УПРАВЛЕНИЕ НА УЧЕБЕН МАНИПУЛАЦИОНЕН РОБОТ - “РОБКО 01” СЪС СЪВРЕМЕННА КОМПЮТЪРНА АРХИТЕКТУРА

Валентин Николов
val_niko@yahoo.com

***Резюме**: В доклада се разглеждат възможностите за управление на разработвания в далечното минало учебен, антропоморфен робот “РОБКО 01”, със съвременни хардуерни и софтуерни средства. Разгледани са някои конструктивни особености на манипулационният робот, които имат отношение към разработването на подходящ интерфейс за връзка между съвременен компютър и робота. Представя се реализиран USB контролер и интерактивен софтуер за управлението на отделните стави и допълнителни компоненти посредством мишка, клавиатура и/или джойстик.*

***Ключови думи:** учебни роботи, микроконтролери, програмиране, Labview.*

CONTROL OF EDUCATIONAL MANIPULATOR - “ROBKO 01” WITH MODERN COMPUTER ARCHITECTURE

Valentin Nikolov
val_niko@yahoo.com

***Abstract:** This report examines opportunities for control of developed in the past academic, anthropomorphic robot – “Robko 01” with modern hardware and software tools. Discussed are some technical features of the robot which have relations to develop a suitable interface for connection between modern computer and Robko. Introduced USB controller and developed interactive software for control of individual joints and external connected components via mouse, keyboard and/or joystick.*

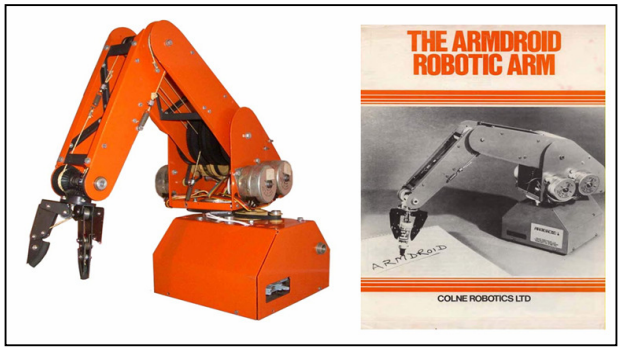
***Keywords:** education robots, microcontrollers, programming, Labview.*

ВЪВЕДЕНИЕ

Робко остава в българската история като първия и единствен учебен, антропоморфен робот. Предпоставки за появата му са положените в България основи в развитието на изчислителната техника, през осемдесетте години на миналия век. Разработката на първия български експериментален микрокомпютър започва през 1979 в “*Институт по техническа кибернетика и роботика*” (ИТКР) към БАН, София. Името му *ИМКО-1* е съкращение от **И**ндивидуален **М**икро **КО**мпютър. Първите бройки са произведени в София през 1980 г., като са произведени около 50 броя. ИМКО-1 е почти пълен аналог на *Apple II Plus*, като липсва флопидисково устройство, тъй като по това време в България подобни устройства не се произвеждат. ИМКО-1 се приема изключително добре за времето си поради сравнително ниската цена, както и заради универсалните си възможности *[3][5]*.

Представянето на ИМКО-1 пред международна общественост се състои в Англия през 1981 г., на Международния симпозиум по роботика. Там е представена и система за управление на робот-ръка на основата на ИМКО-1, наречена *РОБКО 01*. По това време роботите в Япония и САЩ са управлявани от миникомпютри, а не микрокомпютри, така че българската разработка предизвиква истинско възхищение заради лекотата на работа със системата, както и заради цената на цялостното оборудване, което било десетки пъти по-евтино от западните си аналози *[3][4][5]*.

Робко 01 е близко копие на придобилия популярност *Armdroid 1 (фиг.1)*. Първият произведен модел на *Armdroid 01* е през 1980 г., от компания *Colne Robotics* (Twickenham, England) *[2]*. Мини манипулационните роботи от тази серия са представени като роботи за обучение и за кратко време стават изключително популярни в училищата и академичните среди, поради ниската цена и лесна експлоатация. Българският аналог на *Armdroid 01* е разработен в ИТКР, с цел обучение по роботика и кибернетика. Проектиран е за работа единствено с *ИМКО-1* и последващите го *ИМКО-2* и *Правец-8х*. *Робко 01* се произвежда в Завода за медицинска техника, София до 1989 г. Реализирани са около 4 000 броя.



Фиг.1. Учебен манипулатор Armdroid 01 (1980 г.).

Фиг.2. Учебен антропоморфен робот “РОБКО 01” (1981-1989 г.).

След 1989 г., с преустановяване на производството на 8-битови компютри от серията “Правец-8х” и ограниченията във възможностите за управление, роботите *Робко 01* стават неизползваеми. Тъй като по-голяма част от произведените модели са за българския пазар, след тридесет години тези манипулационни роботи са често срещан, бездействащ експонат сред училищната и университетската база. Тъй като не са ползвани, въпреки годините, голяма част от тях са напълно запазени и функциониращи. Не само заради предизвикателството от възможността да се задвижи отново подобен модел от миналото, но и от практична гледна точка, “съживяването” на учебния манипулационен робот успешно би могло да подпомогне обучението по роботика. Не бива да се забравя и фактът, че цената на предлаганите днес учебни,

манипулационни роботи с подобни на *Робко 01* характеристики, възлизат на няколко хиляди евро. Именно тези съображения мотивираха проучването на възможностите *Робко 01* да бъде управляван чрез съвременни хардуерни и софтуерни средства.

