Geekcreit UNO R3 starter kit (2) - digitalni vhodi in izhodi

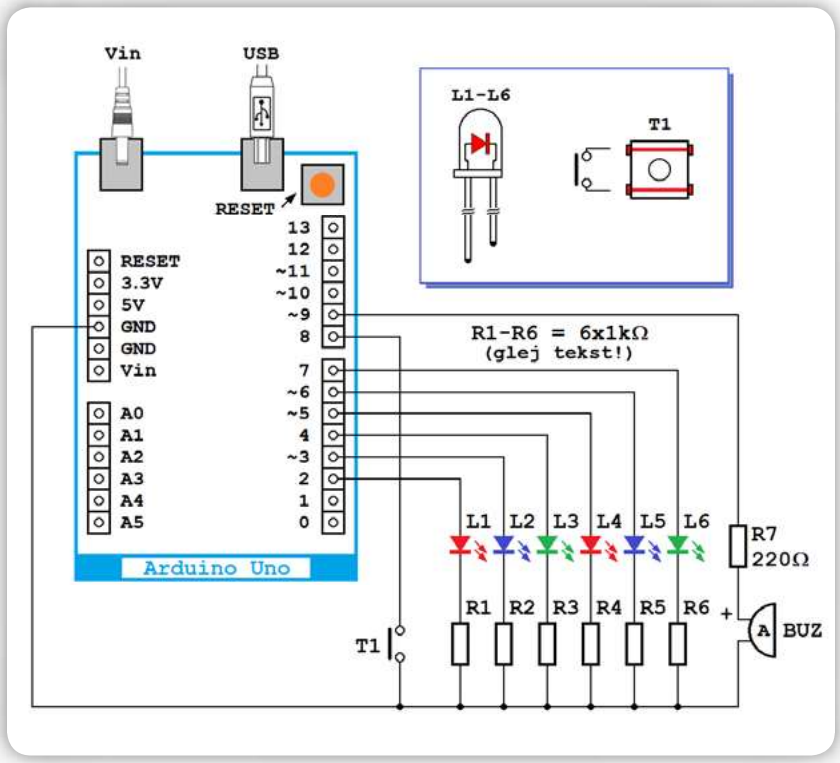
V tem poglavju bomo pokazali kako uporabljati digitalne vhode in izhode Arduino UNO ploščice.

Pod pojmom „digitalni izhod" razumemo priključek, ki lahko zavzame dve stanji: logično ničlo in logično enico. Te logični stanji ustrezata napetostim 0 V in 5 V. Vsak od digitalnih izhodov lahko generira (kadar je v stanju „1") ali sprejme (kadar je v stanju „0") tok do 40 mA. Mi bomo naša vezja projektirali tako, da ta tok ne bo večji od 20 mA.

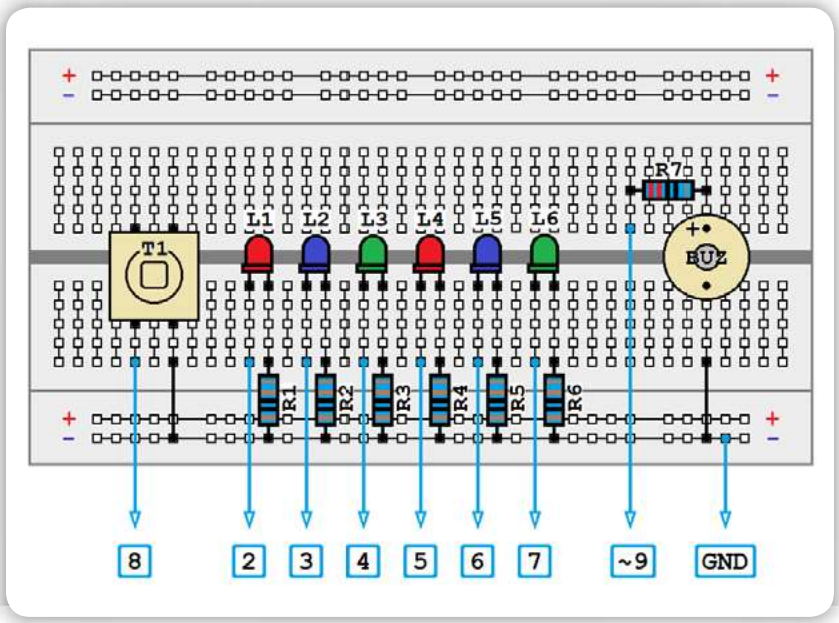
Pod pojmom „digitalni vhod" razumemo priključek, ki lahko razlikuje isti dve logični stanji, „0" in „1". Mikrokontroler ATmega328P bo vhodno napetost v razponu od -0,5 V do 1,5 V tolmačiti kot logično ničlo, medtem ko bo napetosti v razponu od 3,0 V do 5,5 V tolmačiti kot logično enico. Navedeni razponi ustrezajo 5 V napajalni napetosti, kar uporablja Arduino UNO.

