Elektrotehniški vestnik 75(3): 136-142, 2008
Electrotechnical Review: Ljubljana, Slovenija

Implementacija in uporaba prosto dostopnega operacijskega sistema realnega časa na vgrajenem robotskem krmilniku

Darko Hercog, Aljaž Kapun, Karel Jezernik

Univerza v Mariboru, Fakulteta za elektrotehniko računalništvo in informatiko,Smetanova 17, 2000 Maribor, Slovenija
E-pošta: darko.hercog@uni-mb.si, aljaz.kapun@uni-mb.si, karel.jezernik@uni-mb.si

Povzetek. V članku sta predstavljeni implementacija in uporaba prosto dostopnega operacijskega sistema realnega časa na robotskem krmilniku DSP-2. Vgrajeni robotski krmilnik, ki je bil razvit na Fakulteti za elektrotehniko računalništvo in informatiko Univerze v Mariboru, se uporablja pretežno v pedagoškem procesu za vodenje večosnih mehanizmov. Dodatno je bila k sami implementaciji operacijskega sistema realnega časa FreeRTOS realizirana podpora za programski jezik MATLAB/Simulink. Ta programska podpora omogoča preprosto kreiranje več opravilnih algoritmov v blokovno orientiranem programu Simulink ter hitro implementacijo in verifikacijo zasnovanih algoritmov vodenja na vgrajenem robotskem krmilniku. Članek je podkrepljen z opisom primera uporabe implementiranega operacijskega sistema realnega časa pri vodenju dvoosnega mehanizma.

Ključne besede: operacijski sistemi realnega časa, vgrajeni sistemi, robotski krmilniki, hitra implementacija algoritmov vodenja, SCARA mehanizmi, izobraževanje

Implementation and usage of a freely available real-time operating system on an embedded robot controller

Extended abstract: The paper presents implementation and usage of a freely available real-time operating system (RTOS) on the DSP-2 Robotic Controller (Figure 1). This is in-house developed embedded controller based on the Texas Instruments (TI) TMS320C32 digital signal processor (DSP) and Xilinx FPGA. It is mainly used in the educational process for control of up to four-axes mechatronic devices. By using the Simulink built-in blocks and blocks from the DSP-2 add-on blockset (Figure 2), programming of this controller can be easily achieved using the MATLAB/Simulink block-oriented development environment.

Robotic control algorithms are very complex and usually composed of multiple logical tasks. Execution of such multitasking applications requires a real-time operating system on the control hardware. TI has never developed RTOS for TI 'C3x family of DSP's, however, few companies can be found on the market that offer RTOS'es for this type of DSP's. Commercially available solutions are very powerful but are also relatively expensive and they often require royalty payments. Therefore, a decision has been made to implement freely available RTOS on the DSP-2 Robotic Controller. From a set of freely available RTOS'es, a FreeRTOS operating system was selected because it fulfilled all the given requirements. In addition to the FreeRTOS port for the DSP-2 Robotic Controller, an additional Simulink RTOS blockset was developed. This blockset (Figure 3) includes blocks for tasks, message queues and semaphores creation, blocks for sending and receiving messages using message queues, blocks for managing binary semaphores, etc. Using these blocks (Figure 3), Simulink built-in blocks and blocks from DSP-2 blockset (Figure 2), multitasking algorithms can now be easily developed under the MATLAB/Simulink frame and, after automatic code generation and deploying process, verified on the DSP-2 Robotic Controller (Figure 4). The paper includes a description of a simple application of the implemented operating system on the DSP-2 Robotic Controller that drives a two-axis mechatronic device (Figure 4).

Keywords: real-time operating systems, embedded systems, robotics controllers, rapid control prototyping, SCARA mechanisms, education

Prejet 31. maj, 2007
Odobren 5. februar, 2008

1 Uvod

Eksperimentalno delo igra in bo zagotovo tudi v prihodnje igralo pomembno vlogo pri poučevanju študentov tehničnih strok [1]. Pri poučevanju sistemov avtomatskega vodenja so v preteklosti vaje temeljile predvsem na snovanju koncepta algoritma vodenja, izvedbi simulacij in verifikacij rezultatov na podlagi simulacijskih podatkov, medtem ko je bilo eksperimentalnega dela in validacij zasnovanih algoritmov na podlagi eksperimentalnih rezultatov malo. S prihodom novih orodij in metod, ki omogočajo hitrejši prehod iz simulacij na delo v realnem času na realni strojni opremi, so se odprle nove možnosti za povečanje eksperimentalnega dela v pedagoškem procesu in za izboljšanje kakovosti poučevanja. Tako imenovani sistemi za hitro načrtovanje in izvedbo algoritmov vodenja (rapid control prototyping – RCP) snovalcem sistemov omogočajo, da se posvetijo reševanju problemov t. j. načrtovanju, implementaciji in testiranju algoritmov vodenja in ne kodiranju, kot je bilo do zdaj v stalni praksi. Orodja, kot so MATLAB/Simulink, MATRIXx in VisSIM, omogočajo blokovno snovanje sistemov vodenja in izvedbo simulacij. Z dodatnimi orodji (Real-Time Workshop, AutoCode in C-Code) je omogočena avtomatska pretvorba blokovnega diagrama v C kodo in ob uporabi