ТЕХНИЧЕСКИ ХАРАКТЕРИСТИКИ НА "РОБКО 01"

Робко 01 е антропоморфен робот с шест степени на свобода, реализирани от ротационни кинематични връзки. Задвижването се осъществява от шест четирифазни, стъпкови електродвигателя разположени в първото звено на робота. Движението към звената се предава посредством система от въжета и ролки. Конфигурацията на манипулационният робот осигурява работна зона с обем 350 mm^3. Захранването е 12 V, а консумацията достига до 2 A. Товароносимостта на робота е 1500 g.*[1][3]*

Въпреки ограничените ресурси на *"Правец 8х",* са осигурени богати възможности за програмно осигуряване. Компютърът формира управляващи сигнали, като комуникира с електронния модул на *Робко,* чрез специализиран интерфейс. Данните се предават паралелно по лентов кабел, който се подвързва към *ПОРТ В* на робота (единият от двата информационни порта, разположени в задната част на основата). Електронната платка е изцяло базирана на твърда, TTL логика. Всеки от двигателите се идентифицира със специфичен адрес, като изборът им става от адресен селектор. Управлението на селектора става с три бита (*A0, A1, A2*), което позволява координацията на осем звена. Селекторът допуска най-много един активен изход, което налага ограничението към определен момент да бъде възможно управлението само на една става. От друга страна, подобна организация позволява на електродвигателите да споделят обща, четири-битова информационна шина. При необходимост от едновременно задвижване на повече звена, управлението трябва да осигури последователно обхождане на адресите на съответните двигатели. Четири-битовите входни данни представляват логическа комбинация за управлението на четирите фази на стъпковите двигатели. Тази специфична информация формира последователност от стъпки до ориентиране на отделните стави в желана конфигурация.

Фиг.3. Робко 01 обслужващ въртяща маса и линеен конвейер.

Освен шестте управлявани степени на свобода, електронният модул на робота позволява подвързване на допълнителни устройства към външен входно/изходен порт – *ПОРТ А*. Входно/изходната шина условно е разделена на две тетради, като управлението на всяка от тях става от свободните два изходни адреса на селектора. Единият от входните битове се използва вътрешно, като индикатор за отворен/затворен хващач. По този начин се осигуряват общо 7 входа и 8 изхода за връзка с външно оборудване. Стандартно към *Робко* се предлагат *[3][4]:*

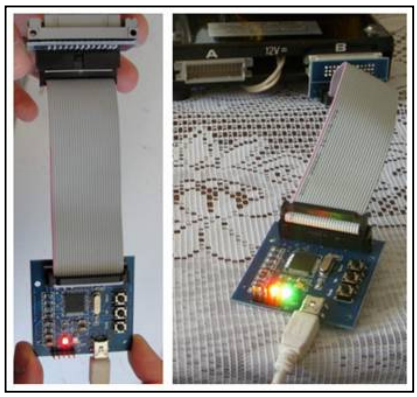
- Учебна въртяща маса, предназначена за кръгово преместване на детайли (*фиг.3. долу, в ляво*).
- Учебен конвейер за линейно преместване на детайли (*фиг.3. долу, в дясно*).
- Сензорен фотооптичен хващач. Реагира при наличието на обекти между пръстите на хващача.
- Магнитен хващач. Дава възможност за манипулиране на магнитно чувствителни материали с маса до 50 g.

За разрешаване работата с *Робко* е необходимо да се подсигури определена логическа комбинация на допълнителни пет адресни входа – *{A3=1, A4=1, A5=0, A6=0, A7=1}*. Освен това процесът на запис и четене в/от електронния модул се контролира с битове *Y/OW* и *Y/OR*, като активното ниво е *лог. 0*.

РОБКО 01 USB КОНТРОЛЕР

На база представените изходящи, технически характеристики бе реализиран *USB контролер* (*фиг. 4*), чрез който да бъдат формирани необходимите управляващи сигнали за робота. Контролерът представлява посредник между съвременен компютър и робота, като комуникацията се осъществява през популярния USB интерфейс.

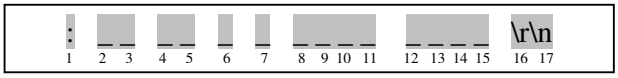
Връзката между контролера и електронния модул на *Робко* е чрез лентов кабел, като за улеснение на монтажа, оригиналния конектор е запоен на малка разширителна платка. Консумацията на контролера е под *20 mA*, което позволява захранването му директно от USB шината. Осигурени са два режима на работа – *ръчен* и *управление през компютър*. При първия вариант управлението на робота става директно от контролера, чрез натискане на бутоните разположени върху платката. При еднократно натискане на бутон *Mode* се избира определен двигател, който ще бъде обект на управление. По този начин последователно се обхождат всички стави. С другите два бутона *Up* и *Down* избраната става бива задвижвана съответно нагоре или надолу. При управлението на хващача е въведена опция за контрол над захващането. Когато хващачът е напълно затворен, се активира вграден от производителя индикаторен ключ. Контролерът следи състоянието на ключа и при превключването му се блокира по нататъшната възможност за затваряне. Ако тези мерки не се вземат, електродвигателят прави неуспешни опити да се завърти с прескачане на стъпки, поради повишения момент. В ръчния режим на работа преходът между стъпките е фиксиран на *10 ms*, като не се предоставя възможност за регулиране на скоростта на движение.



