; височина - 80mm. За да изчислим колко милилитра (ml) вода може да побере този съд, можем да използваме формулата за обем на правоъгълен паралелепипед:

$$\text{Обем} = \text{Дължина} * \text{Ширина} * \text{Височина};$$

С размери 120mm (дължина), 70mm (ширина) и 80mm (височина), изчислението е следното:

$$\text{Обем} = 120\text{mm} * 70\text{mm} * 80\text{mm} = 672\ 000\text{mm}^3;$$

Тъй като 1 ml е равен на 1000 кубични милиметра (mm^3), то следва, че: $672\ 000\text{mm}^3 = 672\text{ml}$. Така че този съд може да побере 672 ml вода. На фиг. 5.17 са показани снимки на съда за вода, а на фиг. 5.18 - съдът с вода, както и държачът за батерии, служещ за захранването му от 4.5 - 5V:

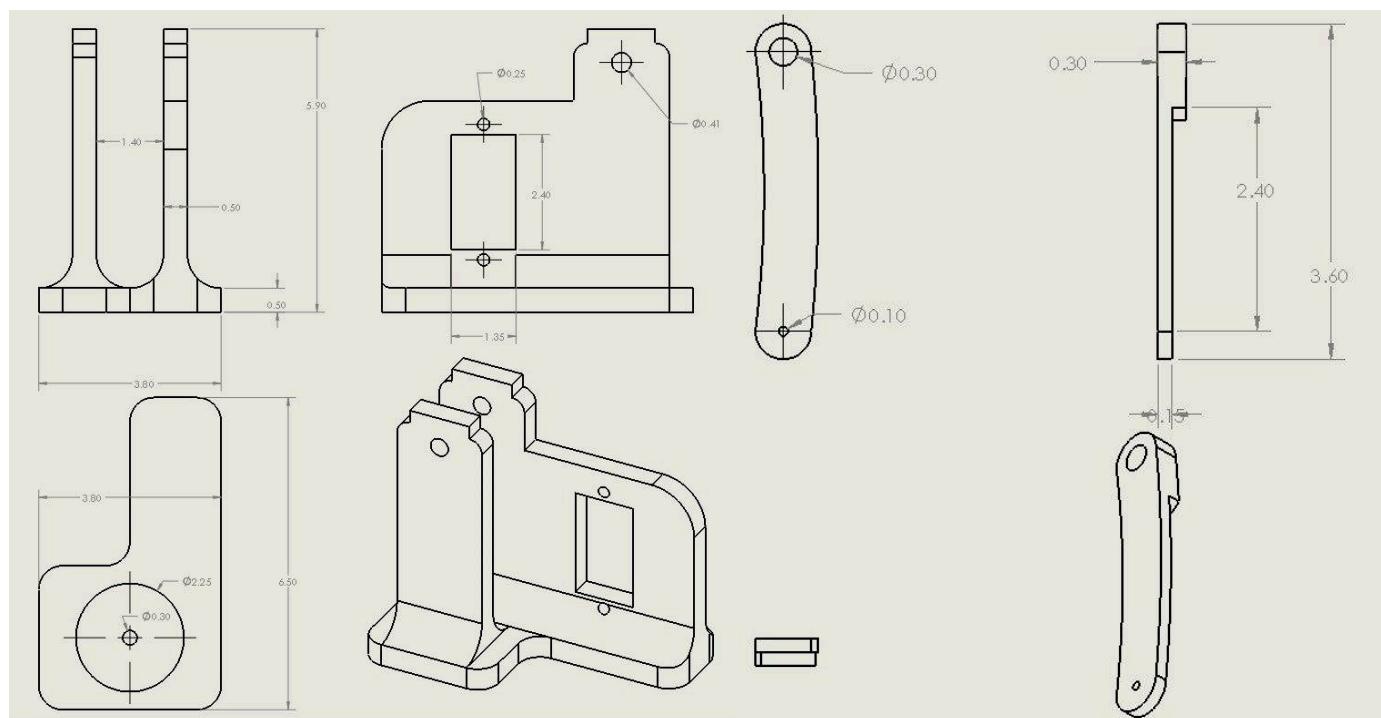


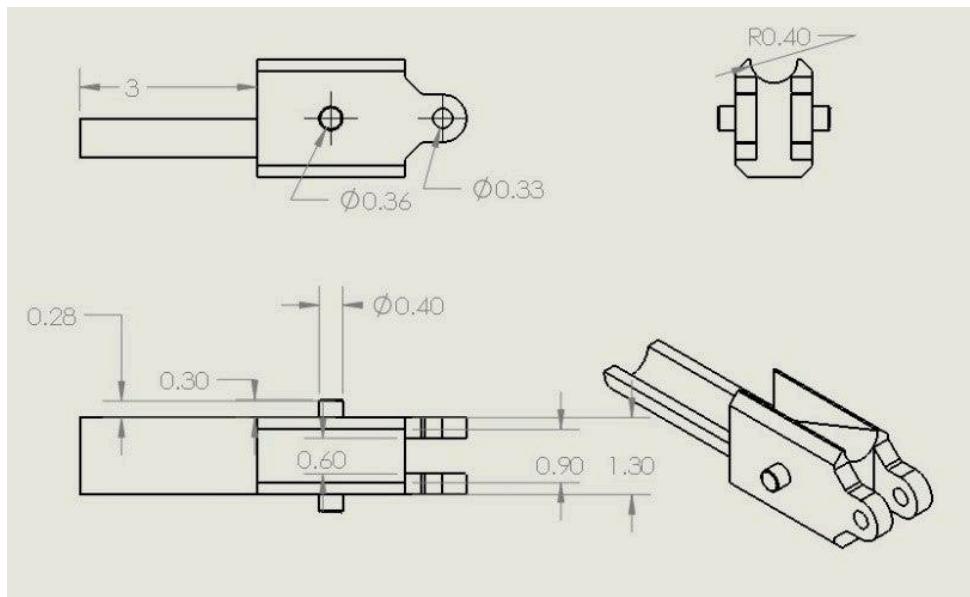
Фиг. 5.17 Съдът с вода, заедно с водната помпа



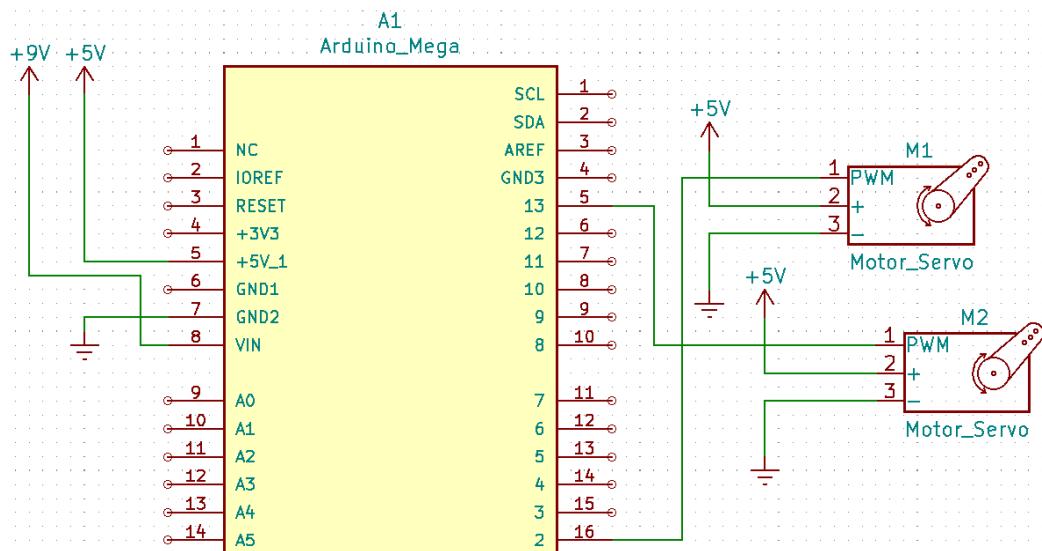
Фиг. 5.18 Съдът с вода, заедно с държача за батерии за захранването на помпата

Остават още 1-2 стъпки за завършването на проекта. Маркучът, прикрепен за водната помпа, трябва да може да бъде управляван наляво, надясно, нагоре и надолу. Това ще става с помощта на втория джойстик от контролера (това е причината те да са два). За да решим този проблем, може да се използват два серво мотора и един 3D модел, с цел по-красив вид на проекта и по-лесно прикрепване на моторите за него. За въртенето наляво и надясно на маркуча ще бъде отговорен серво мотор. По-конкретно може да се използва серво мотор S3003-180, който е с пластмасови зъбни колела и с PWM управляващ сигнал.. Моторът се захранва и управлява по три проводника свързани към конектор. Захранващото напрежение е между 4,8 - 6V DC, а теглото му е 38g. Той пък, от своя страна, се прикрепя към 3D модела. Този модел е начертан в програмния продукт SolidWorks и е принтиран с помощта на 3D принтер. В него има отвор, който е с точните размери за микро серво мотора, който ще е отговорен за движението на маркуча нагоре и надолу. Микро серво моторът е електромеханично устройство, състоящо се от постояннотоков мотор, редуктор и управляваща електроника. Изходящият вал се върти по определен ъгъл във всяка посока. В случая е избран модел SG90 - 180. Захранващото му напрежение е +5V DC. В заключение, тези серво мотори могат да осигурят относително прецизно движение в определен диапазон, както и консумират малко енергия. На фиг. 5.19 са показани снимки на размерите на 3D модела, на фиг. 5.20 - електрическа схема на свързването на двата серво мотора към главния микроконтролер - Arduino Mega, а на фиг. 5.21 - целият, обединен механизъм за въртене на маркуча:

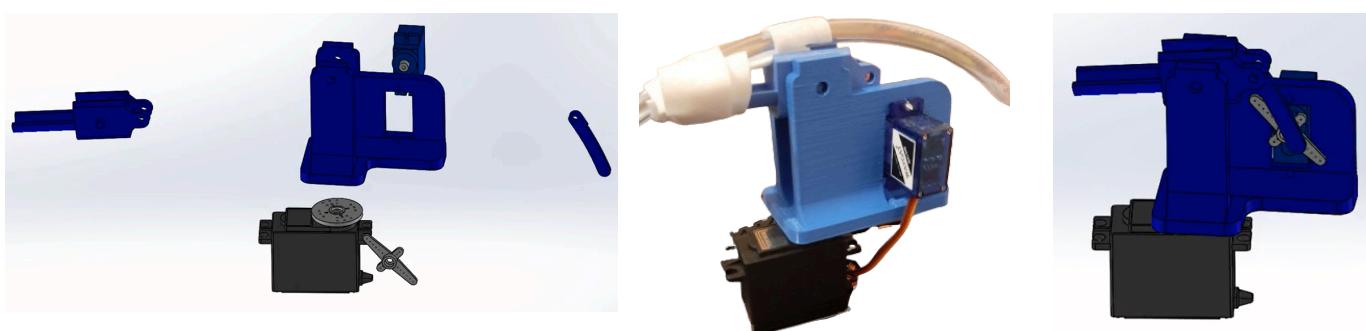




Фиг. 5.19 Размери на 3D частите за механизма за въртене на маркуча



Фиг. 5.20 Електрическа схема на двата серво мотора, свързани към Arduino Mega



Фиг. 5.21 Целият механизъм за въртене на маркуча

След като приложим програма за управление на маркуча от разстояние, с помощта на втория джойстик от контролера, то вече можем да кажем, че проектът е във финалната си фаза на работа. Не на последно място, всяко превозно средство има фарове и стопове. В нашия случай за фарове ще служат два бели светодиода, а за стопове - два червени. За повече сигурност, ще бъде добавен още един червен светодиод, чиято цел е да изльчи светлина, когато някой от трите сензора за пламък засече пожар. Всеки светодиод се нуждае от последователно (в повечето случаи) свързан резистор, който го защитава, ограничавайки тока, който тече през светодиода. Без резистор, ако се свърже светодиод директно към източник на напрежение, той може да претърпи прекомерно натоварване и да се повреди/изгори. За да определим подходящия резистор за червен и бял светодиод, трябва да знаем няколко неща:

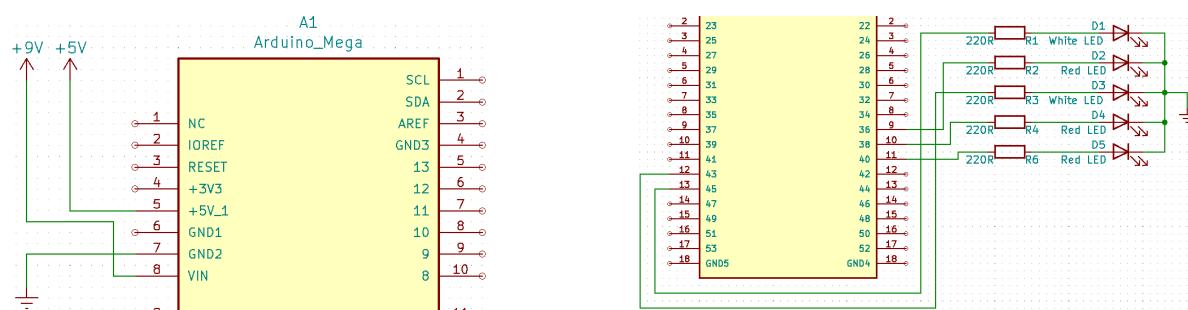
- 1) Напрежение на захранване (V_{in}): Това е напрежението, което се прилага към светодиода, в случая това е $V_{in} = 5V$;
- 2) Напрежение на светодиода (V_f): Това е напрежението, което светодиодът изисква, за да светне. За червен светодиод, обикновено V_f е около 1.8V до 2.2V, а за бял - около 3.0V до 3.6V;
- 3) Максимален ток на светодиода (I_f): Това е максималният ток, който светодиодът може да поеме, преди да се повреди/изгори. За много светодиоди I_f е обикновено около 20mA.

