OHJELMAN PROTOTYYPIN

VAATIMUSMÄÄRITTELY

Taitaja9 sovellus

KEHITYSTYÖ

Kirjasto: C:\Users\matsk\OneDrive - Raision seudun koulutuskuntayhtymä

Tiedosto: Taitaja9-kilpailu R.H, A.M, D.T, J.O.docx

Versio: 0.1

Tila: luonnos [x]
ehdotus []
hyväksytty []

Muutoshistoria:
Versio Pvm. Tekijä Huomautus
0.1 23.8.2024 Rasmus

JAKELU: Projektin johtoryhmä
Jussi Kuosa
Rasmus, (Arthur, Dan, Joona ja Aldin)

LYHENTEET

MTBF (Mean Time Between Failures) on keskimääräinen aika laitteen vikaantumiseen sen edellisestä alkuperäiseen kuntoon saattamisesta (korjauksesta).

PC Personal Computer

SW Software

1. YLEISTÄ

Tässä dokumentissa on kuvattu vaatimukset taitaja9-kilpailuun.

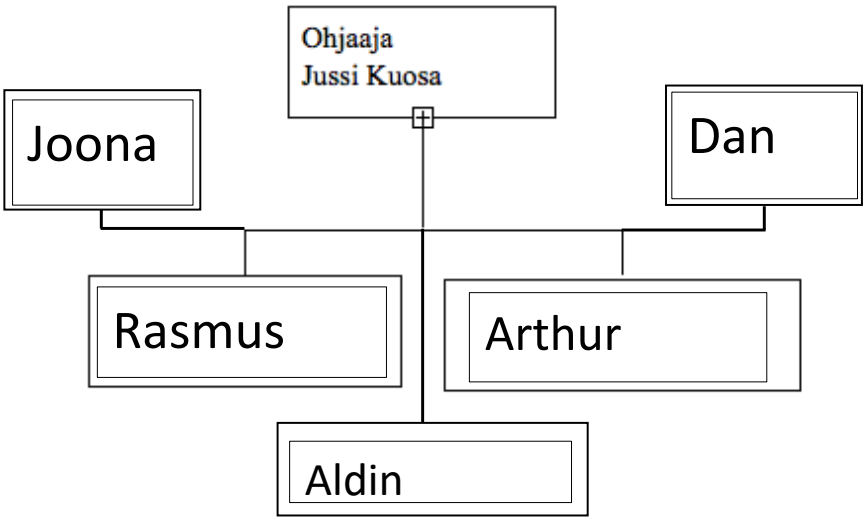
Tässä vaatimusmäärittelyssä otetaan kantaa myös taitaja9-kilpailun kehitysvaiheisiin ja vaatimuksiin.

Projektin tavoitteena on kehittää taitaja9-kilpailua.

[määrittele tähän, miten ohjelma toimii pääpiirteittäin. Määrittele tähän kenelle ohjelma tehdään. Määrittele tähän mitä ohjelmalla voidaan tehdä ja mitä se sisältää pääpiirteittäin].
Katso projektin speksit dokumentista viivojen kohtaan tarkennuksia.

Yleistä kappaleeseen tulee korkeintaan yksi A4 tekstiä (Hyvin lyhyesti kuvausta spekseistä, kirjoita tarkempi kuvaus Johdanto-kappaleeseen)

- Projektihenkilöstö



1. JOHDANTO / TOTEUTUSSUUNNITELMA

- vaatimusmäärittelyssä otetaan ensimmäisen kerran kantaa tuotteelle asetettaviin teknisiin vaatimuksiin
- vaatimusmäärittelyssä asiat esitetään asiakkaan kannalta nähtynä
- vaatimusmäärittelyssä asiat esitetään tavalla, joka on kenenkä tahansa ymmärrettävissä (formaalien kielten ja kuvaustapojen käyttöä harkittava tarkoin)
- Toteutussuunnitelmassa viitataan liitteessä olevaan projektiaikatauluun

Kommentoinut [JK1]: Katso spekseistä mistä järjestelmässä on kyse ja kirjoita omin sanoin kuvaus järjestelmästä ilman teknisiä yksityiskohtia

