Open Source Robotics Foundation

Robot Operation System

Tom Mingneau, Pieter Kellens

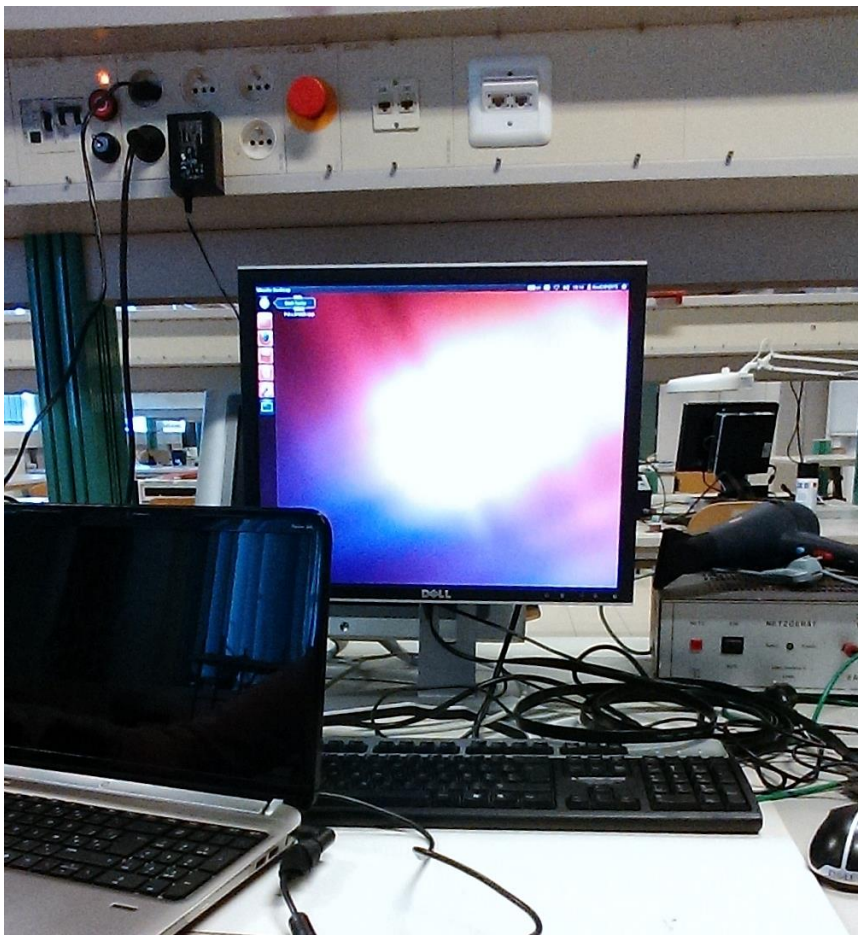
Inhoudsopgaven

Inleiding

Het doel van ons project is om via ROS een robot aan te sturen. Er is dan gekozen voor het Pandaboard om hier ROS op te installeren omdat dat al eens gelukt was en dat hier ook meer informatie over te vinden is. Eerst hebben we een ARM versie van Ubuntu 14.04 LTS op dit Pandaboard geïnstalleerd. Daarna hebben we de ROS Hydro installatie tutorial gevolgd omdat deze de enige nieuwste versie is die op ons Pandaboard wilde werken. Hierna hebben we een klein testprogramma uitgeprobeerd om te kijken of ROS werkte en voor onszelf om ROS te leren kennen. Nu we weten dat ROS werkt kunnen we ons richten op het aansturen van een transportkar. Dit gebeurt via een seriële verbinding tussen het Pandaboard en twee motor controllers die op de transportkar zitten. Om via ROS een seriële verbinding te kunnen maken hebben we een ARM versie van ROSserial gevonden die we dan geïnstalleerd hebben. ROS pin 23 en 24 werken via I²C seriële communicatie. Dan hebben we een klein programma dat we gevonden hadden en herschreven geprobeerd te testen met de transportkar maar dit werkte niet. Hierna hebben we ondervonden dat we best een Arduino aan onze ROS konden koppelen zodat de Arduino de code uitvoerd maar ROS zegt welke waarde er doorgestuurd moet worden. De Arduino is gekoppeld aan ROS via de USB poort en de Arduino is gekoppeld aan de MD03 via I²C. Hier hebben we dan een programma voor herschreven uit verschillende sources.
In de rest van onze paper vind je informatie over de verschillende elementen van ons project.

