Vodenje avtomatskih drsnih vrat z avtomati stanj

Franc Hanžič, Karel Jezernik, Slavko Cehner
Univerza v Mariboru, FERI
Smetanova 17, 2000 Maribor
Doorson d.o.o.
Milenkova ulica 9, 2000 Maribor
hanzicf@siol.net, karel.jezernik@uni-mb.si

Automatic sliding door control with state machine

Today's product development time is very fast, or we can say concurrence competition for new product manufacture in a short time as can be. Use of the software development and design tools is a must for fastest product development in these times. These kinds of development have negative properties which points in product function stability. This paper describes software code design in graphical method (finite state machine) that is easier for discover eventual software bugs, which frequently appear in fast development. We have to consider on a working collective (programmers, developers, servicemen etc.), which frequently swaps with the new one. New working collective have to know product's software function and properties which can be easier presented with graphical method. A difficulty level of function understanding will be presented on automatic sliding door control between the graphical method and programming language C in finite state machine. Finite state machine is graphical programming language with blocks (states), connections between them (transitions), which include events and actions.

Kratek pregled prispevka

Dandanes se soočamo z zelo hitrim časom razvoja produktov, tekmovanje s konkurenco za nov izdelan izdelek v čim krajšem času. V tako hitrem razvoju je potrebno upoštevati programska, razvojna in oblikovna orodja, ki pomagajo pri hitrejši izdelavi izdelka. Takšen način izdelave ima negativne vplive, ki so usmerjeni na zanesljivost delovanja. Prispevek opisuje delovanje in oblikovanje programske kode, katera je predstavljena v grafični metodi (avtomati stanj) s tem pa tudi enostavnejše odkrivanje morebitnih programskih napak pri hitrem razvoju. Prav tako je potrebno upoštevati, da se v gospodarstvu vedno zamenjuje kolektiv (programerji, razvojniki, serviserji itd.), kateremu je potrebno predstaviti delovanje in lastnosti izdelka, ki jih je razvil predhodni kolektiv. V prispevku je predstavljeno razumevanje delovanja avtomatskih drsnih vrat v grafični in v obliki C programske kode z metodo avtomata stanj. Avtomat stanj predstavlja bloke (stanja), ki so medsebojno povezani (tranzicije). Tranzicija predstavlja dogodek in akcijo.

1 Uvod

Vodenje avtomatskih drsnih vrat [1], ki so namenjene za prehod ljudi, poteka s pomočjo jermenskega mehanizma, rotacijskega motorja in krmilne enote. Krmilna enota ima mikrokrmilnik na katerem se izvaja programska koda, ki ima funkcijo vodenja vrat. Programska koda je kritični dejavnik za stabilno delovanje translacijskih vrat. Na prvi pogled vrata niso kompleksna za vodenje, saj jih samo odpiramo in zapiramo. Vendar v ozadju se skriva veliko drugih funkcij, katere zajemajo varnost, diagnostiko, komunikacije itd. S tem se kompleksnost programske kode stopnjuje in postaja vedno bolj nepregledna. Izgradnja programske kode za trenutne mikrokrmilnike poteka v programskem jeziku C, to pa ni edini jezik, ki se uporablja za izdelavo kode. Najbolj pomembno je oblikovanje programske kode, kajti z neustreznim oblikovanjem se izdela nepregleden, nestabilni program. Takšen način nas privede do zapravljanja dragocenega razvojnega časa in težko odpravljanje morebitnih napak v kodi. Čeprav je program izdelan brez napak, pa se morajo upoštevati primeri po dodatnih programskih funkcijah. Vključitev nove funkcije v neurejeno obstoječo kodo je zelo težko in zamudno in ponovno nastopa vprašanje zanesljivosti delovanja programske kode. V industrijskih projektih torej nastopa zahteva za izvedbo pregledne programske kode. Zamenjava kolektiva je neizbežna v podjetjih in tako je potrebno predati obstoječe projekte novemu članu kolektiva. Vpogled v neurejeno kodo predhodnega avtorja je za razumevanje dolgoročno in nepravilno vključevanje novih funkcij. V dolgoročnih projektih se lahko zamenja več avtorjev. Pri takšnem delu je potrebno upoštevati določena pravila urejevanja programske kode zaradi boljše preglednosti.

V nadaljevanju bo opisano urejevanje kode v avtomatu stanj. Avtomat stanj zajema dve vrsti urejanja, to sta grafični način in oblikovanje v programskem jeziku C. Grafični način je namenjen za lažje razumevanje delovanja kode za osebo, ki se ne spozna na programske jezike. Tekstovno oblikovanje pa se nanaša na izdelovalce programske kode. Bistvo tega članka je prikazati povezljivost med grafičnim in tekstovnim načinom vodenja drsnih vrat. Načina morata biti medsebojno berljiva za dobro razumljiv projekt.

