

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

**ДИПЛОМНА РАБОТА**

Тема: Управление на български робот РОБКО 01  
с ARM-базиран микроконтролер STM32L476RG  
и отдалечен достъп през интернет

Дипломант:

*Владимир Гаристов*

Научен ръководител:

*маг. инж. Росен Витанов*

**С О Ф И Я**

**2 0 1 8**





ТЕХНОЛОГИЧНО УЧИЛИЩЕ  
ЕЛЕКТРОННИ СИСТЕМИ  
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Дата на заданието: 06.11.2017 г.

Утвърждавам:.....

Дата на предаване: 06.02.2018 г.

/проф. д-р инж. Т. Василева/

## ЗАДАНИЕ за дипломна работа

на ученика Владимир Мариев Гаристов 12 В клас

1. Тема: Управление на робот Робко-01 с микроконтролер STM32L476RG и отдалечен достъп през интернет

2. Изисквания:

- Управляващ софтуер, изпълняван на микроконтролер STM32L476RG.
- Драйверна платка, разполагаща със захранващ блок и действаща като адаптер между РОБКО-01 и микроконтролера.
- Софтуер за отдалечено управление през интернет в две части: сървър и клиент.

3. Съдържание 3.1 Обзор

3.2 Същинска част

3.3 Приложение

Дипломант :.....

Ръководител:.....

/маг. инж. Росен Витанов/

Директор:.....

/ доц. д-р инж. Ст. Стефанова /



# Съдържание

1 РОБКО 01	9
1.1 История . . . . .	9
1.2 РОБКО 01 в настоящето . . . . .	13
1.3 Настояща разработка . . . . .	16
1.4 Тенденции в автоматизацията . . . . .	16
2 Система за автоматизирано отдалечено управление на РОБКО 01 с възможност за ръчен контрол	18
2.1 Функционални изисквания . . . . .	18
2.2 Топология на системата . . . . .	18
2.3 Компоненти на системата . . . . .	20
2.3.1 РОБКО 01 . . . . .	20
2.3.2 STM32L476RG Nucleo . . . . .	32
2.3.3 Основна платка и адаптери . . . . .	33
2.3.4 USB/UART модул . . . . .	33
2.3.5 Компютри и софтуер за отдалечено управление . . . . .	34
2.3.6 Джойстици . . . . .	35
2.3.7 Захранване . . . . .	38
3 Проектирани печатни платки	39
3.1 Принципна електрическа схема . . . . .	39
3.1.1 PATA адаптер и тестов адаптер . . . . .	39
3.1.2 Захранващ блок . . . . .	41
3.1.3 Връзка между STM32L476RG и РОБКО 01 . . . . .	43
3.1.4 Индикация . . . . .	43
3.1.5 Ръчно управление . . . . .	43
3.2 Графични оригинали на печатните платки . . . . .	46
3.2.1 Тестов адаптер . . . . .	46
3.2.2 PATA адаптер . . . . .	47
3.2.3 Основна платка . . . . .	49
4 Софтуер за отдалечено управление	54

4.1	Формат на командите за отдалечно управление . . . . .	54
4.2	Клиент . . . . .	56
4.2.1	TCP сокет . . . . .	57
4.2.2	Декодиране на команди . . . . .	57
4.3	Сървър . . . . .	57
4.3.1	Серийна комуникация . . . . .	59
4.3.2	TCP сокет . . . . .	59
4.3.3	Препращане и изпълнение на команди . . . . .	59
4.4	Наблюдение на TCP трафик . . . . .	60
5	Софтуер на микроконтролера	61
5.1	Основна функция . . . . .	61
5.2	Ръчно управление . . . . .	65
5.3	Получаване на команди през серийна комуникация . . . . .	65
5.4	Изпълнение на команди от опашката . . . . .	68
5.5	Управление на моторите . . . . .	70
5.6	Засичане на предмети с оптичен сензор . . . . .	70
6	Изработка на работоспособен модел	71
7	Заключение	73
7.1	Постигнати резултати . . . . .	73
7.2	Бъдещо развитие . . . . .	74
	Библиография	75

## Списък на фигурите

1.1	РОБКО 01 и Mini-mover 5 . . . . .	10
1.2	Различни модели от серията РОБКО . . . . .	11
1.3	Колекция от различни модели РОБКО и Правец 8М . . . . .	11
1.4	Аксесоари за РОБКО 01 . . . . .	12
1.5	Разширителна карта за ИМКО-2/Правец82, свързваща се с драйверната платка на РОБКО 01 . . . . .	13
1.6	Микроконтролерна платка, свързваща драйверната платка на РОБКО 01 с персонален компютър . . . . .	14

1.7	Платка, заменяща оригиналната драйверна платка на РОБКО 01 . . . . .	15
1.8	Пример за отдалечно управление на поточна линия . . . . .	17
2.1	Блокова схема на системата . . . . .	19
2.2	РОБКО 01 - схема на механиката . . . . .	21
2.3	Микроконтролерна платка STM32L476RG Nucleo . . . . .	32
2.4	USB/UART адаптерна платка . . . . .	34
2.5	Модифицирани джойстици за Правец 8 . . . . .	35
2.6	Захранващ източник . . . . .	38
3.1	Страна спойки . . . . .	46
3.2	Страна компоненти . . . . .	46
3.3	Ситопечат, страна спойки . . . . .	46
3.4	Ситопечат, страна компоненти . . . . .	46
3.5	Изолационен лак, страна спойки . . . . .	47
3.6	Изолационен лак, страна компоненти . . . . .	47
3.7	Очертание на печатната платка . . . . .	47
3.8	Отвори . . . . .	47
3.9	Страна компоненти . . . . .	47
3.10	Страна спойки . . . . .	47
3.11	Ситопечат, страна компоненти . . . . .	48
3.12	Ситопечат, страна спойки . . . . .	48
3.13	Изолационен лак, страна компоненти . . . . .	48
3.14	Изолационен лак, страна спойки . . . . .	48
3.15	Очертание на печатната платка . . . . .	48
3.16	Отвори . . . . .	48
3.17	Страна компоненти . . . . .	49
3.18	Страна спойки . . . . .	49
3.19	Ситопечат, страна компоненти . . . . .	50
3.20	Ситопечат, страна спойки . . . . .	50
3.21	Изолационен лак, страна компоненти . . . . .	51
3.22	Изолационен лак, страна спойки . . . . .	51
3.23	Припой, страна компоненти . . . . .	52
3.24	Очертание на печатната платка . . . . .	52
3.25	Отвори . . . . .	53
4.1	Блок схема на клиентската програма . . . . .	56
4.2	Блок схема на сървърната програма . . . . .	58
4.3	Примерна TCP сесия . . . . .	60

5.1	Блокова схема на функция main() . . . . .	62
5.2	Блок схема на функция read_cmd() . . . . .	67
5.3	Блок схема на функция remote_control() . . . . .	69
6.1	Тестов адаптер и микроконтролерна платка . . . . .	71
6.2	Основна платка . . . . .	71
6.3	Основна платка, микроконтролер и USB/UART модул . . . . .	72
6.4	Завършена система . . . . .	72
7.1	"На Кафе"с туесари . . . . .	73

## Списък на таблиците

2.1	Поредност на стъпките на моторите за въртене в режим на полуступка . . . . .	28
2.2	Адресно пространство на РОБКО 01 . . . . .	29
2.3	Съответствия между джойстици и движения на РОБКО 01 . .	36
4.1	Команди за отдалечно управление и техните параметри . .	55

Благодарности на  
 маг. инж. Константин Несторов,  
 маг. инж. Иван Учеджииев и  
 гл. ас. др. инж. Димитър Николов

# РОБКО 01

## 1.1 История

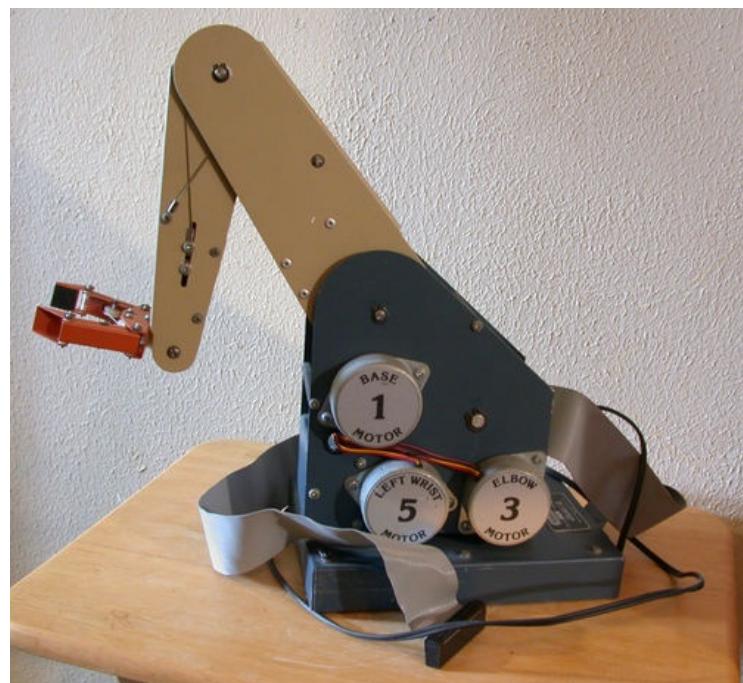
През 80-те години на 20-ти век в България се наблюдава бурно развитие на електрониката. [12] Управляваната от компютри автоматика навлиза все повече в индустрията. Поради нуждата от квалифицирани специалисти по автоматика и мехатроника е създадена серията учебни роботи РОБКО. Представляват неголеми механизми, задвижвани от стъпкови електродвигатели. Наподобяват индустриални роботи. Предназначени са да се управляват от персонален компютър Правец 82 или ИМКО-2.

Серията РОБКО е разработена в Института по Техническа Кибернетика и Роботика към Българската Академия на Науките (ИТКР-БАН) и произвеждана от Завод за медицинска техника в град София. Повечето модели РОБКО са еквиваленти на проектирани в западна Европа и САЩ роботи. Например РОБКО 9 е копие на Heathkit HERO 1, а РОБКО 10 силно наподобява Microbot TeachMover. [22] Първият робот от серията - РОБКО 01 (показан на фиг. 1.1) е директно копие на американския робот Mini-mover 5. Той е най- популяренят РОБКО, произведени са между 4 и 17 хиляди броя. [22]

Серията РОБКО включва въртяща се маса и конвейер (фиг. 1.2а), вариант на РОБКО 01 с монтирани на ставите мотори вместо метални жила за предаване на движението (фиг. 1.2б), модел със собствена работна маса (фиг. 1.2в), робот с деликатна щипка за стъкленици (фиг. 1.2г) и подобен на кран робот (фиг. 1.3, горе-ляво и център-дясно). Произвеждани са захранващ блок (фиг. 1.4в) и три различни накрайници за РОБКО 01 - обикновен хващащ, оптичен сензорен хващащ (фиг. 1.4а) и електромагнит (фиг. 1.4б). Оптичният сензорен хващащ се разглежда подробно в точка 2.3.1.



(a) РОБКО 01



(6) Mini-mover 5

Фигура 1.1: РОБКО 01 и Mini-mover 5



(а) Въртяща се маса и конвейер



(б) Подобен на РОБКО 01 модел



(в) РОБКО Вела 1

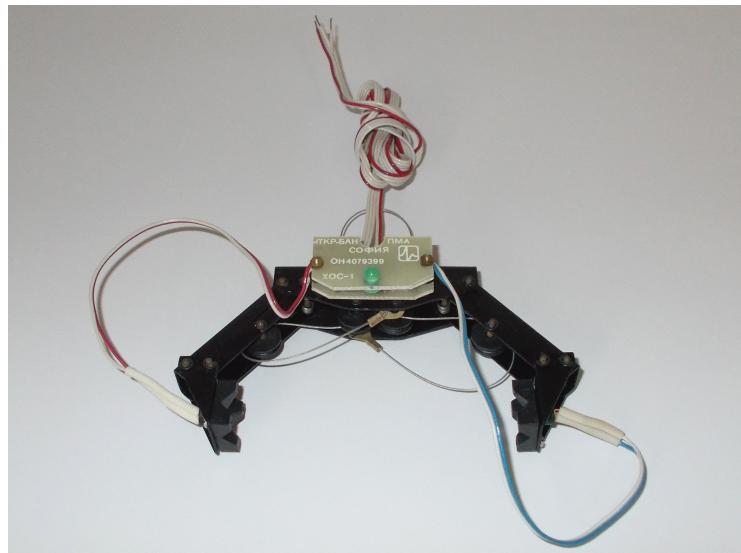


(г) Лабораторен модел

Фигура 1.2: Различни модели от серията РОБКО



Фигура 1.3: Колекция от различни модели РОБКО и Правец 8М



(а) Оптичен сензорен хващаč

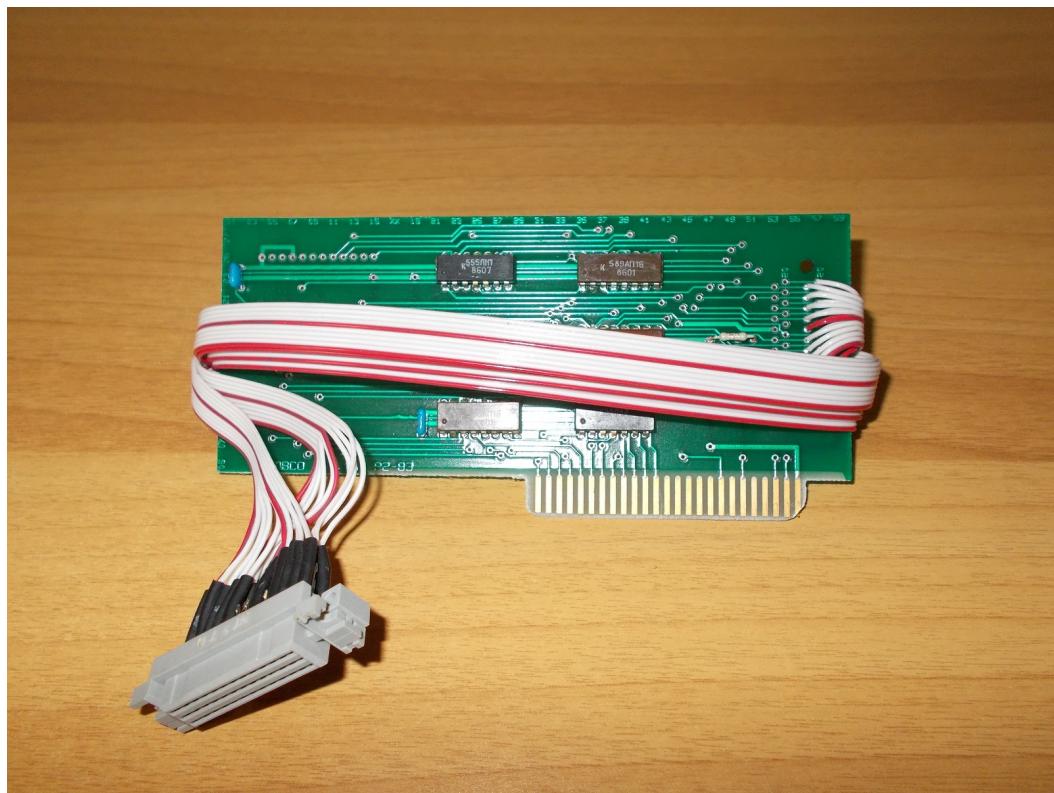


(б) Електромагнит



(в) Захранващ модул

Фигура 1.4: Аксесоари за РОБКО 01



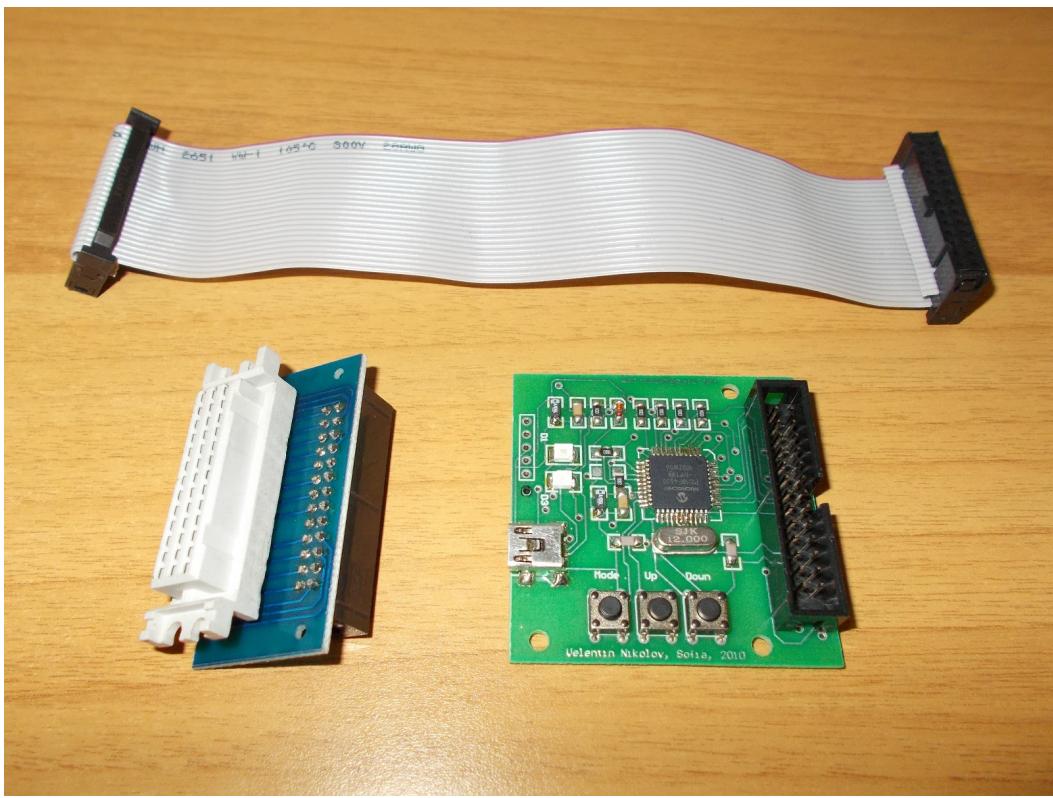
Фигура 1.5: Разширителна карта за ИМКО-2/Правец82, свързваща се с драйверната платка на РОБКО 01

## 1.2 РОБКО 01 в настоящето

Малкото запазени екземпляри от серията РОБКО почти винаги са от модел 01 или комплект въртяща се маса и конвейер. Съществуват различни проекти, успешно реализирали управление на РОБКО 01. Някои от тях се свързват с драйверната платка на робота, а други я заменят. Разработени са различни потребителски интерфейси: програми с графична среда, програми с команден ред и управление с джойс тици.

РОБКО 01 е проектиран да се управлява от ИМКО-2/Правец82 чрез разширителната карта показана на фигура 1.5. Повечето модерни РОБКО проекти заменят тази платка с микроконтролер.

Пример за такава реализация е проектът на Валентин Николов. [21] Проектираната от него платка (фиг. 1.6) е базирана на микроконтролер PIC4620 от Microchip. Разполага с бутони за ръчно управление и връзка с компютър през USB интерфейс. Неговият софтуер, изпълняван на персонален компютър, предоставя графична среда за управление на робота.

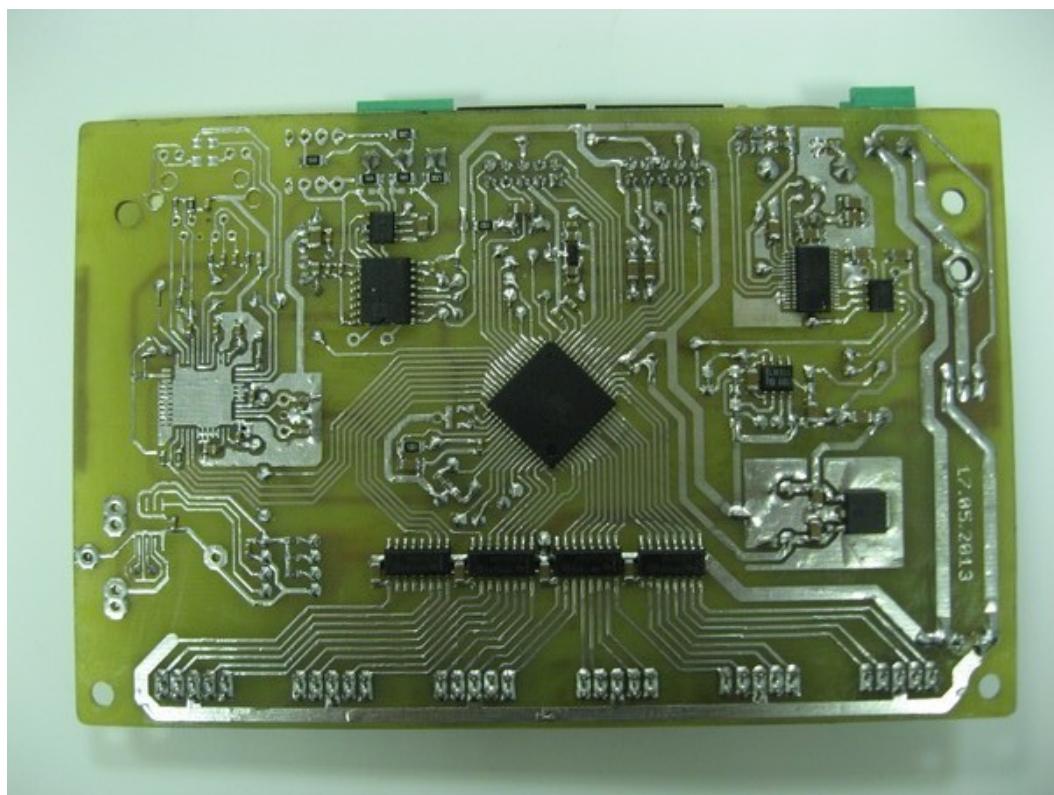


Фигура 1.6: Микроконтролерна платка, свързваща драйверната платка на РОБКО 01 с персонален компютър

Може да се контролира от мишка, клавиатура или джойстик и позволява едновременно движение на повече от един мотор. Демонстрация на функционалностите на проекта на Валентин Николов е налична в интернет.[20]

Подобна система е разработена от Орлин Димитров - студент в Технически Университет - Габрово. Контролерът, използван от него, е Arduino UNO. Микроконтролерната платка се свързва към РОБКО 01 и към персонален компютър, подаващ команди през сериен връзка. Роботът се управлява от програма с графична среда, изпълнявана на персоналния компютър. Софтуерът позволява както ръчно управление, така и изпълнение на поредица от предварително зададени движения. Отличителна черта на тази система е способността ѝ плавно да ускорява и забавя въртенето на моторите. [16] Проектът на Орлин Димитров е единственият известен в интернет, успешно реализирал едновременното управление на повече от един РОБКО 01. [15]

Друг подход за управлението на РОБКО 01 е да се подмени драйверната платка на робота. Този метод е приложен от Симеон Иванов,



Фигура 1.7: Платка, заменяща оригиналната драйверна платка на РОБКО 01

студент от Русенският университет "Ангел Кънчев". [19] Разработената от него платка (показана на фигура 1.7) се монтира на мястото на оригиналната драйверна платка в основата на робота. Включва в себе си микроконтролер ATMega128, Ethernet контролер ENC424J600, захранващ блок, шест интегрални схеми за управление на стъпкови мотори и интегрални схеми за комуникация по UART, RS232 и USB. Тази разработка е най-функционалната имплементация на РОБКО 01 сред известните в интернет такива.

