No mirdzdiodes līdz simbolam

Ivars Driķis

2019. gada 4. septembrī

1 **Programma** Blink

Arduino programmēšanu parasti sāk ar programmu Blink [1], kas ieslēdz un izslēdz iebūvēto spīddiodi. Arduino Uno gadījumā tā ir pieslēgta 13 izvadam. Pareizi būtu izmantot iebūvēto parametru LED_BUILTIN, bet šoreiz nākošo uzdevumu vārdā nedaudz atkāpsimies no labā stila.

Listing 1: Blink.ino

```
1  const int pinLed = 13;          // LED_BUILTIN;
2
3  void setup() {
4    pinMode(pinLed, OUTPUT);     // Didgitalo liniju 13 iesledz par izeju
5  }
6
7  void loop() {
8    digitalWrite(pinLed, HIGH);// iesledz mirdzdiodi (HIGH ir sprieguma limenis)
9    delay(1000);                // gaida vienu sekundi
10   digitalWrite(pinLed, LOW); // izsledz mirdzdiodi (LOW ir sprieguma limenis)
11   delay(1000);                // gaida vienu sekundi
12 }
```

Šeit izmantotas trīs *funkcijas*

pinMode(pin, mode) - konfigurē norādīto digitālo līniju vai nu par ieeju vai arī par izeju, [2]:

> pin: digitālās līnijas numurs
>
> mode: ja OUTPUT, tad izeja, ja INPUT vai INPUT_PULLUP tad ieeja.

digitalWrite(pin, value) - norādīto pinu uzstāda zemā vai augstā līmenī, [3]

> pin: digitālās līnijas numurs
>
> value: spriegums digitālās līnijas izejā, LOW vai HIGH

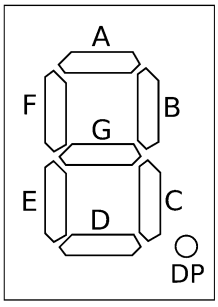
delay(ms) - aptur programmas izpildi uz noteiktu laiku [4];

> ms: laiks milisekundēs no 0 līdz $2^{32} - 1$. Tātad ilgākais laiks ir apmēram 50 diennaktis!

Savukārt const nozīmē, ka mainīgā pinLed vērtību programmas darbības laikā nemainīsim. Tas ļauj kompilatoram veikt papildus optimizāciju. Protams, šo konstrukciju var nelietot, bet ir lietderīgi jau no paša sākuma radināties darīt tā, kā ir prātīgi.

Uzdevumi patstāvīgajam darbam:

1. Mainām aiztures laiku komandā delay() programmas 7 un 9 līnijā un novērojam
2. Palielināt mirdzdiodes spīdēšanas frekvenci līdz izskatās ka tā spīd nepārtraukti.



Segments	0	1	...	8	9
A	HIGH	LOW	...	HIGH	HIGH
B	HIGH	HIGH	...	HIGH	HIGH
C	HIGH	HIGH	...	HIGH	HIGH
D	HIGH	LOW	...	HIGH	HIGH
E	HIGH	LOW	...	HIGH	LOW
F	HIGH	LOW	...	HIGH	HIGH
G	LOW	LOW	...	HIGH	HIGH

Att. 1: Segmentu izvietojums indikatorā. Simbolu veidošanas tabula

2 Septiņu segmentu indikators

Ciparu un dažu burtu attēlošanai jau sen lieto 7 segmentu indikatorus [5]. Mūsdienās tas vienkārši ir 8 mirdzdiodes, kas izvietotas kā redzams zīmējumā 1. Ciparu ka arī dažu burtu simboli veidojas pareizi ieslēdzot un izslēdzot *īstās* mirdzdiodes.

Uzdevums patstāvīgajam darbam: Aizpildām tabulu 1 zīmējuma labajā pusē katram simbolam no 2 līdz 7 norādot kuri no segmentiem ir jāieslēdz (HIGH) un kuri ir jāizslēdz (LOW) tieši tā, kā tas ir izdarīts simboliem 0, 1, 8 un 9.

2.1 Septiņu segmentu indikatora panelis

Šā darba veikšanai izveidots indikatora panelis, kura elektriskā shēma un fotogrāfija dota 2 zīmējuma augšā. Šā paša zīmējuma apakšā parādīts, kā to pieslēgt Arduino. Interesanti, ka gatavu šādu paneli neizdevās atrast un tas arī kalpoja par man iemeslu izveidot savu platīšu komplektu. Kāpēc šādu platīšu nav, ir acīmredzams. Izņemot šo mācību uzdevumu, tās ir absolūti bezjēdzīgas un tāpēc nevajadzīgas, jo vienam simbolam tiek patērētas 8 izejas.