2. TOIMINNALLISET VAATIMUKSET (tiimissä toteutetut toiminnot)

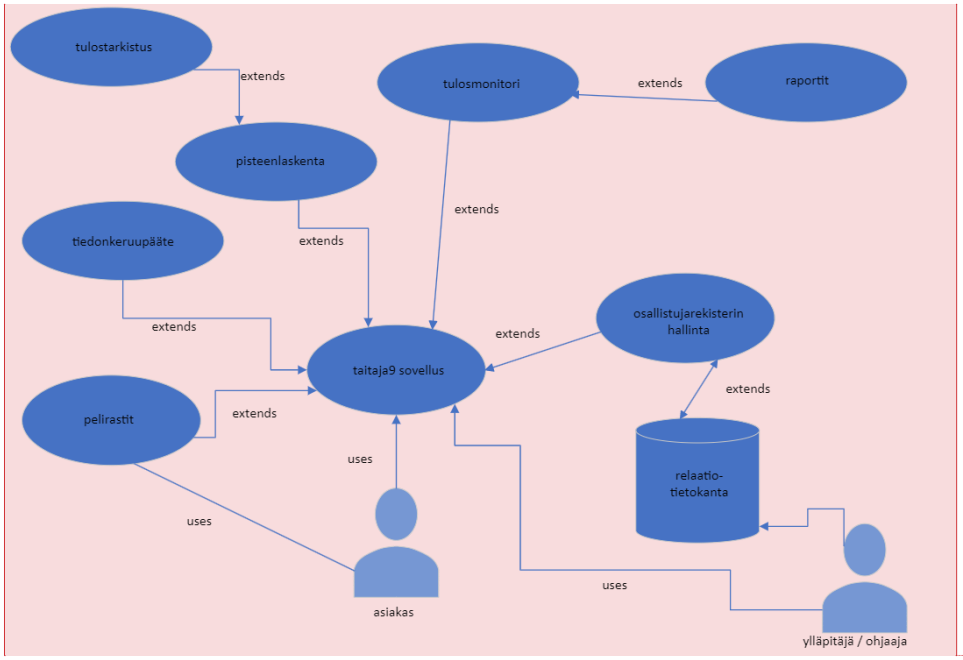
- toimintokuvaus selkokielellä (sanalliset kuvaukset toiminnoille)
- Jokainen Taitaja9-projektiin osallistuva kuvaa oman projektiosuutensa sanallisen toimintakuvauksen tähän kappaleeseen (tarkennettu kuvaus ja vuokaavio tulee kohtaan "yksityiskohtainen suunnittelu")

Kommentoinut [JK2]: Listaa tähän alla olevasta pallurakaaviosta toiminnot ja kirjoita mitä mikin toiminto tekee. Jos et tiedä mitä toiminto tekee niin kirjoita toiminnon kuvauken kohtaan että määritellään myöhemmin

3. ARKITEHTUURI (Use Case)

- Määrittele tähän millaisia Java Paketteja, kirjastoja ja luokkia ohjelma sisältää. Millaisia käyttäjiä ohjelmistolla on. Tässä voidaan esitellä käyttötapauskaavio ja luokkakaaviot.

- Use Case kaavio
- Jokainen Taitaja9-projektiin osallistuva lisää oman projektiosuutensa lisätyn toiminnon Use Case- kaavioonja lisää päivitetyn kuvan tähän kappaleeseen
- Jos lisätty toiminto on laajennus jo olemassa olevaan toimintoon niin se kuvataan "exclude" nuolella ja jos se on lisätty ominaisuus kuten tietorakenne niin se kuvataan "include" nuolella olemassa olevaan toimintoon (toiminnot kuvataan ellipsillä).



Kommentoinut [JK3]: Kirjoita kuvateksti UseCase (ohjelmiston arkkitehtuuri)

4. YKSITYISKOHTAINEN MÄÄRITTELY (toiminnalliset kuvaukset)
Määrittele tähän millaisia toimintoja ohjelma sisältää.
Esimerkiksi Taitaja9– ohjelma sisältää käyttöliittymän josta käyttäjä voi syöttää rekisteröintitietoa osallistuviin joukkueisiin liittyen (kuvattu esim. vuokaaviolla).

- Yksityiskohtainen suunnittelu (Vuokaaviot Use Casen toiminnoille)
- Jokainen Taitaja9-projektiin osallistuva lisää oman projektiosuutensa lisätyn toiminnon yksitiskohtaisen kuvauksen vuokaaviona tähän kappaleeseen

Huomioitavat asiat kuvauksessa:

- Tietorakenteet (kuvaukset tauluista ja kyselyistä sekä csv- ja txt-tiedostot, ja raportit)

Muut määriteltävät asiat:

- ohjelmointiympäristön määritely
- parametrit, tietoliikenne, protokollat
- liityntöjen määritykset

Määrittele tähän kaikki ohjelman liitynnät esim. Jos ohjelmassa on tietokanta- liityntä tms. (käyttöliittymät on kuvattu myöhemmin omassa kappaleessaan)

