Kratka šola programiranja mikrokontrolerjev

Jure MIKELN

1 Uvod

Smo priče silnemu napredku na vseh področjih tehničnih znanosti. Za ta napredek se morajo novodobne tehnične znanosti zahvaliti predvsem razvoju procesorske moči osebnih računalnikov, kar je omogočilo razvoj novih zmogljivih programskih orodij. Elektroniki poznamo veliko programskih orodij, brez katerih si sodobnega dela ne znamo več predstavljati. Podobno velja tudi za moderne elektronske komponente, izmed katerih bi poudaril mikrokontrolerje, ki so v zadnjih 15 letih tudi naredili ogromen korak naprej. Ta napredek se vidi na cenovnem, predvsem pa na tehničnem področju. Tako danes za nekaj evrov dobimo izredno zmogljiv ARM-mikrokontroler (32-bitni mikrokontroler), ki ima izredno zmogljivo periferijo, kot so analogno/digitalni in digitalno/analogni (A/D in D/A) pretvorniki, ethernet, CAN ter USB-vmesniki, krmilniki za motorje, moduli za realno uro in podobno. Pri tem pa ne zamešajmo izrazov: v PC-ju ne deluje mikrokontroler, pač pa zmogljiv procesor (64-bitni), ki signale po vodilu pošilja na razne enote, kot so video, spominska, krmilnik trdih diskov itd. Razlika med procesorjem in mikrokontrolerjem je v tem, da procesor nima vgrajene periferije.

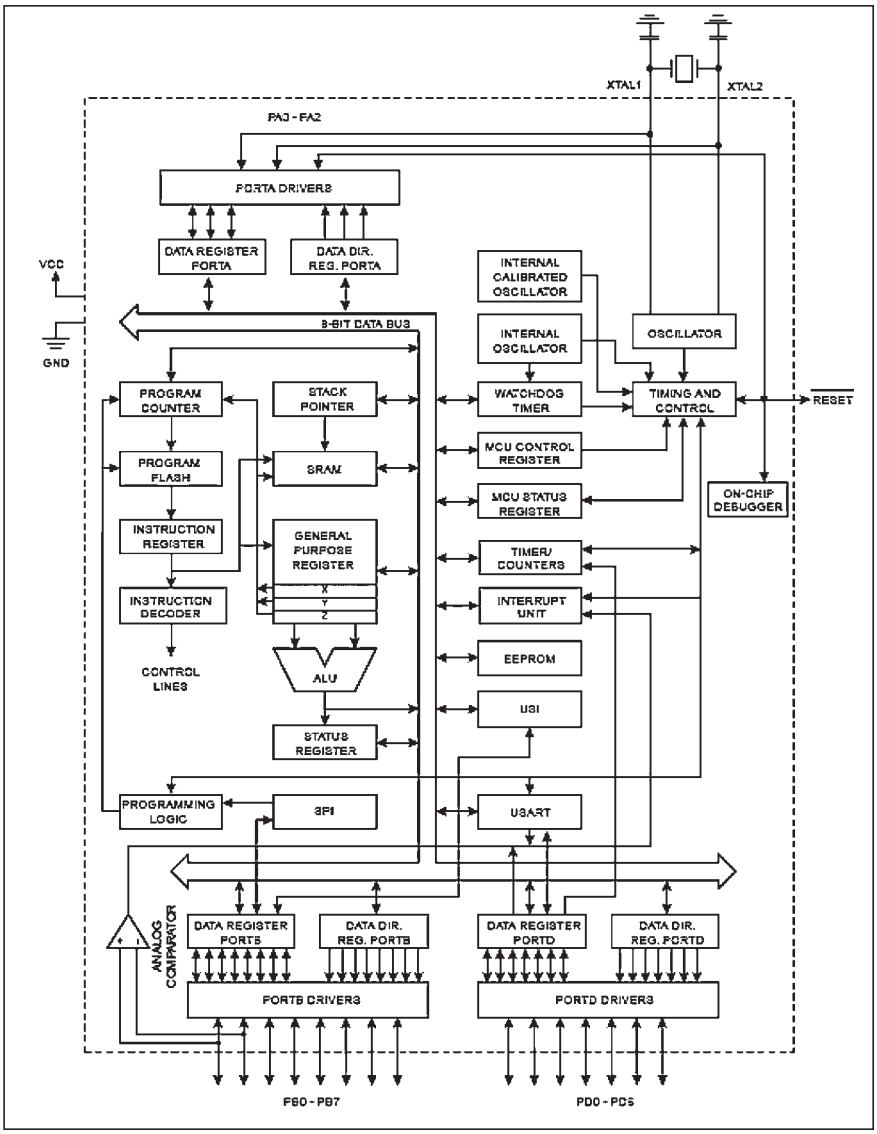
V seriji člankov, ki je pred vami, se ne bomo spoznavali z ARM-mikrokontrolerji, pač pa smo pripravili osnovne informacije, ki naj bi jih poznal strojnik, da bi lahko suvereno začel programirati mikrokontrolerje družine AVR. Takšen mikrokontroler bo brez težav poganjal motorje, vklapljal ter izklapljal releje, sprejemal signale z različnih senzorjev in tipk, signale obdeloval ter s pomočjo LCD-prikazovalnika prikazoval podatke. Serija člankov bo imela več delov, kjer bomo prikazali osnove programiranja mikrokontrolerjev. V zadnjem delu boste suvereno znali sprogramirati mikrokontroler in sami nadgrajevati osvojeno znanje.