2.2 Blinkojam indikatoru

Nedaudz pārveidojam iepriekš doto Blink programmu 1. Mainām pinLed vērtību uz 2 un novērojam, kā blinko indikatora segments G. Izmēģinām citas pinLed vērtības!

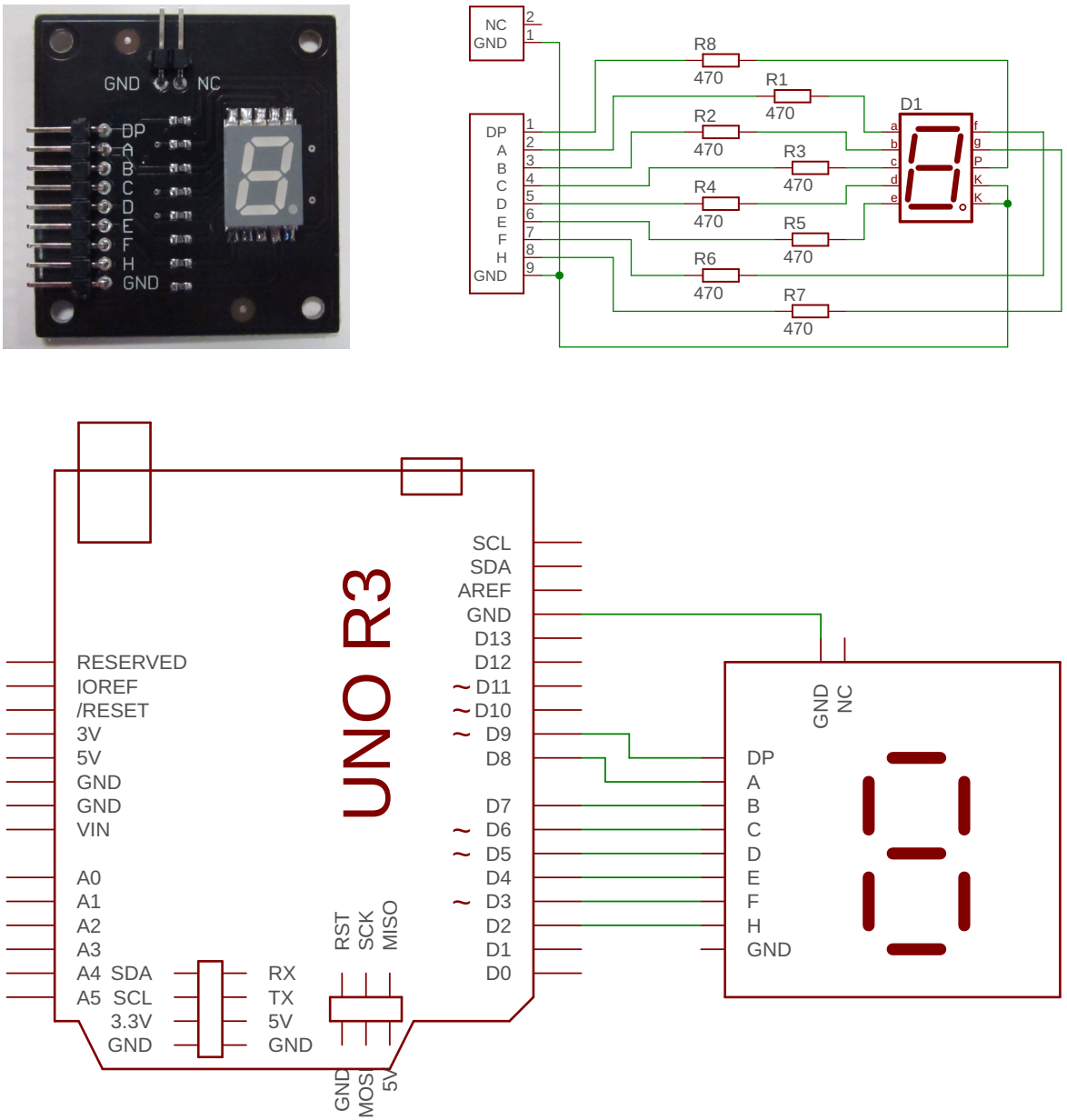
2.3 Veidojam simbolus

Izskatīsim programmu, kas attēlo uz indikatora ciparus 0, 1, 8 un 9. Sākam ar to, ka definēsim pie kuriem Arduino izvadiem ir pieslēgti indikatora segmenti. Šī programmas daļa mums kalpos arī kā dokumentācija kur ko pieslēgt.