Когато имате тези данни, можете да използвате закона на Ом за да пресметнете стойността на необходимия резистор: $R = (V_{in} - V_f) / I_f$;

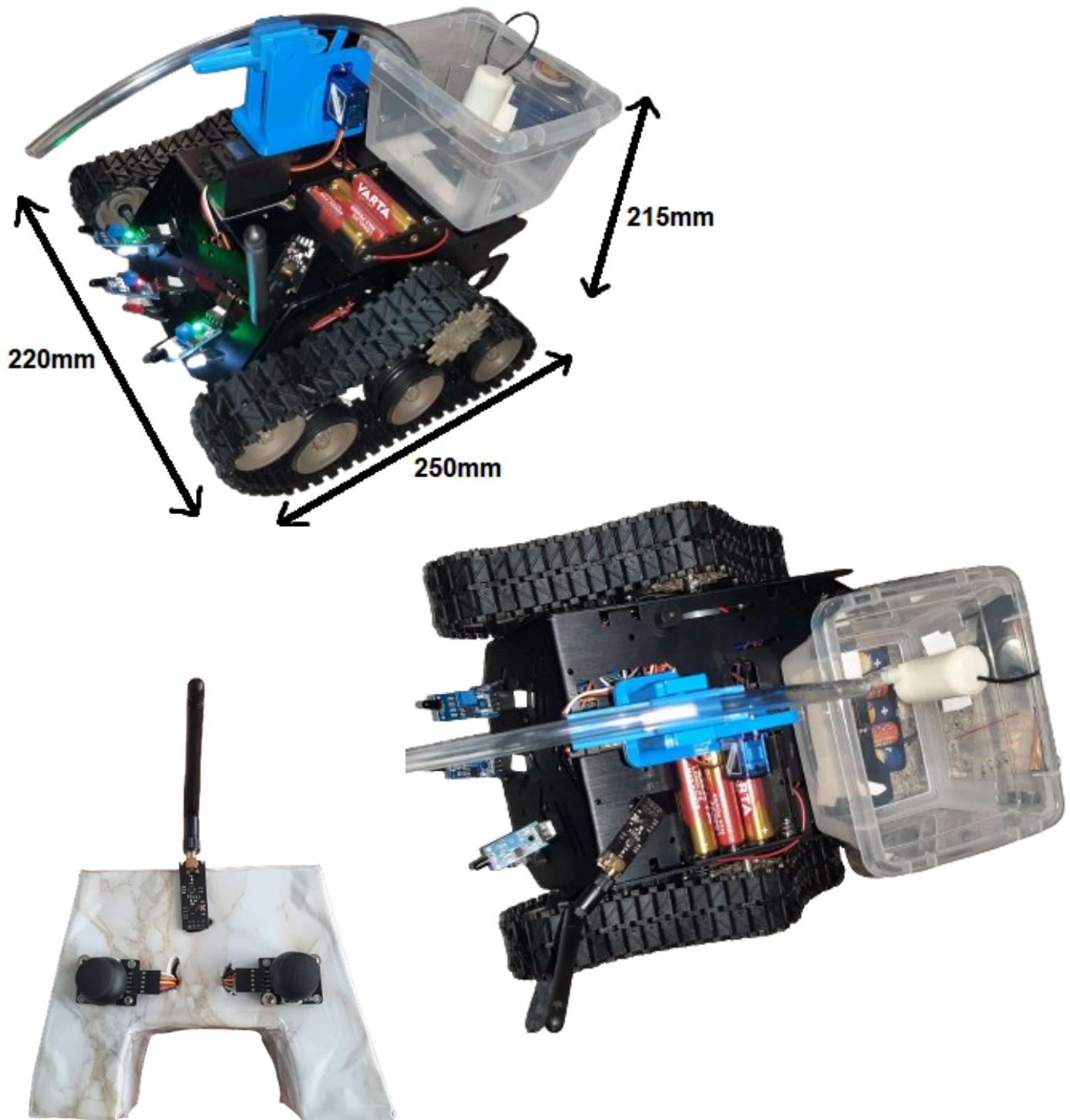
$$R_{RedLED} = (5V - 2V) / 0.02A = 150\Omega$$

$$R_{WhiteLED} = (5V - 3.2V) / 0.02A = 90\Omega$$

Получихме стойностите на резисторите, от които имаме нужда за светодиодите. Поради неналичие на други, са използвани резистори със стойност 220Ω. С 220Ω резистор може да се удължи живота на светодиода, но може също да го направи по-слабо светещ, особено ако е по-ярко осветление. След тест се забелязва, че всичко е наред със светодиода и той изльчва светлина достатъчно ярко. На фиг. 5.22 е представена електрическата схема на светодиодите, свързани към Arduino Mega.



След като се пусне програма и се види, че светодиодите работят по предназначение, то вече може да се каже, че проектът е напълно готов. Пълните електрически схеми са представени в “Глава 3. Принципна електрическа схема”. На фиг. 5.23 е представен готов вид на противопожарния робот, както и на контролера за управлението му. Също така, проектът е оразмерен, дължината му е 250mm, ширината - 220mm, а височината му - 215mm.



Фиг. 5.23 Снимки на готовия проект - противопожарен робот и контролер

ГЛАВА 6. Терминологии в Arduino. Комуникации в проекта

6.1 Въведение в АЦП и ЦАП:

Аналогово-цифровите преобразуватели (ADC) преобразуват аналогови величини, които са характерни за повечето явления в "реалния свят", до цифровия език, използван в обработка на информация, изчисления, предаване на данни и системи за управление. Цифрово-аналоговите преобразуватели (DAC) се използват за трансформиране на предавани или съхранени данни, или резултати от цифрова обработка, обратно към променливи от "реалния свят" за контрол, информация дисплей или допълнителна аналогова обработка.

Аналоговите входни променливи, независимо от техния произход, най-често се преобразуват в напрежения или токове. Предавателната характеристика на ЦАП и АЦП представлява начупена линия, поради дискретния характер на цифровата стойност, докато аналоговата величина може да заема произволни стойности в съответстващия обхват, с някакво отклонение. АЦП-то има следните характеристики: честота, разделителна способност и грешка.

Разделителната способност представлява всички различни стойности, които могат да бъдат изведени в определена област и се мери в битове. Интервалът на цифровите величини е от 0 до 2^n , като n са броят битове. В европейските телекомуникационни системи се използват 8 бита разделителна способност, а в Arduino Uno - 10 (стойности между 0 - 1023).

6.2 Въведение в PWM:

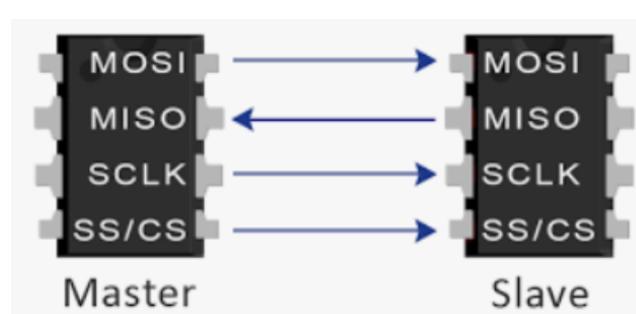
Цифровите сигнали са сигнали, които могат да бъдат представени с 0 или 1. Аналоговите сигнали, от друга страна, имат по-голям диапазон от възможни стойности, отколкото просто 0 или 1. И двата сигнала се използват в електрониката около нас, но се обработват много различно. Ако трябва да вземем аналогов вход, можем да получим аналоговите данни в реално време от сензор и след това с помощта на аналогово-цифров преобразувател (ADC) да ги преобразуваме в цифрови данни за микроконтролер. Но какво ще стане, ако трябва да управляваме аналогово устройство от нашия микроконтролер? Някои микроконтролери имат вграден цифрово-аналогов преобразувател (DAC) за извеждане на истински аналогов сигнал, за да управляват аналогови устройства и дори може да бъде използван външен DAC. Но DAC е сравнително скъп за производство по отношение на разходите. За да се преодолее тези проблеми и лесно да се постигне функционалността на ЦАП по много по-рентабилен начин, може да използва

техниката на PWM. Pulse Width Modulation [23] или ШИМ (Широчинно-импулсна модулация) е техника, използвана за управление на аналогови устройства, използвайки цифров сигнал. Тази техника може да се използва за извеждане на аналогов сигнал от цифрово устройство, като микроконтролер. Можем да се управляват мотори, светлини, задвижващи механизми и други, използвайки генерирания PWM сигнал. Важно нещо, което трябва да се отбележи е, че ШИМ не е истински аналогов сигнал. Цифровият сигнал е модифициран по начин, който фалшифицира аналогов сигнал. В контекста на Arduino, микроконтролерът има 8 бита разделителна способност ($2^8 = 255$ стъпки), когато извежда сигнал с помощта на PWM. Диапазонът на изходното напрежение е от 0 до 5V.

6.3 Комуникация в проекта - SPI протокол:

SPI (Serial Peripheral Interface) [24] е вид комуникация, която се използва за къси разстояния. Типичните приложения включват SD карти, дисплей с течни кристали и др. SPI устройствата използват архитектурата "Master-Slave". Главното устройство създава рамката за четене и писане. SPI шината определя четири логически сигнала: 1) SCLK - сериен часовник (изход от главното устройство); 2) MOSI - основен изход (извеждане на данни от главното устройство); 3) MISO - основен вход (извеждане на данни от вторичното устройство); 4) SS - избор на вторично устройство (изход от главното устройство). SPI може да работи с едно главно и няколко подчинени устройства.

На Фиг. 5.1 е представена SPI комуникация:



Фиг. 6.1 SPI комуникация

В разглеждания проект безжичният модул nRF24L01 си комуникира с микроконтролерите Arduino Mega 2560 и Arduino Uno с помощта на SPI протокола, като използва библиотека като RF24. На Фиг. 5.2 е представена таблица с цифровите пинове на двата микроконтролера, предназначени за SPI комуникация.

Arduino	SCK	MISO	MOSI	SS
Uno	13	12	11	10
Mega	52	50	51	53

Фиг. 6.2 Таблица с номерата на пиновете на Arduino Mega и Uno, предназначени за SPI комуникация

С помощта на SPI, микроконтролерите прехвърлят данни към и от модула nRF24L01, като използват специфични команди и функции, за да настройват параметрите на комуникация и да обменят информация в безжичната мрежа.

6.4 Комуникация в проекта - ISM:

Модулът nRF24L01 е проектиран да работи с честотна лента 2.4 - 2.4835GHz. Това е безжичен приемо-предавател, предназначен за комуникация във високочестотен ISM (Industrial, Scientific, and Medical) диапазон. Той използва протоколи за безжична комуникация, които могат да осигурят надеждна и бърза връзка между устройства на кратко разстояние (до 100 m). Скоростта на трансфер на данни е между 250kbps до 2Mbps [baud]. В комуникацията между устройства, модулите nRF24L01 работят като предаватели и приемници на радиосигнали, като използват протокол за управление на безжичния обмен на данни. Те са изключително полезни за проекти, които изискват безжична комуникация с ниска консумация на енергия и минимално електромагнитно смущение. Точно това се изисква и в текущия проект.

