DigiGO! 2017-2019 Työpajoissa käytetyt opetusmateriaalit

Aku Kesti, Tommi Kokko, Maisa Mielikäinen & Tauno Tepsa
DigiGO! –hanke, Lapin ammattikorkeakoulu 2019

LAPIN YLIOPISTO
UNIVERSITY OF LAPLAND
Pohjoisen puolesta – maailmaa varten

LAPIN AMK
Lapland University of Applied Sciences

Elinkeino-, liikenne- ja ympäristökeskus

Vipuvoimaa EU:lta 2014–2020

Euroopan unioni
Euroopan sosiaalirahasto

SISÄLTÖ

Johdanto

Materiaalit on tehty vuosina 2017-2019 ja materiaaleja on käytetty eri työpajoissa DigiGo! hankeen aikana. Materiaaleista osat 2,3,4,5,6,7 on tehty yhdessä Lapin amk:n asiantuntijoiden ja oppilaiden kanssa.

Materiaalista osa 1, 2, 3, 4 ovat olleet käytössä digituutoreille suunnatuissa työpajoissa. Materiaaleista osat 4, 5, 6 ja 7 ovat olleet käytössä yläaste ja lukion oppilaille suunnatuissa työpajoissa.

Osat 1-4 vaativat yhden päivän mittaisen työpajan, vaativuuden vuoksi. Osat 4,5,6 ja 7 on suunniteltu tehtäväksi 45min pituisen työpajan aikana.

Työohje on valmistunut keväällä 2019

Työpajat ja ohjeet suunnittelivat DigiGO! –hankkeen Lapin amk:n asiantuntijat:
Aku Kesti, Tommi Kokko, Maisa Mielikäinen & Tauno Tepsa

Yhteistyössä Lapin amk:n opiskelijat: Tapio Moilanen, Jukka Seilonen & Valtteri Silmu

OSA 1
Ohjelmointi-opas

Sisältö ja tavoitteet

Opintomateriaaliin on koottu DigiGo -koulutuksissa opiskeltavien ohjelmoinnin sisältöjä. Materiaali on tarkoitettu avuksi c-kielen perusteiden oppimiseen Digi-Go hankkeessa. Oppaan tekemisessä on hyödynnetty muun muassa Simo Silanderin Ohjelmointi – Pro Training kirjaa ja eri www-lähteistä löytyviä hyödyllisiä materiaaleja.

Lisäksi hankkeen koulutuspäivän aikana ja itseopiskeluna suoritettavat ohjelmointitehtävät ovat olennainen osa ohjelmoinnin ajattelutavan sisäistämisessä. Ohjelmointia oppii yleensä parhaiten itse tekemällä ja soveltamalla teoriaa käytäntöön.

Tarkoituksena ei ole käydä kaikkia c-kielen asioita läpi perusteellisesti, vaan päästä sisään ohjelmoinnin ajattelumaailmaan oikean merkkipohjaisen kielen avulla. Tavoitteena on, että ohjelmoinnin opettamiseen liittyvien erilaisten aputyökalujen ja pelillisten ohjelmien ymmärtäminen helpottuu, kun päästään kiinni ohjelmoinnin ydinasioihin. Ymmärrys ohjelmoinnin maailmaan avartuu parhaiten näkemällä miten asioita tehdään oikeasti perustasolla. Koulutuksessa annetaan myös virikkeitä miten ohjelmointia ja algoritmista ajattelutapaa voidaan hyödyntää muissa oppiaineissa.

Koulutuksen materiaalin keskiössä on c-kieliset esimerkit, jotka parhaiten havainnollistavat tärkeimpien asioiden oppimista. C-kieltä oppii parhaiten kokeilemalla esimerkkejä itse ja muokkaamalla niitä omiin tarpeisiin. Tärkeintä on, että uskaltaa kokeilla ja tehdä rohkeasti yksinkertaisia ohjelmia. Virheitä ei kannata ohjelmoinnin alkuvaiheessa pelätä, koska niitä tekevät kokeneemmatkin ohjelmoijat.

Luvussa 2 on kokoelma erilaisista hyödyllisistä oppaista, joista voi opiskella enemmän ohjelmoinnista ja algoritmisesta ajattelutavasta sekä saada virikeitä oppitunneille. Luku 3 esittelee ohjelmoinnin käsitteitä ja ensimmäisiä c-kielisiä ohjelmia, joissa käsitellään tulostamista, muuttujia ja käyttäjän syötteitä. Luvussa 4 esitellään valintarakenteita ja luvussa 5 toistorakenteita. Tämän oppaan viimeinen luku 6 käsittelee funktioita.

"Virheiden pelko on tyhmyyden alku."

- Hakkarainen K. et al. (2004). *Tutkiva oppiminen, Järki, tunteet ja kulttuuri oppimisen sytyttäjinä.* Helsinkin: WSOY.

Linkkejä ohjelmointiin ja algoritmiseen ajattelutapaan liittyvistä oppaista ja artikkeleista

Tähän lukuun on luettelona koottu lähteitä, joita voi hyödyntää ohjelmoinnin opiskelussa.

- Linda Liukas, Juhani Mykkänen: https://s3-eu-west-1.amazonaws.com/koodi2016/Koodi2016_LR.pdf
- Ohjelmointiputka: http://www.ohjelmointiputka.net/
- Scratch: https://scratch.mit.edu/help/
- Virikkeitä opetukseen: http://koodi2016.fi/opetus.html
- Artikkeli: http://www.theedadvocate.org/why-coding-should-be-a-compulsory-subject-for-students/
-

Ensimmäiset ohjelmat

Ohjelmoinnin käsitteitä

Ohjelmalla tarkoitetaan ihmisen kirjoittamia toimenpideohjeita tietokoneelle, joiden avulla suoritetaan jokin tehtävä. Esimerkiksi voitaisiin kirjoittaa ohjelma, joka kysyy käyttäjältä luokan oppilaiden tenttiarvosanat ja laskee syötteiden perusteella arvosanojen keskiarvon ja keskihajonnan.

Tietokoneen suoritin eli prosessori suorittaa aina *konekieltä*. Konekielinen koodi koostuu yksinkertaisista komennoista, joilla voidaan pyytää suoritinta esimerkiksi tallentamaan tiettyyn muistipaikkaan tietty arvo tai laskemaan yksinkertaisia laskutoimituksia. Suorittimen ymmärtämä konekieli riippuu sen käyttämästä arkkitehtuurista, esimerkiksi x86-arkkitehtuurin suoritin ymmärtää x86 konekieltä. Ihmisen on kuitenkin lähes mahdoton tai ainakin erittäin vaikeaa kirjoittaa järkeviä ohjelmia käyttämällä suoraan konekielistä koodia. Ohjelmat pitäisi lisäksi kirjoittaa erikseen toimimaan kussakin halutussa suoritinarkkitehtuurissa.

Konekielistä koodin kirjoittamista helpottamaan on kehitetty *symbolisia konekieliä*, jossa kutakin koneen ymmärtämää käskyä vastaa jokin symbolinen sana. Lisäksi koneen muistipaikoille voidaan antaa symbolisia nimiä. Kullakin konekielellä on oma symbolinen vastineensa, esimerkiksi x86- konekieltä vastaa x86-assembly. Esimerkiksi symbolisen konekielen käsky voisi olla seuraavanlainen:
SUB PALKKA, VERO.

Symbolinen konekielinen koodi on kuitenkin vielä vaikeasti luettavaa ja ohjelmien koon kasvaessa koodin kirjoittaminen muuttuu yhä työläämmäksi. Ihmisen työtä helpottamaan on kehitetty korkeammalla tasolla olevia *ohjelmointikieliä*, kuten c, c++, java, php jne. Ohjelmointikielten tarkoituksena on päästä ohjelmoinnissa lähemmäksi ihmisen luontaista ajattelutapaa. Ohjelmoijan tarvitsee lähdekoodia kirjoittaessa tuntea ainoastaan ohjelmointikielen syntaksin. Nämä kielet, kuten assemblykin, *käännetään* konekieleksi tai *tulkataan* konekielisellä ohjelmalla. Korkean tason ohjelmointikielet mahdollistavat myös samojen ohjelmien käytön tietokoneissa, jotka eivät käytä samaa konekieltä.

Käännettävissä ohjelmointikielissä, kuten c ja c++, tarvitaan erillinen *kääntäjä* lähdekoodin kääntämiseksi konekielelle. Kääntäminen on siis muunnos, jossa ihmiselle helpossa muodossa oleva *lähdekoodi* muutetaan tietokoneen ymmärtämään muotoon. Kääntäjä ei kuitenkaan käännä lähdekoodia konekieliseksi ennen kuin kaikki kääntäjän löytämät syntaksivirheet on

koodista korjattu. Vasta käännettyä konekielistä ohjelmaa voidaan ajaa eli suorittaa.

C-kieli

C-kieli on standardisoitu vuonna 1990. C-kieli mahdollistaa suorituskyvyn ja muistinkäytön kannalta optimaalisten ohjelmien tekemisen, jonka ansiosta sitä käytetään yhä paljon esimerkiksi sulatetuissa järjestelmissä. Lisäksi c-kieli on siirrettävä eli c-kielinen koodi toimii hyvin eri laitealustoilla.

C-kieli tarjoaa ohjelmoijalle varsin vapaat kädet. Kielessä itsessään on varsin vähän rajoituksia, esimerkiksi muistipaikkojen käsittelyyn liittyen. Ohjelmoijalle tarjotun vapauden varjopuolena on, että kieli mahdollistaa myös varsin tiiviin ja sekavan koodin kirjoittamisen. Yleensä tavoitteena on mieluummin kirjoittaa selkeää ja helppolukuista, vaikkakin pidempää, ohjelmakoodia.

C-kieli sisältää joukon varattuja sanoja, joita ei voida käyttää muuhun kuin niille tarkoitettuun käyttöön. Esimerkiksi sana int on varattu c-kielessä tarkoittamaan kokonaislukutyyppiä ja sitä ei voida käyttää muussa tarkoituksessa, kuten muuttujan tai funktion nimenä. Varatuista sanoista, muuttujista ja funktioista on kirjoitettu tarkemmin myöhemmin. Hyvä linkki c-kielen standardeihin on http://www.cppreference.com/. Tutoriaaleissa tullaan harjoittelemaan standardien tarjoamista funktioista.

Ensimmäinen.c-kielinen ohjelma

1. *Lähdekoodin kirjoittaminen*.

```
/*
*Tämä ohjelma tulostaa näytölle Hello World
*/
#include <stdio.h>

/* Pääohjelma
*/ int main
(void)
{
        /* Pääohjelman koodi kirjoitetaan tähän, voi olla esim. kymmeniä rivejä
        */ printf(”Hello World \n");
        return 0;
}
```

Lähdekoodi kirjoitetaan jollain editorilla ja tallennetaan esimerkiksi nimellä hello.c. C-kielisten ohjelmien nimissä käytetään aina .c –päätettä.

Seuraavassa on vielä joitain tärkeitä huomioita yllä olevasta koodista.

- Kommentit kirjoitetaan aina /* ja */ merkkien väliselle alueelle. Käännettäessä lähdekoodia konekieliseksi koodiksi, kääntäjä jättää kommenttimerkkien väliin kirjoitetut asiat huomiotta.
- Koodissa saa olla tyhjiä rivejä välissä ja ylimääräisiä välilyöntejä sanojen välissä. Usein väljyys tekee koodista luettavampaa.
- Otsikkotiedosto (engl. header file) stdio.h on liitetty ohjelmaan mukaan lähdekoodin ensimmäisellä rivillä. Tämä mahdollistaa otsikkotiedoston sisältämien funktioiden, esimerkiksi printf-funktion käytön myöhemmin lähdekoodissa. Otsikkotiedosto stdio.h sisältää myös monia muita hyödyllisiä lähdekoodissa hyödynnettäviä funktioita.
- Pääohjelma eli main kirjoitetaan aaltosulkujen väliselle alueelle. Huomaa, että pää-ohjelman koodirivit on sisennetty oikealle. Tämä on tärkeä ohjelmoinnin tyyliseikka. Muistisääntönä jatkoa varten on, että aaltosulun jälkeen seuraavat

koodirivit sisennetään aina tietty määrä oikealle. Sisennyksen määrän voi ohjelmoija päättää itse.

- Jokaisessa c-kielisessä ohjelmassa on oltava pääohjelma eli main-funktio. **Ohjelman suoritus aloitetaan aina pääohjelman ensimmäiseltä riviltä**.
- Jokaisen suoritettavan lauseen perään kirjoitetaan ;. Tässä ohjelmassa on lause (printf-rivi). printf lauseessa \n tarkoittaa rivinvaihtoa. system(”pause”); rivi lisätään tässä, koska joissain editoreissa suoritus pitää pysäyttää, että ohjelma ei vain katoa ruudulta saavutettuaan pääohjelman viimeisen rivin. system –komennon suorittaminen vaatii myös kirjaston stdlib.h käyttöä, joka on sisällytetty ohjelmaan #include -komennolla. Viimeinen rivi pääohjelmassa tarkoittaa, että funktiosta palautetaan luku 0. Tämä on normaalikäytäntö pääohjelman yhteydessä.

2. *Lähdekoodin kääntäminen*
   Ohjelmakoodin kirjoittamisen jälkeen ohjelma käännetään *kääntäjän* avulla konekieliseksi koodiksi.

   Mikäli kääntäjä löytää lähdekoodista ohjelmointikielen syntaksin vastaisia rivejä, ilmoittaa se virheistä erillisessä virheikkunassa käyttäjälle. Ohjelman konekielistä objektitiedostoa eli .obj –tiedostoa ei tallenneta lainkaan ennen kuin kaikki kääntäjän ilmoittamat virheet on koodista korjattu.

3. *Konekielisten koodien linkittäminen suoritettavaksi ohjelmaksi*
   Kun kaikki kooditiedostot (.obj –tiedostot) on saatu käännettyä, linkitetään ne yhdeksi suoritettavaksi ohjelmaksi.
4. *Ohjelman suorittaminen*
   Ohjelma voidaan suorittaa joko käynnistämällä.exe tiedosto hakemistosta normaaliin tapaan tai valitsemalla editorista toiminto Run. Mikäli ohjelmoija haluaa suoraan oikoa välivaiheet 2-4, voidaan lähdekoodin kirjoittamisen jälkeen heti yrittää ajaa ohjelmaa. Editori ilmoittaa mahdollisista käännös- tai linkitysvirheistä, jotka on korjattava ennen kuin ohjelman suorittaminen onnistuu.

Kommentoinnista

Ohjelmiin kannattaa lisätä kommentteja luettavuuden parantamiseksi. Erityisesti kannattaa lisätä kommentteja

- Tiedoston alkuun kertomaan mitä ohjelma yleisesti tekee.
- Selittämään muuttujien käyttötarkoitusta.
- Sellaisiin ohjelman kohtiin, joka voi olla lukijalle vaikea ymmärtää (voi olla myös itselle muutaman kuukauden kuluttua).

Muuttujat

Muuttujien avulla voidaan tietokoneen muistipaikkoihin tallentaa haluttua tietoa. Talletettua tietoa voidaan myöhemmin muuttaa. Muuttujasta kerrotaan aina koodissa tyyppi ja nimi. Tyyppi kertoo minkä tyyppistä tietoa ohjelmoija haluaa muuttujaan tallentaa. C-kielen tavallisimmat tietotyypit on esitelty alla:

- **int** Kokonaisluku (16 bittiä eli luku joka on välillä -32767…32767)
- **unsigned int** Etumerkitön kokonaisluku (16 bittiä, mahtuu siis luku väliltä 0…65334)
- **long** Kokonaisluku (32 bittiä)
- **float** Desimaaliluku
- **char** Merkki
- **char[]** Merkkijonot

Muuttujat esitellään hyvän tavan mukaan heti aaltosulun jälkeisellä rivillä.

