



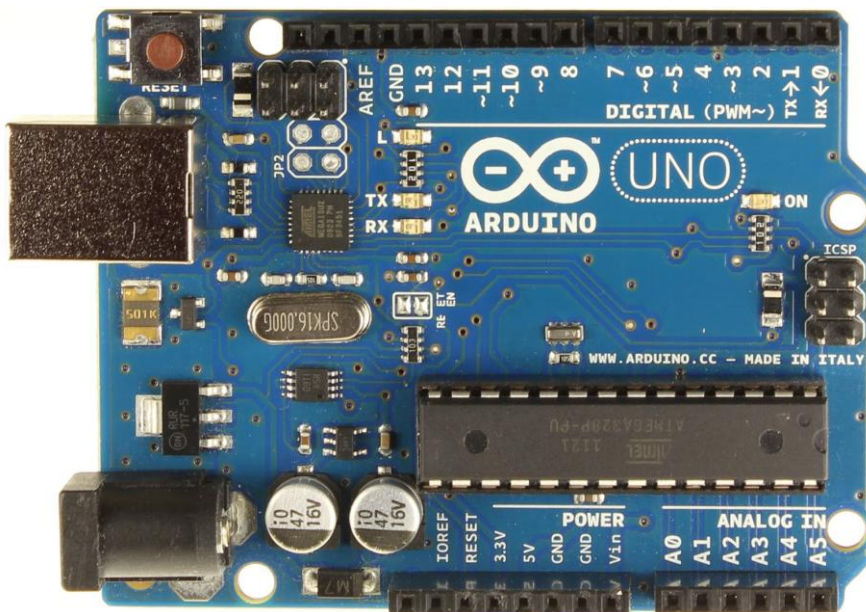
## Arduino mikrokontrollerite simuleerimine keskkonnas [www.tinkercad.com](http://www.tinkercad.com)

Õppematerjali koostajad: Indrek Karo ja Angela Leppik

Käesoleva õppematerjali eesmärgiks on tutvustada Arduino mikrokontrolleri kasutamist virtuaalsel kujul [www.tinkercad.com](http://www.tinkercad.com) keskkonnas. Materjal on ette nähtud lihtsustamaks interneti teel läbi viidavat koolitust ning aitamaks juhendmaterjalina. Antud abivahend ei ole otseselt suunatud füüsiliselt Arduino mikrokontrollerite abil automaatikaseadmete loomise abistamiseks, aga kasu võib sellest ikka olla. Viimase kohta on loodud mitmeid muid põhjalikke materjale. Piltidena on kasutatud peamiselt Tinkercadi keskkonna pilte, et oleks lihtne võrrelda ja ise tegutseda.

### Esimene tund – Arduino lühiülevaade

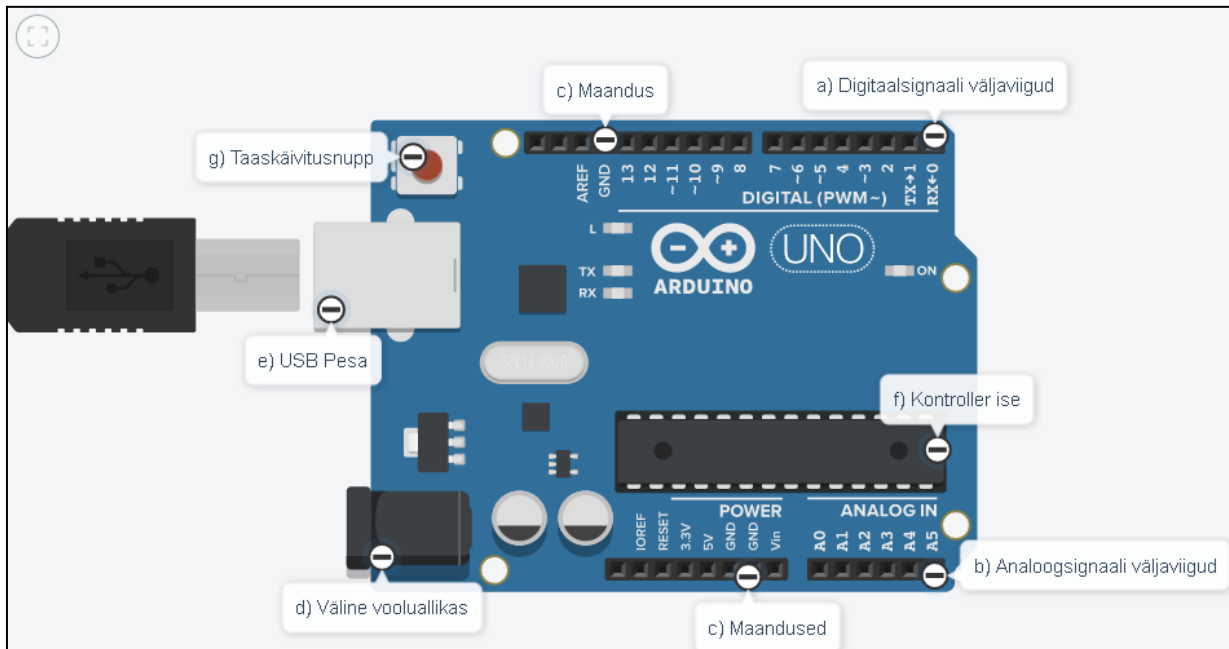
Arduino on lihtsalt kasutatav mikrokontrolleriga plaat, millele on võimalik ühendada erinevaid andureid ja muid seadmeid. Arduino plaate on mitut tüüpi ning alati on mõistlik valida enda vajadustele vastav. Samuti on Arduinol mitmeid erinevaid kloone, mis on tavaliselt soodsama hinnaga ning tihtipeale on mõningad osad ka erinevad. Täpsemalt on Arduino mikrokontrollerite omaduste, ajaloo ja võltsingute kohta kirjutatud Alo Peets (link: [http://kodu.ut.ee/~alop/Arduino\\_Robot.pdf](http://kodu.ut.ee/~alop/Arduino_Robot.pdf)). Kõige levinum Arduino mikrokontroller, peamiselt oma vastupidavuse ning lihtsa kasutuse tõttu on all oleval pildil nr. 1 nähtav Arduino Uno.