Listing 2: Seg7Basic.ino

```
1 const int pinA = 8;
2 const int pinB = 7;
3 const int pinC = 6;
4 const int pinD = 5;
5 const int pinE = 4;
6 const int pinF = 3;
7 const int pinG = 2;
8 const int pinDp = 9;
```

Tālāk funkcijā setup() par izvadiem uzstādām *visus* pinus, kuriem pieslēgti indikatora segmenti



Att. 2: Septiņu segmentu indikatora panelis, tā shēma un savienojums ar Arduino.

```
10 void setup()
11 {
12   pinMode(pinA, OUTPUT);
13   pinMode(pinB, OUTPUT);
14   pinMode(pinC, OUTPUT);
15   pinMode(pinD, OUTPUT);
16   pinMode(pinE, OUTPUT);
17   pinMode(pinF, OUTPUT);
18   pinMode(pinG, OUTPUT);
19   pinMode(pinDp, OUTPUT);
20 }
```

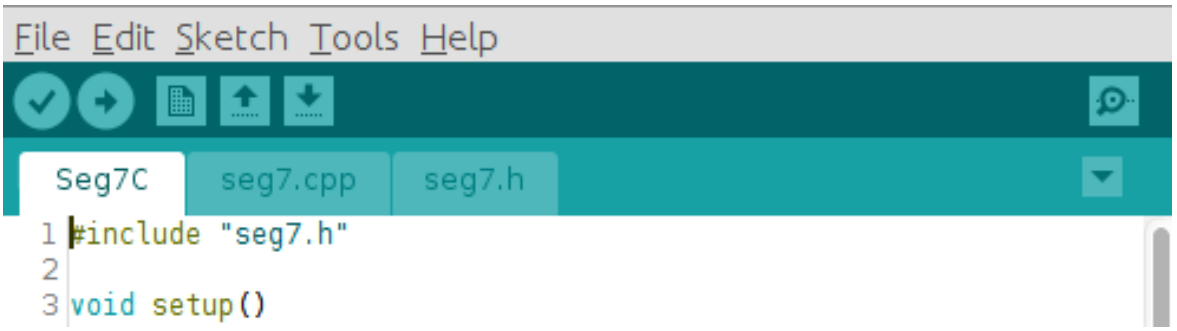
Tālāk funkcijās seg7_print0, seg7_print1, seg7_print8 un seg7_print9 izveidojam programmas koda daļas, kuras izveido uz indikatora ciparus 0, 1, 8 un 9. Pievērsiet uzmanību, ka vajadzīgie segmenti tiek ieslēgti un nevajadzīgie izslēgti. Tas vajadzīgs, lai funkcija parādītu

nepieciešamo ciparu *neatkarīgi* no tā, kas uz indikatora bija pirms tam.

```
22 void seg7_print0( void )
23 {
24   digitalWrite(pinA, HIGH);
25   digitalWrite(pinB, HIGH);
26   digitalWrite(pinC, HIGH);
27   digitalWrite(pinD, HIGH);
28   digitalWrite(pinE, HIGH);
29   digitalWrite(pinF, HIGH);
30   digitalWrite(pinG, LOW);
31   digitalWrite(pinDp, LOW);
32 }
33
34 void seg7_print1( void )
35 {
36   digitalWrite(pinA, LOW);
37   digitalWrite(pinB, HIGH);
38   digitalWrite(pinC, HIGH);
39   digitalWrite(pinD, LOW);
40   digitalWrite(pinE, LOW);
41   digitalWrite(pinF, LOW);
42   digitalWrite(pinG, LOW);
43   digitalWrite(pinDp, LOW);
44 }
45
46 void seg7_print8( void )
47 {
48   digitalWrite(pinA, HIGH);
49   digitalWrite(pinB, HIGH);
50   digitalWrite(pinC, HIGH);
51   digitalWrite(pinD, HIGH);
52   digitalWrite(pinE, HIGH);
53   digitalWrite(pinF, HIGH);
54   digitalWrite(pinG, HIGH);
55   digitalWrite(pinDp, LOW);
56 }
57
58 void seg7_print9( void )
59 {
60   digitalWrite(pinA, HIGH);
61   digitalWrite(pinB, HIGH);
62   digitalWrite(pinC, HIGH);
63   digitalWrite(pinD, HIGH);
64   digitalWrite(pinE, LOW);
65   digitalWrite(pinF, HIGH);
66   digitalWrite(pinG, HIGH);
67   digitalWrite(pinDp, LOW);
68 }
```