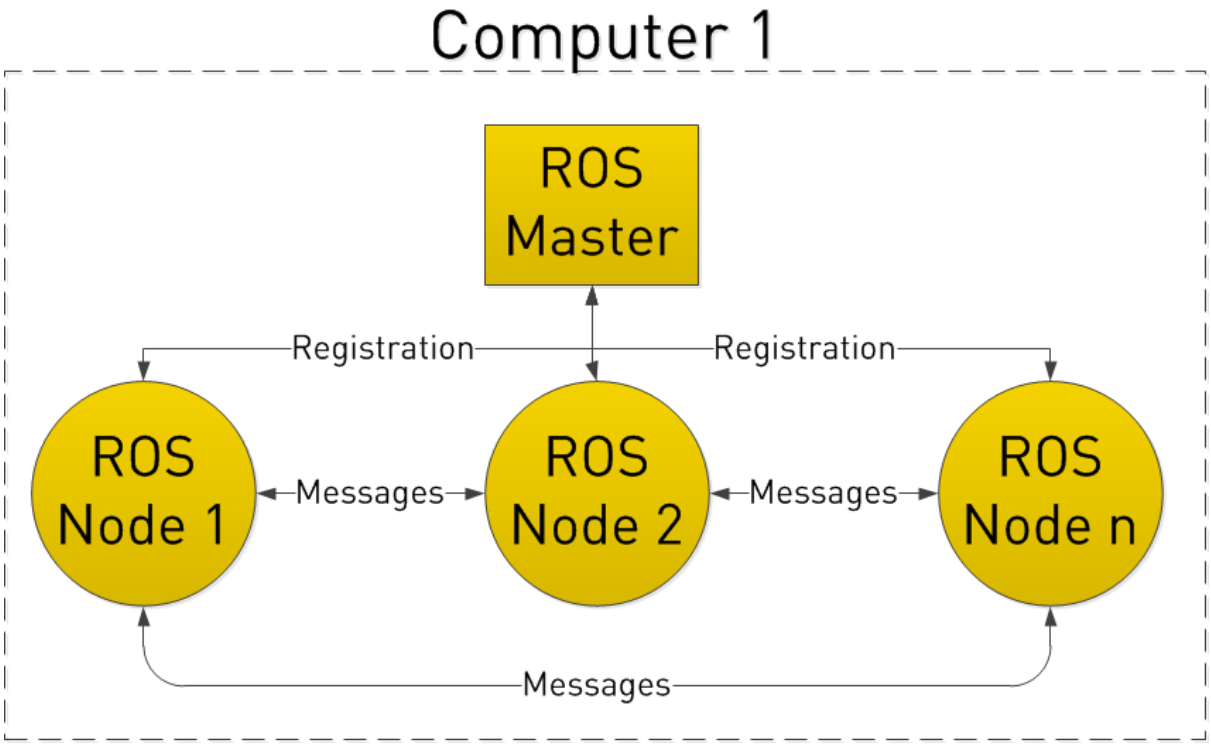
Hardware

Om via een microcontroller te kunnen programmeren hebben we verschillende externe onderdelen nodig. We hebben een Pandaboard ES REV B1 gebruikt als microcontroller. Hierop hebben we een toetsenbord, muis en scherm aangesloten. Omdat het scherm een DVI poort had gebruikte we een DVI naar HDMI tussenstuk zodat het scherm aangesloten kan worden met een HDMI kabel op de HDMI out van het Pandaboard. Dan hebben we het Pandaboard ook nog internet toegang gegeven met een ethernet kabel. Dan is er nog een SD kaart gebruik van 16GB om de software op te plaatsen zodat de microcontroller dit kan inlezen. En als laatste is er de adapter van 5V 2A die ons Pandaboard voedt. Hier ziet u onze opstelling.