Контролерът комуникира с персонален компютър по ModBUS протокол, който е стандартен за автоматизацията в индустриални условия. Това позволява отдалечно управление на робота по интернет. Могат да се движат множество мотори едновременно. Софтуерът на микроконтролерът извършва изчисления за права и обратна кинематика - може да позиционира щипката на робота на зададени координати и да определя текущото местоположение. Командите към контролера се подават под формата на G код. Това на практика превръща РОБКО 01 в CNC (Цифрово-Програмно Управляема) машина. Тази способност на разработката е показана на видеоклип в интернет. [18]

## 1.3 Настояща разработка

Целта на настоящия проект е да се проектира и изработи управляващо устройство за РОБКО 01 базирано на микроконтролер STM32L476RG Nucleo. Основните възможности на РОБКО 01 вече са реализирани в гореспоменатите разработки. Устройството, разглеждано в тази дипломна работа, се отличава със способността да засича предмети чрез оптичния сензорен хващащ. Няма известна в интернет разработка, успешно реализираща тази функционалност на РОБКО 01. Това се дължи до голяма степен на малкият брой оцелели до днес оптични сензорни хващащи за РОБКО 01.

Ключово внимание в текущата работа е обърнато на управлението по интернет. Други важни функционалности са изпълнението на команди, предварително записани във файл и буферирането на получените команди. Реализиран е и ръчен контрол чрез джойстици.

Не е разработена графична среда, понеже функционалностите които би предоставила биха се застъпвали с ръчното управление и софтуерът за отдалечено управление. Предвидени са възможности за движение по координати и запомняне на позиции, но не са реализирани на днешна дата.

## 1.4 Тенденции в автоматизацията

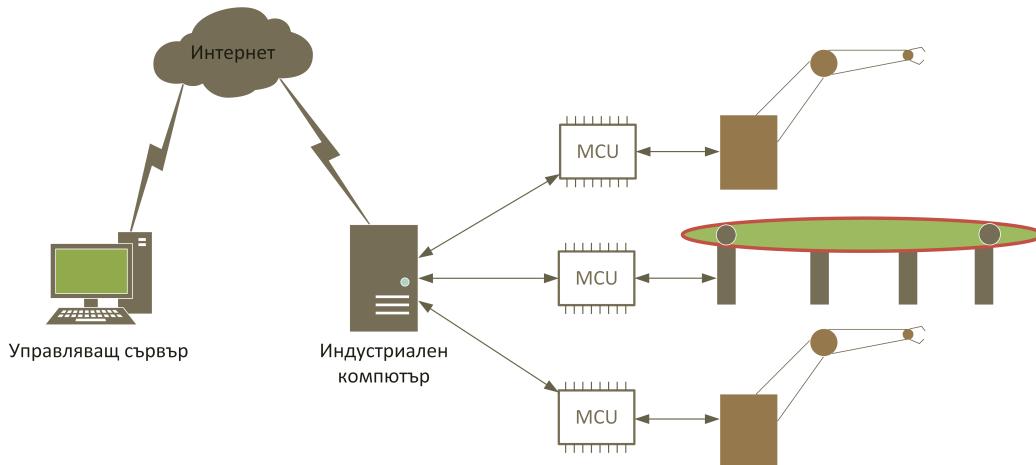
Вследствие на бурното развитие на компютърните мрежи все повече устройства разполагат с мрежова свързаност. Практиката да се добавя способност за комуникация през интернет в разнообразни уреди е известна като "Интернет на вещите". Това позволява множество уреди, поначало неразполагащи с мрежова свързаност, да комуникират по между си, с хора и с централен възел или сървър. Реализира се с помощта на вградени микроконтролерни системи и мрежови технологии, нерядко безжични. [11]

На базата на тази концепция се изграждат системи за автоматизация и управление, често в домашни условия. Такива системи се състоят от съвкупност от сензори (температура, светлина, натиск и много други), изпълнителни устройства (мотори, климатици, лампи и др.) и управляващи устройства (компютър, микроконтролер). [3] В близкото бъдеще е възможно сложни системи за домашна автоматизация да се интегрират с цифрови асистенти като Google Home и Amazon Echo.

Комютрите отдавна се използват в индустрията за автоматизация, но с ограничена комуникация между отделните машини. Навлизането на модерните мрежови технологии в индустрията позволява по-бърз обмен на големи количества данни и по-голяма взаимосвързаност между машините. Това е една от предпоставките за настъпващата четвърта индустриална революция. [4]

Заводите на новото време ще се нуждаят от все по-малка човешка намеса. Когато има нужда от такава, в много случаи може да се извърши отдалечно чрез подходящи мрежови технологии. С оглед на това текущата разработка не цели единствено да реставрира музен експонат от времето на социалистическа България, а да демонстрира нагледно възможността за отдалечно управление на индустриални машини.

Пример за отдалечно управление на завод е показан на фигура 1.8. Заводът разполага с множество машини, всяка с вграден микроконтролер. Индустриският компютър е свързан към контролерите на машините съставящи една поточна линия или сегмент от такава. Компютърът съгласува действията на отделните машини и разполага с връзка към Интернет. Ресурсите на завода могат да се разширят като се добавят нови групи от машини и компютри. Операторът на машините може да работи в офис далеч от сградата на завода и да управлява производствения процес от там.



Фигура 1.8: Пример за отдалечно управление на поточна линия

# Система за автоматизирано отдалечено управление на РОБКО 01 с възможност за ръчен контрол

## 2.1 Функционални изисквания

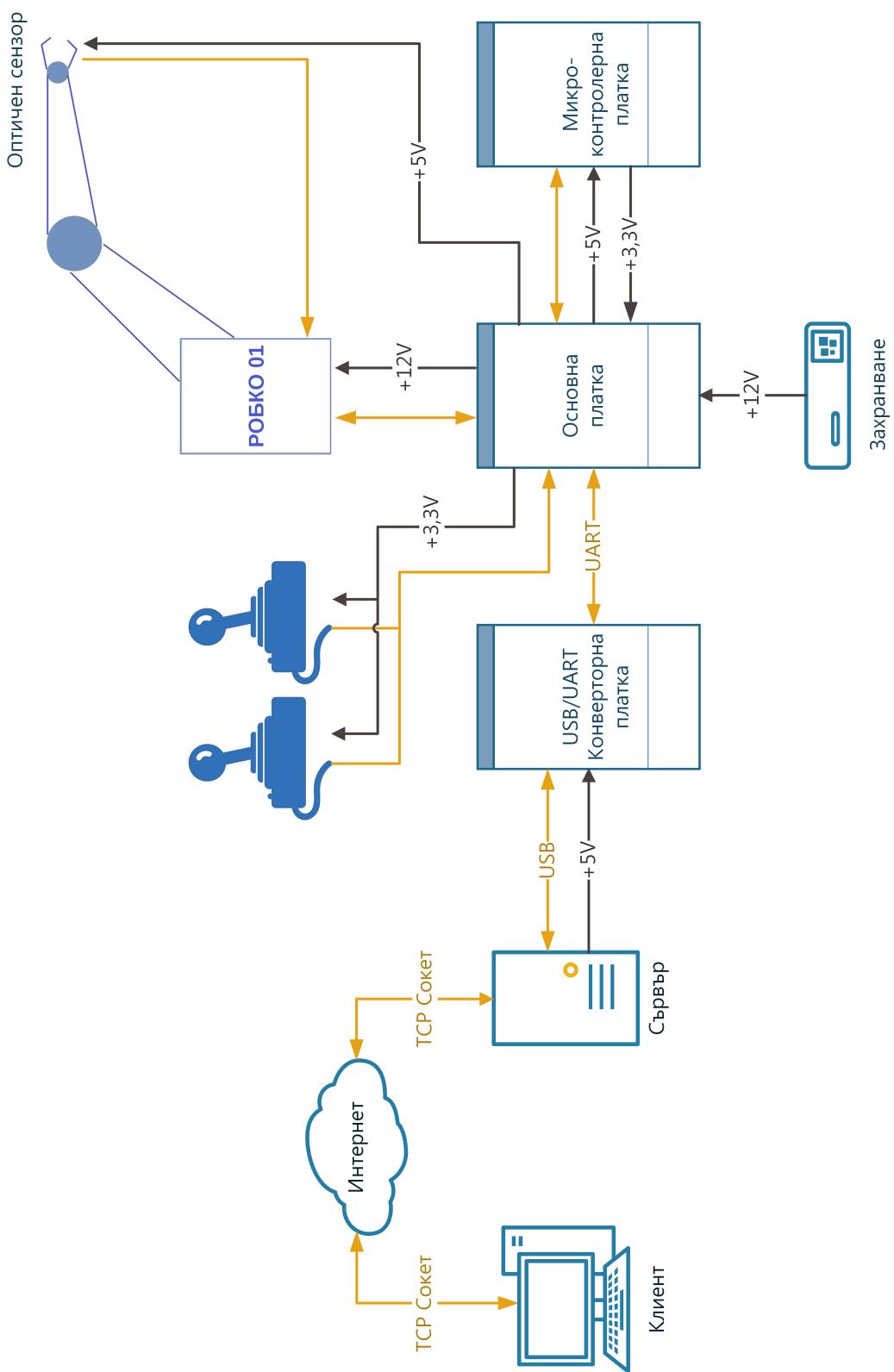
Към настоящата разработка са поставени следните изисквания:

- ръчно управление на РОБКО 01
- управление на робота чрез текстови команди
- подаване на команди през интернет
- четене на команди от файл
- опашка за команди
- едновременно движение на всички мотори
- засичане на предмет в щипката на робота

## 2.2 Топология на системата

На фигура 2.1 е показана блокова схема на проектираната система. Жълтите стрелки обозначават информационни сигнали, а кафявите - захранващи напрежения и токове.

Основната платка реализира връзките между отделните компоненти



Фигура 2.1: Блокова схема на системата

на системата. Прякото управление на робота се извършва от микроконтролер STM32L476RG, разположен на собствена платка. Микроконтролерът получава информация относно какви движения трябва да изпълни роботът от два източника: два джойстика (ръчно управление) и компютър в интернет, наричан клиент. Клиентът изпраща команди през интернет до втори компютър - сървъра. Сървърът е свързан с микроконтролера и препраща командите към него.

## 2.3 Компоненти на системата

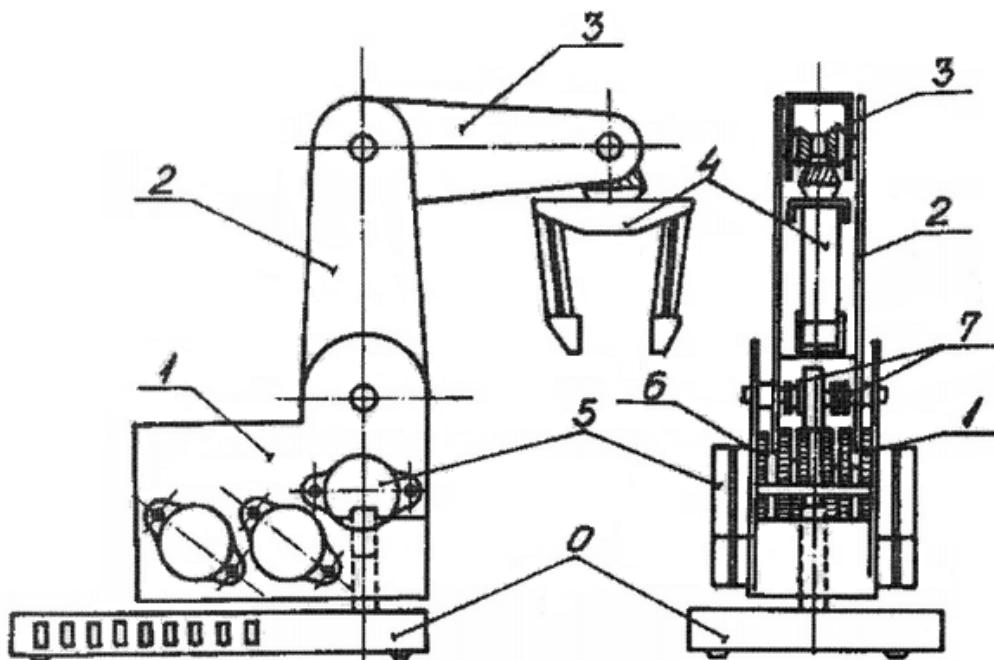
### 2.3.1 РОБКО 01

#### Механика

РОБКО 01 е антропоиден робот със шест степени на свобода, реализирани с кинематични връзки. [17] Изграден е от неподвижна основа, три звена и хващащ. Задвижва се от шест стъпкови електромотора, означени с 5 на фигура 2.2. Те са монтирани от външната страна на първото звено(1). Във вътрешността му се помещават шест зъбни редуктора(6). Движението на моторите се предава към звената и щипката(4) чрез метални въжета и ролки(7).

Първото звено се върти около ос, перпендикулярна на основата. Второто(2) и третото(3) звено, наричани съответно рамо и лакът, се накланят съответно напред-назад и нагоре-надолу. Хващащът се върти и накланя нагоре-надолу. Тези движения се реализират посредством диференциална зъбна предавка, разположена между третото звено и хващача. Тя се задвижва от два мотора. При съпосочено въртене на моторите хващащът се накланя в съответната посока. Когато двата мотора се въртят противопосочно, едната страна на хващача се накланя нагоре, а другата надолу. Резултатното движение е въртене.

Върху второто звено е монтиран прекъсвач, който се затваря когато металното въже, затварящо хващача, се обтегне достатъчно. Това позволява да се засича дали хващащът е стиснал предмет. Тази функционалност не се използва в текущата разработка заради голямата сила, нужна за да се затвори прекъсвача, и опасността въжето да се скъса.



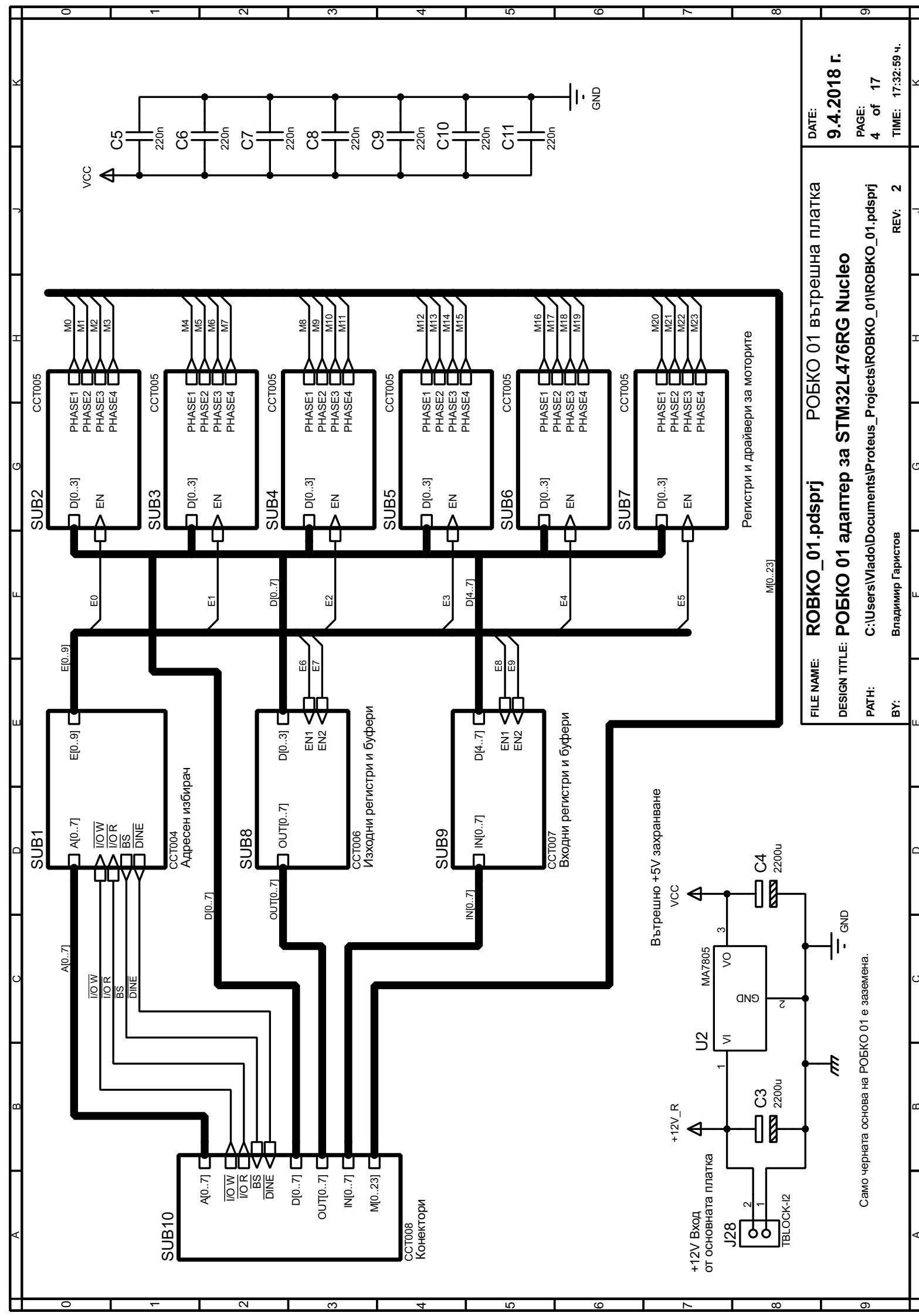
Фигура 2.2: РОБКО 01 - схема на механиката