ГЛАВА 7. Софтуер на проекта

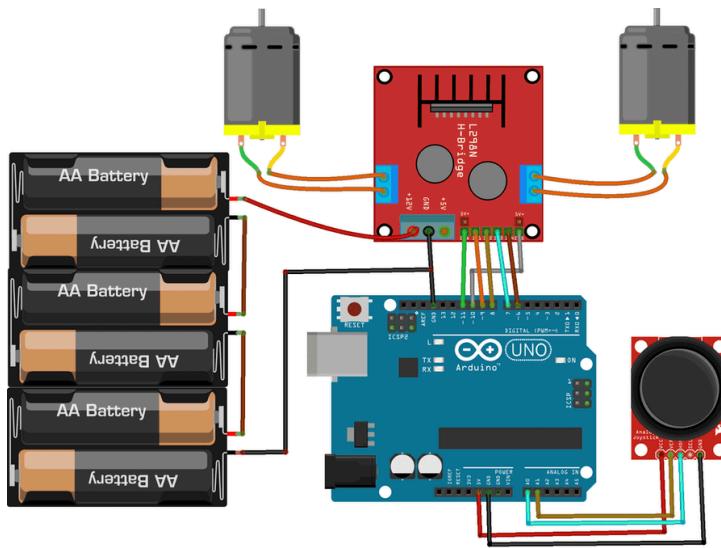
Контролът на версии е осъществен чрез GitHub. Това е линкът към ханилището - <https://github.com/Boyan7577/VMKS>.

С цел яснота, преди да бъде разгледан и обяснен кодът на текущия проект, първо ще бъдат показани примерни програми за управление на отделни електронни елементи, нужни за изграждането на един противопожарен робот.

7.1 Примерна програма за движение на DC мотор в различни посоки:

Първата примерна програма показва как да задвижваме DC мотор с Arduino [27]. Тъй като Arduino не може да осигури достатъчно мощен сигнал, който да задвижи постояннотоков двигател, ето защо трябва да се усили сигналът и да се преобразува в достатъчно мощен, за да захрани постояннотоковия мотор. За да се направи това, трябва да се използва драйвер за мотори, който ще приеме идващия сигнал от Arduino и ще захрани моторите с добавяне на допълнителна мощност, взета от някаква батерия/източник на напрежение.

За този проект ще трябват следните неща: 1) Arduino; 2) Модул за двигател L298N; 3) 2 DC мотора; 4) Джойстик модул; 5) 12V батерия. На фиг. 7.1 е представена схемата на свързване на проекта:



Фиг. 7.1 Схема на свързване на проекта

Свързванията са както следва: ENA - извод 11; IN1 - извод 9; IN2 - извод 8; IN3 - извод 7; IN4 - извод 6; ENB - извод 10; 12V - 5 до 12V захранване или батерия; GND - GND (минус на захранване или батерия). Накрая се свързват двата DC мотора от двете страни на L298N. За захранване на L298N модула се използват 6 AA батерии от 1.5V всяка, т.е. общо 9V. Следва свързването на джойстик модула с

Arduino, а то е както следва: VCC - 5V; VRX - A1; VRY - A0; GND - GND. ENA и ENB изводите на L298N драйвера се използват чрез премахване на джъмпера върху тях и служат за контролиране на скоростта на DC моторите; трябва да се свържат към PWM изводите на Arduino. Логическите изводи на драйвера (IN1, IN2, IN3, IN4) се свързват към цифровите изводи на Arduino, които помагат за контролиране на въртенето и скоростта на DC моторите. 5V-ят линеен регулатор ще намали захранващото напрежение до 5V. Следва програмата за управление на проекта, първо се инициализират изводите на джойстика, а след това и тези на драйвера, както и променливи за съхраняване на данни:

```
//Joystick Pins
int x_key = A0;
int y_key = A1;
int x_pos;
int y_pos;

//Motor Pins
int EN_A = 11; //Enable pin for first motor
int IN1 = 9; //control pin for first motor
int IN2 = 8; //control pin for first motor
int IN3 = 7; //control pin for second motor
int IN4 = 6; //control pin for second motor
int EN_B = 10; //Enable pin for second motor
//Initializing variables to store data
int motor_speed;
int motor_speed1;
```

Следва задължителната функция `setup()`, която инициализира и задава първоначалните стойности на изводите. Тук се инициализират изводите на драйвера като изходни (OUTPUT), а на джойстика - като входни (INPUT).

```
void setup () {
  Serial.begin (9600); //Starting the serial
  communication at 9600 baud rate
  //Initializing the motor pins as output
  pinMode(EN_A, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(EN_B, OUTPUT);

  //Initializng the joystick pins as input
  pinMode (x_key, INPUT) ;
  pinMode (y_key, INPUT) ;
}
```

След създаването на функция `setup()`, която инициализира и задава първоначалните стойности, функцията `loop()` прави точно това, което подсказва името ѝ, и повтаря последователно, позволявайки на програмата да се промени и да реагира. Тук се четат стойностите по абсцисата и ординатата на джойстик модула:

```
void loop () {
  x_pos = analogRead (x_key) ; //Reading the horizontal movement value
  y_pos = analogRead (y_key) ; //Reading the vertical movement value
```

Този блок код управлява мотора въз основа на позицията на джойстика. Ако “x_pos” (положението на джойстика по оста X) е по-малко от 400, моторът се завърта в посока на часовниковата стрелка, като скоростта му се задава чрез функцията map, която превръща стойностите от диапазона 400 - 0 в диапазона 0 - 255. Ако “x_pos” е между 400 и 600, моторът спира да се движи:

```

if (x_pos < 400){ //Rotating the left motor in clockwise direction
  motor_speed = map(x_pos, 400, 0, 0, 255); //Mapping the values to 0-255 to move the motor
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(EN_A, motor_speed);
}

else if (x_pos>400 && x_pos <600){ //Motors will not move when the joystick will be at center
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}

```

Следващият блок код добавя управление на мотора в зависимост от позицията на джойстика. Ако “x_pos” е над 600, левият мотор се завърта обратно на часовниковата стрелка, като скоростта му се задава от “map”, която превръща стойностите от диапазона 600 - 1023 в 0 - 255. Ако “y_pos” е под 400, десният мотор се завърта в посока на часовниковата стрелка, като скоростта му също се задава чрез “map”, която превръща стойностите от 400 - 0 в 0 - 255.

```

else if (x_pos > 600){ //Rotating the left motor in anticlockwise direction
  motor_speed = map(x_pos, 600, 1023, 0, 255);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(EN_A, motor_speed);
}

if (y_pos < 400){ //Rotating the right motor in clockwise direction
  motor_speed1 = map(y_pos, 400, 0, 0, 255);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(EN_B, motor_speed1);
}

```

Последният блок код завършва управлението на десния мотор. Ако “y_pos” е между 400 и 600, десният мотор спира. Ако “y_pos” е над 600, десният мотор се завърта обратно на часовниковата стрелка, като скоростта му се задава от “map”, която превръща стойностите от диапазона 600 - 1023 в 0 - 255:

```

else if (y_pos>400 && y_pos <600) {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
}

else if (y_pos > 600){ //Rotating the right motor in anticlockwise direction
    motor_speed1 = map(y_pos, 600, 1023, 0, 255);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(EN_B, motor_speed1);
}
}

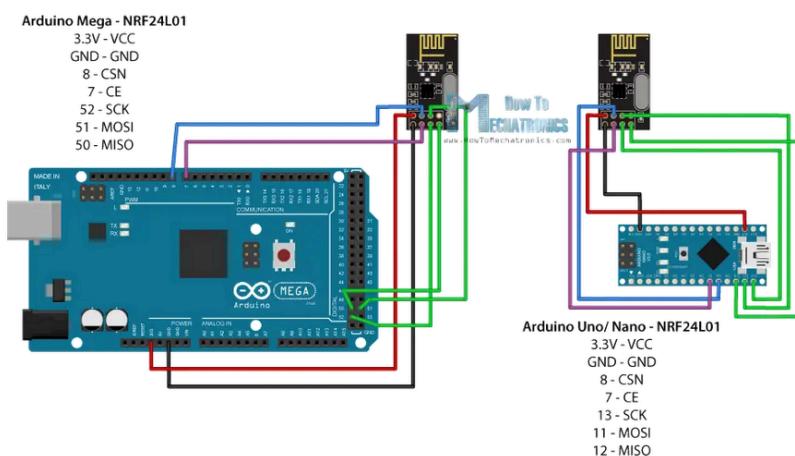
```

След като се качи кода на Arduino микроконтролера, с преместване на джойстика в различни посоки, може да се види, че според движението на джойстика, моторите ще работят правилно. Това помага за направата на радиоуправляема кола робот.

7.2 Примерна програма за комуникация между два NRF24L01 модула:

Втората примерна програма показва как да осъществява безжична комуникация между две платки Arduino с помощта на transciever модулите nRF24L01 [28]. Модулът nRF24L01 е много популярен избор за безжична комуникация при използване на Arduino.

За да се обясни безжичната комуникация, ще бъде направена програма, която изпраща съобщение „Hello World“ от едно Arduino към друго. За целта се използват Arduino Mega и Arduino Nano (фиг. 7.2), а свързването между безжичния модул и микроконтролера трябва да съответства на правилата, описани в “Глава 6. Терминологии в Arduino. Комуникации в проекта”:



Фиг. 7.2 Схема на свързване на проекта

След като свържем модулите nRF24L01 модулите към Arduino платките, ние сме готови да направим програмите както за предавателя, така и за приемника. Първо трябва да се изтегли и инсталира библиотеката RF24, което прави програмирането по-лесно. Можем също да се инсталира тази библиотека директно от Arduino IDE Library Manager. Ще бъдат обяснени паралелно програмите както за предавателя, така и за приемника.. Първо са включени основния SPI и новоинсталираните RF24 библиотеки и се създава RF24 обект. Двата аргумента след това са CSN и CE изводите. След това се създава масив от байтове, който представлява адресът, чрез който двата модула ще комуникират. Може да се промени стойността на този адрес на произволен низ от 5 букви.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";
```

В setup() функцията трябва да се инициализира радио обекта и с помощта на функцията radio.openWritingPipe() се задава адресът на приемника, на който ще се изпращат данни, 5-буквеният низ, който е зададен преди това. От другата страна, в приемника, чрез функцията radio.openReadingPipe() задаваме същия адрес и по този начин позволяваме комуникацията между двата модула. След това с помощта на функцията radio.setPAlevel() се задава нивото на усилвателя на мощността, в нашия случай е настроен на минимум. След това имаме функцията radio.stopListening(), която задава модула като предавател, а от другата страна имаме функцията radio.startListening(), която задава модула като приемник.

<pre>void setup() { radio.begin(); radio.openWritingPipe(address); radio.setPAlevel(RF24_PA_MIN); radio.stopListening(); }</pre>	<pre>void setup() { Serial.begin(9600); radio.begin(); radio.openReadingPipe(0, address); radio.setPAlevel(RF24_PA_MIN); radio.startListening();</pre>
--	--

Предавател

Приемник

В `loop()` функцията, на предавателя, се създава масив от знаци, на които се присвояват съобщението „Hello World“. С помощта на функцията `radio.write()` се изпраща това съобщение до получателя. Първият аргумент е променливата, която трябва се изпрати. Използвайки „&“ преди името на променливата, всъщност се задава индикация на променливата, която съхранява данните, които трябва да бъдат изпратени, и използвайки втория аргумент, се задава броят байтове, които трябва да се вземат от променливата. В този случай функцията `sizeof()` получава всички байтове от низа „text“. В края на програмата се добавя 1 секунда забавяне.

От другата страна, в приемника, в `loop()` функцията, използвайки функцията `radio.available()`, се проверява дали има данни за получаване. Ако това е вярно, първо се създава масив от 32 елемента, наречен „text“, в който се записват входящите данни. С помощта на функцията `radio.read()` се четат и съхраняват данните в променливата „text“. Накрая се отпечатва текст на серийния монитор. След като се качат и двете програми, може да се стартира серийният монитор на приемника и ще се забелжи, че съобщението „Hello World“ се отпечатва всяка секунда.

```
void loop() {
    const char text[] = "Hello World";
    radio.write(&text, sizeof(text));
    delay(1000);
}
```

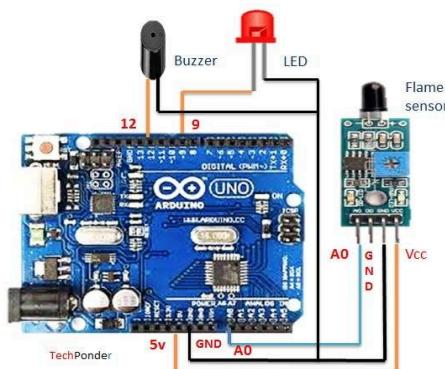
Предавател

```
void loop() {
    if (radio.available()) {
        char text[32] = "";
        radio.read(&text, sizeof(text));
        Serial.println(text);
    }
}
```

Приемник

7.3 Примерна програма за сензор за пламък, заедно със зумер и LED:

Следващата програма представлява откриване на пожар с помощта на Arduino и сензор за пламък [29]. Необходимите хардуерни компоненти са: 1) Сензор за пламък (аналогов изход); 2) Arduino; 3) LED и резистор; 4) Зумер; 5) Свързващи проводници. Това е схемата на свързване на проекта (фиг. 7.3):



Фиг. 7.3 Схема на свързване на проекта

Ето и описание на свързването на отделните компоненти към микроконтролера, който в случая е Arduino Uno:

Сензор за пламък към Arduino: 1) VCC -> VCC; 2) GND -> GND; 3) A0 -> A0;
LED светодиод към Arduino: 1) LED+ (анодът) е свързан към деветия извод на Arduino Uno; 2) LED- (катодът) е свързан към извода за земя на Arduino Uno;
Зумер към Arduino: 1) Buzzer+ е свързан към дванадесетия извод на Arduino Uno; 2) Buzzer- е свързан към GND извода на Arduino Uno. Да преминем към кода на проекта. В началото се дефинират изводите на електронните елементи, а в setup() функцията, изводите на LED светодиода и зумера се слагат като изходи (OUTPUT) и се стартира серийна комуникация с 9600 [baud].