Jure Mikeln, dipl. inž.,
AX Elektronika, d. o. o., Ljubljana

2 Mikrokontrolerji, procesorji in krmilniki

Poskusimo najprej definirati razliko med mikrokontrolerjem, procesorjem in krmilnikom. Izraz krmilnik uporabljamo takrat, kadar govorimo o industrijskih PLC-krmilnikih, kot so npr. Simatic in podobno. Krmilnik je torej naprava v svojem ohišju s priključnimi konektorji. Velikokrat je na ohišju tudi nekaj tipk in prikazovalnik. Procesor je izraz za zmogljiv čip, ki se ponavadi nahaja v naših PC-jih in ne vsebuje periferije, kot so A/D- ali D/A-pretvorniki, spomin in podobno. Procesor podatke prejema in jih pošilja preko vodila na ostale enote. Mikrokontroler pa je čip, ki vsebuje šibek procesor in že ima vgrajeno periferijo ter spomin.

AVR-mikrokontrolerji so eni izmed popularnih mikrokontrolerjev, ki so dobavljivi že približno 10 let. Ker bom prispevek skušal napisati kar se da razumljivo, naj predstavim mikrokontroler kot "črno škatlo", ki ima vhodne in izhodne priključke. Mikrokontroler brez programa je mrtev in kot tak res neuporaben. Zaživi le, ko se v njem vpisan program prične izvajati. Jasno pa je, da je za to potrebna tudi napajalna napetost. Mikrokontrolerje dandanes uporabljamo v praktično vsaki napravi, bodisi da je to igrača ali zahteven industrijski PLC-krmilnik. V vseh teh napravah je vsaj en mikrokontroler, vemo pa, da v sodobnih avtomobilih deluje več mikrokontrolerjev. Obstaja več vrst mikrokontrolerjev, od 1-, 4-, 8- pa tja do 32-bitnih. Danes se največ uporabljajo 8- in 32-bitni, redko pa še kakšni drugi. Mikrokontrolerji, o katerih bo govor v seriji člankov, so 8-bitni. AVR mikrokontroler vsebuje ALU-enoto (arithmetical logical unit), več vhodno/izhodnih (V/I, v tuji literaturi je uveljavljen izraz I/O – input/output port) vrat ali portov, spomin FLASH in še več enot, o katerih bo govor kasneje. V spomin FLASH s pomočjo programatorja naložimo naš program, ki ga v zaporednih korakih izvaja ALU. V/I porti so večinoma dvosmerni, pri čemer moramo s programom definirati, ali bo posamezen pin vhodni ali izhodni. Mikrokontroler za svoje delovanje potrebuje napajalno napetost in večinoma tudi kvarčni kristal oziroma keramični resonator, ki daje ALU-ju t. i. taktni signal, s katerim ALU izvaja programsko kodo. Če ALU nima takta, se njeno delovanje ustavi in se spet nadaljuje, ko obnovimo taktni signal. V določenih primerih pa tudi kvarčni kristal ali resonator nista potrebna, saj imajo AVR-mikrokontrolerji vgrajen t. i. RC-oscilator, ki daje takt ALU-ju in s tem omogoča delovanje mikrokontrolerja. Izbor med RC-oscilatorjem ali zunanjim kvarčnim kristalom oz. resonatorjem definiramo med postopkom programiranja.