- uudelleenkäytettävyys suunnittelussa
- toimintaan liittyvät standardit

5. KÄYTETTÄVYYSVAATIMUKSET

- käyttöliittymän vaatimukset

esim. Jos ohjelmassa on web- liityntä ja Java UI-liityntämoduli

- Käyttöliittymän määritys
- Asennusvaatimukset

6. TESTAUSRAPORTTI (testauskoodit)

- Taitaja9-projektissa testausvastuussa oleva lisää seuraavat asiat tähän kappaleeseen. Jos testausvastaavaa ei ole tiimissä niin kukin täydentää tätä kappaletta omalta osaltaan.
- Testaussuunnitelma ja vuokaavio
- Junit Test Case kuvakaappaukset liitteeksi
- Yksikkötestit (vuokaaviot)
- Aja koodi ja laita ajotuloksista kuvakaappaukset tähän kappaleeseen

7. LIITTEET

PROJEKTIAIKATAULU
Gant- kaavio tai muu vastaava (Määrittele tähän Excelillä tekemäsi projektin aikataulu)

TOTEUTUSSUUNNITELMA TUOTANTOVERSIOLLE

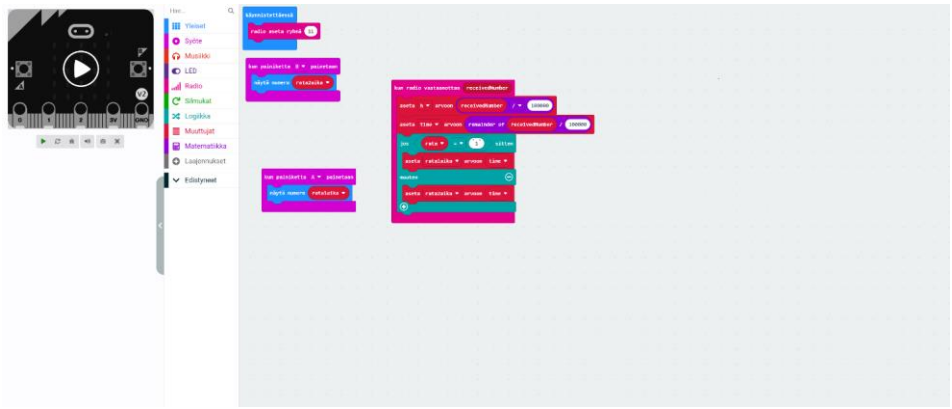
- Tavoitteet projektille
- Työnjako tiimin sisällä, kuka vastaa mistäkin toiminnosta
- Aikataulun päivitys (gant-kaavio)
- Toiminnallisten vaatimusten päivitys
- Arkkitehtuurin päivitys
- Yksityiskohtaisen määrittelyn päivitys
- Käytettävyys / käyttöliittymän päivitys
- Testaussuunnitelman päivitys

Koodit:

Lisää liitteeseen Notepad++ ohjelman avulla koodiin rivinumerot seuraavasti.
a) asenna Notepad++, asenna AcrobatReader
b) kopioi koodi Notepad++ ohjelmaan
c) valitse koodi ctrl-a ja poista rivitoiminnolla tyhjät rivit
d) printtaa koodi pdf-tiedostoon
e) valitse koodi pdf-tiedostosta ctrl-a, jolloin saat rivinumerot mukaan koodiin
f) liitä koodi, jossa on rivinumerot liitteeseen Loppuraporttiin