2 Krmilna enota

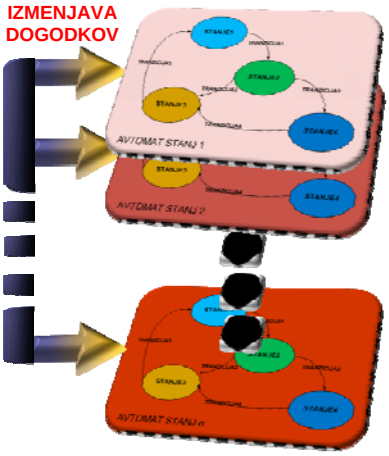
Vodenje avtomatskih vrat poteka preko krmilne enote (Slika 1) na katero so priključeni aktuatorji in senzorji, ki so nameščeni v vratih. Sama krmilna enota ima vključene komunikacije (CAN – Controller Area Network in serijsko RS232). CAN komunikacija zagotavlja komunikacijo z določenimi senzorji in univerzalnim modulom (vključuje požarno in redundantno delovanje vrat). Redundantnost zagotavlja, da se vrata odprejo ob okvari krmilne enote. Povezljivost z osebnim računalnikom zagotavlja uporabniku ali serviserju enostavno nastavljanje parametrov, prenos programske kode, diagnostiko in nadzor. Jedro krmilne enote, 32-bitni mikrokrmilnik ARM Cortex M3 LPC1768 [2], upravlja delovanje avtomatskih vrat. Programska koda, katera poganja mikrokrmilnik, je izdelana v programskem jeziku C s pomočjo programskega urejevalnika [3].

Slika 1: Prototipna krmilna enota - Doorson

3 Programska koda

Programska koda mikrokrmilnika je razdeljena na več opravil (komunikacije, varnost, generator giba, upravljanje z vrati, upravljanje z vhodi krmilnika, upravljanje s komandnim stikalom itd.) za ustrezno delovanje avtomatskih vratih. Vsako opravilo je oblikovano v avtomatu stanj. Majhen operacijski sistem pa časovno preklaplja med

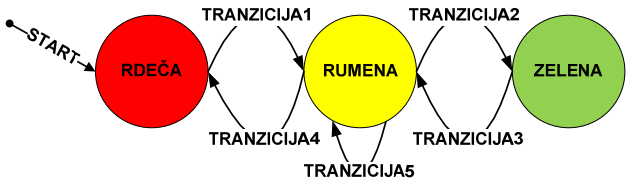
opravili. Opravila predstavljajo avtomate stanj, ki so medsebojno povezani s podatkovnim kanalom (Slika 2).



Slika 2: Programska koda z večjim številom opravil (avtomati stanj)

4 Avtomat stanj

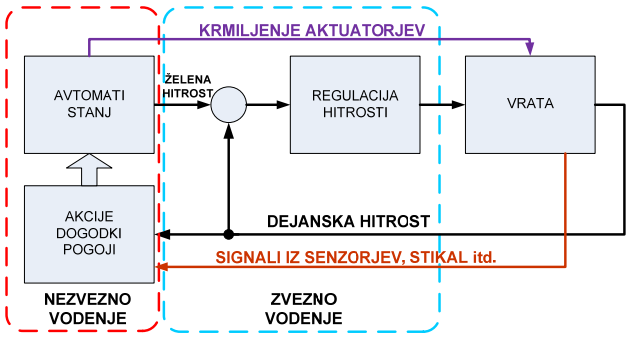
Tehnika programiranja z metodo avtomata stanj ni novost saj je že zelo stara in poznana. Teorija avtomata stanj je preprosta, zajema stanja, tranzicije, dogodke in akcije. Prehodi med stanji se imenujejo tranzicije. Začetek tranzicije izvede dogodek, konec tranzicije pa predstavlja akcijo. Dogodek ima pogoje, torej ob izpolnjenih pogojih se zgodi dogodek. Pogoji se lahko nanašajo na vhode krmilne enote ali spremenljivko v programu. Dogodek se lahko zgodi ob enem pogoju ali v kombinaciji z več pogoji z logičnimi funkcijami (AND, OR, NAND itd.). Akcija napoveduje, kaj se bo zgodilo ob določenem dogodku. Na semaforju imamo tri stanja, to so rdeča, rumena, zelena. Ta stanja so med seboj povezna s tranzicijami. Preprosti semafor deluje s pomočjo štetja časa, ko preteče čas, je prisoten dogodek »čas je potekel za rdečo luč«, po dogodku pa je prisotna akcija »preklopi na rumeno«. Dogodek in akcijo predstavlja tranzicija, v primeru tranzicija1 (Slika 3), povezavo med stanjem rdeča in rumena. Na primeru opazimo, da ima stanje rumena tranzicijo5. Ta se izvaja v primeru napake sistema ali v posebnem režimu, kjer nam utripa rumena luč. Dogodki in akcije se lahko izvajajo v enem samem stanju, kar predstavlja stanje rumena.



Slika 3: Primer avtomata stanj za semafor

5 Avtomat stanj v avtomatskih vratih

Kot je že bilo povedano, imajo avtomatska vrata več avtomatov stanj. V prispevku je osnovno predstavljen avtomat stanj za delovanje vrat, podrobneje je predstavljen avtomat stanj za generiranje giba (hitrostni profil). Oba avtomata si izmenjujeta informacije. Avtomat stanj osnovnega delovanja daje zahtevo generatorju giba (avtomatu stanja za generiranje giba), ta pa mu sporoči, da je opravil nalogo. Avtomat stanj za generiranje giba se je zgradil v programskem okolju Matlab/Simulink z orodjem »Stateflow«. Cilj naloge je izdelati gibanje vrat z S hitrostnim profilom, ki bi se zamenjal s trapeznim hitrostnim profilom. Splošni algoritem vodenja vrat (Slika 4) je razdeljen na zvezno in nezvezno vodenje. Zvezno vodenje predstavlja regulacijo ter opravljanje mikrokrmilniške periferije. Nezvezno vodenje pa predstavlja avtomate stanj, ki vodijo in upravljajo vrata.



