# Arduino koolituse kuues tund – map käsk, mootor ja jadamonitor, heliefektid, temperatuuriandur

Õppematerjali koostajad: Indrek Karo ja Angela Leppik

**Map – käsk** (lambiga, mootoriga) potentsiomeetriga.

„Map“ käsku kasutatakse ühe väärtuste vahemiku muutmiseks teiseks. Tihti kasutatakse seda analoogsignaali muutmiseks sellisele kujule, et saaks kasutada pulsilaiusmodulatsooni (saata signaali välja pesast, kus ees on “~” märk) ning Arduino andmetes on märgitud, et tegemist on PWM pesaga. „Map“ käsku kasutatakse ka muudel juhtudel, kui on vaja muuta ühte vahemikku teiseks.

„Map“ käsu probleemiks on see, et sellega ei saa teha tehteid murdarvudega ning kui selline tegevus vajalikuks osutub siis tuleb ikkagi kasutada matemaatilisi funktsioone.

„Map“ käsu loogika näeb välja järgmine:

map (muutuja, algne väikseim, algne suurim, uus väikseim, uus suurim).

Mis lahti seletatult tähendab järgmist:

map – hakkame kasutama “map” käsku;

() – sulgude sees on muutuja ja need muutuja väärtused, mida muutma hakatakse;

muutuja – see on muutuja nimi, tavaliselt on selleks mingist pesast tulev signaal;

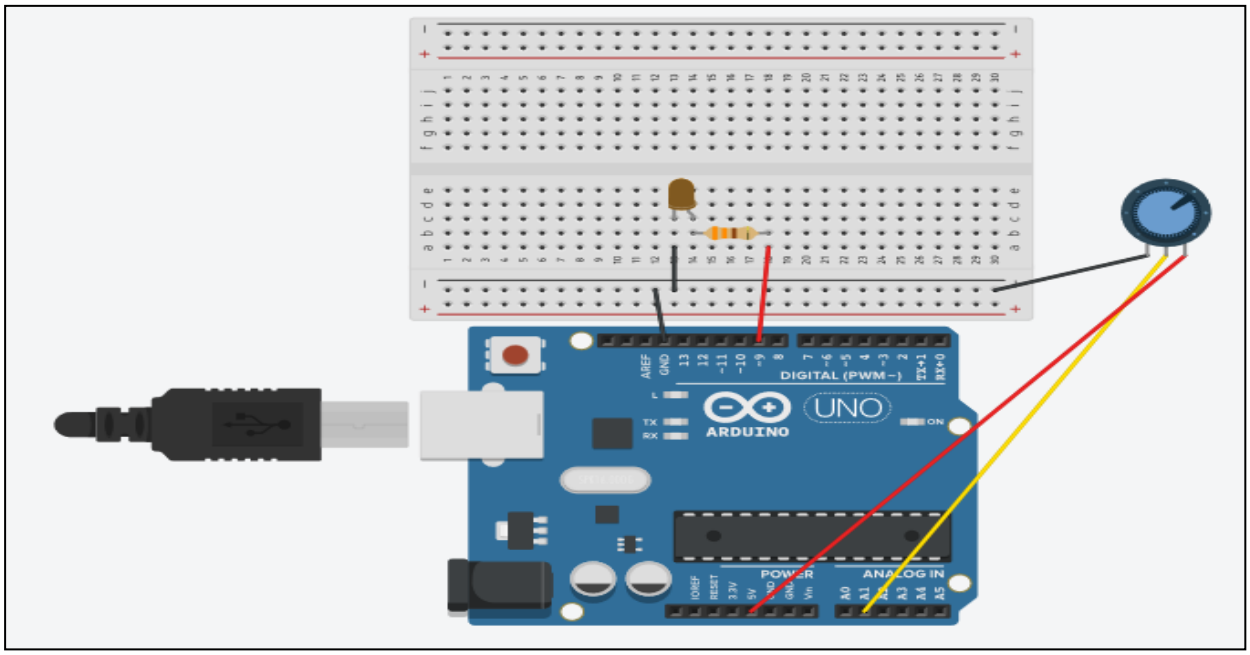
algne väikseim – see on muutuja väärtustest algne väikseim väärtus;

algne suurim – see on muutuja väärtustest algne suurim väärtus;

uus väikseim – see on nüüd uus väikseim väärtus, mis muutujal olla saab (see on tavaliselt välja minevaks signaaliks, mida kas töödeldakse edasi või näidatakse);

uus suurim – see on muutuja uus suurim võimalik väärtus. Tavaliselt on nii, et väikseim väärtus jäätakse „map“ käsu puhul samaks ning muutub ainult uus suurim võimalik väärtus.