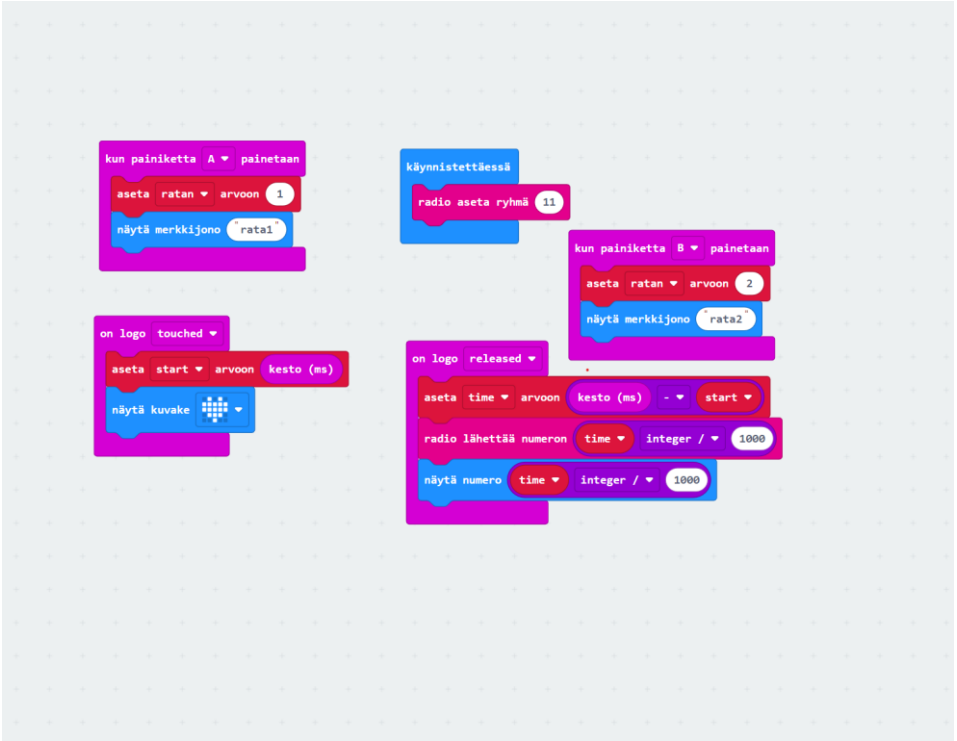
```
def on_received_number(receivedNumber):
    global h, time, rata1aika, rata2aika
    rata = ""
    h = receivedNumber / 100000
    time = receivedNumber % 100000
    if rata == "1":
```

```
        rata1aika = time
    else:
        rata2aika = time
radio.on_received_number(on_received_number)

def on_button_pressed_a():
    basic.show_number(rata1aika)
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_button_pressed_b():
    basic.show_number(rata2aika)
input.on_button_pressed(Button.B, on_button_pressed_b)

rata2aika = 0
rata1aika = 0
time = 0
h = 0
radio.set_group(11)
```

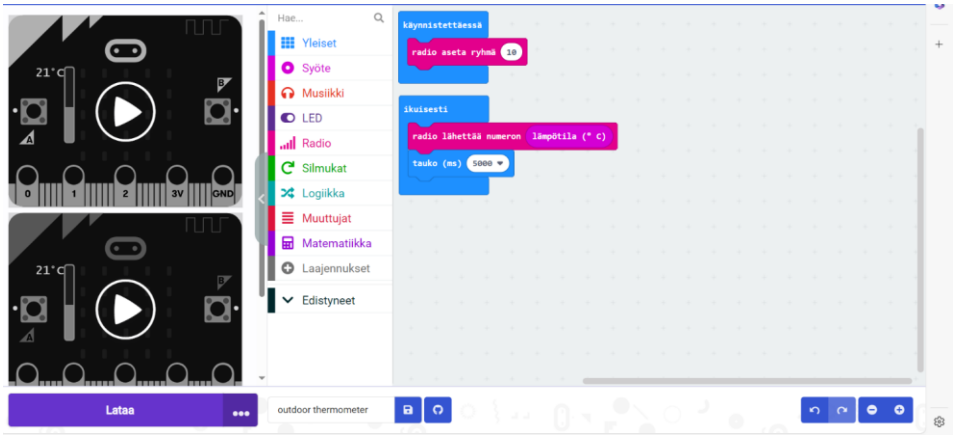


```
def on_button_pressed_a():
    global ratan
    ratan = 1
    basic.show_string("rata1")
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_button_pressed_b():
    global ratan
    ratan = 2
    basic.show_string("rata2")
input.on_button_pressed(Button.B, on_button_pressed_b)

def on_logo_touched():
    global start
    start = input.running_time()
    basic.show_icon(IconNames.HEART)
input.on_logo_event(TouchButtonEvent.TOUCHED, on_logo_touched)

def on_logo_released():
    global time
    time = input.running_time() - start
    radio.send_number(Math.idiv(time, 1000))
    basic.show_number(Math.idiv(time, 1000))
input.on_logo_event(TouchButtonEvent.RELEASED, on_logo_released)
```

```
time = 0
start = 0
ratan = 0
radio.set_group(11)
```



```
radio.set_group(10)

def on_forever():
    radio.send_number(input.temperature())
    basic.pause(5000)
basic.forever(on_forever)
```




```
from microbit import *
import radio


rata_1_time = 0
rata_2_time = 0


radio.set_group(11)
radio.on()
```

```
def on_received_number(receivedNumber):
    global rata_1_time, rata_2_time
    rata = receivedNumber // 100000
    time = receivedNumber % 100000

    if rata == 1:
        rata_1_time = time
    elif rata == 2:
        rata_2_time = time

radio.on_received(receivedNumber=on_received_number)


def on_button_a_pressed():
    display.scroll(rata_1_time)


def on_button_b_pressed():
    display.scroll(rata_2_time)


button_a.was_pressed(on_button_a_pressed)
button_b.was_pressed(on_button_b_pressed)
```