```
int sensorPin = A0; // select the input pin for the LDR
int sensorValue = 0; // variable to store the value coming from the sensor
int led = 9; // Output pin for LED
int buzzer = 12; // Output pin for Buzzer

void setup() {
    // declare the ledPin and buzzer as an OUTPUT:
    pinMode(led, OUTPUT);
    pinMode(buzzer, OUTPUT);
    Serial.begin(9600);
}
```

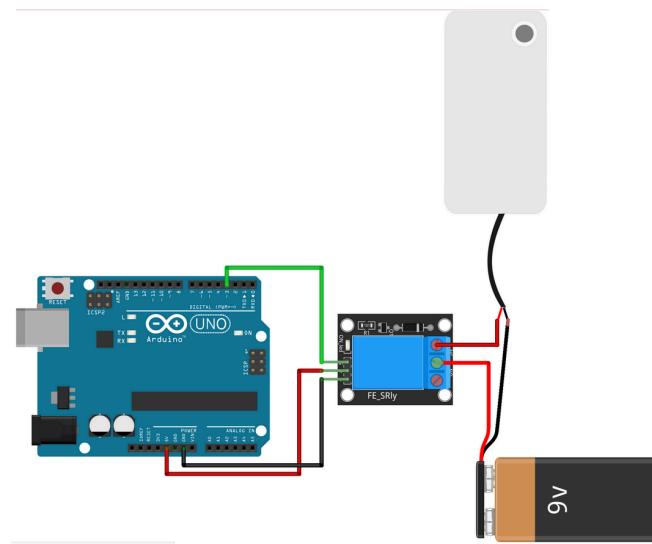
В loop() функцията се чете стойността от сензора и се отпечатва в серийния монитор, заедно с приветствено съобщение. Ако стойността от сензора е по-малка от 100 (индикатор за наличие на огън), LED светодиодът и зумерът се активират за 1 секунда, след което се изключват и цикълът се повтаря с пауза, равна на стойността от сензора.

```
void loop() {
    Serial.println("Welcome to TechPonder Flame Sensor Tutorial");
    sensorValue = analogRead(sensorPin);
    Serial.println(sensorValue);
    if (sensorValue < 100) {
        Serial.println("Fire Detected");
        Serial.println("LED on");
        digitalWrite(led, HIGH);
        digitalWrite(buzzer, HIGH);
        delay(1000);
    }
    digitalWrite(led, LOW);
    digitalWrite(buzzer, LOW);
    delay(sensorValue);
}
```

Резултатите се показват в серийния монитор. Когато има пламък, светодиодът и зумерът автоматично се включват, а когато няма пламък, Arduino Uno автоматично изключва светодиода и зумера. Тук, въз основа на състоянието на стаята, праговата стойност, която е взета, е 100 за сензора за пламък. Когато се постави сензор за пламък близо до пламък, Arduino автоматично включва светодиода и зумера. Когато се премахне пламъкът от сензора за пламък, Arduino автоматично изключва светодиода и зумера.

7.4 Примерна програма за водна помпа, управлявана с реле:

В този проект се използва водна помпа и едноканален 5V релеен модул [30]. Изводите на релето са както следва: 1) VCC - напрежение, мощност за модула; 2) GND - земя, захранване на модула, от GND на микроконтролера; 3) IN/SIG - вход/сигнал, цифров изход на микроконтролера; 4) COM - общ, свързване към общия проводник на външното захранване; 5) NO - нормално отворен, свързва се към положителния извод на устройството; 6) NC - нормално затворен, свържете се към положителния извод на устройството. На фигура 7.4 е представена схемата на свързване на проекта:



Фиг. 7.4 Схема на свързване на проекта

Ето как са осъществена свързаността: 1) VCC -> 5V; 2) GND -> GND; 3) IN/SIG -> извод 3; 4) NO - червен (+) проводник на водната помпа; 5) COM - червен (+) проводник на батерията. Черният (-) проводник на водната помпа и черният (-) проводник на батерията трябва да се свържат заедно. Следва прост код, който кара водната помпа да се включи за 1 секунда и да се изключи за 1 секунда.

Този блок код контролира реле чрез Arduino Uno. В началото се дефинира IN/SIG извода на релето. В setup() функцията изводът, свързан към релето, се слага като изход (OUTPUT). В loop() функцията релето се включва за 1 секунда, след което се изключва за 1 секунда, и този цикъл се повтаря непрекъснато.

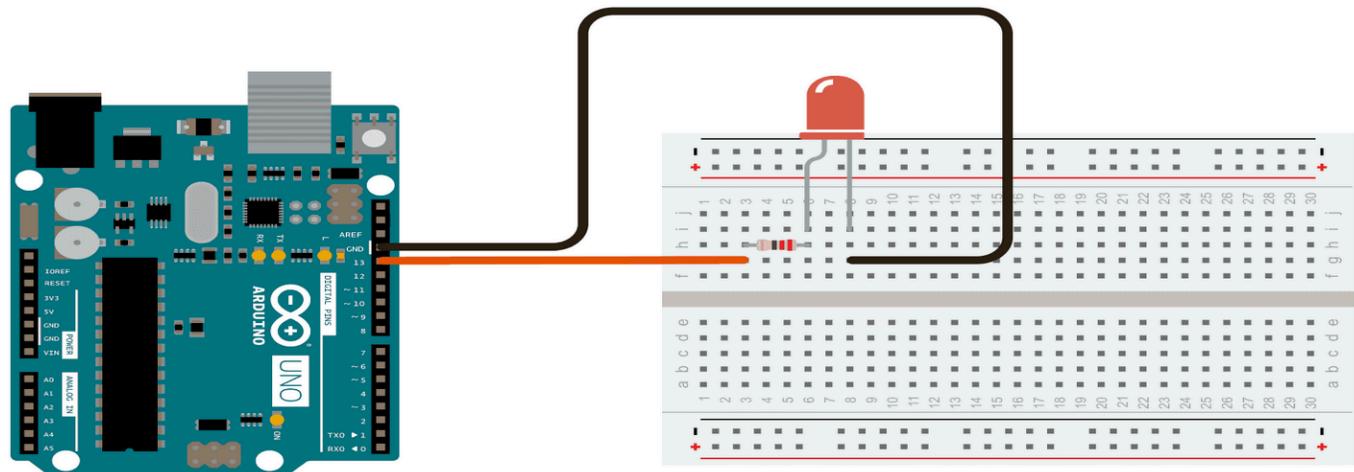
```
const int RELAY_PIN = 3; // the Arduino pin, which connects to the IN pin of relay

// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin as an output.
    pinMode(RELAY_PIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(RELAY_PIN, HIGH); //turn on
    delay(1000);
    digitalWrite(RELAY_PIN, LOW); //turn off
    delay(1000);
}
```

7.5 Примерна програма за мигане на светодиод:

Този пример показва най-простото нещо, което може да се направи с Arduino, а именно - мигащ светодиод [31]. Нужният хардуер е: 1) Arduino платка; 2) LED светодиод; Резистор - 220Ω . На фиг. 7.5 е показана схемата на свързване на проекта:



Фиг. 7.5 Схема на свързване на проекта

Дългият крак на светодиода (анод) се свързва към единия край на резистора. Другият край на резистора се свързва към извод 13 на Arduino Uno. Късото краче на светодиода (катод) се свързва към GND извода на Arduino Uno. Стойността на резистора, свързан последователно със светодиода, е 220Ω . Ето го и обяснението на

кода: този блок код управлява светодиод (LED) свързан към пин 13 на Arduino. В `setup()` функцията извод 13 се слага като изход (OUTPUT). В `loop()` функцията LED светодиодът се включва за 1 секунда, след което се изключва за 1 секунда, и този цикъл се повтаря непрекъснато.

```
int LEDpin = 13;

void setup() {
    // put your setup code here, to run once:
    pinMode(LEDpin, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(LEDpin, HIGH);
    delay(1000);
    digitalWrite(LEDpin, LOW);
    delay(1000);
}
```

7.6 Arduino IDE:

Arduino IDE (Integrated Development Environment) е софтуерна програма, която се използва за писане, компилиране и качване на кодове на Arduino платки. Тя предоставя лесен за използване интерфейс, който включва текстов редактор за писане на код, област за съобщения, текстова конзола, лента с инструменти за общи функции и серия от менюта. Програмите, написани в Arduino IDE, се наричат скици (sketches) и се пишат на езика Arduino, който е базиран на C/C++. IDE-то включва и различни библиотеки, които улесняват интеграцията със сензори, модули и други периферни устройства. Когато кодът е написан, той може да бъде компилиран и качен на Arduino платката чрез USB свързаност. Arduino IDE е достъпна за различни операционни системи като Windows, macOS и Linux и е свободен софтуер с отворен код. На фиг. 7.6 е представено логото на програмата:

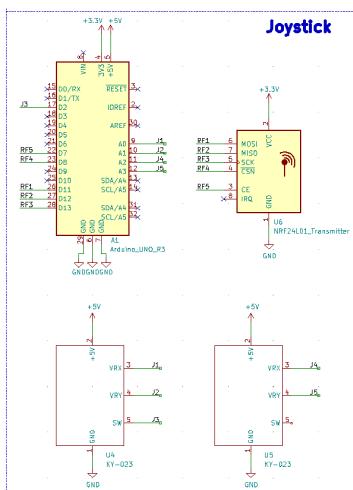


Фиг. 7.6 Лого на Arduino IDE

След като е разгледана концепцията на работа на отделните компоненти, сега ще се върнем в същинския проект и електронните елементи ще бъдат обединени в две програми, едната за контролера, който играе ролята на “предавател”, а другата - за противопожарния робот, който е “приемник”.

7.7 Програма за функционалност на контролера, предавател:

За реализирането на контролера се използват Arduino Uno, nRF24L01 безжичен модул и два джойстика KY-023. Двата джойстика имат следните функции: 1) При натискане на първия джойстик водата започва да тече от съда през маркуча. След това можем отново да натиснем бутона, за да спрем този процес. Възможно е и да движим джойстика, който има функциите: Forward (напред), Backward (назад), Left (наляво) и Right (надясно). Захранва се с 5V от микроконтролера. Когато движим джойстика, стойностите се променят от 0 до 1023, в зависимост от позицията му, поради което VRX и VRY пиновете на джойстика са свързани към аналогови пинове на ардуиното. SW пинът отговаря за това дали е натиснат джойстика, или не, и е свързан към цифров пин на ардуиното; 2) Вторият джойстик има функциите: Up (нагоре), Down (надолу), Turn Left (завъртане наляво) и Turn Right (Завъртане надясно). Свързването на пиновете съответства на правилата, казани за първия джойстик. След това имаме Arduino Uno, за което са свързани джойстиците и безжичният модул nRF24L01. Последният елемент от електрическата схема е единият nRF24L01 модул от проекта. Той използва SPI интерфейс и се управлява с помощта на команди, изпращани чрез този интерфейс. За използване на модула, изводите му са свързани към микроконтролера, за да се инициализира SPI интерфейса и да се изпратят нужните команди. За да се предават данни с модула, трябва да се настройт двата модула - един като предавател и един като получател. В случая nRF24L01 модулът е настроен като предавател. Той изпраща данни на определен адрес, който получателят е настроен да слуша. На фиг. 7.7 е представена електрическата схема на контролера:



Фиг. 7.7 Електрическа схема на контролера

Целият код на контролера може да бъде намерен в GitHub хранилището, файл “Transmitter.ino”, но сега той ще бъде разделен на отделни части с обяснение на всяка една от тях.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

// Motor Control Joystick Pins
const int joystickX = A0;
const int joystickY = A1;

// Servo Control Joystick Pins
const int servoJoystickX = A2;
const int servoJoystickY = A3;

// Water Pump Control Joystick Pin
const int pumpControlPin = 2;
// Connect to the first joystick pin

bool pumpOn = false;
// Variable to track pump state
```

имаме един и същ адрес както на приемника, така и на предавателя. Променливите “joystickX”, “joystickY” са аналогови изводи, свързани към първия джойстик за управление от разстояние на DC моторите. “servoJoystickX”, “servoJoystickY” променливите също са аналогови изводи, свързани към втория джойстик за управление от разстояние на серво моторите за механизма за маркуча. Променлива “pumpControlPin” е цифров извод, свързан към първия джойстик и неговия SW бутон за управление на пускане/спиране на помпата. “pumpOn” е булева променлива за проследяване на състоянието на водната помпа, по подразбиране е зададена на “false”.

```
void setup() {

radio.begin();
radio.openWritingPipe(address);
radio.setPALevel(RF24_PA_LOW);
radio.stopListening();
Serial.begin(9600);

pinMode(pumpControlPin, INPUT_PULLUP);
// Set the pin to input with internal pull-up resistor
}
```

Библиотеки:

- 1) SPI.h - тази библиотека позволява на Arduino да комуникира чрез SPI протокола;
- 2) nRF24L01.h - тази библиотека предоставя необходимите функции за взаимодействие с модула nRF24L01;
- 3) RF24.h - тази библиотека опростява използването на безжичния модул nRF24L01;

След това се създава RF24 обект, наречен radio, с извод 7 за CE (Chip Enable) и извод 8 за CSN (Chip Select Not).