Фиг.4. РОБКО 01 USB контролер.

Както бе подчертано по-горе, за да бъде възможно управлението, адресни изходи *{A3÷A7}* трябва да осигурят определена логическа комбинация. Когато контролерът бива захранен, първоначално се пристъпва към инициализация на тези изходи. При свързване към робота, успешната комуникация се индицира от светещ зелен светодиод върху платката.

Вторият режим на работа осигурява по-богати и интуитивни възможности за управление на *Робко*. При първоначално свързване към компютъра, контролерът бива разпознат като виртуален сериен порт, благодарение на специализирания чип *FT232R,* интегриран в платката. Чиповете от тази фамилия представляват преобразуватели от USB към сериен интерфейс. По този начин, от гледна точка на управлението, потребителят работи с виртуален сериен (RS-232) порт. Управлението от компютър се осъществява от реализиран за конкретното приложение, стрингово базира протокол. Форматът на обменяните съобщения е следния:



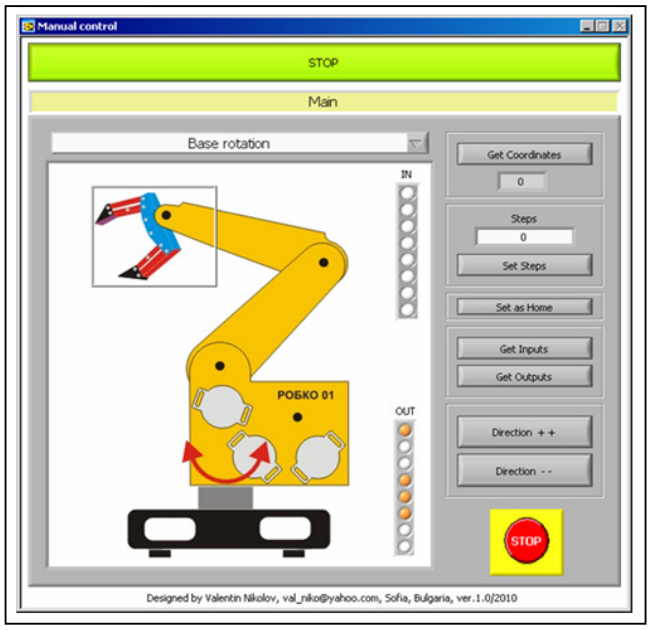
Дължината на съобщенията е фиксиран на седемнадесет байта. Всеки пакет започва със символ ':' оказващ началото на съобщението. Тъй като в бъдеще се предвижда кооперативно управление на два или повече манипулатора, вторият и третият байт задават адреса на робота, като се допускат стойности от 1÷99. Различните функции и начини на управление на контролера са обособени в отделни регистри. Така например, при оказан достъп до регистър *00*, контролерът обхожда последователно всички електродвигатели и нулира фазите. В *табл.1* са описани някои от най-често използваните регистри. С байт четири и пет се задава номера на регистъра (функцията), която ще се изпълнява. Байт шест и седем са служебни и обикновено задават знака (0 – минус; 1 – плюс) на последващите байтове. Данните от байт 8 до 15 са информационни. Обикновено съдържат определена информация, необходима за работата на оказания регистър. Съобщението завършва със служебни указатели за край на ред ("\r\n"). След упешно обработване на командата, контролерът връща "OK\r\n" или конкретен резултат в зависимост от използвания регистър.

Робко-01 адрес	*Регистър*	*Пример*	*Предназначение*
01-99	*00*	*: 01 00 0 0 0000 0000 \r\n*	*Обхожда и последователно нулира фазите на всички стъпкови двигатели*
01-99	*01*	*: 01 01 3 1 0001 0010 \r\n*	*Стартира се движението на става оказана с пореден номер (3), в посока нагоре (1) в режим на управление - пълна стъпка (0001). Пауза между стъпките 10 ms (0010).*
01-99	*02*	*: 01 02 0 0 0000 0000 \r\n*	*Прочита съдържанието на входовете на робота и връща резултата към компютъра.*
01-99	*03*	*: 01 03 0 0 0000 1199 \r\n*	*Променя активното състояние на адрес А5, разрешава/забранява нулирането на двигателите, променя адреса на Робко (01-99).*
01-99	*04*	*: 01 04 0 0 1010 1010 \r\n*	*Управлява изходите на ПОРТ А, като ги зарежда със стойност 10101010*
01-99	*05*	*: 01 05 0 0 0000 0000 \r\n*	*Прочита зададените стойности за изходите и връща резултата.*
01-99	*06*	*: 01 06 3 1 2222 2010 \r\n*	*Задава на двигател (3) да извърши (2222) стъпки в посока – нагоре (1), при скорост 10 ms (0010), в режим полу-стъпка (2)..*
01-99	*07*	*: 01 07 0 0 0000 0000 \r\n*	*Нулира координатите на всички става.*
01-99	*08*	*: 01 08 3 0 0000 0000 \r\n*	*Прочита координатите (изразени като брои стъпки) на става (3) и връща резултата.*
01-99	*09*	*: 01 09 0 0 2557 0100 \r\n*	*Ръчно multyjoins управление*
01-99	*10*	*: 01 10 0 0 0988 0000 \r\n*	*Задава координати за програмно multyjoins управление*
01-99	*11*	*: 01 11 0 0 0002 0100 \r\n*	*Изпълнява се програмно multyjoins управление, в режим полу-стъпка (002), със скорост 100 ms (0100).*