Esimerkki – Auton tiedot

```
#include <stdio.h>
int main(void)
{
     int auton_vuosimalli = 2017;
     char auton_merkki[20] = "Nissan Primera";

     printf("Autoni \n");
     printf("Merkki: %s \n", auton_merkki);
     printf("Vuosimalli: %i (uusi!) \n", auton_vuosimalli);
     return 0;
}
```

Esimerkistä selviää miten määritellään kokonaislukutyyppisiä muuttujia (int autonvuosimalli) ja merkkijonoja (auton_merkki[20]). Lisäksi koodista nähdään miten kyseisiä muuttujia voidaan tulostaa näytölle.

On tärkeää, että muuttujat nimetään järkevästi. Esimerkiksi edellisessä esimerkissä
int a = 2007;
olisi paljon huonompi ratkaisu nimetä muuttuja kuin
int auton_vuosimalli = 2007;

Muuttujien nimessä isot ja pienet kirjaimet ovat merkittäviä. Kun ohjelmassa myöhemmin haluttaisiin muuttaa auton vuosimalliksi 2008, seuraava ei toimisi:
Auton_vuosimalli = 2008;
Kääntäjä antaisi tästä virheilmoitukset ”undefined symbol” tai vastaavaa. Muuttujaa on siis käytettävä myöhemmin täsmälleen samalla nimellä kuin se on määritelty. Yleensä muuttujien nimet aloitetaan pienellä alkukirjaimella.

Muuttujien nimessä ei sallita skandinaavisia merkkejä (ä ja ö) ei välilyöntejä. Usein välilyöntejä korvaamaan käytetään alaviivaa. Toinen tapa on kirjoittaa toinen sana isolla alkukirjaimella. Esimerkiksi autonVuosimalli.

Esimerkki – Henkilötietojen lukeminen käyttäjän syötteistä

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
     char nimi[21];
     int ika;
     char sukupuoli = 'x'; /* Tallennetaan myöhemmin joko 'm' tai 'n' */
     float palkka;
     printf("Syöte nimesi: ");
     gets(nimi);

     printf("Syöte ikäsi: ");
     scanf("%i", &ika);
     fflush(stdin);

     printf("Syöte sukupuoli (m/n): ");
     scanf("%c", &sukupuoli);

     printf("Syöte palkka: ");
     scanf("%f", &palkka);

     printf("Syöttämäsi tiedot: Nimi %s, ika %i, sukupuoli %c, palkka %.2f \n",
                    nimi, ika, sukupuoli, palkka);
     return 0;
}
```

Esimerkissä on määritelty erityyppisiä muuttujia ja kysytty niiden arvot käyttäjältä. Käyttäjän syöttämät arvot luetaan talteen muuttujiin gets ja scanf –funktioiden avulla.

On hyvän tavan mukaista alustaa muuttujat heti esittelyvaiheessa johonkin alkuarvoon. Edellisessä esimerkissä nimi–muuttuja olisi voitu alustaa vaikkapa tyhjäksi. Vastaavasti ikä olisi voitu alustaa alkuarvoon 0.
char nimi[21] = "";
int ika = 0;

Jos muuttujalle ei anneta mitään alkuarvoa, niin tietokone itse asettaa sille jonkin määrittelemättömän arvon. Mikäli muuttujalle ei muisteta myöhemmin ohjelmassa antaa mitään järkevää arvoa, voi tällainen aiheuttaa kummallisia virhetilanteita ohjelmaa ajettaessa. Esimerkiksi, jos ohjelmoija unohtaisi kysyä käyttäjältä edellisessä esimerkissä nimi tiedon, tulostuu printf-lauseessa nimeksi jotain mystistä.

Sijoituslauseet

Muuttujan arvoa voidaan muuttaa myöhemmin ohjelmassa. Edellisessä esimerkissä muuttujan arvoa "muutettiin", kun käyttäjän syöte tallennettiin muuttujaan. Toinen yleisempi tapa muuttaa muuttujan arvoa on tehdä se sijoituslausekkeen avulla. Sijoituslause on aina muotoa

muuttujan nimi = muuttujan uusi arvo;

Esimerkiksi:
ika = 50;

Huomaa, että muuttujan tyyppiä (esim. int) ei enää kirjoiteta muuttujan nimen eteen. Se tehdään vain ensimmäistä kertaa muuttujaa esiteltäessä. Tässä muuttujan uusi arvo voi olla myös lauseke (ks. seuraava esimerkki).

Esimerkki – Ympyrän ala

```
#include <stdio.h>
int main(void)
{
     float pii = 3.14;
     float sade = 0;
     float ala = 0;

     printf("Anna ympyrän säde: ");
     scanf("%f", &sade);
     ala = pii * sade*sade;
     printf("Ala on %.2f \n", ala);
     return 0;
}
```

Esimerkissä ala-muuttujan uudeksi arvoksi lasketaan ympyrän ala rivillä *ala = pii * sade*sade;*

Esimerkki – Suorakulmion ala

```
/*
  Tämä ohjelma määrittelee suorakulmion leveyden ja korkeuden ja laskee niiden
  Perusteella suorakulmion pinta-alan
  Sivujen pituudet ja pinta-ala tulostetaan sen jälkeen näytölle
*/
#include <stdio.h>
int main(void)
{
       /* Alustetaan aluksi vain nollaksi, kun ei tiedetä oikeaa arvoa */
     float ala = 0;
     float leveys = 2.5;
     float pituus = 4.0;

     /* Lasketaan alalle oikea arvo */
     ala = leveys * pituus;
     /* Tulostetaan luvut yhden desimaalin tarkkuudella */
     printf("Suorakumion sivut ovat %.1f ja %.1f ja pinta-ala %.1f \n",
            leveys, pituus, ala);
     return 0;
}
```

Erikoistapauksia

- +=
  - Voidaan lisätä muuttujan arvoon tietty luku:
    ika += 10; / Sama asia olisi kirjoittaa ika = ika +10; */*
- -=
  - Voidaan vähentää muuttujasta tietty luku tai toinen muuttuja
    ika -= 10; / Sama asia olisi kirjoittaa ika = ika -10; */*
- ++
  - Voidaan kasvattaa muuttujan arvoa yhdellä
    ika++; / Sama asia olisi kirjoittaa ika = ika +1; */*
- --
  - Voidaan vähentää muuttujan arvoa yhdellä
    ika--; / Sama asia olisi kirjoittaa ika = ika -1; */*

Matemaattiset funktiot

Matemaattiset funktiot ovat kirjastossa math.h. Harjoituksissa käsitellään ainakin sqrt ja pow – funktioita. Taulukossa (ohjelmointiputka) on mainittu yleisimmin käytettyjä matemaattisia funktioita.

pow(luku, eksp)	potenssilasku
sqrt(luku)	neliöjuuri
ceil(luku)	pyöristys ylöspäin (kok.luvuksi)
floor(luku)	pyöristys alaspäin (kok.luvuksi)
sin(luku)	sinifunktio
cos(luku)	kosinifunktio
tan(luku)	tangenttifunktio

Esimerkki – Pallon tilavuus

Lasketaan pallon tilavuus:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

const float PII = 3.14; /* Tai voitaisiin kirjoittaa myös #define PII 3.14

*/ int main(void)
{
    float sade =
    0; float
    tilavuus;

    printf("Anna pallon säde:
    "); scanf("%f", &sade);
    tilavuus = 4/3 * PII * pow(sade,3);
    printf("Tilavuus on %.2f \n",
    tilavuus);
    system(”pause”);
    return 0;
}
```

Esimerkki – Luvun neliöjuuri

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float neliojuuri =
    0; float luku = 0;

    printf("Anna jokin luku, lasken neliöjuuren:
    "); scanf("%f", &luku);
    neliojuuri = sqrt(luku);
    printf("Neliöjuuri on %.2f \n",
    neliojuuri);
    return 0;
}
```

Vakiot

Vakion arvo pysyy samana koko ohjelman ajan. Kääntäjä antaa virheilmoituksen, mikäli ohjelmassa yritetään muuttaa sellaisen ”muuttujan” arvoa, joka on määritelty vakioksi. Tämä on hyvä siksi, että ohjelmoija ei voi edes itse vahingossa muuttaa sellaisen muuttujan arvoa toiseksi, jonka hän on alun perin ajatellut olevan muuttumaton. Edellisessä esimerkissä voitaisiin piin arvo määrittää vakioksi.Esimerkki – Ympyrä

```
#include <stdio.h>
const float PII = 3.14;

int main(void)
{
      float sade = 0,;
      float ala = 0;

      printf("Anna ympyrän säde: ");
      scanf("%f", &sade);
      ala = PII * sade*sade;
      printf("Ala on %.2f \n", ala);
      return 0;
}
```

Tässä vakio PII on määritelty const –avainsanan avulla. Vakio on määritelty globaalina vakiona eli main ohjelman ulkopuolella. Globaalit muuttujat käsitellään myöhemmin. Muuten vakion määrittely ei eroa tavallisen muuttujan määrittelystä.

Toinen tapa määritellä vakio on tehdä se #define määreen avulla. Tällöin ohjelma muuttuisi ainoastaan siten, että
const float PII = 3.14;
rivin tilalle kirjoitettaisiin
#define PII 3.14

Huomaa, että #define rivin perään ei tule puolipistettä, eikä =-merkkiä käytetä.

Ohjelma toimii täysin samalla tavalla, käytettiin kumpaa tahansa edellä mainittua tapaa määritellä vakio. const varaa hieman enemmän tilaa #define –määrittelyyn verrattuna, mutta on kuitenkin suositeltavampi tyyppitarkastuksen vuoksi.

Muotoilumääreet

Muotoilumääreitä tarvitaan ainakin tulostuslauseissa (printf) ja käyttäjän syötteiden lukemisessa scanf-funktiolla. Muotoilumääreillä ilmoitetaan minkä tyyppistä tietoa ollaan tulostamassa tai lukemassa. Tulostusmääreet eri tyyppisille muuttujille:

int	*%i tai %d*
long	*%ld tai %li*
float	*%f*
char	*%c*
char[]	*%s*

Esimerkki – Eri tyyppisten muuttujien tulostus

```
#include <stdio.h>
int main(void)
{
     char alkukirjain = 'U';
     float pituus = 189.5;
     int ika = 29;

     printf("Nimen alkukirjain on %c \n", alkukirjain);
     printf("Pituus on %.1f ", pituus);
     printf(" ja ika on %i \n", ika);
     return 0;
}
```

Tulostuskentän määreet

Tulostuskentälle voidaan määrittää leveys prosenttimerkin ja muotoilumääreen kirjaimen väliin. Kirjoittamalla viivan heti prosenttimerkin jälkeen, saadaan tulostus alkamaan kentän vasemmasta reunasta. Oletuksena tulostetaan oikeaan reunaan. Näin voidaan eripituisia merkkijonoja tai lukuja tulostaa siististi rinnakkain ja allekkain.

Esimerkki – Tulostaminen siististi allekkain

Tulostetaan luokan oppilaiden nimet, opintoviikkomäärät ja keskiarvot näytölle siististi allekkain

NIMI	OV	KA
Matti Meikäläinen	**7**	**3.5**
Aku Kesti	**23**	**3.0**
Malli Oppilas	**160**	**5.0**

```
#include
<stdio.h>
int main(void)
{
    char oppilas1[] = "Matti
    Meikäläinen"; char oppilas2[] = "Aku
    Kesti";
    char oppilas3[] = "Malli Oppilas";

    int ov1 = 7;
    int ov2 = 23;
    int ov3 =
    160;

    float arvosana1 = 3.5;
    float arvosana2 = 3.0;
    float arvosana3 = 5.0;

    printf("%-20s %-5s %-4s\n", "NIMI", "OV", "KA");
    printf("\n");
    printf("%-20s %-5i %-4.1f\n", oppilas1, ov1, arvosana1);
    printf("%-20s %-5i %-4.1f\n", oppilas2, ov2, arvosana2);
    printf("%-20s %-5i %-4.1f\n", oppilas3, ov3, arvosana3);
    return 0;
}
```

Aritmeettiset operaattorit

Ohjelmakoodissa on mahdollista laskea muuttujien ja lukujen avulla laskutoimituksia. Tätä varten on c-kielessä määritelty aritmeettisten operaattoreiden symbolit:

+	*yhteenlasku*
-	*vähennyslasku*
*	*kertolasku*
/	*jakolasku*
%	*jakojäännös*

Modulo-operaattorilla (%) voidaan laskea kahden luvun jakolaskun jakojäännös. Esimerkiksi:

- *int jakojaannos = 10 % 2; /'tallentaisi jakojäännos-muuttujaan arvon 0. */*
- *int jakojaannos = 10 % 3; /*tallentaisi jakojaannos-muuttujaan arvon 1.*/*

Usein jakojäännöstä käytetään tarkasteltaessa onko luku jaollinen jollain toisella luvulla

Esimerkki - Jakojäännös

```
#include <stdio.h>
int main(void)
{
    int syote, jakojaannos;
    printf("Anna luku, tarkastan onko se parillinen: ");
    scanf("%i", &syote);
    jakojaannos = syote % 2;
    if(jakojaannos == 0)
    {
     printf("Luku oli parillinen \n");
    }
    else
    {
        printf("Luku oli pariton \n");
    }
    return 0;
}
```

Ohjelmassa tarkastellaan onko käyttäjän syöttämä luku jaollinen 2:lla. if ja else –lauseet käsitellään tarkemmin myöhemmin.

Esimerkki – Luvun arpominen

Ohjelmassa arvotaan luku välille 0-9. srand() funktiolla voidaan alustaa satunnaislukugeneraattori siten, että jokaisella ohjelman ajokerralla arvotaan eri luku. rand() –funktio arpoo satunnaisen ison positiivisen luvun (joka mahtuu int-lukualueelle). Modulo-operaattorilla (%) voidaan luku skaalata halutulle välille, tässä välille 0-9.

```
#include <stdio.h>
#include <time.h>

int main(void)
{
     int arvottuLuku;
     int valitulos;
     srand(time(NULL)); /* Alustetaan satunnaislukugeneraattori */
     valitulos = rand(); /* Arvotaan satunnainen luku */
     printf("valitulos = %i \n", valitulos);

     arvottuLuku = valitulos % 10; /* Skaalataan välille 0-9 */
     printf("Arvottu luku on %i \n", arvottuLuku);
     return 0;
}
```

Valintarakenteet

Valintarakenteiden avulla voidaan ohjelmassa haarautua tilanteen mukaan suorittamaan oikea koodilohko.

Vertailuoperaattorit

Vertailuoperaattoreita käytetään valinta- ja toistorakenteiden yhteydessä

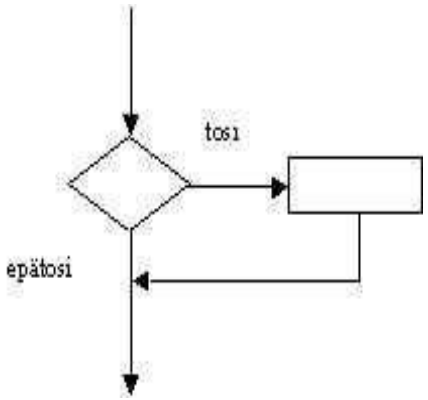
==	yhtä suuri kuin
<	pienempi kuin
>	suurempi kuin
<=	pienempi tai yhtä suuri kuin
>=	suurempi tai yhtä suuri kuin
!=	eri suuri kuin

Vuokaavioista

Vuokaavioilla voidaan helpottaa valintarakenteiden ja ohjelman suunnittelua. Tässä esimerkit ovat suhteellisen pelkistettyjä, mutta normaalisti ohjelmat ovat sen verran laajoja ja monimutkaisia, että vuokaavioista on oikeasti hyötyä. Ohjelman tekeminen pysyy paremmin hallinnassa kun voi tarkistaa ohjelman logiikan paperilta. Myös mahdolliset korjaukset on helpompi suunnitella ensin kaavioon kuin suoraan koodiin.

If-rakenne

If-rakenteen vuokaaviosta nähdään, että ehdon ollessa tosi suoritetaan if-rakenteeseen liittyvät lauseet. Laatikko kuvaa suoritettavia lauseita. Mikäli ehto on epätosi, jatketaan ohjelman suoritusta if-rakenteen jälkeisestä lauseesta.



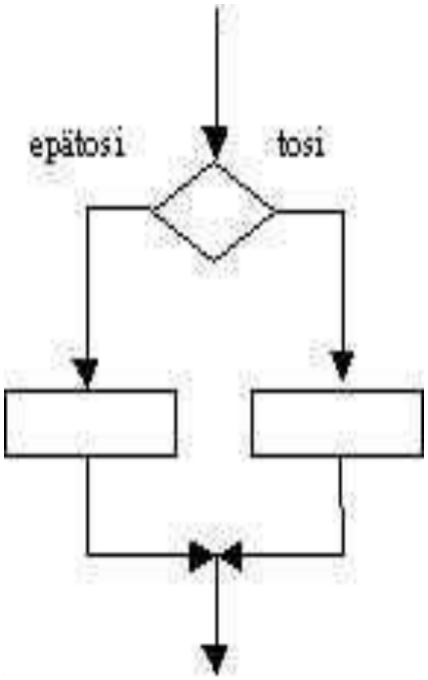
Esimerkki – Ikävertailu

Ohjelmassa verrataan onko ikä suurempi tai yhtäsuuri kuin 18 vuotta ja tulostetaan tässä tapauksessa ”Olet aikuinen”.

