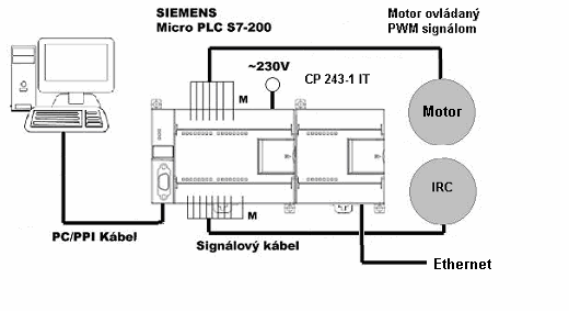
Ovládanie motora použitím IRC a PWM

Ľubomír Čapucha, Ing. Richard Balogh*
Ústav riadenia a priemyselnej informatiky
E-mail: night@ynet.sk

Abstrakt

Predkladané zariadenie vzniklo ako požiadavka na meranie rozbehových a dobehových charakteristík jednosmerného motora, ktorý je ovládaný PWM reguláciou. Charakteristiky je možné sledovať na PC z rozhraním RS-232 použitím špeciálneho software, alebo cez internet použitím štandartného web prehliadača, ktorý je schopný spracovať java applety. Základ zariadenia tvorí PLC firmy Siemens S7-200 rozšírené o IT modul CP 243-1 IT.

1. Úvod

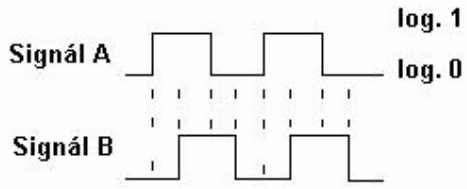


Obr.1. Schéma zapojenia

Na začiatku sme sa rozhodovali, aký systém použiť pre riadenie otáčok. Rozhodli sme sa pre ovládanie otáčok pomocou PLC. Použitím PLC sa zjednoduší elektronická schéma a bude možné jednoducho rozširovať zariadenie o ďalšie možnosti riadenia resp. merania. Ovládať je možné každý motor ku ktorému je možné zostrojiť prevodník z PWM (pulse width modulation) na akčný zásah pre motor. Momentálne sa počíta s tým, že sa ovláda jednosmerný motor. Preto ako prevodník slúži jednoduchý mostík ktorý má dva vstupy. Na prvý sa privádza PWM signál a na druhý smer otáčania motora. Ak by sa odoslal systému príkaz, ktorý by mal za úlohu zmeniť smer otáčania motora, tak sa nevyhodnotí pokiaľ PLC nedostane informáciu od IRC, že motor sa už neotáča. Pokiaľ by užívateľ chcel, môže PLC nastaviť tak, aby nečakalo na to pokiaľ motor zastaví, ale aby preplo výstup už pri nižších otáčkach. Princíp využitia toho, že PLC prepne smer otáčania motora pri nenulových otáčkach spočíva v tom, že aj prevodník si sníma rýchlosť otáčania motora. Potom ak mu príde príkaz na zmenu smeru motora vie, že má motor najskôr zastaviť a až potom pustiť motor do opačného smeru.

2. Snímanie otáčok

Otáčky sú snímané z IRC (inkrementálny rotačný snímač) snímača. IRC je napájané napätím 24 V, toto napätie je odoberané z PLC. IRC má rozlíšenie 2500 impulzov na otáčku a je schopné spracovať rýchlosť 7000 ot/min a maximálna dovolená konštrukčná rýchlosť je 12000 ot/min. Na výstupe IRC je niekoľko signálov.



Obr.2. Príklad výstupu signálov z IRC

Signály A a B, ich negácia, signál ktorý sa generuje raz za otáčku a jeho negácia a ešte signál identifikujúci chybu IRC. Signály majú dve úrovne low a high. úrovni low zodpovedá napätie 0V a úrovni high zodpovedá napätie 24V. Signály A a B majú nasledovnú funkciu. Pokiaľ sa otáča IRC v smere hodinových ručičiek signál A predbieha signál B o štvrtinu periódy. Ak sa otáča IRC opačným smerom predbieha signál B signál A o štvrtinu periódy. Tieto signály sú spracované systémom PLC. Systém PLC je už hotové zariadenie. Mi používame PLC firmy Siemens S7-200. PLC pracuje z 24V logikou alebo môže obsahovať aj relé výstupy.