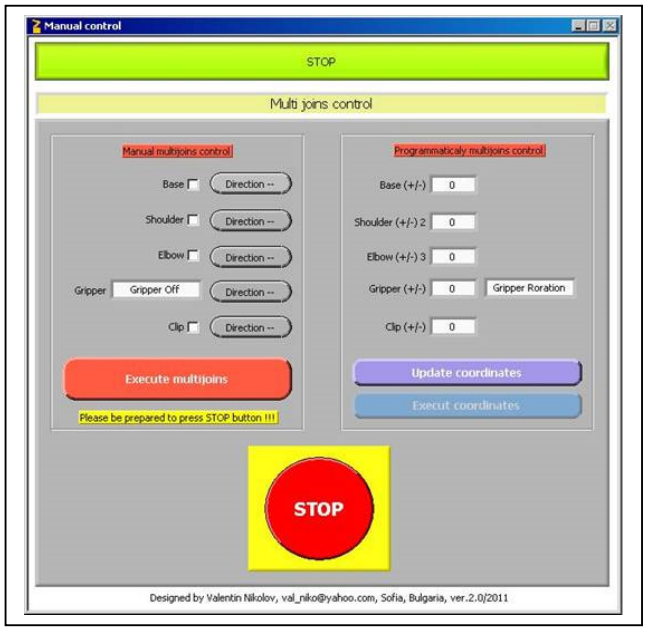
При така реализирания протокол, управлението на робота може да стане директно от произволна терминална програма (например *Hyper Terminal,* вграден в Windows). Тъй като ръчното въвеждане на съобщенията би затруднило нормалната работа, бе създаден интерактивен софтуер за управление на манипулационният робот.

ПРОГРАМНО ОСИГУРЯВАНЕ

За реализацията на софтуера бе използвана програмната среда на *Labview*. Програмата е разделена на три менюта – *Connection*, *Main* и *Settings*. Първото меню се зарежда автоматично, непосредствено след стартиране на програмата. В *Connection* се задава виртуалния сериен порт за връзка с *Робко*, адреса на робота, както и начина по който ще се управляват движенията. Възможностите са между мишка, мишка-клавиатура или мишка-джойстик, като съответна икона изобразява текущия избор на потребителя. От това меню се оказва и начинът, по който ще бъдат управлявани изходите. Когато към изходите на *ПОРТ А* са включени въртяща маса и/или конвейер за линейно премесване, тъй като задвижването им е със стъпкови двигатели, то управлението е аналогично на задвижването на робота. В този случай се изключва опцията *Output Manual Control* и когато бъдат избрани, изходите се управляват като превключват фазите на свързаните стъпкови двигатели. При активиране на *Output Manual Control* операторът задава ръчно логическото ниво на всеки от осемте изхода. По този начин могат да бъдат включвани/изключвани различни устройства, свързани към манипулационния робот. *Изходите на Робко са с отворен колектор*.



Фиг. 5. Главен панел на реализираната програма за управление на Робко 01.



Фиг. 6. Панел за настройка и управление на няколко (всички) стави едновременно.

Фаза 0	*Фаза 1*	*Фаза 2*	*Фаза 3*
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*

а)

Фаза 0	*Фаза 1*	*Фаза 2*	*Фаза 3*
1	*0*	*1*	*0*
0	*1*	*1*	*0*
0	*1*	*0*	*1*
1	*0*	*0*	*1*

б)

Табл.2. Управление на стъпковите двигатели в режими "пълна стъпка" – а) и "полустъпка" – б).

При натискане на бутон *START* се установява връзка с контролера, като индикатор за това е светване на червен светодиод върху платката. Ако комуникацията е успешна, се активират другите две менюта на програмата. *Main* е основно меню *(фиг.5),* в което става управлението на отделните звена на робота. За улеснение се приема, че е активиран режим на управление само с мишка. Управлението с клавиатура или джойстик е аналогично, като всяка от опциите може да бъде достъпна с натискане на конкретен бутон.

От падащо меню операторът избира ставата която ще бъде управлявана. Изборът се визуализира с подходящо изображение, като графично се посочва кой двигател е обект на задвижване. С бутони "Direction + +" и "Direction – – " ставата бива задвижвана в съответна посока. При управление на захващането, когато хващачът е максимално затворен, това състояние се визуализира с промяна в изображението. Върху изображението е разположен и индикатор "*IN*" за състоянието на входовете. При активиране на четенето, чрез "*Get Inputs*" всеки от осемте индикатора светва или загасва в зависимост от приложеното логическо ниво. В отделна колона са поставени осем контроли "*OUT*" за промяна логическите нива на изходите. При включване/изключване на отделните изходи, логическото ниво на всеки от тях може да се променя индивидуално.