После се създава масив от байтове, който представлява адресът, чрез който двата модула ще комуникират. Може да се промени стойността на този адрес на произволен низ от 5 букви и това ни позволява да изберем към кой приемник ще говорим, така че в нашия случай ще

следва задължителната функция setup(), която инициализира първоначалните стойности на изводите. “radio.begin()” инициализира nRF24L01 модула. С помощта на функцията radio.openWritingPipe() се задава

адресът за запис на данни към приемника, на който ще се изпращат данни, 5-буквеният низ, който е зададен преди това. След това функцията radio.setPAlevel() настройва нивото на усилвателя на мощност, в нашия случай е настроен на минимум, за да се пести енергия. Функцията radio.stopListening() пък задава модула като предавател. “Serial.begin(9600)” инициализира серийна комуникация за целите на отстраняване на грешки. Накрая “pinMode(pumpControlPin, INPUT_PULLUP)” задава извода за управление на помпата като вход (INPUT) с вътрешен pull-up резистор.

```

void loop() {
    int xValue = analogRead(joystickX);
    int yValue = analogRead(joystickY);

    int xValueServo = analogRead(servoJoystickX);
    int yValueServo = analogRead(servoJoystickY);

    // Read the state of the pump control joystick
    bool pumpButtonPressed = digitalRead(pumpControlPin) == LOW;

    // Create data packet
    int data[5] = {xValue, yValue, xValueServo, yValueServo, pumpButtonPressed};

    // Send data packet
    radio.write(&data, sizeof(data));
}

```

След създаването на функция setup(), която инициализира и задава първоначалните стойности, функцията loop() прави точно това, което подсказва името ѝ, и повтаря последователно, позволявайки на програмата да се промени и да реагира. Променливите “xValue = analogRead(joystickX)” и ”yValue = analogRead(joystickY)” четат аналоговите стойности от джойстика за управление на DC моторите. Аналогично, променливите “xValueServo = analogRead(servoJoystickX)”, както и “yValueServo = analogRead(servoJoystickY)” четат аналоговите стойности от джойстика за управление на двата серво мотора. Булевата променлива “pumpButtonPressed = digitalRead(pumpControlPin) == LOW” проверява дали бутона на първия джойстик за управление на помпата е натиснат (Active Low). “int data[5] = {xValue, yValue, xValueServo, yValueServo, pumpButtonPressed}” създава масив за съхранение на стойностите на джойстика и състоянието на бутона за водната помпа (пусната/спряна). С помощта на функцията radio.write(&data, sizeof(data)) се изпраща масивът от данни към приемника. Първият аргумент е променливата, която трябва се изпрати. Използвайки „&“ преди името на променливата, всъщност се задава индикация на променливата, която съхранява

дannите, които трябва да бъдат изпратени, и използвайки втория аргумент, се задава броят байтове, които трябва да се вземат от променливата. В този случай функцията `sizeof()` получава всички байтове от „`data`“.

```
// Print joystick readings to serial monitor
Serial.print("Motor Control - X Value: ");
Serial.print(xValue);
Serial.print("\tY Value: ");
Serial.println(yValue);

Serial.print("Servo Control - X Value: ");
Serial.print(xValueServo);
Serial.print("\tY Value: ");
Serial.println(yValueServo);

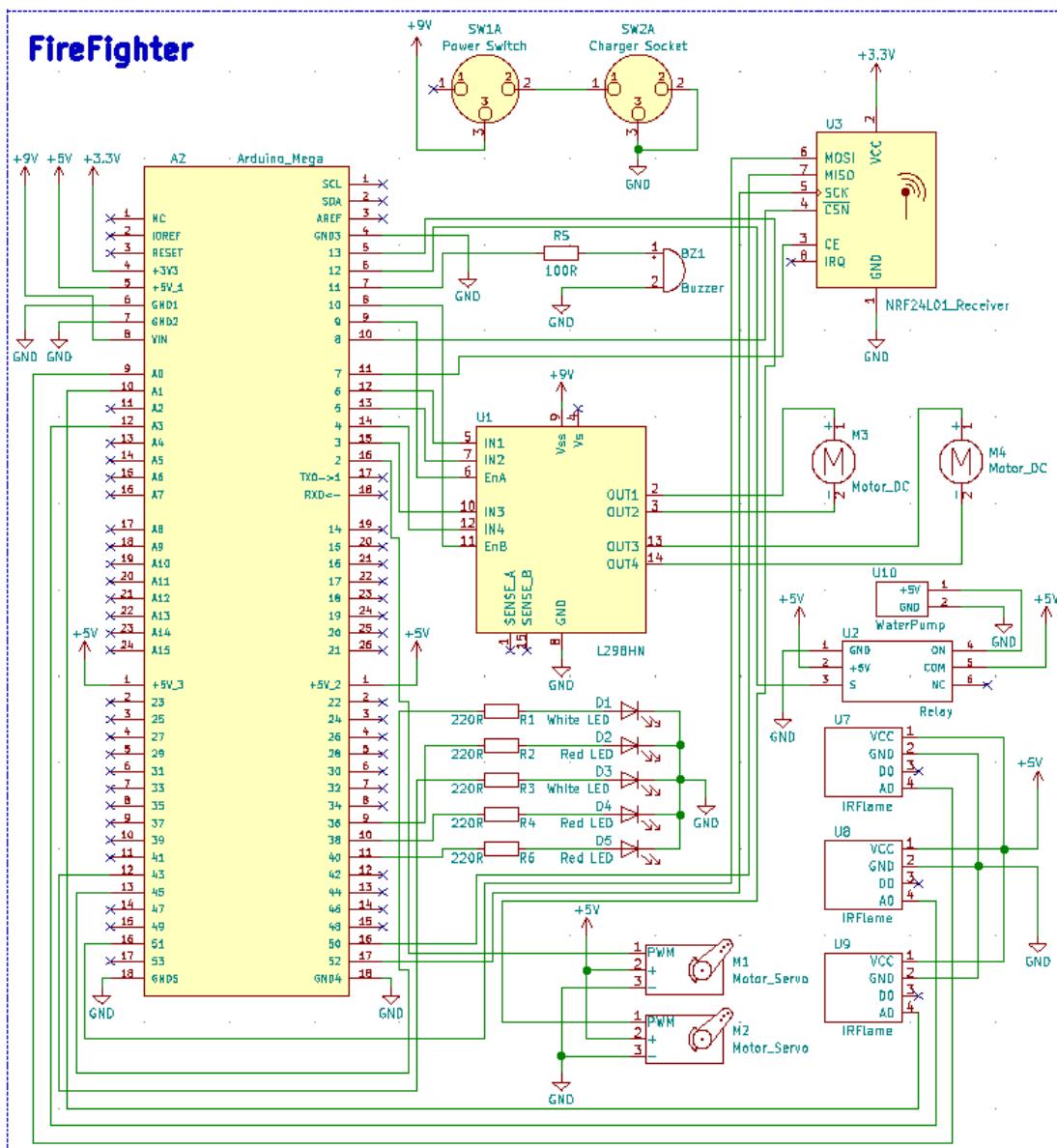
delay(100); // Adjust delay as needed for communication stability
}
```

“`Serial.print`” отпечатва стойностите на джойстика на серийния монитор за отстраняване на грешки. “`delay(100)`” настройва кратко забавяне (100ms) за осигуряване на стабилна и не твърде бърза комуникация.

7.8 Програма за функционалност на робота, приемник:

За реализирането на пожарникарския робот се използват Arduino Mega, един nRF24L01 безжичен модул, един L298N драйвер, един зумер, едно реле, два серво мотора, три сензора за пламък, една водна помпа, два DC мотора, две гнезда за батерии, един превключвател за пускане/спиране, 5 светодиода и 5 резистора. Двата бели светодиода имат ролята на фарове, а двата червени - на стопове. Използват се последователно свързани резистори към тях, защото те ограничават тока, който преминава през светодиодите. Превключвателят има за цел да пуска и да спира цялата система, докато шест AA батерии помогнат за подаването на захранване на цялата пожарна машина. Второто гнездо за батерии, с три AA батерии в него, служи за захранване на водната помпа, работеща на 5V. NRF24L01 безжичният модул тук е настроен като приемник. Той получава данни на определен адрес, на който е настроен да слуша. Чрез двета DC мотора, работещи на 9-12V, се задвижва тялото на пожарната машина с определена скорост. Използван е H-bridge (L298N), който контролира посоката на въртене и на двета DC мотора (свързани към задвижващи колела). Едно реле (KY-019) служи за включване/изключване на водната помпа. Използват се общо три сензора за пламък, чиято функция е да засекат пламъци и да

изпратят команда за спиране на автомобила на подходящо разстояние от пламъка/огъня, така че в същия момент да се стартира и водната помпа, която да потуши огъня. Когато сензорите засекат информация, тогава те изпращат сигнал чрез зумер, който издава писклив звук и известява потребителя за наличие на пламък или на пожар. За целия механизъм на водния пистолет се използват един серво и един микро серво мотор. Тези мотори могат да осигурят относително прецизно движение в определен диапазон, както и консумират малко енергия. Серво моторът служи за задвижване на водния пистолет наляво или надясно, докато микро серво моторът мести маркуча нагоре или надолу. На фиг. 7.8 е представена електрическата схема на контролера:



Фиг. 7.8 Електрическа схема на пожарникарския робот

Целият код на контролера може да бъде намерен в GitHub хранилището, файл “Receiver.ino”, но сега той ще бъде разделен на отделни части с обяснение на всяка една от тях.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

// Motor A Pins
const int ENA_PIN = 9; // EN1 pin of L298N
const int IN1_PIN = 6; // IN1 pin of L298N
const int IN2_PIN = 5; // IN2 pin of L298N

// Motor B Pins
const int ENB_PIN = 10; // EN2 pin of L298N
const int IN3_PIN = 3; // IN3 pin of L298N
const int IN4_PIN = 4; // IN4 pin of L298N
```

След това се създава RF24 обект, наречен radio, с извод 7 за CE (Chip Enable) и извод 8 за CSN (Chip Select Not). После се създава масив от байтове, който представлява адресът, чрез който двата модула ще комуникират. В случая имаме един и същ адрес както на приемника, така и на предавателя.

Следват имена на променливи, съответстващи на имената на изводите на L298N драйвера, които са свързани към определени изводи на микроконтролера Arduino Mega за управление на двата DC мотора. ENA и ENB изводите на L298N драйвера се използват чрез премахване на джъмпера върху тях и служат за контролиране на скоростта на DC моторите; трябва да се свържат към PWM изводите на Arduino. Логическите изводи на драйвера (IN1, IN2, IN3, IN4) се свързват към цифровите изводи на Arduino, които помагат за контролиране на въртенето и скоростта на DC моторите.