Arduino UNO ploščica ima 14 digitalnih priključkov, označenih z oznakami 0 do 13 (slike 2 in 11). Priključki so povezani s priključki mikrokontrolerja ATmega328P, ki so lahko programsko konfigurirani kot vhodni ali kot izhodni. Posledično bomo tudi vsakega od priključkov 0-13 lahko uporabili kot digitalni vhod ali kot digitalni izhod, odvisno od tega, kako smo napisali naš program. Pravzaprav se tudi analogni vhodi na Arduino UNO ploščici, A0 do A5, lahko po potrebi uporabljajo tudi kot digitalni vhodi ali izhodi. V tem primeru jih lahko naslavljamo tudi z numeričnimi oznakami 14 do 19. Tako na Arduino UNO ploščici razpolagamo s skupaj 20 priključki, katerih digitalne karakteristike so enake prej navedenim.

Shema vezja, za katero bomo pisati programe v tem nadaljevanju, je prikazana na sliki 11. Tipko T1 smo vezali na priključek 8, zato bomo ta priključek uporabili kot digitalni vhod. Priključki 2-7, na katere smo vezali LED-ice L1-L6, bodo digitalni izhodi: z njimi bomo vklapljali in izklapljali LED-ice. Posamezna dioda bo zasvetila takrat, ko pridruženi izhod programsko postavimo v stanje „1", tokovi skozi diode so omejeni z upori R1-R6 na 2,5-3 mA (tok je odvisen tudi od padca napetosti na diodi, ki se razlikuje pri diodah različne barve). Uporabljamo še en digitalni izhod 9, s katerim krmilimo aktivni buzzer BUZ. Aktivni buzzer ima vgrajen oscilator in bo zapiskal, če ga vežemo na napetost 5 V, oziroma ko izhod



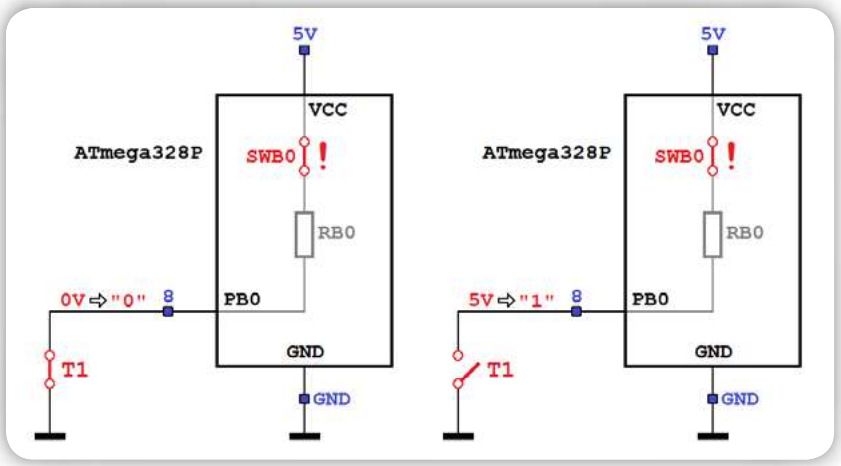
Slika 11: Shema vezja, ki ga bomo „oživeli" s programi Geekcreit_2.ino in Geekcreit_3.ino.