При управлението на двигателите, контролерът автоматично записва всяка отработена стъпка, като това би могло да се използва при обучение и възпроизвеждане на предварително записани точки. Преди този процес роботът се поставя в желана изходна позиция – "*home*". С натискане на бутон "*Set as Home*" всички броячи се нулират. Броят на желаните стъпки които трябва да се отработят се задават в поле "*Steps*" и се изпращат към контролера с натискане на "*Set Steps*". Текущото състояние на всеки от броячите се визуализира в индикаторното поле на "*Get Coordinates*". *В процес на разработване е модул за автоматично и последователно възпроизвеждане на записани точки.*

Вторият панел (*фиг.6*) обслужва многоставно управление. До този момент бе коментирано управлението само на един двигател в конкретен момент от време. Контролерът подържа функция за едновременното управление на няколко (всички) електродвигатели. Тази функция е разделена на две подразделения. Едното е ръчно, многоставно управление. В този случай операторът избира кои двигатели ще са активни и съответно в какви посоки ще се движат. След изпълнението на командата определените двигатели се задвижват, като това продължава, или до натискане на STOP бутона или до изпращане на нова актуализираща команда. Втората под-функция е програмно многоставно управление. В този случай се задават стъпки, който двигателите трябва да отработят, както и посока на въртене. Ако определен двигател трябва да остане в покой (няма да бъде управляван) то тогава в полето стъпки се задава – 0. След тази първоначална дефиниция, параметрите се записват в паметта на контролера след натискане на бутон *Update coordinates*. При успешен запис, контролера връща потвърждение, а при неуспешен на екрана се визуализира съобщение за грешка. Контролерът е зареден с необходимите параметри, при което след натискане на бутон *Execute coordinates* се стартира изпълнението на програмата. Всеки от двигателите се движи до отработване на зададените стъпки.

Последното меню от програмата е служебно и в него се настройват някои от параметрите на управление. Тук се дефинира интервала при прехода между отделните стъпки, а оттам и скоростта на въртене на двигателите. Допустимите времена са в интервал *6÷100 ms*. Поради особеностите на стъпковите двигатели, по големите скорости водят до по-малък максимален момент, който може да се осигури. Това трябва да се има в предвид, особено при повдигане на обекти с по-голяма маса. В противен случай се наблюдава прескачане на стъпки или

невъзможност за завъртане. Друг начин по-който могат да се контролират скоростта и точността е чрез промяна на режимите на превключване между фазите. В случая е предвидено управление в режими "*пълна стъпка*" (*табл.2. а*) и "*полустъпка*" (*табл.2.б*). Когато е необходимо по-голяма точност (доколкото това се изисква изобщо) и момент, режимите "*пълна стъпка*" са за предпочитане. За по-непретенциозни приложния, режим "*полустъпка*" осигурява по-висока скорост на преместване. Други параметри, които са достъпни от това меню са промяната на адреса на манипулационния робот, активността на бит *A5* (*0 или 1*), както и възможността за автоматичното нулиране на фазите при управление на изходите (*ПОРТ А*) след всяка манипулация върху тях.

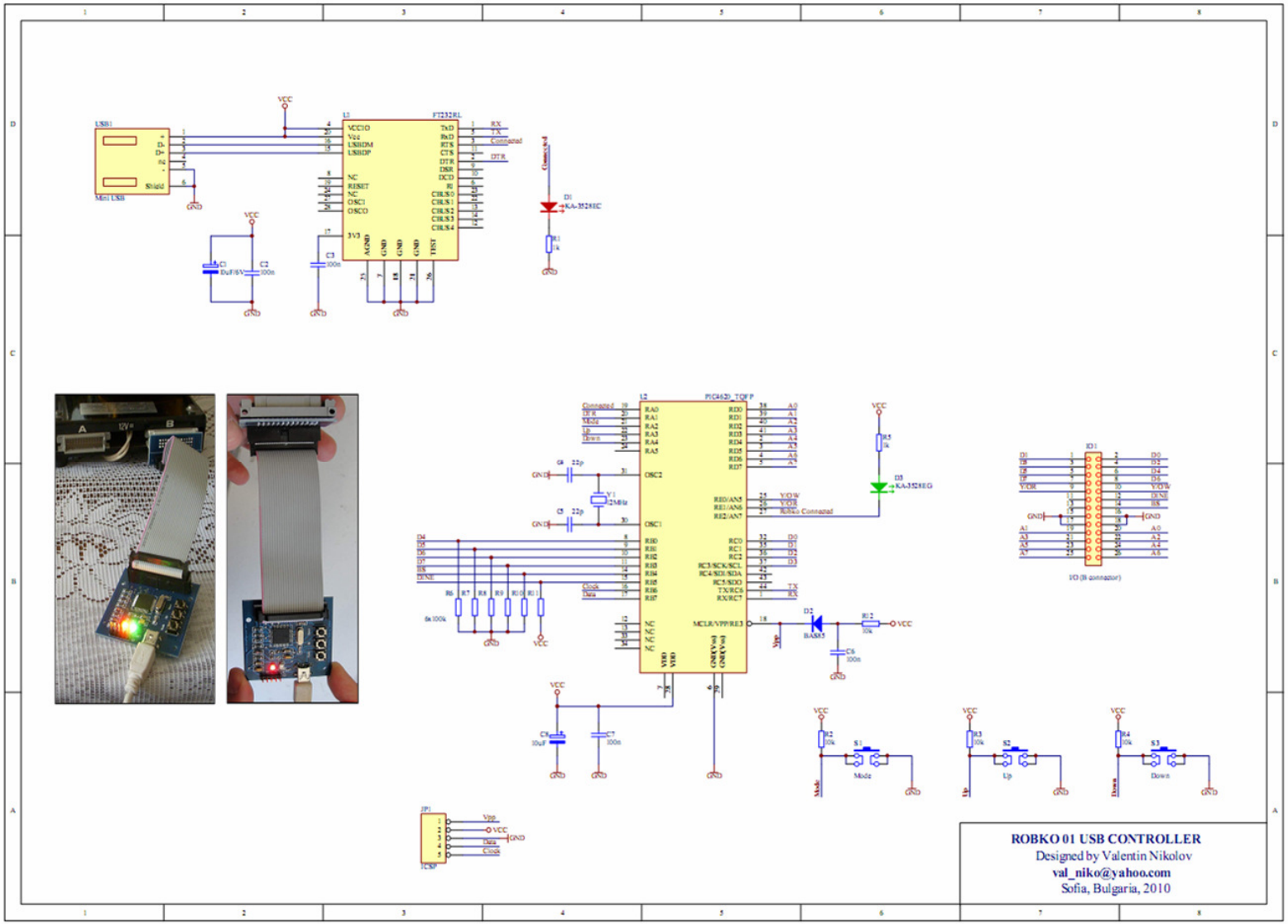
Начални експерименти около възможностите за управление на Робко 01 с USB контролерът са представени в кратък видео материал в [6] и [7].

