Turun ammattikorkeakoulu

Matlab-harjoitustyöt 2015, Matematiikka 4

Jarmo Kivekäs 1302928

21. maaliskuuta 2015

Sisältö

1 Johdanto

Tämä raportti käsittelee robottia joka toteutettiin harjoitustyönä Ohjelmoinnin perusteet kurssia varten. Harjoitustyön tarkoituksen oli oppia järjestelmäläheistä ohjelmointia ARM sekä AVR ympäristäissä. Raportin sekä siin esitetyn materiaalin on laatinut Jarmo Kivekäs.

2 Järjestelmä arkkitehtuuri

3 Käyttöliittymä

Käyttöliittymä robotin ojausta varten toteutettiin RaspberryPi -alustalla. Robotin ohjaus käyttöliittymän kautta tapahtuu kokonaan RPI:n liitetyn ulkoisen USB näppäimistön kautta.

3.1 USB käyttöliittymälaitteiden tulkitseminen

Näppäimisän kautta annetut käskyt luetaan erityisestä `/dev/input/eventX` tiedostosta.

4 Ohjaus teoria

Robotin toteuttamista varten sovellettiin useaa eri mateemaattista mallinnusta robotin oletetusta käyttäytymisestä. Mallit phojautuvat hyvin tunnettuihun ohjausmalleihin joiden oikea toimivuus on näyhty käytännön sovelluksissa jo ennestään.

4.1 Kineettinen malli

Mekaanisesti robotti on rakennettu niin, että liikkuminen toteutetaan ensisijaisesti kahdella laitteen sivuilla sijaitsevan renkaan avulla. Robotti liikkuu käyttäen kahta ns. differentiaalista ajo-moottoria. Käytännössä tämä tarkoittaa sitä, että sivulla olevia moottoerita voidaan ohjeta toisitaan riippumatta. Robotti kääntyy kun renkaat pyörivät eri nopeutta suhteessa toisiinsa. Lisäksi robotilla on vapaasti liikkuvia tukipyöriä jotta se pytyy tasapainossa.

Robotin ohjaamista varten käytetty matemaattinen malli on seuraavanlainen:

$$\begin{cases} \dot{x} = \dfrac{R}{2}(v_r + v_l)\cos(\phi) \\ \dot{y} = \dfrac{R}{2}(v_r + v_l)\sin(\phi) \\ \dot{\phi} = \dfrac{R}{L}(v_r - v_l) \end{cases} \tag{4.1}$$

Robotin oletetaan kulkevan tasaisen tason pinnalla. x ja y kuvaavat laitteen paikkaa tasolla, ϕ robotin etuosan osittamaa suuntaan, v_r ja v_l ovat oikean sekä vasemmanpuolisen renkaan pyörimisnopeudet. Lisäksi mallissa esiintyy vakio R joka on ohjaukseen käytettyjen renkaiden säde. Vakio L on renkaiden etäisyys toisistaan.

—

Yllä esitelty malli toimii hyvin ohjausta varten. Ohjausalgorimejä kehittäessä päädyttiin kuitenkin siihen tulokseen, että järjestelmää kannattaa käsitellä yksinkertaisemalla mallilla. Robotin liikkeitä on hankala hahmottaa ajattelemalla pelkästään renkaiden pyörimisnopeutta.

Ohjausalgoritmien kehittämistä varten käytetty matemaattinen malli on seuraavanlainen:

$$\begin{cases} \dot{x} = v\cos(\phi) \\ \dot{y} = v\sin(\phi) \\ \dot{\phi} = \omega \end{cases} \tag{4.2}$$

Mallin avulla voidaan määrittää robotin liikke käyttäen hyväksi pelkästään sen nopeutta v sekä kulmanopeutta ω.

Soveltamalla malleja (4.1) ja (4.2) saadaan korrelaatio robotin translaation sekä rotaation (v ja ω) ja renkaiden pyörimisnopeuksien (v_r ja v_l) välille:

$$\begin{cases} \dfrac{vR}{L} = v_r + v_l \\ \dfrac{\omega L}{R}(v_r - v_l) \end{cases} \tag{4.3}$$

4.2 Ohjaus silmukan malli

PID -ohjain

5 Kommunikointi protokolla

Robotti vastaanottaa käskyjä, ja lähettää sensoreista luettua tietoa takaisin sarjaväylän yli.

Sarjaväylän langaton tiedonsiirto on toteutettuu 2.4 GHz pakettiradio moduuleilla. Moduulien tiedosiirtokyky on riittävän luotettava, että

6 Oppitulosket

Projektin tarkoituksena oli syventyä järjestelmäläheiseen C-kilen ohjelmointiin AVR sekä ARM prosessoriarkkitehtuureilla ja yelisemmin kielen eri ominaisuuksiin.

USB käyttöliitymälaitteiden raa'an datan tulkistemista varten ei löytynyt kovin perusteellista oppimateriaalia. Suuri osa työskently- ja oppimisprosessia koostui eri asiaankuuluvien C otsaketiedostojen lukemisesta, koska tietoa ei suurikaan muualta löytynyt helposti. – Muita kiinnostavia oli esimerkiski keskiä millä datatyypillä AVR:n erikoistoimintorekisterit (SFR) ovat esitetty. Esitytavan tunteminen oli olennaista jotta pystyi esimerkiksi kirjoittamaan funktioita joille annetaan I/O -nasta argumenttina.

```
volatile uint8_t *
```

– `Struct` tietorakentiden vertaileminen eri arkkitehtuureilla oli myös kiinnostavaa. RPI:n 32-bittin ARM arkkitehtuuri pyrkii asettamaan datan muistiosotteisiin jotka ovat neljällä jaollisia. 8-bittinen AVR ei kuitenkaan muistiosoitteien jaollisuudesta välitä. Tämä johtaa siihen, että structiin pakattua dataa ei voida turvallisesti siirtää suoraan ARM arkkitehtuurilta AVR arkkitehtuurille ilman että erikseen vamistetaan että tietorakenteiden esitys todella on sama molemmilla arkkitehtuuriella. Kirjoittaessa koodia jota on tarkoitus suorittaa molemilla arkkitehtuureilla pitää pitää mielessä että myös datatyyppien koot eroavat toisistaan. AVR:n `int` muuttuja on 16-bittinen, kun 32-bittisen ARM:n datatyyppi on 32-bittinen. Yleispätevää koodia saa kuitenkin kätevästi kirjoitettua käyttämällä `<inttypes.h>` määrittämiä datapyyppejä kuten `uint8_t` ja `int32_t` – printf puskurointi

- POSIX
- ioctl