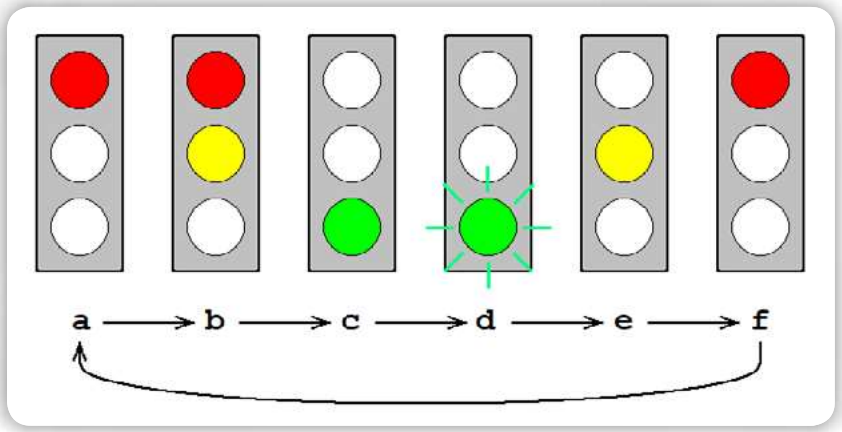
Slika 12: Tako bomo razporedili komponente na veliko testno ploščico.

9 postavimo v stanje „1“. V Geekcreit kompletu sta dva buzzerja, aktiven in pasiven: aktiven je tisti z zaščitno nalepko, ki jo je treba odstraniti pred uporabo. Da piskanje ne bi bilo preglasno, smo v serijo z buzzerjem vezali upori R7 – z njim smo znižali delovno napetost buzzerja in tok skozi njega na okoli 8 mA, pri čemer je buzzer še vedno dovolj glasen.

Kako bomo komponente razporediti na veliki testni ploščici, prikazuje slika 12. Pri postavljanju LED-ic moramo paziti kako jih obrnemo: katoda je krajši priključek, oziroma priključek poleg katerega je rob ohišja odrezan. Ta „odrezan“ rob je prikazan tudi na sliki 12, vendar je bolje viden na sliki 11 zgoraj desno. Poleg LED-ice na sliki 11 je prikazana tudi risba tipk iz Geekcreit kompleta. Opazili bomo, da sta od štirih priključkov na teh tipkah, dva in dva medsebojno povezana znotraj ohišja, zato je tipko T1 potrebno postaviti točno tako, kot je prikazano na sliki 12. Na ohišju aktivnega buzzerja je označen + priključek in tudi njega je potrebno obrniti proti uporu R7. Oznaka se včasih ne vidi jasno in če ga slučajno obrnemo napačno, buzzer ne bo piskal. Pri tem se ne bo zgodilo nič slabega: upor R7 bo pred uničenjem zaščitil tako buzzer, kot tudi mikrokontroler.



Slika 13: Ilustracija poleg pojasnjevanja digitalnih vhodov.



Slika 14: Faze dela semaforja.

Preden postavimo programsko nalogo, se naučimo še nekaj o digitalnih vhodih. Če povežemo digitalni vhod na izhod nekega vezja ali senzorja, ki „proizvaja“ napetosti znotraj prej navedenih razponov, jih bo mikrokontroler prepoznal kot digitalno ničlo ali enico. Če pa povežemo na digitalni vhod tipko ali vezje, bo mikrokontrolerju potrebno pomagati da ugotovi, v kakšnem stanju se ta tipka ali vezje nahaja.