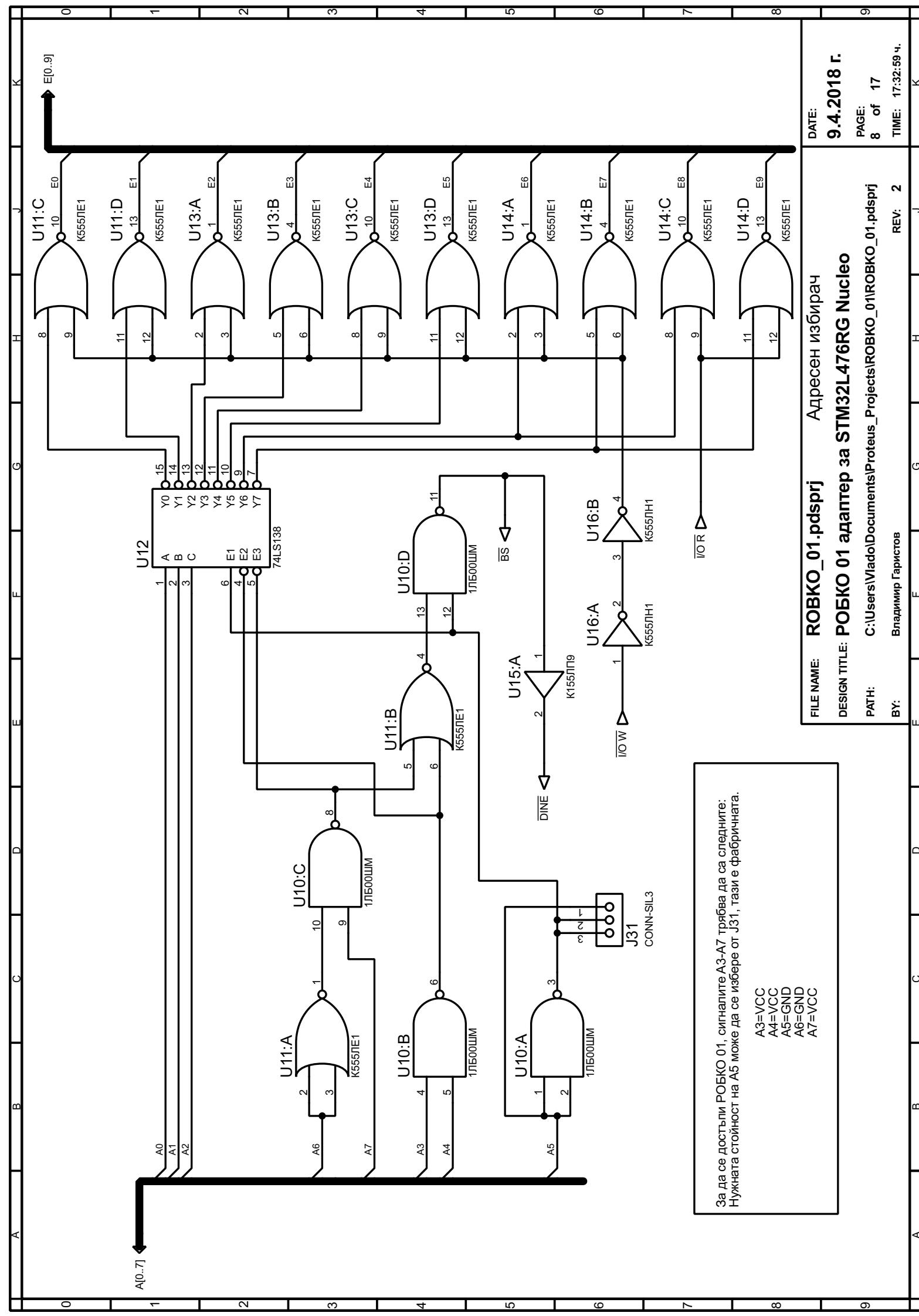
### Драйверна платка

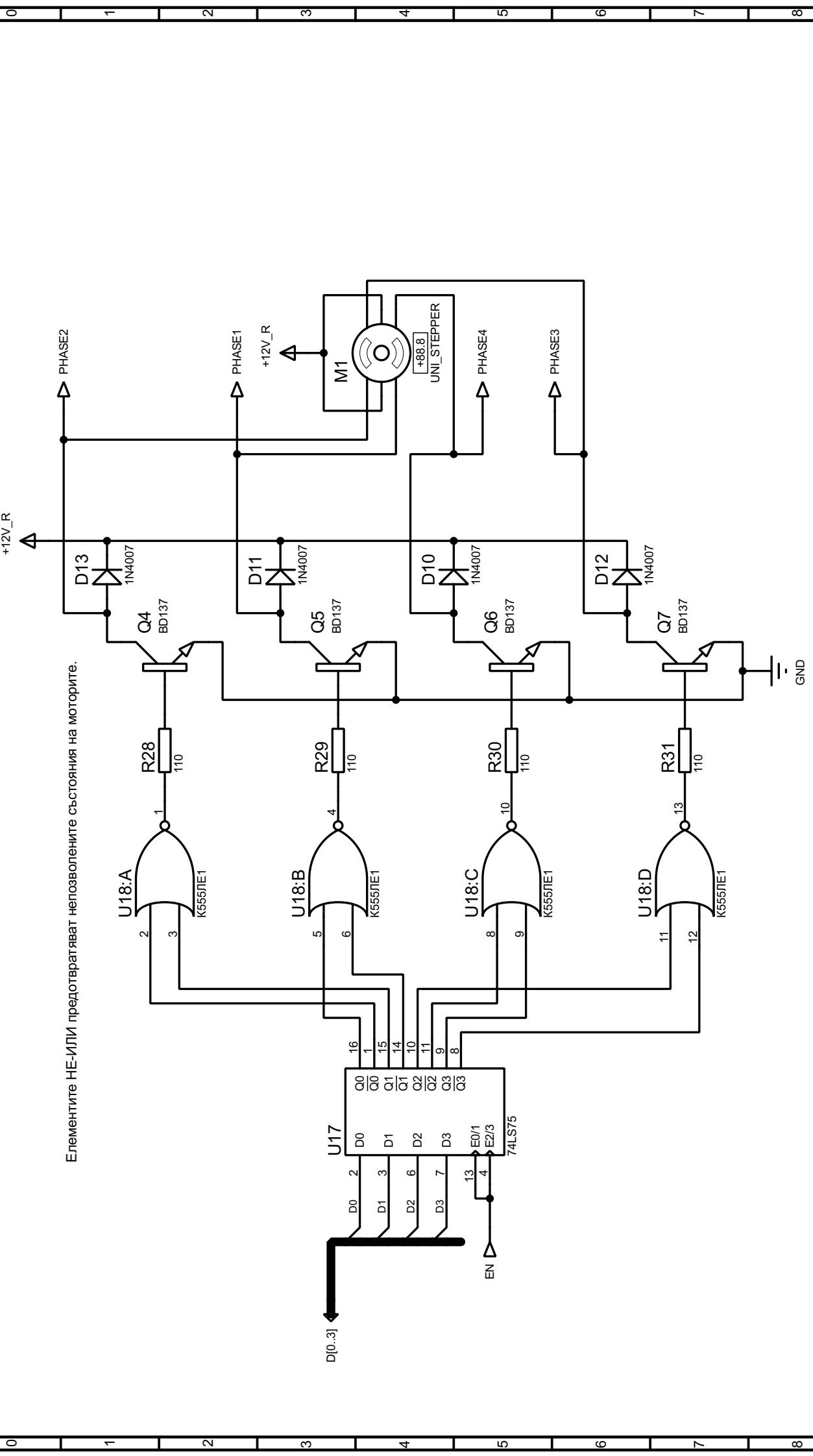
Драйверната платка на РОБКО 01 е описана подробно в ръководството за експлоатация [17], но липсват важни подробности. Не е обозначено кой сигнал на кой пин на конекторите е изведен. Липсват стойностите на пасивните елементи. Пропуснати са четири кондензатора. Не са дадени моделите на компонентите.

След анализ на печатната платка и с помощта на информация от интернет и от предишния собственик на робота, принципната електрическа схема на РОБКО 01 беше въстановена и попълнена с липсващата информация. На следващите страници е дадена въстановената принципна електрическа схема. Схемните страници от 10-та до 14-та включително са идентични на 9-та схемна страница и не са показани.

Цифровата логика на схемата е реализирана с ИС от серията 74LSxx и техни еквиваленти. Интегралните схеми се захранват от линеен регулатор на напрежение MA7805. На моторите и на регулатора се подава напрежение 12V.





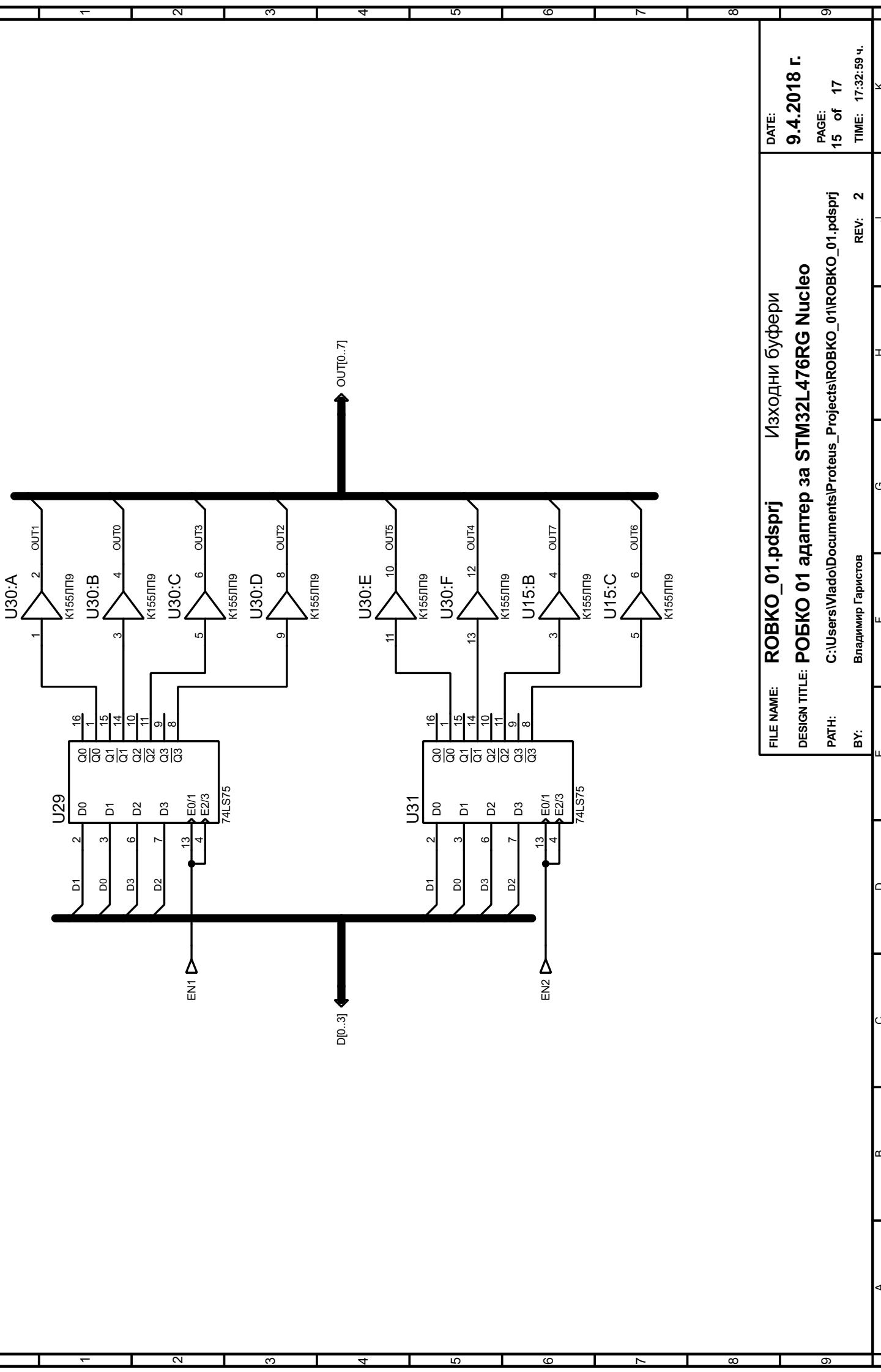


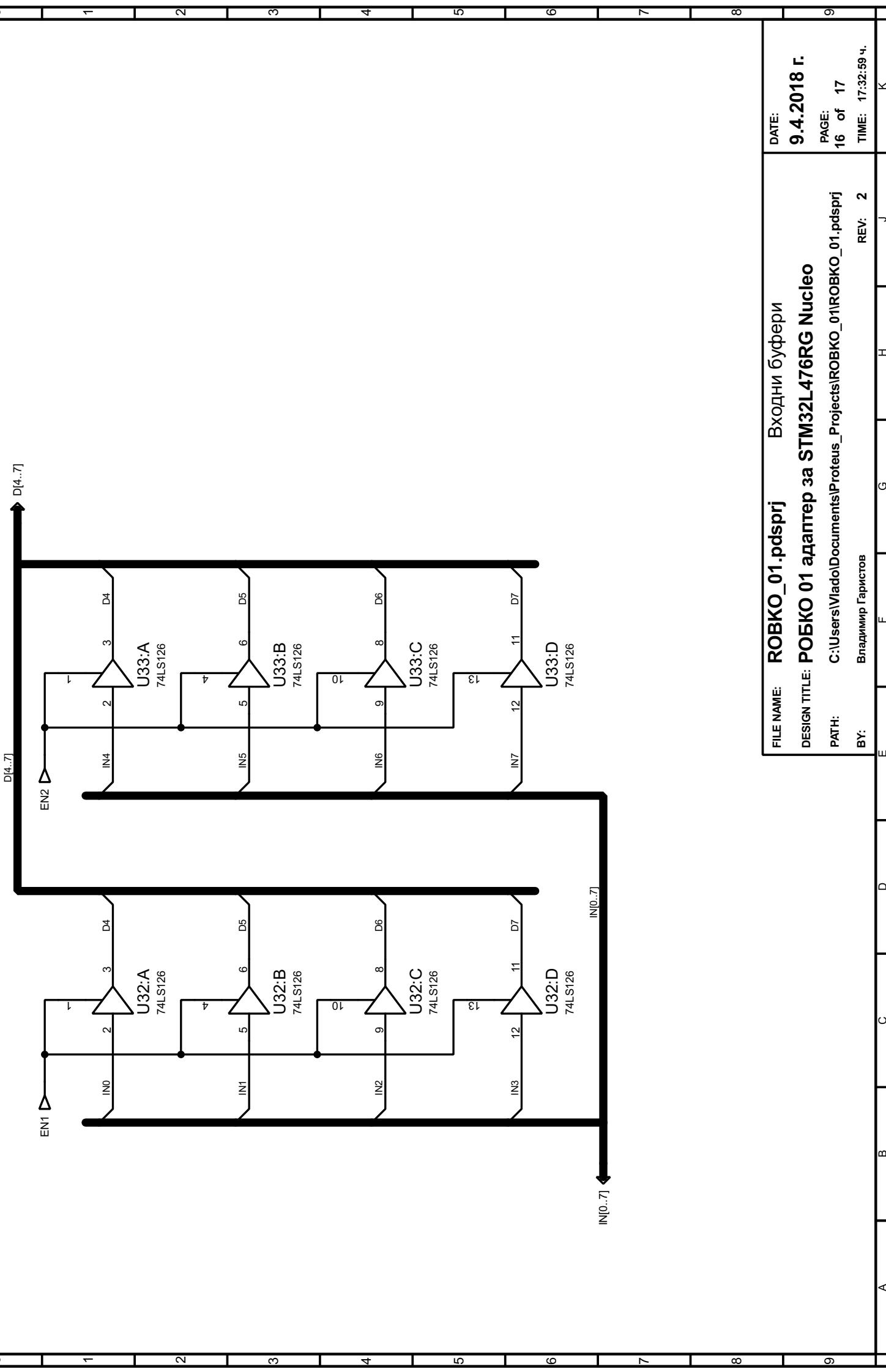
**Драйвер за стъпков мотор**  
**РОБКО 01 адаптер за STM32L476RG Nucleo**

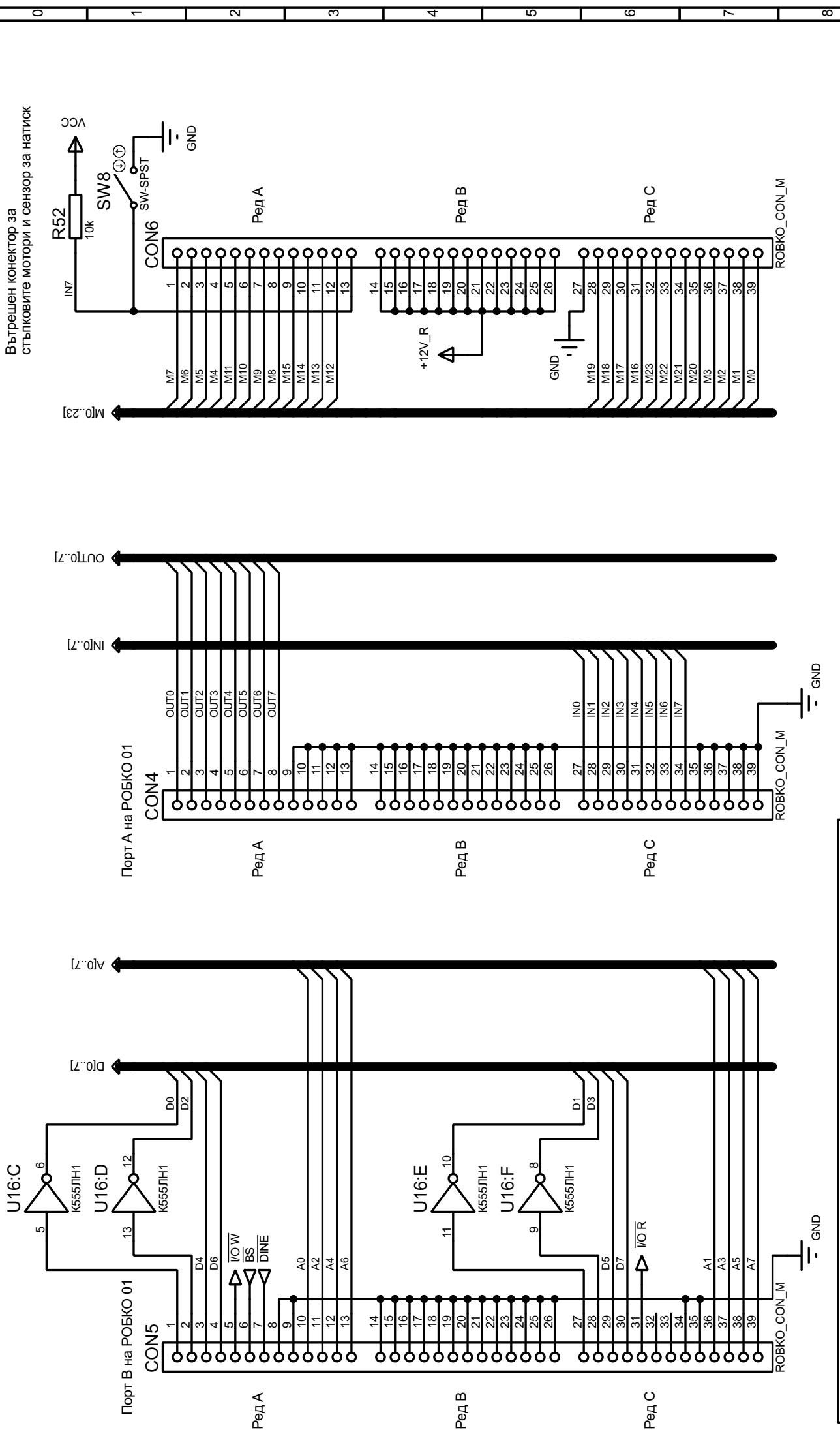
FILE NAME: ROBKO\_01.pdsprj  
DESIGN TITLE: РОБКО 01 адаптер за STM32L476RG Nucleo  
PATH: C:\Users\Vlado\Documents\Proteus\_Projects\ROBKO\_01\ROBKO\_01.pdsprj  
BY: Владимира Гаристов

DATE: 9.4.2018 г.  
PAGE: 9 of 17  
TIME: 17:32:59 ч.

REV: 2  
A B C D E F G H I J K







Понеже платката се монтира със страна компоненти гледаша надолу, конекторите са обрънати. Ред С е горният ред на конекторите, В е средният, а А е долният. Ножерата на пиновете нарастват от ляво надясно.

Стъпковите електромотори са двуфазни униполлярни. Двете фази са изградени от по две статорни намотки. Шестте мотора се управляват от транзисторни крайни стъпала с диодна защита от обратно напрежение. За всеки мотор отговаря по един 4-битов регистър, съхраняващ състоянията на намотките/фазите му. Чрез записване на данни в регистрите се контролира през кои намотки на стъпковите мотори да протича ток.

Не всяка от 16-те възможни стойности на регистрите съответства на позволено състояние на моторите. Непозволените комбинации се предотвратяват от логически елементи ИЛИ-НЕ. В таблица 2.1 са дадени позволените комбинации от токове през намотките на моторите. Индексите на токовете съответстват на индексите на битовете в регистрите за съответните мотори. За въртене в режим на полуустъпка се изпълняват в дадения ред, а за режим на цяла стъпка се пропускат полуустъпките при които протича ток само през една намотка.

I <sub>0</sub>	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>
1	0	0	0
1	0	1	0
0	0	1	0
0	1	1	0
0	1	0	0
0	1	0	1
0	0	0	1
1	0	0	1

Таблица 2.1: Поредност на стъпките на моторите за въртене в режим на полуустъпка

Два допълнителни регистра са заделени за осемте изходни пина на платката (OUT0-OUT7). Към изходите на тези регистри са свързани буфери с изходи от тип отворен колектор. Това позволява от изходните пинове на робота да се управляват товари, работещи на по-високо напрежение от ТТЛ интегралните схеми.

РОБКО 01 разполага и с осем входни пина (IN0-IN7). Данните от тях се буферират в повторители с високоимпедансно изходно състояние.

Достъпът до регистрите се контролира от 3-битов дешифратор, обуславящ 8 адреса. Първите шест адреса се отнасят за регистрите на стъпковите мотори. Последните два адреса отговарят едновременно за 8-те изходни пина и за 8-те входни пина.

Понеже РОБКО 01 е проектиран да се управлява от 8-битовия Правец

82, разполага с 8-битова <sup>1</sup> адресна магистрала (A0-A7) и 8-битова магистрала за данни (D0-D7). Младшите <sup>2</sup> 4 бита от магистралата за данни се използват от компютъра (или микроконтролера) за да записва данни в регистрите на робота. Старшите четири бита се използват за четене на входните пинове на РОБКО 01.

С трите най-младши адресни бита се избира регистър или входящ буфер, а останалите адресни битове транслират тези адреси в адресното пространство на Правец 82. Те се свързват към логически елементи И-НЕ и ИЛИ-НЕ, които подават разрешаващ сигнал на дешифратора само ако старшите адресни битове имат правилните стойности. В таблица 2.2 е описано адресното пространство на РОБКО 01. За права посока се приема реда на стъпки даден в таблица 2.1, изпълняван от горе надолу.

Адрес	Мотор / буфер	Движение в права посока
0x98	0 - Ос на основата	Въртене наляво
0x99	1 - Раменна става	Накланяне напред
0x9A	2 - Лакътна става	Повдигане нагоре
0x9B	3 - Щипка - диф. предавка, десен	Накланяне надолу
0x9C	4 - Щипка - диф. предавка, ляв	Накланяне нагоре
0x9D	5 - Щипка - пръсти	Отваряне
0x9E	Младши входни/изходни битове	
0x9F	Старши входни/изходни битове	

Таблица 2.2: Адресно пространство на РОБКО 01

РОБКО 01 има четири контролни сигнала -  $\overline{BS}$ ,  $\overline{DINE}$ ,  $\overline{I/O\ R}$  и  $\overline{I/O\ W}$ . Първите два не се използват.

$\overline{I/O\ R}$  е разрешаващ сигнал за четене. Когато е в активно ниско ниво, входящите буфери подават на D4-D7 състоянията на избраниите входни пинове. Когато е в неактивно ниво или е избран адрес, различен от този на входно/изходните буфери, изходите на входните буфери се намират във високоимпедансно състояние.

$\overline{I/O\ W}$  е разрешаващ сигнал за записване. При преминаването му от неактивно в активно ниво данните, подадени на D0-D3, се записват в регистъра, чийто адрес е избран.

<sup>1</sup> Адресната магистрала на Правец 82 е 16-битова. [13] Разширителната платка, показана на фигура 1.5, свързва РОБКО 01 с Правец 82 и транслира адресното пространство на робота в това на компютъра.

<sup>2</sup> При Правец 82 и РОБКО 01 нулевият бит е най-младши.