Visbeidzot funkcijā loop() vienu pēc otra ieslēdz šos ciparus. Izveidojas programma, kuru saucu par Džeimsa Bonda infarktu.

```
70 void loop()
71 {
72   seg7_print9();
73   delay(1000);
74   seg7_print8();
75   delay(1000);
76   seg7_print1();
77   delay(1000);
78   seg7_print0();
```



Att. 3: Projekts Arduino IDE vidē

```
79    delay(1000);
80 }
```

3 Programmas dalījums vairākos failos

Lai mūsu iepriekšējo programmu 2 varētu viegli lietot arī citos projektos, sadalīsim to vairākos failos divās daļās: īsā galvenajā daļā un otrajā daļā, uz kuru pārvietosim ciparu zīmēšanas funkcijas. Darba kārtība varētu būt sekojoša

1. Strādājam ar failu menedžeri
   - Izveido jaunu mapi Seg7C
   - Iekopē tur faila Seg7Basic.ino kopiju un nosauc to par Seg7C.ino
   - Šajā paša katalogā izveido faila Seg7C.ino kopiju un nosauc to par seg7.cpp
   - Izveido failu seg7.h
2. Atver failu Seg7C.ino ar Arduino IDE. Visam ir jāizskatās kā zīmējumā 3

Sākam ar galveno programmas daļu. Principā visu nodzēšam un rakstām par jaunu.

Listing 3: Seg7C.ino

```
1 #include "seg7.h"
2
3 void setup()
4 {
5   seg7_setup();
6 }
7
8 void loop()
9 {
10   for(int i=0; i<11; i++) {
11     seg7_print( i );
12     delay( 500 );
13   }
14 }
```

Pēc tam failu seg7.cpp pārveidojam šādi

Listing 4: seg7.cpp


```
1  #include <arduino.h>
2  #include "seg7.h"
3
4  const int pinA = 8;
5  const int pinB = 7;
6  const int pinC = 6;
7  const int pinD = 5;
8  const int pinE = 4;
9  const int pinF = 3;
10 const int pinG = 2;
11 const int pinDp = 9;
12
13 void seg7_setup( void )
14 {
15   pinMode(pinA, OUTPUT);
16   pinMode(pinB, OUTPUT);
17   pinMode(pinC, OUTPUT);
18   pinMode(pinD, OUTPUT);
19   pinMode(pinE, OUTPUT);
20   pinMode(pinF, OUTPUT);
21   pinMode(pinG, OUTPUT);
22   pinMode(pinDp, OUTPUT);
23 }
24
25 void seg7_print( int i )
26 {
27   switch( i ) {
28     case 0:
29       digitalWrite(pinA, HIGH);
30       digitalWrite(pinB, HIGH);
31       digitalWrite(pinC, HIGH);
32       digitalWrite(pinD, HIGH);
33       digitalWrite(pinE, HIGH);
34       digitalWrite(pinF, HIGH);
35       digitalWrite(pinG, LOW);
36       digitalWrite(pinDp, LOW);
37       break;
38
39     case 1:
40       digitalWrite(pinA, LOW);
41       digitalWrite(pinB, HIGH);
42       digitalWrite(pinC, HIGH);
43       digitalWrite(pinD, LOW);
44       digitalWrite(pinE, LOW);
45       digitalWrite(pinF, LOW);
46       digitalWrite(pinG, LOW);
47       digitalWrite(pinDp, LOW);
48       break;
49
50     case 8:
51       digitalWrite(pinA, HIGH);
52       digitalWrite(pinB, HIGH);
53       digitalWrite(pinC, HIGH);
54       digitalWrite(pinD, HIGH);
55       digitalWrite(pinE, HIGH);
56       digitalWrite(pinF, HIGH);
57       digitalWrite(pinG, HIGH);
58       digitalWrite(pinDp, LOW);
59       break;
60
61     case 9:
62       digitalWrite(pinA, HIGH);
```

```
63          digitalWrite(pinB, HIGH);
64          digitalWrite(pinC, HIGH);
65          digitalWrite(pinD, HIGH);
66          digitalWrite(pinE, LOW);
67          digitalWrite(pinF, HIGH);
68          digitalWrite(pinG, HIGH);
69          digitalWrite(pinDp, LOW);
70          break;
71
72        default:
73          digitalWrite(pinA, LOW);
74          digitalWrite(pinB, LOW);
75          digitalWrite(pinC, LOW);
76          digitalWrite(pinD, LOW);
77          digitalWrite(pinE, LOW);
78          digitalWrite(pinF, LOW);
79          digitalWrite(pinG, LOW);
80          digitalWrite(pinDp, HIGH);
81      }
82  }
```

