Robotinohjaussovellus

1 Lyhyesti

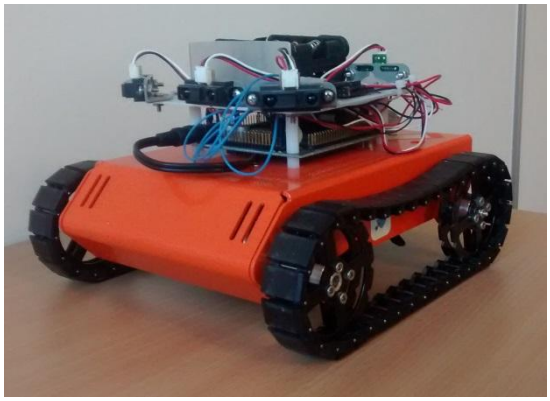
Harjoitustyön aiheena on toteuttaa ohjaussovellus itsenäisesti liikkuvalle robotille. Harjoitustyöhön kuuluu toteutuksen lisäksi sovelluksen määrittely, suunnittelu ja testaaminen sekä eri vaiheiden asianmukainen dokumentointi. Robotin laitteiden käyttöä varten on erityinen luokkakirjasto, jonka myötä ohjelmointi tapahtuu korkealla abstraktiotasolla.

2 Laitteen perustiedot

Traxster II on RoboticsConnectionin valmistama alusta robottien rakentamista varten. Runkoon on kiinnitetty erilaisia antureita ja toimilaitteita robotin käyttämiseksi. Robotin ohjaaminen tapahtuu Windows 10 IoT Core -alustalla, jota ajetaan Raspberry PI -kortilla.

Robotissa on seuraavat laitteet. Niitä käytetään tarkoitukseen kehitetyn DLL-kirjaston kautta.

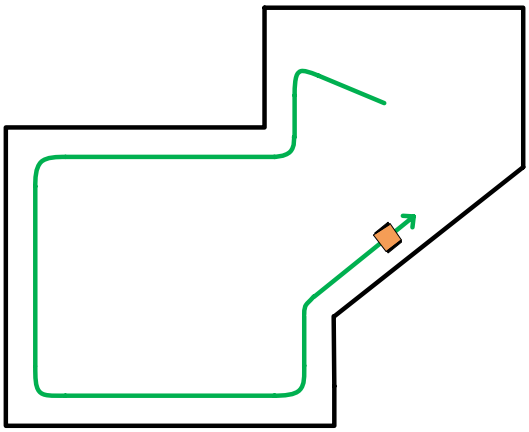
Ajomoottori	2 kpl	7,2 V
Etäisyysanturi (infrapuna)	4 kpl (3 eteen, 1 taakse)	Kantama 10–80 cm, tarkka minimikantaman lähellä; kapea keila
Painonappi	2 kpl	
Led	2 kpl	Vihreä



Robotti.

3 Työn suorittaminen

Robotille kirjoitetaan navigointiohjelma, jolla se pystyy liikkumaan itsenäisesti suljetussa tilassa seiniä seuraten. Voidaan olettaa, että ”pistemäisiä” esteitä (ohuita tuolinjalkoja tms.) ei ole, sillä niiden havaitseminen robotin antureilla on likipitäen mahdotonta.



Ideaalisesti toimivan robotin kulkua suljetussa tilassa.

4 Työn vaatimukset

Seuraavat vaatimukset toimivat *toteutuksen* arvostelun perustana.

4.1 Toiminnalliset vaatimukset

1. Robotti voidaan sekä laittaa liikkeelle että pysäyttää nappia painamalla niin monta kertaa kuin halutaan.
2. Robotti osaa etsiä seinän, mikäli sellaista ei aluksi antureilla näy.
3. Robotti osaa seurata suoraa seinää smoothisti.
4. Robotti huomaa etusektorilta mahdollisesti lähestyvän seinän ja osaa kääntyä vähitellen sen suuntaisesti (seinän sisäkulma).
5. Robotti suoriutuu myös tilanteesta, jossa sivulla ollut seinä yhtäkkiä loppuu (ts. seinän ulkokulma).
6. Robotin ei tarvitse koskaan pysähtyä (ts. moottoreita käytetään jatkuvasti) pl. hätäseistila.
7. Sovelluksen pitää toteuttaa *hätäseistila*, joka aktivoituu, jos este on jossakin suunnassa liian lähellä. Hätäseistilaan joutunut robotti on voitava saada uudelleen liikkeelle nappia painamalla.
    - Hätäseistila ei ole sovelluksen normaalia toimintaa vaan *turvatoiminto* esim. sovellusvirheen tai eteen juoksevan lapsen tai poron varalta.
8. Robotti kerää toiminnastaan lokia.
    1. Kun robotti on pysäytetty, lokin sisältö voidaan tulostaa sarjaporttiin nappia painamalla.
    2. Lokista pitää käydä ilmi ainakin syntyneet virhetilanteet.
    3. Jos virheitä ei ole tullut, myös siitä pitää tulostua tieto.