РОБКО 01 разполага с четири конектора. Вътрешният конектор се свързва към стъпковите електромотори и сензора за затваряне на щипката. На конекторът, маркиран като порт А са изведени входните и изходните пинове. Порт В е предназначен за връзка с компютър и на него са изведени адресната магистрала, магистралата за данни и контролните сигнали. Между тези два конектора се намира захранващия конектор. Изискванията към захранването са постоянно напрежение  $12V \pm 0,6V$  и максимален ток 5A. [17]

#### Оптичен сензорен хващащ

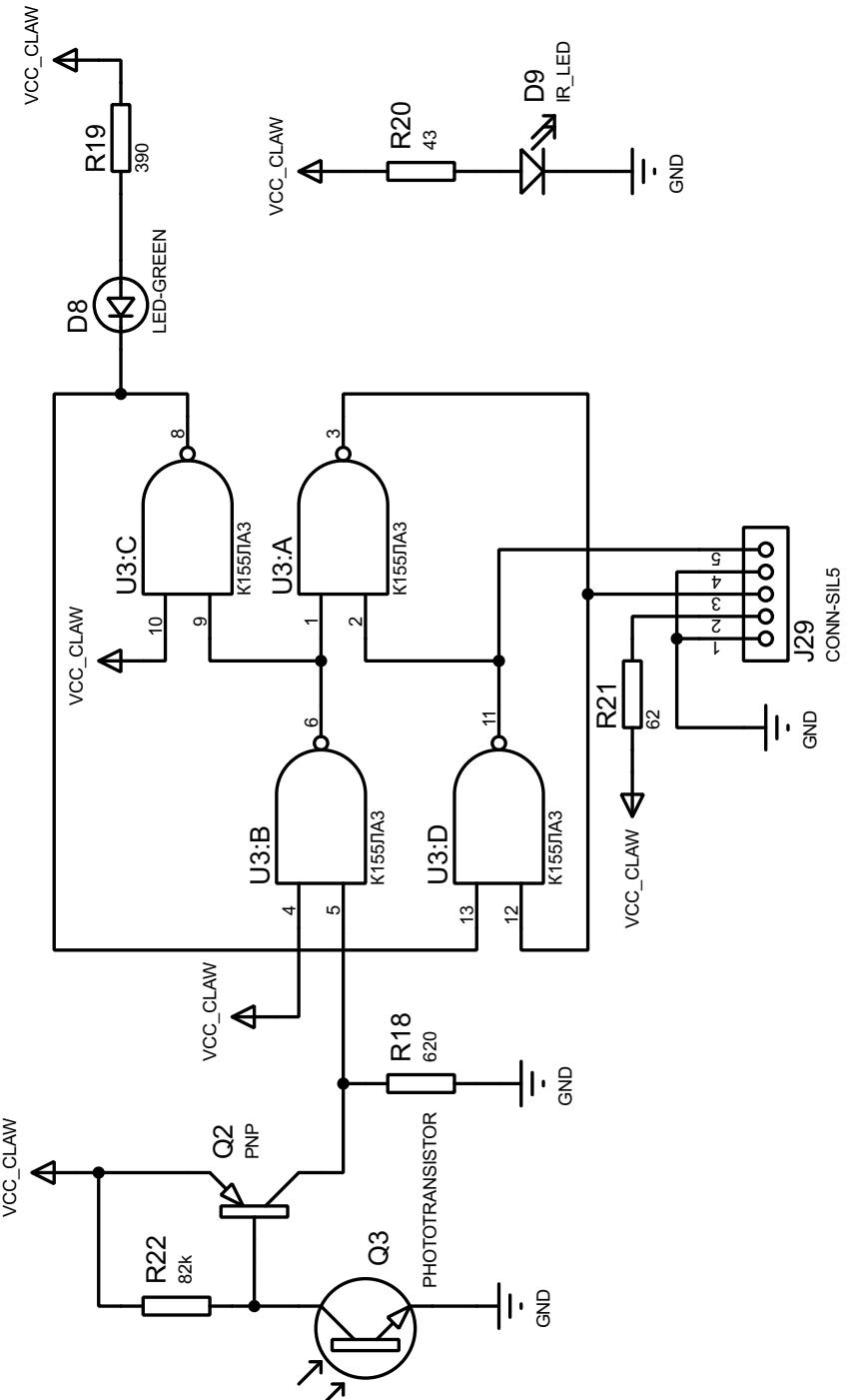
От трите накрайника на РОБКО 01 се използва оптичния сензорен хващащ. Представлява щипка с монтирани инфрачервен светодиод на единия "пръст фототранзистор на другия и малка печатна платка върху тялото (виж фигура 1.4a).

На следващата страница е показана принципната електрическа схема на оптичния сензорен хващащ. Такава схема не присъства в ръководството за експлоатация на РОБКО 01 [17], не е налична и в интернет. Показаната схема е изработена след анализ на печатната платка на хващача.

D9 е инфрачервеният светодиод, който осветява фототранзистора Q3. Q3 работи в активен режим и пропуска достатъчно ток през базата на Q2 за да го отпуши. Входът на U3:B попада във високо ниво и изходът му преминава в ниско. В следствие на това изходите на U3:A и U3:C преминават във високо ниво. D8 не свети, изходът на U3:D е в ниско ниво. Пинове 3 и 5 на конектор J29 са съответно във високо и ниско ниво.

Когато предмет попадне между "пръстите" на щипката, светлинния лъч от D9 се препречва. Транзисторите Q3 и Q2 се запушват и четирите логически елемента сменят състоянията на изходите си. D8 започва да свети, а пиновете 3 и 5 на J29 разменят състоянията си (3-ти преминава в ниско, 5-ти във високо).

Пин 5 на конектор J29 се свързва към пин 33 (IN6) на порт А на РОБКО 01. Това позволява на микроконтролера да чете състоянието му през регистрите на драйверната платка. Пинове 1 и 2 се свързват съответно към пинове 2 и 1 на конектор J23 на основната платка, откъдето се захранва оптичния сензорен хващащ. Причината хващачът да не се захранва от драйверната платка е, че на портовете на робота не е изведено  $+5V$  захранващо напрежение и всички изходни пинове са от тип отворен колектор.



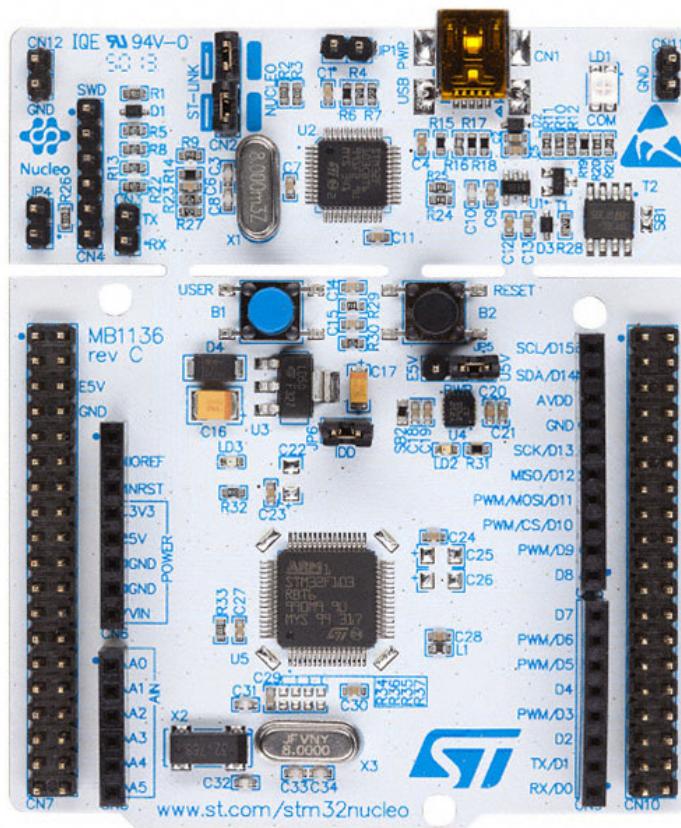
Малка платка, монтирана върху щипката на РОБКО 01.  
Ако нищо не стои между D9 и Q3, D8 не свети, на конектор J29 пин 3 е във високо ниво, а пин 5 е в ниско.  
Когато се засече предмет, D8 светва и пинове 3 и 5 на J29 разменят състояниета си.

FILE NAME:	<b>ROBK0_01.pdsprj</b>	Оптичен сензорен хващащ	DATE:	<b>9.4.2018 г.</b>
DESIGN TITLE:	<b>РОБКО 01 адаптер за STM32L476RG Nucleo</b>		PAGE:	5 of 17
PATH:	C:\Users\Vlado\Documents\Proteus_Projects\ROBK0_01\ROBK0_01.pdsprj		TIME:	17:32:59 ч.
BY:	Владимир Гаристов		REV:	2

### 2.3.2 STM32L476RG Nucleo

Системата за управление на РОБКО 01 се базира на микроконтролерната платка STM32L476RG Nucleo, показана на фигура 2.3. Платките от серията Nucleo на ST Microelectronics комбинират микроконтролер, програматор, дебъгер и помощни елементи (светодиоди, прекъсвачи и конектори).[7] Програматор/дебъгерът е втори, по-малък микроконтролер. Намира се в горната част на платката.

Използваният микроконтролер - 32-битовият STM32L476RG разполага с ARM ядро с вграден FPU (Floating Point Unit). Отличава се със свръхниска консумация - 30nA в най-маломощен режим. Ядрото поддържа прекъсвания и работи с максимална тактова честота 80MHz. Микроконтролерът разполага с 1MB флеш памет и 128KB статична RAM памет. Има DMA (Direct Memory Access) блок. Съдържа множество периферни модули - UART, SPI, I2C, LCD драйвер, 12-битови АЦП и ЦАП, таймери и други. [9]



Фигура 2.3: Микроконтролерна платка STM32L476RG Nucleo

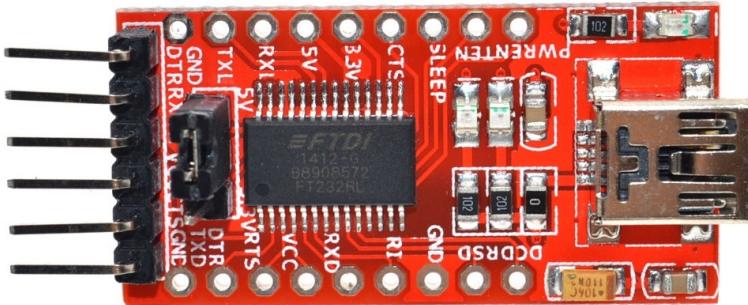
Микроконтролерът се свързва с адресната магистрала, магистралата за данни и контролните сигнали на РОБКО 01. Това му позволява да чете и пише в регистрите на драйверната платка. Софтуерът на микроконтролера се разглежда подробно в глава 5. Основната му функция е да определя какви стойности да записва в регистрите на драйверната платка на РОБКО 01 за да бъдат изпълнени желаните движения.

### 2.3.3 Основна платка и адаптери

В рамките на настоящата разработка са проектирани и изработени три печатни платки. Две от тях са пасивни адаптери за специфичните конектори на РОБКО 01. Печатната платка означена на фигура 2.1 като "Основна платка" служи за централен възел на системата - свързва отделните модули един към друг. Също така предоставя визуална индикация за състоянието на моторите, възможност за настройка и петволтово захранване. Проектираните печатни платки се разглеждат подробно в глава 3.

### 2.3.4 USB/UART модул

Комуникацията между сървърният компютър и микроконтролера се извършва по серийна UART връзка. Скоростта е 115200 бода, изпращат се осембитови думи с по един допълнителен бит за проверка по четност и предаването на данни се завършва с един стоп бит. Битът за проверка по четност е единица, когато в осемте бита на предаваната дума има четен брой единици. Връзката между микроконтролера и компютъра преминава през USB/UART адаптерна платка, показана на фигура 2.4. Сървърният софтуер достъпва USB/UART модула като виртуален сериен порт, свързан по USB връзка.



Фигура 2.4: USB/UART адаптерна платка

### 2.3.5 Компютри и софтуер за отдалечно управление

Отдалеченото управление през Интернет се реализира чрез две програми - клиент и сървър. Клиентската програма чете текстови команди, въвеждани от потребителя в команден ред и ги изпраща през TCP сокет към сървърната програма. На сървъра се съхраняват файлове, съдържащи предварително въведени поредици от команди. Сървърната програма препраща получените команди към микроконтролера или прочита файл и изпраща записаните в него команди. Двете програми и форматът на командите се описват в глава 4.

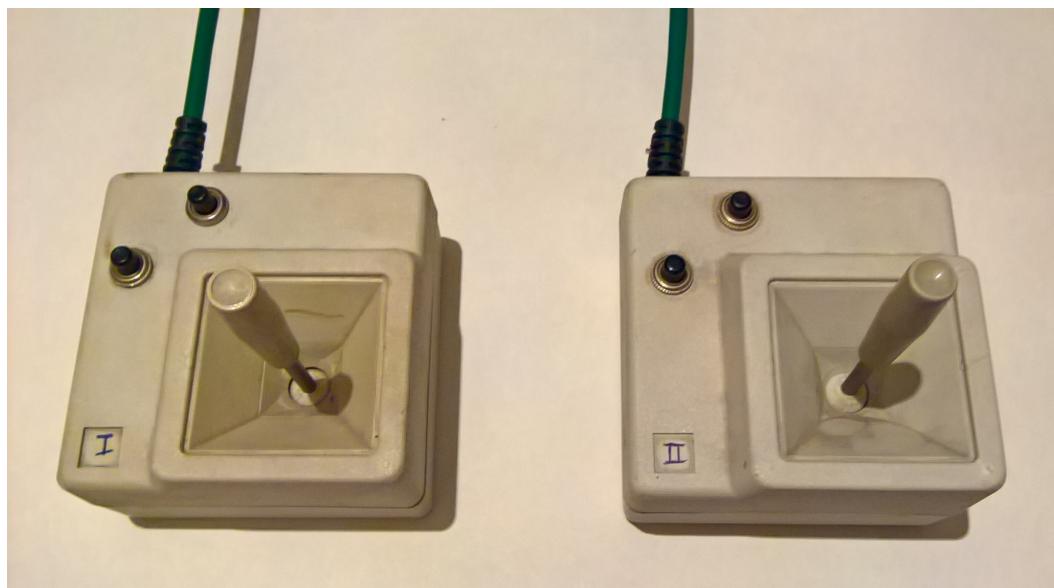
За реализацията на проекта един и същи компютър е използван и като клиент, и като сървър. Клиентската програма се изпълнява на виртуална машина Ubuntu Server. Употребата на виртуална машина позволява клиентската и сървърната програма да използват различни IP адреси, което е наложително за тестване на мрежовата свързаност.

При имплементация на подобна система в промишлени условия сървърният компютър вероятно ще бъде индустриски, а клиентският - обикновен персонален компютър. Индустриските компютри се отличават с по-високи изисквания за издръжливост на температура, прах, механични вибрации и удари, електромагнитни смущения и влага. Често разполагат със специализирани конектори и са предназначени за монтаж в табла, на стени или в шкафове.

### 2.3.6 Джойстици

Ръчното управление на робота се осъществява посредством два джойстика, показани на фигура 2.5. Разполагат с по два бутона и една дръжка. Джойститеците са произведени като периферни устройства за Правец 8[14] и са директно копие на джойстик A2M2002, произвеждан от Apple.[1] Движението на дръжката завърта два потенциометъра. Измерването на съпротивлението им се е извършвало с помощта на таймер 558, свързан като чакаш мултивибратор. Продължителността на генерираните от таймера импулси е пропорционална на съпротивлението и съответно на позицията на дръжката на джойстика. [5]

За да се използват в настоящата разработка двета джойстика са



Фигура 2.5: Модифицирани джойстици за Правец 8

модифицирани. Оригиналните им кабели са заменени с UTP Cat 5 кабели с накрайници RJ45. Неизползваните изводи на потенциометрите са свързани към захранващото напрежение, образувайки делител на напрежение. На следващата страница е показана принципната електрическа схема на джойститеците след извършените модификации. Първите два пина се използват за засичане на джойститеците от микроконтролера. Когато и двета джойстика са свързани към основната платка, през тези окъсени пинове се изгражда връзка между пин PC7 на микроконтролера и 3,3V захранване.

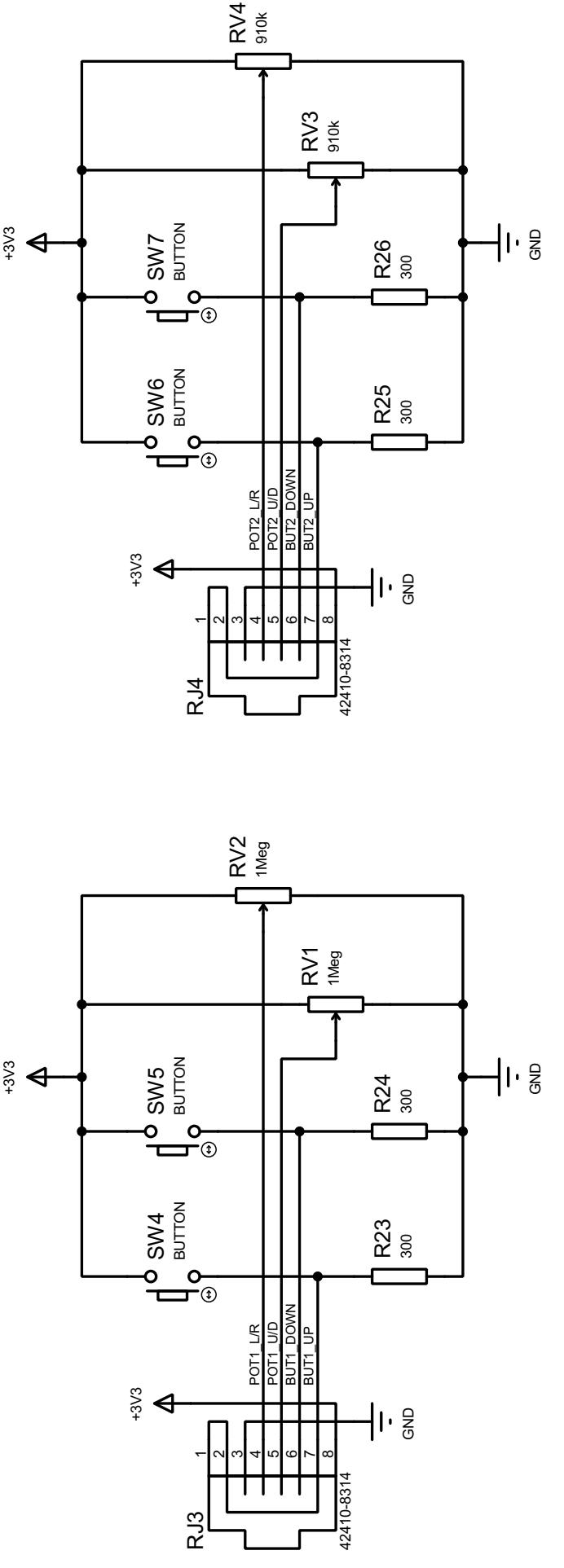
Първият джойстик се свързва към конектор RJ1 на основната платка, а вторият към RJ2 (виж точка 3.1.5). За горна страна се приема страната,

от която излиза кабелът. В таблица 2.3 се описва кой джойстик се използва за управлението на какви движения на РОБКО 01.

Джойстик	Позиция/натиснат буто	Движение на РОБКО 01
Първи	дръжка наляво	щипка - въртене наляво
	дръжка надясно	щипка - въртене надясно
	дръжка нагоре	повдигане лакът
	дръжка надолу	сваляне лакът
Втори	горен буто	щипка - завъртане нагоре
	долен буто	щипка - завъртане надолу
	дръжка наляво	основа - въртене наляво
	дръжка надясно	основа - въртене надясно
	дръжка нагоре	рамо назад
	дръжка надолу	рамо напред
	горен буто	щипка - стискане
	долен буто	щипка - отпускане

Таблица 2.3: Съответствия между джойстици и движения на РОБКО 01

Позициите на потенциометрите RV1 и RV3 се променят при движение на джойстиците нагоре и надолу, а RV2 и RV4 - при движение наляво и надясно.  
Получените напрежения на пинове 4 и 5 на конекторите нарастват при движение наляво или нагоре.



FILE NAME:	<b>ROBK0_01.pdsprj</b>	Джойстици	DATE:
DESIGN TITLE:	<b>РОБК0 01 адаптер за STM32L476RG Nucleo</b>		<b>9.4.2018 г.</b>
PATH:	C:\Users\Viado\Documents\Proteus_Projects\ROBK0_01\ROBK0_01.pdsprj	PAGE:	6 of 17
BY:	Владимир Гаристов	TIME:	17:32:59 ч.

### 2.3.7 Захранване

Системата се захранва от 12-волтово импулсно захранване<sup>3</sup>, показвано на фигура 2.6. Максималният изходен ток на захранването е 10A, максималната мощност е 120W. Според спецификациите на импулсното захранване входното му напрежение е в обхвата от 110V до 220V, но функционира правилно и при входно напрежение 230V.



