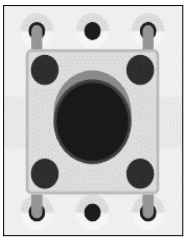
Arduino koolituse teine tund – nupu kasutamine

Õppematerjali koostajad: Indrek Karo ja Angela Leppik

Nupp on tore

Nupud on elektroonikas oma olemuselt peamiselt lülitid, mille eesmärgiks on kas katkestada või ühendada vooluringi (pilt nr. 18).



Pilt nr. 18

Nuppe on põhiliselt kahte sorti:

a) nupud, mis alla vajutades katkestavad vooluringi

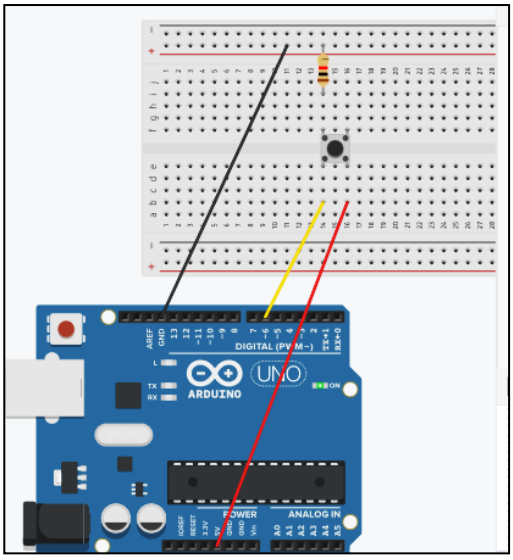
b) nupud, mis alla vajutades ühendavad vooluringi

On ka muid sorti nuppe. Näiteks selliseid, millel on rohkem ühendusi, aga neid ei ole mõistlik siin käsitleda.

Millist tüüpi nupuga tegemist on, saab kontrollida nupu spetsifikatsioonist (tavaliselt on seerianumbrit vaadates võimalik see leida internetist) või siis katsetades. Katsetamiseks on vajalik ehitada skeem ning vaadata, mis juhtub signaaliga siis, kui nupp alla vajutatakse. Kas elekter liigub vooluringis või mitte. Skeemi näite leiame pildilt nr. 19.

Selle skeemi jaoks saab koostada programmi, mis võimaldab kontrollida, kas paigutatud nupp ühendab või katkestab vooluringi. Selleks on aga vaja enne teada, kuidas saada tagasisidet Arduinos toimuva kohta. Selle jaoks kasutame jadaliidest.

Koolituse läbiviimist toetas Eesti Teadushuvihariduse Liit ja Haridus- ja Teadusministeerium



Pilt nr. 19

Jadaliides ja jadapordi monitor

Jadaliides on liides arvuti ja mingi seadme vahel andmete vahetamiseks. Antud liidese puhul saadetakse andmeid sisse ja välja ühe biti kaupa ehk siis järjest. Jadaliideseid on mitut sorti, nii USB kui ka näiteks FireWire. Arduino jadaliidese erisus on selles, et see ühildub RS 232 standardiga, mis näeb välja nagu VGA pistik, omades üheksat väljaviiku. Simulaatorites on jadaliides virtuaalne, mis siis tähendab, et tegemist on standardse jadaliidese virtuaalse järgi tegemisega.

Arduino puhul kasutatakse jadaliidest tihtipeale mingi anduri tulemuste näitamiseks arvutis, et saaks teha programmi muudatusi. Samuti saab jadaliidest kasutada Arduinole erinevate käskude saatmiseks, kui need on programmis kirjas, kuid sellest täpsemalt juba hiljem.

Jadaliidest kasutatakse Arduino puhul läbi jadapordi monitori (Serial Monitor), mis jälgib andmete liikumist ning näitab muutusi. Tavaliselt on ühenduse kiiruseks 9600 bitti sekundis.

Jadaliidese puhul tuleb sellest Arduinole alati programmi alguses funktsioonis „void setup ()“ teada anda käsuga: Serial.begin (9600); – see käsk teatab Arduinole, et nüüd hakatakse andmeid vahetama kiirusega 9600 bitti sekundis.

