IoT ja sulautetut järjestelmät – ohjelmoinnin harjoitustehtävät

IoT ja sulautetut järjestelmät -moduulin ohjelmointiosion harjoitustehtäviä. Kaikkia tehtäviä ei ole pakko tehdä, jo osan tekeminen vaikuttaa arvosanaan nostavasti. Osa tehtävistä on tehtävänannoltaan hyvin laajoja, ja jos laaja tehtävänanto tuntuu liian työläältä, sitä voi myös vapaasti rajata itse.

Tehtäviä lisätään dokumenttiin myöhemmillä kursseilla, mutta myös nykyisiä tehtäviä saa jatkaa ja palauttaa myöhemmilläkin kursseilla.

Lisää harjoitustehtäviä voi etsiä verkosta, esimerkiksi hakusanalla "random project idea generator" tms. Yksinkertaisia ohjelmointiesimerkkejä ja harjotustehtäviä löytyy myös valmiina paketteina, kuten *w3resource.com:n harjoitustehtävät.*

- https://www.w3resource.com/c-programming-exercises/

1. kirjoita ensimmäinen c-ohjelmasi

Tehtävänanto

Kirjoita C-kielellä ohjelma, joka tulostaa Pico:lla tekstin "Hello, Pico!" oletusulostuloon (stdout).

Haluttu lopputulos

Ohjelman tulee kääntyä ja suorittua ongelmitta, kääntäminen ja esimerkkiajo:

```
# Ohjelman suoritus alkaa
Hello, Pico!
# Ohjelman suoritus päättyy
```

2. teksti-I/O

Tehtävänanto

Kirjoita C-kielellä ohjelma, joka lukee Pico:lla terminaalista käyttäjän antamat kaksi kokonaislukua, ja suorittaa niille käyttäjän haluaman aritmeettisen operaation (yhteen-, vähennys-, kerto-, jakolasku tai jakojäännös).

Ohjelman tulee raportoida käyttäjän virheistä (numerot, joita ei voida käyttää, virheelliset laskuoperaatiot, tyhjä syöte) omavalintaisilla virheilmoituksilla. Virheellinen syöte ei saa kaataa ohjelmaa.

Ohjelman lopettamiseen on kirjoitettava vapaavalintainen komento.

Haluttu lopputulos

Ohjelman tulee kääntyä ja suorittua ongelmitta, kääntäminen ja esimerkkiajo:

```
# Ohjelman suoritus alkaa
6 + 6
12
5 * 5
25
10 / 3
3
4 +
Virheellinen syöte (puuttuvia argumentteja)

Virheellinen syöte (tyhjä) kirjoita "q" lopettaaksesi ohjelman
q
# Ohjelman suoritus päättyy
```

3. muistinhallinta

Tehtävänanto

Laajenna edellisen tehtävänannon ohjelmaa siten, että ohjelma tallentaa viimeisimmät 10 operaatiota, ja tulostaa komentohistorian käyttäjän syöttäessä valitun komennon. Komento `h` tulostaa historian, komento `c` tyhjentää historian.

Komentohistoria on tulostettava rivinumeroin tyyliteltynä, komento !numero (esim. `!2`) toistaa historiassa kyseisellä rivinumerolla olevan laskun.

Haluttu lopputulos

Ohjelman tulee käyttää historiaa varten dynaamista muista (`malloc()/free()`) ja vapauttaa varaamansa muisti oikeaoppisesti. Mahdolliset muistivuodot tarkistetaan ohjelmallisesti.

```
# Ohjelman suoritus alkaa
6 + 6
12
5 * 5
25
10 / 3
3
h
Komentohistoria:
1: 6 + 6
2: 5 * 5
3: 10 / 3
!2
5 * 5 = 25
c
h
```

```
Komentohistoria on tyhjä.
q
# Ohjelman suoritus päättyy
```

4. GPIO – taskulampun ohjaus

Tehtävänanto

Kirjoita ohjelma, joka lukee vapaavalintaisesta pinnistä painonapin tilan, jolla ohjataan Picoon kytkettyä LED-valoa. LED-valoa voidaan ohjata napilla seuraavasti:

- ensimmäinen painallus kytkee valon päälle
- toinen painallus himmentää valoa (pwm duty cycle 25%)
- kolmas painallus vaihtaa valon tilan vilkkuvaksi (vilkkumisen taajuus vapaavalintainen)
- neljäs painallus sammuttaa valon uudelleen

Haluttaessa kirjoittaa kehittyneempi versio, voi ohjelmaan lisätä muistiominaisuuden. Tällöin valo voidaan sammuttaa mistä tahansa tilasta pitkällä (> 3s) painalluksella, jolloin yksi painallus käynnistää valon samaan tilaan kuin aiemminkin.

Huomioitavaa on, että painonapin antama signaali voi kohista – ohjelman on toimittava mahdollisesta kohinasta huolimatta oikein (debouncing).

Haluttu lopputulos

Ohjelman tulee toimia edellä kuvatulla tavalla, eikä painonapin mahdollisesti kohiseva signaali saa haitata normaalia käyttöä.

Ohjelma saa tulostaa diagnostiikkatietoja sarjaportin kautta, mikäli kyseinen ominaiuus nähdään tarpeelliseksi.

Kohinanpoiston tapa on vapaavalintainen, mutta sen on oltava toimiva.

5. Kehittynyt I/O ja kirjastot

Tehtävänanto

Kirjoita kirjasto jonkin vapaavalintaisen asian tekemiselle, sekä esimerkki kirjaston käytöstä. Kirjasto voi olla esimerkiksi jonkin anturin tai laitteen käyttöön, aiemmissa tehtävissä käytetyn ominaisuuden muuttaminen kirjastoksi, tai mikä tahansa muu toiminto, jonka muokkaaminen kirjastoksi on mielekästä.

Esimerkki-ideoita: * Lämpötilasensorin tekstin muotoilu valmiiksi * Yksinkertainen tilatietotkanta Pico:lle * Sekuntikello (ajanottofunktiot, valmiit tulostusfunktiot niiden muotoilemiseen) * Luonnontieteen vakioiden määrittely sopivassa kirjastossa

Pyörän saa halutessaan keksiä uudelleen, eli kirjaston ei tarvitse olla mikään uusi ominaisuus, tai edes hyödyllinen. Pääasia on oikea rakenne, jossa kirjasto on oma erillinen cmake-projektinsa, jota toinen projekti hyödyntää kirjastona.

Haluttu lopputulos

Ohjelman tulee olla kokonaisuus, johon kuuluu kirjasto, sekä sitä hyödyntävä esimerkkiohjelma. Ohjelman tulee kääntyä ja suorittua ongelmitta, ja esimerkkiohjelmassa ja -kirjastossa on oltava toimintaa ja käyttötarkoitusta kuvaava `README`-tiedosto.

Ohjelman tulee hyödyntää jotain Picon IO-ominaisuutta, esimerkiksi GPIO-pinnejä tai sarjaporttia.