Pilt nr. 1

## Arduino UNO peamised komponendid ja ühenduskohad

Arduino plaadil on mitmeid erinevaid komponente ning ühenduskohti, esialgu ei ole vaja kõiki teada. Pildil nr. 2 on välja toodud peamised osad mida võiks teada.



Pilt nr. 2

Esimesed kolm osa (a–c) on olulisemad, simuleeritud Arduino keskkonnas, sest nende kohta saab kirjutada programmi ning sinna saab ühendada erinevaid komponente. Ülejäänud on olulisemad pigem siis, kui tegemist on füüsilise Arduino plaadiga. Seetõttu ei mängi need eriti suurt rolli simuleeritud süsteemides, aga kasulik on nende eesmärki ikkagi teada.

a) Tavalisel Arduino plaadil on 13 digitaalsignaali töötlemiseks sobilikku väljaviiku ehk pesa ning osa neist suudavad vajadusel väljastada ka analoogsignaali (pesad 0-13).

Digitaalsignaali on lihtsustatult seletades signaal, millel on kaks väärtust. Kas 0 või 1, kas elekter on või ei ole (ehk siis, kas palju või vähe elektrit), näiteks triipkood.

b) Analoogsignaali väljaviike on 6 (pesad A0-A5) ning need on peamiselt analoogsignaali töötlemiseks, kuigi nende abil saab töödelda ka digitaalsignaale.

Analoogsignaali on lihtsustatult seletades signaal, millel on palju väärtusi ning vaadatakse signaali muutumist. Näiteks kõne, kus kõneleja hääletoon ja tekitatava heli tugevus muutub.

c) Maanduse jaoks on kolm väljaviiku ning need on vajalikud elektroonikaskeemide tegemiseks, et moodustada vooluringe.



d) Väline vooluallikas on tähtis siis, kui tegemist on näiteks eraldi seisva Arduino poolt kontrollitava süsteemiga (liikuv robot, pimedas ise tööle hakkav lamp).

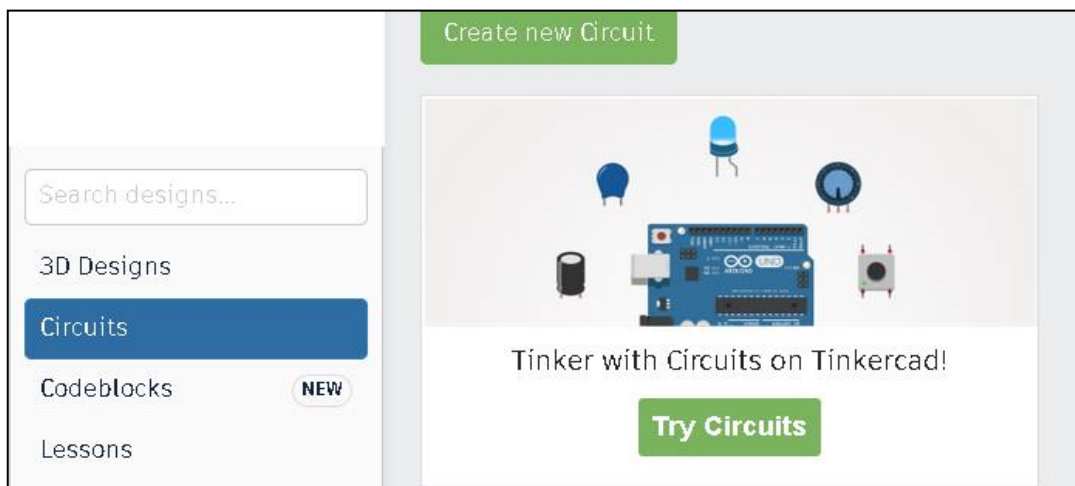
e) USB pesa – see on ühendamiseks arvutiga, et programmi laadida.

f) Kontroller ise – see on põhimõtteliselt nagu arvuti protsessor ja mälu üheskoos, seal toimub signaalide töötlus, andmete mälu pesadesse jagamine ning töötlemine.

g) Taaskäivitusnupp – on vajalik siis, kui Arduino, nagu ka iga teine arvuti, „kokku jookseb“. Sellega saab protsesse uuesti alustada.

### Saagu valgus – ehk esimene skeem

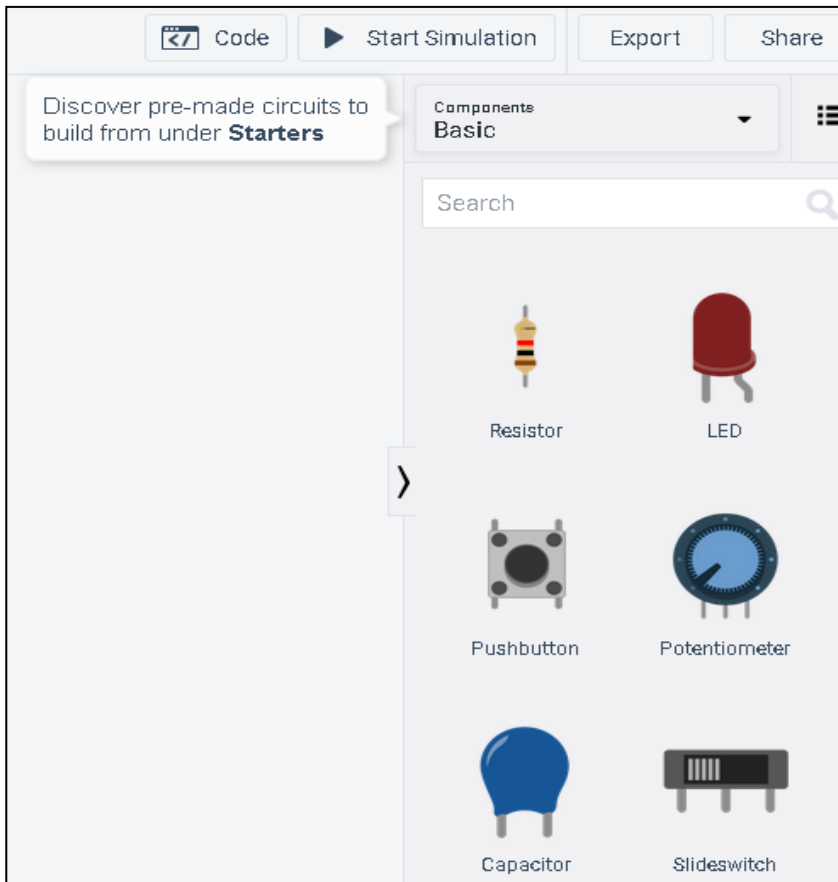
Käesoleval juhul kasutame oma skeemi koostamiseks ning koostatud süsteemi simuleerimiseks keskkonda nimega [www.tinkercad.com](http://www.tinkercad.com). Mõistlik on teha endale isiklik kasutaja, siis saab igal ajal internetiühenduse ning piisava kiirusega arvuti olemasolul oma skeemide kallal nokitseda. Selles keskkonnas on võimalik modelleerida mitmes erinevas valdkonnas, kuid meil on hetkel oluline elektroonikaseadmete osa. Selle leiame peale sisselogimist ekraani vasakult poolt jaotise „Circuits“ alt, nagu on näha alljärgneval pildil nr. 3.