```
#include <stdio.h>
int main (void)
{
    int ika;
    printf("Anna ikäsi: ");
    scanf("%i", &ika);
    if (ika >= 18)
    {
            printf(”Olet yli 18 eli ohjelman mukaan ”);
            printf("Olet aikuinen \n");
     }
     printf(”Ohjelma loppu.”);
     return 0;
}
```

Huomaa sisennys if-lauseen jälkeen.

If-else –rakenne



Esimerkki – Kolmella jaollisuus

Esimerkissä tarkastellaan onko käyttäjän syöttämä luku jaollinen kolmella. Mikäli if-ehto ei toteudu mennään else-haaraan. Ohjelmasta suoritetaan pakosti aina joko if- tai else-haara.

```
#include <stdio.h>
int main (void)
{
    int luku;
    printf("Syötä luku: ");
    scanf("%i", &luku);
    if (luku % 3 == 0)
    {
         printf("Luku oli jaollinen kolmella. \n");
    }
    else
    {
         printf("Luku ei ollut jaollinen kolmella. \n");

    }
    return 0;
}
```

Esimerkki – Luvun vertailu

Esimerkissä tarkastellaan ensin onko käyttäjän syöttämä luku eri suuri kuin nolla. Jos näin on vertaillaan onko se pienempi kuin nolla ja tulostetaan tässä tapauksessa ”Luku oli negatiivinen”. Muussa tapauksessa ”Luku oli positiivinen”. Jos ensimmäinen ehto, jossa vertailtiin oliko luku eri suuri kuin nolla oli epätosi, mennään suorittamaan else-haara, jossa tulostetaan ”Luku oli nolla”.

```
#include <stdio.h>
int main(void)
{
   int luku;
   printf("Anna luku: ");
   scanf("%i", &luku);
   if (luku != 0)
   {
      if (luku < 0)
      {
         printf("Luku oli negatiivinen \n");
      }
      else
      {
         printf("Luku oli positiivinen \n");
      }
   }
   else
   {
      printf("Luku oli nolla \n");
   }
   return 0;
}
```

Else if
Esimerkki – Luvun vertailu toisin

Tehdään edellisen esimerkin ohjelma toisin. Toiminta on täysin sama. Valintarakenteista suoritetaan ohjelmassa aina joko if – tai else if – tai else –rakenne.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int luku;
   printf("Anna luku: ");
   scanf("%i", &luku);
   if (luku < 0)
   {
      printf("Luku oli negatiivinen \n");
   }
   else if(luku > 0)
   {
       printf("Luku oli positiivinen \n");
   }
   else
   {
      printf("Luku oli nolla \n");
   }
   return 0;
}
```

Esimerkki – Ikähaarukointi

```
#include <stdio.h>

int main(void)
{
    int ika;

    printf("Anna ikäsi: ");
    scanf("%i", &ika);

    if(ika < 18)
    {
           printf("Olet alaikäinen \n");

    }
    else if(ika >= 18 && ika < 30)
    {
         printf("Ikäsi on 18-30 vuotta \n");
    }
    else if(ika >= 30 && ika < 50)
    {
         printf("Ikäsi on 30-50 vuotta \n");
    }
    else if(ika >= 50 && ika < 60)
    {
         printf("Ikäsi on 50-60 vuotta \n");
    }
    else
    {
        printf("Ikäsi on yli 60 vuotta \n");
    }
    return 0;
}
```

Loogiset operaattorit

Loogisten operaattoreiden avulla voidaan yhdistellä eri ehtoja yhden ehtolausekkeen sisään. C-kielessä on kolme loogista operaattoria

&&	*looginen JA (AND)*
||	*looginen TAI (OR)*
!	*looginen negaatio (NOT)*

Loogisten operaattoreiden totuusarvot määräytyvät matematiikasta tutulla tavalla.

JA-operaattorin totuusarvo (&&)

- Ehtolauseke on tosi vain, jos kaikki ehdot ovat tosia
  if (a<3 && b>8 && c==0)
  Tämä on tosi esimerkiksi jos muuttujilla olisi seuraavat arvot: a=2, b=21; c=0

TAI –operaattorin totuusarvo (||)

- Ehtolauseke on tosi, jos vähintään yksi ehdoista on tosi
  if (a==0 || b == 3)
  ehtolauseke olisi tosi esim. arvoilla a=0 ja b=4

EI-operaattorin totuusarvo (!)

- Tosi, jos ehtolauseke on epätosi
  if (!(a==0 && b==4)
  Ehtolauseke olisi tosi esim. arvoilla a=0 ja b=8

Esimerkki – Ikä- sukupuolitutkimus

```
#include <stdio.h>
int main(void)
{
   int syntymavuosi;
   char nimi[21];
   char sukupuoli;

   printf("Anna syntymävuosi: ");
   scanf("%i", &syntymavuosi);
   fflush(stdin);

   printf("Anna nimi:");
   gets(nimi);

   printf("Anna sukupuoli: (m/n) ");
   sukupuoli = getchar(); /* tai scanf("%c", &sukupuoli); */

   if(syntymavuosi < 1980 && syntymavuosi >= 1970 && sukupuoli == 'm')
   {
          printf("Olet 70-luvulla syntynyt mies, kuulut liikkumistutkimuksen kohderyhmään.
   \n");
   }
```

```
    else if((syntymavuosi < 1970 || syntymavuosi >= 1980) && sukupuoli == 'm')
    {
        printf("Olet joko liian vanha tai liian nuori mies liikkumistutkimukseen. \n");
    }
    else
    {
        printf("Tässä tutkimuksessa tutkitaan vain 70 luvun miehiä. \n");
    }
    return 0;
}
```

Switch-rakenne

Switch-rakenne on joissain tapauksissa vaihtoehto if-else –rakenteelle. Sen yleinen muoto näyttää seuraavalta:

switch (ehtolauseke) {
case vakio1: case vakio2: case vakio3:

default:

lause1; break;

lause2; break;

lause3; break;

lause4;
}

Rakenteen
käytöstä.

- Ehtolausekkeen arvon täytyy olla joko int- tai char-tyyppiä. Arvoa verrataan kunkin case-lauseen vakion arvoon. Jos arvot täsmäävät suoritetaan case-lausetta seuraavia lauseita, kunnes kohdataan break-komento tai tullaan rakenteen loppuun.
- Huomaa, että jos unohdat break-sanan case:n lopusta 'valutaan' suorittamaan seuraavankin case:n lauseet automaattisesti ilman vertailua
- Lauseita ei tarvitse laittaa aaltosulkujen sisään.
- Kuitenkin koko switch-komennon runko on laitettava { ja } -sulkeiden sisään.break-lauseen suorittamisen jälkeen poistutaan rakenteesta.
- Jos täsmäävää vaihtoehtoa ei löydy suoritetaan default –vaihtoehdon lauseet. default-osa voi myös puuttua

Esimerkki – Hedelmät

```
#include <stdio.h>
int main(void)
{
    int valinta;
    printf("Valitse tuotteen numero: \n");
    printf("1. Banaani \n");
    printf("2. Omena \n");
    printf("3. Appelsiini \n");
    printf("\n");

    printf("Valintasi: ");
    scanf("%i", &valinta);
    printf("\n");

    switch(valinta)
    {
        int maara; /* Määritellään paikallinen muuttuja */
        case 1:
                    printf("Banaanin kilohinta on 1.50 euroa \n");
                    printf("Montako kiloa ostat: ");
                    scanf("%i", &maara);
                    printf("Ostokset maksaa %.2f \n", maara * 1.50);
                    break;

        case 2:
                    printf("Omenan kilohinta on 1.30 euroa \n");
                    break;


        case 3:
                    printf("Appelsiinin kilohinta on 0.90 euroa
                    \n");
                    break;
        default:
                    printf("Tuotetta ei ollut valikoimassa \n");
                    break;
        return 0;
}
                                        }
```

Toistorakenteet

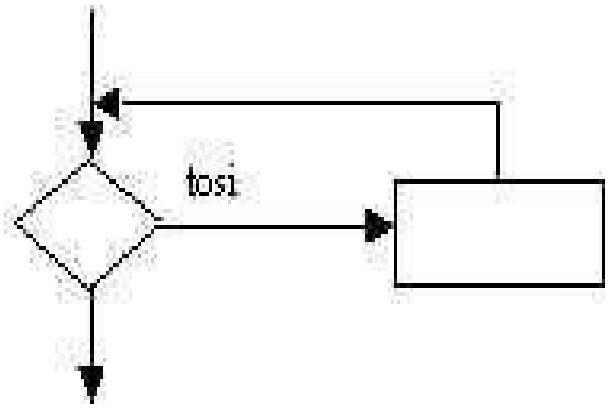
Toistorakenteita tarvitaan, jos ohjelmassa tarvitsee toistaa tiettyjä asioita.

C-kielessä on 3 erilaista toistorakennetta, jotka ovat vaihtoehtoisia keskenään. Joskus toinen rakenne on kuitenkin toista kätevämpi, usein toistorakenteen valinta on makuasia.

While -toistorakenne

```
while (ehto)
{
        lauseet;
}
```

Runko-osaa eli aaltosulkujen välissä olevia lauseita suoritetaan niin kauan kun ehto on tosi. Jos ehto on heti toistorakenteeseen siirryttäessä epätosi, lauseita ei suoriteta lainkaan.



Ehdon (salmiakki) ollessa tosi suoritetaan silmukan sisäiset lauseet (suorakaide). Tämän jälkeen palataan tutkimaan onko ehto yhä tosi. Kun ehto on muuttunut epätodeksi, ohjelmaa jatkaa eteenpäin toistorakennetta seuraavista lauseista.

Esimerkki – Lukujen tulostus

```
#include <stdio.h>
int main(void)
{
    /* Tulostetaan luvut 1-10 näytölle */
    int k=1; /* Kierrosmuuttujan alustus alkuarvoon */
    while (k<=10)
    {
      printf("%i \n", k); /* Tulostus */
      k++; /* Kierrosmuuttujan arvoa kasvatetaan yhdellä */
    }

    return 0;
}
```

Mitä tapahtuisi, jos jätät ohjelmasta pois rivin i++? Miten vastaavasti tulostat luvut nollasta sataan?

Esimerkki – Koepisteden summa

```
/*
* Lasketaan oppilaan 5 koetehtävän yhteispistemäärä
*/
#include <stdio.h>
int main(void)
{
    int tehtava = 1; /* Ttehtävän järjestysnumero */
    int pisteet = 0; /* Yksittäisen tehtävän pistemäärä */
    int summa = 0; /* Summa on alussa nolla */
    while (tehtava<=5)
    {
       printf("Anna pistemäärä: ");
       scanf("%i", &pisteet);
       summa = summa + pisteet;
       tehtava++;
    }
    return 0;
}
```

Esimerkki – Lukujen summa

```
#include <stdio.h>
int main(void)
{
        int pistemaara;
        int summa = 0;
        int lkm = 0;
        while (lkm < 10)
        {
                printf ("Anna pistemäärä: ");
                scanf ("%d", &pistemaara);
                while (pistemaara < 0 || pistemaara > 5)
                {
                        printf ("Ei kelpaa, anna väliltä 0-5: ");
                        scanf ("%d", &pistemaara);
                }
                summa += pistemaara;
                lkm++;
        }
        printf ("Pistemäärien summa on %d", summa);
        return 0;
}
```

Virheentarkistuksen toteuttava *while*-silmukka on ohjelman muusta toiminnasta riippumaton, erillinen "palikka". Se voidaan ottaa ohjelmasta pois ja lisätä siihen ilman, että ohjelman toiminta kelvollisilla syötteillä muuttuu millään tavalla. Kun ohjelmaa tällä tavalla rakennetaan "palikoista", saadaan aikaan ymmärrettävää ja ylläpidettävää ohjelmakoodia.

Ikuinen silmukka

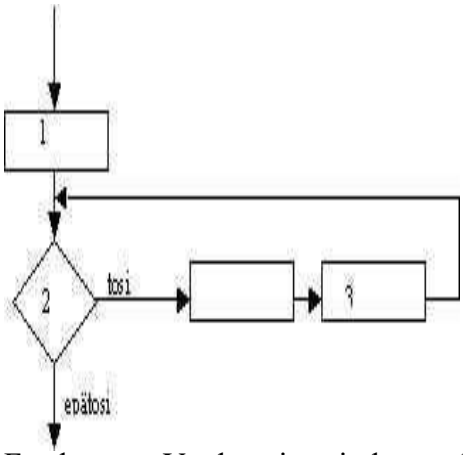
Ikuisella silmukalla tarkoitetaan tilannetta, jolloin ohjelman suoritus jää ikuisesti pyörimään silmukkarakenteeseen. Tämän vuoksi on aina muistettava vaikuttaa ehto-lausekkeen arvoon lauselohkon sisällä siten, että se muuttuu joskus epätodeksi. Esimerkiksi seuraava ohjelma jää ikuisesti pyörimään while-silmukassa.

```
#include <stdio.h>
int main(void)
{
        /* Tulostetaan luvut 1-10 näytölle */
        int k=1; /* Kierrosmuuttujan alustus alkuarvoon */
        while (k<=10)
        {
                        printf("%i \n", k); /* Tulostus */
        }
        return 0;
}
```

For -toistorakenne

```
for (alustus; toistoehto; päivitys)
{
        lauseita;
}
```

- Alustus suoritetaan vain ensimmäisellä kierroksella for-silmukkaan tultaessa.
- Runko-osa suoritetaan, jos toistoehto on tosi.
- Runko-osan suorittamisen jälkeen suoritetaan päivitys.
- Päivityksen jälkeen mennään testaamaan toistoehto. Jos tosi, niin suoritetaan jälleen runko-osa jne. Jos toistoehto ei ole tosi, jatketaan ohjelman suoritusta toistorakennetta seuraavasta lauseesta.



For-lauseen Vuokaavio esityksessä kolme operaatioita

1. laskurin alkuarvon asettaminen
2. laskurin arvon vertaaminen loppuarvoon
3. laskurin arvon kasvattaminen yhdellä

Esimerkki – Lukujen tulostus

```
/*
* Esimerkki for -rakenteen käytöstä. Tulostetaan näytölle allekkain
  luvut 1-10
*/
#include <stdio.h>
int main(void)
{
        int i;
        for(i=1; i<=10; i++)
        {
                printf("%i \n", i);

        }
        return 0;
}
```

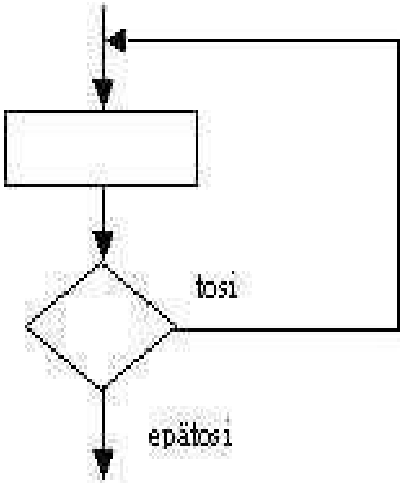
- Miten ohjelma toimisi, jos kirjoitettaisiin i++:n sijasta i = i+3?
- Mitä tapahtuu, jos for-rivin perään lisätään ’;’?