Фигура 2.6: Захранващ източник

---

<sup>3</sup>с изключение на двата компютъра и USB/UART конверторната платка, която се захранва от сървърния компютър, виж фиг. 2.1

# Проектирани печатни платки

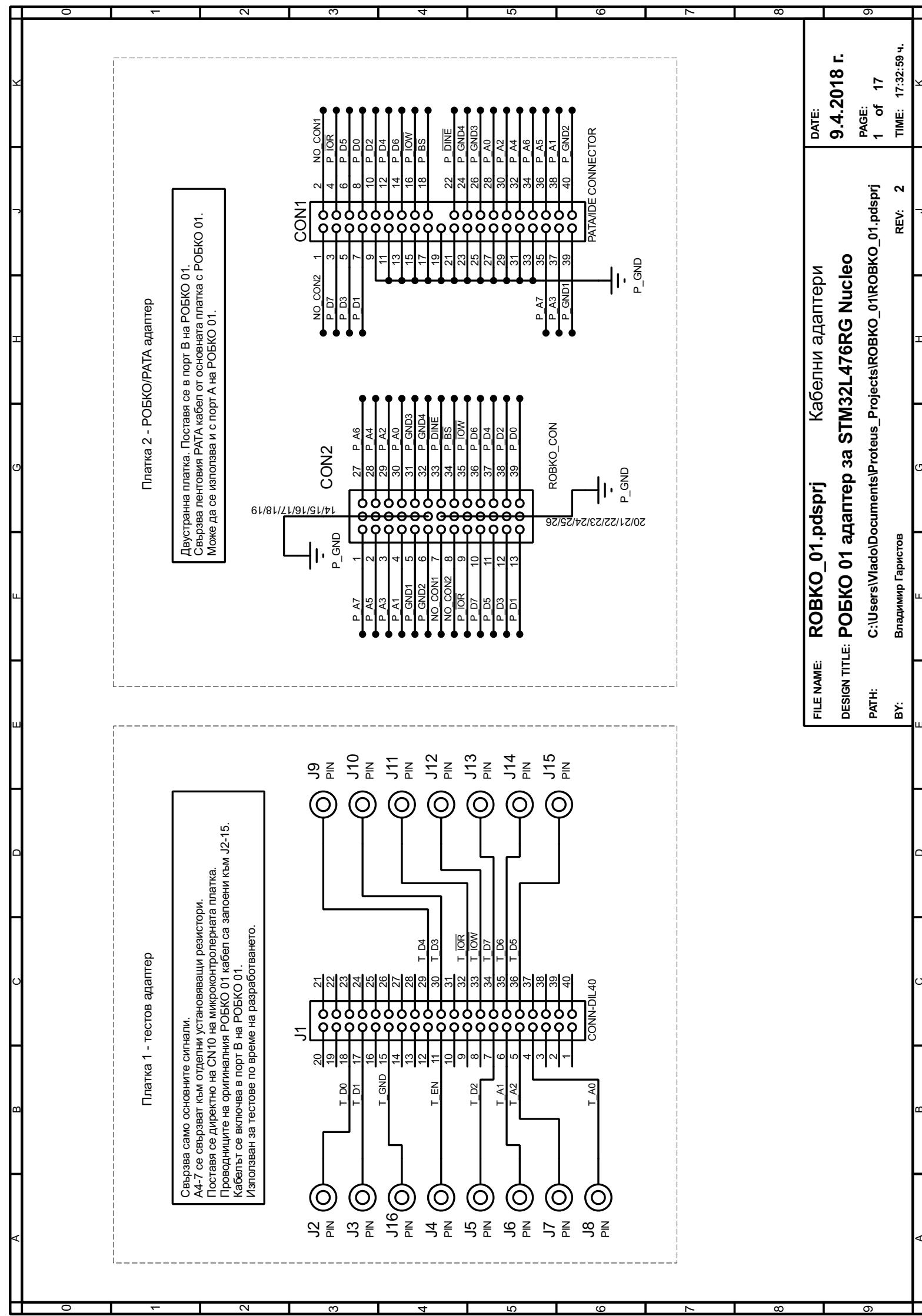
## 3.1 Принципна електрическа схема

### 3.1.1 PATA адаптер и тестов адаптер

Тестовият адаптер свързва оригиналният кабел на РОБКО 01 с микроконтролерната платка. Отделните проводници от кабела са запоени в адаптерната платка и са свързани към стандартен двуредов конектор. Сигналите  $\overline{DINE}$  и  $\overline{BS}$  не са изведени, а сигналите A4-A7 се свързват към външни установяващи съпротивления върху прототипна платка. Този адаптер е използван само за тестване на разработката преди да бъдат произведени другите две платки.

Серията РОБКО разполага със собствен специфичен конектор. Броят пинове на този конектор и на стандартния PATA конектор съвпадат. Това позволява РОБКО 01 да се свързва към управляващо устройство през стандартен PATA лентов кабел с подходящ адаптер. Това е функцията на втората платка от настоящата система. Всеки проводник от РОБКО конектора е свързан със съответен проводник от PATA конектора. Изключение прави средният ред на РОБКО конектора. Тези проводници са свързани заедно и към същия брой пинове в PATA конектора. В различните портове на РОБКО 01 те са свързани или към земя или към захранване, но винаги всички заедно.

РОБКО/PATA адаптерът е двустранна платка. От едната страна е PATA конекторът, а от другата РОБКО конекторът или три стандартни едноредови конектори. Поставя се директно в порт B на РОБКО 01. Може да се използва и за порт A на РОБКО 01.



### 3.1.2 Захранващ блок

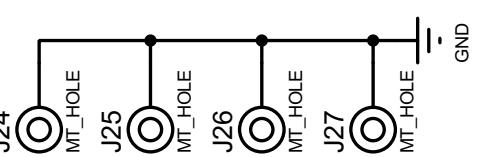
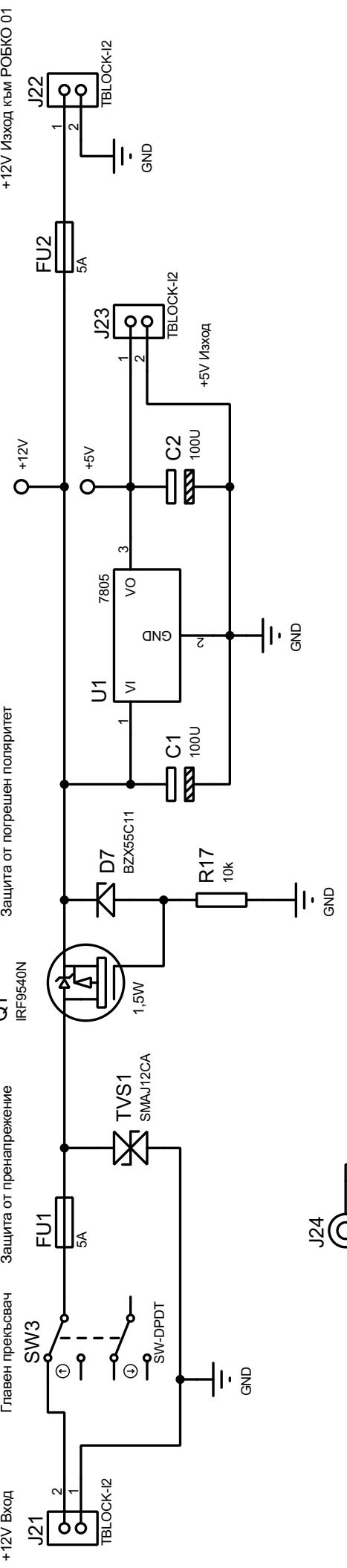
Захранващият блок е част от основната платка. SW3 е главният прекъсвач. През него преминава захранващият ток за РОБКО 01 и за микроконтролерната платка. Схемата черпи енергия от 12-волтов източник, свързан към конектор J21. Конектор J22 подава напрежение на РОБКО 01, а конектор J23 на оптичния сензорен хващащ. Микроконтролерната платка се захранва през конектор J17, показан на схемна страница "Конектори и индикация". През същия конектор основната платка получава 3,3V захранващо напрежение от линеен регулатор на микроконтролерната платка. С 3,3V се захранват джойстиците.

Захранването включва три защити - против късо съединение, против пренапрежение и против грешен поляритет на напрежението. Защитата от късо съединение се реализира с два 5-амперови бушона. Първият от тях обхваща РОБКО 01 и микроконтролерната платка, а другият само робота. Бушоните са бавнодействащи с цел да се избегне изгарянето им при кратки токови импулси, породени от индуктивния характер на моторите. Защитата против пренапрежение се осъществява чрез двупосочен трансил с номинално пробивно напрежение 13,3 V. Защитата от грешен поляритет на захранващото напрежение се реализира от P-канален MOSFET Q1, цепнеров диод D7 и резистор R17. При правилен поляритет на напрежението преминаващият ток през паразитния диод на Q1. D7 и R17 образуват делител на напрежение. Напрежението гейт-сорс на Q1 става равно на пробивното напрежение на D7 и транзисторът се отпушва. Ако на схемата се подаде отрицателно захранващо напрежение, диодът на Q1 се явява свързан в обратна посока и транзисторът остава запущен.

Микроконтролерната платка и оптичният сензорен хващащ се захранват от U1 - линеен регулатор на напрежение 7805 с максимален ток 1,5A. Двете захранващи напрежения +12V и +5V се стабилизират от един  $100\mu F$  кондензатор. Монтажните отвори са заземени. През металните болтове и отстъпите те се свързват електрически с металното шаси на захранващия източник, а през него и със заземявящия проводник на сградната електроинсталация.

Захранващ блок на основната платка

Зашитата от пренапрежение се задейства между 13,3V и 14,7V.  
Булооните са закъснителни.  
Изгарят със закъснение от 150ms до 5s при ток 20A.



Захранване

FILE NAME: ROBKO\_01.pdsprj  
DESIGN TITLE: РОБКО 01 адаптер за STM32L476RG Nucleo  
PATH: C:\Users\Viado\Documents\Proteus\_Projects\ROBKO\_01\ROBKO\_01.pdsprj  
BY: Владимира Гаристов

DATE: 9.4.2018 г.  
PAGE: 3 of 17  
TIME: 17:32:59 ч.

### 3.1.3 Връзка между STM32L476RG и РОБКО 01

Основната платка играе ролята на централен възел, свързващ другите компоненти на системата. Конектори CN7 и CN10 на микроконтролерната платка се свързват съответно с конектори J17 и J20 от основната. Конектори J17 и J20 се намират на страна спойки с цел микроконтролерната платка да се поставя под основната. Двете платки се прикрепят една към друга и към корпуса на захранващия източник с помощта на метални отстъпи и болтове. В конектор CON3 на основната платка се поставя единия край на лентовия PATA кабел към РОБКО 01. Резисторите R1-R4 установяват адресните сигнали A4-A7. Конектори J18 и J19 се свързват с USB-UART конверторната платка, която се свързва с компютъра изпълняващ сървърния софтуер.

### 3.1.4 Индикация

Светодиодите D1-D6 предоставят визуална индикация на състоянията на стъпковите мотори. Диод D1 съответства на мотор 0, D2 на мотор 1 и т.н. Номерата на моторите са обозначени на ситопечата от страна компоненти на платката и в таблица 2.2. Светодиодите са по два в корпус - зелен и червен. Зеленият цвят означава движение напред, а червеният - движение назад. При неподвижен мотор съответните светодиоди не светят.

### 3.1.5 Ръчно управление

Джойстиците за ръчно управление се описват подробно в точка 2.3.6. Кабелите им се свързват към конектори RJ1 и RJ2. За определяне на ориентацията на джойстиците се използва вградения в микроконтролера аналогово-цифров преобразувател. Различните режими на управление се конфигурират от прекъсвачите SW1.

Първият прекъсвач (обозначен с единица върху корпуса на компонента; между пинове 4 и 5 на схемата) конфигурира скоростта на въртене на моторите. Когато е отворен е избрана бавна скорост, а когато е затворен - бърза. Чрез вторият прекъсвач се избира дали стъпковите мотори да се задвижват в режим на пълна стъпка (отворен) или на полустанция (затворен). Третият прекъсвач се взима под внимание само ако четвъртият е отворен. Чрез него се избира режима на управление, като отворен

съответства на отдалечено управление, а затворен на ръчно. Последният прекъсвач включва възможността на микроконтролера да засича дали двата джойстика са свързани към платката и ако са, автоматично да премине в ръчен режим. При отворен прекъсвач тази функционалност е изключена. Бутонаят SW2 рестартира микроконтролера.

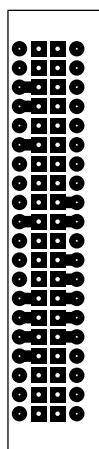


## 3.2 Графични оригинали на печатните платки

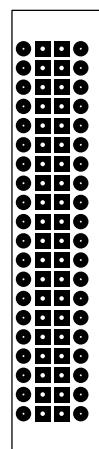
Всички графични оригинали са в отношение 1:1 спрямо физически реализираните печатни платки.

Тестовият адаптер е ръчно изработен на базата на фабрично пробита и метализирана прототипна платка. Другите две печатни платки са произведени машинно.

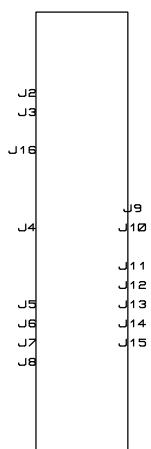
### 3.2.1 Тестов адаптер



Фигура 3.1: Страна спойки



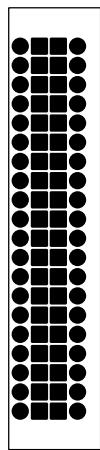
Фигура 3.2: Страна компоненти



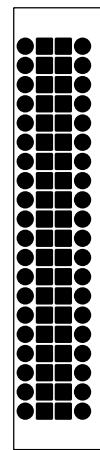
Фигура 3.3: Ситопечат,  
страница спойки



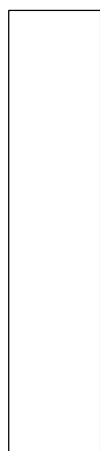
Фигура 3.4: Ситопечат,  
страница компоненти



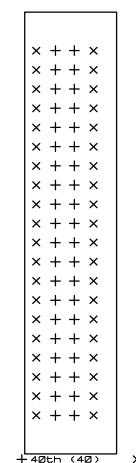
Фигура 3.5: Изолационен лак,  
страна спойки



Фигура 3.6: Изолационен лак,  
страна компоненти

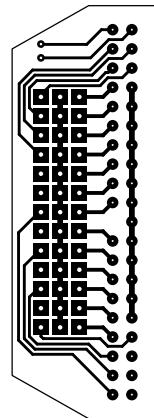


Фигура 3.7: Очертание  
на печатната платка

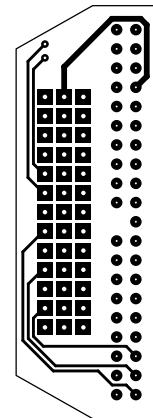


Фигура 3.8: Отвори

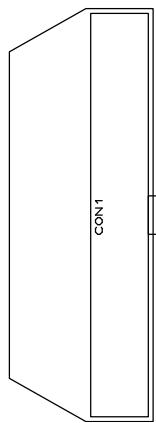
### 3.2.2 PATA адаптер



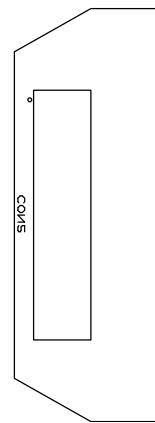
Фигура 3.9: Страна компоненти



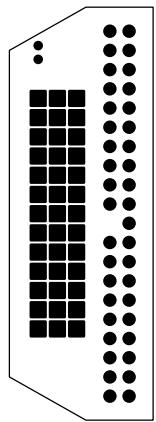
Фигура 3.10: Страна спойки



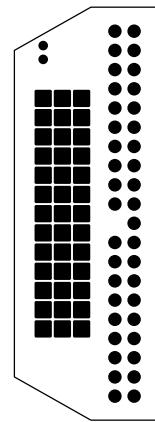
Фигура 3.11: Ситопечат,  
страна компоненти



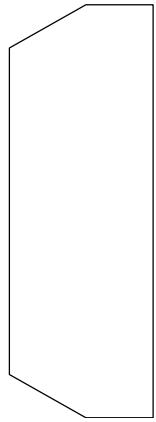
Фигура 3.12: Ситопечат,  
страна спойки



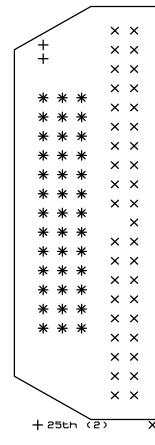
Фигура 3.13: Изолационен лак,  
страна компоненти