3. Princíp komunikácie PC s PLC

PC môže komunikovať s PLC dvoma spôsobmi. Buď po linke RS-232, kde je potrebný prevodník RS-232 na RS-

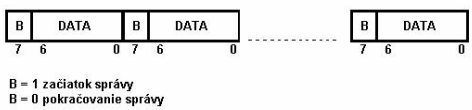
* Vedúci práce

485, alebo po Ethernete resp. Internete. Pri vývoji komunikácii po linke RS-232 vzniklo hneď niekoľko problémov. Prvým a to aj najväčším bol fakt, že PLC podporuje iba komunikáciu half-duplex. Znamená to toľko, že PLC nie je schopné súčasne prijímať a odosielať informácie. Ak by nastala situácia, že sa PLC pokúsi o súčasné vysielanie a zápis po linke RS-232 vyhodnotil by sa tento fakt ako chyba. Na strane PC sa nevyskytuje takýto problém pretože linka RS-232 má 9 vodičov a z nich slúži jeden na odosielanie a ďalší na prijímanie bytov. Ďalšia výhoda na strane PC je, že má komunikáciu bufferovanú. Bufferovaná komunikácia znamená, ak program nečakal na linke RS-232 nebude prijatý paket zahodený, ale bude uložený do buffera (zásobníka). Toto na strane PLC nie je možne, ak nebola spustená funkcia RCV a prišiel by paket od PC, PLC nebude registrovať, že mu bolo niečo odoslané. Na to aby sa odstránili všetky tieto nedostatky sme navrhli a odskúšali niekoľko protokolov:

1. prenos byte-byte
2. prenos pomocou premenlivej dĺžky paketu (správy)
3. prenos s pevnou dĺžkou paketu

3.1. Protokol byte-byte

Je založený na princípe komunikácie master (PC) slave(PLC). Master v pravidelných intervaloch prijíma a odosiela pakety dĺžky jeden byte. V týchto bytoch sa nesie informácia resp. príkaz čo sa má vykonať a to ako pre komunikáciu s PC do PLC aj pre komunikáciu s PLC do PC. Na synchronizovanie takejto komunikácie by musela byť použitá zložitá štruktúra paketu, nakoľko komunikačná zbernica PLC nedisponuje riadiacimi vodičmi. Pri použití veľkého paketu t.j. obsahoval by úvodnú postupnosť znakov, samotnú informáciu teda príkaz a údaje a nakoniec kontrolný súčet by bola linka príliš zahltená informáciami. Preto bola navrhnutá jednoduchšia štruktúra ktorá je znázornená na Obr. 3.



Obr. 3. Štruktúra protokolu byte-byte

Každý byte je rozdelený nasledovne: 7. bit nesie informáciu o tom či sa jedná o úvod správy tzn. ak má hodnotu 1 jedná sa o úvod správy a to signalizuje systému PLC (slave), aby vykonal predchádzajúci príkaz. Bity 6. až 0. nesú údaje. Tie sú rozdelené na príkaz a údaje napr. príkaz NOP nemá žiadne údaje a číslo príkazu je 0 tak má celú údajovú časť nastavenú na 0. Ako vidno ide o pomerne zložitú komunikáciu kde musí master v pravidelných intervaloch volať funkciu na odoslanie a prijatie jedného znaku. Plus musí informáciu rozložiť na 7 bitovú komunikáciu pretože 7. bit (MSB) nesie informáciu o začiatku správy. Pre ilustráciu je tu uvedená časť kódu ktorá sa starala o odosielanie jedného znaku z frontu príkazov Obr. 4.

```
1.  s7it = s7_data_send.begin();
2.  if(!s7_data_send.empty()){
3.      if(s7it->pos != s7it->length){
4.          printf("\n%c\n",s7it->data[s7it->pos]);
5.          send((char*)&s7it->data[s7it->pos],1);
6.          s7it->pos++;
7.      } else {
8.          printf("\n%c\n",s7it->chcksum);
9.          send((char*)&s7it->chcksum,1);
10.         s7_data_send.erase(s7_data_send.begin());
11.     }
12. }
```

Obr.4. Použitý kód na protokol byte-byte

Stručný popis kódu:

1. nastavíme sa na začiatok vektora
2. ak mame príkaz na odoslanie pokračujeme
3. ak sme ešte neodoslali všetky znaky odošli ďalší
4. výpis čo sa odoslalo
5. odošli znak
6. inkrementuj ukazovateľ na odosielané znaky o 1
7. ak sme odoslali všetky znaky správy odošli kontrolný súčet
8. vypíš odosielaný kontrolný súčet
9. odošli kontrolný sučet
10. vymaž príkaz z frontu príkazov

