Ponavljanje

Več diod

Na Arduina priključi sedem diod. Najboljše, da uporabiš pine od `D3` do `D9`. Če imaš ploščico s sedmimi diodami, toliko boljše. Če nimaš sedmih diod in upornikov, jih priključi manj - a kakih pet naj jih vendarle bo. V besedilu pa bomo predpostavljali, da jih sedem.

Naloga 2.1. Želimo si, da bi vse diode hkrati utripale. Napiši program za to.

Spremenljivke

Program, ki si ga napisal, je verjetno dolg in dolgočasen, drži? Del `setup` je bil najbrž takšen:

```
void setup() {
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
}
```

V `loop` se potem ponovita še dva podobna dolgčasa. Naučili se bomo, kako takšne reči napisati krajše. Veliko krajše. Izvedeli bomo, kako programu reči, naj nekaj ponavlja. Najprej pa moramo spoznati nekaj drugega: spremenljivke.

Si se kdaj igral s programom Scratch? Potem najbrž veš za spremenljivke? Če ne, pa nič hudega: spremenljivke so kot škatlice, v katere lahko shraniš kako število ali besedilo. Vsaka škatlica (spremenljivka) mora imeti ime. Po imenih jih namreč ločimo. Imena so sestavljena iz števk, črk (šumniki č, š, ž so prepovedani) in podčrtajev; začeti se morajo s črko ali podčrtajem.

Spremenljivko moramo najprej ustvariti. Recimo, da bo naši spremenljivki ime `i`. Ustvarimo jo tako, da napišemo

```
int i;
```

Zakaj ravno z besedo `int`? Kaj pravzaprav pomeni `int`? `int` je okrajšava za *integer*

oziroma *integer number*, po angleško *celo število*. S tem torej povemo, da bo `i` vseboval cela števila.

Ko ustvarimo spremenljivko, ta obstaja znotraj bloka, v katerem smo jo ustvarili. Če jo naredimo znotraj `setup`, jo lahko uporabljamo v `setup`, če v `loop`, pa v `loop`. Lahko jo naredimo tudi izven, vendar tega za zdaj ne bomo počeli. Če potrebujemo `i` v obeh, ga bomo tudi ustvarili v obeh.

Kako spravimo število v to "škatlico" `i`? Tako, da `i`-ju *priredimo vrednost*, recimo

```
i = 3;
```

Zdaj lahko `i` uporabljamo povsod, kjer bi sicer napisali številko. Napišemo lahko, na primer

```
pinMode(i, OUTPUT);
```

in to bo pomenilo isto kot

```
pinMode(3, OUTPUT);
```

Spremenljivka se imenuje spremenljivka zato, ker jo lahko spreminjamo. Čeprav smo zgoraj napisali `i = 3;`, si lahko malo kasneje v programu premislimo in nadaljujemo z

```
i = 6;
```

ali celo z

```
i = i + 1;
```

Ta, slednja vrstica je kar zanimiva in pogosta: z njo povečamo `i` za 1. Če je bil prej `3`, bo zdaj postal `i + 1`, se pravi `3 + 1`, torej `4`.

Naš `setup` lahko spremenimo v

```
void setup() {
    int i;
    i = 3;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
    pinMode(i, OUTPUT);
    i = i + 1;
}
```

Z lepo slovensko frazo se temu reče, da smo *prišli z dežja pod kap*: program, ki je bil dolg in dolgočasen je zdaj še daljši in dolgočasnejši. Vendar poglejmo, kaj dela: le dve vrstici ponavlja. Bi mu lahko rekli, naj ju ponavlja, dokler … hm, dokler je `i` manjši (ali enak) 9.

Seveda lahko. Prav s tem namenom smo šli v to smer.

Zanka

Narediti moramo takole.

```
void setup() {
    int i;
    i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i = i + 1;
    }
}
```

Da bomo lažje razumeli, prevedimo v slovenščino

```
ustvari spremenljivko i
nastavi i na 3
dokler je i manjši ali enak 9:
    nastavi pinMode i na OUTPUT
    povečaj i za 1
```

Jasno?

Temu, kar smo pravkar naredili - skupino vrstic, ki se ponavlja -, rečemo *zanka*. Konkretno, zanka *while*. Če boste pridni, boste spoznali še druge vrste zank.