„Map“ käsu kasutamise näitena on hea vaadata pildil nr. 39 olevat skeemi, kus potentsiomeetri abiga saab muuta LED näivat heledust.




Pilt nr. 39

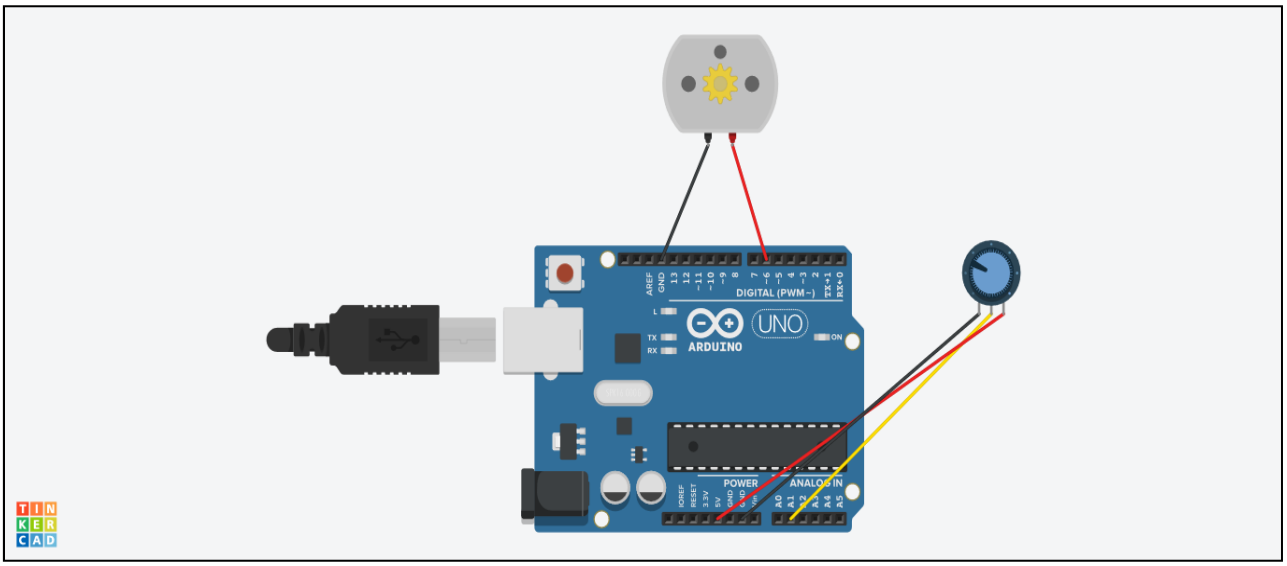
Pildil nr. 39 on LED lamp ühendatud pesasse 9, mis võimaldab kasutada pulsilaiusmodulatsiooni ning potentsiomeeter pesasse A1, et sealt saaks lugeda analoogsignaali. Kui kasutada käsku “map”, siis programm koos kommentaaridega peaks olema järgmine:

```
void setup ()
{
  pinMode (9, OUTPUT);      //signaal väljub pesast 9
  pinMode (A1, INPUT);      //loetakse signaali pesast A1
}
void loop ()
{
  int valgus = analogRead (A1);          //luuakse muutuja “valgus” milleks on pesast A1 loetav
                                         analoogsignaal
  valgus = map (valgus, 0, 1023, 0, 255); //muudetakse vahemik 0 kuni 1023 vahemikuks 0 kuni
                                         255 andes sellega muutujale “valgus” uued väärtused
  analogWrite (9, valgus);               //kirjutatakse välja muutuja “valgus” uus väärtus
                                         pesasse 9
}
```

Selline on üks lihtsamaid näiteid „map“ käsu kasutamiseks. Samas saab kasutada seda ka muudel juhtudel ning paljude erinevate väärtuste puhul. Samuti ei ole “map” käsu puhul probleemiks negatiivsed väärtused, kuna antud käsu puhul on tegemist mitmete lihtsamate matemaatiliste tehete koondamisega ühte funktsiooni, et kasutamise keerukust vähendada. Suurema huvi korral saab selle käsu kohta lugeda kodulehele www.arduino.cc/reference/, kus on pikemalt kirjutatud erinevate funktsioonide omaduste kohta, aga kahjuks mitte eesti keeles.