3.2. Protokol s premenlivou dĺžkou paketu

Tento protokol mal za úlohu odstrániť nedostatky predchádzajúceho protokolu. Premenlivá dĺžka paketu spočíva v tom, že PLC prijíma vždy určitý počet znakov a naše príkazy majú rôzny počet znakov napr. NOP má 2 znaky (číslo príkazu a kontrolný súčet) a SET_PWM_WIDTH má 4 znaky (číslo príkazu, 2 znaky pre údaje a kontrolný súčet). Preto by sa musel pri zmene príkazu odosielať špeciálny paket v ktorom by sa oznamovalo systému PLC aký veľký príkaz má očakávať. Tento paket by musel mať vždy dĺžku posledného paketu, inak by ho PLC nevedel vyhodnotiť. Dala by sa tu využiť aj možnosť PLC a to taká kde by PLC po určitej dobe prestalo prijímať znaky tzn. vypršal by timeout a PLC by začalo spracovávať údaje. Táto alternatíva mala veľkú nevýhodu a to tú, že by sa muselo neustále čakať kým vyprší timeout. Tento protokol nebol ani zrealizovaný, pretože už pri jeho vývoji sa narazilo na lepšie riešenie a to sa aj používa.

3.3. Protokol s pevnou dĺžkou paketu

Na prvý pohľad sa možno nezdá, ale tento protokol odstraňuje všetky predchádzajúce nedostatky. To čo sa môže zdať ako nevýhoda oproti predchádzajúcemu je, že má pevnú dĺžku paketu a príkazy z väčšou dĺžkou ako je určené by nemohli byť odoslané. Pri riešení tejto situácie spôsobom, kde by sa dĺžka paketu odvíjala len od najväčšieho príkazu by dochádzalo k zbytočnému zahlteniu komunikačnej linky. Preto ak je potrebné

odoslať príkaz s väčšou dĺžkou ako je určené, rieši sa to odoslaním príkazu na toľko krát koľko je potrebné. Program v PLC vie, že ak nedostal všetky údaje tak nezačne so spracúvaním údajov. Uvedieme príklad: príkaz NOP má číslo 0, príkaz SET_PWM_WIDTH má číslo 1, príkaz SET_IP má číslo 10 a 11. To znamená, že najskôr odošleme príkaz číslo 10, ktorý obsahuje prvé dva byty IP adresy. Potom pošleme príkaz číslo 11 ktorý obsahuje druhé dva byty IP adresy. Program v PLC po prijatí príkazu 11 zmení IP adresu IT modulu. Vec na ktorú treba dávať pozor je to, aby sa pakety odosielali v správnom poradí.

3.4. Komunikácia po ethernete

Na to aby bolo PLC schopné komunikovať po ethernete je potrebné PLC rozšíriť o modul CP 243-1 IT. Tento modul obsahuje všetko potrebné pre komunikáciu po ethernete resp. internete. Vie komunikovať cez FTP protokol, obsahuje HTTP server spolu s predpísanými Java appletmi. Tak isto je možné pomocou tohto modulu odoslať e-mail.

4. Inštalácia IT modulu

Pripojí sa IT modul k PLC ako je uvedené v manuáli [3]. Do siete ethernet sa pripojí štandartným UTP káblom s koncovkou RJ-45. Po zapnutí PLC je potrebné IT modul nakonfigurovať. Na konfiguráciu slúži wizard ktorý je súčasťou programu Step 7 MicroWin. Počas konfigurácie sa nastavujú parametre ako IP adresa, maska siete, default gateway, heslo ktoré sa použije pri FTP prenose resp. pri sledovaní činnosti PLC cez web rozhranie. Po skončení konfigurácie nám wizard vygeneruje potrebné funkcie a nastaví údaje v dátovej oblasti. Počas každého cyklu programu je potrebné volať funkciu ETHx_CTRL, ktorej popis je uvedený nižšie. Týmto by konfigurácia PLC a IT modulu bola hotová. Nasleduje vytvorenie web rozhrania.