Pilt nr. 3

Uue skeemi tegemiseks vali „Create new Circuit“ ning siis saab asuda endale vajalikku seadet koostama.

Selleks, et koostada endale vajalik skeem, tuleb vaadata ekraani paremasse serva, mida kujutab endast pilt nr. 4.



Pilt nr. 4

Et koostada skeem, tuleb meil valida erinevad komponendid, mis on leitavad komponentide menüüst. Skeemi koostamiseks tuleb need lohista vasakule ekraani tühja osa poole ning sobivas kohas lahti lasta (vastab inglise keelsele loogikale „Drag and Drop“).

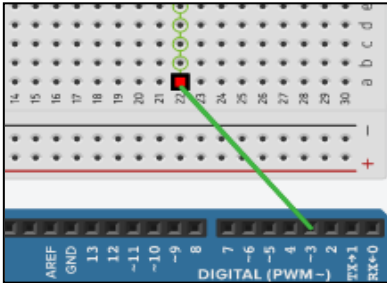
Komponentide ühendamiseks on kaks erinevat meetodit.

1. Komponentid ühendatakse otse sobivasse pesa, mis ei ole kõige tavalisem, sest päris skeemidel võib see tekitada liigset segadust.
2. Levinum meetod on kasutada ühendamist juhtmete abil (tavaliselt on kasutusel ka makettplaat, mille vajalikkust on näha esimese skeemi koostamisel). See meetod ei pruugi Tinkercadi keskkonnas esimesel paari korral õnnestuda, kuid loogika on selles, et tuleb minna hiirega pesa peale, kuhu tahetakse ühendada komponenti ning vajutada hiire vasak klahv alla ning siis klahvi all hoides liikuda ühendatava komponendi või asukohani ning siis hiirega klõpsata, pilt nr. 5.

Peamine probleem on selles, et hiire klahv lastakse liiga vara lahti või hakatakse klõpsutama, mis omakorda ühendab juhtme valesse kohta ning siis on selle paika saamine üsna tüütu. Tavaliselt on



lihtsam juhe eemaldada (klõpsata hiirega juhtme peale ja vajutada „delete“ nuppu) ning panna uus juhe.



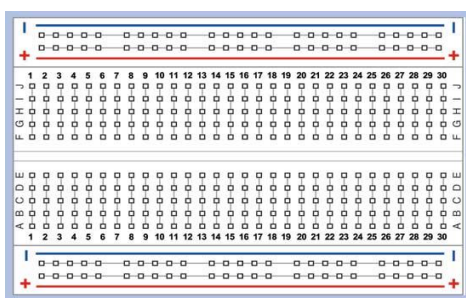
Pilt nr. 5.

Arduinot programmeeritakse C++ keeles, Arduinole kohaldatud variandis. Tinkercadi keskkonnas on see leitav ülevalt paremalt osa „Code“ alt. Vaata pilti nr. 4. Täpsemalt vaadatakse programmeerimist peale esimese skeemi koostamist, sest siis on midagi juba ka programmeerida.

### Esimese skeemi koostamine

Esimese skeemi koostamiseks on vaja tõsta paika järgnevad komponendid:

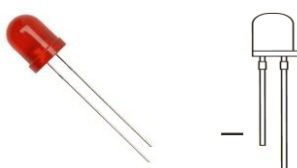
- Arduino ise (Arduino Uno – näha pildil nr. 1).
- Makettplaat – selle eesmärk on aidata komponentide ühendamisel. Tinkercadi keskkonnas võib ühendada ka palju juhtmeid kokku ning ei pea tekitama neile pistikuid. Samas, kui Arduinoga skeemi hakatakse füüsiliselt kokku panema, siis on vaja asju kokku joota. Makettplaati kasutades ei pea seda tegema. Samuti on eeliseks see, et pesad on horisontaalselt ja vertikaalselt



omavahel ühendatud. Ehk siis, kui ühendada näiteks pesasse 1A vool, siis on voolu all ka 1B, 1C, 1D ja 1E, mis võimaldab mitmeid asju korruga ühendada. Samamoodi on omavahel ühendatud ka „-“ ja „+“ märkidega pesade veerud.

Pilt nr. 6

- LED lambid ehk valgusdiodid on elektri mõjul valgust kiirgavad pooljuhtelemendid.



Pilt nr. 7

Nende üks omapärasid on see, et valgus süttib ainult siis, kui lambid on „õigetpidi“ ühendatud. See tähendab, et vool liigub Arduino plaadilt pikema jalani ning maandus on ühendatud lühema jalaga. Tinkercadi keskkonnas on pikem jalg see, mis on lambi suhtes nurga all.