**Mootor ja jadamonitor**

„Map“ käsu toimimist saab jälgida ka jadamonitoris ning selleks on sobilik pildil nr. 40 näidatud skeem, kus on ühendatud Arduinoga nii mootor kui ka potentsiomeeter ning jadapordi monitorist on võimalik vaadata andmete muutumist.




Pilt nr. 40

Pildil nr. 40 näidatud skeemil on sarnane loogika eelnevale pildile, kus potentsiomeeter on ühendatud pesasse A1, aga mootor on ühendatud pesasse 6, mis samuti võimaldab pulsilaiusmodulatsiooni kasutamist. Järgnevalt on toodud pildil oleva programmi loogika.

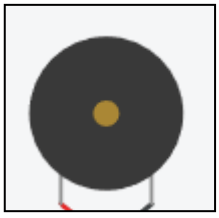
```
void setup ()
{
  pinMode (6, OUTPUT);      //kasutab mootori väljundina pesa 6
  pinMode (A1, INPUT);      //kasutab sisendina pesa A1
  Serial.begin (9600);
```

```
}

void loop ()
{
  int kiirus = analogRead (A1);                  //tekitab muutuja “kiirus”, milleks on pesast A1 loetav analoogsignaal
  kiirus = map (kiirus, 0, 1023, 0, 255);        //kasutab käsklust “map” muutuja “kiirus” väärtuste muutmiseks vahemikust 0-1023 vahemikku 0-255 (põhimõtteliselt jagab kõik väärtused neljaga)
  analogWrite (6, kiirus);                       //väljastab muudetud kiiruse väärtuse pesast 6
  Serial.print (analogRead (A1));                //kirjutab jadapordi monitori potentsiomeetrist loetud väärtuse, aga jääb samale reale
  Serial.print ("\t");                           //tekitab jadapordi monitoris eelneva väärtuse järgi tühiku
  Serial.println (kiirus);                       //kirjutab jadapordi monitori muutuja “kiirus” uue väärtuse peale “map” käsu kasutamist ning läheb järgmisele reale
```