Risba na sliki 13 ilustrira, kako to običajno delamo. Tipka T1 je vezana na priključek 8 Arduino UNO ploščice, enako kot na shemi na sliki 11, priključek 8 pa je povezan s priključkom PB0 mikrokontrolerja ATmega328P. Znotraj mikrokontrolerja so vsem vhodnim priključkom pridruženi t.i. pull-up upori in vezja, s katerimi lahko te upore povežemo na napajalno napetost Vcc. Pull-up upor in vezje PB0 smo na shemi označili z oznakama RB0 in SWB0. Ko je stikalo SWB0 zaprto, bo napetost na PB0, oziroma na priključku 8 Arduino UNO ploščice, odvisna od stanja tipke T1:

- *Če je tipka T1 pritisnjena, je njen kontakt sklenjen in naredi kratek stik proti masi; napetost priključka 8 in pridruženega pina bo 0 V, zato program v mikrokontrolerju prebere logično ničlo in po tem «ve», da je tipka pritisnjena.*
- *Če tipka ni pritisnjena, bo njen kontakt odprt; napetost priključka 8 in pridruženega pina bo 5 V. Zato bo program v mikrokontrolerju prebral logično enico in po tem «ve», da tipka ni pritisnjena.*

2. programska naloga

LED-ice L1 (rdeča), L2 (modra) in L3 (zelena) na slikah 11 in 12 tvorijo semafor (ker v Geekcreit kompletu ni rumenih LED-ic, bo naš semafor namesto rumene imel modro barvo!). Semafor dela po režimu ilustriranem na sliki 14:

- *rdeča LED-ica L1 sveti, dokler na kratko ne pritisnemo tipke T1;*
- *ko pritisnemo tipko T1, se poleg rdeče vklopi tudi rumena/modra LED-ica L2;*
- *po izteku 2 sekund, se izklopi rdeča in rumena/modra, vklopi pa se zelena LED-ica L3;*
- *to stanje se zadrži dokler ponovno na kratko ne pritisnemo tipke T1;*
- *ko pritisnemo T1, se bo zelena LED-ica štirikrat izmenično vklopila in izklopila v trajanju po 250 ms;*
- *nato se zelena izklopi in vklopi se rumena/modra LED -ica;*
- *po izteku dveh sekund se izklopi rumena/modra in vklopi se rdeča LED-ica;*
- *to stanje se zadržuje dokler se ponovno ne pritisne tipka T1.*

Vsak pritisk na tipko T1 je potrebno potrditi s kratkotrajnim zvočnim signalom buzzerja BUZ.

Arduino rešitev (program Geekcreit_2.ino)

Opazili bomo, da ima semafor dve delovni fazi: prehod svetlobe od rdečo v zeleno in prehod svetlobe iz zelene v rdeče. Zato bomo na ustrezen način morati spremljati, v kateri fazi delovanja se semafor nahaja. To bomo dosegli z uporabo spremenljivke SF, v kateri zapisujemo trenutno stanje semaforja. Stanjem bomo pridružiti vrednosti 0 in 1, pri čemer „0" označuje, da na semaforju sveti rdeča LED-ica, stanje „1" pa označuje, da na semaforju sveti zelena LED-ica. Za vsako fazo bomo sprogramirali posebno funkcijo.

Pa začnimo! Najprej bomo definirali spremenljivke z imeni komponent in pripadajočimi pini. Na primer, za LED-ico L1 bomo uporabili spremenljivko z imenom L1 in jo pridružili pin številka 2:

```
byte L1 = 2;
byte L2 = 3;
byte L3 = 4;
byte T1 = 8;
byte BUZ = 9;
byte SF = 0;
```

Sedaj moramo v funkciji setup() konfigurirati priključke za LED-ice L1, L2, L3 in buzzer BUZ kot izhodne, priključek za tipko T1 definiramo kot vhodni pin z vključenim pull-up uporom. Takoj bomo vključili rdečo LED-ico:

```
void setup() {
  pinMode(L1, OUTPUT);
  pinMode(L2, OUTPUT);
  pinMode(L3, OUTPUT);
  pinMode(T1, INPUT_PULLUP);
  pinMode(BUZ, OUTPUT);
  digitalWrite(L1, HIGH);
}
```