Фигура 3.14: Изолационен лак,  
страна спойки

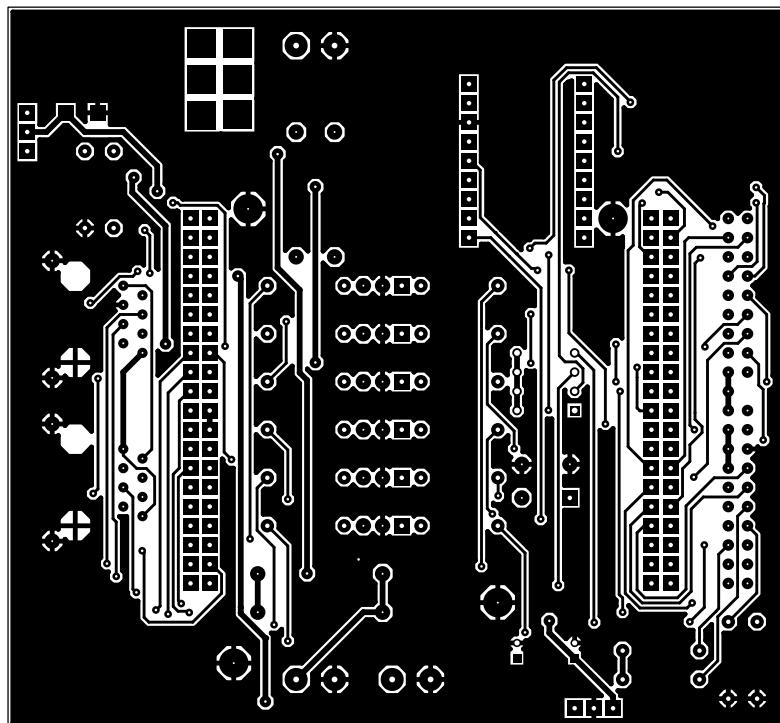


Фигура 3.15: Очертание на  
печатната платка

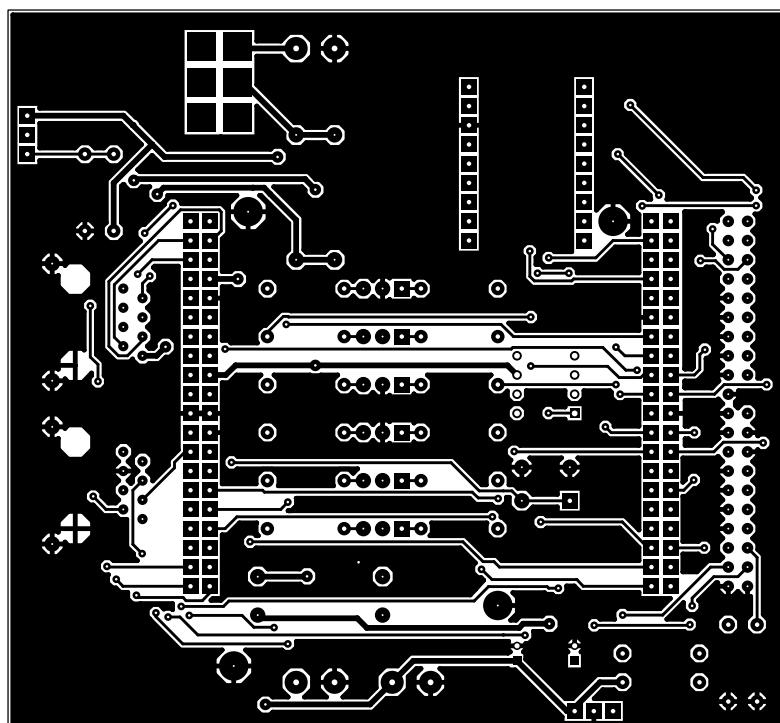


Фигура 3.16: Отвори

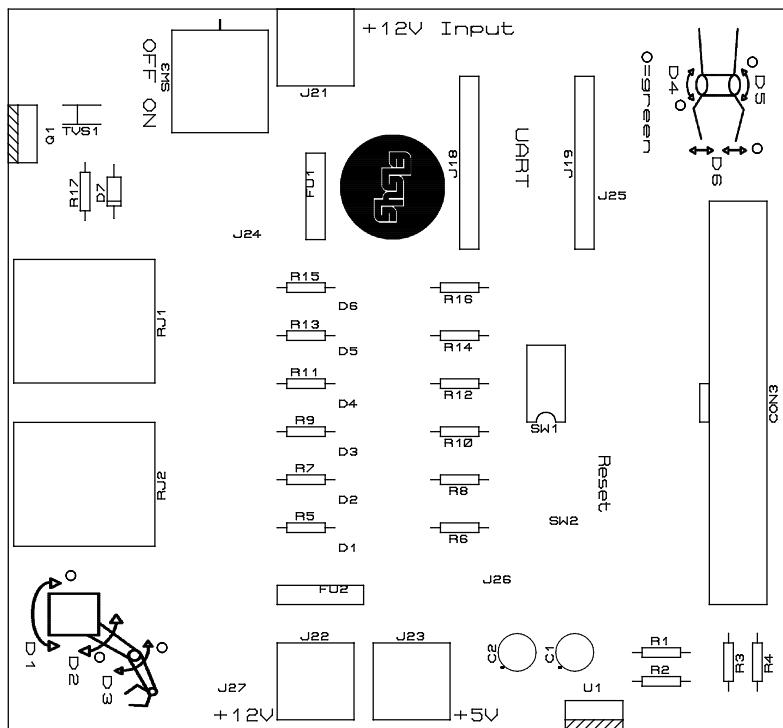
### 3.2.3 Основна платка



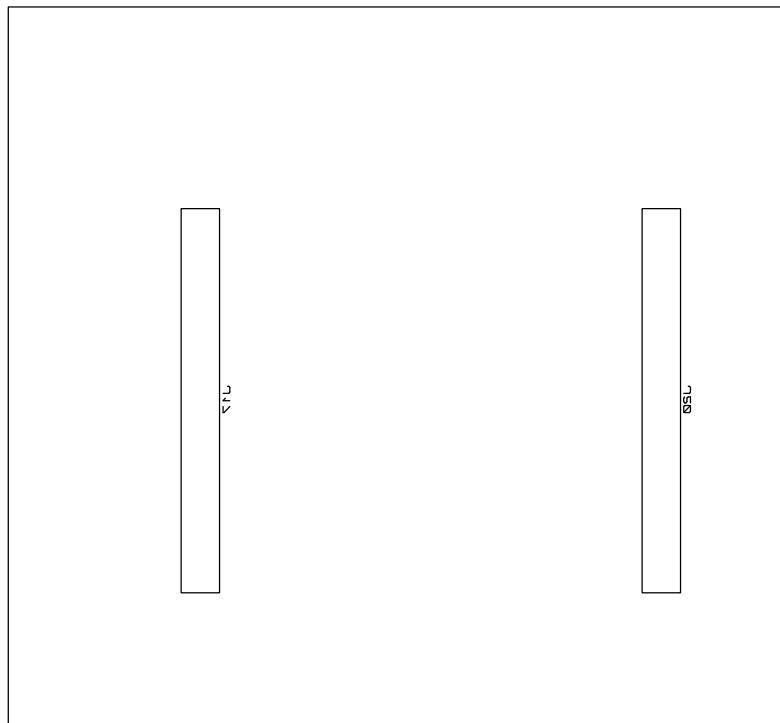
Фигура 3.17: Страна компоненти



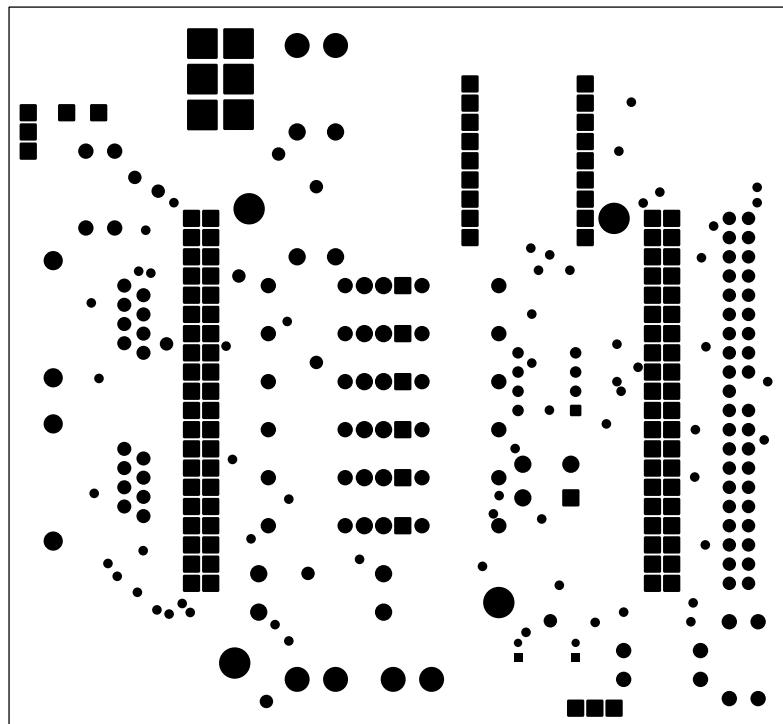
Фигура 3.18: Страна спойки



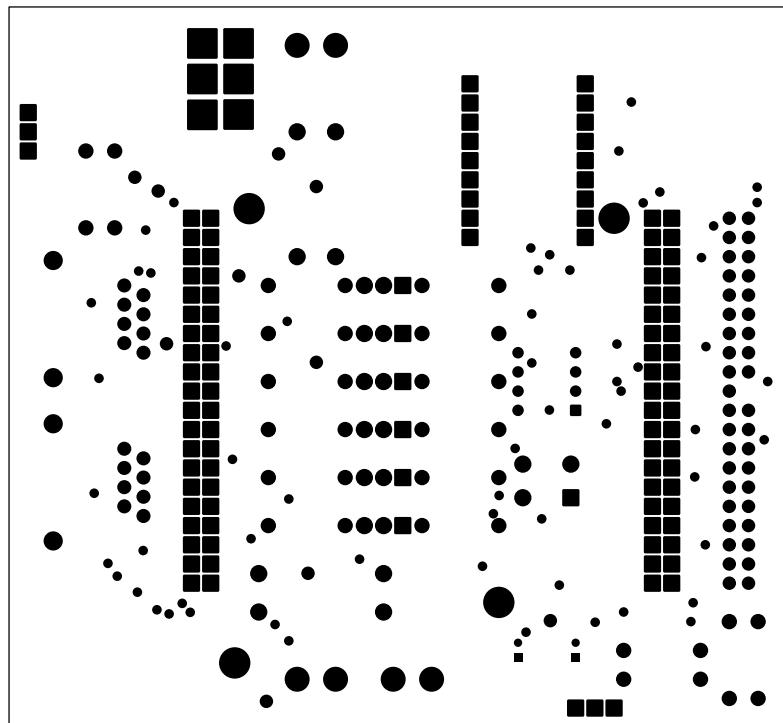
Фигура 3.19: Ситопечат, страна компоненти



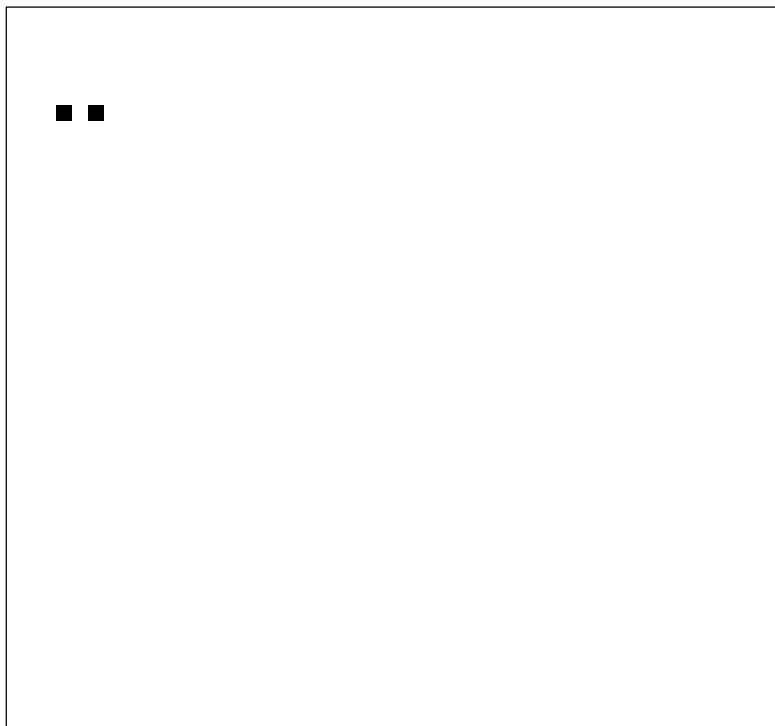
Фигура 3.20: Ситопечат, страна спойки



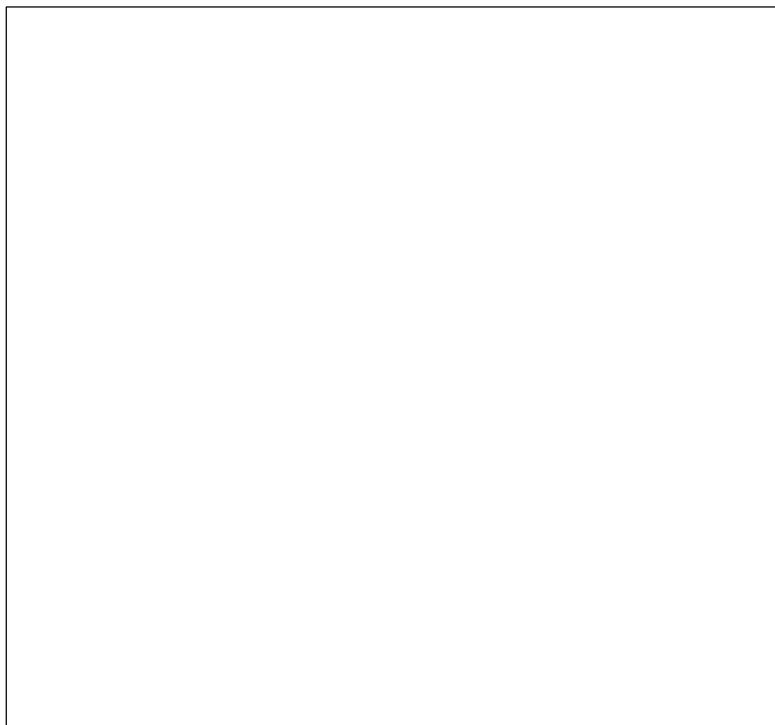
Фигура 3.21: Изолационен лак, страна компоненти



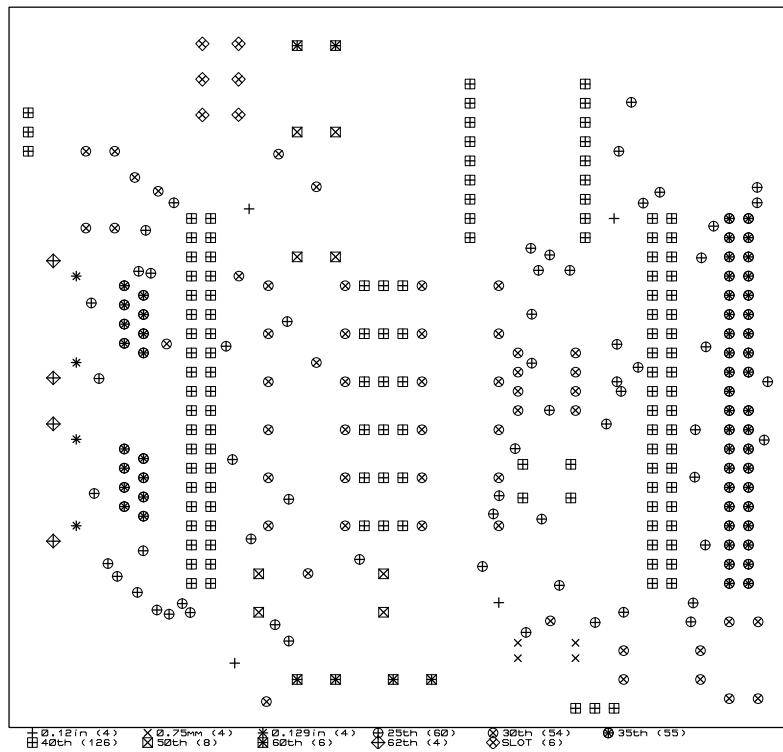
Фигура 3.22: Изолационен лак, страна спойки



Фигура 3.23: Припой, страна компоненти



Фигура 3.24: Очертание на печатната платка



Фигура 3.25: Отвори

# Софтуер за отдалечно управление

## 4.1 Формат на командите за отдалечно управление

Отдалеченото управление се извършва посредством текстовите команди, описани в таблица 4.1. Командите се идентифицират по номер, когато се изпращат от клиента до сървъра, и по име, когато се въвеждат от команден ред или във файл. Командите, получени от сървъра, се изпращат към микроконтролера, където се записват в опашка и изпълняват.

Движението на робота се задава от командите MOV и MOVE, съответно за движение на един или на множество мотори. Команда MOV приема два параметъра - номер на мотор (виж таблица 2.2) и брой стъпки, които да изпълни. Команда MOVE приема шест параметъра - брой стъпки за изпълнение от всеки мотор, в нарастващ ред на номерата на моторите. Тази команда позволява едновременното движение на повече от един мотора. Отрицателен брой стъпки се използва за движение в обратна посока (виж таблица 2.2).

Команда OFF изключва захранването на мотора, чийто номер е подаден като параметър. За изключване на всички мотори се въвежда команда OFF със стойност на параметъра 6. Използва се когато е нужно ръчно преместване на механичните части на робота. Команда FREEZE поставя робота в режим на изчакване - прекратява изпълнението на опашката от команди. Команда OPTO се използва за засичане на предмети с оптичния сензорен хващащ. Когато е изпратена команда OPTO с параметър 1 и по време на изпълнението на следващата MOV или MOVE команда бъде засечен предмет се прекратява текущата команда и се преминава към следваща. Команда OPTO с параметър 2 има същото поведение, с тази разлика, че командалата за движение се прекратява, когато сензорът спре да засича предмет. Командата SET\_STEP променя режима на стъпка на моторите. Единственият ѝ параметър има три валидни стойности: 0, 1 и

№	Име	Пояснение	Параметри		
			8	16	24
1	MOV	Определен мотор извършва даден брой стъпки	мотор	стъпки	
2	MOVE	Всички мотори извършват даден брой стъпки	стъпки x6		
3	OFF	Изключва определен мотор	мотор		
4	OPEN_FILE	Изпълнява файл, записан на сървъра		име на файла променлива дължина	
5	GOTO_POS	Запазена за бъдещо развитие	позиция x6		
6	GET_POS	Запазена за бъдещо развитие			
7	KILL	Аварийно спиране и изключване на всички мотори			
8	CLEAR	Изчистване на опашката от команди			
9	FREEZE	Спира изпълнението на команди			
10	RESUME	Възобновява изпълнението на команди			
11	SAVE_POS	Запазена за бъдещо развитие			
12	OPTO	Прекратяване на следващата команда, при сигнал от оптичния сензор	режим		
13	SET_STEP	Избор на движение с цяла стъпка или с полуистъпка	стъпка		
14	SET_SPEED	Настройка на скоростта на стъпките	време		

Таблица 4.1: Команди за отдалечено управление и техните параметри

2. Те означават съответно: да се използва режимът, избран от SW1 (виж 3.1.5), полуистъпка и пълна стъпка. Команда SET\_SPEED задава периода от време, който се изчаква след всяка стъпка. Параметърът й е времето в милисекунди или 0 за да се използва стойността, зададена чрез SW1.

Изброените до тук команди се записват в опашка в паметта на микророконтролера и се изпълняват в реда, в който са получени. Следващите команди са приоритетни - изпълняват се веднага при получаването им.

Такава команда е KILL. Използва се за аварийно изключване на робота. Не приема параметри, прекратява изпълнението на текущата команда и изключва всички мотори. Текущата команда не се изтрива и

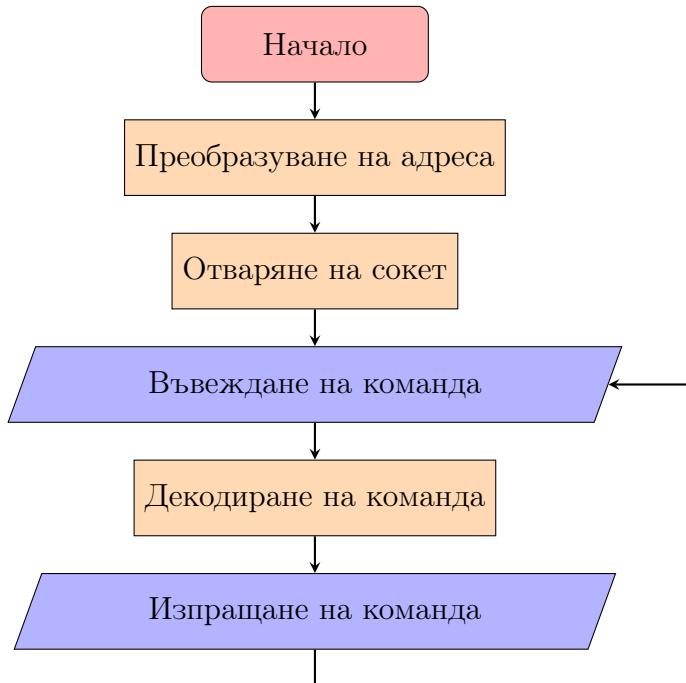
може да бъде възстановена. Команда RESUME също е приоритетна. Тя възстановява изпълнението на команди, спряно чрез FREEZE или KILL. Команда CLEAR изтрива цялата опашка от команди, включително и текущо изпълняваната команда.

Командата OPEN\_FILE е уникална с това, че не се изпълнява от микроконтролера, а от сървъра. Като параметър приема име на файл, чието съдържание да се изпрати от сървъра към микроконтролера. Файлът трябва да съдържа команди в описания тук формат. Ако файлът не се намира в папката, където е изпълнимия файл на сървъра, трябва към името на файла да се добави път спрямо тази папка.

Командите GOTO\_POS, GET\_POS и SAVE\_POS са запазени за бъдещо развитие.

## 4.2 Клиент

Клиентската програма се използва от потребителя за изпращане на команди към сървъра. Състои се от файловете robko\_client.c, robko\_decode.c и съответните заглавни файлове. Програмата не разполага с графична среда и се изпълнява от команден ред. IP адреса на сървъра се подава като единствен параметър. На фигура 4.1 е показана блок схема на програмата.



Фигура 4.1: Блок схема на клиентската програма

### 4.2.1 TCP сокет

Програмата започва с преобразуване на IP адреса на сървъра от текстов низ към тип struct in\_addr. Отваря се TCP сокет към сървъра на порт 55321 чрез функцията connect(), предоставена от Berkeley sockets API. Файловият дескриптор на сокета се създава предварително от функцията socket(), а адресът и портът се задават от функцията bind().

Останалата част от алгоритъма се изпълнява в безкраен цикъл. Потребителят въвежда команда като текст в командния ред, разделяйки името на командата и параметрите с интервал. Програмата прочита въведения текстов низ от стандартния вход чрез функцията fgets(). Командата се декодира и изпраща по TCP сокета към сървъра с функцията write().

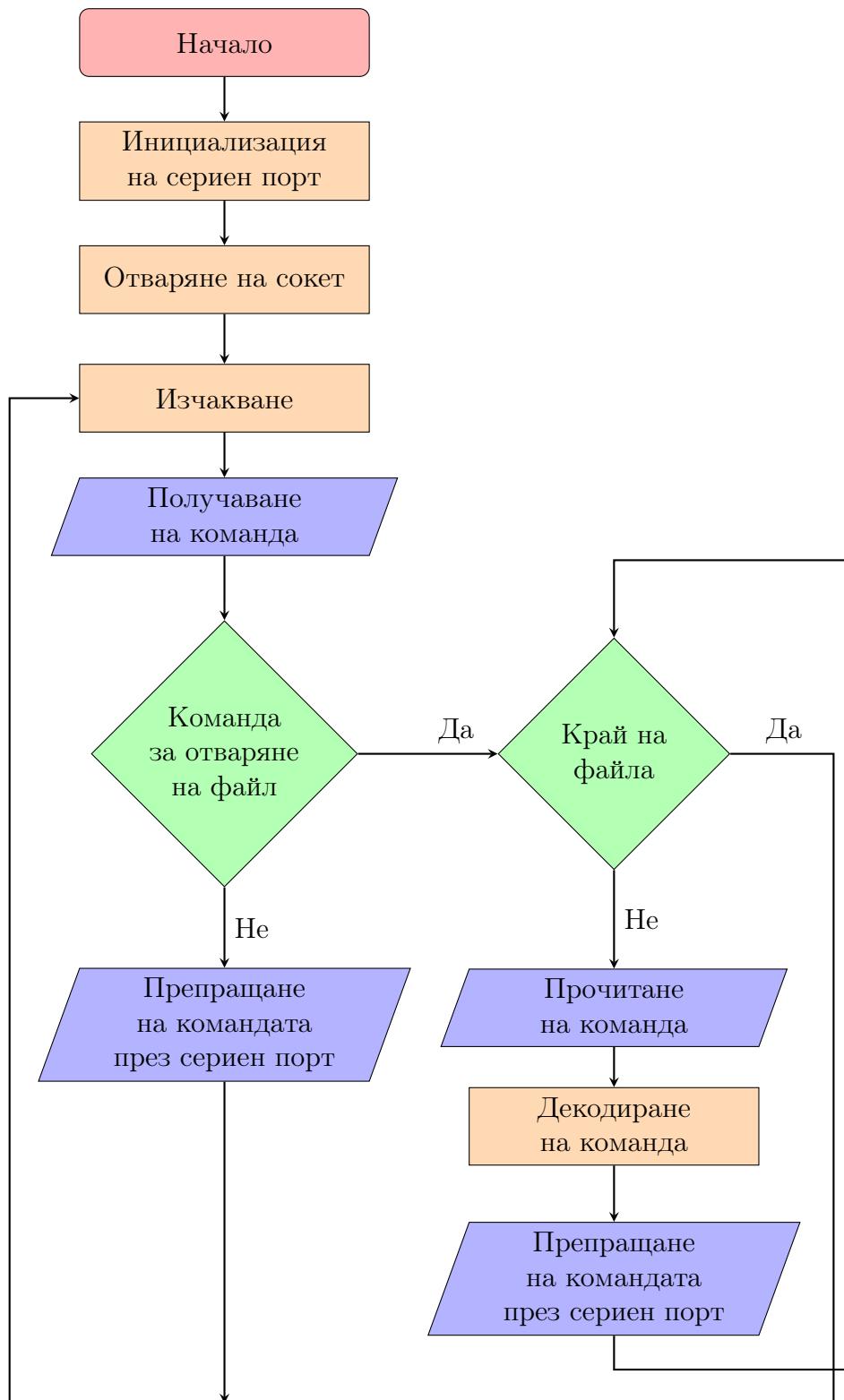
### 4.2.2 Декодиране на команди

Декодирането на команди се извършва от функцията decode\_cmd(), дефинирана в robko\_decode.c. Приема два параметъра - указател към въведения от потребителя текстов низ и указател към началото на uint8\_t масив, в който се записва декодираната команда. Разпознаването на командата се извършва чрез последователно търсене на валидните имена на команди във въведения от потребителя низ чрез функцията strstr(). При съвпадение се извършват следващите стъпки. На нулева позиция в масива за декодирана команда се записва дължината на декодираната команда в байтове, включително байта за дължина. На първа позиция се записва номерът на командата. На следващите позиции се записват аргументите, ако има такива. Аргументите на някои команди са от тип int16\_t и заемат по две позиции от масива. Размерите на параметрите в битове са дадени в таблица 4.1. Преобразуването на параметрите от текст към целочислени типове се извършва от функцията strtol() и изрично конвертиране.

## 4.3 Сървър

Сървърната програма приема командите по TCP сокета и ги препраща към микроконтролера през сериен порт. При получаване на команда за изпълнение на файл прочита командите от съответния файл и ги изпраща към микроконтролера. Програмата се състои от файловете robko\_server.c,

`robko_decode.c` и съответните заглавни файлове. На фигура 4.2 е показана блок-схема на програмата.