ustreznega C prevajalnika v binarno kodo. Za izvajanje le-te se uporabljajo različne platforme, kot so osebni računalniki z vgrajenimi vhodno/izhodnimi karticami brez ali z lastnim procesorjem, samostojni vgrajeni sistemi na osnovi digitalnih signalnih procesorjev (DSP) in mikrokrmilnikov. Komercialno dobavljivi RCP sistemi, kot so npr. produkti podjetja dSPACE, so zmogljivi in dovršeni, a so relativno dragi in posledično primernejši za tista področja, kjer ima uporabnost prednost pred ceno (npr. raziskave in razvoj). V pedagoškem procesu so zaželeni manj sofisticirani, cenejši vendar relativno zmogljivi, intuitivni in odprti RCP sistemi.

V literaturi je mogoče zaslediti komercialno dobavljiva pedagoška učila (npr. [2], [3], [4]), prilagojena za vodenje enosmernih in izmeničnih motorjev, pri čemer je omogočeno blokovno programiranje. Algoritmi vodenja se v takšnih učilih večinoma izvajajo na osebnih računalnikih z vgrajenimi vhodno/izhodnimi karticami, zato so ti sistemi premalo zmogljivi in tako niso primerni za vodenje hitrih dinamičnih sistemov. Da bi se izognili tem slabostim, so bili na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, izdelani RCP-sistemi vodenja (DSP-2 kartica, DSP-2 pedagoško učilo, DSP-2 robotski krmilnik), ki temeljijo na digitalnem signalnem procesorju (DSP). DSP-2 pedagoško učilo (DSP-2 LM) se uporablja predvsem za vodenje enosmernih in izmeničnih motorjev, medtem ko je robotski krmilnik DSP-2 (DSP-2 RC) namenjen vodenju večosnih mehatronskih sistemov. Za te sisteme vodenja je bila izdelana programska podpora, ki omogoča hitro in preprosto programiranje z uporabo uveljavljenega blokovno orientiranega programskega paketa MATLAB/Simulink. Sistemi vodenja DSP-2 temeljijo na signalnem procesorju Texas Instruments (TI) C32. Čeprav nastanek družine signalnih procesorjev C3x sega v leto 1990, TI še razvija in izdeluje posamezne procesorje omenjene družine, vendar pa proizvajalec ni nikoli razvil operacijskega sistem za to družino. V članku je predstavljena implementacija prosto dostopnega operacijskega sistema (OS) realnega časa na robotskem krmilniku DSP-2. Dodatno k realizaciji »porta« je bila realizirana tudi podpora za MATLAB/Simulink. Podpora omogoča preprosto kreiranje opravil v programu Simulink in hitro implementacijo večopravilnih algoritmov na robotskem krmilniku DSP-2.

Članek je sestavljen takole: v poglavju 2 je na kratko predstavljena uporabljena strojna in programska oprema. Poglavje 3 se navezuje na operacijske sisteme realnega časa. V poglavju so podana merila za izbiro operacijskega sistema realnega časa, opisan je izbran operacijski sistem in primerjava le-tega z RT-Linux OS realnega časa. Poglavje 4 opisuje primer uporabe implementiranega operacijskega sistema realnega časa pri vodenju dvoosnega mehatronskega sistema.

2 Robotski krmilnik DSP-2

2.1 Strojna oprema

Robotski krmilnik DSP-2 (slika 1) vsebuje periferijo, ki je potrebna za upravljanje štiriosnih sistemov. DSP-2 RC vsebuje naslednje pomembnejše komponente:

- stikalni napajalnik z vhodno napetostjo 12 V (9 do 20 V).
- 4 x RS-422 vhod za inkrementalni dajalnik (a, b in referenca)
- 4 x 12-bitni analogni vhod (± 10 V)
- 4 x 12-bitni analogni izhod (± 10 V)
- 8 galvansko ločenih logičnih izhodov (12 V)
- 16 galvansko ločenih logičnih vhodov (12 V)
- CAN vmesnik
- USB vmesnik
- Ethernet vmesnik

