Comunicatie met de iRobot Roomba platform met Raspberry Pi

Introductie

Het communiceren tussen de PC en de Roomba gebeurt als volgt:

- Op de PC, staat een MATLAB toolbox. Deze bevat functies zodat er op een gebruiksvriendelijke manier commando's naar de Roomba kunnen gestuurd worden.
- Deze toolbox is gebaseerd op de iRobot Roomba 500 Open interface (datasheet van iRobot zelf) en een oude toolbox om de iRobot Create 1 te besturen.
- De toolbox zal de correcte bytes sturen over UDP, naar een Rasberry Pi (RPi) die verbonden is met de Roomba.
- De RPi is dus een "simpele" reapeter. Deze zal "klakkeloos" via seriële communicatie doorsturen wat hij binnen krijgt. Hij weet ook wanner data moeten gelezen worden van de seriële poort en kan deze dan ook terug naar de PC zenden (ook over UDP).

Concrete locatie van de files, op de Pi

Alles betreffende de communicatie met de Roomba en de PC bevind zich in ~/Pi/server (gewoon cd /server/) bij inloggen. De python code "udp_receive.py" wordt uitgevoerd bij het opstarten van de RPi. Deze maakt gebruikt van een python bibliotheek, "pici.py".

Problemen & mogelijke oplossingen

Python2.7 : string over UDP

Een oorspronkelijk probleem tijdens het programmeren van de het zenden over UDP is dat Python2.7 eist dat de data onder "string" vorm verzonden wordt. Dit gaf, **oorspronkelijk problemen onder Linux:** bij het ontvangen van data, onder stringvorm, kon de PC onder Windows correct de data vertalen van 'strings' naar bytes. Onder Linux niet : deze kon maar de helft vertalen : d.w.z. dat wanneer deze decimale waardes (hiermee bedoel ik dus de decimale representatie van een byte), d.m.v. encoderen, hoger waren als 127, kreeg ik een "". Na wat onderzoeken, ben ik tot de conclusie gekomen dat de Python code aan de ene hand, en de MATLAB onder Linux niet dezelfde manier van encoders en decoderen hadden.

Ik heb dus een pragmatische oplossing gehanteerd : in plaats van efficiënt deze 2 bytes door te sturen als string, ga ik over elke decimale waarde van die byte, en stuur deze door als string. Een voorbeeld om dit duidelijk te maken :

Roomba stuurt "128"(DEC) door. Deze zou normaalgezien als "d" volgens ASCII/UTF8 table vertaald moeten worden. Voor het te versturen, verander ik dit in "1" "0" "0", zodat ik **nooit** decimale waardes van strings heb boven de 127.

Tragere data transmissie door pragmatische oplossing

In eerste instantie leek dit een goed werkende oplossing. Onder Linux werkt de data ontvangst. Onder deze OS ben ik blijven werken. Tot mijn verbazing werkte opeens mijn code niet meer onder windows werkt. Het "niet werken" bleek na veel testen niet waar te zijn. Een betere forumulatie is :

De UDP-time out tijd bij het krijgen van UDP data was getuned voor Linux : na 0.2sec wachten gaat het verder, omdat de code dan verondersteld niks meer te krijgen. Ask-receive testen hebben dan ook onder Linux een goede resultaat : 99% van de pakketten komen onder de 0.2s na EN zijn geldig. Onder Windows komt na 0.4sec 20% van de pakketten aan EN zijn geldig. De andere 80% zijn "verloren".

Trage ask-receive met uitgeleende laptops

Tijdens de labo zittingen is het duidelijk geweest voor mij dat de intensieve string manipulaties die ik doe voor een "trage"/"normale" PC duidelijk het proces vertragen. 12Hz met mijn laptop, 6Hz met uitgeleende laptops (nog steeds onder Linux natuurlijk)

Python3 migratie: een voor de hand liggende oplossing ?

Een van de grote veranderingen onder Python3 is dat men nu over UDP bytes moet sturen. Dit is exact wat met nodig heeft ! Maar de bibliotheek van de iRobot Roomba communicatie onder python is niet geschreven voor Python3 moet dus grondig aangepast worden.

Python3 op RPi

Distro "Wheezy" ondersteunt niet volledig Python3

Python3 is maar deels ondersteund onder de RPi distro die momenteel op de RPi draait. Versie op die RPi : Debian Wheezy. Actuele, met een groter python3 bibliotheek mogelijkheden : Debian Jessie. Dit betekend enkel dat met **NIET** met $ sudo apt-get install python3-naamBibliotheek aan een bepaalde bibliotheek kan geraken, maar dat met deze moet instaleren door de source via internet de downloaden, en dan sudo python3 install naamBibliotheek uit te voeren, in de ge-unzipte source file.

Stand van zaken (op de RPi waar Roomba2 staat)

Door pics.py (de simulator) uit te schakelen in pici.py (de communicatie bibliotheek) kan men met de python 3.1.* werken die op de Pi staat (en die nog ondersteund word door de Distro).

Het is me gelukt om AL de bibliotheken te importeren. Ik heb zoals hierboven uitgelegd PySerial via de source code moeten compileren (https://github.com/pyserial/pyserial).

Ik heb daarna zoveel mogelijk syntax error proberen op te lossen. Deze waren eerst in onze udp_receive code. Het waren enkel de print functies, die vanaf Python3 deze, minder vrije, syntax moeten hebben: print(). Eenmaal de compiler met de communicatiebibliotheek begon, werd het moelijker voor mij, ik had vanaf dat moment te weinig tijd om mij in deze debugging te verdiepen.

Een oplossing waardoor een nieuwe zou ontstaan

Vanaf het moment dat de RPi onder Python3 draait en de laptops ook, zullen we ook in de MATLAB de "mini-toolbox" van python moeten veranderen naar Python3. Hier heb ik al een potentieel probleem gevonden : laat de command line toe om non-string values te krijgen ? Ik kreeg hierdoor een rare error in mijn MATLAB terminal (deze heb ik jammer genoeg niet opgeslagen).

Wat als het nog steeds niet werkt onder Windows ?

Indien deze oplossing niet werkt hebben we nog steeds een stap vooruit gedaan. Er zal VEEL minder string manipulatie gedaan worden en hierdoor zal de code veel rapper uitgevoerd worden. Zo kunnen we naar een veel hogere communicatie frequentie gaan, die momenteel vrij laag is.