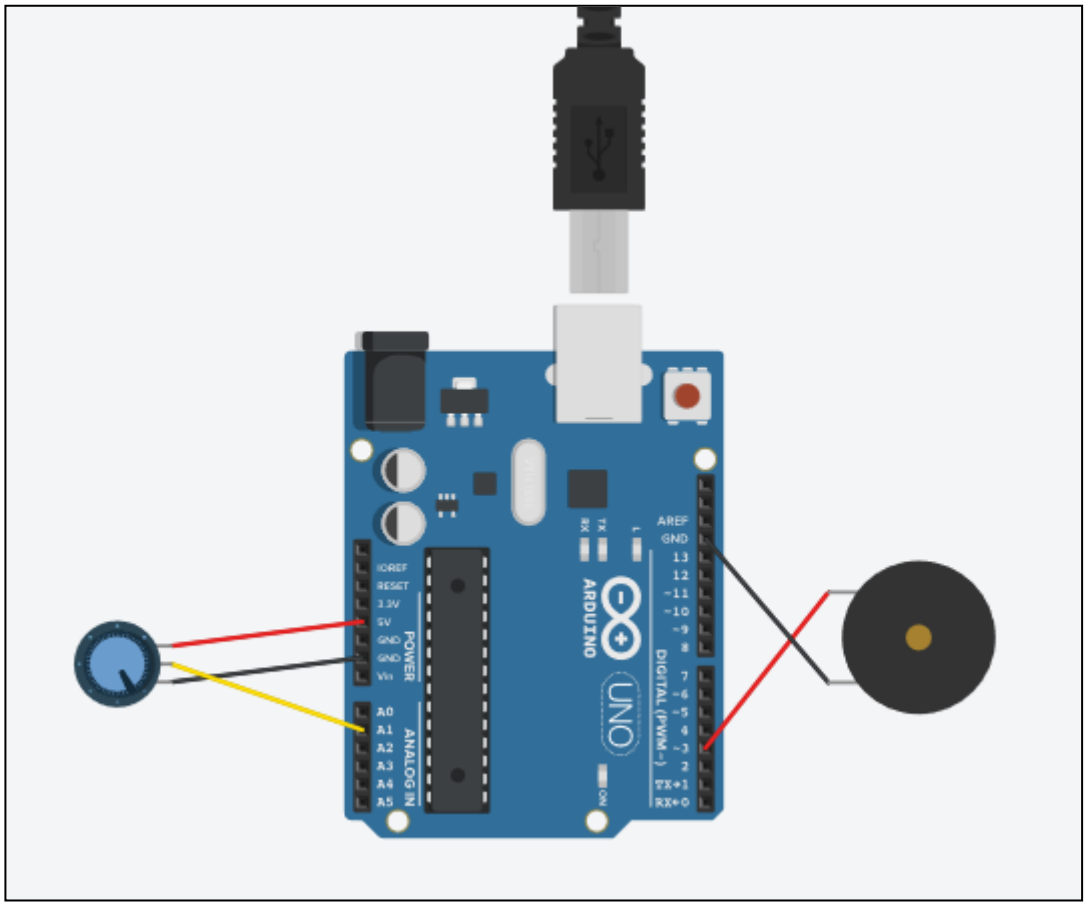
**Heliefektid**

Arduino puhul saab helisid tekitada mitmel viisil, kuid kõige levinum neist on sumisti, mis meenutab oma olemuselt üpris niru kõlarit, millega saab erinevaid piikse teha. Tinkercadis ning ka füüsilistel Arduinodel on sumistiks tavaliselt piesoelektriline sumisti (pilt nr. 41), mis põhimõtteliselt on materjal, mille kuju muutub elektrivoolu toimel. Seade ise toimib nii, et piesoelektriline materjal on ühendatud õhukesele metallplaadile ning siis saab voolu abil väga kiiresti metallplaadikest painutada, mis omakorda tekitab häält. Mida kiiremini metallplaati painutada, seda kõrgem heli (peenem hääl) tekib.



Pilt nr. 41

Seda seadet annab kontrollida mitmel erineval viisil. Tavaliselt ühendatakse sumisti pulsilaiusmodulatsiooni võimaldavasse pesasse, sest siis saab tekitatava heli sagedust muuta ning erinevaid toone tekitada. Sellisel juhul on võimalus, kas ise valida tekitatava heli sagedus, seda programmi käsitsi sisestades või saab kasutada ka näiteks potentsiomeetrit koos käsuga “map”, mis võimaldab kasutada loetud analoogsignaali heli sageduse muutmiseks (ehk toonide mängimiseks). Viimase kohta leiame skeemi pildilt nr. 42. Kus potentsiomeeter on analoogsignaali saamiseks ühendatud väljaviiku A1 ning sumisti on ühendatud väljaviiku 3, kuna see võimaldab pulsilaiusmodulatsiooni kasutamist (ees on “~” märk). Peale pilti on ka programm koos kommentaaridega.




Pilt nr. 42

Käsk nimega “tone” on mingi kindla sagedusega heli esitamiseks kasutades sumistit. Selle käsuga saab ära märkida ka kui kaua mingi heli kestab. Kui seda aga ei tehta, siis Arduino vait ei jää.
Heli lõpetamiseks kasutatakse kas käsku noTone või pannakse heli tekitamisele mingi piirang (näiteks, heli tekib ainult siis, kui keegi vajutab nuppu). Korraga saab Arduino tekitada vaid ühte

tooni. Tavaliselt kasutatakse käsku tone () PWM tüüpi pesades või siis analoogpesades, sest seal saab sagedust muuta. Arduinoga ei saa käsuga „tone“ madalamaid sagedusi kui 31 hertsi.

Käsul on kaks erinevat varianti:

a) tone (väljaviigu nr, sagedus) – näiteks tone (3, 238);

b) tone (väljaviigu nr, sagedus, kestus) – näiteks tone (3,459,55).