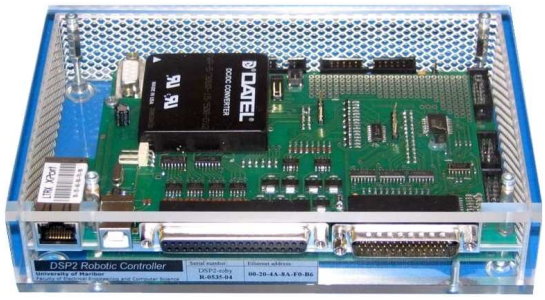
2.2 Programska oprema

Za DSP-2 sisteme vodenja je bila razvita MATLAB/Simulink podpora [5]. Ta vsebuje nabor Simulink blokov, ki so specifični za DSP-2 sisteme in ob uporabi Real-Time Workshopa omogočajo preprosto programiranje DSP-2 sistemov s programom MATLAB/Simulink. Primer Simulink blokov za robotski krmilnik DSP-2 je prikazan na sliki 2. Več o blokovnem programiranju in uporabi DSP-2 sistemov v pedagoškem procesu je v [6] in [7].

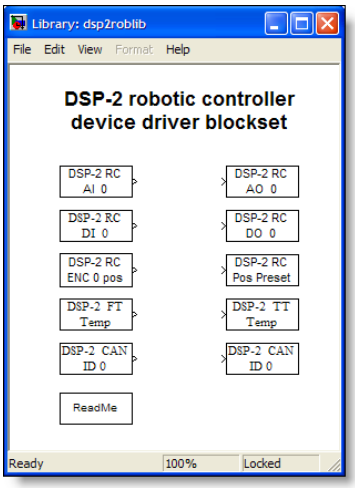
3 OS realnega časa

Z razvojem računalništva sta se splošno gledano razvila dva tipa operacijskih sistemov: (1) splošnonamenski operacijski sistemi (*General-Purpose Operating Systems - GPOS*), kot sta na primer UNIX in Windows, in (2) manjši, kompaktnejši operacijski sistemi realnega časa (*Real-Time Operating Systems - RTOS*) kot je npr. VxWorks. RTOS so operacijski sistemi, ki se od GPOS razlikujejo predvsem po manjši porabi pomnilnika, po hitrejšem in zanesljivejšem izvajanju, ter po veliki stopnji prilagodljivosti. Večina OS realnega časa omogoča vključitev ali izključitev določenih komponent operacijskega sistema, s čimer je mogoče doseči dobro prilagoditev RTOS glede na uporabljeno strojno opremo in samo aplikacijo [8].

Podjetje Texas Instruments je razvilo operacijski sistem realnega časa DSP/BIOS za DSP platforme TMS320C6000, TMS320C5000 in TMS320C28x. Družina signalnih procesorjev C3x ni podprta s tem operacijskim sistemom, je pa na trgu nekaj podjetij, ki ponujajo OS realnega časa za to družino signalnih procesorjev, kot so Precise Software Technologies (MQX™ RTOS), Eonic Systems (Virtuoso), Spectron Microsystems (SPOX), CMX Systems (CMX-RTX RTOS). Žal so takšni sistemi precej dragi, saj stanejo 10.000 $ in več.



Slika 1: Robotski krmilnik DSP-2
Figure 1: DSP-2 robotic controller



Slika 2: Nabor Simulink blokov za DSP-2 robotski krmilnik
Figure 2: Simulink blockset for the DSP-2 Robotic Controller

3.1 Izbira operacijskega sistema

Pri izbiri ustreznega operacijskega sistema za robotski krmilnik DSP-2 so bila pomembna predvsem sledeča merila:

- izvorna koda OS mora biti prosto dostopna;
- koda OS mora biti napisana v programskem jeziku C;
- OS mora biti dovolj kompakten (majhna zahteva po pomnilniku), da ga je mogoče implementirati na vgrajenem krmilniku DSP-2;
- OS mora vsebovati vsaj osnovne komponente operacijskega sistema (razvrščevalnik, sporočilne vrste in semaforje);
- če so za delovanje in analizo OS potrebna dodatna orodja, naj bodo le-ta podprta v okolju Windows - zaradi lažje integracije OS z obstoječim sistemom DSP-2, saj razvojno orodje za DSP-2 temelji na okolju Windows;

Izmed prosto dostopnih sistemov je bil izbran FreeRTOS OS realnega časa, saj je ustrezal vsem podanim zahtevam.