4.1. Vytvorenie web rozhrania

Na vytvorenie web rozhrania môžeme použiť už predprogramované Java applety uložené v IT module alebo si môžeme napísať vlastné java applety. Podrobná konfigurácia IT modulu je popísaná v manuáli[2]. Súčasťou tohto manuálu sú aj podrobné príklady na vytvorenie www stránky.

Po napísaní www stránky, v akomkoľvek textovom editore (textový editor si nemôže ukladať do súboru formát textu!) napr. Notepad je potrebné nakopírovať tento textový súbor s koncovkou .htm do IT modulu. Na to môžeme použiť aplikáciu TotalCommander, alebo použiť príkazový riadok.

5. Komunikácia PC s PLC po linke RS-232

Ako bolo spomenuté vyššie, na komunikáciu sa používa prenos s pevnou dĺžkou paketu (správy). Paket má dĺžku 5 bytov. MSB má v sebe uložené číslo príkazu napr. 0 = NOP, 1 = SET_PWM_WIDTH. Nasledujúce 3 byty obsahujú údaje. Pre príkaz NOP majú hodnotu 0, pre príkaz SET_PWM_WIDTH obsahujú šírku PWM. Posledný byte LSB nesie kontrolný súčet. Používa sa pre kontrolu, či prišli všetky údaje správne.

Pred tým ako sa odošle príkaz PLC musí byť vytvorený v užívateľskej aplikácii. Spôsob vytvárania a napĺňania štruktúry je pevne stanovený v objekte S7-Interface (popísaný nižšie). Po vytvorení príkazu sa príkaz zaradí do fronty príkazov a čaká na odoslanie. Odosielanie prebieha v pravidelných intervaloch a počas jedného intervalu sa odošle jeden paket a jeden sa prijme.

6. Program v PLC

PLC systém má už v sebe zabudované všetky potrebné periférie ktoré využijeme pri ovládaní motora. Pri ovládaní motora použijeme nasledovné periférie: počítanie impulzov, prijímanie a odosielanie údajov, generovanie PWM impulzov.

Pre počítanie impulzov sa používa vysoko rýchlostne počítadlo impulzov HSC (High Speed Counter). Existujú štyri základné typy počítadiel: jednofázové počítadlo s vnútorným riadením smeru, jednofázové počítadlo z vonkajším riadením smeru, dvojfázové počítadlo s dvoma hodinovými vstupmi a A/B kvadratické počítadlo. Každý typ môžeme používať bez nulovacieho a štartovacieho vstupu, s nulovacím vstupom a bez štartovacieho vstupu alebo s štartovacím a nulovacím vstupom.

Pred použitím vysoko rýchlostného počítadla je potrebné nastaviť jeho režim inštrukciou HDEF. Táto inštrukcia sa volá iba raz a to na začiatku prvého programového cyklu.

Na komunikáciu medzi PC a PLC sa používa komunikácia typu freeport. Túto komunikáciu obsluhujú dva podprogramy XMT a RCV. Na to aby bol program v PC schopný komunikovať s PLC musí sa prepnúť aj prevodník PC/PPI do režimu freeport. Na prevodníku sa musia nastaviť aj parametre komunikácie ako prenosová rýchlosť, počet bitov a pod. Ak sa používa režim freeport na komunikáciu s PC je nutné pri programovaní programu v PLC počítať aj s časom, ktorý potrebuje prevodník na prepnutie a to v nasledujúcich stavoch:

- PLC S7-200 odpovedá na správy vysielané zo zariadenia s rozhraním RS-232. Po prijatí požiadavky z RS-232 musí PLC S7-200 pozastaviť vysielanie odpovedi po dobu dlhšiu alebo rovnú dobe prepnutiu kábla.

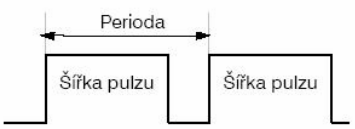
- Zariadenie s rozhraním RS-232 odpovedá na správy vysielané z PLC S7-200. Po prijatí odpovedi z RS-232 musí PLC S7-200 pozastaviť vysielanie ďalšieho požiadavku po dobu dlhšiu alebo rovnú dobe prepnutiu kábla.