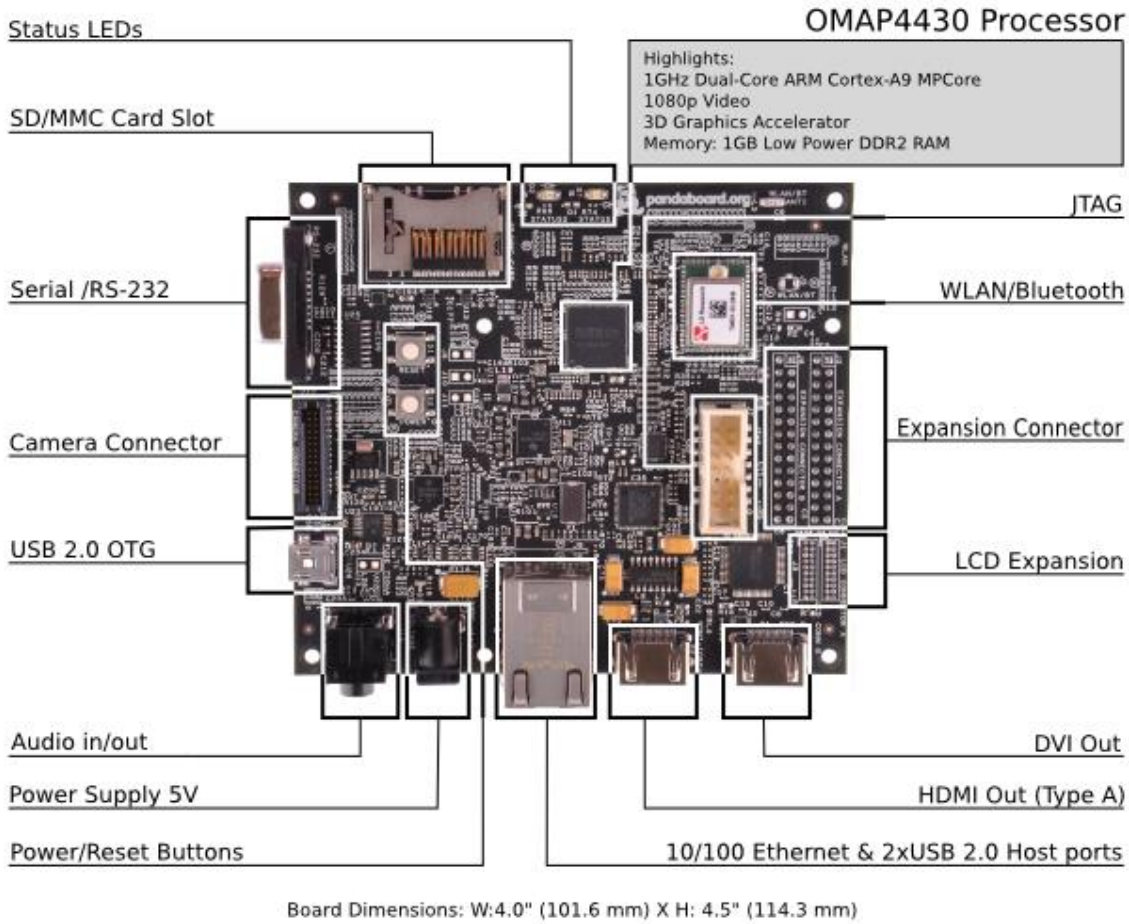
ROS

Ros is een open-source besturingssysteem om robots mee aan te sturen. het systeem biedt hardware-abstractie , low-level apparaat-controle, boodschappen doorgeven tussen processen en taakbeheer.Het biedt ook gereedschappen en bibliotheken voor het verkrijgen, het bouwen, het schrijven en het uitvoeren van code op meerdere computers.
ROS biedt ook een uitstekende communicatie- infrastructuur die meerdere stijlen van communicatie tussen allerlei verschillende apparaten ondersteunt.ROS is geen real-time framework hoewel het in sommige gevallen mogelijk is real-time code te integreren.