Slika 4: Algoritem vodenja vrat

5.1 Avtomat stanj za osnovno delovanje vrat

V osnovi imajo vrata pet stanj. Ta stanja so inicializacija, odprto, zapiranje, zaprto in odpiranje. Ob vklopu avtomatskih vrat na

napajanje (230VAC) gredo vrata v inicializacijo, kjer se ponastavijo parametri vrat (hitrosti, čas odprtosti, zaklep itd.) in ponastavljanje dolžine giba. Pred ponastavitvijo dolžine giba, vrata izvedejo cikel zapiranja in odpiranja. Pri končnih legah se krila vrat naslonijo na odbijač in s tem krmilnik zazna končno lego (meritev toka ali pa hitrost nenadno pade), tako se v pomnilnik vpiše dejanski položaj. Iz podatka položaja zaprtosti in odprtosti se izračuna dolžina giba kril. Po uspešni inicializaciji gredo vrata v stanje odprto. Ko preteče čas ali je izpolnjen kakšen drugi pogoj (zaklep, požarni signal itd.) gredo v stanje zapiranje. Tranzicija iz stanja zapiranje v stanje zaprto se izvede ob dosegu končnega zaprtega položaja. Ob prisotnosti osebe (ali drugi pogoj) se izvede tranzicija med stanjem zaprto in stanjem odpiranje. Ob dosegu končnega odprtega položaja se izvede tranzicija med stanjem odpiranjem in stanjem odprto. Tako se cikel stanj ponavlja ob izpolnjenih pogojih. Opisno delovanje je prikazano v tabeli tranzicij (Tabela 1).

Tabela 1: Tranzicijska tabela

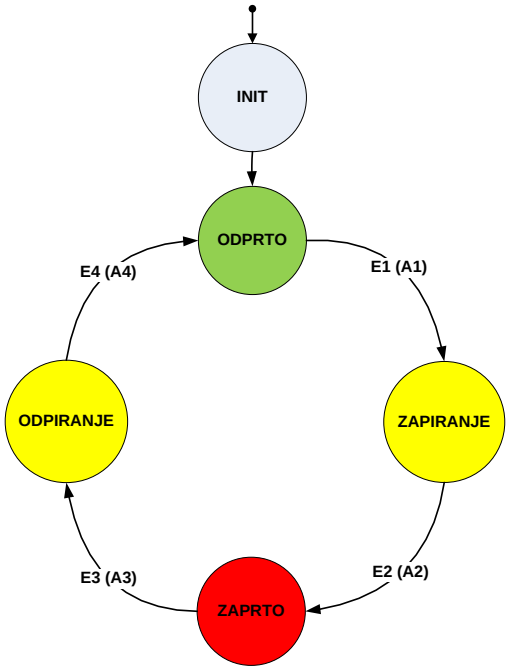
DOGODKI	POGOJI	OPIS DOGODKA
E1	Meritev časa, požarni signal, signal zaklepa itd.	Pretekel je čas odprtosti, prisoten požar, zaklep vrat
E2	Meritev položaja	Vrata dosegla končni položaj pri zapiranju
E3	Signal senzorja, signal komandnega stikala itd.	Prisotna oseba, sprememba režima na odprto
E4	Meritev položaja	Vrata dosegla končni položaj pri odpiranju
AKCIJA	POGOJ	OPIS AKCIJE
A1	Stanje ODPRTO in dogodek E1	Zapri vrata
A2	Stanje ZAPIRANJE in dogodek E2	Ustavi vrata
A3	Stanje ZAPRTO in dogodek E3	Odpri vrata
A4	Stanje ODPIRANJE in dogodek E4	Ustavi vrata

TRANZICIJSKA TABELA				
DOGODKI / STANJA VRAT	E1	E2	E3	E4
ODPRTO	A1/ZAPIRANJE	-	-	-
ZAPIRANJE	-	A2/ZAPRTO	-	-
ZAPRTO	-	-	A3/ODPIRANJE	-
ODPIRANJE	-	-	-	A4/ODPRTO

Princip je prikazan kot osnovni in ne predstavlja enako delovanje v komercialnih avtomatskih drsnih vratih. Popolno delovanje še vključuje dodatna stanja in dodatne tranzicije med nekaterimi stanji.

Tabelo tranzicij predstavimo v grafični obliki (Slika 5), katero lahko primerjamo z delovanjem semaforja (Slika 3).

Avtomat stanj za osnovno delovanje vrat, komunicira z avtomatom stanja za generacijo giba vrat. Osnovni avtomat stanj daje opravila drugemu avtomatu, ta pa mu da odgovor, da je končal z opravilom.