Et sellest käsust kasu oleks, tuleb Arduinole teada anda, kust kohast andmeid saadakse ning kuidas neid välja kirjutada. Selle jaoks on funktsioonis „void loop ()“ vaja kasutada käsku:

Serial.print – tulemus väljastatakse ühes reas, kuid seda on ebamugav lugeda

või siis käsku:

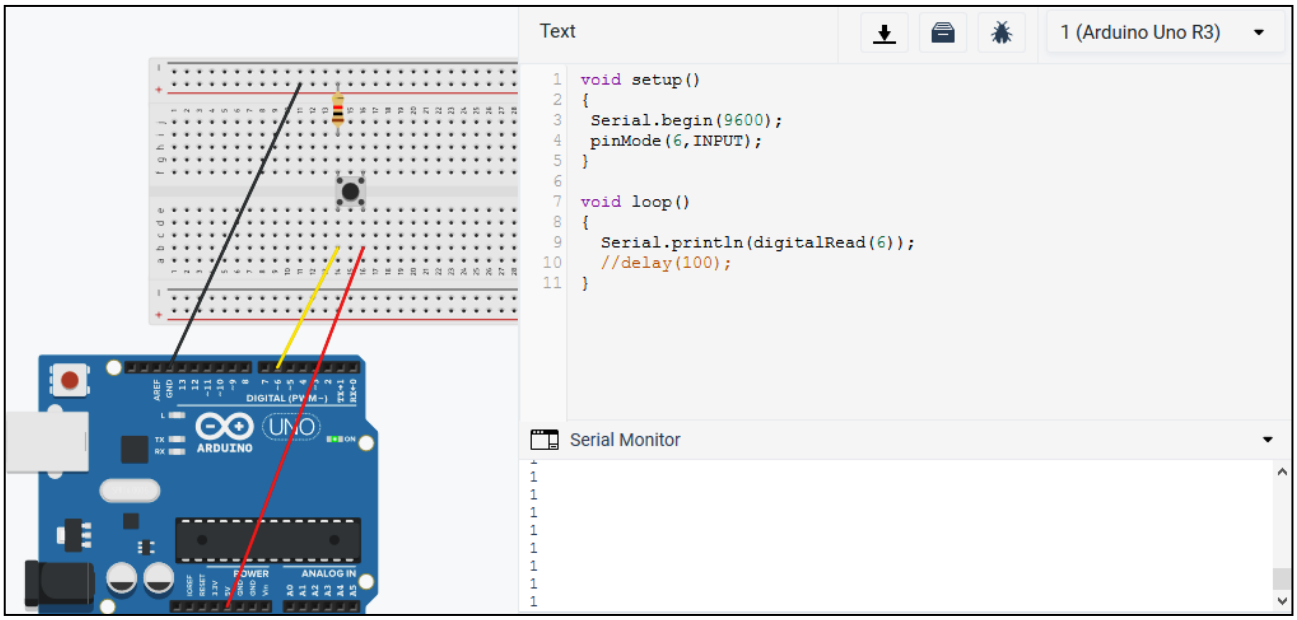
Serial.println – tulemus väljastatakse alati uuel real ning seda on lihtsam lugeda.

Seega kui me tahaksime teha Tinkercadis programmi, mis kontrollib kas pesasse nr. 6 ühendatud nupu vajutamine katkestab või hoopis ühendab vooluringi, tuleks meil teha järgmine programm.

```
 void setup ()
 {
        Serial.begin (9600);
        pinMode (6, INPUT);
 }
 void loop()
 {
        Serial.println (digitalRead(6));
 }
```

See programm väljastab tulemusena kas „1“ või „0“. Kui on „1“, siis vooluring on ühendatud ning kui on „0“, siis vooluring ei ole ühendatud. Programm koos skeemiga on ära toodud pildil nr. 20. Kui programm on koostatud ning simulatsioon tööle pandud, siis on vaja vajutada koodi all ekraani alaosas olevale pildikesele „Serial Monitor“. Seejärel ilmub ekraani alaossa aken, kus on vastavalt nupu vajutusele kas „1“ või „0“ ning see muutub pidevalt. Mõnikord lisatakse programmi korduse osasse ka viiteaeg käsuga „delay“, kuna siis on kergem jälgida.