ЛИТЕРАТУРА

1. Министерство на машиностроенето, ДСО "Приборостроене и автоматизация", Завод за медицинска техника, ИТКР – БАН, "*Мини робот РОБКО-01: Ръководство за експлоатация*", София, 1981.
2. http://www.anf.nildram.co.uk/beebcontrol/arms/index.html
3. http://bg.wikipedia.org/wiki/%D0%A0%D0%9E%D0%91%D0%9A%D0%9E-1
4. http://pc-history.hit.bg/index_files/Page518.htm
5. http://www.pravetz8.com/Robko_TNTM_Tech.html
6. **http://www.youtube.com/watch?v=0PiUocIpvKk**
7. **http://vbox7.com/play:2a0696a3c6**

НА СЛЕДВАЩИТЕ СТРАНИЦИ СА ПОМЕСТЕНИ ПРИЛОЖЕНИЯ 1 И 2 ПРЕДСТАВЯЩИ ХАРДУЕРА И ПРОТОКОЛА ЗА КОМУНИКАЦИЯ

Приложение 1: Принципна схема



Приложение 2: Описание на регистрите за управление

Описанието на регистрите е направено на база конкретни примерни стойности за всеки параметър:

1. Регистър 00: Обхожда и последователно нулира фазите на всички стъпкови двигатели, ако това е разрешено от регистър *03*, *Бит[13].*

Пример:

: 01 00 0 0 0000 0000 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '00' – задава номера на регистъра

Стойностите на останалите битове не са от значение, но се препоръчва да бъдат 0.
След приключване на командата, контролера връща ехо (същото съобщение което е получил).

2. Регистър 01: Управление на движението на конкретна става (двигател)

За всеки от двигателите има дефиниран пореден номер: 0 – за двигателя в основата ..... 5 – за единият вдигател от хващача. Номера 6 и 7 управляват *Aux0* и *Aux1* от изходния *ПОРТ А* (*за повече информация виж техническата документация на РОБКО 01 от литературните източници по-горе [1]*)

Пример:

: 01 01 3 1 0001 0010 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '01' – задава номера на регистъра
Бит [6] – '3' – заявка за управление на двигател с пореден номер '3'
Бит [7] – '1' – задава посока на въртене '1' – нагоре, '0' – надолу (посоките "горе" и "долу" са условни)
Бит [8-11] – '0001' – режим на управление на стъпковия двигател: '0001' – режим пълна стъпка, '0002' – режим полустъпка.
Бит [12-15] – '0010' – задава скорост на въртене (пауза между стъпките) в мили секунди. В случа 10 ms.

За спиране на движението на стартиран двигател се изпълнява регистър 00 (виж по-горе). След спиране на движението, както е оказано в рег.00, контролерът връща ехо.

При управление на двигател 2, поради зависимите движения, хващачът на Робко се отваря или затваря без операторът да е задал подобно действие. За да се компенсира това нежелано действие е добавена функция за автоматично компенириане движенията на хвачаща. Това е валидно за всички регистри по-долу който са свързани с управление на движенията.

3. Регистър 02: Прочита съдържанието на входовете на робота и връща резултата към компютъра

Пример:

: 01 02 0 0 0000 0000 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '02' – задава номера на регистъра

Стойностите на останалите битове не са от значение, но се препоръчва да бъдат 0.

Контролера връща информация за логическите нива на двата четири битови входа на ПОРТА. Форматът на отговора е:

Пример:

: 01 02 0 0 1010 1111 \r\n

Битове 1- 5 са аналогични с представените по-горе.

Бит [8-11] – '1010' – прочетени примерни логически нива на първи входен порт
Бит [12-15] – '1111' – прочетени примерни логически нива на втори входен порт

4. Регистър 03: Настройка на служебни параметри

Пример:

: 01 03 0 0 0000 1199 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '03' – задава номера на регистъра
Бит [6-11] – не са от значение. Препоръчва се да бъдат 0.

Бит [12] – променя активното състояние на адрес A5 (за повече информация виж [1]). Ако не са правени хардуерни промени по платката на РОБКО 01, промяната на този бит не се препоръчва, тъй като управлението ще бъде невъзможно. По подразбиране е '1'.