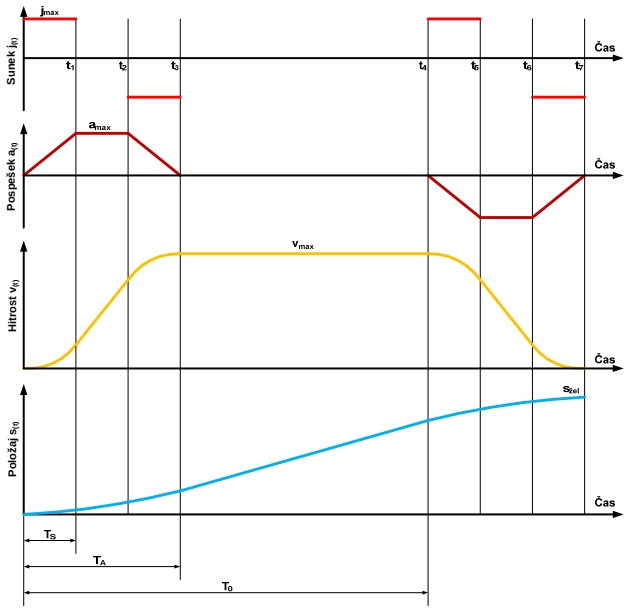


Slika 5: Avtomat stanj pri avtomatskih drsnih vratih

5.2 Avtomat stanj za generacijo giba

Generator giba daje referenčne vrednosti (pospešek, hitrost in položaj) za izvedbo odpiranja ali zapiranja vrat. Trenutna tehnika uporablja referenčno vrednost hitrosti, ki s pomočjo regulatorja regulira trenutno hitrost z dejansko (Slika 4). Avtomat stanj prejema opravila (akcije), kot so odpri, zapri, ustavi in oddajniku vrne sporočilo odprto, zaprto, zaustavljeno (dogodki). Generator je lahko izveden v več oblikah, to so trapezna, zvončasta, S, $\sin^2$ itd. Vsaka izmed metod ima svoje prednosti in slabosti. Trenutni avtomat stanj uporablja generator trapezne hitrosti in je enostaven za izdelavo. Vendar trapezna oblika hitrosti ima nenadne sunke (nezveznost med prehodi hitrosti) in povzroča dodatna nihanja in večjo potrošnjo električne energije pri vodenju

vrat. Za rešitev nezveznosti in drugih negativnih lastnosti bo uporabljen generator S hitrostnega profila, ki ima zveznost med prehodi. Za primerjavo s trapeznim profilom ima S profil zaokrožitve pri spremembi vrednosti hitrosti in s tem zveznost med prehodi. Profil delimo na 7 stanj (Slika 6), kateri se bodo uporabili za predstavitev v avtomatu stanj. Generator izdela profil po podanih zahtevanih vhodnih parametrih (maksimalni sunek, pospešek, hitrost in razdaljo giba). Z sunkom se določuje intenzivnost prehodov pri hitrosti. Večji je sunek, bolj so zaostreni prehodi. Profil ima sedem prehodov, ki so odvisni od časa in so označeni od t_1 do t_7.



Slika 6: S profil hitrosti

Kjer je:

$j_{\max}$ – maksimalni sunek

$a_{\max}$ – maksimalni pospesek

$v_{\max}$ – maksimalna hitrost

s_{zel} – zeleni polozaj

T_S – cas sunka

T_A – cas pospesevanja

T_0 – cas do zaustavljanja

Sunek, pospešek, hitrost in položaj so parametri, ki se določijo glede na lastnosti vrat. Maksimalni sunek j_{max} se navezuje na strmino pospeška. Pri S hitrostnem profilu je pospešek trapezne oblike. V primeru, ko pa so sunki veliki, je pospešek pravokotne oblike in tako trapezna oblika hitrosti. Sunek pri drsnih vratih se uporablja za nivo zveznosti med različnimi hitrostmi. Želja je, da so vrata pri odpiranju zelo hitra in seveda ni zaželeno, da je prehod iz mirovanja v premikanje velika razlika hitrosti, vzrok so velike mehanske obremenitve in vibracije. S stopnjo sunka lahko določujemo, da na začetku vrata postopno zvišujejo hitrost. Maksimalni pospešek a_{max} se navezuje na strmino hitrosti. Večji je pospešek, vrata hitreje dosežejo želeno hitrost. Pri odpiranju je zaželen večji pospešek, kot pri zapiranju. Pri pospešku se morajo upoštevati meje glede na sposobnost mehanizma vrat (motor). Motor, ki poganja vrata, ne zmore izvesti giba s katerokoli vrednostjo pospeška. Maksimalni pospešek se navezuje na sposobnost motorja (se navezuje na dovoljeni tok motorja). Maksimalna hitrost v_{max} je omejena na maso vrat, katera se navezuje na varnost pred trki ovir (osebe, predmeti). Pri težjih vratih mora biti zagotovljena manjša hitrost, kot pri lažjih vratih. Težja vrata z višjo hitrostjo imajo večjo vztrajnost, katera je nevarna za ljudi ob morebitnih trkih.