- Juhtmed – neid saab ühendada, pildi nr. 5 juures mainitud meetodil. Elektroonikaseadmete puhul on üheks oluliseks asjaks juhtmete värvikoodid. Tavaliselt on juhtmeid kolme erinevat värvi. Punane juhe on voolu juhe (5V või lihtsalt numbriga pesasse, kui seadme toimimiseks ei ole vaja eraldi signaalijuhet), must juhe on maandus (läheb GND – pesasse) ning kollane juhe on signaalijuhe (see läheb tavaliselt numbriga pesasse). Igal komponendil ei ole signaalijuhet ning siis pannakse tavaliselt kas punane juhe või näiteks lampide puhul lambi värvi juhe.
- Takistid on erineva elektritakistusega elemendid, mille peamine eesmärk on muuta vooluringi elektritakistust, seda siis kas erineva tulemuse saamiseks või takistamaks teiste komponentide



kahjustumist. Tinkercadi keskkonnas saab takisti väärtust muuta selle peale klõpsates ning vastavat väärtust siis muutes. Füüsilist skeemi koostades tuleb vaadata takistil olevaid triipe, lugeda karbi pealt või siis vastava seadmega mõõta.

Pilt nr. 8

- Enne skeemi koostamisele asumist on mõistlik täpselt läbi mõtelda, mida on vaja ning otsida asjad virtuaalses keskkonnas üles või füüsilist süsteemi kokku pannes lihtsalt välja otsida. Samuti tuleks kindlaks teha, millist takistit on vaja, sest erinevatel seadmetel on erinevad nõuded.

Esimeseks koostatavaks skeemiks sobib hästi näiteks LED lambi töölepaneku skeem (pilt nr. 9), mis aitab harjutada nii juhtmete ühendamist kui ka vajalike väljaviikude meelde jätmist.

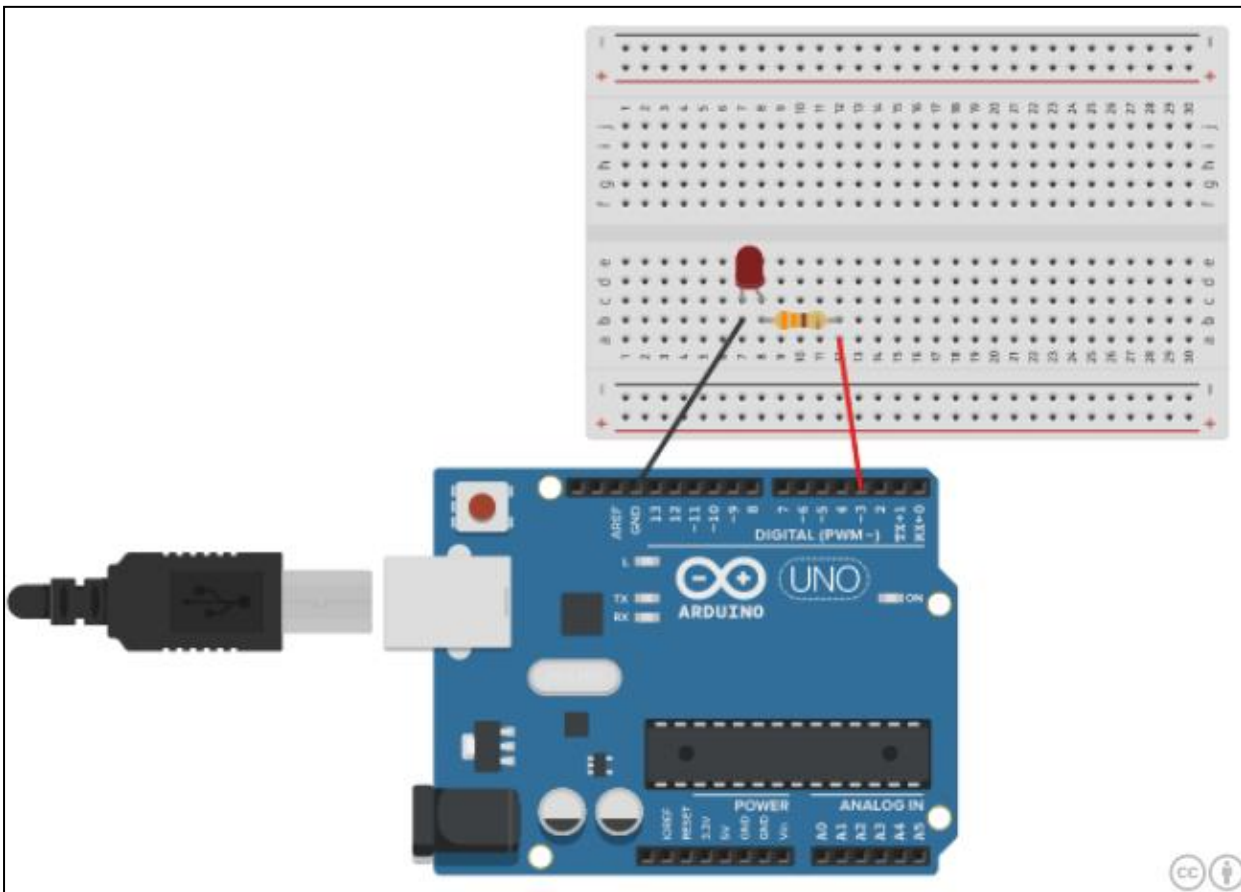
Siin on tavaliselt tekkimas järgnevad probleemid:

1. Peamine probleem, mis õpilastel Arduinoga töötades tekib, on see et kiputakse segi ajama numbreid Arduino plaadil ja makettplaadil ning siis imestatakse, miks tuli põlema ei lähe kuigi enda arvates on programmis kirjas õige pesa.
2. Samuti on üsna tavaline, et ühendatakse diood valepidi ning taas ei hakka tuli põlema.
3. Esineb ka juhtumeid, kus maandus jäetakse ühendamata ning kiputakse küsima, miks seda vaja on. Tavaliselt on noorematele õpilastele (kes pole koolis vooluringe õppinud) mõjunud selgitus, et elektrile meeldib liikuda ringiratast.