Pandaboard

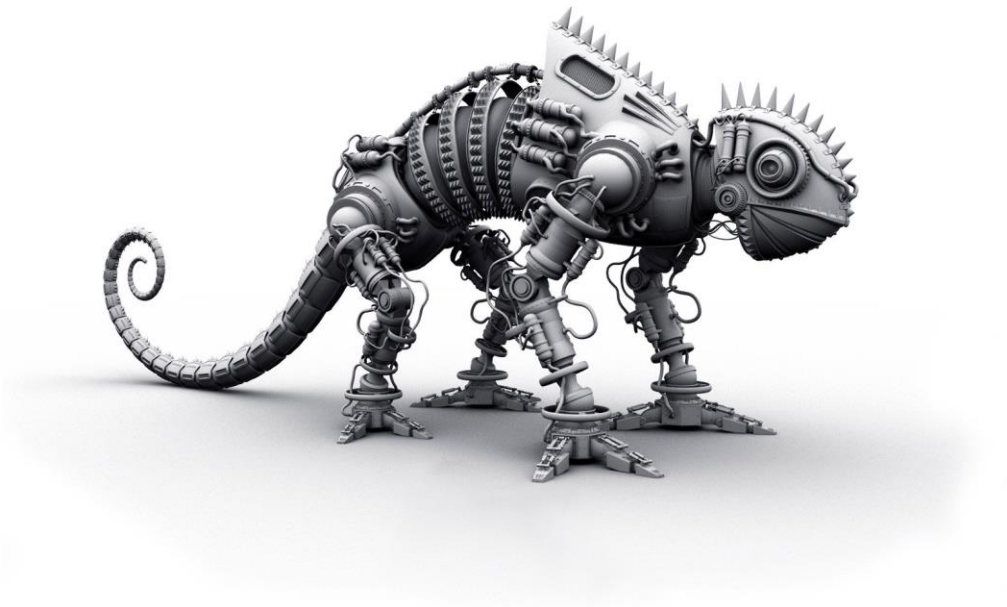
Het PandaBoard is een single-chip ontwikkelingscomputer met een lage kost. Het ontwikkelingsplatform werd gebaseerd op de Texas OMAP4430 dual core processor.Bij de Pandaboard ES versie die wij gebruiken draaien de CPU en GPU op een hogere kloksnelheid.
Het toestel draait de LINUX-kernel met ofwel de traditionele varianten (ubuntu) of android of mozzila firefox OS.
Op dit bordje hebben wij dan ubuntu geïnstalleerd en daarna ROS. Dit heeft veel tijd en moeite gekost omdat we vaak de verkeerde tutorials volgden.



Ubuntu

ubuntu is een volledig gratis open-source besturingssysteem dat gebaseerd is op linux. Het is gebaseerd op linux-distributie Debian maar onderscheidt zich hiervan door 2 keer per jaar een nieuwe versie te brengen.

Aangezien wij met het pandaboard werkten dat compatibel is met ubuntu hebben we gewerkt met dit besturingssysteem. Ook ROS is het meest compatibel met ubuntu.
De versie die wij gebruikten is wel een oudere versie namelijk ubuntu precise pangolin 12.04 LTS. Deze gebruikten we omdat de nieuwere versies niet werkten met ROS op een ARM als het pandaboard. De nieuwere versies hebben we eerst wel uitgeprobeerd maar we zijn tot de conclusie gekomen dat de deze niet ondersteund werden door ROS. Hierdoor hebben we dan ook weer tijd verloren.



Testprogramma

Het programma wat wij schreven in ROS is het testprogramma “hello world”.

Dit hebben wij als volgt gedaan :

1. Installatie van ROS via de tutorial op de ROS wiki : http://wiki.ros.org/hydro/Installation/UbuntuARM

2. Toevoegen van de binaire opslagplaats voor ROS packages. Hierdoor kunnen we voorgecompileerde packages van ROS gebruiken.

3. Installeren van de basis ROS packages

4.Daarna maakten we een directory aan die we hello_world_tutorial noemden.