Бит [13] – контролира дали двигателите да остават под напрежение, след като са направили необходимите стъпки или се освобождават. Ако този бит е 0 двигателите остават под напрежение (цялата ръка остава твърда). При този режим консумацията е по-висока. Ако стойността на този бит е 1 се нулират само шестте двигателя на ръката. Ако стойността е 2 се нулират след управление и евентуално включените към порт А външни периферни устройства.

Бит [14,15] – променя адреса на контролера. По подразбиране адресът е 01, но в конкретния пример той е променен на 99. Когато адресът бъде променен всички съобщения трябва да бъдат изпращани с новя адрес. В противен случай командата няма да бъде изпълнена. Пример:

: 99 03 0 0 0000 1199 \r\n

След приключване на командата, контролера връща ехо.

5. Регистър 04: Управлява двата изходни, четири битови порта на ПОРТА.

Пример:

: 01 04 0 0 1010 1110 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '04' – задава номера на регистъра
Бит [6-7] – не са от значение. Препоръчва се да бъдат 0.

Бит [8-11] – '1010' – зарежда първи порт с оказаните логически нива
Бит [12-15] – '1110 – зарежда втори порт с оказаните логически нива

Важно е да се помни, че тези портове са достъпни за управление и от регистър 01. При рег. 01 управлението става на стъпки с оглед контрол на евентуално външно включени към този порт стъпкови двигатели. При регистър 04 управлението е побитово.

*След приключване на командата, контролера връща **ехо.***

6. Регистър 05: Прочита зададените стойности за изходите (посредством регистър 01 или 04) и връща резултата.

Пример:

: 01 05 0 0 0000 0000 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '05' – задава номера на регистъра

Стойностите на останалите битове не са от значение, но се препоръчва да бъдат 0.

Контролера връща информация за логическите нива на двата четири битови изхода. Ако се разгледа приемра от рег. 04 (*: 01 04 0 0 1010 1110 \r\n),* при който е зададена логическа комбинация 1010 1110, то отговора на контролера би трябвало да изглежда по следния начин:

: 01 05 0 0 1010 1110 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '05' – задава номера на регистъра
Бит [6-7] – не са от значение.

Бит [8-11] – '1010' – прочетени логически нива на първи порт
Бит [12-15] – '1110' – прочетени логически нива на втори порт

7. Регистър 06: Програмно управление.

Пример:

: 01 06 3 1 2234 1010 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '06' – задава номера на регистъра
Бит [6] – '3' – задава номер на двигател, който ще бъде управляван
Бит [7] – '1' – задава посока на движение на двигателя: '1' – нагоре, '0' – надолу.

Бит [8-11] – '2234' – задава брой стъпки който рябва да се отработят, в случая 2234 стъпки.
Бит [12] – '1' режим на управление на стъпковия двигател: '1' – режим пълна стъпка, '2' – режим полустъпка.
Бит [13-15] – '010' – задава скорост на движение (пауза между стъпките) в мили секунди, в случая 10 ms (Максималната пауза е 100ms).

*След приключване на командат и зададеното движение, автоматично се стартира регистър 00, след което контролера връща **“OK\r\n”.***

8. Регистър 07: *“HOME” позиция - Инициализира координатите на всички става.*

Независимо от начина на управление на всеки от двигателите, изпълнение на определена стъпка се записва в индивидуален брояч. В зависимост от посоката на въртене съдържанието на брояча се увеличва или намалява адекватно на отработените стъпки. Стойноста за брой стъпки е положително или отрицателно число в зависимостр от посоката на въртене. Регистър 07 нулира всички броячи.

*След приключване на командата, контролера връща ехо**.***

Пример:

: 01 07 0 0 0000 0001 \r\n

Бит [1] – ‘:’ – старт на съобщението
Бит [2,3] – ‘01’ – задава адреса на контролера
Бит [4,5] – ‘07’ – задава номера на регистъра
Бит [12-15] – 0001 – задава конфигурацията на HOME позицията. При стойност 0000 – координатите на всички стави се нулират, т.е. зарежда се стойност 0. За да бъде корекно управлението при решаването на правите и обратни задачи, тази конфигурация на HOME трябва да се използва, когато ръката е максимално, хоризонтално изпъната напред (Q1=Q2=Q3=Q4=Q5=P=R=0 deg. виж документацията ROBKO0q [1]).
При стойности 0001 (или различна от 0000) – ръката застава под формата на буквата Г. В тази конфигурация единствено Q2=90 deg., което съответства приблизително на 1824 стъпки. Това число се зарежда като координата на тази става. Всички останали координата се нулират.

9. Регистър 08: Прочита координатите (изразени като брои стъпки) на определен двигател и връща резултата.

Пример:

: 01 08 3 0 0000 0000 \r\n

Бит [1] – ‘:’ – старт на съобщението
Бит [2,3] – ‘01’ – задава адреса на контролера
Бит [4,5] – ‘08’ – задава номера на регистъра

Стойностите на останалите битове не са от значение, но се препоръчва да бъдат 0.