Фигура 4.2: Блок схема на сървърната програма

### 4.3.1 Серийна комуникация

Сървърната програма използва библиотеката `libserialport` за комуникация с микроконтролера през сериен порт върху USB. Името на порта се избира чрез стойността на макроса `SERIAL_PORT` във файла `robko_server.c`. В началото на програмата серииният порт се инициализира и настройва със следните параметри: осембитова дума, проверка за четност (битът за четност е 1, ако в думата има четен брой единици), един стоп бит и скорост 115200 бода.

### 4.3.2 TCP сокет

Създаването на сокет се извършва както при клиента. Вместо функцията `connect()` се използва функцията `listen()`, която отваря сокета за заявки от страна на клиента. Функцията `accept()` изчаква клиентът да се свърже и когато го направи създава нов сокет през който да се обменят данните. Командите, изпращани от клиента, се приемат чрез функцията `read()`.

### 4.3.3 Препращане и изпълнение на команди

Когато се получи команда се проверява дали е `OPEN_FILE`. Ако е такава команда, се отваря съответния файл и една по една, командите от него се прочитат, декодират и изпращат към микроконтролера. За декодиране се използва същата функция, използвана и при клиента (виж 4.2.2). Всички други команди директно се препращат. Първият байт от командата не се изпраща, а посочва дължината на командата в брой байтове. Причината при декодирането да се добавя дължината на командата в началото ѝ е, че трябва да се знае колко байта трябва да бъдат изпратени по серииния порт. Не може да се предават данни до засичането на нулев байт, защото синтаксисът на командите позволява да се използват нулеви байтове в параметрите на командите.

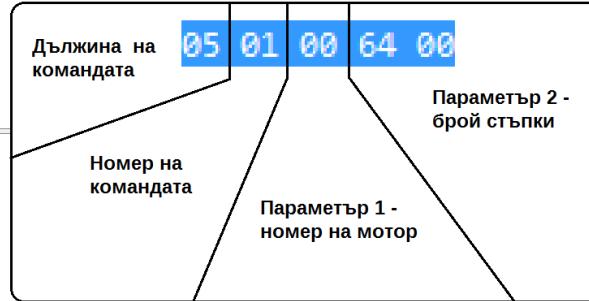
## 4.4 Наблюдение на TCP трафик

Трафикът, обменян между клиента и сървъра, се наблюдава с програмата Wireshark. На фигура 4.3 са показани сегментите, съставящи TCP сесия за управление на РОБКО 01. Първите три сегмента изграждат сесията в съответствие с последователността на three-way handshake. Сегментите с четни номера от 4 до 14 включително съдържат команди, а сегментите с нечетни номера от 5 до 15 включително са потвърждения за успешно получаване на данни. Последните три сегмента завършват сесията.

Сегмент номер 6 е разгледан подробно на фигура 4.3. Първият байт от данните показва, че сегментът съдържа общо 5 байта полезни данни. Вторият байт показва, че изпращаната команда е MOV (виж таблица 4.1). Първият параметър на командата е третия байт. От него става ясно, че командата се отнася за мотор номер 0. Последните два байта съставят втория параметър. Понеже сървърът и клиента се изпълняват на машини с little endian подредба на байтовете, първо се изпращат по-младшите байтове. В този пример вторият параметър има стойност  $0x0064 = 100_{(dec)}$  стъпки.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	192.168.1.135	192.168.1.102	TCP	74	55142 + 55321 [SYN] Seq=0 Win=29204 Len=0 MSS=1460 SACK_PERM=1 Tsva1=4294926074 TSerr=0 VS=64
2	0.0001147999	192.168.1.102	192.168.1.135	TCP	74	55321 + 55142 [SYN ACK] Seq=0 Ack=1 Win=28950 Len=0 MSS=1460 SACK_PERM=1 Tsva1=693489 TSerr=0
3	0.0003461929	192.168.1.135	192.168.1.102	TCP	66	55142 + 55321 [ACK] Seq=1 Ack=1 Win=29248 Len=0 Tsva1=4294926075 TSerr=693489
4	4.931315246	192.168.1.135	192.168.1.102	TCP	69	55142 + 55321 [PSH ACK] Seq=1 Ack=1 Win=29248 Len=0 Tsva1=4294927307 TSerr=693489
5	4.9398161936	192.168.1.102	192.168.1.135	TCP	66	55321 + 55142 [ACK] Seq=1 Ack=1 Win=29056 Len=0 Tsva1=694723 TSerr=4294927308
6	1.21.41758465	192.168.1.135	192.168.1.102	TCP	71	55142 + 55321 [PSH ACK] Seq=1 Ack=1 Win=29248 Len=5 Tsva1=4294956432 TSerr=694722
7	1.21.41758465	192.168.1.102	192.168.1.135	TCP	68	55321 + 55142 [ACK] Seq=0 Ack=9 Win=29056 Len=0 Tsva1=4294956433 TSerr=4294956433
8	1.28.59097292	192.168.1.135	192.168.1.102	TCP	71	55142 + 55321 [PSH ACK] Seq=0 Ack=9 Win=29248 Len=5 Tsva1=4294958223 TSerr=723843
9	1.28.643010406	192.168.1.102	192.168.1.135	TCP	66	55321 + 55142 [ACK] Seq=0 Ack=9 Win=29056 Len=0 Tsva1=4294958223 TSerr=723843
10	4.13.75493132	192.168.1.135	192.168.1.102	TCP	71	55142 + 55321 [ACK] Seq=1 Ack=14 Win=29248 Len=0 Tsva1=725637 TSerr=4294958223
11	4.13.75501446	192.168.1.102	192.168.1.135	TCP	66	55321 + 55142 [ACK] Seq=1 Ack=14 Win=29056 Len=0 Tsva1=798928 TSerr=725637
12	14.09.97071919	192.168.1.135	192.168.1.102	TCP	80	55142 + 55321 [PSH ACK] Seq=19 Ack=1 Win=29248 Len=14 Tsva1=311269 TSerr=796928
13	14.09.97081919	192.168.1.102	192.168.1.135	TCP	66	55321 + 55142 [ACK] Seq=19 Ack=33 Win=29056 Len=0 Tsva1=1045982 TSerr=311269
14	14.30.5206979	192.168.1.135	192.168.1.102	TCP	80	55142 + 55321 [PSH ACK] Seq=33 Ack=1 Win=29248 Len=14 Tsva1=316409 TSerr=1045982
15	14.30.520788	192.168.1.102	192.168.1.135	TCP	66	55321 + 55142 [ACK] Seq=33 Ack=7 Win=29056 Len=0 Tsva1=1051211 TSerr=316409
16	16.51.44764499	192.168.1.135	192.168.1.102	TCP	66	55142 + 55321 [FIN ACK] Seq=41 Ack=1 Win=29248 Len=0 Tsva1=371639 TSerr=1051121
17	16.51.447357	192.168.1.102	192.168.1.135	TCP	66	55321 + 55142 [FIN ACK] Seq=41 Ack=48 Win=29056 Len=0 Tsva1=106351 TSerr=371639
18	16.51.4478711	192.168.1.135	192.168.1.102	TCP	66	55142 + 55321 [ACK] Seq=48 Ack=2 Win=0 Tsva1=371639 TSerr=1106351

Frame 6: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0  
 ▷ Ethernet II, Src: HonHubP\_20:3b:81 (08:76:3f:20:3b:81), Dst: HonHubP\_20:3b:81 (08:76:3f:20:3b:81)  
 ▷ Internet Protocol Version 4, Src: 192.168.1.135, Dst: 192.168.1.102  
 ▷ Transmission Control Protocol, Src Port: 55142, Dst Port: 55321, Seq: 4, Ack: 1, Len: 5  
 ▷ Data (5 bytes)  
 Data: 0x0000006400  
 [Length: 5]



Фигура 4.3: Примерна TCP сесия

# Софтуер на микроконтролера

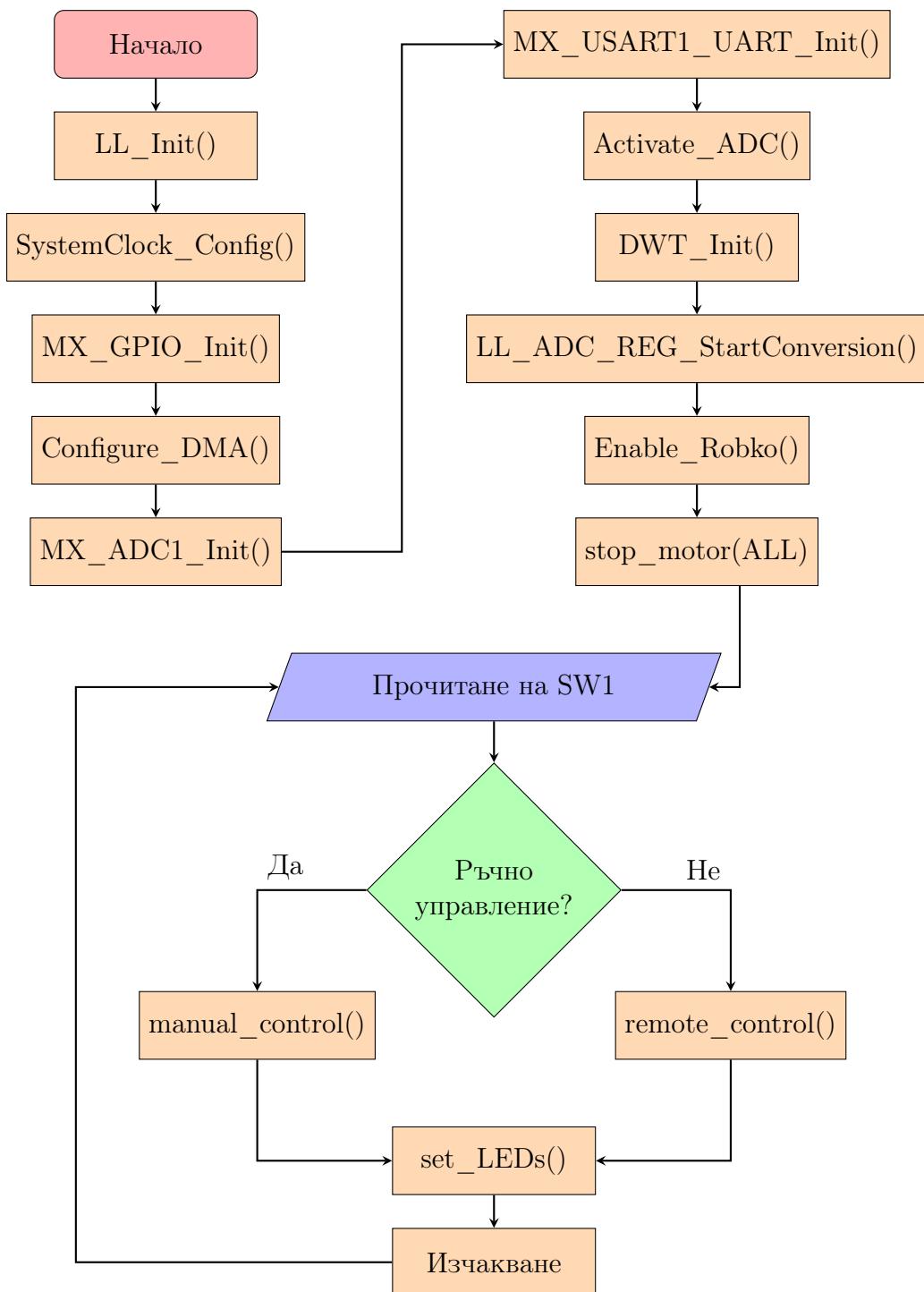
Софтуерът на микроконтролера е изграден на базата на Low-Layer Drivers (LL) и Hardware Abstraction Layer (HAL) библиотеките, предоставени бесплатно от ST Microelectronics. Те, на свой ред, разчитат на Cortex Microcontroller Software Interface Standard (CMSIS) - API на ARM Holdings.

CMSIS отговаря за спецификата на конкретния модел микроконтролер. Дефинира като предпроцесорни макроси адресните пространства на различните модули и дефинира структури от данни, съответстващи на значението на отделните адреси в тези пространства. [8] Това позволява една и съща програма да се изпълнява на различни микроконтролери и освобождава програмиста от бремето да борави с конкретни адреси от паметта. Достъпът до дадедн модул на микроконтролера се извършва по следния начин: Макросът, отговарящ за началото на адресното пространство на този модул, се третира като указател към структура от тип, специфичен за този модул. Полетата на структурата съответстват на регистрите на модула. За миграция на програмата към друг Cortex M4 микроконтролер просто се сменя заглавния файл, съдържащ физическите адреси от паметта.

Библиотеките LL и HAL на STM добавят слой на абстракция над CMSIS и предоставят полезни функционалности. При работа с тези библиотеки програмистът не настройва модулите на микроконтролера на ниво регистри, а подава параметри на библиотечните функции, които извършват операции с регистрите. [6]

## 5.1 Основна функция

При стартиране на контролера първо се инициализират програмният бояч, стековият указател и области от паметта, в една от които се зареждат адресите на функциите, които се изпълняват при прекъсвания. Настройва се основният тактов сигнал с честота 40MHz. Този синхронизиращ сигнал се получава от вътрешен осцилатор с честота 4MHz умножена



по 20 от PLL и разделена на 2. Извършва се инициализация на библиотеката на средата за изпълнение (C runtime library), след което се изпълнява функцията main(), чиято блок схема е показана на фигура 5.1. Тялото на цикълът в долната част на схемата е отделен във функцията check\_mode(), която се извиква в безкраен цикъл в main(). Функциите, чиито имена започ-

ват с MX, и функцията SystemClock\_Config() са генериирани от CubeMX - програма на ST Microelectronics, която генерира инициализиращ код спрямо настройки, въведени в графична среда. Функциите ADC\_Init() и Configure\_DMA() са взети от примерни програми, предоставени безплатно от ST Microelectronics, и променени спрямо нуждите на текущата разработка.

main() функцията се състои от инициализиращи и конфигуриращи функции и основен цикъл. Тази функция никога не трябва да завърши. Ако това все пак се случи, контролерът влиза в безкраен цикъл, който не изпълнява други инструкции.

Първата от инициализиращите функции е LL\_init(). В нея се настройват приоритетите на различните прекъсвания. В програмата се използва само прекъсването, генерирано от получаване на дума в USART модула. Функцията, която се изпълнява при получаването на това прекъсване, се разглежда в точка 5.3

Функцията SystemClock\_Config() настройва синхронизиращите сигнали. Повтаря вече направената конфигурация на основния тактов сигнал и го подава на периферните модули. Конфигурира 8MHz тактов сигнал за аналогово-цифровия преобразувател (АЦП), получен от вътрешния осцилатор с честота 4MHz умножена по 16 от PLL и разделена на 8.

MX\_GPIO\_Init() настройва режимите на пиновете. Пиновете, свързани към светодиодите LED0÷LED11 (виж принципната схема "Конектори и индикация" към глава 3), A0÷A2, ENABLE (A3), D0÷D3,  $\overline{IOR}$  и  $\overline{IOW}$  са изходни. По подразбиране пинове A0÷A2,  $\overline{IOR}$  и  $\overline{IOW}$  се установяват във високо ниво, а другите в ниско. Пинове D4÷D7 и пиновете, свързани с джойстиците са входни. Пиновете, свързани към SW1, също са входни и използват вградените в микроконтролера установяващи съпротивления.

Configure\_DMA() функцията настройва блока за директен достъп до паметта (DMA). Когато аналогово-цифровият преобразувател завърши конвертирането на входен сигнал се подава заявка към DMA модула. Стойността на изходния регистър на АЦП се копира от DMA на адрес в RAM, съответстващ на променливата pot\_vals (масив от четири 16-битови естествени числа). След всяка заявка адресът в паметта (елементът от масива) се инкрементира. След четвъртата заявка се връща на нулевия елемент от масива. Всичко това се извършва без намеса от страна на изчислителното ядро.

Следващата функция конфигурира параметрите на АЦП. Микроконтролерът разполага с 3 АЦП модула с общо 29 канала (някои от каналите са общи). Използват се четири канала от първия АЦП модул. Пиновете,

свързани към потенциометрите, се настройват като аналогови входове. Използва се максималната разредност от 12 бита. Периодът на дискретизация е 92,5 пъти по-голям от периода на синхронизиращия сигнал.

$$T_s = 92,5 \frac{1}{f_c} = 92,5 \frac{1}{8 \cdot 10^6} = 11,5625 \cdot 10^{-6} s \approx 11,5 \mu s \quad (5.1)$$

$$f_s = \frac{1}{T_s} \approx 86,5 kHz \quad (5.2)$$

Относително ниската честота на дискретизация е избрана заради времето, нужно за зареждането на кондензатора във входната верига на АЦП. Капацитетът на този кондензатор е 5pF [9] и се зарежда през потенциометрите в джойстиците. Съпротивленията на потенциометрите зависят от позициите им, но в най-лошия случай са  $1M\Omega$ . Това дава максимална времеконстанта:

$$\tau = R \cdot C = 1 \cdot 10^6 \cdot 5 \cdot 10^{-12} = 5 \cdot 10^{-6} = 5 \mu s \quad (5.3)$$

АЦП модулът се конфигурира да обработва канали от 1 до 4 в нарастващ ред, повтарящи тази поредица постоянно. Към пиновете, на които са изведени четирите канала, се свързват потенциометрите на джойстиците.

Функцията за инициализация на USART модула го настройва в асинхронен (UART) режим със скорост 115200 бода. Използва се един стоп бит, думата е осмебитова. Към нея се добавя един бит за четност, който е 1, ако в думата има четен брой единици. Пиновете RX и TX се настройват в алтернативен режим, което ги свързва към USART модула.

Функцията `Activate_ADC()` извършва автоматична калибрация на АЦП модула и изчаква за завършването ѝ. След това активира АЦП модула и проверява дали успешно се е задействал.

Следващата функция инициализира Data Watchpoint Trigger модула. Този блок предоставя функционалности, предназначени за дебъгване. Сред тях е брояч, инкрементиращ се при всеки такт на ядрото. [2] Този брояч се използва от функцията `DWT_Delay()` за засичане на време в  $\mu s$ .

След извикването на функцията `LL_ADC_REG_StartConversion()` АЦП започва да извършва преобразувания според вече настроена последователност от канали  $1 \div 4$ .

Функцията `Enable_Robko()` подава разрешаващ сигнал към драйверната платка на робота като поставя пин ENABLE (A3) във високо ниво.

Регистрите на драйверната платка на робота се нулират чрез функцията `stop_motor()` с параметър ALL. С това приключва инициализиращата част на програмата.

Основният цикъл започва с прочитане на състоянията на SW1. От тях зависят локалните настройки за режим пълна/полустъпка и скорост (виж 3.1.5), но командите за отдалечно управление, настройващи тези параметри, имат по-голяма тежест. Според състоянията на SW1 се определя дали да се извика функцията `manual_control()` за ръчно управление или `remote_control()` за отдалечно. След това се изпълнява функцията `set_LEDs()`, която управлява светодиодите на основната платка и чрез тях показва в каква посока се движат моторите. В края на основния цикъл се изчаква моторите да се преместят до новите си позиции. Скоростта на въртене се регулира чрез продължителността на това изчакване.

## 5.2 Ръчно управление

Функцията за ръчно управление прочита състоянията на пиновете, свързани към бутоните на джойстиците, и ако са натиснати премества съответните мотори с по една стъпка в нужната посока. За управлението на моторите отговаря функцията `step_motor()`, разгледана в точка 5.5. `manual_control()` сравнява резултатите от последните преобразувания на АЦП модула с гранични стойности. Ако някои потенциометри са завъртяни достатъчно (дръжките на джойстиците са наклонени) съответните мотори се преместват с по една стъпка в нужната посока.

## 5.3 Получаване на команди през серийна КОМУНИКАЦИЯ

Опашката от команди е реализирана като свързан списък от структури със следните елементи:

```
11  typedef struct cmd_buffer_t
12  {
13      uint8_t cmd[13];
14      uint8_t incomplete;
15      struct cmd_buffer_t *next_cmd;
16  }
17  cmd_buffer_t;
```

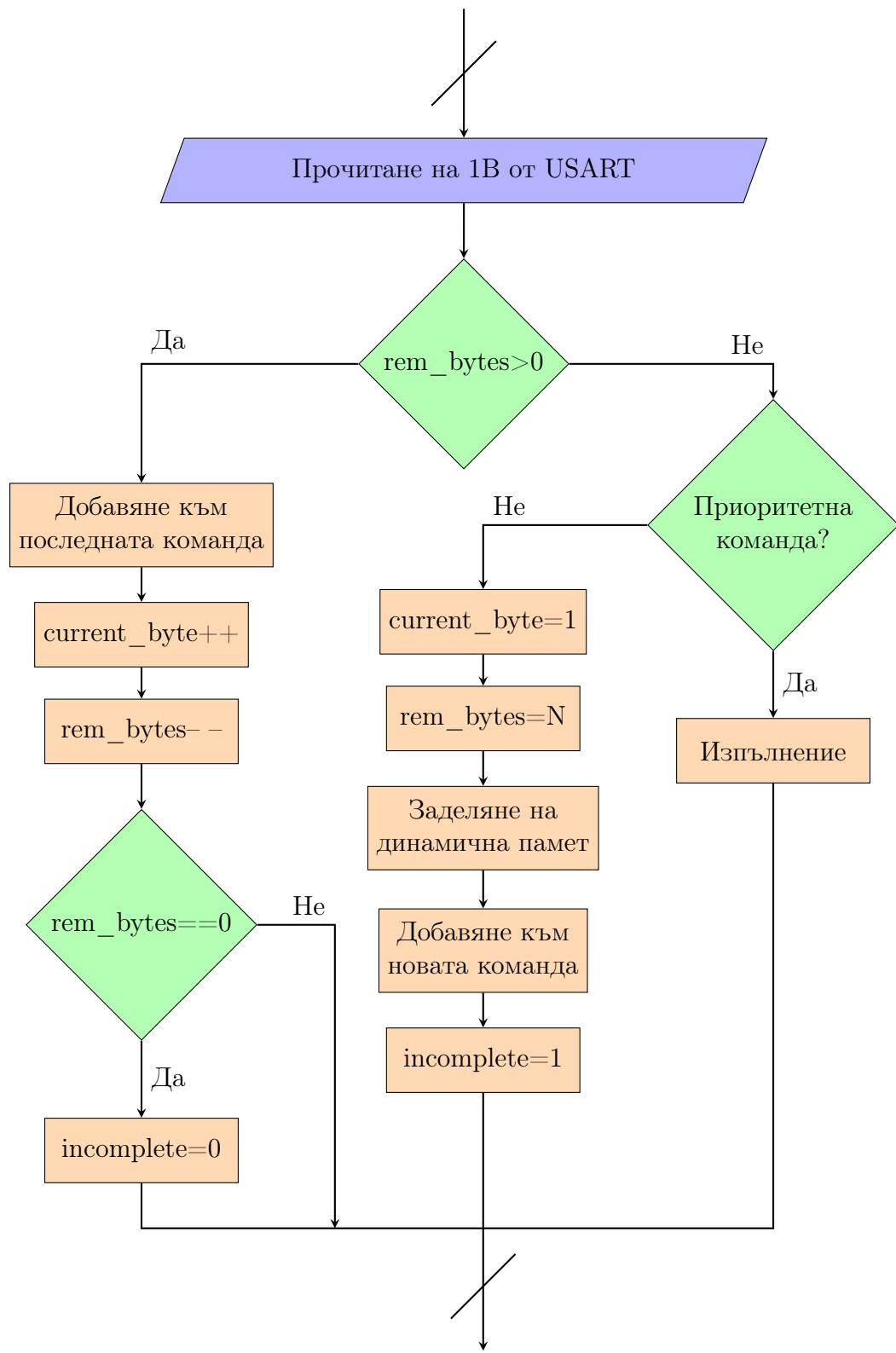
Когато се добавя нова команда към опашката се заделя динамично памет за нея чрез функцията `malloc()`. Масивът `cmd[]` съдържа номера на командата на нулема позиция и параметрите на командата (ако има такива) на следващите позиции. `incomplete` е флаг, който показва, че само част от командата е получена и преди да се пристъпи към изпълнението ѝ трябва да се изчака USART модулът да получи и останалата част от командата. `next_cmd` е указател към следващата команда. Ако команда е последна в опашката, стойността му е `NULL`. Няма указател към предишна команда - свързаният списък е еднопосочен.