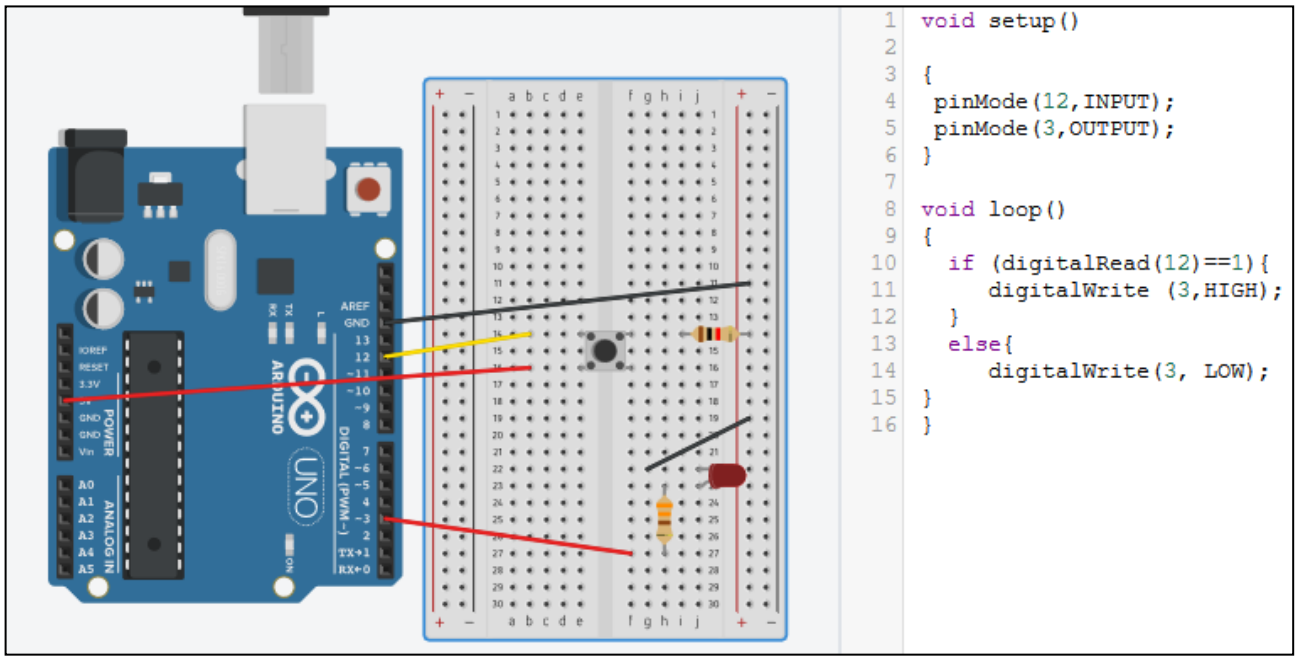
Antud skeemil on nupu takisti väärtuseks 1 kilo-oom.



Pilt nr. 20

Signaali lugemine

Signaali saab lugeda jadapordi monitorist ning kui see on teada, siis saame saadud andmeid kasutada ka muude programmide tegemisel. Näiteks on nüüd teada, et nupu vajutamisel on signaal 1 ehk vooluring on ühendatud ning see teadmine võimaldab teha nupuga lülitatava lambi, mille skeemi ja programmi on näha pildil nr. 21.



Pilt nr. 21

Pildil nr. 21 olev programm kasutab ära teadmist, et nupule vajutamise puhul ühendub vooluring (digitaalsignaal on „1“) ning seetõttu süttib valgus.

Seega on meil vaja programmi alguses Arduinole selgeks teha, milliseid väljaviike kasutada ning mida nendega peale hakata.

```
void setup ()
{
pinMode (12, INPUT);
pinMode (3, OUTPUT);
}
```

Antud programmis on nupp ühendatud väljaviiku 12 ning ootab sealt sissetulevat signaali ja saadab omakorda välja signaali pesast 3.