Do…while -toistorakenne

```
do
{
        lauseet;
} while (toistoehto);
```

- Tässä runko-osa suoritetaan vähintään kerran

- Vasta lopussa tarkastetaan toistoehto. Jos ehto on tosi, niin mennään suorittamaan runko-osan lauseet uudelleen. Jos ehto on epätosi, jatketaan suoritusta seuraavasta suoritettavasta lauseesta.



- Toistettaviksi tarkoitetut lauseet (suorakulmio) suoritetaan ensin yhden kerran.
- Sen jälkeen tutkitaan (salmiakki), pitääkö lauseet suorittaa uudestaan.

Esimerkki - Tunnusluku

```
#include <stdio.h>
int main(void)
{
        int luku;
        do
        {
            printf ("Anna tunnusluku");
            scanf ("%d",&luku);
        }while (luku != 1234);
        printf("Arvasit oikein!");
        return 0;
}
```

Esimerkki - Yhteenlaskukone

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
    int luku1, luku2;
    char vastaus;
    do
    {
      printf ("Olen yhteenlaskukone");
      printf ("\nAnna kaksi kokonaislukua: ");
      scanf ("%d %d", &luku1, &luku2);
      printf ("Summa on: %d",luku1+luku2);
      printf ("\nJatketaanko (k/e): ");
      vastaus = getche();
   }while (vastaus == 'k');
        return 0;
}
```

Funktiot

Funktio on rajatun tehtävän suorittava pieni ohjelma. Funktioiden avulla toteutetaan ohjelman modulaarisuutta eli ohjelman jakamista osiin. Ohjelmien koon kasvaessa ohjelmien hallittavuus ja luettavuus paranevat funktioiden ansiosta. Kaikkea ei tarvitse eikä pidä kirjoittaa pääohjelmaan. Hyvin valitut funktioiden nimet vaikuttavat osaltaan ohjelman ymmärrettävyyteen. Funktioiden ansiosta ohjelmaa voidaan rakentaa ja testata pala palalta. Näitä palasia (funkioita) voidaan käyttää mahdollisesti myös muiden ohjelmien rakennusosina.

Funktiot kannattaa suunnitella siten, että yksi funktio suorittaa aina yhden kokonaisuuden. Kannattaa mieluummin tehdä useita pieniä funktioita kuin yksi iso kaikenkattava funktio. Pieniä, selkeitä funktioita yhden asian suorittavia funktioita on helpompi ja joustavampi käyttää ohjelmassa.

Funktiot voidaan jakaa kahteen kategoriaan: valmiisiin c-standardin kirjastofunktioihin ja itse ohjelmassa toteutettuihin funktioihin. c-standardin mukaisia kirjastofunktiot saadaan käyttöön sisällyttämällä funktion sisältämä otsikkotiedosto ohjelmaan. Esimerkiksi tähän saakka ollaan käytetty printf, scanf ja gets –funktoita kirjastosta stdio.h.

Kirjastofunktioiden käyttäminen edellyttää, että tiedät minkä tyyppisiä parametreja funktiolle voidaan antaa ja mikä on funktion paluuarvo.. Esimerkiksi *scanf("%f", &luku);*
Tässä parametrit ovat "%f" ja &luku. Mikäli parametreja ei tarvita lainkaan, sulut on kuitenkin laitettava funktion perään. Esimerkiksi rand();

Seuraavaksi keskitymme kuitenkin omien funktioiden suunnitteluun ja toteuttamiseen.
Omia funktioita voidaan kutsua samalla tavalla kuin kirjastofunktioita. Funktion toteuttamisessa kannattaa lähteä liikkeelle rajapinnan suunnittelusta. Rajapinnalla tarkoitetaan funktion nimeä, sen parametreja ja paluuarvoa. Rajapintaa suunniteltaessa on siis mietittävä

- Mitä funktio tekee
- Mikä on funktiolle sopiva nimi
- Mitä parametreja tarvitaan
- Mitä funktion pitää palauttaa paluuarvonaan

(lähde http://goblin.tkk.fi/c/fin/lectures06/luento1.pdf)

Kuvassa funktiota on kuvattu isolla laatikolla. Sisään tulevalla nuolella on kuvattu parametreja ja ulosmenevällä paluuarvoa. Parametrit ovat muuttujia, jotka välittyvät funktiolle sitä kutsuneesta koodista. Paluuarvolla voidaan välittää tietoa takaisinpäin funktiota kutsuneeseen koodiin.

Funktio, jolla ei ole paluuarvoa eikä parametreja

```
/* Funktio kirjoittaa näytölle yksinkertaisesti sanan Kukkuu */
void kukkuu () /* OTSIKKORIVI */
{
      printf ("Kukkuu\n");
}
```

- void sanalla voidaan ilmaistaan, että funktiolla ei ole paluuarvoa. Funktion ei tällöin tarvitse palauttaa mitään lopputulosta eli yksittäistä arvoa sitä kutsuneeseen koodiin. Tyypillisesti funktiot, joissa ainoastaan tulostetaan jotain, eivät palauta mitään.
- Parametrit ilmoitetaan sulkujen sisällä heti funktion nimen jälkeen..Kuten huomataan, jos parametreja ei tarvitse välittää, suluissa ei välttämättä tarvitse olla mitään
- Funktiota suoritetaan ohjelmassa vain, jos sitä on kutsuttu (ks. seuraava kokonainen ohjelmaesimerkki).

Esimerkki - Kukkuu

Funktio koostuu kahdesta osasta: koodin alussa olevasta esittelystä ja itse funktiosta.

```
#include <stdio.h>
void kukkuu (void); /* Funktion prototyyppi eli esittely */
int main(void)
{
          kukkuu();
          return 0;
}

void kukkuu (void) /* Funktion määrittely eli toteutus */
{
          printf ("Kukkuu\n");
}
```

Esimerkki - Moikka

Seuraava esimerkki on ohjelmointiputkan: (http://www.ohjelmointiputka.net/opas.php?tunnus=cohj_3#funktio) sivuilta. Esimerkistä nähdään, että samaa funktiota voi kutsua ohjelmassa useaan kertaan. Tässä kutsutaan kolmesti funktiota moikka. Eli ”Moikka” tulostuu näytölle kolme kertaa.

```
#include <stdio.h>

/* funktion esittely */
void moikka(void);

int main(void) {
    /* funktiokutsu */
    moikka();
     moikka();
     moikka();
     return 0;
}

/* itse funktio */
void moikka(void) {
    printf("Moikka!\n");
}
```

Funktion parametrit

Funktion parametreja voidaan käyttää samalla tavoin kuin muitakin paikallisia muuttujia.

Parametreille annetaan arvo funktion ulkopuolelta. Arvoa voi toki muuttaa funktiossa, mutta arvon muutos ei näy funktiota kutsuneessa aliohjelmassa. Muutokset parametrien arvoon näkyvät siis ainoastaan funktion sisällä. Tällaisesta parametreista, joissa parametrien arvo välitetään funktiota kutsuneesta koodista, käytetään termiä arvoparametrit. On olemassa myös muuttujaparametreja, joihin palataan muuttujien yhteydessä. Käytännössä arvoparametrien käyttäytymistä on havainnollistettu seuraavassa esimerkissä.

Esimerkki – Tulosta luvut

```
#include <stdio.h>
/* Funktio tulostaa kymmenen seuraavaa lukua parametrin luku perusteella */
void tulostaSeuraavatLuvut(int luku);

int main(void)
{
    int luku;
    printf("Syötä luku, tulostan 10 seuraavaa: ");
    scanf("%i", &luku);
    tulostaSeuraavatLuvut(luku);

    /* Luku-muuttujan arvon muutokset eivät näy tänne */
    printf("Luku muuttujan arvo funktiokutsun jälkeen on %i \n", luku);
    return 0;
}

void tulostaSeuraavatLuvut(int luku)
{
    int i = 0;
    luku++; /* Kasvatetaan luku-muuttujaa heti yhtä suuremmaksi */
    while(i<10)
    {
            printf("%i, ", luku);
            luku++; /* Kasvatetaan luku-muuttujaa */
            i++;
    }
    printf("\n");
}
```

luku: 12

luku: ~~12 13 14 ...~~ 22

Kun esimerkki koodi ajetaan, havaitaan että luku-muuttujan arvo ei ole muuttunut pääohjelmassa. Tätä voidaan selittää viereen piirretyllä kuvalla ohjelman muuttujista. Kun ohjelmaa suoritetaan ja ollaan pääohjelman ensimmäisellä rivillä, varaa kääntäjä tietokoneen muistista tilaa muuttujalle luku. Käyttäjältä kysytään arvo, joka muuttujaan tallennetaan (arvo 12). Tätä kuvaa ylempi laatikoista. Tämän jälkeen tulee funktiokutsu eli ohjelman suoritus siirtyy funktioon tulostaSeuraavatLuvut. Funktiolle välitetään parametrina luku-muuttujan arvo. Kääntäjä luo kuitenkin funktiota varten oman väliaikaisen uuden luku-muuttujan tietokoneen muistiin. Tähän uuteen muuttujaan kääntäjä kopioi parametrina välitetyn arvon. Vaikka funktiossa muutetaan luku- muuttujan arvoa, muuttuu käytännössä vain tämän väliaikaisen muuttujan arvo. Pääohjelmassa voimassa olevaan luku-muuttujaan ei kosketa!

Se, että parametreista tehdään väliaikaisia muuttujia funktion suorituksen ajaksi, on koodin hallinnan kannalta hyvä asia. Esimerkiksi tässä pääohjelmaa koodatessa voidaan olla varmoja siitä, että funktio ei pääse muuttamaan käyttäjän syöttämää luku-muuttujan arvoa, olipa funktio toteutettu miten tahansa.

```
#include <stdio.h>
/* Funktio laskee kaupingin väkiluvun prosenttiosuuden koko maan väestöstä  */ void
tulostaProsenttiOsuus(long maan_vakiluku, long kaupungin_vakiluku);
int main(void)
{
    long suomi, rovaniemi;
    printf("Suomen väkiluku: ");
    scanf("%li", &suomi);
    printf("Rovaniemen väkiluku: ");
    scanf("%li", &rovaniemi);
    tulostaProsenttiOsuus(suomi,
    rovaniemi);
    return 0;
```

```
}

void tulostaProsenttiOsuus(long maan_vakiluku, long kaupungin_vakiluku)
{
          float prosenttiosuus = (float)kaupungin_vakiluku /
          maan_vakiluku; prosenttiosuus = prosenttiosuus * 100;
          printf("Kaupungin väkiluku on %.2f %% koko maan väkiluvusta. \n",
                      prosenttiosuus);
}
```

- Huomaa, että tässä on jouduttu käyttämään tietotyyppiä long, koska väkiluvut eivät mahtuisi int-tyypin lukualueeseen.
- Funktion kutsussa välitettyjen parametrien nimet ei tarvitse olla samat kuin funktiossa. Ainoastaan välitettyjen parametrien tyypit pitää täsmätä funktion otsikkorivillä esiteltyjen parametrien tyyppeihin. Tässä välitetään kaksi parametria ja molempien tyypit pitää olla long.
- Kääntäjä huolehtii siitä, että ensimmäisenä välitetyn parametrin arvo eli suomi-muuttujan arvo kopioidaan ensimmäisen parametrin arvoksi funktiossa (maan_vakiluku-muuttuja). Vastaavasti tehdään toisen välitetyn parametrin kanssa.

- Huomaa myös, että kun funktio on suoritettu, palataan suorituksessa takaisin kutsua seuraavalle riville.Kun funktio on toteutettu, mikään ei estä käyttämästä sitä useampaan kertaan samassa ohjelmassa. Esimerkiksi edellistä esimerkkiä voitaisiin laajentaa seuraavasti. Nuolista selviää ohjelman suoritusjärjestys.

```
#include <stdio.h>
/* Funktio laskee kaupingin väkiluvun prosenttiosuuden koko maan väestöstä

*/
void tulostaProsenttiOsuus(long maan_vakiluku, long kaupungin_vakiluku);

int main(void)
{
    long suomi, rovaniemi; long ruotsi, tukholma;
    printf("Suomen väkiluku: "); scanf("%li", &suomi);
    printf("Rovaniemen väkiluku: "); scanf("%li", &rovaniemi);
    tulostaProsenttiOsuus(suomi, rovaniemi);

    printf("Ruotsinväkiluku:");
    scanf("%li", &ruotsi); printf("Tukholman väkiluku: ");
    scanf("%li", &tukholma);
    tulostaProsenttiOsuus(ruotsi,tukholma);
    return 0;
}

void tulostaProsenttiOsuus(long maan_vakiluku, long kaupungin_vakiluku)
{
       float prosenttiosuus = (float)kaupungin_vakiluku /
       maan_vakiluku; prosenttiosuus = prosenttiosuus * 100;
       printf("Kaupungin väkiluku on %.2f prosenttia koko maan väkiluvusta. \n",
             prosenttiosuus);
}
```

Todelliset ja muodolliset parametrit

Termistösekaannusten vuoksi funktion otsikko-osassa määritellyistä parametreista käytetään eri nimeä kuin funktiokutsun yhteydessä määritellyistä parametreista.

- Funktion otsikko-osassa esitettävät parametrit ovat nimeltään *muodollisia* (formal) parametreja.
- Funktion kutsussa käytettävät parametrit ovat *todellisia* (actual) parametreja.

Funktion paluuarvo

Funktiosta palataan takaisin kutsuvaan ohjelmanosaan sen viimeisen rivin suorittamisen jälkeen tai

return-lauseella. return-lause on yleensä funktion viimeinen lause. Mikäli funktio palauttaa arvon, on paluuarvon oltava funktion otsikossa ilmoitettua tyyppiä. Ellei funktio palauta arvoa, *return*-lausetta ei tarvita. Funktion paluuarvon tyyppinä on tällöin *void*.. Funktiolla voi olla myös useita *return*-lauseita (vrt. seur. esimerkki).

Esimerkki - Parillinen

```
#include <stdio.h>
/* Seuraava funktio palauttaa luvun 1, jos parametrina välitetty luku on parillinen
ja muutoin luvun 0 */
int parillinen (int luku);

int main(void)
{
    int syote;
    int oliko_parillinen;
    printf("Anna syöte:
    "); scanf("%i",
    &syote);
    oliko_parillinen = parillinen(syote);if(oliko_parillinen == 1)
    printf("Luku oli parillinen \n");
 else
     printf("Luku oli pariton \n");

     return 0;
}

int parillinen (int luku)
{
  if (luku %2 == 0)
    return 1;
 else
     return 0;
}
```

- Usean return lauseen käyttäminen ei ole niin suositeltavaa (ei toki virhekään), koska se saattaa sekoittaa funktion lukijaa
- Eli tässä palautetaan funktiosta kokonaisluku (int). Tämä on ilmaistu funktion otsikkorivillä ensimmäisellä sanalla **int** parillinen (int luku)
- Return-lauseella palautettu arvo voidaan lukea muistiin funktiota kutsuneessa koodissa
  oliko_parillinen = parillinen(syote);
  Eli funktion palauttama paluuarvo luetaan muistiin muuttujaan oliko_parillinen.

Esimerkki – Maksimi kahdesta luvusta

```
#include <stdio.h>
int max (int luku1, int luku2);
void main (void)
{
  int x, y, suurempi;
  printf ("Anna kaksi kokonaislukua, niin ilmoitan kumpi niistä on suurempi.\n");
  scanf ("%i %i",&x, &y);
  suurempi = max (x,y);
  printf ("Suurempi luvuista on %i",suurempi);
  return 0;
}

int max (int luku1, int luku2)
{
  int maksimi;

  if (luku1 > luku2)
```

```
  {
      maksimi = luku1;
  }
  else
  {
      maksimi =luku2;
  }
  return maksimi;
}
```

Esimerkki - Mäkihyppy

Esimerkissä luetaan 5 mäkihyppytuomarin pisteet ja lasketaan niiden summa. Vaaditaan, että jokainen pistemäärä on 0 ja 20 välillä. Lukeminen, tarkistus ja mahdollinen uudelleen kysyminen tehdään funktiossa. Parametreina välitetään rajat. Paluuarvona funktio palauttaa luetun luvun, joka on kyseisellä välillä.