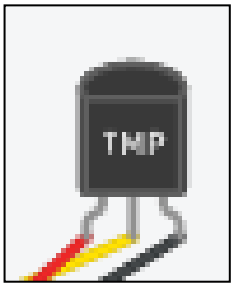
Pildil nr. 42 toodud skeemi puhul ongi kasutatud käsku „tone“ ning selle muutumist kontrollitakse potentsiomeetri abil.

```
void setup ()
{
  pinMode (3, OUTPUT);          //väljutab signaali
  pinMode (A1, INPUT);          //loeb signaali
}

void loop ()
{                               //korduse algus
  tone (3, (analogRead (A1)));  //kasutab käsku “tone” heli tekitamiseks
}                               //korduse lõpp
```

**Temperatuurianduri kasutamine**

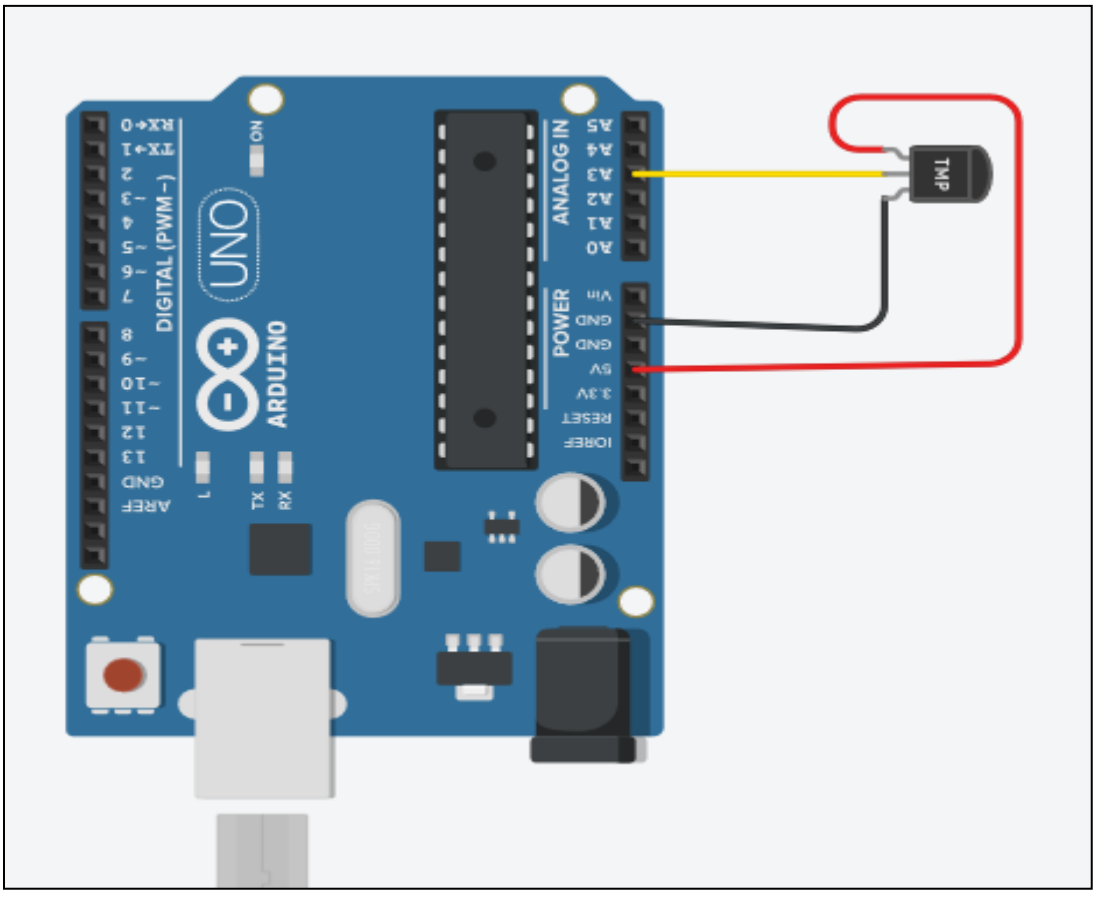
Temperatuuriandureid on mitmeid liike, kuid Tinkercadis on kasutusel ainult üks sort, mis on ära toodud pildil nr. 43. Tegemist on TMP 36 tüüpi temperatuurianduriga.




Pilt nr. 43

Erinevalt termistoridest ei põhine selle anduri töö temperatuuri tõttu muutuva elektritakistusega takistil, vaid hoopis dioodide omadusel, et temperatuuri muutudes muutub pinge sama tempoga. Selle anduri omapära seisneb ka selles, et väljundpinge on vahemikus 0,1 kuni maksimaalselt 2,0 V (tavaliselt kuni 1,75V), mistõttu ei saa analogRead käsuga pesadest lugeda vahemikku 0 kuni 1023, vaid hoopis 20 kuni 358. Et vältida segadust on mõistlik vaadata mitte analoogsignaali, vaid lasta Arduinol see kohe pingeks ümber arvutada nagu me näeme ka pildi nr. 44 juurde käivas programmis.