Slika 1. *Notranja blok shema mikrokontrolerja ATTiny2313*

Kot vidite, je v notranjosti mikrokontrolerja veliko blokov, o katerih ne bomo izgubljali besed. Zaenkrat naj bo mikrokontroler za nas črna škatla z vhodi in izhodi, ki jo bomo sprogramirali z našim programom.

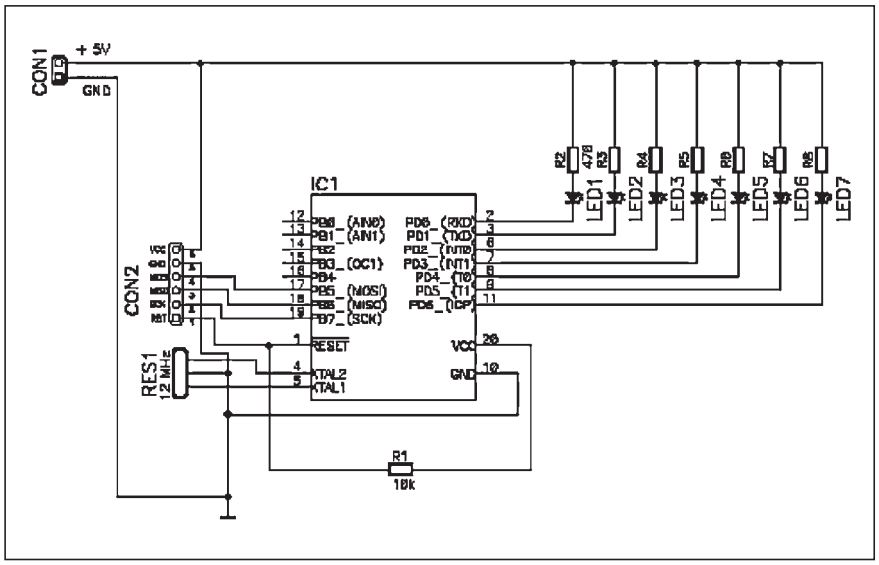
Program v mikrokontrolerju poteka glede na ukaze t. i. strojne kode, ki ALU-ju določa, kaj bo naredila. Ker deluje ALU z binarno matematiko, je ukaz v strojni kodi sestavljen iz ničel in enic – pač v binarni kodi. Na začetkih mikrokontrolerjev so programerji za programiranje uporabljali strojno kodo, vendar so kmalu ugotovili, da je to izredno nepraktično. Zato so pričeli uporabljati zbirnik (v tuji literaturi se uporablja izraz assembler), dandanes pa vedno bolj posegamo po višjih programerskih jezikih, kot so C, različne vrste „visual basic" programov ter basic programov, ki programsko kodo, napisano v basicu, prevedejo v strojno kodo. Med basic programe spada tudi programski jezik bascom, ki se je v Evropi, pa tudi v Sloveniji lepo uveljavil. Zavedam se, da ima bascom (kot tudi drugi višji programski jeziki) svoje prednosti in slabosti, vendar trdno stojim za stališčem, da je bascom idealno programsko orodje za začetnike v programiranju, kar bom pokazal v nadaljevanju.

3 Predstavitev razvojnega okolja Bascom-AVR

Bascom-AVR obstaja v dveh verzijah: Bascom-AVR in Bascom-8051. Končnica nam nakazuje, za katero vrsto mikrokontrolerjev je namenjen prevajalnik. Mi bomo seveda uporabljali Bascom-AVR, ki si ga lahko brezplačno snamete s spletne strani *www.mcselec.com*. Z navedene spletne strani si snemite DEMO verzijo, ki je popolnoma delujoča verzija z omejitvijo velikosti programske kode, ki jo lahko prevedemo. Kot programerjem začetnikom nam bo DEMO verzija popolnoma zadostovala, saj bodo naši programi kratki in ne bodo dosegali omejitve DEMO verzije.