## Koostame esimese skeemi



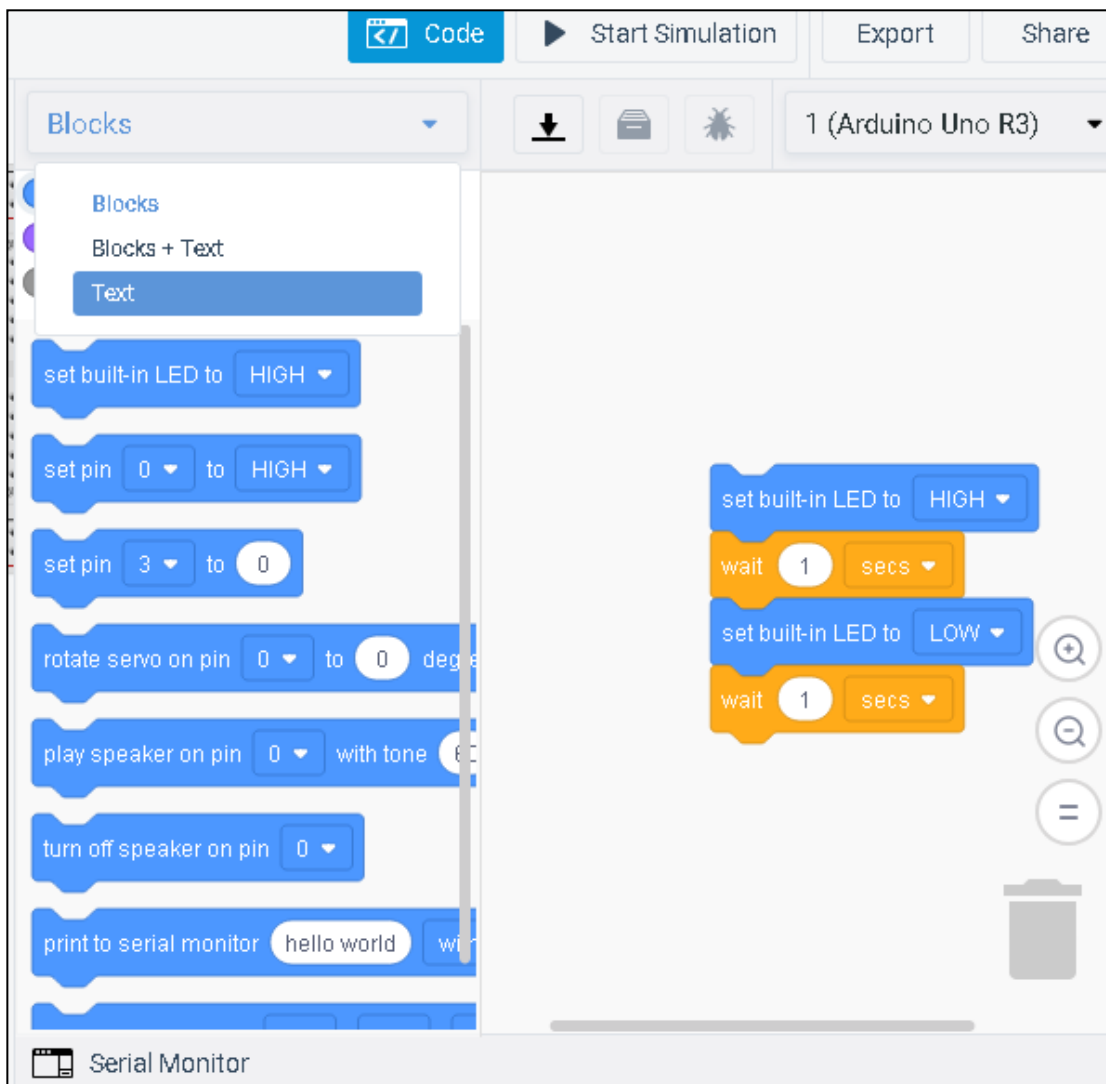
Pilt nr. 9

Siin on ühendatud Arduino plaadi pesa 3 punase juhtme abil makettplaadiga. Sealt omakorda jõuab vool läbi takisti valgusdiodi pikema jalani, pannes lambi vajaduse korral põlema. Diodi lühemast jalast läheb vool musta juhtme kaudu tagasi Arduino pesasse GND (maanduse pesa), moodustades vooluringi. Tinkercadis saab takisti horisontaalsesse asendisse viia „R“ klahvi vajutades. Nüüd tuleb aga vastvalminud elektroonikaseadet programmeerima hakata.



## Arduino programmeerimine

Programmeerimise leiab Tinkercadi keskkonnas üles ekraani parempoolsest ülemisest osast, kus on kirjas „Code“ (nagu on näha pildil nr. 10). Kui seda vajutada, ilmub esialgne plokkidest koosnev kood, mis meenutab programmeerimiskeelt Scratch. Sellega on lapsed ning suurem osa õpetajaid ka varem kokku puutunud. See algne kood paneb vilkuma Arduino sisse ehitatud pisikese kollase valgusdiodi (LED – lambi). Kuna tavaliselt on Arduino programmeerimine siiski tekstipõhine, siis valime „Blocks“ asemel „Text“. Seejärel ilmub arvuti ekraanile aken, kus küsitakse kinnitust jätkamiseks tekstipõhise programmeerimisega. Selleks tuleks vajutada „Continue“. Võimalik oleks programmeerida ka nii, et näha on plokkid kui tekst. See võimalus on hea õppimaks C++ koodist arusaamist, kuid nii võib koostatud skeemi vaatamine osutada keerukaks.



Pilt nr. 10





Peale kinnitust muutub programmeerimine tekstpõhiseks ning saame jätkata C++ keeles programmi kirjutamist. Siin tuleb meeles pidada, et kui vahetada uuesti programmeerimise meetod tagasi plokkide juurde, siis kipub Tinkercad ära unustama kõik selle, mis tekstina on sisestatud. Seega oleks väga mõistlik valmis kirjutatud programmi tekst mingisse tekstiredaktorisse salvestada.

Peale programmeerimise tekstipõhiseks muutmist, saab hakata vaatama Arduino programmeerimise eripärasid. Suurt erinevust loogikas Tinkercadi ja Arduino enda programmeerimiskeskonna vahel ei esine, viimasel on keerukamate süsteemide loomise puhul rohkem võimalusi. Samuti on programmi osade värvikoodid erinevad võrreldes Arduino enda programmeerimiskeskonnaga, kuid funktsionaalsuses erinevusi ei ole.