Et signaaliga midagi kasulikku peale hakata on mõistlik kasutada käsklust „if“, mida nimetatakse ka tingimuslauseks. Seda võib nimetada ka käsuks „kui“ ehk kui juhtub mingi sündmus (näiteks keegi vajutab nuppu ja sealt tuleb välja signaal „1“), siis Arduino lülitab lambi sisse. Kui seda sündmust ei juhtu, siis lampi sisse ei lülitata.

```
void loop ()
{
if (digitalRead (12)= =1) {
  digitalWrite (3, HIGH);
}
else {
  digitalWrite (3, LOW);
}
}
```

Koodi ridade selgitus.

if (digitalRead (12)==1){

if – pane tööle tingimus ning sellele peab sulgudes järgnema tingimus.

digitalRead (12) – see käsib lugeda signaali pesast 12.

„==1“ – kaks võrdusmärki käsivad võrrelda väärtusi, esimene võrdusmärk käsib võrdlema hakata, ning teine seab tingimuse (hetkel peab olema võrdne ühega).

„{“ – sellega algab tegevus, mis toimub ainult siis, kui tingimus on täidetud.

digitalWrite (3, HIGH); – näitab tegevust, mis toimub siis, kui tingimus on täidetud ehk lamp põleb.

„}“ – sellega lõpeb tegevus, mis toimub siis, kui tingimus on täidetud.

else { – sellega märgitakse ära tegevus, mis toimub siis, kui tingimus ei ole täidetud.

digitalWrite (3, LOW); – näitab tegevust, mis toimub siis, kui tingimus ei ole täidetud ehk lamp ei põle.

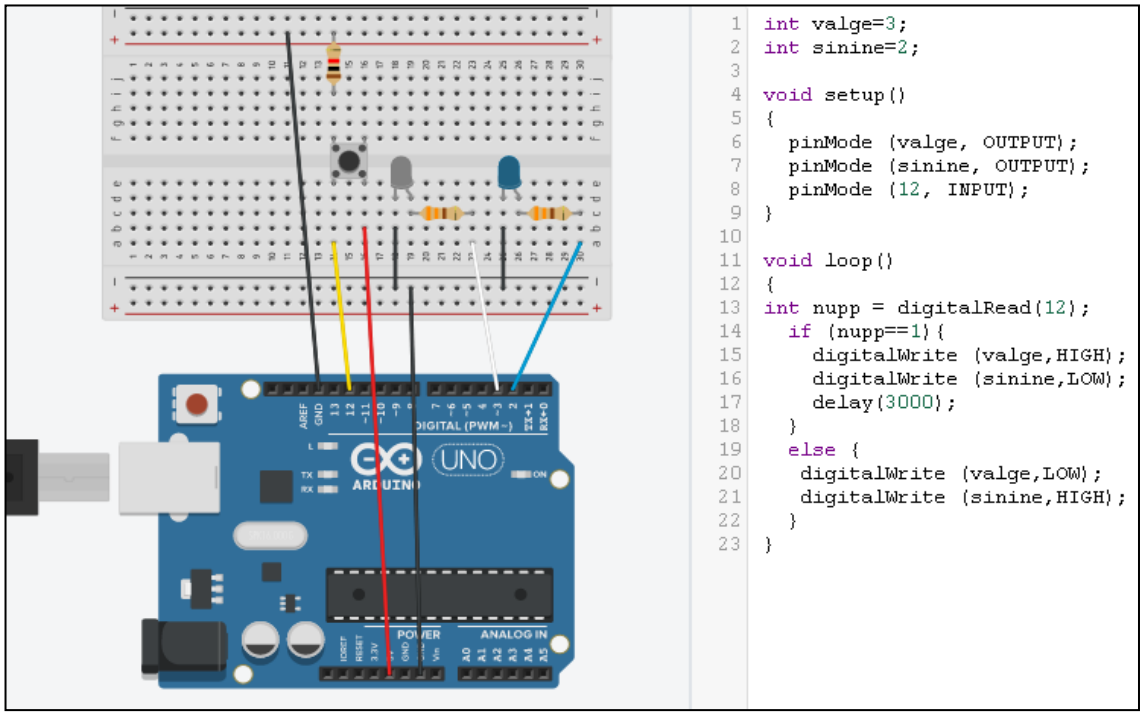
„}“ – sellega lõpeb ära tegevus, mis toimub siis, kui tingimust ei ole täidetud.