Bascom je identičen večini programov, ki tečejo v MS Oknih (Windows), zato tukaj ne bi izgubljali besed. Ker je Bascom-AVR programsko orodje, je prav, da se spoznamo z vrstami števil – spremenljivk, ki jih lahko uporabljamo v naših programih. Na voljo so nam spremenljivke Bit (1 bit), Byte (8 bitov = 1 byte), Integer (2 byte, vrednost spremenljivke je lahko od –32,768 do +32,767), Word (2 byte, vrednost spremenljivke je lahko od 0 do 65535), Long (4 byte, vrednost spremenljivke je lahko od –2147483648 do 2147483647, Single (32 bitov, vrednost spremenljivke je lahko od 1.5 x 10^–45 do 3.4 x 10^38), Double (64 bitov, vrednost spremenljivke je lahko od 5.0 x 10^–324 do 1.7 x 10^308) in String (do 264 byte, spremenljivke so tipa byte z dodatnim bytom vrednosti nič na koncu).

Spremenljivke imajo lahko poljubna imena, ki jih ponavadi izberemo tako, da so med programiranjem prepoznavna. Načeloma lahko spremenljivke poimenujmo X1, X2, X3 itd., vendar bomo med programiranjem program lažje razumeli, če bomo spremenljivko, ki ponazarja vklop/izklop, npr. releja, poimenovali Rele. Vsekakor pa ne smemo uporabljati t. i. rezerviranih imen. Teh je preveč za ta prispevek, zato si jih le oglejte v Helpu. Nekaj teh imen pa lahko naštejem za boljši občutek: OFF, ON, OR, OUT, OUTPUT, LCASE, LCD, LCDAT, LEFT, LEFT, LEN, LINE in podobno.



Slika 2. *Shema vezave mikrokontrolerja*

Preden bomo začeli s programiranjem, se podrobno spoznajmo še z izrazi bit in byte. Oba izraza smo srečali že v osnovni šoli pri predmetu matematika, kjer smo spoznavali binarni številski sistem, ki ima samo dve števili: 0 in 1. Spremenljivka vrste bit ima tudi dve vrednosti 0 ali 1. Medtem ko ima število vrste byte vrednosti med 0 do 255 (binarno: 00000000 do 11111111).

Zdaj, ko smo spoznali nekaj osnov, pa že lahko pričnemo s programiranjem. Za začetek si zapomnimo enega najpogostejših ukazov: Do Loop. Ta ukaz pomeni, da program, ki je napisan med Do in Loop, teoretično poteka v neskončnost. Prične se pri Do, ko pa med izvajanjem od ukaza do ukaza pride do Loop, se vrne na Do in tako naprej, dokler ne izklopimo napajalne napetosti. Morda se bo komu to zdelo nelogično, vendar vam bom pokazal, da večina programov poteka v zanki Do Loop, izjemno redki programski primeri so narejeni brez te zanke.

4 Prvi program

Prvi program bo izjemno enostaven. Na mikrokontroler bomo priključili LED-ice (enostavno povedano: to so polprevodniške lučke, ki se v primerjavi z lučkami z žarečo nitko izjemno malo grejejo in so tudi fizično majhne), ki jih bomo z ustreznim programom vklapljali in izklapljali. LED-ice so zelo primerne za krmiljenje z mikrokontrolerjem, saj za normalno svetilnost potrebujejo le nekaj voltov napetosti in nekaj mA toka, kar mikrokontroler, ki ga bomo obravnavali, z lahkoto zagotovi. Električno vezje je prikazano na *sliki 2*. Pozorni bralci bodo na shemi poleg mikrokontrolerja in LED-ic opazili tudi upore in resonator. Upor R1 služi za ustrezno resetiranje mikrokontrolerja, resonator služi generiranju takta, ki ga za svoje delovanje potrebuje ALU, ostali upori služijo omejitvi toka, ki teče skozi LED-ice. Vezju je še dodan konektor za programiranje mikrokontrolerja in s tem je osnovni opis vezja zaključen.

Prvi primer programa, s katerim bomo vklapljali in izklapljali LED-ice, bo videti takšen:

```
Dim Ledica As Bit

Do

Config Portd = Output
Portd.0 = Ledica
Ledica = Not Ledica
Wait 1

Loop
```

V tem programu smo najprej definirali vrsto spremenljivke, ki jo bomo uporabili. Ker vklapljamo oziroma izklapljamo samo eno LED-ico, je dovolj, da imamo spremenljivko vrste Bit.