### Arduino programmi omapärad C++ keeles

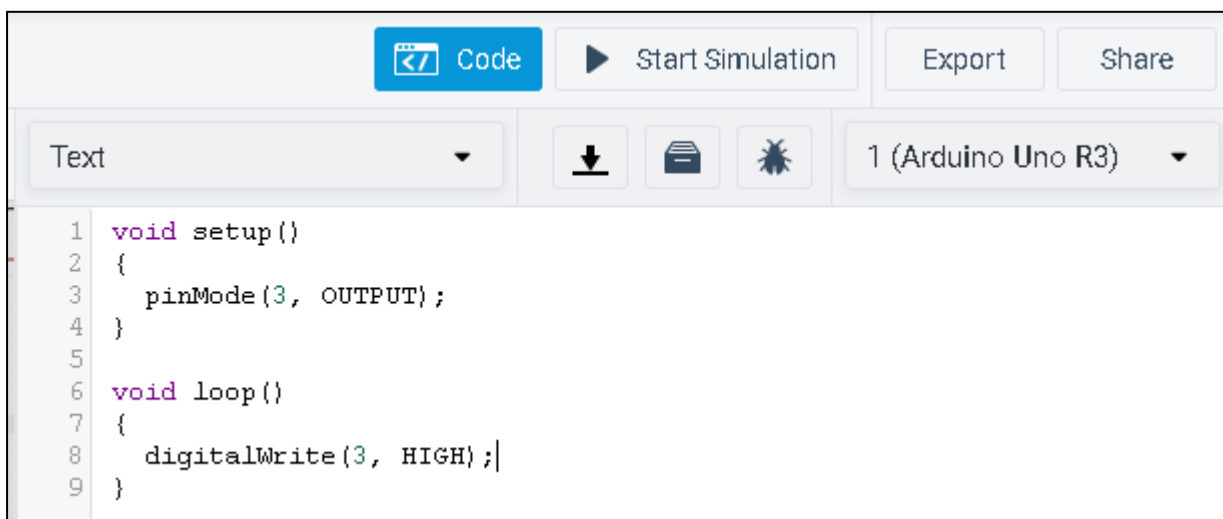
Lihtsamatel Arduino programmidel on ainult kaks funktsiooni (keerukamate puhul saab neid alati juurde teha). Nende funktsioonideta Arduino tööle ei hakka ning neid võiks vaadata kui programmi sees olevaid programme. Need funktsioonid on „void setup“ ja void loop“.

Sõna „void“ eesmärgiks on näidata, et need on tühemikud, mida saab täita funktsiooni sees olevate käskudega.

„void setup“ – toimub ainult ühe korra kogu programmi jooksul. Selle funktsiooni eesmärk on Arduinole teada anda, millistesse pesadesse on asjad ühendatud ning mida nendega peale hakata.

„void loop“ – see on programmi osa, mis tegutseb lõputult ehk siis kordab pidevalt ennast.

Näitena saab tuua lambi põlema paneku programmi pildil nr. 11.



```
Code Start Simulation Export Share
Text 1 (Arduino Uno R3)
1 void setup()
2 {
3   pinMode(3, OUTPUT);
4 }
5
6 void loop()
7 {
8   digitalWrite(3, HIGH);
9 }
```

Pilt nr. 11



Siin on näha funktsioonid „setup“ ja „loop“. Programm algab funktsiooniga „setup“.

void setup () – siin pannakse tööle funktsioon ning sulud näitavad seda, et midagi peab funktsiooni sees olema. Oluline on tähele panna loogelisi sulge, sest nendega algab ja lõpeb funktsioon ning kõik mis funktsioonis toimub, peab olema nende vahel. Kui see nii ei ole, siis programm lihtsalt ei toimi.

pinMode (3, OUTPUT); – siin antakse Arduinole teada, et olemas on pesa 3 ning sealt hakkab vool välja minema. Oluline on tähele panna rea lõpus olevat semikoolonit, sest see lõpetab rea ning programm läheb edasi järgmisele reale. Kui semikoolonit ei ole, siis Arduino ei tea, et on vaja järgmisele reale minna ning programm ei hakka tööle.

void loop () – siin hakkab toimuma programmi n-ö põhiosa ehk kordus. Siin on sama teema nagu igal funktsioonil ehk märk () näitab, et sinna on vaja tegevus sisse panna. Samuti peab funktsioon olema piiratud loogeliste sulgudega, et Arduino teaks millal see lõpeb ning võiks hakata tegelema järgmiste asjadega.

digitalWrite (3, HIGH); – siin antakse Arduinole käsk saata palju elektrit pesasse 3 ning see omakorda paneb lambi põlema. Loogika on siin selles, et saadetakse välja digitaalsignaali, millel on ainult kaks erinevat väärtust. Miks sisaldub käsu nimes sõna „Write“? Põhjus on selles, et Arduinol on põhimõtteliselt tegemist mälu pesade infoga täitmisega. Kui pesasse mis kannab nime, kirjutatakse midagi, siis Arduino teab, mida sellega peale hakata.

Programmi testimiseks on vaja vajutada ekraani paremas osas olevat nuppu „Start Simulation“, mis muutub seejärel roheliseks ning simulatsioon hakkab tööle. Kui kõik on õigesti ühendatud, siis hakkab lamp põlema. Simulatsiooni lõpetamiseks on vaja uuesti vajutada sama nuppu, kus nüüd on kirjas „Stop Simulation“ ning seejärel saab vajadusel teha programmis muudatusi, muidu muudatusi teha ei saa.

Kuna digitaalsignaali on võimalik välja saata kahte moodi, siis oleks mõistlik ka seda katsetada ning panna lamp vilkuma. Selle teostamiseks on vaja uurida paari uut käsku. Programm ise on pildil nr. 12.



```
Text [Download] [Save] [Arduino] 1 (Arduino Uno R3)
1 void setup()
2 {
3   pinMode(3, OUTPUT);
4 }
5
6 void loop()
7 {
8   digitalWrite(3, HIGH);
9   delay(100);
10  digitalWrite(3, LOW);
11  delay(100);
12 }
```

Pilt nr. 12

Nagu pildil nr. 12 näha, on juures 3 rida koodi ning kaks uut käsku.

delay (100); – see tekitab viiteaja järgmisele reale minekuks ning number näitab viiteaja kestust millisekundites (tuhandik sekundites). Lamp läheb põlema, siis ootab 100 millisekundit ning seejärel läheb programm edasi järgmisele reale.

digitalWrite (3, LOW); – siin on näha käsu „digitalWrite“ teine variant, kus Arduino saadab pesasse 3 vähe elektrit ning lamp enam valgust ei kiirga.

delay (100); – siin ootab Arduino jällegi 100 millisekundit, enne kui läheb järgmise rea juurde. Kuna aga järgmist rida enam ei tule ning tegemist on kordusega, siis hakkab programm jälle algusest peale ja lamp süttib uuesti.