Za izvedbo profila je potrebno poznati enačbe za pospešek (1), hitrost (2) in položaj (3).

$$a_{(t)} = \begin{cases} j_{\max} \cdot t & 0 < t \le t_1 = T_S \\ a_{\max} & t_1 < t \le t_2 = T_A - T_S \\ -j_{\max} \cdot (t - t_2) + a_{\max} & t_2 < t \le t_3 = T_A \\ 0 & t_3 < t \le t_4 = T_0 \\ -j_{\max} \cdot (t - t_4) & t_4 < t \le t_5 = T_0 + T_S \\ -a_{\max} & t_5 < t \le t_6 = T_0 + T_A - T_S \\ j_{\max} \cdot (t - t_6) - a_{\max} & t_6 < t \le t_7 = T_0 + T_A \end{cases} \quad (1)$$

$$v_{(t)} = \begin{cases} \frac{1}{2} \cdot j_{\max} \cdot t^2 & 0 < t \le t_1 = T_S \\ a_{\max} \cdot (t - t_1) + v_{(t_1)} & t_1 < t \le t_2 = T_A - T_S \\ -\frac{1}{2} \cdot j_{\max} \cdot (t - t_2)^2 + a_{\max} \cdot (t - t_2) + v_{(t_2)} & t_2 < t \le t_3 = T_A \\ v_{(t_3)} & t_3 < t \le t_4 = T_0 \\ -\frac{1}{2} \cdot j_{\max} \cdot (t - t_4)^2 + v_{(t_4)} & t_4 < t \le t_5 = T_0 + T_S \\ -a_{\max} \cdot (t - t_5) + v_{(t_5)} & t_5 < t \le t_6 = T_0 + T_A - T_S \\ \frac{1}{2} \cdot j_{\max} \cdot (t - t_6)^2 - a_{\max} \cdot (t - t_6) + v_{(t_6)} & t_6 < t \le t_7 = T_0 + T_A \end{cases} \quad (2)$$

$$v_{(t)} = \begin{cases} \frac{1}{6}\cdot j_{\max}\cdot t^3 & 0 < t \le t_1 = T_S \\ \frac{1}{2}\cdot a_{\max}\cdot(t-t_1)^2 + v_{(t_1)}\cdot(t-t_1) + s_{(t_1)} & t_1 < t \le t_2 = T_A - T_S \\ -\frac{1}{6}\cdot j_{\max}\cdot(t-t_2)^3 + \frac{1}{2}\cdot a_{\max}\cdot(t-t_2)^2 + v_{(t_2)}\cdot(t-t_2) + s_{(t_2)} & t_2 < t \le t_3 = T_A \\ v_{(t_3)}\cdot(t-t_3) + s_{(t_3)} & t_3 < t \le t_4 = T_0 \\ -\frac{1}{6}\cdot j_{\max}\cdot(t-t_4)^3 + v_{(t_4)}\cdot(t-t_4) + s_{(t_4)} & t_4 < t \le t_5 = T_0 + T_S \\ -\frac{1}{2}\cdot a_{\max}\cdot(t-t_5)^2 + v_{(t_5)}\cdot(t-t_5) + s_{(t_5)} & t_5 < t \le t_6 = T_0 + T_A - T_S \\ \frac{1}{6}\cdot j_{\max}\cdot(t-t_6)^3 - \frac{1}{2}\cdot a_{\max}\cdot(t-t_6)^2 + v_{(t_6)}\cdot(t-t_6) + s_{(t_6)} & t_6 < t \le t_7 = T_0 + T_A \end{cases} \quad (3)$$

Potrebno je tudi izračunati časovne intervale (T_S, T_A, T_0), kateri so odvisni od dolžine giba, maksimalnega pospeška, sunka in hitrosti. Za izračun časovnih intervalov so vključene različne enačbe in pogoji, ki izbirajo način izračuna časovnega intervala. Z upoštevanjem pogojev se časovni intervali izračunajo z naslednjimi enačbami (4,5,6).

$$T_S = \frac{a_{\max}}{j_{\max}}$$

$$T_A = \frac{v_{\max}}{a_{\max}} + T_S$$

$$T_0 = \frac{s_{zel}}{v_{\max}} \quad (4)$$

$$T_S = \frac{a_{\max}}{j_{\max}}$$

$$T_A = \frac{-a_{\max}^2 + \sqrt{a_{\max}^4 + 4\cdot a_{\max}\cdot j_{\max}^2 \cdot s_{zel}}}{2\cdot a_{\max}\cdot j_{\max}} + T_S$$

$$T_0 = T_A \quad (5)$$

$$T_S = \frac{a_{\max}}{j_{\max}}$$

$$T_A = 2\cdot T_S$$

$$T_0 = T_A \quad (6)$$

Predstavljene enačbe zajemajo samo pozitivno generiranje hitrosti. Avtomat stanj avtomatskih vrat še vključuje negativno hitrost in stop funkcijo. Izvedba S hitrostnega profila je izdelana v programskem orodju Matlab/ Simulink v avtomatu stanj in ima naslednje vhode in izhode (Tabela 2).

