ROBOTIKA AKO HOBBY
STAVIAME A PROGRAMUJEME

V minulom čísle sme prenikli do sveta amatérskych robotov a bližšie rozobrali anatómiu robota. Na to, aby sme sami postavili robota, potrebujeme pochopiť, ako funguje mozog robota – procesor. Tento článok bude návodom ako zostrojiť základnú elektronickú schému robota a naprogramovať procesor na vykonávanie jednoduchých úloh. Odtiaľ je už len skok k dotváraniu podoby robota.

Staviame hardvér

Našim prvým cieľom je osadiť procesor do dosky, pripojiť ho na vhodný zdroj napätia a pripojiť mu na výstup svetelnú LED diódu, ktorú bude riadiť. Potom urobíme jednoduchý program, aby dióda blikala podľa pokynov. Vstup procesora si vysvetlíme pripojením spínača a opäť ukážkovým programom reagujúcim na zapnutie spínača.

Elektrický zdroj

Ako elektrický zdroj pre naše experimenty poslúži akýkoľvek adaptér dodávajúci jednosmerné napätie aspoň 6 V, môže byť až 12 V.

Neskôr ho môžeme nahradiť napríklad šiestimi ceruzkovými batériami typu AA. Ak použijeme nabíjacie po 1,2 V, dostaneme celkové napätie 7,2 V. Toto napätie však nesmieme použiť priamo. Procesor je najjemnejšia elektronická súčiastka skladajúca sa z miliónov tranzistorov na jednom centimetri štvorcovom vykonávajúcich milióny aritmetických operácií za sekundu. Už to napovedá, že sa k nemu musíme správať nanajvýš opatrne. Šum v obvode či prudké elektrické výkyvy môžu spôsobiť jeho nesprávne fungovanie alebo zničenie. Najprv potrebujeme vyriešiť, aby sme procesoru trvale dodávali stabilné a čisté napätie päť voltov. Riešením je *stabilizátor* a *kondenzátor*.

Stabilizátor s chladičom (vľavo) a kondenzátor (vpravo)

Stabilizátor s katalógovým označením 7805 pretvára akékoľvek vyššie napätie na päť voltov a kondenzátor vyhladí jemné výkyvy a odstráni šum.

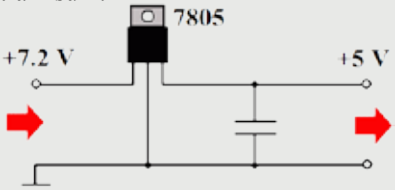


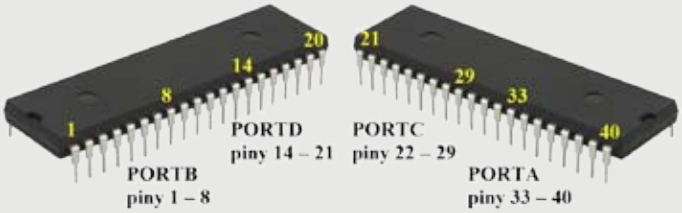
Schéma zapojenia stabilného zdroja. Dohoda: záporný pól označujeme ako zem ⊥.

Kombináciou stabilizátora a kondenzátora s kapacitou rádovo 100 µF dostaneme stabilný zdroj. Ak vyhladenie nestačí (čo môžeme pozorovať ako samovoľné resetnutie procesora – ochrana proti zničeniu), použijeme kondenzátor s väčšou kapacitou. Po pripojení motorov robota vytvárajúcich silný šum v celom obvode sa odporúča kapacita 1 000 µF až 10 000 µF.

Frekvencia procesora

Typická frekvencia procesora dnešných počítačov sa udáva v GHz alebo v MHz. Pre náš procesor toto číslo znamená, koľko základných operácií dokáže vykonať za jednu sekundu. Základnou operáciou je napríklad priradenie čísla do premennej. Náš procesor môže fungovať na rôznych frekvenciách, ktoré určuje *piezoelektrický kryštál*. Táto súčiastka tvorí akési hodiny procesora. Je to špeciálna kryštalická látka, ktorá pod vplyvom elektrického napätia veľmi rýchlo kmitá s presnou frekvenciou závislou od atómovej štruktúry. Na naše účely použijeme kryštál s frekvenciou 16 MHz. Podľa technickej špecifikácie procesora budeme potrebovať aj dva kondenzátory s kapacitou 27 pF.