Přenosová rychlost	Doba přepnutí	Nastavení (1 = nahoru)
38400 až 115200	0,5 ms	000
19200	1,0 ms	001
9600	2,0 ms	010
4800	4,0 ms	011
2400	7,0 ms	100
1200	14,0 ms	101

Tab. 1. Doby prepínania kábla

Z tohto vyplýva jedna nevýhoda a to, že ak chce PLC komunikovať so zriadením po zbernici RS-232, tak môže komunikovať iba v móde half-duplex. Ďalšiu vec, ktorú je potrebné vedieť pri komunikácii je, že pri prepnutí prepínača na PLC do stavu STOP sa komunikácia v režime freeport zastaví a obnoví sa normálna komunikácia napr. pre nahranie nového programu do PLC. V programe je komunikácia riešená cez prerušenia. V prvom programovom cykle sa nastavia všetky premenné potrebné pri používaní režimu freeport. Ďalej sa nastaví prerušenie na príznak koniec prijímania a odosielania správy. Potom už len povolíme prerušenia príkazom ENI a spustíme prijímanie príkazom RCV. Akonáhle sa príjme správa vygeneruje sa prerušenie dokončenie príjmu, v tomto prerušení sa pripojí časované prerušenie ktoré nastane za určitý počet milisekúnd. Počet milisekúnd závisí od použitej prenosovej rýchlosti. Počet milisekúnd potrebných na prepnutie kábla je uvedený v tab. 1 pre všetky podporované rýchlosti prevodníka PC/PPI. Po uplynutí nastavenej doby sa vykoná časované prerušenie. V tomto prerušení sa odpojí časované prerušenie ktoré sa práve vykonáva a odošle sa správa do PC. Je to potrebné pretože ak by sme časované prerušenie neodpojili nastalo by časované prerušenie znovu za určitý počet milisekúnd (koľko sme nastavili pri vytváraní) a správa by sa poslala znovu do PC. Toto nemôžme dovoliť pretože ak sa jedna správa odošle nastane prerušenie vysielanie dokončené v ktorom sa znovu začne prijímať a vznikla by chyba: pokus o súčasný zápis a čítanie z portu.

Výstupom z PLC je šírkovo modulovaný signál. Ak na jednom z výstupov Q0.0 resp. Q0.1 nastavíme, aby pracoval v móde PWM, nie je možné s týmto výstupom pracovať bežným spôsobom, pretože ho bude ovládať PWM generátor. PWM sa nastaví v prvom programovom cykle. Pre PWM reguláciu možno nastavovať periódu a šírku pulzu v milisekundových alebo v mikrosekundových časových intervaloch:



Obr. 5. Priebeh PWM signálu

- perióda: 50 µs až 65535 µs
  2 ms až 65535 ms
- šírka pulzu: 0 µs až 65535 µs
  0 ms až 65535 ms

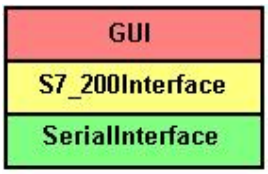
Ako už bolo spomenuté, ak nastavíme PWM generátor na niektorom z výstupov Q0.0 resp. Q0.1 nie je možné meniť stav tohto výstupu bežným zápisom do registra. Ak by sme potrebovali výstup, ktorý ovláda PWM generátor nastaviť na log. 1 nastavíme periódu menšiu alebo rovnú ako je šírka pulzu. Pre nastavenie výstupu do log. 0 nastavíme šírku pulzu na 0 µs.

Ak sa nastaví IT modul spôsobom spomenutým vyššie je potrebné na začiatku každého programového cyklu volať podprogram ETHx_CTRL (znak x sa nahrádza číslom slotu v ktorom je zasunutý IT modul). Návratové hodnoty tohto podprogramu obsahujú stav IT modulu, stav ôsmich možných komunikačných kanálov a stav IT služieb.

Ak nastane chyba v IT module (CP 243-1 IT) , vieme túto chybu vyčítať z návratovej hodnoty podprogramu. Kód chyby je prístupný maximálne po 60 sec. Po uplynutí tejto doby IT modul vymaže danú chybu zo svojej pamäti a návratová hodnota chyby bude rovná nule. Týmto algoritmom sa zamedzí tomu, aby PLC systém bol informovaný chybami, ktoré už nie sú platné.

7. Program v PC

Program v PC je rozdelený do 3. vrstiev. Prvá a druhá vrstva je napísaná v OOP C++ a tretia je napísaná vo Visual C++ a využíva MFC knižnice.