Sledi zanka Do-Loop, v kateri ni skoraj nič programa. Najprej je potrebno definirati, ali je določen port vhodni ali izhodni. V našem primeru je izhodni, zato smo ga tako tudi definirali. Nadalje smo definirali, na katerem V/I-portu se nahaja ta LED-ica. V principu lahko LED-ico priključimo na kateri koli prosti port mikrokontrolerja in to definiramo v programu – kar je s stališča izdelave vezja zelo praktično. Pri zahtevnejših projektih je namreč težje narediti enostavno tiskano vezje kot pa spremeniti program.

Sledi ukaz.

```
Ledica = Not Ledica in Wait 1
```

Ta stavek negira vrednost spremenljivke Ledica: če je bila predhodno 1, bo zdaj 0 in obratno. Ukaz

```
Wait 1
```

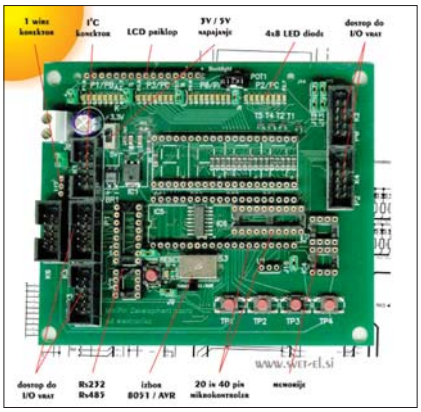
je trivialen in pomeni, da mikrokontroler na tem mestu v programu počaka 1 sekundo.

Enostaven program bo povzročil, da bo LED-ica utripala v enosekundnem taktu.

Zdaj smo napisali program, vezje pa bo treba tudi realizirati. Tukaj se bomo zamudili malce več časa, saj bom predstavil razvojno okolje MiniPin, s katerim veliko elektronikov programira pa tudi testira svoje programe.

5 Opis razvojnega orodja MiniPin in programiranje

Razvojno orodje MiniPin ni nič drugega kot ploščica tiskanega vezja, na katerem so že prispajkani določeni elektronski elementi, ki jih programer, ko programira mikrokontroler, s pridom uporablja. MiniPin lahko uporabimo tako za 20- kot tudi za 40-pinske mikrokontrolerje družine 8051 in AVR. Ker v našem nadaljevanju teče beseda o programiranju AVR-jev, se bomo osredotočili na AVR-je. Nadalje je na MiniPin že

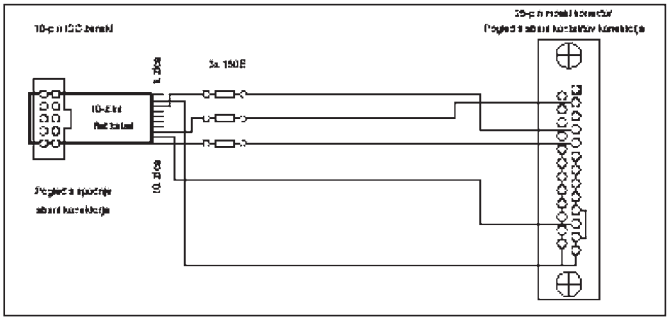


Slika 3. *Fotografija MiniPina z označenimi elementi*

predviden prostor za kvarčni kristal oziroma 3-pin keramični resonator, ki ga lahko uporabimo namesto kvarčnega kristala. Na vezju najdemo 4 x 8 LED-ic (ki so vezane tako, kot je prikazano na *sliki 2*), 4 tipke + 1 reset tipko, priključke za LED- in LCD-prikazovalnike, elemente1Wire, spominske elemente I2C, RS232, RS485 ter integrirani regulator napetosti, s katerim napajalno napetost mikrokontrolerja nastavimo na 5 V ali na vedno bolj popularnih 3,3 V. MiniPin ima tudi več razširitvenih konektorjev, ki jih bomo spoznali sproti.

MiniPin napajamo z enosmerno ali izmenično napetostjo od 9 do 15 V. V vezje vtaknemo mikrokontroler ATTiny2313-20PU na mesto IC6, na mesto U1 vtaknemo 12-MHz kvarc ali resonator, stikalo S3 preklopimo v položaj AVR in že smo pripravljeni, da povežemo MiniPin s programatorjem in sprogramiramo mikrokontroler.

Za programator lahko uporabimo programator Proggy ali pa si sami naredimo programator po načrtu, ki ga vidite na *sliki 4*.