Tabela 2: Vhodi in izhodi generatorja

Vhod	Tip podatka	Opis
MAXPOS	število	Vhod za določitev končnega položaja [m]
SETTIME	število	Vhod za določitev časovnega intervala T_i v katerem se izvaja avtomat stanj (T_i < 1e-3)
ACT_POS	število	Vhod za dejanski položaj
ACT_VEL	število	Vhod za dejansko hitrost
MAX_VELP	število	Vhod za določitev maksimalne hitrosti pozitivnega hitrostnega profila [m/s]
MAX_VELN	število	Vhod za določitev maksimalne hitrosti negativnega hitrostnega profila [m/s]
MAXACC_P	število	Vhod za določitev maksimalnega pospeška pozitivnega hitrostnega profila [m/s²]
MAXACC_N	število	Vhod za določitev maksimalnega pospeška negativnega hitrostnega profila [m/s²]
JERK_P	število	Vhod za določitev maksimalnega sunka pozitivnega hitrostnega profila [m/s³]
JERK_N	število	Vhod za določitev maksimalnega sunka negativnega hitrostnega profila [m/s³]
PROMACHINE_IN	število	Vhod za upravljanje avtomata stanj (generator profila), komunikacija med raznimi drugimi avtomati stanj »0« - ne naredi ničesar »1« - izvedi pozitivni hitrostni profil »2« - izvedi negativni hitrostni profil »3« - izvedi zaustavljanje
JERK_S	število	Vhod za določitev maksimalnega sunka pri zaustavljanju [m/s³]
MAXACC_S	število	Vhod za določitev maksimalnega pospeška pri zaustavljanju [m/s²]
MAX_VELS	število	Vhod za določitev maksimalne hitrosti pri zaustavljanju [m/s]
Izhod	**Tip podatka**	**Opis**
ACC	število	Izhod iz generatorja profila, kateri zajema potek pospeška [m/s²]
POS	število	Izhod iz generatorja profila, kateri zajema potek položaja [m]
VEL	število	Izhod iz generatorja profila, kateri zajema potek hitrosti [m/s]
PROMACHINE_OUT	število	Izhod iz generatorja profila, kateri zajema informacije o prehodih med stanji
STATUS	število	Izhod iz generatorja profila, kateri zajema informacije o izvedbi hitrostnega profila, komunikacija med raznimi drugimi avtomati stanj »0« - v izvajanju »1« - izvedel pozitivni hitrostni profil »2« - izvedel negativni hitrostni profil »3« - izvedel zaustavljanje

Avtomat stanj zajema stanje čakanja (IDLE), stanje za začetne izračune in nastavitve generatorja profila (INIT), stanje pozitivnega hitrostnega profila (POSITIVE), stanje negativnega hitrostnega profila (NEGATIVE) in stanje zaustavljanja (STOP), skupaj 5 stanj. Stanja so med seboj povezana s prehodi (tranzicijami), kateri imajo zastavljen pogoj (dogodek) za dovoljenje prehoda iz enega stanja v drugega (akcija). Pogoji so spremenljivke, ki se lahko navezujejo na periferijo (digitalni vhodi, komunikacije itd.) ali programsko, katere se spreminjajo v določenem stanju. Avtomat stanj ima naslednje tranzicije (pogoji, dogodki, akcije) in stanja (Tabela 3).

Tabela 3: Tabela tranzicij z opisi

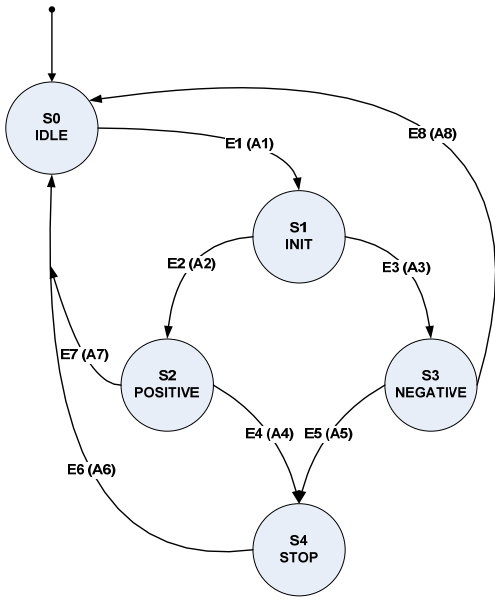
Stanja	Opis							
S0	Stanje čakanja (IDLE)							
S1	Stanje začetnih izračunov in nastavitev osnovnih parametrov (INIT)							
S2	Stanje za izvedbo pozitivnega hitrostnega profila (POSITIVE)							
S3	Stanje za izvedbo negativnega hitrostnega profila (NEGATIVE)							
S4	Stanje za zaustavljanje (STOP)							
Tabela tranzicij								
Dogodki Stanja	E1	E2	E3	E4	E5	E6	E7	E8
S0	A1/S1	-	-	-	-	-	-	-
S1	-	A2/S2	A3/S3	-	-	-	-	-
S2	-	-	-	A4/S4	-	-	A7/S0	-
S3	-	-	-	-	A5/S4	-	-	A8/S0
S4	-	-	-	-	-	A6/S0	-	-

Pogoji	Tip podatka	Opis
PROMACHINE_IN	število	Vhod v avtomat stanj »PROFILE MACHINE INPUT«, s katerim določimo operacijo: »0« - ne naredi ničesar »1« - izvedi pozitivni hitrostni profil »2« - izvedi negativni hitrostni profil »3« - izvedi zaustavljanje
STATEINIT	logična spremenljivka	Spremenljivka, katera nosi informacijo o izvedbi začetnih nastavitev »0« - še niso nastavljene »1« - so nastavljene
NEGATIVEM	logična spremenljivka	Spremenljivka, katera nosi informacijo o stanju negativnega hitrostnega profila »0« - še ni končal negativnega profila »1« - je končal negativni profil
POSITIVEM	logična spremenljivka	Spremenljivka, katera nosi informacijo o stanju pozitivnega hitrostnega profila »0« - še ni končal pozitivnega profila »1« - je končal pozitivni profil
STOPPED	logična spremenljivka	Spremenljivka, katera nosi informacijo o stanju zaustavljanja »0« - še ni končal zaustavljanje »1« - se je zaustavil