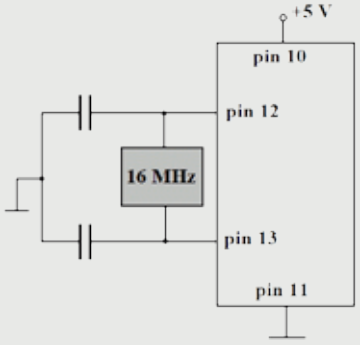
Piny procesora



Procesor typu Atmega32 má spolu 40 pinov, 32 z nich môžeme použiť ako vstup alebo výstup. Ich číslovanie je dohodnuté 1 až 40 v protismere hodinových ručičiek počnúc ľavým horným rohom. Vrch procesora identifikujeme štrbinou. Vstupno-výstupné piny sú kvôli prehľadnosti rozdelené do štyroch skupín po osem pinov a tieto skupiny dostali pomenovania PORTA, PORTB, PORTC a PORTD. Okrem nich využijeme piny 10 a 11 pre zdroj napätia a piny 12 a 13 pre kryštál.

Schéma zapojenia

Načrtneme si schému zapojenia. Na piny 12 a 13 pripojíme kryštál. K nemu pripojíme dva malé kondenzátory po 27 pF. Na piny 10 a 11 pripojíme kladný a záporný pól už stabilného zdroja.

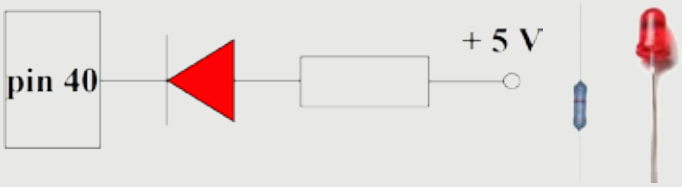


Praktická realizácia zapojenia

Niektorí konštruktéri si dosky plošných spojov zhotovujú tradične: ručne kreslia fixkou na pomedenú dosku, leptajú kyselinou, vŕtajú a spájkujú. Skúsenejší používajú na zhotovenie plošného spoja UV lampu a fotocitlivú dosku, na ktorú premietnu obrazec. Iní používajú univerzálny plošný spoj, v ktorom stačí dotvoriť požadovaný obvod spájkovaním medených mostíkov, odporúča sa začiatočníkom. Na experimentálne účely postačí kontaktné pole na obrázku v dolnej časti stránky. Tento variant je obľúbený aj u hotových robotov. Výhodou je bezprácnosť a flexibilnosť (obvod sa dá ľahko pretvárať), nevýhodou môže byť veľkosť a menšia spoľahlivosť (možnosť vypadnutia súčiastky).

Kontaktné pole je doska s dierkami navrhnutá tak, aby sme vedeli postaviť ľubovoľnú schému. Do dierok môžeme zasúvať súčiastky aj drôtiky. Každých päť susedných dierok vodorovne označených ABCDE je vodivo spojených dohromady. Rovnako je to s dierkami FGHIJ. Dva stredné zvislé stĺpce dierok označené dlhou červenou a modrou čiarou tvoria dva vodiče a kvôli prehľadnosti sa zvyknú používať ako kladný a záporný pól zdroja. Nám budú reprezentovať +5 V a zem.

Aby sme mohli sledovať činnosť procesora, na výstupný pin pripojíme svetelnú LED diódu a budeme ju riadiť programom.

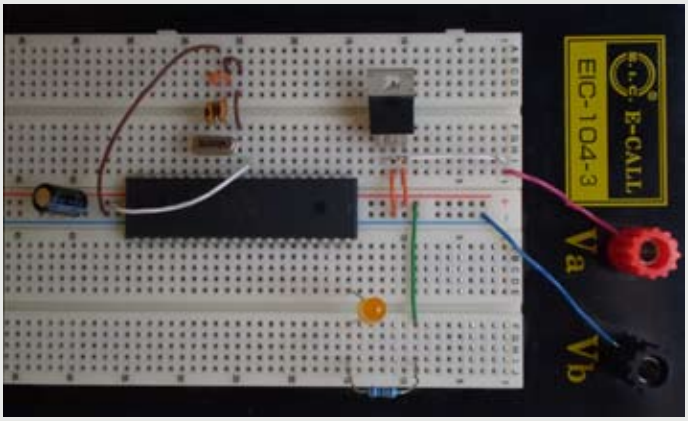


Dôležité je nezabudnúť na rezistor. Ak by sme ho nepoužili, diódou by pretekal maximálny prúd, zhorela by, a čo je horšie, odpálili by sme hneď aj procesor. Optimálny prúd pre LED diódu a procesor (ako aj všetky bežné polovodičové súčiastky) je približne 20 mA. Túto požiadavku splníme, ak pred diódu vložíme rezistor s odporom vypočítaným z Ohmovho zákona (používame napätie 5 V):