V funkciji loop() preverjamo, ali je pritisnjena tipka T1 s pomočjo funkcije digitalRead(). Spomnimo se: ko tipka T1 ni pritisnjena, se na pinu 8 nahaja 5 V in prebere se logična enica; ko je tipka T1 pritisnjena, se na pinu 8 nahaja 0 V in prebere se logična ničla. Če je tipka T1 pritisnjena, za 100 ms vključimo buzzer in nato ga izklopimo.

```
void loop() {
  if (digitalRead(T1) == 0){
    digitalWrite(BUZ, HIGH);
    delay(100);
    digitalWrite(BUZ, LOW);
```

Nato preverjamo v kakšnem stanju je semafor. V kolikor je v stanju vključene rdeče LED-ice (vrednost spremenljivke SF je 0), zaženemo funkcijo sf0() in spremenimo stanje spremenljivke SF u 1. V nasprotnem primeru zaženemo funkcijo sf1() in spremenimo stanje spremenljivke SF v 0.

```
  if (SF == 0){
      sf0();
      SF = !SF;
    } else {
      sf1();
      SF = !SF;
    }
  }
}
```

V funkciji sf0() svetloba prehaja iz rdeče v zeleno: vklopimo modro LED-ico L2, počakamo dve sekundi (= 2000 ms), nato izklopimo rdečo in modro LED-ico L1 in L2 ter vklopimo zeleno LED-ico L3:

```
void sf0(){
  digitalWrite(L2, HIGH);
  delay(2000);
  digitalWrite(L1, LOW);
  digitalWrite(L2, LOW);
  digitalWrite(L3, HIGH);
}
```

V funkciji sf1() svetloba prehaja iz zelene v rdečo: zelena LED-ica L3 se izmenično štirikrat vklopi in izklopi v trajanju po 250 ms, nato jo bomo izklopili. Kmalu nato vklopimo modro LED-ico L2, počakamo dve sekundi, jo izklopimo in končno vklopimo rdečo LED-ico L2. Za izmenično vklapljanje in izklapljanje LED-ice L3 uporabljamo for zanko v kateri definiramo, da se vrednost spremenljivke tudi menja od 0 (int i = 0) do 3 (i < 4) s korakom 1 (i++) in vklapljamo ter izklapljamo LED-ico L3 z 250 ms pavzami. Ker se ukazi za vklapljanje in izklapljanje LED-ice nahajajo znotraj zanke, se bo izvršila 4-krat:

```
void sf1(){
  for (int i = 0; i < 4; i++){
    digitalWrite(L3, LOW);
    delay(250);
    digitalWrite(L3, HIGH);
    delay(250);
  }
  digitalWrite(L3, LOW);
  digitalWrite(L2, HIGH);
  delay(2000);
  digitalWrite(L2, LOW);
  digitalWrite(L1, HIGH);
}
```

Naloga za napredne programerje

LED-ice L4, L5 in L6 tvorijo drugi semafor, ki se nahaja v nasprotnem stanju od prvega:

- *kadar je na prvem semaforju vklopljena rdeča, je na drugem vklopljena zelena in obratno;*
- *kadar prvi semafor prehaja fazo od rdeče proti zeleni svetlobi, drugi semafor prehaja faze od zelene proti rdeči svetlobi;*
- *kadar prvi semafor prehaja faze od zelene proti rdeči svetlobi, drugi semafor prehaja faze od rdeče proti zeleni svetlobi.*

Prilagodite trajanja posameznih faz tako, da prehod skozi oba semaforja niti v enem trenutku ne bo istočasno odprt! Programsko rešitev Geekcreit_2a.ino ne bomo analizirali tukaj, lahko pa jo dobite od uredništva revije Svet elektronike.