Dogodki	Pogoji	Opis
E1	POSITIVEM==0&& PROMACHINE_IN==1 || NEGATIVEM==0&& PROMACHINE_IN==2	Dodeljena zahteva za gib vrat
E2	STATEINIT==1&&PROMACHINE_IN==1	Opravljena inicializacija in dodeljena zahteva za odpiranje vrat
E3	STATEINIT==1&& PROMACHINE_IN==2	Opravljena inicializacija in dodeljena zahteva za zapiranje vrat
E4	PROMACHINE_IN==3	Zaznana ovira
E5	PROMACHINE_IN==3	Zaznana ovira ali prisotna oseba
E6	STOPPED==1	Vrata so se ustavila
E7	POSITIVEM==1	Vrata so odprta
E8	NEGATIVEM==1	Vrata so zaprta

Akcije	Pogoji	Opis
A1	S0&&E1	Opravi inicializacijo
A2	S1&&E2	Opravi pozitivni profil hitrosti
A3	S1&&E3	Opravi negativni profil hitrosti
A4	S2&&E4	Opravi STOP funkcijo v pozitivnem profilu
A5	S3&&E5	Opravi STOP funkcijo v negativnem profilu
A6	S4&&E6	Vrni se v pripravljenost iz STOP funkcije
A7	S2&&E7	Vrni se v pripravljenost iz pozitivnega profila hitrosti
A8	S3&&E8	Vrni se v pripravljenost iz negativnega profila hitrosti

Avtomat stanj s pomočjo tranzicijske tabele predstavimo v grafični obliki (Slika 7).

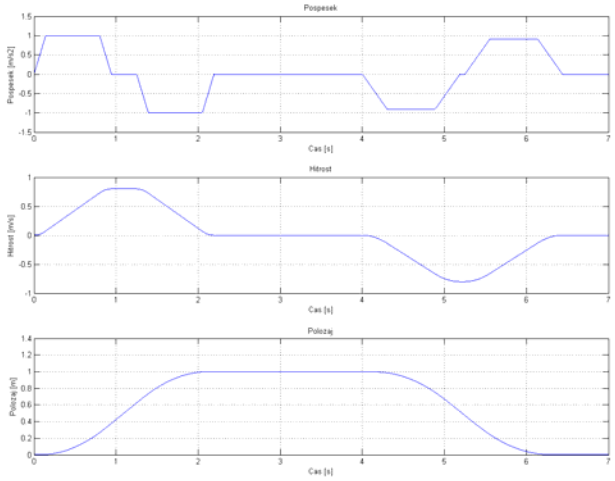


Slika 7: Avtomat stanj generatorja giba

6 Simulacijski eksperimenti

Generator hitrostnega profila je bil uspešno preizkušen v Matlab/Simulink okolju. Izvedlo se je odpiranje in zapiranje in zaustavljanje med gibanjem. Predstavljen test je zajel odpiranje in zapiranje z naslednjimi parametri (Slika 8).

Začetek odpiranja pri času 0 s
Začetek zapiranja pri času 4 s
Dolžina odpiranja/zapiranja:1 m
Maksimalna hitrost (pozitivni/negativni):0.8 m/s
Maksimalni pospešek (pozitivni):1 m/s^2
Maksimalni pospešek (negativni):0.9 m/s^2
Maksimalni sunek (pozitivni):7 m/s^3
Maksimalni sunek (negativni):3 m/s^3



Slika 8: Odzivi pri odpiranju in zapiranju

7 Oblika avtomata stanj v programskem jeziku C

Programska koda za krmilnik vrat je napisana v programskem jeziku C. Tako je potrebno grafično obliko avtomatov stanj pretvoriti v ustrezno obliko v C jeziku [4]. V nadaljevanju je predstavljen avtomat stanj generatorja v C jeziku. Program prikazuje samo preklop stanj in tranzicije in ne zajema kodo za izvedbo profila. Najprej se definirajo tranzicije (dogodki, akcije) in stanja (Slika 9). Pogoji so izvedeni z IF funkcijo in logičnimi operacijami. Pogoji so lahko oblikovani v matrični obliki ali pa se definirajo v IF stavkih. Pri izpolnitvi pogoja se izvede dogodek, po dogodku pa sledi akcija. Dogodek in akcijo predstavlja tranzicija, ki povezuje dve stanji. Tranzicija predstavlja funkcijo SWITCH v C jeziku, ki preklaplja med stanji (Slika 10).

```
/***Stanja, Spremenljivke, Akcije in dogodki*****/
/*****************STANJA**********************/
#define IDLE_S0             0x00  //Stanje IDLE
#define INIT_S1             0x01  //Inicializacija hit. prof.
#define POSITIVE_S2         0x02  //Gen. poz. hit. profila
#define NEGATIVE_S3         0x03  //Gen. neg. hit. profila
#define STOP_S4             0x04  //Hitro ustavljanje

/***************Spremenljivke*******************/
extern int PROMACHINE_IN;         //Vhod avtomata stanj
extern int POSITIVEM;             //Informacija o izvedbi poz. profila
extern int NEGATIVEM;             //Informacija o izvedbi neg. profila
extern int STOPPED;               //Informacija o izvedbi zaustavitve
extern int STATEINIT;             //Informacija o izvedbi iniciali.

/*********************DOGODKI******************/
#define E1 (POSITIVEM==0 && PROMACHINE_IN==0 || NEGATIVEM==0 &&
           PROMACHINE_IN==2)      //Dogodek E1
#define E2 (STATEINIT==1&&PROMACHINE_IN=1)
                                  //Akcija E2
#define E3 (STATEINIT==1&&PROMACHINE_IN=2)
                                  //Dogodek E3
#define E4 (PROMACHINE_IN==3)     //Dogodek E4
#define E5 (PROMACHINE_IN==3)     //Dogodek E5
#define E6 (STOPPED==1)           //Dogodek E6
#define E7 (POSITIVEM==1)         //Dogodek E7
#define E8 (NEGATIVEM=1)          //Dogodek E8

/*********************AKCIJE******************/
#define A1 (E1&&STATE==0)         //Akcija A1
#define A2 (E2&&STATE==1)         //Akcija A2
#define A3 (E3&&STATE==1)         //Akcija A3
#define A4 (E4&&STATE==2)         //Akcija A4
#define A5 (E5&&STATE==3)         //Akcija A5
#define A6 (E6&&STATE==4)         //Akcija A6
#define A7 (E7&&STATE==2)         //Akcija A7
#define A8 (E8&&STATE==3)         //Akcija A8
```