$$R = \frac{U}{I} = \frac{5\ V}{20\ mA} = 250\ \Omega$$

Mali by sme použiť rezistor s odporom aspoň 250 Ω.

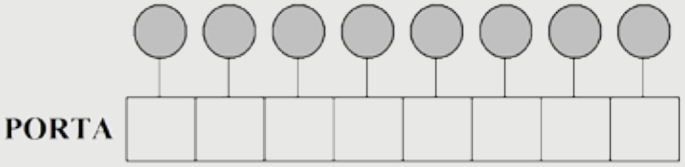
Hotová schéma zapojenia procesora spolu so stabilizátorom a výstupnou LED diódou môže vyzerať takto:



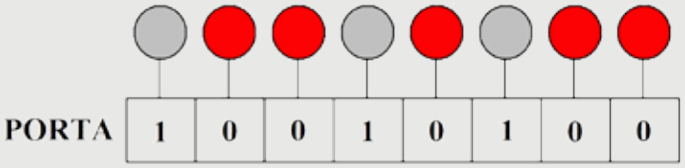
Hardvér je cenovo nenáročný. Maloobchodná cena procesora sa pohybuje okolo 150 korún, cena kontaktného poľa 100 až 300 korún (podľa veľkosti) a cena zvyšných súčiastok by nemala prekročiť 50 korún.

Programujeme softvér

Schému už máme zapojenú, nič však nerobí. Dióda nesvieti, procesor je mŕtvy. Potrebujeme mu vdýchnuť život. Naším cieľom je urobiť program, ktorý bude zapínať a vypínať požadované piny, čo budeme pozorovať ako rozsvecovanie a zhasínanie diódy. Podľa spomenutej dohody má zem binárnu hodnotu 0. Ak výstupnému pinu priradíme hodnotu 0, dióda by sa mala rozsvietiť, pretože jedna jej noha je trvale pripojená na +5 V a druhá noha sa uzemní. Naopak, ak mu priradíme hodnotu 1, dióda zhasne, lebo druhá noha už nebude uzemnená a diódou nebude prechádzať prúd. Podobne ako lastovička spokojne sediaca na elektrickom kábli s vysokým napätím. Nič sa jej nestane, lebo druhou nohou nedočiahne na zem ani na druhý kábel. Aby sme si objasnili, ako funguje riadenie výstupných pinov v jednej skupine, napríklad PORTA, pripojme na každý pin jednu diódu. Spolu ich budeme potrebovať osem.



Skupina pinov PORTA je v programe reprezentovaná premennou s názvom PORTA a môžeme jej priraďovať rôzne číselné hodnoty. Priraďme jej napríklad hodnotu 148. Číslo 148 vyzerá v binárnej sústave ako 10010100. Takto dostaneme hodnoty na jednotlivých pinoch a hneď vieme, ktoré diódy sa rozsvietia – tie, kde sú nuly.



Ak by sme chceli rozsvietiť všetky diódy, napíšeme `PORTA = 0`, lebo v binárnej sústave je to 00000000. Ak by sme ich chceli všetky vypnúť, napíšeme `PORTA = 255`, lebo v binárnom zápise je to 11111111. Riadenie výstupov procesora je teda hranie sa s binárnou sústavou.

Samotný program je napísaný v jazyku C a môžeme ho napísať v akomkoľvek textovom editore. Postupne si vysvetlíme význam jednotlivých riadkov.