„}“ – ära tuleb lõpetada ka kordus, kasutades märki „}“, sest muidu programm tööle ei hakka.

Korduse lõpu unustamine on üks levinumaid vigu, mida õpilased tingimuslauseid ja lisafunktsioone luues teevad, seega on mõistlik sellele tähelepanu pöörata.

Tingimuslauset „if“ saab kasutada näiteks ka ühe lambi kustutamiseks ning teise põlema panemiseks. Selle katsetamiseks on pildil nr. 22 olev skeem ja programmi kood. Antud kood on natuke keerulisem, sest kasutusele on võetud ka „integer“ tüüpi globaalsed muutujad. Muutujaid saab jaotada mitut moodi, kas nende tüübi järgi (sellest räägime hiljem juba põhjalikumalt) või näiteks nende asukoha järgi programmis – globaalsed ja lokaalsed muutujad. Globaalsed muutujad on sellised, mis on kasutuses kogu programmis, nad kirjeldatakse ära kohe programmi alguses. Häda on siin selles, et kuna nad kirjeldatakse ära kohe programmi alguses, siis nad vajavad osa Arduino mälust kogu programmi töötamise ajal, mis omakorda võib teha programmi aeglasemaks. Samas kui programm on lihte ning muutujaid ei ole palju, siis erilist mõju nad oma tüübi tõttu programmi tegevusele ei oma. Näitena on pildil nr. 22 olevas koodis muutujad „valge“ ja „sinine“. Lokaalsed muutujad on aga need muutujad, mis on programmis kindla funktsiooni sees ning kasutatakse ainult siis, kui funktsioon parajasti töötab ning selle näiteks on muutuja „nupp“.

Tihtipeale kasutatakse muutujaid selleks, et programmi kirjutades mitte ära unustada, millisesse pessa on mingi kindel seade ühendatud (kui näiteks on muutuja nimega „valge“, siis sinna pessa on ühendatud valge lamp).



```
int valge=3;
int sinine=2;

void setup()
{
  pinMode (valge, OUTPUT);
  pinMode (sinine, OUTPUT);
  pinMode (12, INPUT);
}

void loop()
{
int nupp = digitalRead(12);
  if (nupp==1){
    digitalWrite (valge,HIGH);
    digitalWrite (sinine,LOW);
    delay(3000);
  }
  else {
   digitalWrite (valge,LOW);
   digitalWrite (sinine,HIGH);
  }
}
```

Pilt nr. 22

Pildil nr. 22 on ära toodud kahe lambiga skeem, kus algselt põleb sinine lamp ning nupule vajutades see kustub ja süttib valge lamp. Programmis on kasutatud nii globaalseid kui ka lokaalseid

muutujaid, et näidata ära nende erinevus. Järgnevalt on toodud programmi koodi kirjeldus.

```
int valge = 3;
int sinine = 2;
```

Need on globaalsed muutujad, mis toimivad kogu programmi jooksul. Tähistavad siis vastavalt väljaviike, kuhu on ühendatud valge ja sinine diood.

```
void setup ()
{
pinMode (valge, OUTPUT);
pinMode (sinine, OUTPUT);
pinMode (12, INPUT);
}
```

Nüüd tutvustame Arduinole sinist ja valget dioodi, millest peab välja saatma voolu (OUTPUT) ning nuppu, mis antud näites paikneb pesas 12 ning sellelt oodatakse sisendsignaali (INPUT).

void loop () – see on korduse funktsioon.

„{„ – sellega algab programmi osa, mis hakkab korduma.

int nupp = digitalRead (12); – siin tutvustatakse lokaalset muutujat „nupp“, milleks on pesast 12 tulev digitaalsignaal.

if (nupp==1) { – siin on tegemist tingimuslausega, kus võrreldakse muutujast nupp tulevat signaali nagu varasemas programmiski mainitud ning märgiga „{„ algab tegevus, mis toimub ainult siis, kui keegi on nupu alla vajutanud (on signaal „1“).

```
digitalWrite (valge, HIGH);
digitalWrite (sinine, LOW);
delay (3000);
```

Siin hakkab valge diood valgust kiirgama ning sinine kustub. Enne kui saab uuesti nuppu vajutada, peab ootama kolm sekundit. Viiteaeg on lisatud selleks, et oleks lihtsam märgata, kas nupule vajutamisest oli kasu lambi süttimisel või mitte, sest mõnikord on Tinkercad aeglane ning ei ole

kohe märgata lambi süttimist.