```
#include <stdio.h>
#define PIENIN 0.0
#define SUURIN 20.0

float ota_vastaan_luku_valilta ( float ala,  float yla );

int main (void)
{
  int i;
  float pisteet, summa = 0.0;
  for (i=1; i<=5; i++)
  {
      printf ("Anna tuomarin %i pisteet: ",i);
      pisteet = ota_vastaan_luku_valilta( PIENIN, SUURIN);
      summa += pisteet;
  }
  printf ("Pisteiden summa on %.1f",summa);
  return 0;
}

float ota_vastaan_luku_valilta (float ala, float yla)
{
  float luku;
  scanf ("%f", &luku);
  while(luku < ala || luku > yla)
  {
      printf ("Luvun on oltava välillä%.1f- %.1f.", ala, yla);
      printf (" Anna uudelleen: ");
      scanf ("%f",&luku);
  }
  return luku;
}
```

Esimerkki – Summa 1+2+3+…+N

Funktio *summa()* laskee summalausekkeen 1+2+3+...+N arvon, kun N välitetään funktiolle parametrina. Funktio palauttaa summalausekkeen arvon kutsujalleen *return*-lauseella:

```
#include <stdio.h>
int  summa  (int  n);
int main (void)
{
  int n;
  int s;
  printf ("Lasken summan 1+2+3+...+N. ");
  printf (" Anna N:");
  scanf ("%d",&n);
  s = summa (n);
  printf ("Summa on %d", s);
  return 0;
}

int summa ( int n )
{
  int i;
```

```
  int valisumma=0;
  for (i= 1; i <= n; i++)
  {
      valisumma += i;
  }
  return valisumma;
}
```

Esimerkki - Arvonta

```
/* Ohjelmassa arvotaan luku ja arvuutellaan sitä käyttäjältä, kunnes arvaa oikein */
#include <stdio.h>
#include <time.h>

int arvoLuku(int max);
void arvaaLuku(int luku, int max);

int main(void) {
          int luku = 0;
          int maksimi = 10;
          luku  =  arvoLuku(maksimi);
          arvaaLuku(luku, maksimi);
          return 0;
}


int arvoLuku(int max)
{
          int arvottuLuku = 0;
          srand(time(NULL));
          arvottuLuku = rand() % max + 1;
          return arvottuLuku;
}

void arvaaLuku(int luku, int maksimi)
{
          int arvaus;
          printf("Arvaa luku väliltä 0-%i: ", maksimi);
          scanf("%i", &arvaus);

          while(arvaus != luku)
          {
                    printf("Väärin. Arvaa uudelleen: ");
                    scanf("%i", &arvaus);
          }
          printf("Arvasit oikein!. \nLuku o
```

LAPIN AMK

Lapland University of Applied Sciences

OSA 2
DigiGO! Goes to Raspberry Pi 3

Tämä osa muodostuu syksyn 2017 DigiGO! -hankkeessa vedettyjen koulutusten työohjeesta.

Raspberry Pi 3

https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi 3 on kolmas sukupolvi Raspberry Pi-perheestä

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

https://pixabay.com/photos/video-game-console-video-game-play-2202618/

Suosittuja sivustoja. Sivustoilta voi ladata esim. Raspbian käyttöjärjestelmän ja katsoa Raspberry pi:llä tehtyjä mielenkiintoisia projekteja.
https://www.raspbian.org/

https://www.raspberrypi.org/

LINUX - KÄYTTÖJÄRJESTELMÄ

https://fi.wikipedia.org/wiki/Linux

Linux viittaa Linux-ydintä käyttävien Unixin kaltaisten käyttöjärjestelmien perheeseen. Linuxia voi käyttää monissa tietokonelaitteissa, muun muassa matkapuhelimissa, taulutietokoneissa, pelikonsoleissa, palvelimissa ja supertietokoneissa.

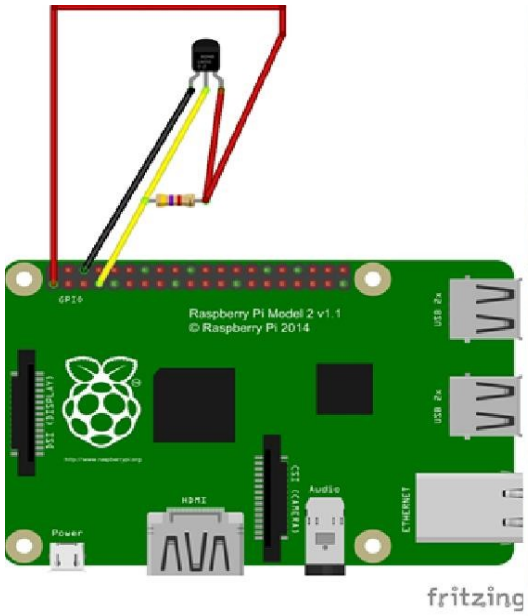
Linux on maailman käytetyin palvelinkäyttöjärjestelmä ja sitä käyttää kymmenen maailman tehokkainta supertietokonetta.

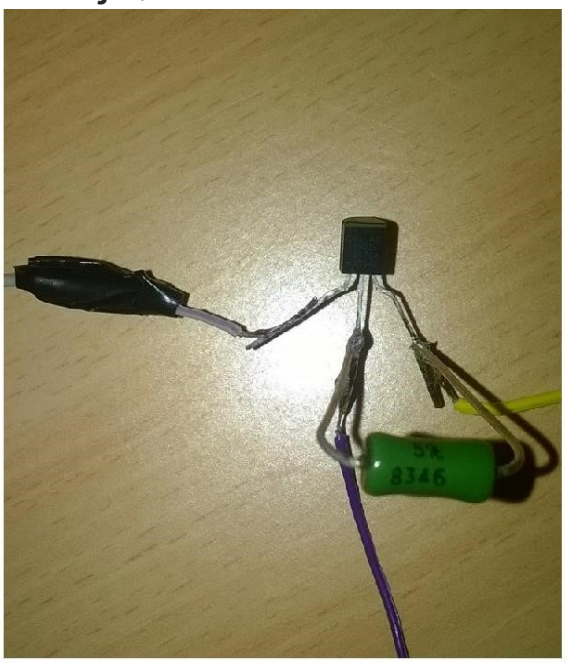
Nimi ”Linux” tulee Linux-ytimestä, jonka alun perin kehitti Linus Torvalds vuonna 1991.

https://pixabay.com/users/clker-free-vector-images-3736/

DS18b20 Temperature Sensor

Pin1 = 3V3, Pin7 = Datalinja, Pin6 = Ground

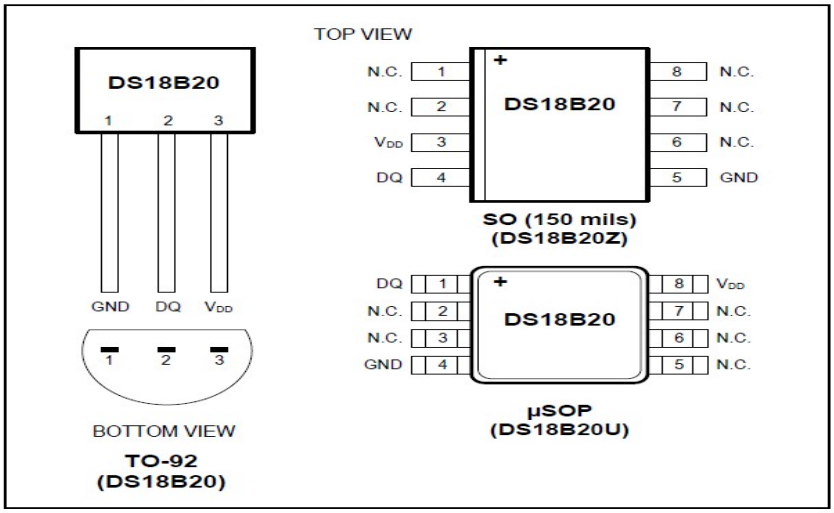




DS18b20 Datasheet

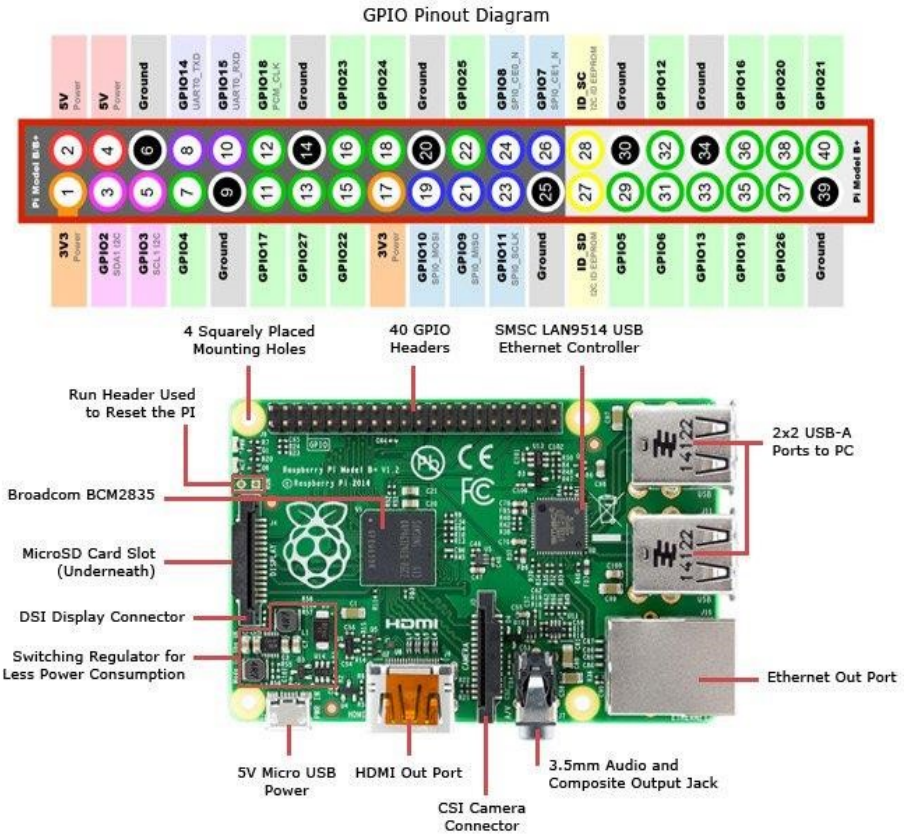
https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Pin Configurations



Raspberry Pi 3 GPIO pinout

- Pin 1 = 3V3



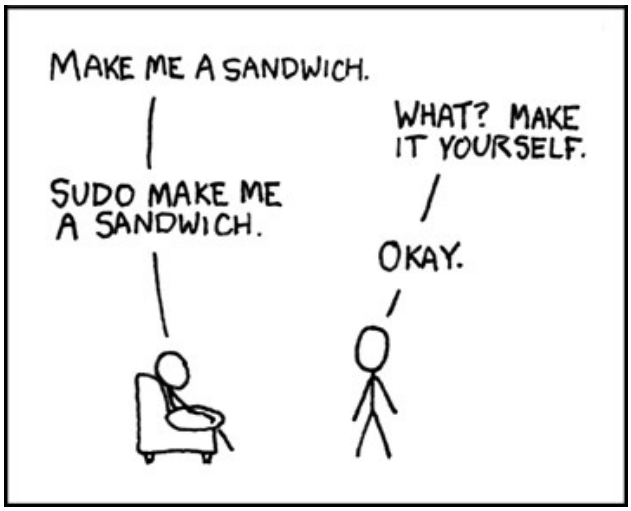
- Pin 6 = Ground
- Pin 7 = GPIO4

GPIO4 pinniä käytetään 1-wire väylän datalinjana

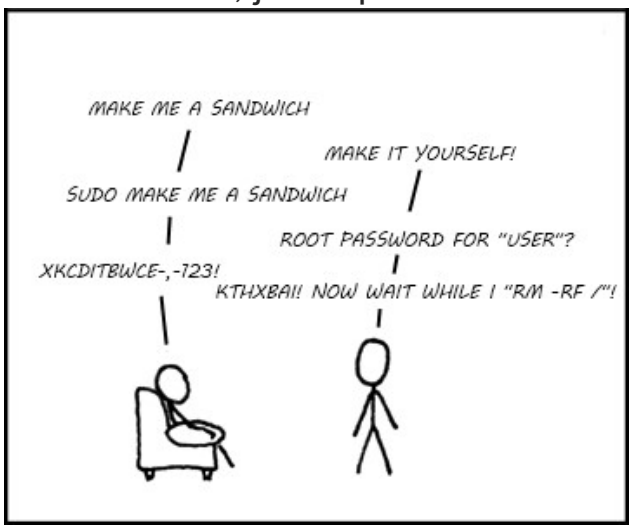
https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html

sudo = SuperUser DO

sudo make me a sandwich



sudo make me a sandwich, jos Super Userilla on salasana



https://xkcd.com/149/

POHJOISTA TEKOA

Olen tuli. Olen jää. Olen myrsky.

Olen tyven. Olen luja ja herkkä.

Olen pohjoista tekoa.

LAPIN AMK
Lapland University of Applied Sciences

OSA 3

DIGIGO goes to RASPI

Tämä osa muodostuu syksyn 2017 DigiGO! -hankkeessa vedettyjen koulutusten työohjeesta.

Raspberry PI 3

Tämä raportti on tehty yhdessä Lapin amk:n asiantuntijoiden ja oppilaiden kanssa.
Raportissa käsitellään Raspberry PI 3 -tietokoneen käyttöönottoa asennusta, lämpötilasensorin liittämistä sekä sensoridatan lukemista ja muokkaamista. Tämän raportin avulla lukija pystyy tekemään itsenäisesti kaikki vaiheet, mikäli kaikki tarvittavat osat ja laitteet löytyvät.

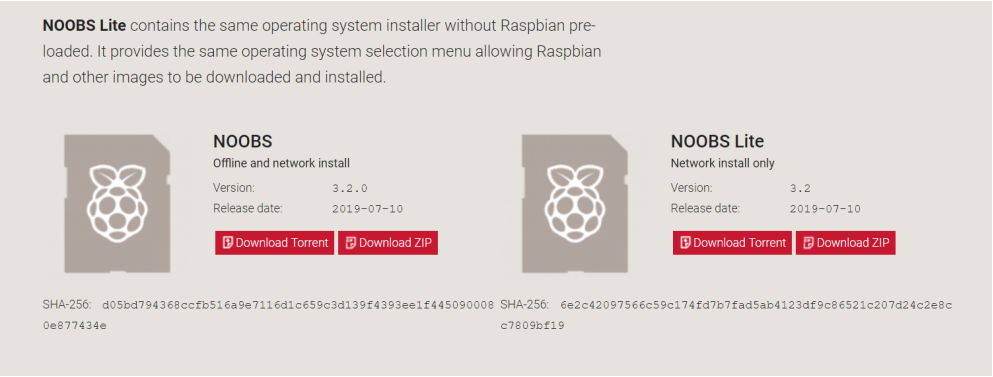
Tarvittavat osat

Raspberry PI 3 -tietokone, virtalähde (min. 2.4A micro-USB-liitännällä), micro-SD-muistikortti (min. 8 GB), HDMI-kaapeli, näyttö (HDMI-liitännällä), näppäimistö ja hiiri. Kotelo olisi myös hyvä olla, mutta se ei ole pakollinen.

Asennuksen valmisteleminen

Ensimmäisenä alusta muistikortti ”SD memory card formatter” -ohjelmalla.Ohjelmaon ilmainen ja se löytyy osoitteesta www.sdcard.org/downloads/formatter_4/ . Ohjelma toimii Windows- ja Mac-tietokoneissa.

Sen jälkeen ladataan asennusohjelma sivulta www.raspberrypi.org/downloads/ . Käyttöjärjestelmänä täytyy käyttää Rasbian Stretch versiota tai vanhempaa, koska uusin Rasbian versio Buster ei tue vielä lightptd:tä. Siinä on kaksi eri vaihtoehtoa, joista valitaan ”NOOBS”. Sen jälkeen pitää valita paikallinen tai verkkoasennus. Valitaan vasemman puolimmainen paikallinen offline asennus. Ladataan se zip-tiedostostona kohdasta ”Download ZIP”.



Asennuspaketin ZIP-tiedosto pitää purkaa SD-muistikorttiin. Jos asennuspaketin sisällä on vain img-tiedosto, sen purkamiseen tarvitaan Etcher-niminen ohjelma, joka löytyy osoitteesta https://etcher.io/. Sen käyttö on helppoa; valitaan

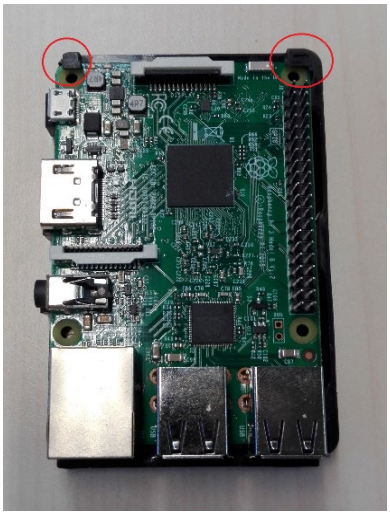
ensin img-tiedosto, sen jälkeen kohde (SD muisti kortti) ja lopuksi klikataan ”Flash”. Tämä ohjelma valmistaa muistikortista käynnistys levyn.

Asennus

Nyt voi asennus vaihe alkaa, jos sinulla kotelo asenna se ensimmäisenä. Aseta ensimmäisenä kotelon pohja pöydälle(kuva1). Aseta sen jälkeen Raspberry Pi piirilevy kotelon pohjalle siten että punaisella merkityt kielekkeet tulevat piirilevyn päälle(kuva2). Sen jälkeen voit asentaa kotelon väliosan(kuva3), joka tulee tasan kotelon pohjan kanssa. Lopuksi asetetaan kotelo kansi(kuva4).

kuva1

kuva2



kuva3

kuva4

Sen jälkeen voi asentaa muistikortin, muistikortin paikka löytyy pohjalta (kuva5). Helpoiten asennat muistinkortin kääntämällä kotelon ylösalaisin. Työnnä muistikortti johdinpinnat alaspäin ja tekstipuoli ylöspäin kotelossa olevaan syvennykseen (kuva5). Työnnä muistikortti pohjaan asti(kuva6).

kuva5

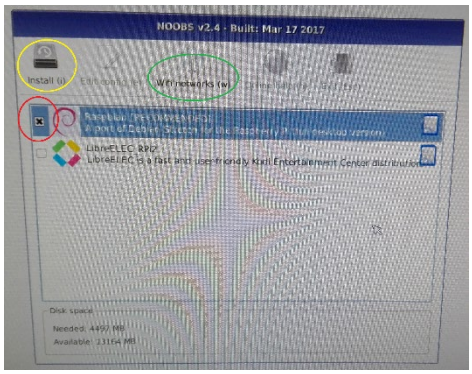
kuva6

TÄSTÄ ALOITETAAN KURSSIPÄIVÄNÄ

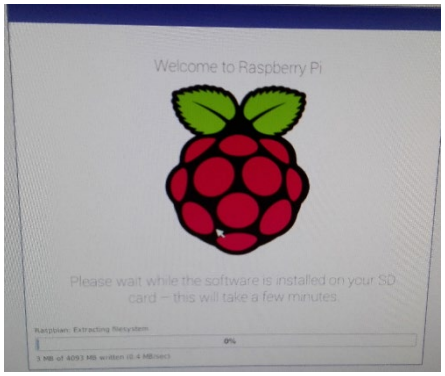
Yhdistetään näytön HDMI-kaapeli, näppäimistö, hiiri ja mahdollinen verkkojohto. Kiinnitä virtajohto viimeisenä. Virtajohdon kytkettyä Raspberry Pi -tietokone käynnistyy välittömästi. Valitaan Raspbian käyttöjärjestelmäksi rastimalla punaisella ympyrällä merkitty laatikko vasemmalla reunalla (kuva 8). Paina Install-painiketta. Asennusohjelma käynnistyy (kuva 9). Asennusvaihe kestää 10-30min. Tämän jälkeen Raspberry Pi käynnistyy uudestaan ja asennus on valmis.

Sen jälkeen voidaan liittyä langattomaan Wifi-verkkoon klikkaamalla hiirellä vihreällä ympyröityä ”Wifi Networks (w)” -kuvaketta(kuva8). Sitten voidaan valita langatonverkko, asettaa sille käyttäjätunnus ja salasana(kuva9). Wifi-verkkoa ei ole pakko asentaa vielä tässä vaiheessa, vaan sen voi myös myöhemmin asentaa.

kuva8



kuva9

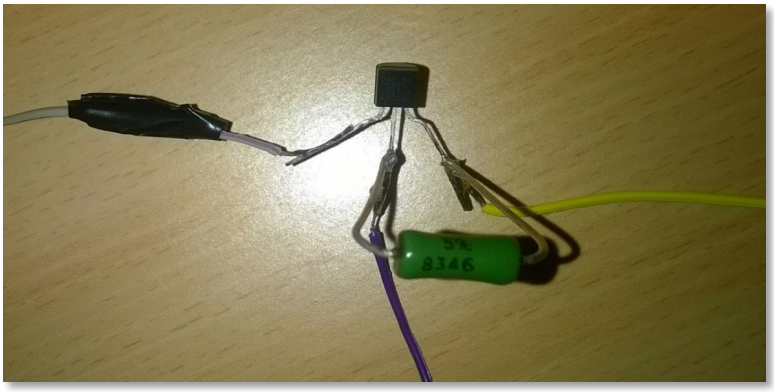


Vaikka nyt on asennettuna uusi käyttöjärjestelmä, niin aina suositellaan tehtäväksi pakettien ja turvallisuuteen liittyvät päivitys seuraavilla komennoilla, päivitykset kannattaa tehdä tasaisin väliajoin muutenkin.

sudo apt-get update
sudo apt-get upgrade

Lämpötila-anturin ja etuvastuksen liittäminen

Liitetään Raspberry Pi -tietokoneeseen DS18b20-lämpötila-anturi ja 4k7 (4700Ω) ylösvetovastus. Ne on valmiiksi tinattu yhteen ja niissä on valmiina johdot.



Ensimmäisenä sammutetaan tietokone (shutdown) ja irroitetaan kaikki johdot siitä. Sen jälkeen aukaistaan Rasberry Pi:in kansi kytkentöjä varten. Anturista tuleva johto liitetään Raspberry Pi:ssä pinniin numero yksi (3,3V). Sitten liitetään maajohto pinniin numero kuusi (maa, ground). Lopuksi liitettään dataväyläjohto pinniin numero seitsemän (GPIO4=General Purpose Input/Output).

Anturin asentaminen

Käynnistetään Raspberry Pi ja avataan Terminal vasemmasta yläkulmasta. Terminal on Raspin komentorivi. Komentorivillä ohjataan järjestelmää erilaisilla käskyillä. Ensimmäisenä täytyy muokata config.txt tiedostoa. Tämä tapahtuu kirjoittamalla käsky

sudo nano /boot/config.txt

Lisää tekstitiedoston loppuun rivi

dtoverlay=w1-gpio

GPIO-pin 4 otetaan sarjaliikennekäyttöön. (4 on default)

Tämän jälkeen lopeta painamalla Ctrl+x ja tallenna painamalla Y (yes). Vielä täytyy käynnistää käyttöjärjestelmä uudelleen komennolla

sudo reboot

Seuraavaksi asennetaan ajuri ja ohjelma sensorille Terminalissa antamalla käskyt

sudo modprobe w1-gpio
sudo modprobe w1-therm

Tämän jälkeen siirry hakemistoon komennolla

cd /sys/bus/w1/devices

Kirjoita *ls* ja näet kansion joka on nimetty 28-xxxxxxxxxxxx, joka on uniikki sarjanumero. Tulette tarvitsemaan tätä numeroa, joten laittakaa se muistiin. Tämän jälkeen voit mennä ko. kansion kirjoittamalla *cd sarjanumero*.

Voit myös käyttää kansion nimen kirjoituksessa apuna tabulaattori- näppäintä. Se toimii siten, että kirjoitat kansion nimen alun ja painat sen jälkeen tabulaattori näppäintä, jolloin tietokone kirjoittaa loput. Kirjoita lopuksi komentoriville

cat w1_slave

Näytöllä näkyvät nyt anturin tiedot:

CRC = paketin oikeellisuustarkistus
t=255000 eli 25,5 astetta

Sensoritiedon lukeminen tiedostoon

Mene juurihakemistoon kirjoittamalla *cd*. Tämän jälkeen kirjoita komentoriville (avaa piilotetun tiedoston .bashrc. Piste kertoo, että piilotettu)

nano .bashrc

Lisää tekstin loppuun rivit, jotta ajurit käynnistyvät automaattisesti, kun Linux käynnistetään

sudo modprobe w1-gpio
sudo modprobe w1-therm

Tallenna painamalla Ctrl+O. Paina lisäksi entteriä ja lopeta painamalla Ctrl+X.

Luodaan www-kansio komennolla

sudo mkdir /var/www
cd /var/www
sudo mkdir html

Annetaan rekursiivisesti oikeudet kaikkiin www-hakemiston kansioihin (tietoturvan kannalta vähän kyseenalainen)

sudo chmod -R 777 www

Tämän jälkeen mennään www-kansioon komennolla *cd www/html*. Luodaan log.py niminen tekstitiedosto komennolla

sudo nano log.py

Kirjoita ao. koodi siihen ja tallenna.