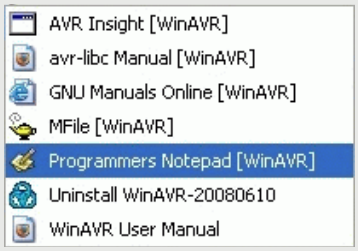
```
#include <avr/io.h>                ❶

int main(void)                     ❷
{
  DDRA = 1;                        ❸
  while (1)                        ❹
  {
    PORTA = 0;                     ❺
    PORTA = 1;                     ❻
  }
}
```

❶ Aby sme mohli pracovať s pinmi procesora, musíme na začiatku uviesť hlavičku (súbor) `io.h`, ktorá obsahuje inštrukcie pre náš procesor. Každá hlavička je súčasťou veľkej knižnice `avr` a vždy ju uvedieme ako cestu k súboru.

❷ Každý program v C musí obsahovať hlavnú funkciu main, jadro programu.

❸ Momentálne používame prvý pin zo skupiny PORTA ako výstup na LED diódu. Daný pin môžeme použiť ako vstup aj ako výstup. To však musíme procesoru vopred povedať. Ak pripájame senzor alebo klávesnicu, ide o vstup. Ak potrebujeme niečo riadiť, napríklad motory alebo svetlá, ide o výstup. V našom prípade riadime LED diódu, takže máme výstup. Ktoré piny skupiny PORTA majú byť vstupy a ktoré výstupy, prikážeme pomocou premennej DDRA, ktorá funguje podobne ako PORTA až na to, že nula znamená vstup a jednotka výstup. Pre PORTB, PORTC a PORTD použijeme analogické premenné DDRB, DDRC a DDRD.
❹ Program bude bežať v nekonečnom cykle. Chceme, aby dióda blikala bez prestania, až kým nevypneme procesor. V nekonečnom cykle striedame ❺ a ❻.
❺ Dióda sa rozsvieti.
❻ Dióda zhasne.

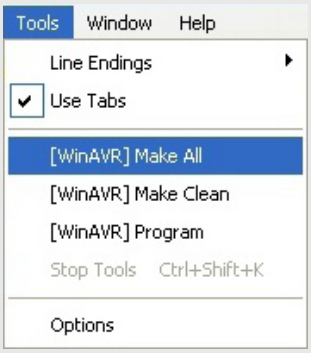
Máme hotový program, čo teraz?



Program v jazyku C je síce zrozumiteľný pre nás, ale procesor ho potrebuje mať vo svojej reči. Reči procesora hovoríme *strojový kód*. Ten je zase ťažko čitateľný pre nás. Potrebujeme nástroj, ktorým program *skompilujeme* (preložíme) do strojového kódu. Ako vhodné prostredie nám poslúži voľne dostupný softvérový balík WinAVR. Na stránke

http://www.mladyvedec.sk/download/05/robotika2.zip

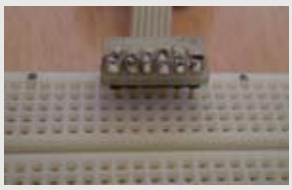
si môžete stiahnuť balík všetkých súborov, ktoré budeme potrebovať. Po inštalácii tohto programu môžeme nájsť textový editor *Programmers Notepad*, v ktorom napíšeme program a uložíme pod názvom *main.c*.



Program skompilujeme príkazom `Tools -> Make All`. Ak preklad prebehol správne, editor to oznámi a výsledok môžeme hneď napáliť do procesora príkazom `Tools -> Program`. Proces trvá niekoľko sekúnd. Ako bolo povedané v predchádzajúcom článku, procesor má vnútornú FLASH pamäť a je možné ju prepísať až desaťtisíckrát. Predtým ako dostaneme prvý program do procesora, čaká nás niekoľko úloh.

Zatiaľ sme nepovedali, s akým typom procesora pracujeme, aká je jeho pamäť, aké inštrukcie pozná atď. To všetko by mal kompilátor vedieť pre správny preklad a napálenie. Na to je potrebné vytvoriť špeciálny konfiguračný súbor v tom istom priečinku, kde máme uložený náš program *main.c*. Už hotový konfiguračný súbor *Makefile* sa nachádza v našom balíku. Keď budete robiť akýkoľvek program, nazvite ho *main.c* a skopírujte k nemu súbor *Makefile*. Upozorňujeme, že je nastavený len pre procesory Atmega32, ale môžete si ho prepísať napríklad pre typ Atmega16.