„**}**“ – lõpeb „if“ tingimuslause (see on ka midagi mida kiputakse unustama).

else { – sellega märgitakse ära tegevus, mis toimub siis, kui nuppu ei ole vajutatud ning see peab samuti olema „{„ja„}“ märkide vahel.

digitalWrite (valge, LOW);
digitalWrite (sinine, HIGH);

Siin toimub vastupidine tegevus, valge diood ei kiirga valgust kuid sinine kiirgab.

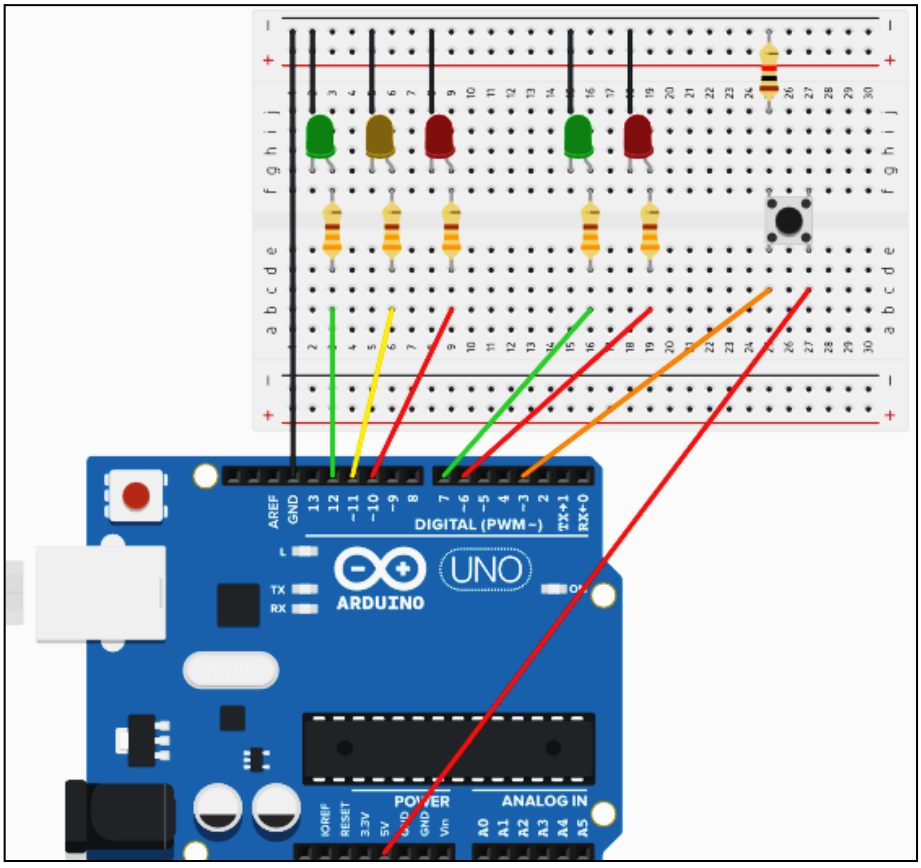
„}“ – see lõpetab ära funktsiooni „else“ ning kogu kordus lõpeb sama märgiga.

„}“ – see on korduse viimane osa, mis lõpetab korduses oleva tegevuse (seejärel hakkab kordus uuesti algusest peale).

Kasutades teadmisi nuppudest, dioodidest ning „for“ tsüklist on võimalik luua nupuga töötav jalakäijate valgusfoor. Seal saab kasutada „if“ lauset nupu jaoks ning „for“ tsüklit rohelise tule vilgutamiseks. Antud süsteem on ära toodud pildil nr. 23. Selle skeemi jaoks on vajalik programm ära toodud kohe peale pilti nr. 23.
Programmi kommenteeritud näidiskoodis on kasutatud globaalseid muutujaid, et programmeerijal ei ununeks ära millistesse väljaviikudesse on dioodid ühendatud.

int r_auto = 12;
int k_auto = 11;
int p_auto = 10;
int r_jala = 7;
int p_jala = 6;

Samuti on roheliste tulede vilgutamiseks kasutatud „for“ käsklust ning nupu vajutamise tuvastamiseks „if“ lauset, mille kohta on täpsem teave varasemalt ära toodud.