```
// Buzzer Pin
const int BUZZER_PIN = 11;

// LED Pins
const int LED1_PIN = 45;
const int LED2_PIN = 36;
const int LED3_PIN = 43;
const int LED4_PIN = 38;
const int LED5_PIN = 40;

// Water Pump Relay Pin
const int PUMP_RELAY_PIN = 12;
// Connect to the relay control pin

// Current speed of motors
int currentSpeed = 200;

// Flame sensor Pin
const int FLAME_SENSOR_PIN_1 = A0;
const int FLAME_SENSOR_PIN_2 = A3;
const int FLAME_SENSOR_PIN_3 = A1;

// Servo motor pins
const int SERVO_PIN_X = 2; // X-axis servo
const int SERVO_PIN_Y = 13; // Y-axis servo

Servo servoX;
Servo servoY;

bool pumpOn = false;
bool fireDetected = false;
```

Библиотеки:

- 1) SPI.h - тази библиотека позволява на Arduino да комуникира чрез SPI протокола;
- 2) nRF24L01.h - тази библиотека предоставя необходимите функции за взаимодействие с модула nRF24L01;
- 3) RF24.h - тази библиотека опростява използването на безжичния модул nRF24L01;
- 4) Servo.h - служи за управление на серво мотори.

След това се дефинират номерата на изводите на Arduino, към които е свързан зумерът (11), както и LED светодиодите (36, 38, 40, 43, 45). Определя се и номерът на извода на микроконтролера (12), който управлява релето, свързано към водната помпа. Редът “currentSpeed = 200” дефинира променлива, която съдържа скоростта за моторите (0 - 255), настроена е на 200. Скоростта се използва в логиката за управление на моторите, за да се определи колко бързо трябва да работят те. След това се определят аналоговите входни изводи на главната платка (A0, A1, A3), към които са свързани сензорите за пламък. После се дефинират номерата на изводите за серво моторите, управляващи осите X (SERVO_PIN_X = 2) и Y (SERVO_PIN_Y = 13) и се създават Servo обекти (servoX, servoY) за серво моторите по оста X и Y. Накрая има две булеви променливи, които се използват като флагове за проследяване на състоянията на водната помпа и откриването на пожар. “pumpOn” показва дали водната помпа в момента е включена или изключена. Превключва се въз основа на състоянието на натискане на бутон, получено чрез модула nRF24L01. “fireDetected” пък показва дали е открит пожар от някой от сензорите за пламък. Помага при контролиране на състоянието на светодиода и зумера за предупреждения за пожар. По подразбиране са зададени на “false”.

```

void setup() {
    Serial.begin(9600);

    pinMode(ENA_PIN, OUTPUT);
    pinMode(IN1_PIN, OUTPUT);
    pinMode(IN2_PIN, OUTPUT);
    pinMode(ENB_PIN, OUTPUT);
    pinMode(IN3_PIN, OUTPUT);
    pinMode(IN4_PIN, OUTPUT);

    pinMode(BUZZER_PIN, OUTPUT);

    pinMode(LED1_PIN, OUTPUT);
    pinMode(LED2_PIN, OUTPUT);
    pinMode(LED3_PIN, OUTPUT);
    pinMode(LED4_PIN, OUTPUT);
    pinMode(LED5_PIN, OUTPUT);

    pinMode(FLAME_SENSOR_PIN_1, INPUT);
    pinMode(FLAME_SENSOR_PIN_2, INPUT);
    pinMode(FLAME_SENSOR_PIN_3, INPUT);
}

pinMode(PUMP_RELAY_PIN, OUTPUT);
// Set the pump relay pin as output

servoX.attach(SERVO_PIN_X);
servoY.attach(SERVO_PIN_Y);

radio.begin();
radio.openReadingPipe(0, address);
radio.setPALevel(RF24_PA_LOW);
radio.startListening();

// Turn on LED1 and LED3
digitalWrite(LED1_PIN, HIGH);
digitalWrite(LED3_PIN, HIGH);
}

```

В setup() функцията първоначално се инициализира сериината комуникация при скорост на предаване от 9600 бита в секунда. Това позволява на робота да изпраща съобщения за отстраняване на грешки чрез серииния монитор. Следва поредица от задаване на режимите на изводите на Arduino Mega за управление на DC моторите чрез L298N драйвера. Задаването на тези изводи като изходи

(OUTPUT) позволява на програмата да контролира скоростта и посоката на моторите. После се задава режимът на изводите на зумера и светодиодите като изходни (OUTPUT), като това позволява на програмата да включва или изключва зумера, осигурявайки звукови сигнали, както и да контролира светодиодите, осигурявайки визуална обратна връзка за различни състояния на робота (напр. посока на движение, откриване на пожар). След това се задават изводите на трите сензора за пламък като входни (INPUT), за да може програмата да чете аналоговите стойности от тях, откривайки наличието на пожар. После изводът на релето за водната помпа се настройва като изходен (OUTPUT), който позволява на програмата да контролира релето, като включва или изключва водната помпа в отговор на откриване на пожар или потребителски команди. Следващите две команди служат за закрепване (attach) на серво моторите към съответните им изводи (2 и 13). Следват няколко реда за nRF24L01 модула, който се инициализира чрез “radio.begin()”. Чрез функцията radio.setReadingPipe() се задава първоначалният адрес и по този начин се позволява комуникацията между двата модула. След това, с помощта на функцията radio.setPALevel(), се задава нивото на усилвателя на мощността, в случая е настроен на минимум. Накрая има функцията radio.startListening(), която задава модула като приемник. Последните два реда включват LED1 и LED3 (белите светодиоди - фарове) при стартиране на програмата.

```
void loop() {
    if (radio.available()) {
        int data[5];
        radio.read(&data, sizeof(data));

        int xValue = data[0];
        int yValue = data[1];
        int xValueServo = data[2];
        int yValueServo = data[3];
        bool pumpButtonPressed = data[4];

        // Reverse the joystick values
        xValue = 1023 - xValue;
        yValue = 1023 - yValue;
        // Print joystick readings to serial monitor
        Serial.print("Motor Control - X Value: ");
        Serial.println(xValue);
        Serial.print("\tY Value: ");
        Serial.println(yValue);
        Serial.print("Servo Control - X Value: ");
        Serial.println(xValueServo);
        Serial.print("\tY Value: ");
        Serial.println(yValueServo);
    }
}
```

В loop() функцията, използвайки функцията radio.available(), се проверява дали има данни за получаване. Ако това е вярно, първо се създава масив от 5 елемента, наречен „data“, в който се записват входящите данни. С помощта на функцията radio.read() се четат и съхраняват данните в променливата „data“. Използвайки „&“ преди името на променливата, всъщност се задава индикация на променливата, която съхранява данните, които трябва да бъдат изпратени, и използвайки втория аргумент, се задава броят байтове, които трябва да се вземат от променливата. В този случай

функцията `sizeof()` получава всички байтове от низа „`data`“. След това се извличат отделни стойности от получения масив от данни:

- 1) `xValue` - представлява стойността на оста X от джойстика, използвана за управление на посоката на движение на робота наляво/надясно;
- 2) `yValue` - представлява стойността на оста Y от джойстика, използвана за контролиране на скоростта на робота и движението напред/назад;
- 3) `xValueServo` - представлява стойността на оста x от джойстика за управление на посоката на първия серво мотора (насочване на водната помпа);
- 4) `yValueServo` - представлява стойността на оста y от джойстика за управление на посоката на втория серво мотора (насочване на водната помпа);
- 5) `pumpButtonPressed` - представлява състоянието на бутона на първия джойстик за управление, използван за включване или изключване на водната помпа.

След това се обръщат стойностите на джойстика за управление на DC моторите, получени от управлението от разстояние на робота. Това се прави, тъй като джойстикът е монтиран на обратно на основата на контролера, с цел симетричност с другия и по-красив външен вид. По този начин се постига съответствие на логиката за управление на DC моторите на робота. Следващите редове код отпечатват показанията на джойстика на серийния монитор, с цел отстраняване на грешки и наблюдение. Това е изключително полезно за отстраняване на неизправности и гарантиране, че системата работи по предназначение.

```
int mappedYSpeed = map(yValue, 0, 1023, -currentSpeed, currentSpeed);
int mappedXSpeed = map(xValue, 0, 1023, -currentSpeed, currentSpeed);
int mappedXServo = map(xValueServo, 0, 1023, 45, 135); // Map x-axis value to servo angle
int mappedYServo = map(yValueServo, 0, 1023, 45, 135); // Map y-axis value to servo angle
servoX.write(mappedXServo);
servoY.write(mappedYServo);

int motorSpeedA = mappedYSpeed;
int motorSpeedB = mappedYSpeed;
```

Функцията “`map`” се използва за преобразуване на стойностите на джойстика от първоначалния им диапазон (0 до 1023) в подходящия диапазон за скорости на DC моторите и ъглите на въртене на серво моторите:

1) “`map(yValue, 0, 1023, -currentSpeed, currentSpeed)`” - преобразува `yValue` (стойност на оста Y на джойстика) в стойност на скоростта за DC моторите. Диапазонът от 0 до 1023 се преобразува в `-currentSpeed` към `currentSpeed`, което позволява движение както напред, така и назад.

2) “`map(xValue, 0, 1023, -currentSpeed, currentSpeed)`” - преобразува `xValue` (стойността на оста X на джойстика) в стойност на скоростта на двигателите. По подобен начин диапазонът от 0 до 1023 се преобразува в `-currentSpeed` към

currentSpeed, което позволява движение наляво и надясно.

3) “map(xValueServo, 0, 1023, 45, 135)” - преобразува xValueServo (стойността на оста X на джойстика за серво мотора) в ъгъл на въртене за сервото по оста X. Диапазонът от 0 до 1023 е преобразуван на 45 до 135 градуса, съответстващ на физическия обхват на движение на сервото.

4) “map(yValueServo, 0, 1023, 45, 135)” - преобразува yValueServo (стойността на оста Y на джойстика за микро серво мотора) в ъгъл на въртене за сервото по оста Y. По подобен начин обхватът от 0 до 1023 се преобразува на 45 до 135 градуса.

Следват две команди, които задават позициите на серво механизмите, въз основа на преобразуваните стойности на джойстика.

- 1) “servoX.write(mappedXServo)” - настройва сервото по оста X на ъгъла на въртене, определен от mappedXServo.
- 2) “servoY.write(mappedYServo)” - настройва сервото по оста Y на ъгъла на въртене, определен от mappedYServo.

Накрая се инициализират променливите за скоростта на DC моторите.

```
if (xValue < 400) {  
    motorSpeedA = 0;  
    motorSpeedB += mappedXSpeed;  
}  
else if (xValue > 600)  
{  
    motorSpeedB = 0;  
    motorSpeedA -= mappedXSpeed;  
  
}  
  
analogWrite(ENA_PIN, abs(motorSpeedA));  
analogWrite(ENB_PIN, abs(motorSpeedB));
```

Следва блокът от код, който служи за въртене на робота наляво и надясно, при зададена команда от джойстика. Проверява се дали стойността на оста X на джойстика е по-малка от 400, което показва значителна лява позиция. “motorSpeedA = 0” задава скоростта на мотор A на 0, спирачи го, а после се увеличава скоростта на мотор B чрез “mappedXSpeed”, катайки робота да завие наляво, тъй като само мотор B е активен.

Аналогично, след това се проверява дали стойността на оста x на джойстика е по-голяма от 600, което показва значителна дясна позиция. “motorSpeedB = 0” задава скоростта на мотор B на 0, спирачи го, а след това “motorSpeedA -= mappedXSpeed” намалява скоростта на мотор A чрез mappedXSpeed, катайки робота да завие надясно, тъй като само мотор A е активен.

Последните два реда прилагат изчислените скорости на моторите към тях, използвайки ШИМ (PWM). “analogWrite(ENA_PIN, abs(motorSpeedA))” - изпраща PWM сигнал до ENA_PIN с работен цикъл, съответстващ на абсолютната стойност на motorSpeedA. Функцията abs гарантира, че скоростта е неотрицателна, тъй като стойностите на ШИМ (PWM) трябва да са между 0 и 255. “analogWrite(ENB_PIN,

`abs(motorSpeedB))”` - изпраща PWM сигнал към ENB_PIN с работен цикъл, съответстващ на абсолютната стойност на motorSpeedB.