`while` je angleška beseda za *dokler*. Vedno ji mora slediti pogoj, zaprt v *navadne oklepaje*. Ta pove, kaj mora držati, da se bo zanka ponavljala. V našem primeru se vrstice znotraj zanke ponavljajo, *dokler je i manjši ali enak 9*, `while (i <= 9)`. Kombinacija `<=` (med znaka ne smemo postaviti presledka!) pomeni “manjše ali enako”, ker znaka `≤`, ki smo ga vajeni iz matematike, ni na tipkovnici. Pogoju morajo (brez podpičja!) slediti *zaviti oklepaji*, v katere zapremo vrstice, ki se morajo ponavljati. V našem primeru sta to vrstici

```
        pinMode(i, OUTPUT);
        i = i + 1;
```

Bodite pozorni še na to, kako smo oblikovali program. Napisali bi lahko tudi

```
void setup() {
int i;
i = 3;
while (i <= 9) {
pinMode(i, OUTPUT);
i = i + 1;
}
}
```

Vendar je to veliko manj pregledno in ko bodo programi postajali vedno daljši, bo preglednost vedno pomembnejša. Dogovorimo se torej, da bo:

- zaviti oklepaj vedno v isti vrstici kot `while`,
- zaviti zaklepaj vedno poravnan z `while`,
- vse, kar je znotraj zavitih oklepajev, zamaknjeno za štiri presledke v desno.

Tega pravila smo se že držali tudi pri oklepajih za `void setup()` in `void loop()`. Tudi pomen zavitih oklepajev je v obeh primerih isti: vse, kar je v zavitih oklepajih za `setup` sodi v blok *setup* in vse, kar je v zavitih oklepajih za `while`, sodi v zanko in se ponavlja.

Naloga 2.2. Zdaj, ko smo se naučili uporabljati zanke, predelaj svoj program za utripanje s sedmimi diodami. Za zdaj pozabi na utripanje: poskrbi le, da se bodo diode prižgale. Pobriši drugo polovico funkcije `loop`, prvo polovico pa predelaj tako, da bo namesto ponavljanja `digitalWrite` uporabila zanko.

Naloga 2.3. Dopolni `loop`, da bo teh sedem diod utripalo.

Naloga 2.4. Spremeni svoj program tako, da se diode ne bodo prižigale in ugašale hkrati temveč ena za drugo; nova dioda naj se prižge oziroma ugasne, recimo, vsakih 200 ms.

Naloga 2.5. Spremeni program tako, da se bodo diode ugašale v obratnem vrstnem redu, kot se prižigajo. Če se prižigajo z leve proti desni, naj se ugašajo z desne proti levi. Namig: vrednosti spremenljivke v drugi zanki ne bodo šle od 9 do 3 temveč od 3 do 9.

Naloga 2.6. Zdaj pa spremeni to, kar si naredil tako, da se bodo diode prižigale in ugašale ena za drugo. Najprej se prižge prva. Nato se prva ugasne, istočasno pa se prižge druga. Ko se ugasne druga, se istočasno prižge tretja… Ko se ugasne sedma, se spet prižge šesta. Ko se ugasne šesta, se spet prižge peta in tako naprej do tretje. Pazi, da sedma dioda ne bo prižgana dvakrat dlje od ostalih. Če hočeš, da bo izgledalo še posebej imenitno, skrajšaj zamik na kakih 20 ali 50 ms.

Bližnjice

Kadar računalnikarji pogosto pišejo ene in iste stvari, si radi uvedejo kako bližnjico. Tule sta dve preprosti.

Namesto

```
int i;
i = 3;
```

lahko pišemo kar

```
int i = 3;
```

Ko kasneje spreminjamo vrednost `i` -ja, seveda še vedno pišemo

```
i = 8;
```

ali

```
i = i + 1;
```

in ne

```
int i = 8;
```

ali

```
int i = i + 1;
```

Druga bližnjica se nanaša na povečevanje vrednosti spremenljivk. Ker tako pogosto povečamo `i` za 1, obstaja za

```
i = i + 1;
```

kar okrajšava

```
i++;
```

To seveda ne velja le za spremenljivke z imenom `i`. Če imamo spremenljivko `blabla`, bomo lahko pisali

```
blabla++;
```

Prav tako lahko namesto

```
i = i - 1;
```

pišemo

```
i--;
```

Takšnih bližnjic je še veliko, vendar človek ne sme zajemati s preveliko žlico.