Slika 9: Definicije avtomata stanj v C jeziku

Obstajajo programski jeziki, ki podpirajo oblikovanje avtomatov stanj v grafičnem načinu »StateChart« in tako generira programsko kodo s pomočjo vgrajenega prevajalnika, tako da ni potrebe po oblikovanju C kode [5].

```
#include "definicije.h"

/*****************TRANZICIJE*****************/
if (A1) STATE=INIT_S1;            //Trancicija med S0 in S1
if (A2) STATE=POSITIVE_S2;        //Trancicija med S1 in S2
if (A3) STATE=NEGATIVE_S3;        //Trancicija med S1 in S3
if (A4) STATE=STOP_S4;            //Trancicija med S2 in S4
if (A5) STATE=STOP_S4;            //Trancicija med S3 in S4
if (A6) STATE=IDLE_S0;            //Trancicija med S4 in S0
if (A7) STATE=IDLE_S0;            //Trancicija med S2 in S0
if (A8) STATE=IDLE_S0;            //Trancicija med S3 in S0

/***************************STANJA************/
     switch (STATE)
     {
          case IDLE_S0:                //Stanje IDLE_S0
          //Ne naredi ničesar
          break;

          case INIT_S1:                //Stanje INIT_S1
          //Izračun spremenljivk
          break;

          case POSITIVE_S2:            //Stanje POSITIVE_S2
          //Izvedi poz. hit. profil
          break;

          case NEGATIVE_S3:            //Stanje NEGATIVE_S3
          //Izvedi neg. hit. profil
          break;

          case STOP_S4:                //Stanje NEGATIVE_S4
          //Izvedi zaustavljanje
          break;
     }
```

Slika 10: Oblika generatorja giba v C jeziku

8 Zaključek

Tehnologija ponuja vedno več modernejših metod krmiljenja, komunikacij in regulacij. Današnji mikrokrmilniki so zelo bogato opremljeni z zmogljivimi jedri in bogato periferijo. Vključitev teh metod v sistem izdelka pa potrebuje kompleksno programsko kodo, ki pa se mora izdelati s strani proizvajalca oz. projektanta. Čeprav današnja programska orodja ponujajo različne rešitve za enostavnejšo izdelavo kode, se stopnjuje problem pri hitrem razvoju in izdelavi zaradi nasičenega trga (konkurence, vedno večje zahteve kupcev itd.). Za izdelavo programske kode je potrebno uporabiti metodo, ki je enostavna za razumevanje, dograjevanje funkcij in vzdrževanja. Metoda z avtomati stanj je ena izmed rešitev za preglednost, razumevanje, vzdrževanje itd. Grafična oblika avtomata stanj je pomembna za izdelavo dobre dokumentacije, ki je pomembna za vzdževanje, odpravljanje morebitnih napak in izpopolnitve programske kode.

9 Potrditev

Operacijo delno financira Evropska unija, in sicer iz Evropskega socialnega sklada. Operacija se izvaja v okviru Operativnega programa razvoja človeških virov za obdobje 2007 – 2013, 1. razvojne prioritete: Spodbujanje podjetništva in prilagodljivosti, prednostne usmeritve 1.1.: Strokovnjaki in raziskovalci za konkurenčnost podjetij.

10 Literatura

[1] Proizvajalec avtomatskih drsnih vrat www.doorson.si
[2] Proizvajalec mikrokrmilnikov www.nxp.com
[3] Programsko orodje za programiranje mikrokrmilnikov www.keil.com
[4] Wagner F., Schmuki R., Wagner T., Wolstenholme P., Modeling Software with Finite State Machines: A Practical Approach, 16 Maj 2006
[5] Programsko orodje za grafično programiranje mikrokrmilnikov v avtomatu stanj http://www.iar.com/website1/1.0.1.0/371/1/