```

if (motorSpeedA > 0)
{
    digitalWrite(IN1_PIN, HIGH); // Forward direction
    digitalWrite(IN2_PIN, LOW);
    // Turn on LED4 and LED5 when moving backward
    digitalWrite(LED4_PIN, HIGH);
    digitalWrite(LED5_PIN, HIGH);
}

else
{
    digitalWrite(IN1_PIN, LOW); // Backward direction
    digitalWrite(IN2_PIN, HIGH);

    digitalWrite(LED4_PIN, LOW);
    digitalWrite(LED5_PIN, LOW);
}

```

```

if (motorSpeedB > 0)
{
    digitalWrite(IN3_PIN, HIGH); // Forward direction
    digitalWrite(IN4_PIN, LOW);

    digitalWrite(LED4_PIN, HIGH);
    digitalWrite(LED5_PIN, HIGH);
}

else
{
    digitalWrite(IN3_PIN, LOW); // Backward direction
    digitalWrite(IN4_PIN, HIGH);

    digitalWrite(LED4_PIN, LOW);
    digitalWrite(LED5_PIN, LOW);
}

```

Първият блок код задава посоката на въртене за мотор А, въз основа на неговата скорост, зададена по-рано от джойстика, като също така включва или изключва двата червени светодиода, които служат като индикатори. Проверява се дали скоростта на мотор А е положителна, което показва движение напред. Задава се “IN1_PIN” на “HIGH”, за да управлява H-Bridge драйвера, карайки мотор А да се движи напред, а пък “IN2_PIN” се задава на “LOW”, което е необходимо за да се потвърди движение напред, а не назад, на същия мотор. Накрая се пускат двата червени светодиода, с цел индикация. В противен случай, ако “motorSpeedA” не е положителна (т.е. нула или отрицателна), това означава спиране или движение назад на мотора. Задава се “IN1_PIN” на “LOW”, с цел потвърждение за движение назад на мотор А, а пък “IN2_PIN” се задава на “HIGH”, за да накара мотор А да се движи назад. Накрая се изключват двата червени светодиода, поради движението назад на мотора.

Подобно на този за мотор А, вторият блок код контролира посоката на мотор В и логиката на червените светодиоди, въз основа на неговата скорост. Проверява се дали скоростта на мотор В е положителна, което показва движение напред. Задава се “IN3_PIN” на HIGH, за да управлява H-Bridge драйвера, карайки мотор В да се движи напред, а “IN4_PIN” се настройва на “LOW”, необходимо за да се потвърди движение напред, а не назад, на същия мотор. Накрая се пускат двата червени светодиода, с цел индикация. В противен случай, ако “motorSpeedB” не е положителна, това означава спиране или движение назад на мотор В. Задава се “IN3_PIN” на “LOW”, с цел потвърждение за движение назад на мотор В, а “IN4_PIN” е настроен на “HIGH”, за да накара мотор В да се движи назад. Накрая се изключват двата червени светодиода, поради движението назад на мотора.

```

// Control the water pump relay
if (pumpButtonPressed) {
    if (!pumpOn) {
        digitalWrite(PUMP_RELAY_PIN, HIGH); // Turn on the water pump
        pumpOn = true;
    } else {
        digitalWrite(PUMP_RELAY_PIN, LOW); // Turn off the water pump
        pumpOn = false;
    }
}
}

```

Това е блокът код, който служи за пускане на водната помпа, когато някой от трите сензора за пламък засече огън. Първо се проверява дали променливата “pumpButtonPressed” е вярна, т.е. дали първият джойстик е натиснат. “pumpOn” е булева променлива, която следи дали водната помпа в момента е включена или изключена. “!pumpOn” проверява дали променливата pumpOn е true, т.е. дали водната помпа е пусната. Ако това е изпълнено, “digitalWrite(PUMP_RELAY_PIN, HIGH)” - задава “PUMP_RELAY_PIN” (12) на “HIGH”, което включва релето, управляващо водната помпа. “pumpOn = true” - актуализира променливата pumpOn до “true”, за да покаже, че помпата вече е включена. Това гарантира, че при следващото натискане на бутона помпата може да бъде изключена.

Другата част от условието се изпълнява, ако “pumpOn” е вярно, което означава, че помпата в момента е включена. Тогава “digitalWrite(PUMP_RELAY_PIN, LOW)” - задава “PUMP_RELAY_PIN” (12) на “LOW”, което изключва релето, управляващо водната помпа. “pumpOn = false” - актуализира променливата “pumpOn” до “false”, за да покаже, че помпата вече е изключена. Това гарантира, че при следващото натискане на бутона помпата може да се включи.

```

// Check flame sensor 1
int flameSensorValue1 = analogRead(FLAME_SENSOR_PIN_1);
// Check flame sensor 2
int flameSensorValue2 = analogRead(FLAME_SENSOR_PIN_2);
// Check flame sensor 3
int flameSensorValue3 = analogRead(FLAME_SENSOR_PIN_3);

```

Предпоследният блок код чете стойностите, засечени от трите сензора за пламък. Функцията “analogRead” чете стойността от посочения аналогов извод, в този случай “FLAME_SENSOR_PIN_1” (A0), “FLAME_SENSOR_PIN_2” (A3) и “FLAME_SENSOR_PIN_3” (A1). След това същата функция връща цяло число, представляващо показанието на сензора. Тази стойност се съхранява в променливата “flameSensorValue1”, “flameSensorValue2” и “flameSensorValue3”.