3.2 FreeRTOS OS realnega časa

FreeRTOS [9] je preprost, prenosljiv in zgoščen operacijski sistem in je prilagojen za vgrajene sisteme z malo pomnilnika. Izvorna koda operacijskega sistema FreeRTOS je zaščitena z licenco z izjemo GNU GPL (General Public License). Večina kode operacijskega sistema je napisana v programskem jeziku C. Takšna koda sicer ni tako optimalna kot zbirna koda, vendar je koda lažje berljiva in preprosto prenosljiva na različne platforme. Zbirna koda je tako uporabljena le tedaj, ko je uporaba le-te neizogibna (npr. pri menjavi kontekstne vsebine).

3.3 Primerjava operacijskih sistemov FreeRTOS in RT-Linux

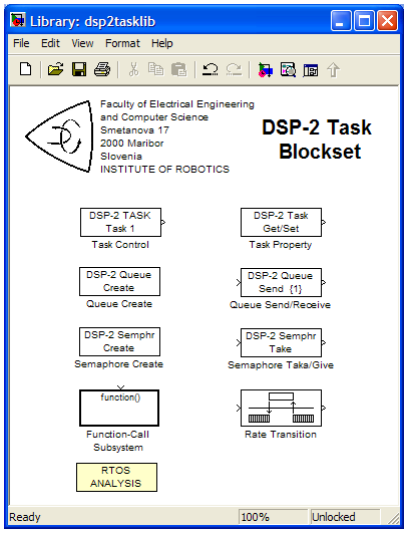
V zadnjem času je v vgrajenih aplikacijah mogoče zaslediti pogosto uporabo operacijskega sistema realnega časa RT-Linux. V nadaljevanju je na kratko opisana primerjava med operacijskima sistemoma RT-Linux in FreeRTOS [10]:

- Velikost: FreeRTOS je prilagodljiv OS realnega časa in je bil namensko pisan za vgrajene sisteme, zato zavzame zelo malo pomnilnika. Osnovna aplikacija, ki se izvaja na procesorju AVR, zavzame npr. le 4.4 kB. RT-Linux je prav tako zelo prilagodljiv RTOS, vendar se velikost potrebnega pomnilnika v večini aplikacij meri v MB.
- Platforme: FreeRTOS je bil do zdaj uspešno implementiran na različnih vgrajenih sistemih kot so mikrokrmilniki in DSP, medtem ko je RT-Linux mogoče zaslediti na platformah x86 in računalnikih PowerPC.
- Komponente OS: FreeRTOS vsebuje le osnovne komponente operacijskega sistema (razvrščevalnik, sporočilne vrste in semaforje), medtem ko RT-Linux ponuja poleg osnovnih komponent še številne dodatne komponente in servise: mrežna podpora, grafično okolje (X11), spletni strežnik, itd. (te komponente se izvajajo v okolju Linux in tako niso del RT-Linuxa).
- Razvrščevalnik: FreeRTOS uporablja razvrščanje po prioriteti s prekinjanjem, pri čemer si opravila z isto prioriteto enakomerno delijo procesorski čas. RT-Linux ponuja več različnih razvrščevalnikov. Ko se izvaja opravilo v ozadju (idle task), se izvaja jedro Linux.

RT-Linux ni bil izbran predvsem zato, ker za svoje delovanje potrebuje več pomnilnika, kot ga premore robotski krmilnik DSP-2.

3.4 FreeRTOS na DSP-2 robotskem krmilniku

Dodatno k implementaciji FreeRTOS OS realnega časa na krmilniku DSP-2 je bila realizirana tudi gradnja Simulink blokov, ki omogočajo preprosto in hitro



Slika 3: Nabor Simulink blokov »DSP-2 Task Blockset«
Figure 3: Simulink DSP-2 Task Blockset

izvedbo večopravilnih algoritmov vodenja na omenjenem ciljnem sistemu z uporabo programa MATLAB/Simulink. Ta nabor Simulink blokov (»*DSP-2 Task Blockset*« - slika 3) vključuje bloke za kreiranje opravil, kreiranje in uporabo semaforjev, kreiranje in uporabo sporočilnih vrst itd. Primer uporabe je prikazan na sliki 4.

4 Primer uporabe na vodenju dvoosnega mehanizma

4.1 Eksperimentalni sistem

Na sliki 4 je prikazan eksperimentalni sistem, ki je sestavljen iz DSP-2 RC, dvoosnega mehanizma in dveh analognih tokovnih regulatorjev.