Temperatuurianduri katsetamiseks on mõistlik ära tuua nii anduri pinge, kui ka välja arvutatud temperatuur, sest siis on lihtsam aru saada anduri toimimisest. Pildi nr. 44 ongi temperatuurianduri andmete pistik ühendatud pesasse A3, et võtta vastu analoogsignaali ning vooluks tarvitatakse 5V pesa (saab ka kasutada 3,3V, kuid siis peab kasutama teistsugust valemit). Maandus on nagu ikka ühendatud maanduse pesas.




Pilt nr. 44

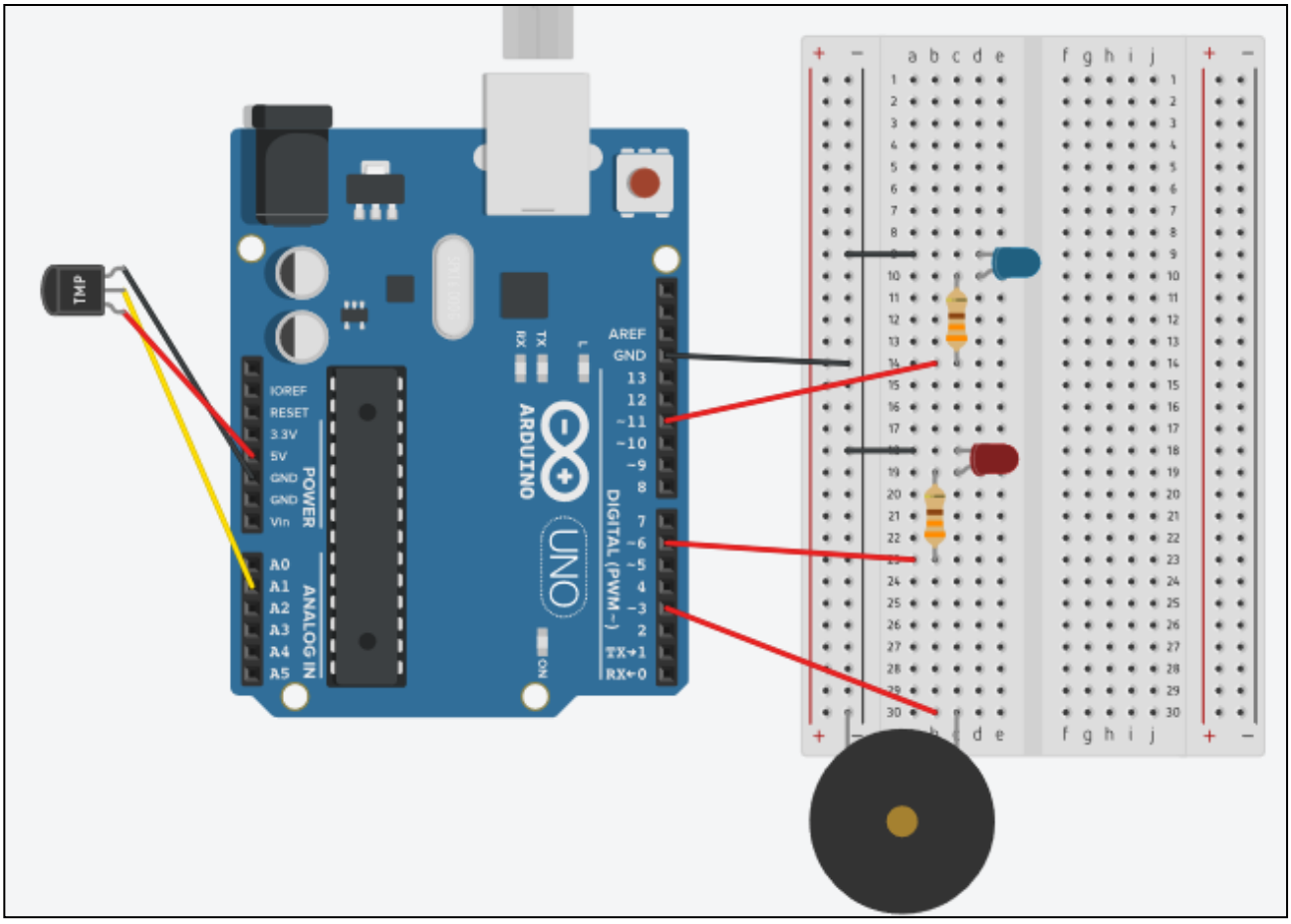
Pildil nr. 44 näidatud skeemi puhul on programmis kasutusele võetud „float" tüüpi muutuja, mille erisuseks on võimalus kasutada komakohtadega arve. „Float" muutuja tüüpi kasutatakse tihti analoogsignaalide puhul, sest seda on tihtipeale paindlikum kasutada.