Pilt nr. 23

Valgusfoori töölepaneku kood

```
int r_auto = 12;        //autode rohelise tule väljaviik on 12
int k_auto = 11;        //autode kollase tule väljaviik on 11
int p_auto = 10;        //autode punase tule väljaviik on 10
int r_jala = 7;         //jalakäijate rohelise tule väljaviik on 7
int p_jala = 6;         //jalakäijate punase tule väljaviik on 6

void setup ()           //siin teatatakse mida väljaviikudega peale hakata, kas saata signaal välja
                        (OUTPUT) või võtta signaal vastu (INPUT)
{
 pinMode (r_auto, OUTPUT);
 pinMode (k_auto, OUTPUT);
 pinMode (p_auto, OUTPUT);
 pinMode (r_jala, OUTPUT);
```

```
pinMode (p_jala, OUTPUT);
pinMode (3, INPUT);
}

void loop ()                    //siin hakkab programmi korduse osa
{
 int nupp = digitalRead (3);    //tutvustatakse lokaalset muutujat „nupp“, milleks on pesast 3
                                loetav digitaalsignaal
 if (nupp==1){                  //see on „if“ lause, mis käsib võrrelda (üks „=“ märk)
                                muutujat „nupp“ numbriga „1“, kui see on võrdne (kaks „=“
                                märki), siis tuleb teha tegevus, mis on märkide „{„ ja „}“ vahel
 for (int i = 0; i <= 3; i++) { //siin tekitatakse „for“ tsükkel, kus märkide „{„ ja „}“ vahel
                                toimuv tegevus toimub nii mitu korda, kui palju muutub „i“
                                väärtus (muutuja „i“ väärtus algab hetkel 0-iga ning lõpeb
                                3-ga), muutudes sammuga 1 (see on kirjas väljendis „i++“)
 delay (500);
 digitalWrite (r_auto, LOW);
 delay (500);                   //tule vilgutamise kood poolesekundilise vahemikuga
 digitalWrite (r_auto, HIGH);
 }                              //lõpeb „for“ tsükkel autode rohelise tule vilgutamiseks
 digitalWrite (r_auto, LOW);    //autode roheline tuli lülitatakse välja
 digitalWrite (k_auto, HIGH);   //süttib autode kollane tuli
 delay (1000);                  //viiteaeg 1 sekund ning põleb kollane tuli
 digitalWrite (k_auto, LOW);    //autode kollane tuli kustub
 digitalWrite (p_auto, HIGH);   //süttib autode punane tuli
 digitalWrite (r_jala, HIGH);   //süttib jalakäijate roheline tuli
 delay (8000);                  //viiteaeg 8 sekundit, autodele põleb punane ning jalakäijatele
                                roheline
 for (int i = 0; i <= 3; i++) { //hakkab jällegi „for“ tsükkel, seekord jalakäijate rohelise tule
 delay (500);                   vilgutamiseks
 digitalWrite (r_jala, LOW);
 delay (500);                   //jalakäijate roheline tuli vilgub 3 korda, poolesekundilise
```

```
digitalWrite (r_jala, HIGH);        vahega
}                                   //lõpeb „for“ tsükkel jalakäijate rohelise tule vilgutamiseks
digitalWrite (r_jala, LOW);         //jalakäijate roheline tuli kustub
digitalWrite (p_auto, LOW);         //autode punane tuli kustub
digitalWrite (k_auto, HIGH);        //autode kollane tuli süttib
delay (1000);                       //viiteaeg 1 sekund, autode kollane tuli põleb samal ajal
digitalWrite (k_auto, LOW);         //autode kollane tuli kustub
}                                   //„if“ lause lõpp
digitalWrite (r_auto, HIGH);        //autodele süttib roheline tuli
digitalWrite (p_jala, HIGH);        //jalakäijatele süttib punane tuli
}                                   //lõpeb korduses olev tegevus (void loop) ning kordus algab
                                    otsast peale
```