Контролерът връща примерен резултат:

: 01 08 3 1 0005 2340 \r\n

Бит [1] – ‘:’ – старт на съобщението
Бит [2,3] – ‘01’ – задава адреса на контролера
Бит [4,5] – ‘08’ – задава номера на регистъра
Бит [6] – ‘3’ – пореден номер на двигателя
Бит [7] – '1' – задава знака на координатите. При стойност '1' знакът е отрицателен.
Бит [8-11] – '0005’ – стойноста се умножава по 10000, в случая резултатът ще бъде 50000;
Бит [12-15] – '2340' – тази стойност се сумира с горното число, като стойноста на отработените от двигател 3 стъпки в конкретния пример е - 52 340 стъпки.

10. Регистър 09: Задава ръчно многоставно (multyjoins) управление.

До този момент бе разгледано управление става по става, като в даден момент се задвижваше само по един електродвигател. Контролера подържа функция за едновременното управление на няколко (всички) двигатели.

Пример:

: 01 09 1 0 2557 0100 \r\n

Бит [1] – ‘:’ – старт на съобщението
Бит [2,3] – ‘01’ – задава адреса на контролера
Бит [4,5] – ‘09’ – задава номера на регистъра
Бит [6] – ‘1’ – режим на управление на стъпковия двигател: '1' – режим пълна стъпка, '2' – режим полустъпка.
Бит [7] – не е от значение.
Бит [8-11] – '2557’ – Първите шест бита от това число съответстват на шестте задвижвани стави на ръката. Ако го разгледаме в двоичен вид, числото изглежда по следния начин: 2557 – 0b0000100111111101. Първите шест бита (в червено) показват кой двигател ще бъде активен, в случая само двигател номер 2 няма да се движи защото е в лог. 0, а останалите са в лог. 1. Вторите шест бита (в жълто) показват посоките на съответните двигатели. Лог. 1 е посока “нагоре”, 0 е противоположната, като посоките “горе-долу” са условни. Последните четири бита от това число не се използват, но се препоръчва да бъдат 0.

Бит [12-15] – '0100' – Последните четири бита задават скоростта за движение в случая 0100 - 100 ms.

*След приемане на командата, контролера връща ехо**.***

При задаване на определени координата, отделните двигатели ще се движат докато не бъде изпратена нова команда оказваща промяна. При необходимост от спиране се стартира регистър 00.

11. Регистър 10: Задаване на координати за програмно multyjoins управление.

С този регистър първоначално се задават брой стъпки които ще се отработват от двигателите, след което със следващия регистър (11) тези координати се изпълняват.

Пример:

: 01 10 2 0 0988 0000 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '10' – задава номера на регистъра
Бит [6] – '2' – задава адреса на двигателя, в случая двигател 2
Бит [7] – '0' – задава посока на движение: '1' – нагоре, '0' – надолу.
Бит [8-11] – '0988' – задава броят стъпки който трябва да направи оказания двигател.
Бит [12-15] – не са от значение.

От представено се вижда, че тази команда трябва да се изпълни ОСЕМ пъти, последователно от става 0 до става 5, за всеки от двигателите на РОБКО 01 + двата допълнителни изхода, които се потдържат за свързване с периферия. Дори дадена става/изход да не бъде използвана е задължително да бъде изпратена команда, като за брой на стъпки се задава 0000. След всяка успешно възприета команда контролера връща отговор ехо.

12. Регистър 11: Изпълнява се програмно multyjoins управление по предварително зададени координати от ***регистър 10***.
Пример:

: 01 11 0 0 0001 0000 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '11' – задава номера на регистъра
Бит [8-11] – '0001' – режим на управление на стъпковия двигател: '0001' – режим пълна стъпка, '0002' – режим полустъпка.
Бит [12-15] – '0100' – задават скоростта за движение за всички двигатели, в случая 0100 - 100 ms.

След приемане на командата, контролера връща ехо.
След изпълнението на командата, записаните с регистър 10 координати се изпълняват едновременно.
След приключване на командата, контролера автоматично изпълнява рег. 00 и съответно ехо на този регистър.

13. Регистър 12: Задаване на координати за програмно multyjoins управление.

Разликата между този и регистър 10 е, че в случая се задават координата (градуси изразени в брой стъпки) до които роботът трябва да достигне, а не просто конкретен брой който трябва да се отработят. Така на примет ако двигателя в основата (Q1) в момента се намира в точка с координати 100 стъпки и с регистър 12 зададем желана дестинация 300 стъпки, контролерът ще преизчисли, че за достигането на тази координата е необходими отработването на 200 стъпки.
С този регистър първоначално се задават координатите, след което със следващия регистър (11) тези координати се изпълняват.

Пример:

: 01 10 2 0 0988 0000 \r\n

Бит [1] – ':' – старт на съобщението
Бит [2,3] – '01' – задава адреса на контролера
Бит [4,5] – '10' – задава номера на регистъра
Бит [6] – '2' – задава адреса на двигателя, в случая двигател 2
Бит [7] – '0' – задава посока на движение: '1' – нагоре, '0' – надолу.
Бит [8-11] – '0988' – задава координатите до които трябва да достигне съответната става (двигател).
Бит [12-15] – не са от значение.

От представено се вижда, че тази команда трябва да се изпълни ОСЕМ пъти, последователно от става 0 до става 5, за всеки от двигателите на РОБКО 01 + двата допълнителни изхода, които се потдържат за свързване с периферия. Ако дадена става/изход не трябва да променя координатите си, то задължително трябва да се изпратят текущите координати (изразени в брой стъпки).
След всяка успешно възприета команда контролера връща отговор ехо.