Antud programm on lihtsalt näidiseks, sest temperatuuri saab välja arvutada ka muude meetoditega.

Kasutades näiteks valemit: //lihtne valem on selline: Temp °C = [(Pinge mV)-500]/10

```
//int voldid = map (analogRead (A1), 20, 358, 99, 1750);
  Serial.println ((voldid-500)/10);
int andur = A3;          //võtame kasutusele globaalse muutuja „andur", mis saab signaali pesast A3, globaalse seetõttu, et siis saab vajaduse korral teha mitu alamprogrammi, mis kasutab sama muutujat
void setup ()
{
Serial.begin (9600);   //pannakse vastava kiirusega tööle jadapordi monitor
}
void loop ()
{
int lugemine = (analogRead (andur));          //tekitatakse muutuja „lugemine", milleks on anduri analoogsignaal
float pinge = lugemine * 5.0;                 //tekitatakse „float" tüüpi muutuja, kasutades loetud singaali ning korrutades seda Arduino väljundpingega
 pinge / = 1024.0;                            //muutuja „pinge" jagatakse 1024 osaks, et seda saaks nii töödelda temperatuuri arvutamisel kui ka pinge muutuse näitamisel
 float temperatuur C = (pinge - 0.5) * 100;   //Arduino arvutab välja temperatuuri, kasutades selleks pinget ning selle muutust
 Serial.print (pinge); Serial.print ("volti");  //näitab jadapordi monitoris pinget ning kirjutab selle taha „volti"
 Serial.print ("\t");                         //teeb jadapordis reale tühiku sisse
 Serial.print (temperatuur C); Serial.println ("kraadi");   //näitab jadapordi monitoris temperatuuri ning kirjutab selle taha „volti"
  delay (1000);                               //tekitab 1 sekundise viiteaja
}                                             //korduse lõpp
```

Tihtipeale ei ole vaja kasutada temperatuuriandurit kraadiklaasina vaid saab lihtsalt kasutada loetavat signaali. Selle näiteks on pildil nr. 45 ära toodud temperatuurianduri ja sumisti kasutamine hoiatussignaali skeemi tegemiseks. Antud juhul on loodud skeem, mis annab hoiatussignaali ning paneb põleva sinise tule asemel vilkuma punase tule, kui temperatuur on liiga kõrge.




Pilt nr. 45

Järgnevalt on toodud pildil nr. 45 kasutatud skeemi programmi selgitus.

```
void setup ()
{
  pinMode (3, OUTPUT);          //sumisti väljundi tutvustus
  pinMode (11, OUTPUT);         //sinise lambi väljundi tutvustus
  pinMode (6, OUTPUT);          //punase lambi väljundi tutvustus
  pinMode (A1, INPUT);          //temperatuurianduri sisendi tutvustus
}
```

```
void loop ()
{
 if (analogRead (A1) <200)          //kui temperatuurianduri sisendväärtus ületab 200, siis hakkab
                                    toimuma järgnev
 {
        digitalWrite (11, LOW);     //sinine lamp on välja lülitatud
        tone (3, 458);              //teeb häält – 3 on sumisti pesa ning 458 on heli sagedus (Hz)
        digitalWrite (6, HIGH);     //süttib punane tuli
        delay (150);                //ootab 150 mikrosekundit
        noTone (3);                 //ei tee häält
        digitalWrite (6, LOW);      //punane tuli kustub
        delay (150);                //ootab 150 mikrosekundit
 }
 else                               //tegevus mis toimub siis, kui temperatuurianduri sisendväärtus
                                    on alla 200
 {
  digitalWrite (11, HIGH);          //põleb sinine lamp
 }
}
```