При получаване на дума от USART модула се генерира прекъсване, което изпълнява функцията `read_cmd()`. Блок схема на тази функция е показана на фигура 5.2. Функцията започва с прочитане на получена-та осембитова дума. Извършва се проверка дали последната команда в опашката е напълно получена. Ако не е (`rem_bytes > 0`), полученият байт се добавя към последната команда. Броят на получени байтове от по-следната команда (`current_byte`) се инкрементира, а броят на оставащите байтове (`rem_bytes`) се декрементира. Ако `rem_bytes` е стигнал 0, флагът `incomplete` в последната команда се нулира.

Ако `rem_bytes` е 0, това означава, че е получен първият байт от нова команда. Първият байт е номерът на команда и по него се определя каква команда се изпраща. Ако полученият байт съвпада с номера на приоритетна команда, веднага се пристъпва към изпълнението ѝ. Това е възможно, понеже всички приоритетни команди се състоят от само един байт.

Командата за аварийно изключване KILL изключва всички мотори чрез изпълнението на `stop_motor(ALL)`, след което спира разрешаващия сигнал към робота - превключва пин `ENABLE` в ниско ниво. Команда `RESUME` подава разрешаващия сигнал към робота - превключва пин `ENABLE` във високо ниво. При получаването на команда `CLEAR` се обхожда опашката от команди и се освобождава динамично заделената памет, което изтрива цялата опашка.

Ако команда не е приоритетна, трябва да се създаде нов елемент в опашката. `current_byte` получава стойност 1, а `rem_bytes` става равно на оставащите байтове от тази команда. Например за команда `MOVE` `rem_bytes` първоначално е 12, а за команда `OFF` е 1. Виж таблица 4.1 за дълчините на командите. Памет за новата команда се заделя динамично чрез функцията `malloc()` и в нулемата позиция на масив `cmd[]` се записва полученият от USART байт. Ако команда се състои от повече от един байт се вдига флагът `incomplete`.



Фигура 5.2: Блок схема на функция `read_cmd()`

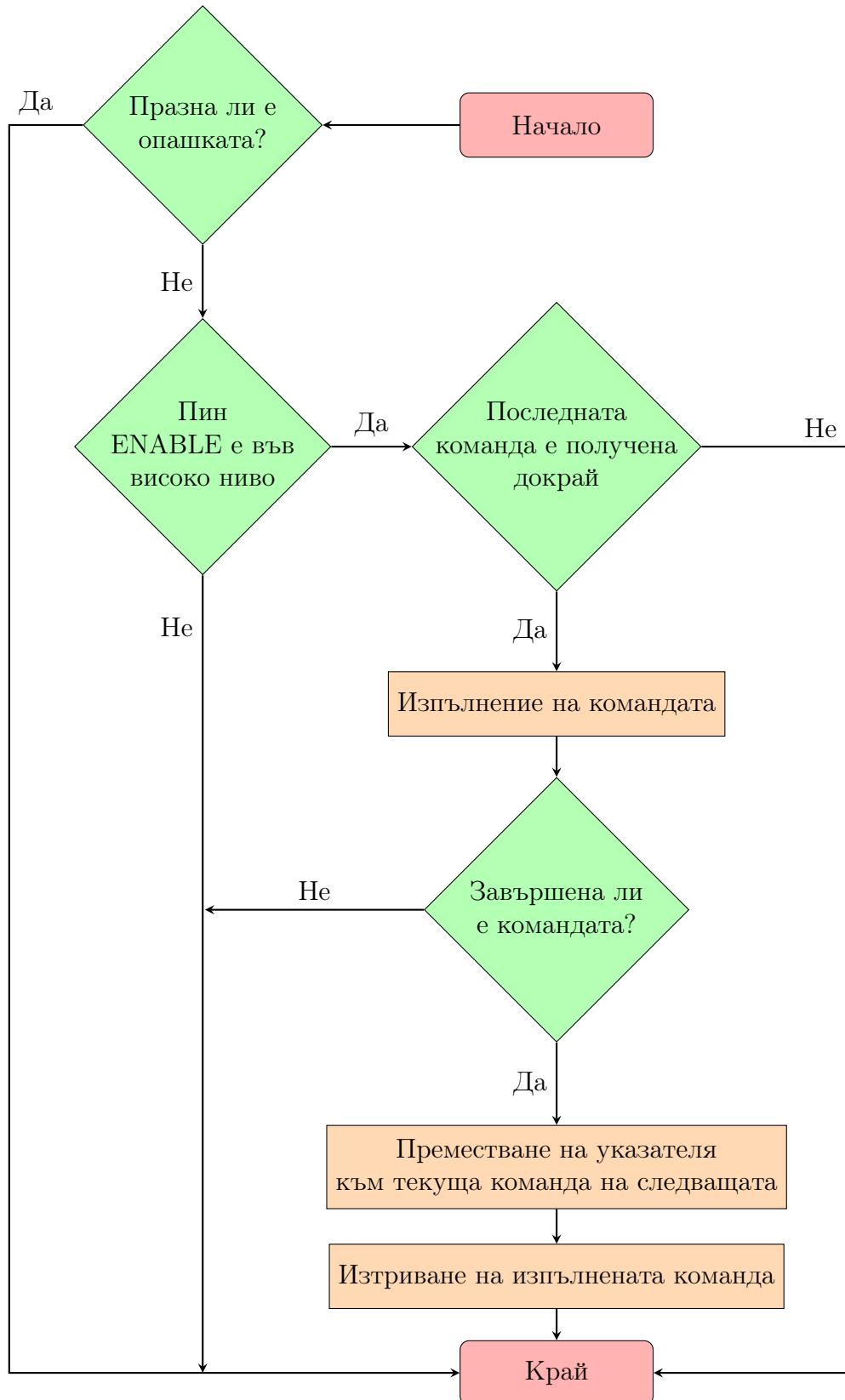
## 5.4 Изпълнение на команди от опашката

Опашката с команди се изпълнява от функцията за отдалечено управление - `remote_control`. Блок схемата ѝ е показана на фигура . Функцията започва с проверки. Ако опашката е празна, не е подаден разрешаващ сигнал към робота или текущата команда не е изцяло получена, функцията завършва без да изпълни команда. В противен случай се преминава към изпълнението на текущата команда - най-старата в опашката. Изпълнението зависи от конкретната команда. Когато текущата команда бъде завършена се вдига флаг. Командите за движение се изпълняват за различен брой извиквания на `remote_control()`, а другите команди за едно извикване на функцията. В края на функцията за отдалечено управление се проверява флага за завършена команда. Ако е вдигнат, указателят към текущата команда се измества с една позиция напред в опашката, а изпълнената команда се изтрива, като заделената ѝ памет се освобождава чрез функцията `free()`.

При изпълнението на команда `MOV` първо се проверява знака на втория параметър на командата. Ако е подаден положителен брой стъпки се извършва една стъпка напред на избрания мотор и броят стъпки се декрементира. При отрицателен брой стъпки се извършва стъпка назад и броят стъпки се инкрементира. Ако броят стъпки стигне 0 се вдига флагът за завършена команда. Чрез функцията `check_opto_flag()`, разгледана в точка 5.6, се проверява дали командата трябва да се прекрати поради сигнал от оптичния сензор и ако трябва, се вдига флагът за завършена команда.

Команда `MOVE` действа подобно на `MOV`. Разликата е, че основният алгоритъм на команда `MOVE` се повтаря шест пъти - по веднъж за всеки мотор. Флагът за завършена команда се вдига когато всички шест параметъра на командата стигнат нула или когато проверката с `check_opto_flag()` прекрати изпълнението на командата.

Изпълняването на следващите команди винаги вдига флага за завършена команда. Команда `OFF` извиква функцията `stop_motor()` с параметър, който ѝ е подаден. Команда `FREEZE` спира разрешаващия сигнал към робота - пин `ENABLE` преминава в ниско ниво. При изпълнението на командите `OPTO`, `SET_STEP` и `SET_SPEED` параметърт им се запаметява в променлива.



Фигура 5.3: Блок схема на функция `remote_control()`

## 5.5 Управление на моторите

Записването на стойности в регистрите на РОБКО 01 се извършва от функциите `stop_motor()` и `step_motor()`. `stop_motor()` приема като параметър номера на мотор, който трябва да бъде изключен, и нулира регистъра му. Извикването ѝ с параметър `ALL` нулира регистрите на всички мотори.

Функцията `step_motor()` придвижва даден мотор с една стъпка. За всеки мотор има брояч, отмерващ колко стъпки е изминал. Таблица 2.1 е въведена в програмата като двуизмерен масив. Функцията `step_motor()` инкрементира или декрементира брояча за избрания мотор според посоката в която трябва да направи стъпка (инкрементира се за права посока). В режим на пълна стъпка се инкрементира/декрементира с 2, а в режим на полустъпка с 1. Стойността на брояча се дели на 8 и се изчислява абсолютната стойност на остатъка. Резултатът е реда от таблица 2.1, който съдържа нужната стойност на регистъра на мотора за да се премести на следващата стъпка. Посоката, в която се придвижва мотора, се записва, за да бъде използвана по-късно от функцията `set_LEDs()`.

Функциите `stop_motor()` и `step_motor()` използват функцията `set_addr()`, която получава като параметър адрес на регистър в робота (номер на мотор) и позволява достъпа до него като настройва пиновете A0÷A2 в нужните състояния.

## 5.6 Засичане на предмети с оптичен сензор

Функцията `check_opto_flag()` прочита третия бит от втория входен буфер на робота (адрес 7,  $\overline{IOR}$  в ниско ниво). Към него е свързан оптичният сензорен хващащ. Според състоянието на оптичния сензор и променливата, съхраняваща последно получения режим на командата ОПТО, определя дали трябва изпълняваната в момента команда да бъде прекратена.

# Изработка на работоспособен модел

Разработването на системата започна с проучване на РОБКО 01 и въстановяване на принципната му схема. Проектирането на основната платка и PATA адаптера протече успоредно с работата по софтуера. Преди завършването на основната платка за връзка между робота и микроконтролера се използваше тестовия адаптер, показан на фигура 6.1.

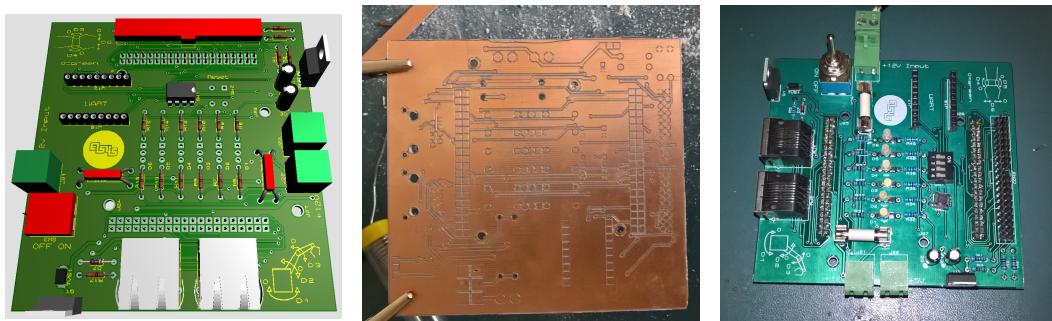
Основната платка е проектирана на CAD програмата Proteus 8, която



Фигура 6.1: Тестов адаптер и микроконтролерна платка

изгражда триизмерен модел на печатните платки (фиг. 6.2а). Прототипът на основната платка (фиг. 6.2б) е произведен с помощта на CNC фреза в София Тех Парк. Чрез него бяха открити някои неточности в проектираната печатна платка. След коригиране на грешките беше произведен по класическа технология втори вариант на платката (фиг. 6.2в).

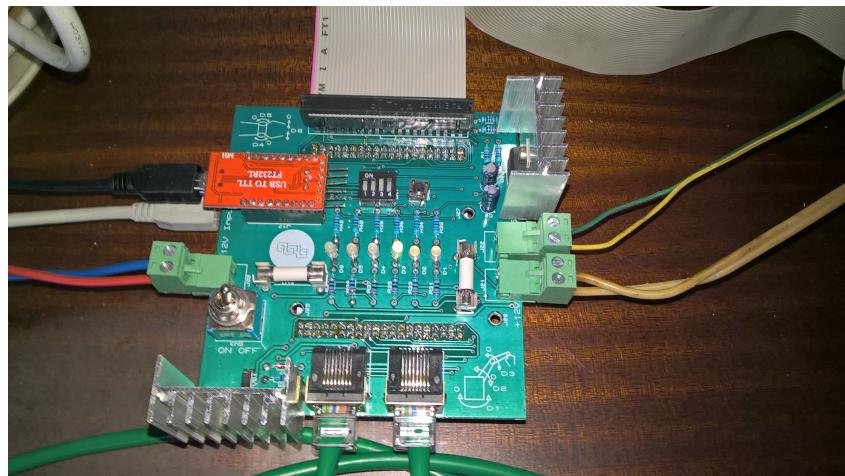
На фигура 6.3 е показано свързването на готовата основна платка.



Фигура 6.2: Основна платка

Микроконтролерната платка се намира под основната. Червено-синият кабел е свързан към захранващия блок, жълто-зеленият кабел е свързан към оптичния сензорен хващащ, а светло кафявият - към захранващия порт на РОБКО 01. Зелените кабели са свързани с джойстиците, а сивият лентов кабел с порт В на РОБКО 01. Черният USB кабел се използва за сериенна комуникация между сървъра и микроконтролера, а сивият USB кабел за програмиране на микроконтролера.

На фигура 6.4 е показана завършената система.



Фигура 6.3: Основна платка, микроконтролер и USB/UART модул



Фигура 6.4: Завършена система

## Заключение

### 7.1 Постигнати резултати

Успешно са реализирани ръчно и отдалечно управление на РОБКО 01. Отдалеченото управление се извършва чрез опашка от текстови команди. Управляващите команди могат да се изпращат по интернет и да се четат от файл. Контролерът е способен да управлява всички мотори на робота едновременно и да засича предмети. Разработената система възстановява оригиналната функционалност на РОБКО 01 и демонстрира възможността за отдалечно управление на индустриски роботи. Проектът е демонстриран в предаването "На Кафе" по Нова Телевизия. [23]



Фигура 7.1: "На Кафе" с туесари

## 7.2 Бъдещо развитие

Разработената система страда от известни недостатъци. Командите между клиента и сървъра се изпращат в чист текст. Това позволява подслушване и неоторизиран достъп до управлението на робота. За да се избегне това към комуникационния протокол трябва да се добавят криптиране и аутентикация.

Друг съществен проблем е употребата на динамична памет в софтуера на микроконтролера. RAM паметта на микроконтролера е само 128KB и може много лесно да бъде препълнена, дори по невнимание. Когато това се случи има опасност данни да бъдат записани в адреси от паметта, които вече се използват. От този момент нататък поведението на микроконтролера става непредвидимо. Този проблем не може да се отстрани без значителни промени по софтуера, но последствията могат да се ограничат с използването на watchdog модула, който рестартира микроконтролера, ако таймерът му не се нулира периодично.

В глава 4 вече се спомена, че се предвиждат три нови команди. Те ще позволяват на робота да запаметява моментната си позиция и да се придвижи до предварително избрани позиции. Голямо подобрение на системата би била способността да изпълнява G код. Това би превърнало РОБКО 01 в CNC машина. Може да се осъществи чрез директно разчитане на G код или чрез преобразуването му в командите, с които работи системата.

Сървърната програма може да се разшири за да комуникира с повече от един клиент и с множество контролери. Тези функционалности ще изискват промени и по комуникационния протокол. Интерфейсите на клиентската и сървърната програма също се нуждаят от подобрения. В по-дългосрочен план може да се разработи графична среда.

## Библиография

- [1] Apple IIc/IIe Joystick Model A2M2002. 2010. URL: <http://www.nightfallcrew.com/02/04/2010/apple-iiciie-joystick-model-a2m2002/> (цитирано на стр. 35).
- [2] ARM. ARM®v7-M Architecture Reference Manual. 2010 (цитирано на стр. 64).
- [3] M. Asadullah and A. Raza. „An overview of home automation systems“. In: 2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI). Nov. 2016, pp. 27–31 (цитирано на стр. 16).
- [4] Industry 4.0: the fourth industrial revolution – guide to Industrie 4.0. 2017. URL: <https://www.i-scoop.eu/industry-4-0/> (цитирано на стр. 17).
- [5] Dave Jones. EEVblog #1054 - How an Analog PC Joystick Works. 2018. URL: <https://www.eevblog.com/2018/02/04/eevblog-1054-how-an-analog-pc-joystick-works/> (цитирано на стр. 35).
- [6] ST Microelectronics. Description of STM32L4 HAL and Low Layer drivers. 2017 (цитирано на стр. 61).
- [7] ST Microelectronics. STM32 Nucleo-64 boards. 2017 (цитирано на стр. 32).
- [8] ST Microelectronics. STM32F3 Series, STM32F4 Series, STM32L4 Series and STM32L4+ Series Cortex®-M4 programming manual. 2017 (цитирано на стр. 61).
- [9] ST Microelectronics. STM32L476xx - Ultra-low-power ARM® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB Flash, 128 KB SRAM, USB OTG FS, LCD, ext. SMPS. 2017 (цитирано на стр. 32, 64).
- [10] ST Microelectronics. STM32L4x5 and STM32L4x6 advanced ARM®-based 32-bit MCUs. 2017.
- [11] Feng Xia. „Internet of Things“. In: International journal of Communication Systems 25 (2012), pp. 1101–1102 (цитирано на стр. 16).
- [12] Огнемир Генчев. Панорама на електронната промишленост на България. Факти и документи. ИК "ДБ Mp", 2003 (цитирано на стр. 9).

- [13] Приборостроителен завод - гр. Правец. Паспорт на персонален компютър Правец 82. Министерство на машиностроенето и електрониката, ДСО "Приборостроене и автоматизация", 1982 (цитирано на стр. 29).
- [14] Джойстик за Правец-8. 2014. URL: <http://www.nesiprav.com/collection/joystick-for-pravetz-8/> (цитирано на стр. 35).
- [15] Орлин Димитров. multi robot controll 2. Технически Университет - Габрово. 2012. URL: <https://www.youtube.com/watch?v=yoRsDOwzwH4> (цитирано на стр. 14).
- [16] Орлин Димитров. Демонстрация на новят безплатен контролер за робко. 2016. URL: <https://www.facebook.com/myrobko01/videos/495895157278674/> (цитирано на стр. 14).
- [17] Завод за медицинска техника ИТКР-БАН. Миниробот РОБКО 01 ръководство за експлоатация. Министерство на машиностроенето, ДСО "Приборостроене и автоматизация", 1986 (цитирано на стр. 20, 21, 30).
- [18] Симеон С. Иванов. ROBKO 01 (CNC). 2013. URL: [https://www.youtube.com/watch?v=DBmUu\\_JXreI](https://www.youtube.com/watch?v=DBmUu_JXreI) (цитирано на стр. 15).
- [19] Симеон С. Иванов. РОБКО 01. 2014. URL: [http://mcu-bg.com/mcu\\_site/viewtopic.php?f=22&t=11875](http://mcu-bg.com/mcu_site/viewtopic.php?f=22&t=11875) (цитирано на стр. 15).
- [20] Валентин Николов. ROBKO 01 controlled via USB. 2011. URL: <https://www.youtube.com/watch?v=0PiUocIpvKk> (цитирано на стр. 14).
- [21] Валентин Николов. Управление на учебен манипулатационен робот - "РОБКО 01" със съвременна компютърна архитектура. 2015 (цитирано на стр. 13).
- [22] Спартак Симеонов. РОБКО 01: Част 2 – История. 2015. URL: <https://obsoletecomputers.wordpress.com/2015/12/31/post198/> (цитирано на стр. 9).
- [23] Нова Телевизия. На кафе - Епизод 142 Сезон 9. 2018. URL: <https://play.nova.bg/programi/na-kafe/931870?autoplay=true> (цитирано на стр. 73).