```
def dotime():
          import datetime
          #tuodaan päivä aika moduuli
          now = datetime.datetime.now()
          #haetaan aika ja päivä
          return now.strftime("%m-%d-%Y %H:%M")

def dotemp(snum):
    global thistext
    tcrc = "NO"
    filename = "/sys/bus/w1/devices/28-
XXXXXXXXXXXX/w1_slave"
    #aseta tiedosto mitä lukea
    while (tcrc == "NO"):
        thisfile = open(filename)
        thistext = thisfile.read()
        thisfile.close()
        fline = thistext.split("\n") [0]
        tcrc = fline.split(" ") [11]

    sline = thistext.split("\n") [1]
    tdata = sline.split (" ") [9]
    temp = float(tdata[2:])
    return (" %+2.1f" % (temp/1000))

sensor = "28- XXXXXXXXXXXX"

tfile = open("/var/www/html/temps.txt", "a+")

tfile.write(dotime())
tfile.write(dotemp(sensor))
tfile.write("\n")

tfile.close()
```

Sensor-kohtaan pitää muuttaa oma yksilöllinen anturin koodi.

Lopuksi ajastetaan äskeinen koodi toimimaan kerran minuutissa.
Tallenna CTRL+O, enter, CTRL+X.

crontab –e

Editoriksi valitaan Nano editor. Kirjoita ensimmäiseksi riviksi:

*/1 * * * * sudo python /var/www/html/log.py

Tallenna CTRL+O ja lopeta CTRL+X.

Odota pari minuuttia. Tarkista, onko crontab-ajastus toiminut kurkistamalla temps.txt-tiedoston sisältöön komennolla

cat temps.txt

Lämpötilatiedon näyttäminen Web-sivulla

Asennetaan web-serveri
Tarkista ensin, että web-yhteys on käytettävissä.

sudo apt-get install lighttpd

Asetetaan www-hakemistot ja annetaan raspille oikeus niihin sekä käynnistetään laite uudelleen.

sudo chown -R www-data:www-data www
sudo usermod –a –G www-data pi
sudo reboot

Avaa nettiselain ja tarkista onko web-palvelin käynnissä kirjoittamalla URL-riville (osoiteriville) *localhost*. Näytölle tulee Placeholder Page, jos kaikki kunnossa.

Asennetaan PHP-tuki

sudo apt-get install php7.1-common php7.1-cgi php7.1

Kerrotaan web-serverille, että PHP käytettävissä

sudo lighty-enable-mod fastcgi-php

Käynnistetään web-serveri uudelleen

sudo service lighttpd force-reload

Asennetaan gnuplot-työkalu lämpötilagrafiikan näyttämistä varten

sudo apt-get install gnuplot

Asennukset kestävät jonkun minuutin.

Aloitetaan kirjoittamalla **yksinkertainen web-sivu**

sudo nano /var/www/html/index.php

Kirjoita editoriin seuraava koodi

```
<?php
exec (‘tail –n 300 temps.txt > 300.txt’);
exec (‘gnuplot 300.plt’);
?>
<img src = temp300.png>
```

Koodi kopioi 300 viimeisintä rivitä temps.txt-tiedostosta temp300.png-tiedostoon. Ohjelma käynnistää gnuplot-työkalun ja hakee asetukset 300.plt-tiedostosta. 300.plt-tiedostossa sanotaan, että hae 300.txt:stä arvot kuvaajaan.

Tallennus ja lopetus: CTRL+O ja CTRL+X.

Seuraavaksi tehdään 300.plt-tiedosto eli **asetukset**

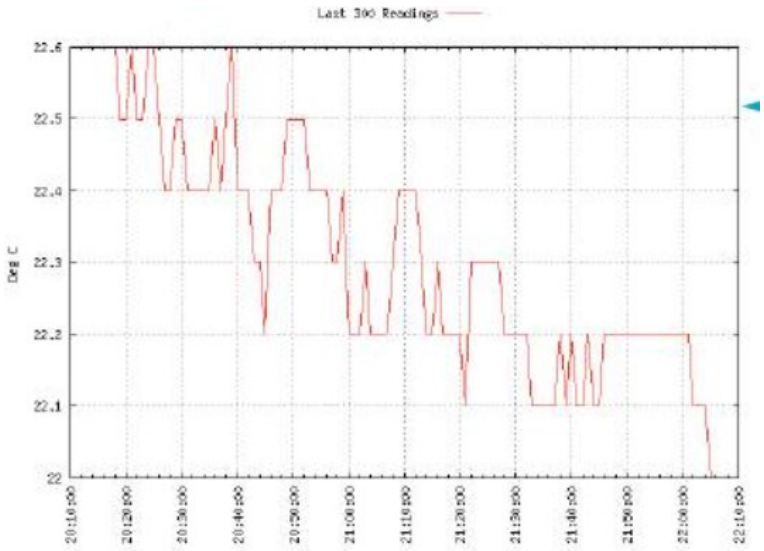
sudo nano /var/www/html/300.plt