Naloga 2.7. Da povadiš bližnjice, skrajšaj zadnji program, ki si ga napisal.

Namigi

Naloga 2.1. Utripanje s sedmimi diodami

Kje se ti je ustavilo?

Ne veš, kako prižgati vse diode hkrati? Tega se v resnici ne da, niti ni potrebno. Če napišeš

```
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
```

ne bo nihče opazil, da sta se diodi prižgali ena za drugo in ne (čisto hkrati), saj bosta vmes minili le kaki dve milijoninki sekunde. Tako kot tidve prižgi še ostalih pet. Nato dodaj malo pavze in nato na enak način še ugasni vse diode.

Se diode nočejo prižgati ali pa so ves čas prižgane? Preveri, kam si postavil `delay`. Primerjaj program s tistim za prižiganje in ugašanje diod. V resnici sta skoraj enaka, le sedem vrstic z `digitalWrite` in različnimi številkami pinov potrebuješ.

Naloga 2.2. Prižiganje diod z zanko

Res razumeš, kako deluje `setup`, opisan v gornjem besedilu? `loop` je namreč popolnoma enak, le namesto `pinMode` vsebuje `digitalWrite` z ustreznimi podatki.

Naloga 2.3. Utripanje z zanko

Bo zadoščala le ena zanka ali pa boš potreboval dve - eno za prižiganje in eno za ugašanje?

Kam boš postavil pavze? Eno ali dve? V mislih sledi programu, v takšnem vrstnem redu kot bo šel prek vrstic - vključno s ponavljanjem - Arduino, ko bo izvajal tvoj program.

Naloga 2.4. Prižiganje in ugašanje po vrsti

Ves trik je v tem, kam postaviti `delay`. V katerem trenutku mora Arduino malo počakati?

Naloga 2.5. Ugašanje v obratnem vrstnem redu

Pri vsaki zanki, kakršne pišemo v teh nalogah moraš razmisliti o treh stvareh.

- Kakšen mora biti `i` v začetku? `3`, `9`, `8`? Če hočeš, da gre od 3 do 9, bo v začetku enak `3`. Pa kadar ga hočeš nagnati od 9 do 3?
- Kakšen je pogoj? Mora biti `i` večji (`>`), manjši (`<`), večji ali enak (`>=`) ali manjši ali enak (`<=`) od nečesa? In od česa? Od 2, 3, 8, 9…?
- Kako moraš v vsakem koraku zanke spremeniti `i`? Ga povečati za 1 (`i = i + 1;`) ali zmanjšati za 1 (`i = i - 1`)?

Polomi ga pri eni od teh treh stvari in program ne bo deloval pravilno.

Naloga 2.6.

Vrstice znotraj zank moramo spremeniti tako, da diodo prižgemo, počakamo in nato ugasnemo.

Naloga 2.7.

Kje se je zataknilo?

Če ti ne uspe prenesti programa na Arduino, preveri, recimo, ali si znotraj `loop` morda dvakrat ustvaril spremenljivko `i`.

Rešitve

Naloga 2.1. Utripanje s sedmimi diodami

Zavihamo rokave, tipkamo in kopiramo.

```
void setup() {
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
}

void loop() {
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
    digitalWrite(9, HIGH);
    delay(500);

    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
    digitalWrite(9, LOW);
    delay(500);
}
```

Prazno vrstico smo vstavili zaradi preglednosti: zgornji del `loop`-a prižiga in spodnji del ugaša.