Slika 4. *Shema programatorja Sample*

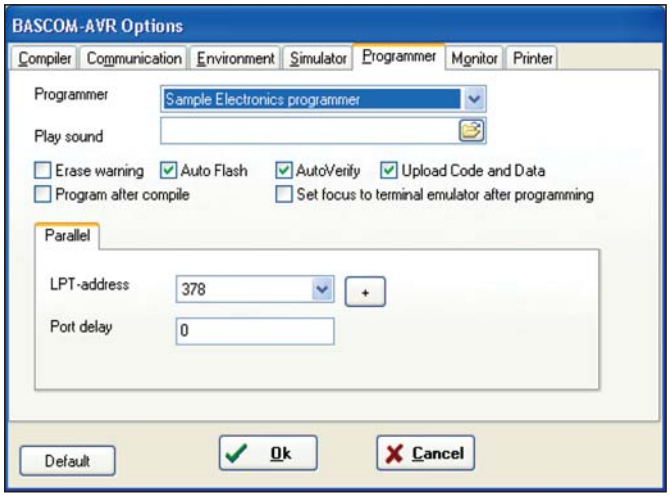
Tako programator Proggy kot programator Sample povežemo na MiniPin na konektor K6. V Bascomu je potrebno nastaviti (Options/Programmer) izbrani programator, kot vidite na *sliki 5*.

V LPT-address morate nastaviti ustrezen naslov, ki ga najdete v nastavitvah BIOS. Če boste uporabljali Proggy, boste nastavitve napravili tako, kot jih vidite na *sliki 6*.

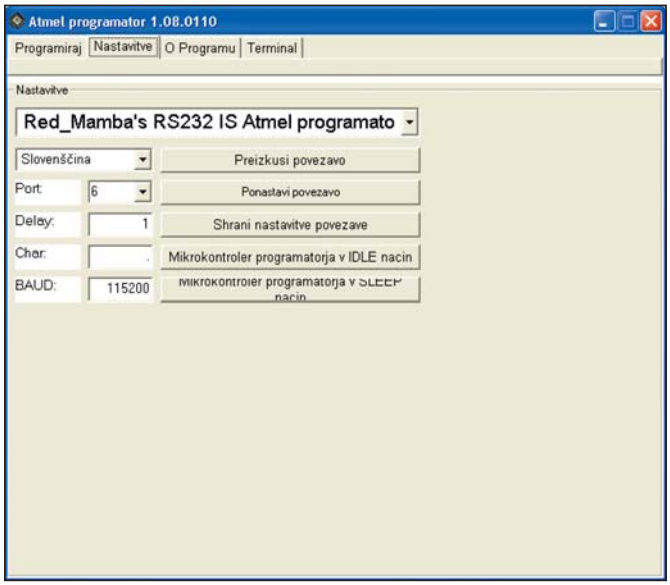
Pri teh nastavitvah izberite USB-vrata in kliknite Preizkusi povezavo. Če je povezava uspešna, se bo to izpisalo z ustreznim sporočilom.

Zdaj smo pa že res pri koncu nastavitev strojne in programske opreme. Prvi program, ki smo ga napisali v Bascomu, prevedemo (Compile ali tipka F7) in v primeru, da uporabljamo programator Sample, kliknemo ikono Program Chip ali pritisnemo F4. Če pa uporabljamo Proggy, v njegovem oknu odpremo datoteko programa (Program1.bin).

Ne glede na uporabljeni programator nastavimo mikrokontroler, ki ga programiramo (ATTiny 2313) in kliknemo Program. S tem smo sprogramirali mikrokontroler in če je bilo vse v redu, bi na MiniPinu že morala utripati LED-ica.



Slika 5. *Nastavitev programatorja*



Slika 6. *Nastavitve programatorja Proggy*

6 Zaključek

V 1. delu smo spoznali nekaj osnov mikrokontrolerjev, programskega jezika Bascom ter razvojnega okolja MiniPin ter sprogramirali prvi program. V naslednjem nadaljevanju bomo posvetili več prostora samemu spoznavanju z Bascomom in osnovnimi električnimi vezavami mikrokontrolerja. Spoznali bomo, kako na mikrokontroler povežete tranzistor FET, rele, tipke, motor in kako so videti programi za krmiljenje omenjenih elementov, zato ne zamudite tega članka.