Dvoosni mehanizem sestavljata dve rotacijski osi, ki sta povezani v ravninsko robotsko roko [11]. Prvo os poganja servomotor prek zobniškega prenosa. V nasprotju z navadno izvedbo, kjer je pogonski motor drugega člena nameščen v njegovem sklepu, se pri opisanem mehanizmu le-ta nahaja na mirujočem delu mehanizma. Pogon drugega člena tako tvori servomotor v kombinaciji z zobniškim in jermenskim prenosom. Prednost takšne konstrukcije je bistveno zmanjšanje mase gibajočih se delov in posledično izboljšanje dinamičnih lastni mehanizma. Kot servomotorja sta uporabljena enosmerna motorja s trajnimi magneti in rotorjem brez železnega jedra. Meritev zasuka posamezne osi omogočata motorjema prigrajena inkrementalna dajalnika.

Za pogon servomotorjev je uporabljena analogna tokovna regulacija, ki zagotavlja generiranje gladkega navora. Pri analogni izvedbi namreč tok motorja ne vsebuje valovitosti, tipične za tokovne regulacije, realizirane s stikalnimi pretvorniki. Tokovno regulacijo sestavljajo vhodni diferenčni ojačevalnik, odštevalnik referenčne in dejanske vrednosti toka, PI regulator, močnostni ojačevalnik ter merilnik toka. Na tiskanini tokovnih regulatorjev se nahajata tudi pretvorniški vezji za pretvorbo izhodnih sinusnih signalov inkrementalnega dajalnika v pravokotne pulze.

4.2 Izvedba poskusa

Cilj zadane naloge je vodenje vrha dvoosnega ravninskega mehanizma po točkah na navidezni elipsi v zunanjem koordinatnem sistemu. Uporabljen je gib točka-točka, pri čemer se osi mehanizma gibljeta po $\sin^2$ pospeškovnem profilu. Kot regulacijski algoritem je uporabljena metoda izračunanega navora.

Operacijski sistem omogoča delo z večopravilnimi sistemi, pri čemer so atributi *n*-tega opravila maksimalen čas izvajanja opravila C_n, perioda opravila T_n in prioriteta opravila P_n. Maksimalen čas izvajanja opravila določimo s časovno analizo algoritma opravila. Periodičnost opravila ne pomeni periodičnosti opravila v klasičnem pomenu, temveč da se opravilo znotraj časovnega intervala $\left[(m-1)T_n, mT_n\right]$ $m \in \mathbb{N}$ izvede natanko enkrat. Na DSP-2 RC so opravila razvrščena po fiksni prioriteti s prekinjanjem. Razvrščanje s prekinjanjem temelji na periodičnem urinem taktu T_{tick}. Urin takt sproži razvrščevalnik, ki določi opravilo, ki se bo izvajalo v naslednji periodi urinega takta. Večopravilni sistem deluje pravilno, če so vsa opravila izvedljiva in če so podatki med opravili ustrezno sinhronizirani. Izvedljivost opravil zagotavlja pravilno izbrana perioda urinega takta glede na atribute opravil.

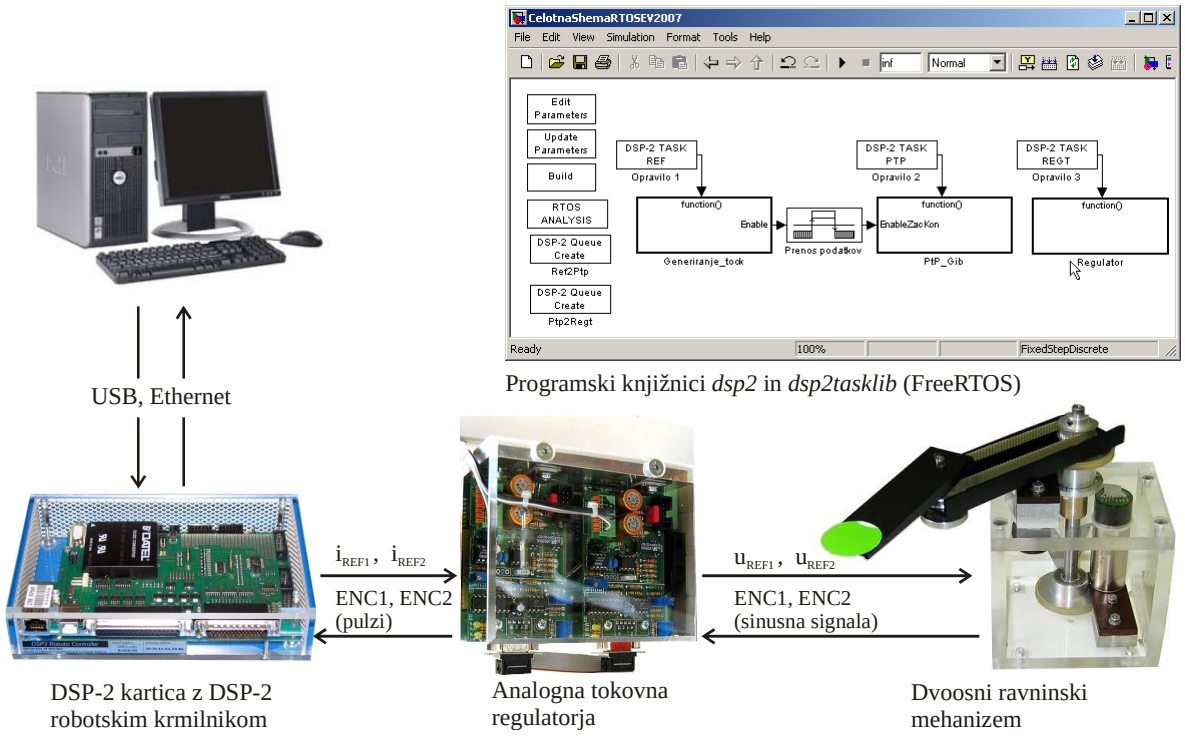
V smislu večopravilnega sistema lahko nalogo vodenja razdelimo na tri opravila, in sicer:

Opravilo 1: določitev točk na elipsi,
Opravilo 2: računanje referenc,
Opravilo 3: računanje regulacijskega algoritma.

Točke na elipsi določimo z uporabo števca *xD*, ki periodično šteje od vrednosti *0* do vrednosti *N*, pri čemer je *N* število točk na elipsi. Pri vsakem povečanju števca določimo koordinati točke po (1), kjer sta *a* in *b* polosi elipse. Algoritem preverja dimenziji s strani uporabnika vnesenih polosi glede na delovno območje mehanizma in ju po potrebi nastavi na prevzeti vrednosti.

$$x = a \cdot \cos\left(xD\frac{2\pi}{N}\right) \quad y = b \cdot \sin\left(xD\frac{2\pi}{N}\right) \qquad (1)$$

Reference določajo pospešek, hitrost in položaj za obe osi glede na izbrani pospeškovni profil (trapezni profil, profil $\sin^2$). Uporaba pospeškovnega profila $\sin^2$ je priporočljiva zaradi zveznosti tako prvega kot drugega odvoda položaja, kar zagotavlja popolnoma kontrolirano gibanje mehanizma brez kakršnih koli sunkov. Reference se nanašajo na gibanje posameznih osi oz. na notranje koordinate, medtem ko so tehnološke



Slika 4: Eksperimentalni sistem
Figure 4: Experimental system

koordinate ponavadi podane glede na zunanji koordinatni sistem. Uporaba različnih koordinatnih sistemov zahteva pretvarjanje zunanjih koordinat v notranje koordinate, kar zahteva uporabo računsko zahtevnih trigonometričnih funkcij.

V opravilu regulacije mehanizma se izvajajo meritve položajev ter hitrosti za posamezno os, računanje regulacijskega algoritma ter izdajanje krmilnih veličin prek D/A pretvornikov na analogna tokovna regulatorja.

Kot regulacijski algoritem je uporabljena metoda izračunanega navora, ki temelji na dinamičnem modelu pasivnega mehanizma, določenem po Langrangeovem postopku modeliranja. Zaradi ravninske izvedbe mehanizma ni vpliva delovanja gravitacije, zato ima dinamični model mehanizma obliko:

$$\vec{M} = J\left(\vec{q}\right)\ddot{\vec{q}} + \vec{C}\left(\vec{q},\dot{\vec{q}}\right), \qquad (2)$$

kjer je:

$\vec{M}$ - vektor navorov,

J - matrika vztrajnostnih momentov,

$\vec{C}$ - vektor navorov zaradi delovanja Coriolisovih sil.

Bistvo metode izračunanega navora je eksterna linearizacija dinamike mehanizma z inverznim dinamičnim modelom mehanizma, kateri je dodan PD regulator, kot je prikazano na sliki 6. Pogrešek linearizacije zaradi netočnosti parametrov dinamičnega modela in nemodelirane dinamike pomeni motnjo, katere vpliv lahko v omejenem obsegu zmanjšamo z nastavitvijo parametrov PD regulatorja (K_P, K_V).