Naloga 2.2. Prižiganje diod z zanko

`loop` bo podoben kot `setup`, le da imamo zanka vsebuje `digitalWrite(i, HIGH)` namesto `pinMode(i, OUTPUT)`

```
void setup() {
    int i;
    i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i = i + 1;
    }
}

void loop() {
    int i;
    i = 3;
    while (i <= 9) {
        digitalWrite(i, HIGH);
        i = i + 1;
    }
}
```

Naloga 2.3. Utripanje z zanko

Glavni trik naloge je, da potrebuješ dve zanki, eno za prižiganje in drugo za ugašanje. Slediti jima mora `delay`, tako da bodo diode nekaj časa prižgane oz. ugasnjene.

```
void setup() {
    int i;
    i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i = i + 1;
    }
}

void loop() {
    int i;

    i = 3;
    while (i <= 9) {
        digitalWrite(i, HIGH);
        i = i + 1;
    }
    delay(500);

    i = 3;
    while (i <= 9) {
        digitalWrite(i, LOW);
        i = i + 1;
    }
    delay(500);
}
```

V `loop` smo `i` ustvarili (z `int i`) le enkrat, čeprav ga potrebujemo v obeh zankah. Za vajo lahko dodaš še en `int i;` pred drugi `i = 3;`, da boš videl, na kakšen način se računalnik pritoži. Tako boš drugič, ko boš to naredil ponesreči, vedel, kaj ti hoče povedati.

Naloga 2.4. Prižiganje in ugašanje po vrsti

Ukaz `delay` moramo prestaviti v zanko: program mora počakati po prižiganju in ugašanju vsake diode.

```
void setup() {
    int i;
    i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i = i + 1;
    }
}

void loop() {
    int i;

    i = 3;
    while (i <= 9) {
        digitalWrite(i, HIGH);
        delay(200);
        i = i + 1;
    }

    i = 3;
    while (i <= 9) {
        digitalWrite(i, LOW);
        delay(200);
        i = i + 1;
    }
}
```

Naloga 2.5. Ugašanje v obratnem vrstnem redu.

Vrednost `i` moramo pred drugo zanko nastaviti na 9 (`i = 9;`), zanka teče, dokler je večji ali enak (`while (i >= 3`) in namesto, da bi povečevali `i`, ga zmanjšujemo (`i = i - 1`).

```
void setup() {
    int i;
    i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i = i + 1;
    }
}

void loop() {
    int i;

    i = 3;
    while (i <= 9) {
        digitalWrite(i, HIGH);
        delay(200);
        i = i + 1;
    }

    i = 9;
    while (i >= 3) {
        digitalWrite(i, LOW);
        delay(200);
        i = i - 1;
    }
}
```

Preveri, ali program deluje tudi, če izpustiš `i = 9;`. Deluje? Hm, kako to?

Naloga 2.6. Zaporedno prižiganje

Vrstice znotraj zank moramo spremeniti tako, da diodo prižgemo, počakamo in nato ugasnemo.

```
void setup() {
    int i;
    i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i = i + 1;
    }
}

void loop() {
    int i;

    i = 3;
    while (i <= 9) {
        digitalWrite(i, HIGH);
        delay(20);
        digitalWrite(i, LOW);
        i = i + 1;
    }

    i = 8;
    while (i >= 3) {
        digitalWrite(i, HIGH);
        delay(20);
        digitalWrite(i, LOW);
        i = i - 1;
    }
}
```

Mimogrede skrajšamo čase, da je reč videti imenitneje.

Paziti moramo, da `i = i + 1;` napišemo po ugašanju diode. Če bi napisali

```
        digitalWrite(i, HIGH);
        delay(20);
        i = i + 1;
        digitalWrite(i, LOW);
```

ali

```
        digitalWrite(i, HIGH);
        i = i + 1;
        delay(20);
        digitalWrite(i, LOW);
```

bi se `i` prehitro povečal in bi z `digitalWrite(i, LOW);` že ugašali naslednjo diodo - ki sploh še ni prižgana.

Si opazil, da smo pred drugo zanko napisali `i = 8;`, ne `i = 9;`? Zakaj?

Naloga 2.7. Naprej in nazaj - krajše

```
void setup() {
    int i = 3;
    while (i <= 9) {
        pinMode(i, OUTPUT);
        i++;
    }
}

void loop() {
    int i = 3;
    while (i <= 9) {
        digitalWrite(i, HIGH);
        delay(20);
        digitalWrite(i, LOW);
        i++;
    }

    i = 8;
    while (i >= 3) {
        digitalWrite(i, HIGH);
        delay(20);
        digitalWrite(i, LOW);
        i--;
    }
}
```