Ďalším zádrheľom môže byť spojenie procesora s počítačom. Namiesto kupovania *programátora* môžeme použiť paralelný kábel. Je potrebné použiť redukciu, aby sme ho mohli zapojiť do kontaktného poľa, kde máme procesor s naším obvodom. Redukcia je známa pod označením STK200 a dá sa buď kúpiť, alebo vyrobiť podľa schémy *stk200.jpg* v balíku. Vyrobená redukcia môže vyzerať takto:



Nepotešujúce pre amatérov je, že novšie notebooky vynechávajú paralelný a sériový port a nahrádza ich modernejší USB port. Tomu sa postupne prispôsobuje aj vývoj nových programovateľných procesorov.

Nastavenie nového procesora

Procesor má namiesto kryštálu svoje vnútorné hodiny, avšak pomalé a nie veľmi presné, preto sme namiesto nich radšej použili vonkajší kryštál. Keď kúpime nový procesor, musíme ho hneď na začiatku nastaviť na naše použitie. Bez úvodného nastavenia nemusí vôbec fungovať. Aby ste sa s tým nemuseli trápiť, v konfiguračnom súbore *Makefile* je na to vytvorený príkaz. V adresári, v ktorom sa nachádza súbor *Makefile*, stačí spustiť príkaz `make nastavcpu`:

```
C:\robotika\pokus>make nastavcpu

Atmel AVR ATmega32 is found.
Fuse Low Byte set to 0xbf
Fuse High Byte set to 0xc1
```

Tento proces stačí vykonať len raz pre nový procesor. Procesor si nastavenie bude navždy pamätať.

Späť k programu

Hneď ako sa podarí napáliť program do procesora, program beží. To je výhoda, že pri testoch nie je potrebné vyberať procesor či vyťahovať kábel. Program môžeme kedykoľvek prepísať a opäť skompilovať a napáliť (`Tools -> Make All; Program`). Budeme sklamaní, keď uvidíme, že dióda nebliká, ako sme chceli, ale svieti. Stroj zvykne robiť presne to, čo mu prikážeme, a nie to, čo od neho chceme. Možno ste postrehli, že v programe niečo chýba. Uvedomme si, že riadky ❺ a ❻ sa opakujú v cykle tak rýchlo, ako to len procesor stíha. Dióda teda bliká (alebo snaží sa blikať) extrémne rýchlo, približne frekvenciou procesora, rádovo miliónkrát za sekundu. To je príliš veľa na to, aby sme to spozorovali. Medzi rozsvietením a zhasnutím potrebujeme nejaký čas počkať. Program opravíme takto:

```
#define F_CPU 16000000UL                    ❼
#include <avr/delay.h>                      ❽
#include <avr/io.h>

int main(void)
{
  DDRA = 1;
  while (1)
  {
    PORTA = 0;
    _delay_ms(10);                          ❾
    PORTA = 1;
    _delay_ms(10);                          ❿
  }
}
```

❼ Zadefinujeme frekvenciu procesora 16 000 000 Hz (16 MHz).
❽ Vložíme knižnicu `delay.h`, ktorá obsahuje funkciu na ovládanie hodín procesora.
❾ Funkcia `_delay_ms` počká daný počet milisekúnd. Dióda sa rozsvieti na 10 ms.
❿ Dióda zhasne na 10 ms.

Dióda teraz blikne každých 20 ms, t. j. 50-krát za sekundu, čo môžeme pozorovať ako chvenie. Ak ju chceme spomaliť, musíme príkaz nahradiť takto:

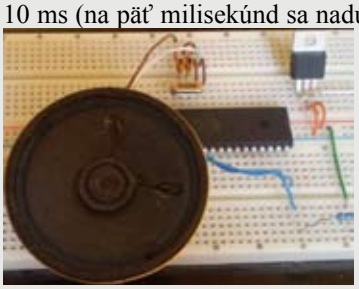
```
for (i = 0; i < 100; i++) _delay_ms(10);
```

Podľa manuálu balíka WinAVR totiž parameter funkcie

`_delay_ms`

nesmie nadobúdať väčšiu hodnotu ako 16 pri frekvencii procesora 16 MHz.

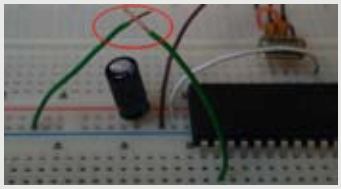
Miesto svetelnej diódy môžeme pripojiť reproduktor. Počuteľný zvuk leží približne v rozsahu 20 Hz až 20 kHz. Ak počkáme napríklad 5 ms, membrána reproduktora urobí kmit každých 10 ms (na päť milisekúnd sa naduje a na päť sa vráti do pôvodnej polohy). To je 100-krát za sekundu, a teda 100 Hz. Reproduktor by mal pískať. Ak chceme zvýšiť tón, môžeme pracovať aj s mikrosekundami. Potrebné funkcie nájdeme v manuáli k balíku WinAVR.