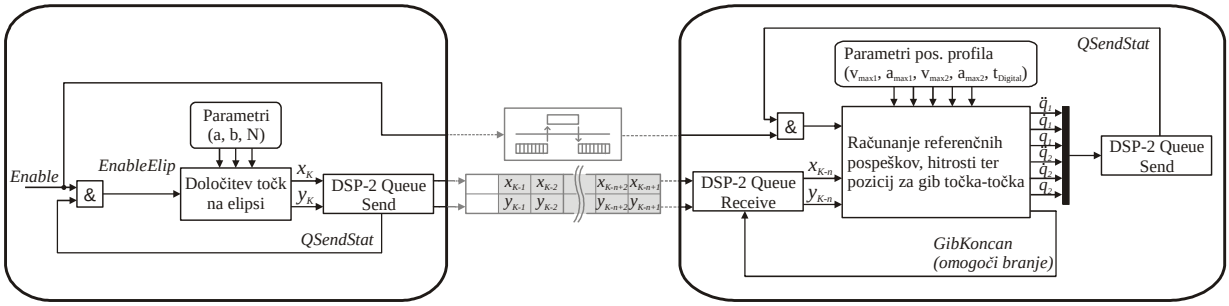
4.3 Prenos podatkov med opravili in eksperimentalni rezultati

Pred samo izvedbo eksperimenta je bila narejena časovna analiza izvajanja opravil, s katero so ocenjeni maksimalni časi izvajanja posameznega opravila. Na podlagi atributov opravil (tabela 1) ter časa, potrebnega za preklop konteksta, je z algoritmom za preverjanje izvedljivosti množice opravil določena ustrezna perioda urinega takta, ki znaša T_{tick} = 300 µs.

Tabela 1: Atributi opravil

n	C_n (µs)	T_n (µs)	P_n
1 (*Opravilo1*)	10	7200	1
2 (*Opravilo2*)	360	1800	2
3 (*Opravilo3*)	100	600	3

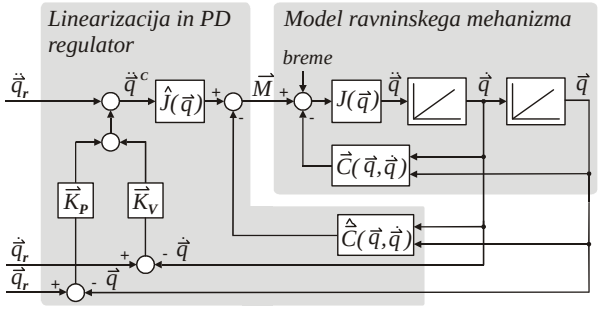
Izvedljivost opravil še ne pomeni pravilnega delovanja večopravilnega sistema, če podatki med opravili zaradi različnih period izvajanja opravil niso pravilno sinhronizirani. Prenosu podatkov, pri katerem ne prihaja do izgube podatkov, so namenjene sporočilne vrste, ki delujejo po principu FIFO (First In First Out) registra. Delu s sporočilnimi vrstami sta v knjižnici *DSP-2 Task Blockset* namenjena dva bloka. Blok *DSP-2 Queue Send* je namenjen pisanju podatkov v sporočilno vrsto, blok *DSP-2 Queue Receive* pa je namenjen branju podatkov iz sporočilne vrste. Bloka imata statusna signala, ki naznanjata nezmožnost pisanja v sporočilno vrsto pri bloku *DSP-2 Queue Send*



Slika 5: Prenos podatkov med *opravilom 1* in *opravilom 2*
Figure 5: Data transfer between *Task1* and *Task2*

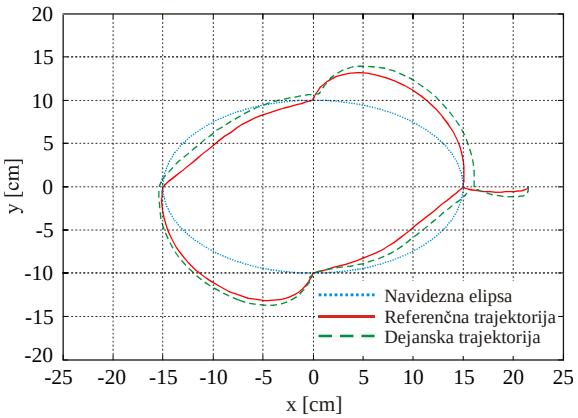
oz. nezmožnost branja iz sporočilne vrste pri bloku *DSP-2 Queue Receive*. Pri implementaciji algoritma vodenja sta uporabljeni dve sporočilni vrsti. Prva je namenjena prenosu koordinat točke na elipsi med *opravilom 1* in *opravilom 2* (slika 5), medtem ko je druga sporočilna vrsta namenjena za prenos referenčnih vrednosti med *opravilom 2* in *opravilom 3*.

Eksperimentalni rezultat gibanja vrha mehanizma (slika 7), velja za gib, določen s podatki: *a=0.15*, *b=0.10* in *N=4*. Statični pogrešek med referenčno in dejansko trajektorijo gibanja vrha mehanizma je pričakovan, saj metoda izračunanega navora ne vsebuje integralnega dela, ki bi odpravljal statični pogrešek. Večina pogreška gibanja vrha mehanizma je posledica pogreška pri gibanju druge osi, ki nastane zaradi elastičnosti pogonskega jermena.