Näiteks on võimalus vilgutada lampi mingi kindel arv kordi ning selle programmi tegemiseks on jällegi mitu meetodit. Kõige lihtsam, mida õpilased kohe tegema hakkavad, on koodi kopeerida niipalju kordi kui vaja ning lõpetada programm lambi kustumise ja pika viiteajaga. See on üsna kindel variant, samas väga tüütu programmeerida. Lihtsam on kasutada käsklust „for“, mis põhimõtteliselt ütleb Arduinole mitu korda on midagi vaja teha. Kood selliseks tegevuseks on toodud ära pildil nr. 13.



```
Text [Download] [Save] [Sun] 1 (Arduino Uno R3)
1 void setup()
2 {
3   pinMode(3, OUTPUT);
4 }
5
6 void loop()
7 {
8   for (int i = 0; i <= 5; i++) {
9     digitalWrite(3, HIGH);
10    delay(300);
11    digitalWrite(3, LOW);
12    delay(300);
13  }
14  digitalWrite(3, LOW);
15  delay(3000);
16 }
```

Pilt nr. 13

Siin on algus sama nagu varem. Teavitades funktsioonis „setup“ Arduinot pesas 3 paiknevast lambist, mida on võimalik välja saadetava elektri abil põlema panna.

Erinevused tulevad sisse korduse juures, sest seal hakatakse kasutama käsku „for“, mis alguses võib pisut arusaamatu paista. Järgneb koodi selgitus.

```
for (int i = 0; i <= 5; i++){
```

„for“ – see tähendab mingi muutuja jaoks hakkab toimuma tegevus.

„int i = 0;“ – see tähendab, et deklareerime muutuja „i“, mille väärtus on alguses 0, see hakkab muutuma funktsiooni sees.

i <= 5; – see tähendab, et muutuja „i“ muutub kuni väärtus on suurem või võrdne viiega.

i++ – see tähendab, et muutuja muutub sammuga 1 ning eelnevale väärtusele liidetakse kogu aeg 1 kuni see jõuab varem mainitud 5-ni.

„{,“ – see ütleb, et nüüd algab tegevus, mida on vaja korrata nii palju kordi, kui suur on muutuja „i“ väärtus.

Loogeliste sulgude sees toimub jällegi lambi vilgutamine nagu varem, kuid nüüd tehakse seda 5 korda.

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

```
delay (300);
```

```
digitalWrite (3, LOW);
```

```
delay (300);
```

```
„}“ – siin lõpeb käsk „for“ ning algab järgnev tegevus.
```



Kokkuvõtteks ütleme Arduinole sulgude sees, et on olemas muutuja „i“, mis hakkab muutuma 0-ist kuni 5-ni. Kui muutuja on võrdne 5-ga, siis lõpeb funktsioon „for“ ning jätkub tegevus, mis on programmis kirjas.

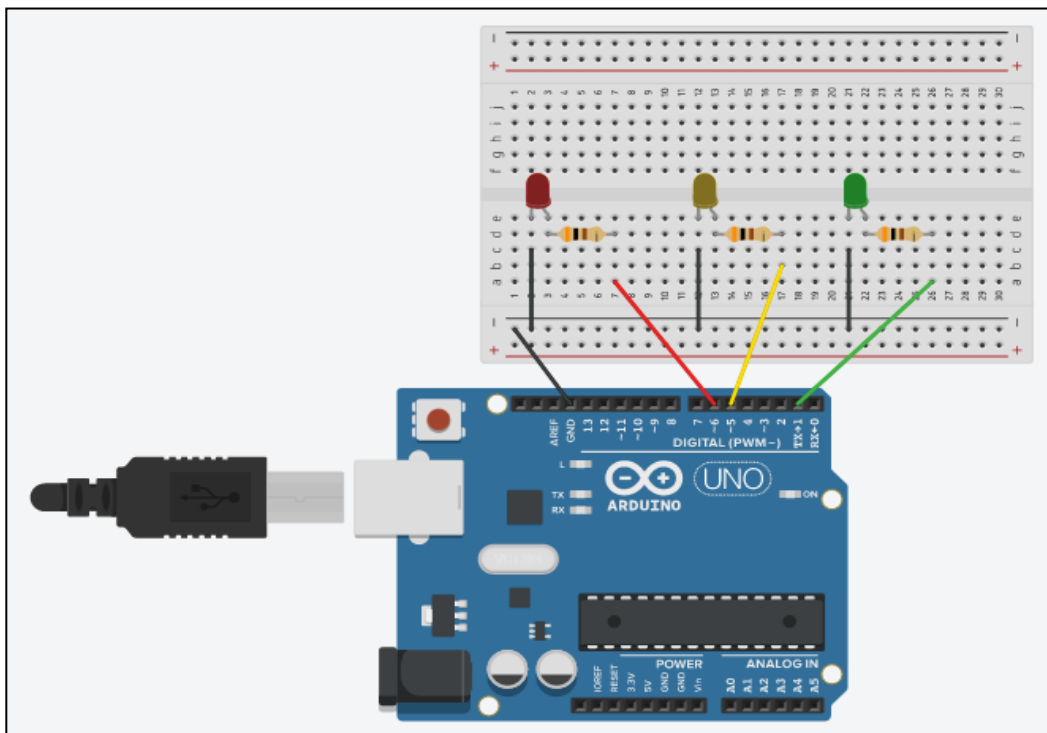
```
digitalWrite (3, LOW);
```

```
delay (3000);
```

See osa lülitab lambi välja kolmeks sekundiks, et peale seda uuesti alustada.

### Valgusfoori skeemi loomine

Valgusfoori saab edasi arendada eelmisest skeemist, lihtsalt on vaja juurde lisada paar erinevat värvi diodi ning takistit nagu on näha pildil nr. 14.