Senzory a spínače

Zatiaľ sme sa naučili, ako riadiť výstupné časti robota. Robot však potrebuje reagovať na podnety a vnímať okolie pomocou senzorov. Ak chceme informovať procesor o vstupe, stačí uzemniť daný pin. Prakticky to znamená, že najprv požadovanému vstupnému pinu priradíme binárnu hodnotu 1. Potom ho fyzicky uzemníme a jeho hodnota sa zmení na 0. Keď ho odpojíme od zeme, získa späť hodnotu 1. Program bude opakovane čítať hodnotu na danom pine a keď zaregistruje zmenu, vykoná, čo mu povieme. Vstup si demonštrujeme na najjednoduchšom type senzora – dotykovom senzore, nárazníku alebo spínači. Pripojme spínač na prvý pin skupiny PORTC.

Spínač môžeme realizovať jednoducho dvoma vodičmi s voľnými koncami. Jeden vodič zapichneme k pinu 22 a druhý vodič k zápornému pólu zdroja. Keď sa konce dotknú, pin sa uzemní.



Zatiaľ sme sa stretli s tým, že do premenných PORTA, PORTB, PORTC a PORTD zapisujeme. Na čítanie však používame premenné PINA, PINB, PINC a PIND.

Naprogramujeme procesor tak, aby dióda svietila len v prípade, že je spínač vypnutý. To je netriviálne správanie, ktoré by sme nevedeli docieliť bez tranzistorovej elektroniky. Očakávali by sme, že keď niečo rozpojíme, voľačo zhasne, ale my to chceme spraviť naopak.

```
#include <avr/io.h>

int main(void)
{
  DDRA = 1; DDRC = 0;                          ❶
  PORTC = 1;                                   ❷
  while (1)
  {
    if (bit_is_set(PINC,0)) PORTA = 0;         ❸
    else PORTA = 1;                            ❹
  }
}
```

❶ Ošetríme vstup a výstup. Prvý pin zo skupiny PORTA používame ako výstup na LED diódu a prvý pin zo skupiny PORTC ako vstup zo spínača.
❷ Vstupnému pinu priradíme hodnotu 1.
❸ V nekonečnom cykle sa pýtame, aká je hodnota vstupného pinu. Vieme, že musíme prečítať premennú PINC. Tá obsahuje hodnoty všetkých ôsmich pinov v jednom veľkom čísle tak, ako sme si ukázali na príklade s ôsmimi LED diódami. Ako vstup sme použili prvý pin a chceme poznať jeho hodnotu. V binárnej sústave je to najpravejší, čiže *nultý* bit. Prečítame teda nultý bit premennej PINC. Funkcia

`bit_is_set(PINC,0)`

prezradí, či je tento bit nastavený, t. j. či má hodnotu jedna. Ak áno (spínač je vypnutý), premennej PORTA priradíme nulu (dióda sa rozsvieti).
❹ V opačnom prípade (spínač je zapnutý) premennej PORTA priradíme jednotku (dióda zhasne).

Vymenením poradia príkazov `PORTA = 0` a `PORTA = 1` dosiahneme, že program bude robiť opak – dióda bude svietiť práve vtedy, keď bude spínač zapnutý.

Manuál

Keď už sme sa naučili vyrobiť krátky program, zvyšok je o experimentovaní. Manuál funkcií a knižníc nájdeme v nainštalovanom balíku.

V skutočnosti už nepotrebujeme poznať žiadne špeciálne funkcie na to, aby sme dokázali naprogramovať robota. Všetko je o vstupoch, výstupoch a správnom načasovaní. Na ich spracovanie je potrebné dobre ovládať jazyk C.

Nabudúce

V budúcom čísle sa môžete tešiť na robota sledujúceho čiaru – stopára. Ukážeme si, ako rozlíšiť tmavú čiaru na svetlom podklade a ako pomocou informácie o polohe čiary riadiť motory tak, aby robot čiaru neopustil.

Andrej Osuský