```
set terminal png small size 640 480
set output “temp300.png”

set xtics rotate
set xdata time
set timefmt “%m-%d-%Y %H:%M”

set ytics
set grid
set ylabel “Deg D” # 0.00000, 0.00000 “”
set xlabel “\n\n”
set key above
plot ‘300.txt’ using 1:3 title “Last 300 Readings” with
lines
set output
```

Tallennus ja lopetus: CTRL+O ja CTRL+X

Kaavio näkyy selaimessa osoitteessa

http:// localhost/index.php


Last 300 Readings
Deg C
22.6
22.5
22.4
22.3
22.2
22.1
22
20:10:00
20:20:00
20:30:00
20:40:00
20:50:00
21:00:00
21:10:00
21:20:00
21:30:00
21:40:00
21:50:00
22:00:00
22:10:00

LAPIN AMK

Lapland University of Applied Sciences

DigiGo! Goes to Arduino Starter Kit

LAPIN YLIOPISTO
UNIVERSITY OF LAPLAND
Pohjoisen puolesta – maailmaa varten

LAPIN AMK
Lapland University of Applied Sciences

Vipuvoimaa EU:lta 2014–2020

Euroopan unioni
Euroopan sosiaalirahasto

OSA 4

DigiGo! Goes to Arduino Starter Kit

Tämä osa muodostuu syksyn 2017-2018 DigiGO! -hankkeessa vedettyjen koulutusten työohjeesta.

Mikä on Arduino?

- Opetuskäyttöön tarkoitettu mikro-ohjain alusta ja ohjelmointiympäristö https://fi.wikipedia.org/wiki/Arduino
- Eräs suomenkielinen oppikirja (Karvinen T. ja K.) http://www.sulautetut.fi/
- Arduino-projektin kotisivut https://www.arduino.cc/
- Aalto Yliopiston kurssi, jossa hyödynnetään Arduinoa https://mycourses.aalto.fi/course/view.php?id=13375
- Aalto Yliopiston materiaali / Peter Kronström https://mycourses.aalto.fi/pluginfile.php/99002/mod_page/content/44/Ohjelmointi-2014.pdf

Arduinon rakenne

- 5 voltin regulaattori
- 16 MHz kide
- USB-sarjamuunnin
- ATmega328 -mikrokontrolleri
  - 20 I/O-pinniä, joista 14 digitaalista ja 6 analogista

Lähde: https://mycourses.aalto.fi/pluginfile.php/99002/mod_page/content/44/Ohjelmointi-2014.pdf

- **DigiGo! Goes to Arduino Starter Kit**

Arduino-ohjelmointikieli

- Arduino-ohjelmointikieli on joukko C-kielen kirjastoja, jotka piilottavat taustalta monimutkaisempia C-kielen kutsuja

```
// Suoraan AVR-C:llä
PORTA = 0xFF;      // alustusarvo HIGH
DDRA = 0x80;       // aseta 8. pinni ulostuloksi

// Arduino-kirjaston avulla
DigitalWrite(8, HIGH);
```

- **DigiGo! Goes to Arduino Starter Kit**

Arduino-IDE

- Kaikki Arduinolla ohjelmoimiseen tarvittava tulee Arduino-kehitysympäristön (Arduino-IDE) mukana.
  - Sisältää mm. AVR C-kääntäjän, joka kääntää C-koodin konekielelle ja poistaa turhat osat
  - Hoitaa kommunikoinnin mikrokontrollerin ja tietokoneen välillä
  - Siirtää koodin sarjaväylän kautta mikrokontrolleriin
- Lisäksi mm. koodi-editori, kirjastonhallintatyökalu ja työkalu sarjaportin kuunteluun

Lähde:
https://mycourses.aalto.fi/pluginfile.php/99002/mod_page/content/44/Ohjelmointi-2014.pdf

DigiGo! Goes to Arduino Starter Kit

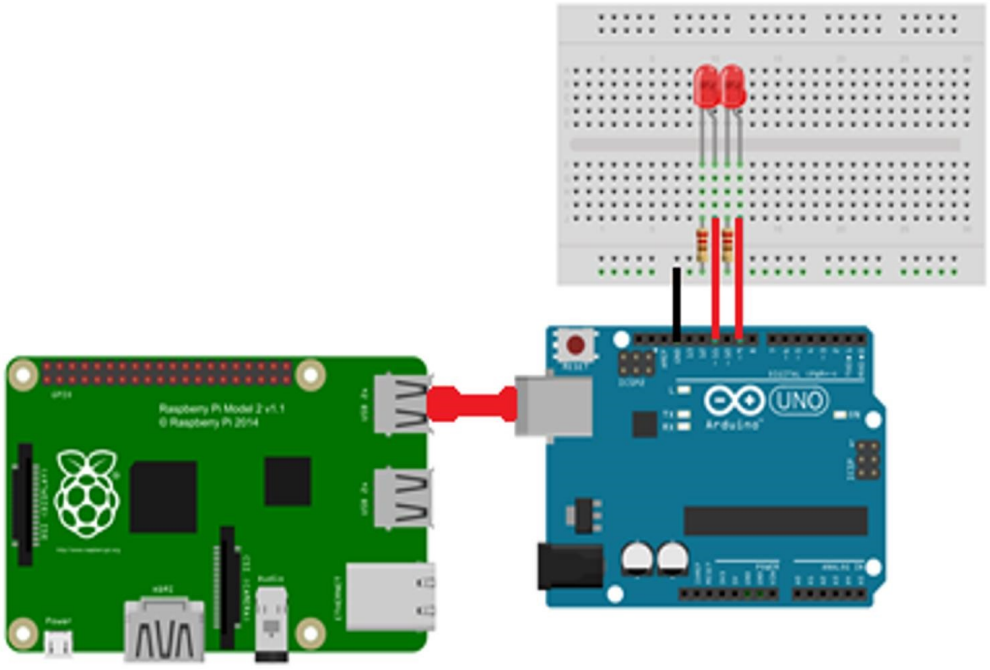
Arduino-ohjelman perusrakenne

```
#include <math.h>                       // Matemaattisia funktioita

int ledPin = 13;                        // Merkkivalo
int sensorPin = 4;                      // Anturi bensatankissa

void setup()                            // Suoritetaan vain kerran alussa
{
    pinMode(ledPin, OUTPUT);            // I/O-porttien alustus
    pinMode(sensorPin, INPUT);
}