5. We volgden een simpel C++ programma dat de ROS library gebruikt.

```
// toevoegen ROS library

#include <ros/ros.h>

// Standaard C++ entry point

int main(int argc, char** argv) {

// Kondigt het programma aan aan de ROS-master als een node genaamd : ‘hello_world_node"

ros::init(argc, argv, "hello_world_node");

// start van de hulpbronnen voor de node (communicatie,tijd, etc)

ros::start();

// uitzenden van een simpele tekst

ROS_INFO_STREAM("Hello, world!");

// opvolgen van ROS callbacks tot er een intelligent signaal gestuurd wordt.

ros::spin();

//Stop de hulpbronnen van de node

ros::shutdown();
// Uitzetten van ROS
return 0;
```

6. daarna compileerden we het programma en kwamen er een aantal errors. Deze konden we oplossen door op een ROS setup file het commando source uit te voeren.

```
source /opt/ros/hydro/setup.sh
```

7. nog een error : ROS kan niet communiceren met de ROS-master. Dit kan men oplossen door ROSCORE in te geven in de command-line. Dit commando zorgt ervoor dat de ROS-master opstart.

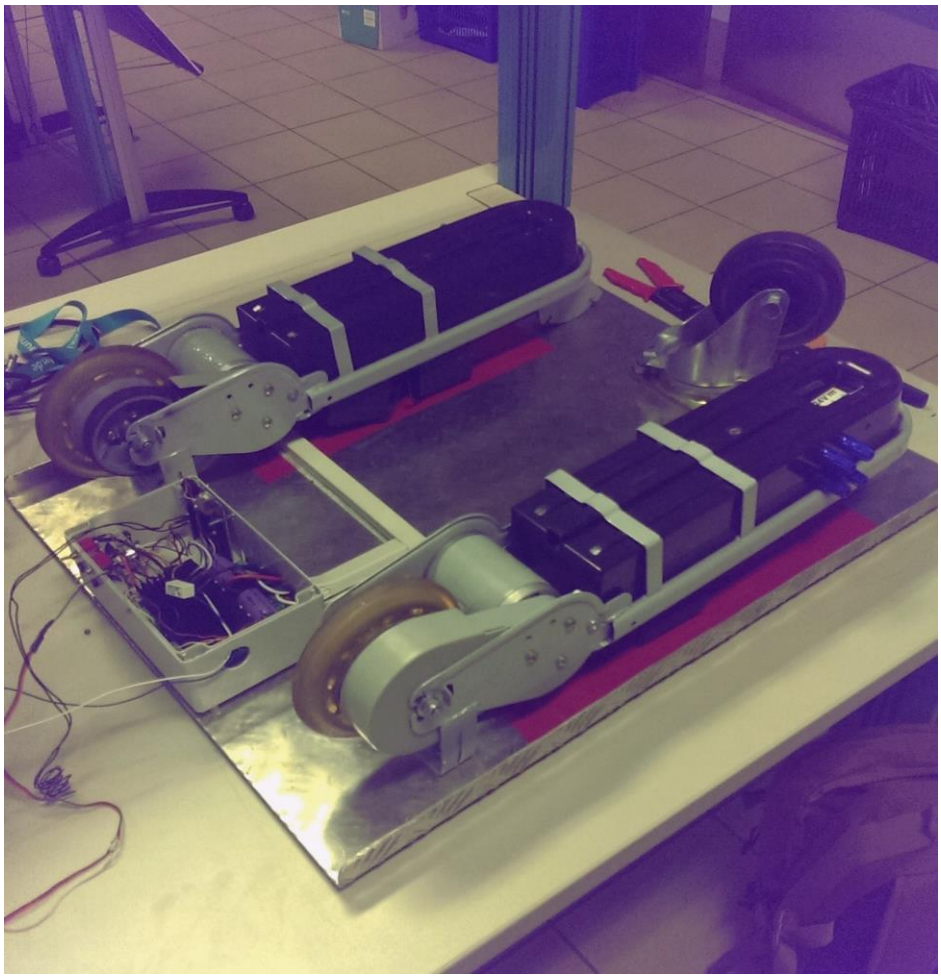
8. daarna compileerden we het programma opnieuw en werkte het perfect.

```
[ INFO] [1392020655.967763511]: Hello, World!
```