Slika 6: Metoda izračunanega navora

Figure 6: Computed torque control scheme



Slika 7: Gibanje vrha mehanizma pri *a=15* cm, *b=10* cm in *N = 4*

Figure 7: Movement of the top of the robot arm at *a=15 cm*, *b= 10 cm* and *N=4*

5 Sklep

V članku je bila predstavljena implementacija prosto dostopnega operacijskega sistema realnega časa na robotskem krmilniku DSP-2. V sklopu zadane naloge je bilo izdelano ogrodje, s pomočjo katerega je bistveno olajšana implementacija algoritmov z uporabo OS realnega časa na DSP-2 robotskem krmilniku. Nabor realiziranih blokov (DSP-2 Task Blockset) je zdaj sestavni del DSP-2 knjižnice za Simulink, s čimer je omogočena hitra izvedba algoritmov z ali brez uporabe operacijskega sistema realnega časa na omenjenem krmilniku. Vpeljava OS ima žal nekatere slabosti, kot sta npr. povečana zahteva po pomnilniku (pomnilnik za jedro OS, pomnilnik za sklad posameznega opravila) in dodaten čas, ki je potreben za menjavo kontekstne vsebine, zato smiselnost uvedbe operacijskega sistema v samo aplikacijo ni trivialna, ampak zahteva podrobno analizo aplikacije.

6 Literatura

[1] P. Horacek, "Laboratory experiments for control theory courses: A survey," *Annual Reviews in Control*, vol. 24, pp. 151-162, 2000.

[2] J. Apkarian and K. J. Astrom, "A laptop servo for control education," *IEEE Control System Magazine*, vol. 24, no. 5, pp. 70-73, 2004.

[3] D. S. Bernstein, "The quanser DC motor control trainer individual or team learning for hands-on control education - Product Review," *IEEE Control Systems Magazine*, vol. 25, no. 3, pp. 90-93, 2005.

[4] C. Rusu and I. Birou, "Matlab Graphical Interface for the DSK243 system used to control a BLDC Motor," in *Proc. IEEE International Conference on Automation, Quality and Testing, Robotics*, 2006, pp. 452-456.

[5] D. Hercog, *DSP-2 Library for Simulink User's Manual*. Maribor, Slovenia: Fac. Electr. Eng. and Comput. Sci., Univ. Maribor, 2006. [Online]. Available: http://www.ro.feri.uni-mb.si/projekti/dsp2

[6] D. Hercog and K. Jezernik, "Rapid control prototyping using MATLAB/Simulink and a DSP-based motor controller," *International Journal of Engineering Education*, vol. 21, no. 4, pp. 596-605, 2005.
[7] S. Uran, D. Hercog, and K. Jezernik, "Experimental control learning based on DSP-2 learning module," in *Proc. of IEEE International Conference on Industrial Technology (ICIT)*, 2004, pp. 310-315.
[8] Q. Li and C. Yao, *Real-Time Concepts for Embedded Systems*, CMP Books, 2003.
[9] FreeRTOS. A FREE open source RTOS for small embedded real time systems. [Online]. Available: http://www.freertos.org/.
[10] K. Andersson and R. Andersson, "A comparison between FreeRTOS and RT-Linux in embedded real-time systems," 2005, [Online]. Available: www.streambag.se/files/ rtproj.pdf.
[11] M. Španer, "Dvoosni aktivni mehanizem," *Zbornik 14. Elektrotehniške in računalniške konference (ERK)*, pp. 247-250, 2005.

Darko Hercog je diplomiral leta 2001 na Fakulteti za elektrotehniko računalništvo in informatiko v Mariboru, kjer je tudi zaposlen kot asistent. Njegovo raziskovalno področje obsega sisteme realnega časa, sisteme za hitro izvedbo algoritmov vodenja in načrtovanje ter izvedbo oddaljenih laboratorijev.

Aljaž Kapun je diplomiral leta 2004 na Fakulteti za elektrotehniko računalništvo in informatiko v Mariboru, kjer je zaposlen kot mladi raziskovalec. Ukvarja se z modeliranjem in vodenjem servosistemov.

Karel Jezernik je diplomiral leta 1968, magistriral leta 1974 in doktoriral leta 1976 na Fakulteti za elektrotehniko Univerze v Ljubljani. Kot docent se je leta 1976 zaposlil na Univerzi v Mariboru, kjer je v letu 1985 postal redni profesor in vodja Inštituta za robotiko. Težišče njegovega raziskovalnega in pedagoškega dela je na področju avtomatskega vodenja, robotike, energetske elektronike in električnih motornih pogonov.