3. programska naloga

Za vezje na slikah 11 in 12 napišite program po naslednjih navodilih:

- *na začetku izvrševanja programa je vklopljena LED-ica L1, vse ostale so izklopljene;*
- *s pritiskom na tipko T1 se izklopi L1 in vklopi L2;*
- *z vsakim novim pritiskom na tipko T1 se vklopi naslednja LED-ica v nizu;*
- *ko se izklopi L6, se ponovno vklopi L1 in proces se prične od začetka.*

Arduino rešitev (program Geekcreit_3.ino)

Rešitvi te naloge bomo pristopiti malo drugače! Imamo šest LED-ic s priključki v nizu; zato ne bomo za vsako LED -ico definirali lastne spremenljivke, ampak bomo definirali spremenljivko, v katero shranimo pin trenutno vključene LED-ice in ga bomo poimenovali ACTIVE_LED. Začetna vrednost spremenljivke ACTIVE_LED bo številka pina, na katero je vezana LED-ica L1. Prav tako definiramo spremenljivko za tipko T1.

```
byte T1 = 8;
byte ACTIVE_LED = 2;
```

Ker so priključki v vrsti, lahko v funkciji setup() konfiguriramo priključke LED-ic s pomočjo for zanke. Priključek, na katerem je vezana tipka T1, je vhodni z vključenim pull-up uporom. Nato vklopimo trenutno aktivno LED-ico:

```
void setup() {
  for (int i = 2; i < 8; i++){
    pinMode(i, OUTPUT);
  }
  pinMode(T1, INPUT_PULLUP);
  digitalWrite(ACTIVE_LED, HIGH);
}
```

V funkciji loop() preverjamo, ali je tipka T1 pritisnjena in če je, izključimo aktivno LED-ico:

```
void loop() {
  if (digitalRead(T1) == 0){
    digitalWrite(ACTIVE_LED, LOW);
```

Nato povečamo vrednost v spremenljivki ACTIVE_LED za ena zato, da bi naslednja LED-ica v vrsti postala aktivna. Izjema se pojavi, če je aktivna LED-ica bila na pinu 7; v tem primeru bo naslednja aktivna LED-ica tista na pinu 2:

```
    if (ACTIVE_LED == 7) {
      ACTIVE_LED = 2;
    } else {
      ACTIVE_LED++;
    }
```

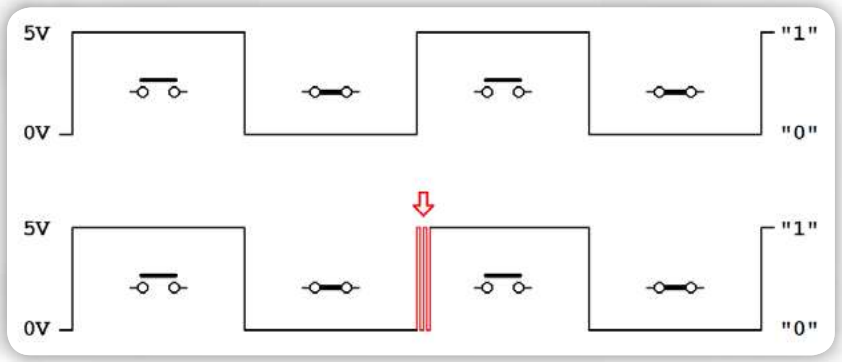
Sedaj lahko vključimo novo aktivno LED-ico tudi s pomočjo prazne while() zanke in počakamo, dokler se tipka ne spusti:

```
 digitalWrite(ACTIVE_LED, HIGH);
    while (digitalRead(T1) == 0){}
  }
}

* * *
```

Zaradi nepopolnosti mehanskih tipk in stikal, sklenitev in odpiranje stika ni vedno idealno: elastično pero včasih zaniha pri spremembi položaja. Pojav se imenuje bouncing in je ilustriran z risbo na sliki 15. Ta nestabilnost traja komaj kakšno milisekundo, ampak tudi mikrokontroler je hiter in bo namesto enega, registriral še več „lažnih" sklenitev kontakta. Zato vrstni red vklapljanja LED-ic ne bo idealen, pač pa bo včasih „preskočila" katera od LED-ic. Kako pogosto se bo to dogajalo je odvisno od iztrošenosti kontaktov, jakosti pritiska na tipko in hitrostjo tipkanja. Pokažimo, kako bi programsko rešili opisani problem!

Arduino rešitev (program Geekcreit_3a.ino)



Slika 15: Nihanje stikala (bouncing).