Bron : http://jbohren.com/articles/roscpp-hello-world/

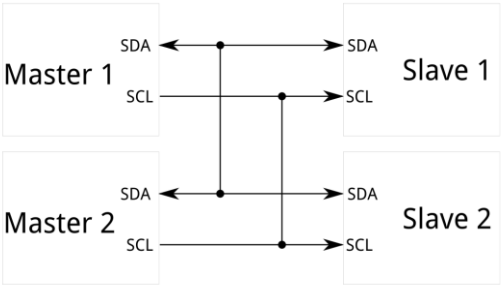
Transportkar

De transportkar bestaat uit 4 batterijen die de twee DC motoren aansturen en de nodige onderdelen die de kar aansturen. De twee DC motoren kunnen elke apart aangestuurd worden omdat elke motor een aparte MD03 controller heeft. MD03 is een motor controller die via verschillende modes een motor kan aansturen. De MD03 controller hebben wij aangestuurd via een seriële communicatie I^2C en afhankelijk van hoe zijn schakelaars staan verandert het adres dat aangestuurd word. In deze seriële communicatie bevinden zich de parameters hoe snel de motor moet versnellen, de snelheid van de motor en of hij vooruit of achteruit moet aangestuurd worden. Maar er kan ook informatie uitgehaald worden over de motor zoals de stroom, acceleratie of temperatuur.

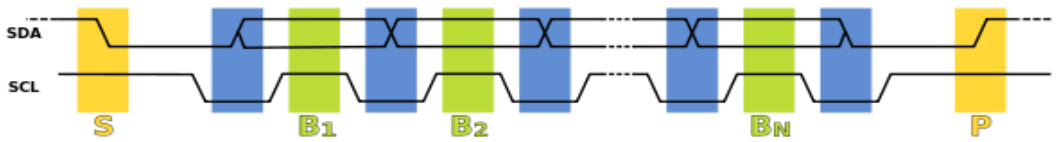


I2C

I²C is een datacommunicatie tussen microprocessoren en IC's. Deze datacommunicatie gebeurt synchroon over een seriële bus. Er kunnen meerdere masters en slaves aangesloten zijn op deze bus. Maar er moet minstens één master en minstens één slave zijn om te kunnen communiceren.



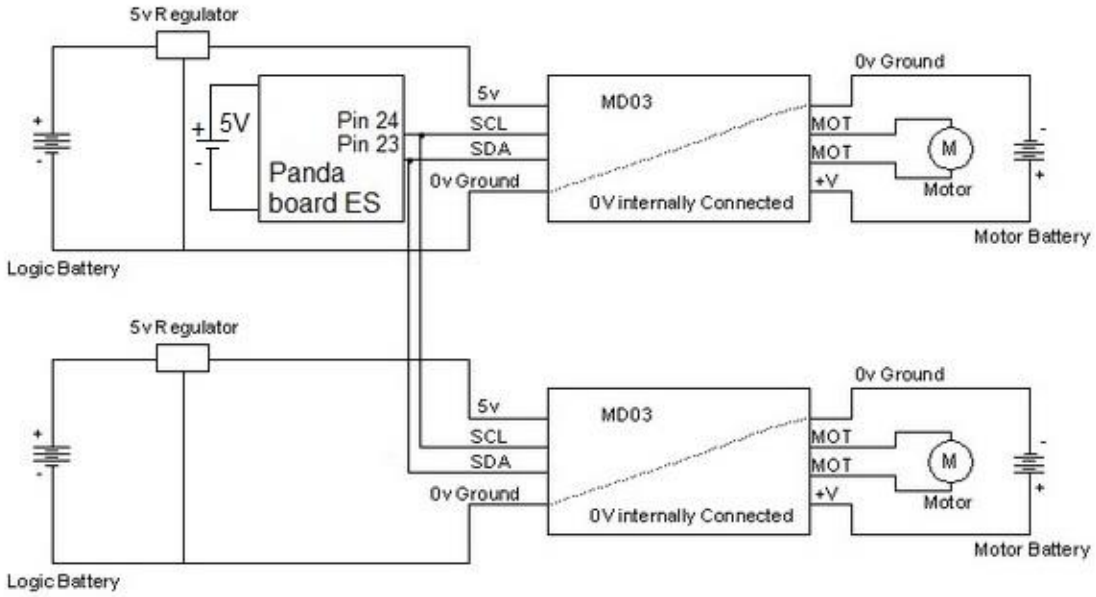
I²C heeft twee lijnen waarmee het kan werken deze noemen SDA of serial data en SCL of serial clock. De naam zegt het zelf over SDA wordt enkel en alleen data verzonden tussen master en slave. En op de SCL lijn wordt enkel een clock verzonden tussen data en master. De slaves kunnen enkel naar de master communiceren als de master hun een verzoek stuurt. Hierdoor heeft de master volledige controle over de I²C bus. Ook genereert de master het kloksignaal en de start/stop bit.