Tagad komentāri. Atgriežamies pie faila Seg7C.ino pirmās rindiņas un faila seg7.cpp otrās rindiņas. Ja šo rindiņu nebūtu, tad mēģinot to kompilēt, tad nekas nesanāks. Kompilators kliegs, ka viņš nepazīst tādu simbolu seg7_setup() un seg7_print(). Atbilstoši programmēšanas valodas C tradīcijām tas ir tāpēc, ka funkcijas seg7_setup() un seg7_print() tiek definētas citā failā un kompilators tās pagaidām vēl neredz. Tāpēc, *pirms* pirmo reiz programmas tekstā sastopams seg7_setup(), seg7_print(), nepieciešams pateikt, ka tās ir funkcijas, kuras darbības rezultāta netiek atgriezta vērtība, pirmais nav parametru un otrai ir int tipa parametrs. Tas arī ir uzrakstīts failā seg7.h

Listing 5: seg7.h

```
1 void seg7_setup( void );
2 void seg7_print( int i );
```

Šī faila saturu var iekopēt faila Seg7C.ino sākumā, vai vienkārši ierakstīt preprocesora direktīvu #include ”seg7.h” un kompilators iekopēs to pats. Lai vēlāk izvairītos no paša radītām kļūdām, jau no paša sākuma radināsimies jau no paša sākuma #include ”seg7.h” tipa rindiņas ievietot arī tā faila sākumā, kur šīs funkcijas tiek uzrakstītas.

Ja aplūkojam faila seg7 sākumu, tad redzam tur vēl vienu preprocesora direktīvu #include <arduino.h>. Tajā ir aprakstītas gan funkcijas pinMode, digitalWrite, delay gan tas ko nozīmē LOW un HIGH, gan arī citas vajadzīgas lietas. Kāpēc tas nebija jādara iepriekš failā Seg7Func.ino? Droši vien tāpēc, ka Arduino IDE to failiem *.ino pievieno pati, jo bez šī faila nav iespējams uzrakstīt nevienu prātīgu Arduino projektu.

Atgriežoties pie #include. Ja faila vārdu ieliek šādi <fails.h>, tad atkal atbilstoši programmēšanas valodas C tradīcijām, tas tiek meklēts pie Arduino sistēmas failiem, bet ja ”fails.h”, tad pa priekšu projekta katalogā un, ja tur neatrod, tad pie Arduino sistēmas failiem.

Kā rīkojas Arduino IDE gadījumā, ja projekts satur vairākus failus? Kā jau varēja noprast no augstāk aprakstītās darba gaitas, ir strikti jāievēro sekojošas lietas

- Katram Arduino projektam ir savā katalogs! Kataloga nosaukums sakrīt ar projekta nosaukumu.
- Projekta galvenā faila vārdu projekta nosaukums plus faila paplašinājums .ino. Tas, pats par sevi saprotams, atrodas projekta katalogā.

Ja projekts sastāv no vairākiem failiem, tad atverot šo projektu, Arduino IDE atver arī visus projekta katalogā atrastos failus ar paplašinājumiem .cpp un .h kā tas šeit aprakstītā projekta gadījumā redzams 3 zīmējumā. Kompilējot tiek apstrādāts gan .ino fails gan *visi* .cpp un pēc tam viss tiek salikts kopā.

Pateicības

Platīte izgatavota Latvijas Universitātes Fonda projekta "Programmējam ar prieku" ietvaros un to finansē AS Mikrotīkls.

Literatūras saraksts

[1] Blink. https://www.arduino.cc/en/tutorial/blink. [Online; accessed 2018.09.24].

[2] Arduino Reference. pinMode(). https://www.arduino.cc/en/Reference/PinMode. [Online; accessed 2016.09.18].

[3] Arduino Reference. digitalWrite(). https://www.arduino.cc/en/Reference/DigitalWrite. [Online; accessed 2016.09.18].

[4] Arduino Reference. delay(). https://www.arduino.cc/en/Reference/Delay. [Online; accessed 2016.09.18].

[5] Seven-segment display. https://en.wikipedia.org/wiki/Seven-segment_display. [Online; accessed 2016.10.29].