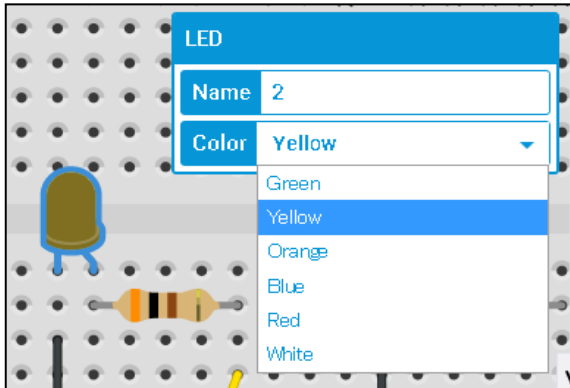


Pilt nr. 14.

Antud skeemi puhul ei ole järgitud mitte tavalist elektroonikaseadme loogikat, vaid juhtmete värv vastab diodi värvile, et oleks lihtsam leida üles, millisesse Arduino pesasse on lamp ühendatud. Samuti on maandus ühendatud eraldi juhtmete abil ühte „–“ veergu, sest Tinkercad ei võimalda otse ühendamist nagu seda saab teha füüsilise Arduino puhul (kuna lambi jalad ei ole piisavalt pikad).



LED-i värvi muutmiseks on vaja klõpsata lambi peale ning seejärel ilmub seadistuse aken, mis võimaldab valida sobiva värvi. Pildil nr. 15 on kujutatud kollast värvi lambi valimine, sest kõik lambid on algselt punased ning värv tuleb ise seadistada.



Pilt nr. 15

Valgusfoori puhul on alguses kaval katsetada, kuidas hakkaksid lambid tööle kordamööda. Üks kustub ära, siis hakkab põlema järgmine jne. Selle jaoks on olemas mitu erinevat koodi võimalust, kuid näidiseks on pildil nr. 16 ära toodud variant, kus saab ilma suurema vaevata muudatusi teostada, et näiteks tööle panna mitu lampi korraga, kui selleks on vastav soov.

```
Text [Download] [Save] [Run] 1 (Arduino Uno R3)
1 void setup()
2 {
3   pinMode(6, OUTPUT);
4   pinMode(5, OUTPUT);
5   pinMode(1, OUTPUT);
6 }
7
8 void loop()
9 {
10  digitalWrite(6, HIGH);
11  digitalWrite(5, LOW);
12  digitalWrite(1, LOW);
13  delay(100);
14  digitalWrite(6, LOW);
15  digitalWrite(5, HIGH);
16  digitalWrite(1, LOW);
17  delay(100);
18  digitalWrite(6, LOW);
19  digitalWrite(5, LOW);
20  digitalWrite(1, HIGH);
21  delay(100);
```

Pilt nr. 16





Antud programmi näidises on ära toodud see, et alguses tuleb alati Arduinole teada anda, mis on kasutusel ning mida nendega täpsemalt peale hakatakse. Praegusel hetkel antakse teada, et kasutusel on väljaviigud (pesad) 6, 5 ja 1 ning nendest saadetakse välja elektrit.

```
void setup ()  
{  
  pinMode (6, OUTPUT);  
  pinMode (5, OUTPUT);  
  pinMode (1, OUTPUT);  
}
```

Tulede vilgutamise kood paistab sihilikult olevat tehtud suur ja kohmakas, kuid sellisel koodil on eelpool mainitud tagamõte, võimaldamaks vajaduse korral seadistada valgusfoori võimalikult vähese vaevaga, muutes ainult ühte käsu osa (näiteks kirjutades HIGH asemele LOW või vastupidi).

```
void loop ()  
{  
  digitalWrite (6, HIGH);  
  digitalWrite (5, LOW);  
  digitalWrite (1, LOW);  
  delay (100);  
  digitalWrite (6, LOW);  
  digitalWrite (5, HIGH);  
  digitalWrite (1, LOW);  
  delay (100);  
  digitalWrite (6, LOW);  
  digitalWrite (5, LOW);  
  digitalWrite (1, HIGH);  
  delay (100);  
}
```

Kui meil aga oleks soov teha valgusfoor, mis toimiks nagu päriselt lii kluses olev foor, siis seda saab samuti teha mitut moodi. Koodi lihtsustamiseks võiks kasutada „for“ käsku, mis vilgutab näiteks



rohelist tuld nagu on näha kommentaaridega varustatud programmis pildil nr. 17.

```
Text [down arrow] [download icon] [print icon] [bug icon] 1 (Arduino Uno R3) [down arrow]
1 void setup () {
2   pinMode (6,OUTPUT); //punane
3   pinMode (5, OUTPUT); //kollane
4   pinMode (1, OUTPUT); //roheline
5 }
6 void loop () {
7   digitalWrite(1, HIGH); //paneb rohelise põlema
8   delay(3000);
9   //nüüd algab "for" tsükkel
10  for (int i = 0; i <= 3; i++) {
11    delay(500); // ootab pool sekundit
12    digitalWrite(1, HIGH); // roheline põleb
13    delay(500); //ootab pool sekundit
14    digitalWrite(1, LOW); //roheline kustub
15  }
16  //siin lõpeb "for" tsükkel ning programm läheb edasi
17  delay(1000); // ootab sek
18  digitalWrite(5, HIGH); //süttib kollane
19  delay(3000); // kollane põleb 3 sek
20  digitalWrite(5, LOW); //kollane kustub
21  delay(1000); // ootab 1 sekund
22  digitalWrite(6, HIGH); //süttib punane
23  delay(10000); // Punane põleb 10 sek
24  digitalWrite(6, LOW); //punane kustub
25  delay(1000); //ootab 1 sek ja kogu kordus hakkab otsast peale
26 }
```

Pilt nr. 17

### Kommentaariid

Sellise koodi puhul on oluline tähele panna kahe kaldkriipsuga („//“) märgitud osi, mis kujutavad endast programmikoodi seletavaid kommentaare. Kõikidest kohtadest (olgu need read või ridade tagumised osad) läheb Arduino lihtsalt sellest koodi osast üle ning ei loe neid programmi ülesande osadeks. Kommenteerimine on kasulik sellepärast, et siis on võimalik meelde jätta, millised asjad kuhugi on ühendatud ning mida mingi koodi osa teeb.