4.2 Muut vaatimukset

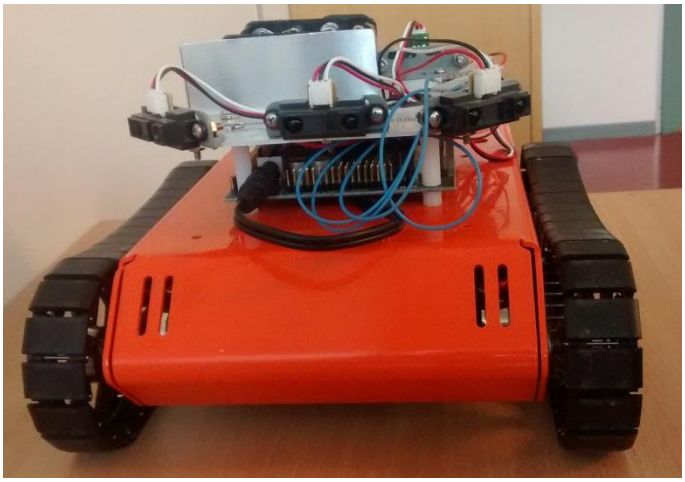
9. *Keskinäinen poissulkeminen* (mutual exclusion) toteutetaan sellaiselle datalle, jota käytetään useammasta kuin yhdestä säikeestä.
10. Anturilukemien *tarkkailu* hätäseisturvatoimintoa varten toteutetaan omassa säikeessään.
11. Infrapuna-anturien raakalukemien käsittely toteutetaan *itse.*
    1. Anturien luku tapahtuu ”taustalla” omassa säikeessään.
    2. Häiriöiden vuoksi dataa on suodatettava (esim. rengaspuskurilla).
    3. Anturiarvoja ei lueta suoraan, vaan arvoille tehdään välimuisti.
12. Robotin *järjestelmätestaus* suoritetaan ajamalla ohjelmaa itse robotilla.

5 Vinkkejä suunnitteluun

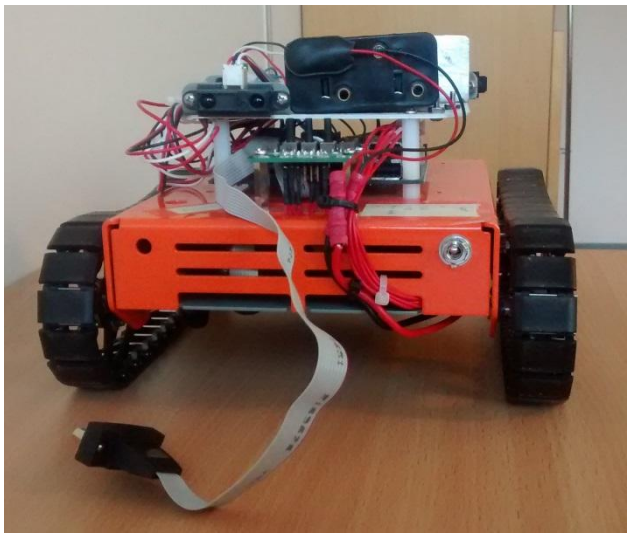
- Robotin kyvyt havainnoida ympäristöään ovat rajalliset, ja toisaalta ”älykkäiden” algoritmien toteuttaminen on työlästä ja vaikeaa.
- Kannattaa siis suhtautua sovelluksen vaatimusmäärittelyyn minimalistisesti.
- P-säädön käyttäminen voi olla viisasta, jotta robotti seuraa seinää sulavasti.

Toteutukseen liittyvät vinkit on kerätty vinkkidokumenttiin, joka tulee saataville erikseen. Laitteiston ohjelmointirajapinta on erillisessä dokumentissa.

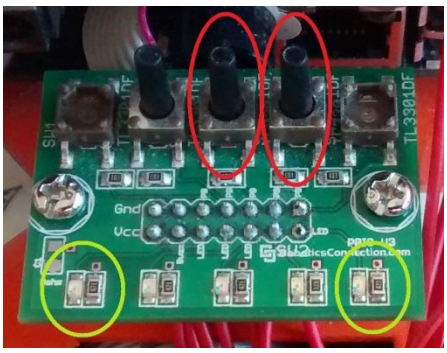
6 Kuvia



Edestä.



Takaa.



Lähikuva PBIO-kortista. Käytössä olevat napit on merkitty punaisella, ledit vihreällä.