```
// Check if any flame sensor detects fire
if (flameSensorValue1 <= 500 || flameSensorValue2 <= 500 || flameSensorValue3 <= 500)
{
    // Fire detected
    digitalWrite(LED2_PIN, HIGH);
    tone(BUZZER_PIN, 1000); // Make a constant sound
    fireDetected = true;
}
else if (fireDetected)
{
    // Fire extinguished
    digitalWrite(LED2_PIN, LOW);
    noTone(BUZZER_PIN);
    fireDetected = false;
}
}
```

Последният блок код е отговорен за откриването на пожар въз основа на показанията от сензорите за пламък и съответното реагиране. Първият оператор if проверява дали някой от сензорите за пламък (flameSensorValue1, flameSensorValue2, flameSensorValue3) има стойност, по-малка или равна на 500. 500 е прагова стойност и е избрана, за да покаже наличието на пламък. Стойности под или равни на този праг предполагат, че сензорът е открил инфрачервена светлина, вероятно от пламък.

Мерките, които се вземат, са: включва се червеният светодиодът, сигнализиратки визуално, че е открит пожар; генерира се непрекъснат звук от зумера, с честота от 1000Hz, осигурявайки звуков сигнал; булевата променлива “fireDetected” се задава на “true”, която показва, че е открит пожар.

След това се проверява дали “fireDetected” е “false” (по подразбиране), което означава, че пожар е бил открит преди това, но вече не е налице. В такъв случай се изключват както светодиодът, така и зумерът. Булевата променлива “fireDetected” се задава на “false”, което показва, че в момента не е открит пожар.

Това е програмата, качена на противопожарния робот. Той може да се управлява безжично в различни посоки, сензорите за пламък засичат огън, в същия момент може да се пусне водната помпа, която да потуши пламъците. Маркучът, прикрепен към водната помпа, може да бъде насочван в различни посоки, с цел ефективно и пълноценно спиране на пожара.

ГЛАВА 8. Проблеми и бъдещо развитие на проекта

8.1 Проблеми, възникнали по време на разработването на проекта:

В процеса на разработка на противопожарния робот се сблъскахме с редица предизвикателства, които изискваха внимание и нови решения. От технически затруднения със сензорите и компонентите, до проблеми с комуникационните модули и захранването, всяка стъпка от проекта ни изправяше пред нови проблеми, които трябваше да бъдат преодолени. В този раздел ще бъдат разгледани основните проблеми, възникнали по време на разработването на проекта, и методите, които са използвани за тяхното разрешаване.

8.1.1 Захранване:

Един от основните проблеми, с които се сблъскахме по време на разработката на противопожарния робот, беше това как ще бъдат захранвани всички отделни компоненти на робота. Макар че Arduino Mega има три 5V извода, те се оказаха недостатъчни за всички елементи (сензори за пламък, серво мотори, модули и др.). Освен това, нямаше място във вътрешността на робота за поставяне на макетна платка (breadboard) за разпределение на захранването.

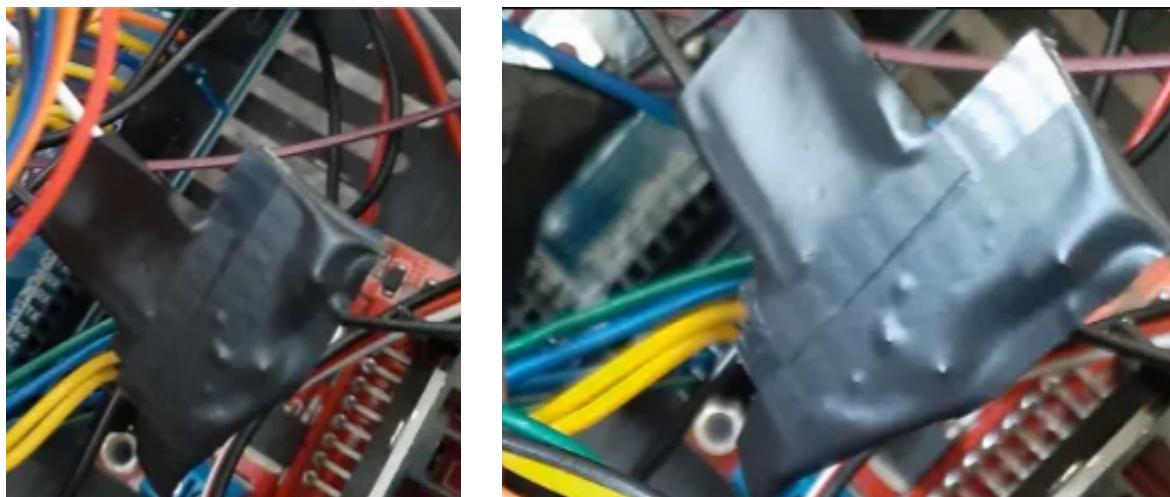
За да решим този проблем, използвахме женска рейка, която ни позволи да разпределим 5V захранване към всички необходими компоненти. На фиг. 8.1 е представена снимка на женска рейка:



Фиг. 8.1 Женска рейка

Първо свързахме главния 5V източник от микроконтролера към входа на захранващата рейка. Това осигури стабилно 5V захранване, което може да се разпределя към другите компоненти. Наложи се да използваме и втора такава рейка, която е за GND връзките. Свързахме повечето сензори за пламък, серво мотори,

модули и др. към рейките, като се уверихме, че всяка връзка е стабилна и надеждна. Сложихме и изолирбанд, за да се осигури безопасност от късо съединение и др. След като всички компоненти бяха свързани, проверихме всяка връзка за стабилност и сигурност. Използвахме мултициет, за да се уверим, че всички компоненти получават точно 5V. Включихме системата и проверихме дали всички компоненти работят правилно. Коригирахме всички несъответствия и осигурихме стабилна работа на захранващата система. На фиг. 8.2 са представени снимки на рейките, предпазени с изолирбанд:



Фиг. 8.2 Женски рейки, използвани в проекта

8.1.2 Запояване на отделни компоненти:

Запояването на отделни компоненти бе критичен етап в изграждането на противопожарния робот. Този процес осигури стабилни и надеждни електрически връзки между различните елементи на робота, но основните проблеми идваха именно от спойките. Наложи се запояване на проводници към + и - на DC моторите, последователно свързване на резистори към светодиоди и към зумера, връзката между адаптера за захранване и превключвателя за пускане/спиране на цялата машина, както и връзката на късо между отделните изводи на женските рейки. Като за начало оголихме краищата на проводниците с помощта на ножица за кабели. След запояването им, оставихме спойките да се охладят и проверихме връзката за здравина и добро електрическо съединение. На око изглеждаше, че всичко е наред. За съжаление, често се случваше спойките да се отлепват, или проводниците да се скъсват, поради прекомерно огъване и дърпане, което изискваше повторно запояване и внимателно закрепване на проводниците. В крайна сметка, използвахме отново мултициет, за да проверим електрическата проводимост на всяка спойка и да се уверим, че всички компоненти са правилно свързани и работят както трябва.

8.1.3 Комуникация:

Установяването на надеждна и правилна комуникация между двата nRF24L01 модула се оказа може би най-предизвикателната задача и изискваше значителни усилия и време. Ето по-подробно описание на проблемите и стъпките, през които преминахме. Имахме в наличност един нормален nRF24L01 приемо-предавателен модул, както и един с антена. На фиг. 8.3 са представени двата модула:



Фиг. 8.3 nRF24L01 безжични модули

Като за начало подготвихме двата nRF24L01 модула, Arduino платки, захранващи източници и свързващи кабели. Използвахме Arduino IDE за програмиране и тестване на модулите. Както е логично, искахме модулът с антена да бъде монтиран към контролера за управление на робота, т.е. той да бъде настроен като предавател. По план нормалният nRF24L01 модул трябваше да бъде монтиран към робота и да бъде настроен като приемник. Уверихме се, че всички връзки са правилно направени и няма хлабави конектори. Когато качихме кода на двете Arduino платки се оказа че модулите не могат да установят комуникация. Пробахме най-различни варианти, сменихме всички проводници, търсихме програми от различни сайтове, но без успех. След четири дни мъка се замислихме какво ще стане ако разменим местата на двата модула. След като настроихме нормалния безжичен модул да бъде предавател, а този с антената - приемник, те успяха да си комуникират. Ето защо във финалната фаза на проекта на контролера е монтиран нормалният nRF24L01 модул, а на робота - този с антената. Всичко работеше нормално и продължихме с работата по другите компоненти за реализацията на проекта. След известно време комуникацията отново спря да работи. След оглед и проверка на свързаността се оказа, че един от модулите е бил захранен с 5V, което е над максималното позволено напрежение за nRF24L01 (3.3V). Това доведе до изгаряне на приемника. След подмяната на модула и правилното му захранване, проведохме серия от тестове, които потвърдиха, че комуникацията между двата nRF24L01 модула функционира правилно.

Въпреки всички тези трудности, ние успяхме да решим проблемите и да установим стабилна безжична връзка, както и като цяло един функциониращ правилно проект.

8.2 Бъдещо развитие на проекта:

Проектът “Изграждане на противопожарен робот, управляем с помощта на радио контролер” има значителен потенциал за бъдещо развитие, като някои от най-важните подобрения, които предвиждаме, включват интеграцията на камера и LCD дисплей. Тези допълнения ще увеличат функционалността и ефективността на робота, предоставяйки по-добра информированост и контрол на потребителите.

8.2.1 Интеграция на камера:

Планираме да монтираме камера върху робота, която ще има за цел да заснема и предава изображения и видео в реално време. Тази функция ще бъде особено полезна, когато роботът влиза в сгради или други затворени пространства, където видимостта може да бъде ограничена поради дим или други препятствия. Камерата ще позволи на потребителите да наблюдават обстановката и да оценяват ситуацията от безопасно разстояние, предоставяйки им визуална информация за евентуални пожари или други опасности. На фиг. 8.4 е представена примерна камера, която може да бъде използвана за бъдещата реализация на проекта:

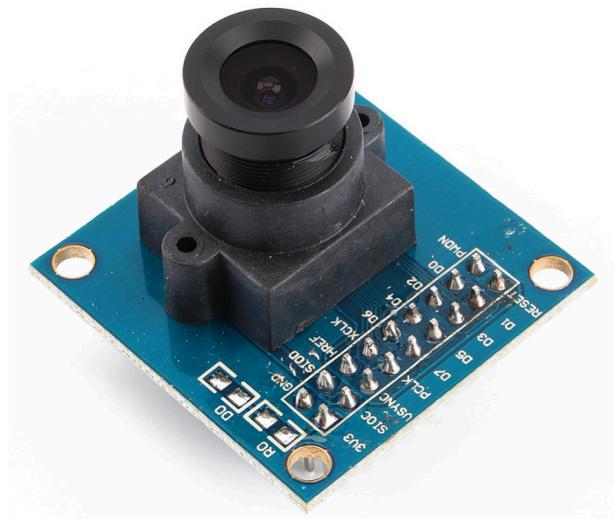
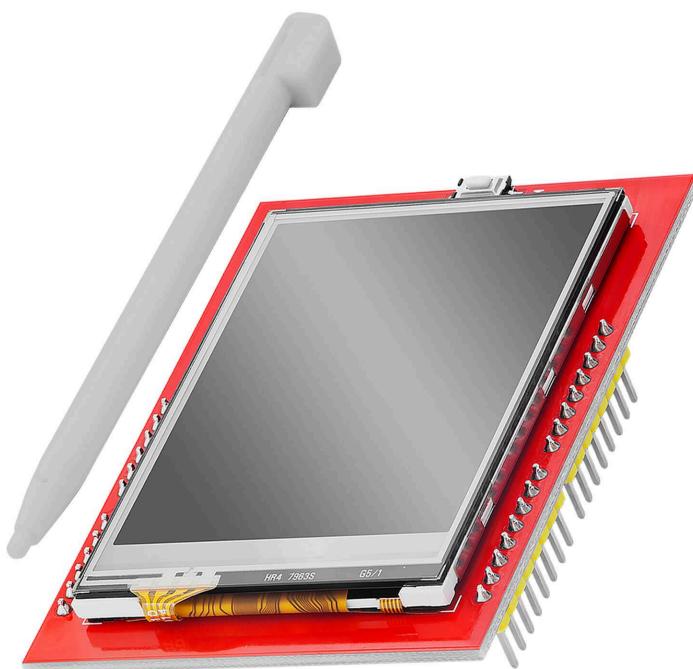


Fig. 8.4 Arduino Camera

Arduino камерата, известна също като OV7670, е цифрова камера сензор, която може да заснема изображения и видео с резолюция от 640x480 пиксела. Тя е съвместима с различни микроконтролери и може да се използва в разнообразни проекти за наблюдение и изображения.

8.2.2 Интеграция на LCD дисплей:

В допълнение към камерата, ще интегрираме TFT LCD (Thin Film Transistor Liquid Crystal Display) дисплей към контролера на робота. Този дисплей ще служи за показване на информация в реално време, като кадри от камерата, данни от сензорите и други важни съобщения. Когато роботът засече огън, на дисплея ще се изписва съобщение, напр. "Fire Detected!!!", което ще предупреждава потребителя за наличието на пожар и ще му позволява да предприеме необходимите мерки бързо и ефективно. На фиг. 8.5 е представена примерен TFT LCD дисплей, който може да бъде използван за бъдещата реализация на проекта:



Фиг. 8.5 TFT LCD Display

8.2.3 Обобщение:

Тези подобрения ще повишат общата функционалност и ефективност на робота, като ще предоставят по-голяма информираност и контрол на потребителите. Със способността да наблюдават обстановката в реално време и да получават незабавни предупреждения за пожари, потребителите ще могат да реагират по-бързо и по-информирано, намалявайки риска за хората и имуществото. Тези нововъведения ще направят робота още по-ценен инструмент в борбата с пожари, особено в труднодостъпни и опасни зони. С нетърпение очакваме да реализираме тези функции и да продължим да подобряваме нашия проект в полза на безопасността при борбата с пожари.

Заключение

В курсовата работа е разгледано решение за изграждането на противопожарен робот. Той е изработен от огнеустойчив материал - алуминий, който е лек и издръжлив на удари и падания. Задвижващият механизъм осигурява достатъчен въртящ момент и мощност, а контролът на робота е подпомогнат от силен сигнал, осигуряващ подходящ обхват за радиоуправление. Проектът е предначен както за ентузиасти, така и за обикновени потребители, подходящ е за малки и средни предприятия, както и за граждани, които искат да го закупят.

Системата включва три интегрирани сензора за пламък, които гарантират надеждно откриване на пожари в различни посоки. Тези сензори предоставят точна информация за възникнал инцидент, като активират светодиод и зумер за звуков сигнал. Това осигурява бързо и ясно предупреждение за пожар, подпомагайки навременната реакция.

Захранването се осъществява с няколко АА батерии, осигуряващи достатъчно енергия за работата на пожарната машина. Комуникационният модул nRF24L01 е интегриран в системата, предоставяйки надеждна безжична връзка между контролера и пожарната кола. Този модул осигурява стабилен пренос на данни, което е от съществено значение за управлението на такива автономни устройства.

Системата включва и два джойстика, осигуряващи лесен безжичен достъп и контрол на робота, и които позволяват бързо и точно подаване на команди.

Използвана литература

- [1] - Микроконтролери. Архитектура и принцип на действие - Първо издание
- [2] - <https://www.instructables.com/Firefighter-Robot/>
- [3] - <https://techatronic.com/fire-fighter-robot-using-arduino-fire-fighting-robot/>
- [4] - <https://techxplore.com/news/2023-02-robot-firefighters-indoor-emergencies.html>
- [5] - <https://digitalcommons.cwu.edu/undergradproj/124/>
- [6] - <https://www.arduino.cc/>
- [7] - <https://store.arduino.cc/products/arduino-uno-rev3?queryID=undefined>
- [8] - <https://store.arduino.cc/products/arduino-mega-2560-rev3>
- [9] - <https://www.magneticinnovations.com/faq/dc-motor-how-it-works/>
- [10] - <https://www.tertiaryrobotics.com/micro-servo-motor-sg90.html>
- [11] - <http://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf>
- [12] - <https://www.elprocus.com/nrf24l01/>
- [13] - <https://components101.com/wireless/nrf24l01-pinout-features-datasheet>
- [14] - <https://element14.com/shop/flame-sensor-module/>
- [15] - <https://www.robotique.tech/robotics/control-a-water-pump-by-arduino/>
- [16] - <https://bg.wiringcable.com/info/what-is-buzzer-50608255.html>
- [17] - https://wiki.dfrobot.com/Devastator_Tank_Mobile_Platform_SKU_ROB0112
- [18] - <https://datasheetspdf.com/pdf/1402034/Joy-IT/KY-023/1>
- [19] - <https://uk.rs-online.com/web/content/ideas-and-advice/aa-batteries-guide>
- [20] - <https://batterycenter.bg/article/osnovni-pravila-za-upotreba-na-li-pol-baterii>
- [21] - <https://www.technika-bg.com/kakvo-e-rezistor/>
- [22] - <https://www.electronicrevolution.bg/bg-news-details-23.html>
- [23] - <https://www.circuitbread.com/ee-faq/what-is-a-pwm-signal>
- [24] - <https://www.electronicrevolution.bg/bg-news-details-47.html>
- [25] - <https://cdn.hackaday.io/Devastatoir%20Instruction%20Manual%20V2.pdf>
- [26] - <https://www.elprocus.com/arduino-relay/>

- [27] - <https://Running-DC-Motor-With-Arduino-Using-L298N-Motor-Dr/>
- [28] - <https://howtomechatronics.com/arduino-wireless-communication-nrf24l01/>
- [29] - <https://www.instructables.com/Flame-detection-using-Arduino/>
- [30] - <https://lab.arts.ac.uk/physical-computing/page/how-to-use-a-relay-module>
- [31] - <https://docs.arduino.cc/built-in-examples/basics/Blink/>

СЪДЪРЖАНИЕ

УВОД	2
ГЛАВА 1. Проблемът и конкуренцията	3
1.1 Описание на проблема:	3
1.2 Проучване на конкуренцията:	4
1.2.1 Firefighter Robot:	4
1.2.2 Fire fighting Robot:	6
1.2.3 Little robot that fight fires:	7
1.2.4 Fire Fighter Robot:	9
1.3 Нашето решение:	10
ГЛАВА 2. Блокова схема и концепция на проекта	11
2.1 Блокова схема на проекта:	11
2.2 Микроконтролер - Arduino:	12
2.2.1 Arduino Uno:	13
2.2.2 Arduino Mega:	13
2.3 Мотор:	14
2.3.1 DC мотор:	14
2.3.2 Servo мотор:	16
2.4 Драйвер:	17
2.5 Безжичен RF модул:	17
2.6 Сензор за пламък:	18
2.7 Водна помпа:	19
2.8 Зумер:	19
2.9 Реле:	20
2.10 Тяло на пожарната машина:	21
2.11 Съд с вода и маркуч:	21
2.12 Джойстик:	22
2.13 Захранване:	23
2.13.1 AA батерия:	23
2.13.2 LiPo батерия:	23
2.14 Други електронни елементи:	24
2.14.1 Резистор:	24
2.14.2 Светодиод:	25
ГЛАВА 3. Принципна електрическа схема	26
3.1 Електрическа схема на радиоконтролера за управление:	26
3.2 Електрическа схема на пожарната машина:	27

ГЛАВА 4. Проектиране и изграждане на хардуера	29
4.1 Реализация на корпуса на робота на проекта:	29
4.2 Реализация на корпуса на контролера на проекта:	31
4.3 Реализация на механизма за въртене на маркуча:	32
4.4 Равносметка и себестойност на проекта:	34
ГЛАВА 5. Логика на проекта и свързване на отделните хардуерни компоненти	35
5.1 Изграждане на контролера за управление на робота:	35
5.2 Изграждане на пожарникарския робот:	36
ГЛАВА 6. Терминология в Arduino. Комуникации в проекта	50
6.1 Въведение в АЦП и ЦАП:	50
6.2 Въведение в PWM:	50
6.3 Комуникация в проекта - SPI протокол:	51
6.4 Комуникация в проекта - ISM:	52
ГЛАВА 7. Софтуер на проекта	53
7.1 Примерна програма за движение на DC мотор в различни посоки:	53
7.2 Примерна програма за комуникация между два NRF24L01 модула:	56
7.3 Примерна програма за сензор за пламък, заедно със зумер и LED:	58
7.4 Примерна програма за водна помпа, управлявана с реле:	60
7.5 Примерна програма за мигане на светодиод:	61
7.6 Arduino IDE:	62
7.7 Програма за функционалност на контролера, предавател:	63
7.8 Програма за функционалност на робота, приемник:	66
ГЛАВА 8. Проблеми и бъдещо развитие на проекта	76
8.1 Проблеми, възникнали по време на разработването на проекта:	76
8.1.1 Захранване:	76
8.1.2 Запояване на отделни компоненти:	77
8.1.3 Комуникация:	78
8.2 Бъдещо развитие на проекта:	79
8.2.1 Интеграция на камера:	79
8.2.2 Интеграция на LCD дисплей:	80
8.2.3 Обобщение:	80
Заключение	81
Използвана литература	82
СЪДЪРЖАНИЕ	84