Obr. 6. Architektúra programu

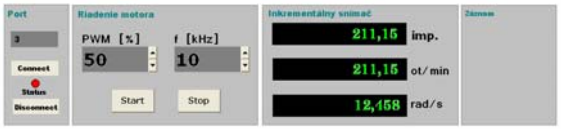
Prvá vrstva sa volá SerialInterface. Obsahuje všetky metódy na nastavenie, spravovanie a prijímanie resp. odosielanie údajov. Pri vytváraní objektu SerialInterface sa musí zadať číslo portu na ktorom bude prebiehať komunikácia. Automaticky sa nastavia aj timeouty pre port. Rýchlosť komunikácie sa nastavuje metódou setDCB(). Pokiaľ by bolo potrebné je možné port

kedykoľvek zavrieť a znovu otvoriť. Používajú sa na to dve metódy openPort(int PortNumber) a closePort(). Ak je potrebné odoslať alebo prijať údaje z portu použije sa na to metóda send(const char* data, unsigned int count) a recive(char* data, bool clearBuffer). Pri metóde send sa nastavuje pointer na údaje ktoré majú byť odoslané a ich počet v bytoch. Pri metóde recive sa nastavuje pointer na dátovú oblasť kde majú byť údaje uložené a parameter clearBuffer sa používa ak napríklad chceme pred príjmom údajov vyčistiť prijímací zásobník. Vtom prípade sa nastaví na hodnotu true. Ak by sa niekto pokúsil odoslať alebo prijať údaje počas toho ako je port uzavretý nastaví sa vnútorná premenná objektu objectStatus na určitú hodnotu. Hodnota 0 znamená všetko je v poriadku. Ak sa počet odoslaných údajov nerovná počtu údajov na odoslanie má hodnotu 2. Hodnota 6 znamená pokus o odoslanie resp. príjem údajov na uzavretom porte.

Druhá vrstva sa volá S7_200Interface. V tejto vrstve je naprogramovaný celý protokol. Objekt tejto vrstvy sa vytvára v 3 vrstve GUI. Pomocou tejto vrstvy odpadá znalosť komunikácie po linke RS-232 pretože všetko spracúva už objekt SerialInterface, ktorý využíva táto vrstva. Ako príklad pokiaľ sa zavolá metóda connectS7_200(int port) tak v jej tele sa nachádza jeden riadok a to volanie metódy openPort(int PortNumber) objektu SerialInterface, Podobne je to aj s metódou disconnectS7_200(), ktorá zavolá metódu objektu SerialInterface closePort(). Ako bolo spomenuté vyššie príkazy pred odoslaním sa radia do fronty príkazov. Na priradenie príkazu slúži metóda s premenlivým počtom parametrov addCommand(Commands cmd,...). Premenlivý počet parametrov je potrebný, pretože nie každý príkaz má parametre a ak má tak nie vždy rovnaký počet. Na to aby sa dodržal protokol z pevnou dĺžkou paketu je vždy nevyžité miesto v pakete doplnené o potrebný počet núl. Nuly, ale ani žiadne iné číslo doplnené na nevyužitom mieste paketu, nezmenia význam príkazu. Pretože program v PLC vie ktoré miesta v pakete má použiť pre prijatý príkaz. Parameter cmd v metóde addCommand je pole enum kde sú vymenované všetky možné príkazy. Takto odpadá nutnosť pamätanie sa čísiel príkazov. Po zaradení príkazov do fronty je potrebné príkazy aj odoslať. Na toto slúži metóda update(). Metóda update() sa volá každých 20 ms a zakaždým sa odošle jeden paket a ak prišiel tak sa prijme paket. Pakety určené na odoslanie a pakety ktoré boli prijaté sú ukladané do STL kontajnera vector. Použitím STL kontajnerov odpadá nutnosť písania zložitých algoritmov.

8. Záver

Naším nasledujúcim cieľom je dokončiť a implementovať všetok navrhnutý hardware a software. Vysledný program by mal vyzerať tak ako je to znázornené na obr. 7.



Obr. 7. Tvar výsledného programu

Takýto istý vzhľad by mal mať aj Java Applet pomocou ktorého sa bude ovládať motor cez internet. Ďalším cieľom je, aby bolo možné v programe nastaviť tvar krivky rýchlosť motora v závislosti na čase a následne podľa nej regulovať motor.

9. Použitá literatúra

[1] SIEMENS, Programovatelný automat S7-200 Systémový manual
[2] SIEMENS, SIMATIC NET S7Beans/Applets for IT-CPs Programming Tips
[3] SIEMENS, SIMATIC NET CP-243-1 IT Communications Processor