void loop() // Suoritetaan kerta toisensa perään
{
    // analogRead palauttaa arvon välillä [0, 1023].
    int gasLevel = analogRead(sensorPin);
    int ledOn = gasLevel < 300; // LED sytytetään kun tankki lähes tyhjä
    digitalWrite(ledPin, ledOn);
}
```

Lähde:
https://mycourses.aalto.fi/pluginfile.php/99002/mod_page/content/44/Ohjelmointi-2014.pdf

DigiGo! Goes to Arduino Starter Kit

Breadboard elikkä näkkäri!

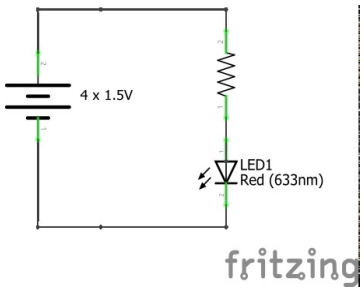


Vastusten värikoodit:
http://www.kouluelektroniikka.fi/vastus.html

Väri	*Kynnysjännite (V)*
Punainen	1,6 ... 1,7V
Vihreä	2,1 ... 2,2V
Keltainen	2,0 ... 2,1V
Oranssi	1,75 ... 2,2V
Sininen	3,6 ... 5V
Valkoinen	3,6 ... 5V

Infrapuna 1,2V

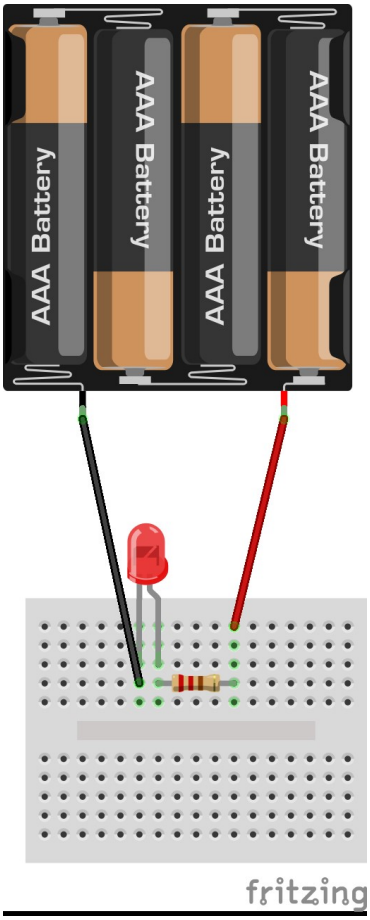
Esimerkiksi….



$$R_{etu} = \frac{4 \cdot U_{bat} - U_{kynnys}}{I_{kynnys}}$$

$$R_{etu} = \frac{4 \cdot 1.5V - 1.6V}{10mA}$$

$$R_{etu} = \frac{4 \cdot 1.5V - 1.6V}{0.01A} \approx 440\Omega$$



POHJOISTA TEKOA

LAPIN YLIOPISTO
UNIVERSITY OF LAPLAND
Pohjoisen puolesta – maailmaa varten

LAPIN AMK
Lapland University of Applied Sciences

Vipuvoimaa EU:lta 2014–2020

Euroopan unioni
Euroopan sosiaalirahasto

LAPIN AMK
Lapland University of Applied Sciences

www.lapinamk.fi

OSA 5
4xpainonappi

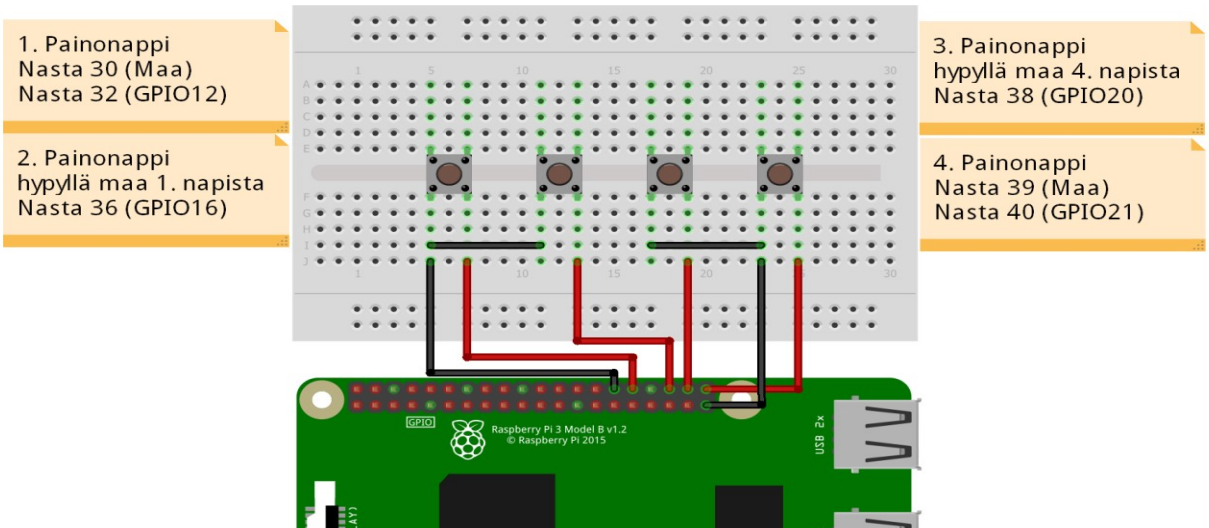
Tämä osa muodostuu syksyn 2018 ja kevään 2019 välisenä aikan DigiGO! -hankkeessa vedettyjen koulutusten työohjeesta.

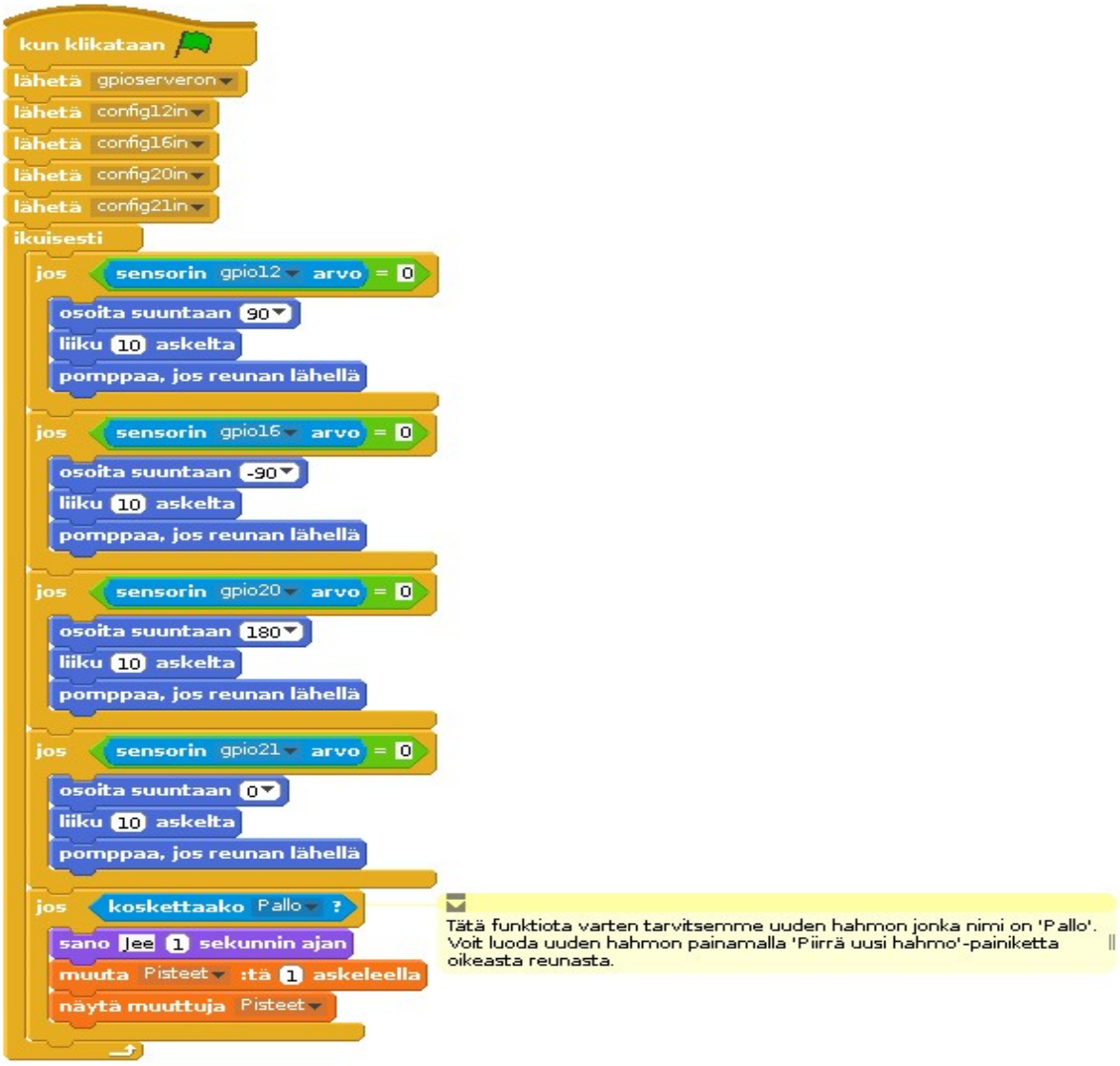
Harjoituksen ohje:

Harjoituksen tarkoituksena on tehdä Scratch-ohjelmointiympäristössä peli. Pelissä ohjataan painonapeilla hahmoa, joka syö palloa.

Laitteistona käytetään Raspberry pi3, jossa Scratch on valmiiksi asenettuna. Raspberry pi:hin on kytketty 7” kosketusnäyttö, mutta tarvittaessa voidaan käyttää ulkoista näyttöä+hiiri+näppäimistöä.

Painonapit kytketään koekytkentäalustalle ja Raspberry pi:lle GPIO kaapeleilla, kuten esimerkki kuvassa on kytketty. Kun kytkennät ovat valmiit, niin avataan Scratch-ohjelmointiympäristö Raspberry pi:n valikosta ja tehdään esimerkin mukainen koodi.




kun klikataan
lähetä gpioserveron
lähetä config12in
lähetä config16in
lähetä config20in
lähetä config21in
ikuisesti
jos sensorin gpio12 arvo = 0
osoita suuntaan 90
liiku 10 askelta
pomppaa, jos reunan lähellä
jos sensorin gpio16 arvo = 0
osoita suuntaan -90
liiku 10 askelta
pomppaa, jos reunan lähellä
jos sensorin gpio20 arvo = 0
osoita suuntaan 180
liiku 10 askelta
pomppaa, jos reunan lähellä
jos sensorin gpio21 arvo = 0
osoita suuntaan 0
liiku 10 askelta
pomppaa, jos reunan lähellä
jos koskettaako Pallo ?
sano Jee 1 sekunnin ajan
muuta Pisteet :tä 1 askeleella
näytä muuttuja Pisteet
Tätä funktiota varten tarvitsemme uuden hahmon jonka nimi on 'Pallo'. Voit luoda uuden hahmon painamalla 'Piirrä uusi hahmo'-painiketta oikeasta reunasta.


kun klikataan
ikuisesti
jos koskettaako Hahmo1 ?
odota 0.01 sekuntia
piilota
odota 3 sekuntia
mene kohtaan x: valitse satunnaisluku väliltä -220 - 220 y: valitse satunnaisluku väliltä -155 - 155
näytä

OSA 6

4xpainonappi ja 2xLed

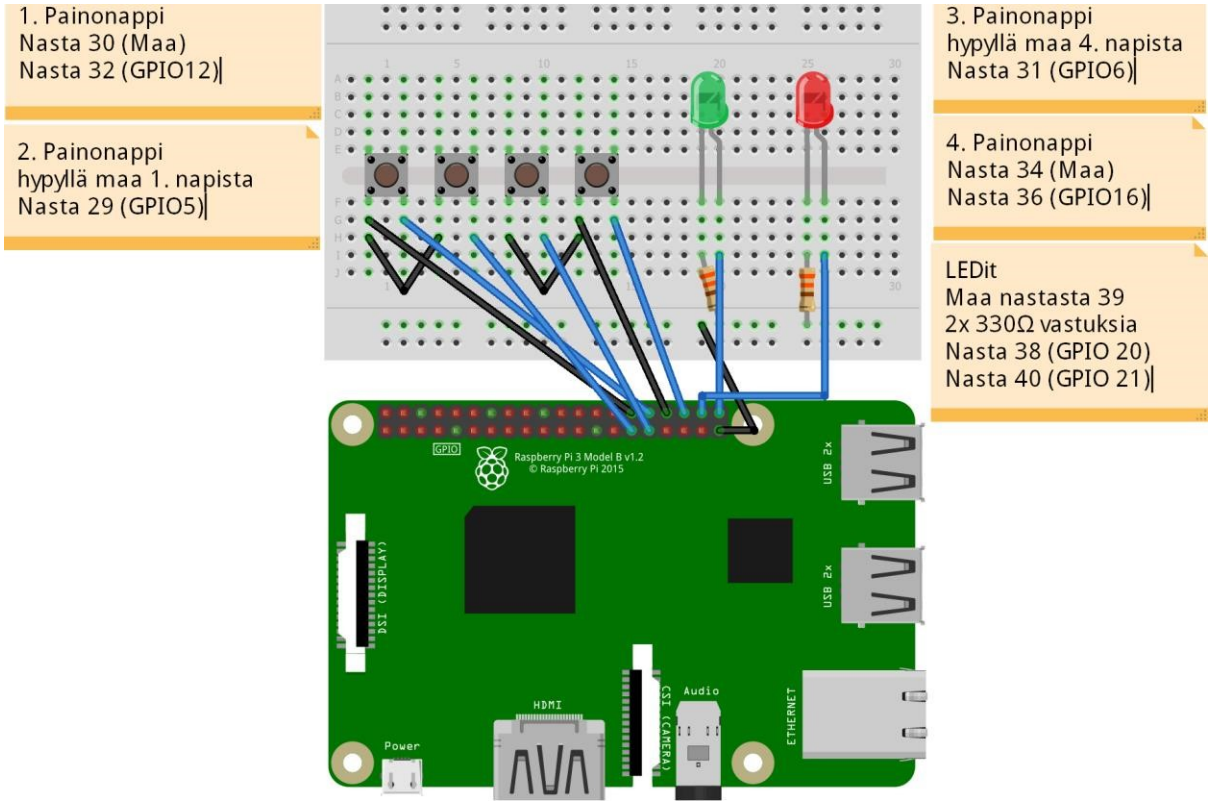
Tämä osa muodostuu syksyn 2018 ja kevään 2019 välisenä aikan DigiGO! -hankkeessa vedettyjen koulutusten työohjeesta.

Harjoituksen ohje:

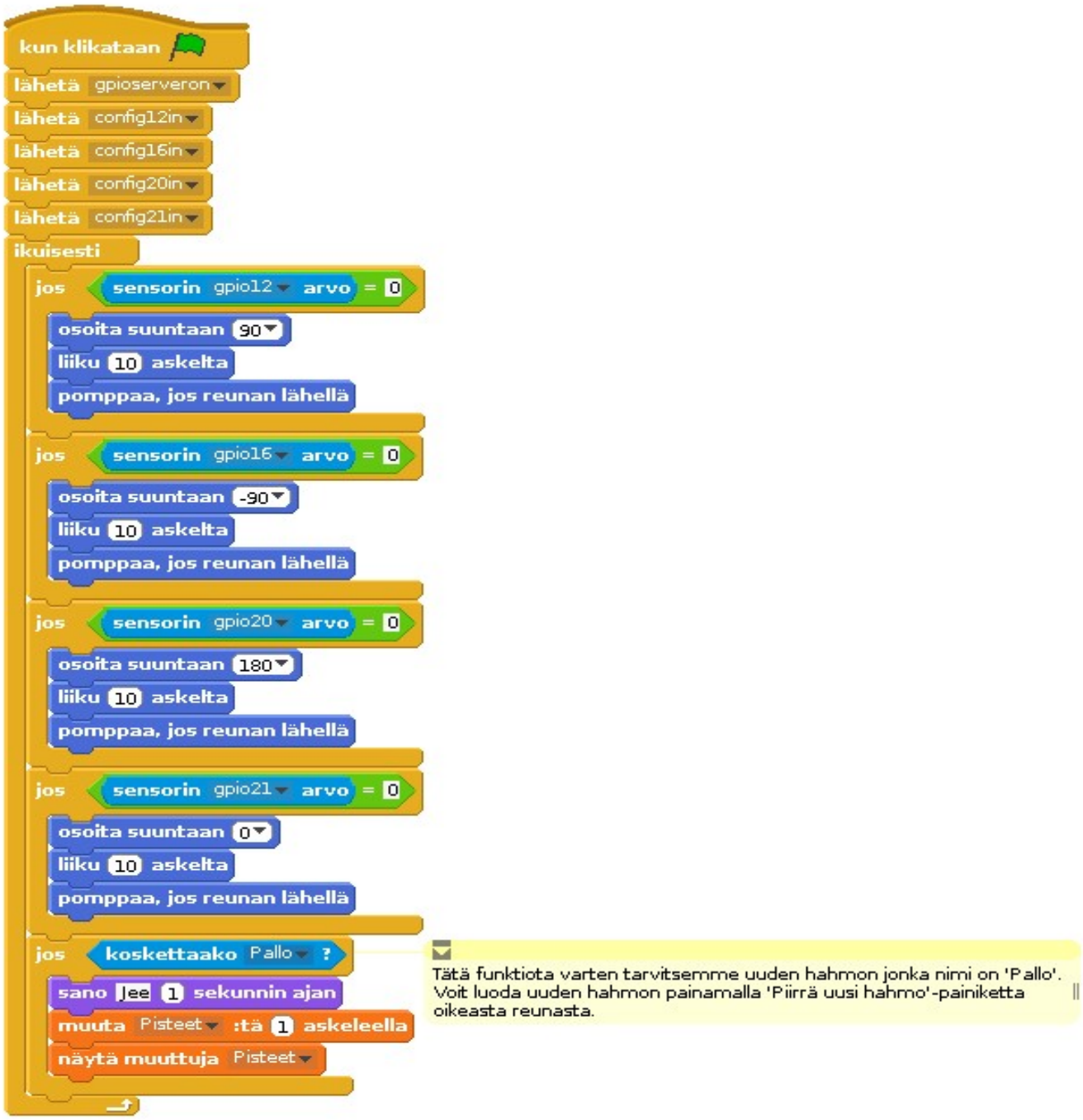
Harjoituksen tarkoituksena on tehdä Scratch-ohjelmointiympäristössä peli. Pelissä ohjataan painonapeilla hahmoa, joka syö palloa. Kun painonapeilla liikutaan oikealle tai vasemmalle, ledit syttyvät.

Laitteistona käytetään Raspberry pi3, jossa Scratch on valmiiksi asenettuna. Raspberry pi:hin on kytketty 7” kosketusnäyttö, mutta tarvittaessa voidaan käyttää ulkoista näyttöä+hiiri+näppäimistöä.

Painonapit ja Ledit kytketään koekytkentäalustalle ja Raspberry pi:lle GPIO kaapeleilla, kuten esimerkki kuvassa on kytketty. Kun kytkennät ovat valmiit, niin avataan Scratch-ohjelmointiympäristö Raspberry pi:n valikosta ja tehdään esimerkin mukainen koodi.


1. Painonappi
Nasta 30 (Maa)
Nasta 32 (GPIO12)
2. Painonappi
hypyllä maa 1. napista
Nasta 29 (GPIO5)
3. Painonappi
hypyllä maa 4. napista
Nasta 31 (GPIO6)
4. Painonappi
Nasta 34 (Maa)
Nasta 36 (GPIO16)
LEDit
Maa nastasta 39
2x 330Ω vastuksia
Nasta 38 (GPIO 20)
Nasta 40 (GPIO 21)
GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
DSI (DISPLAY)
HDMI
CSI (CAMERA)
Audio
ETHERNET
Power

fritzing


kun klikataan
lähetä gpioserveron
lähetä config12in
lähetä config16in
lähetä config20in
lähetä config21in
ikuisesti
jos sensorin gpio12 arvo = 0
osoita suuntaan 90
liiku 10 askelta
pomppaa, jos reunan lähellä
jos sensorin gpio16 arvo = 0
osoita suuntaan -90
liiku 10 askelta
pomppaa, jos reunan lähellä
jos sensorin gpio20 arvo = 0
osoita suuntaan 180
liiku 10 askelta
pomppaa, jos reunan lähellä
jos sensorin gpio21 arvo = 0
osoita suuntaan 0
liiku 10 askelta
pomppaa, jos reunan lähellä
jos koskettaako Pallo ?
sano Jee 1 sekunnin ajan
muuta Pisteet :tä 1 askeleella
näytä muuttuja Pisteet
Tätä funktiota varten tarvitsemme uuden hahmon jonka nimi on 'Pallo'. Voit luoda uuden hahmon painamalla 'Piirrä uusi hahmo'-painiketta oikeasta reunasta.


kun klikataan
ikuisesti
jos koskettaako Hahmo1 ?
odota 0.01 sekuntia
piilota
odota 3 sekuntia
mene kohtaan x: valitse satunnaisluku väliltä -220 - 220 y: valitse satunnaisluku väliltä -155 - 155
näytä

OSA 7
Lämpötilaohjattu tuuletin

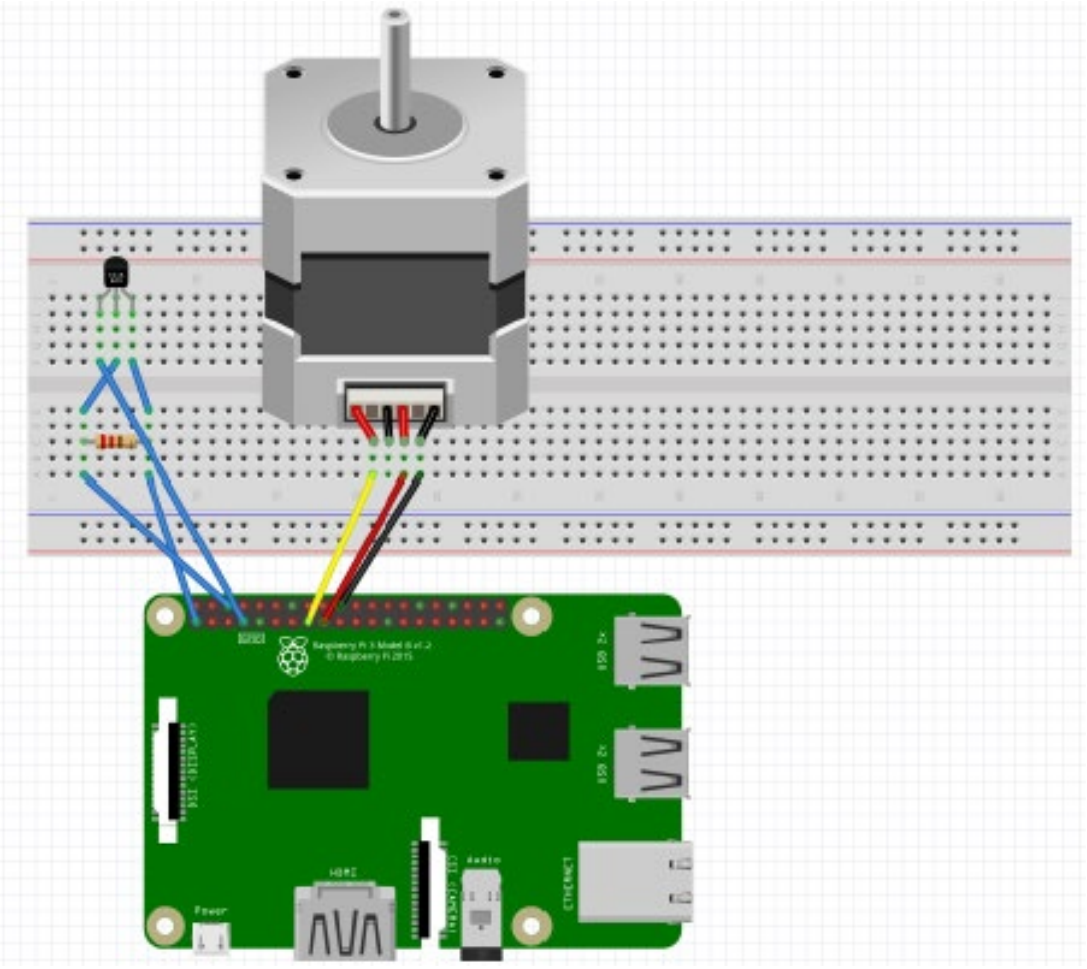
Tämä osa muodostuu syksyn 2018 ja kevään 2019 välisenä aikan DigiGO! -hankkeessa vedettyjen koulutusten työohjeesta.

Harjoituksen ohje:

Harjoituksessa käytetään Raspberry pi 3, tuuletinta, vastus (pullup resistor 4,7 Kohm) ja ds18b20 lämpötila-anturia. Ohjelmointiympäristönä käytetään Python-ohjelmointikieltä.

Tehdään kuvan mukainen kytkentä Rasperry pi:lle ja koekytkentäalustalle. Komponentit kytketään koekytkentäalustalle ja Raspberry pi:lle GPIO kaapeleilla. Kopioidaan valmis koodi python ohjelmointiympäristöön (Python3 Idle) ja suoritetaan koodi.

Lämpötila-anturin lämpötilan noustessa yli 26C asteen alkaa tuuletin toimimaan ja tuuletin pysähtyy lämpötilan tippuessa alle 26C.



```
import time
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode (GPIO.BCM)
GPIO.setup(22, GPIO.OUT)

while True:
        tfile = open("/sys/bus/w1/devices/28-0000079e78d5/w1_slave") text
= tfile.read()
        tfile.close
        secondline = text.split("\n") [1]
        temperaturedata = secondline.split (" ") [9]
        temperature = float(temperaturedata[2:])
        temperature = temperature / 1000
        print (temperature)
        if temperature > 26:
                GPIO.output(22, GPIO.HIGH)
if temperature < 26:
                GPIO.output(22, GPIO.LOW)
```