In de onderstaand figuur kunt u zien hoe dit in zen werk gaat.
Als we willen communiceren moet de master een startbit sturen naar de slave dit moet een van hoog naar laag signaal zijn. Dit gebeurt op de SDA lijn en de SCL moet op dat moment hoog zijn. Daarna stuurt de master een adrespuls met read/write over de I²C lijn, afhankelijk hiervan geeft de master aan of hij wilt ontvangen of schrijven. De slave met welke het adres overeenkomt zal hierop reageren met een acknowledge. Zo weet de master dat er een verbinding is.
Nu kan er data byte per byte verstuurd worden gepaard met een acknowledge van de ontvanger zodat er kan bevestigd worden dat de data ontvangen is.
Als alle data verstuurd is zal de master een stopbit versturen die is een van laag naar hoog signaal op de SDA lijn terwijl SCL op dat moment hoog is.

Schema

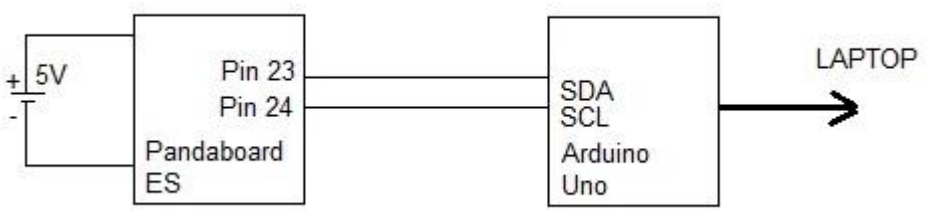
Het origineel schema ziet er als volgt uit, het Pandaboard word gevoed via een adapter en krijgt 5V. Dan wordt pin24 van het Pandaboard met de SCL pin verbonden van de MD03 controller, dit is de clock lijn. Dan word pin 23 van het Pandaboard met de SDA pin verbonden met de MD03 controller, dit is de data lijn. Dan worden deze twee verbindingen parallel verbonden met de tweede MD03 controller en afhankelijk van de schakelaars verandert het adres voor de seriële communicatie. Dan hebben we nog batterijen voor de motors en voor de microcontroller moest deze op de transportkar



komen. De motor wordt ook nog op de controller aangesloten. Hieronder staat het schema voor het testen van de seriële verbinding. De Arduino wordt aangesloten op een laptop en krijgt zo spanning. Het Pandaboard krijgt 5V van zijn adapter en pin 23 wordt aangesloten op de SDA pin van de Arduino. De SCL pin van de Arduino wordt dan nog verbonden met pin 24 van het Pandaboard. Op deze manier kunnen we controleren op de laptop welke waarde er wordt doorgestuurd serieel van het Pandaboard naar de Arduino.

Code

Eerst



wouden we in ROS een code herschrijven die we gevonden hadden zodat ROS rechstreeks via I²C aan de motor controller kan aansturen. We dachten dat we via een rosserial commando rechstreeks de waarde naar de motor controller konden sturen. Zie code ROS in GitBub. Maar na veel opzoekwerk hebben we geen voorbeeld of bruikbare informatie kunnen vinden hoe we via ROS met I²C rechtreeks een waarde kunnen uitsturen. We hebben dan wel een voorbeeld gevonden hoe we via een Arduino kunnen koppelen met onze ROS. Zo kunnen we via ROS een bericht sturen naar onze Arduino met informatie over de motor snelheid, versnelling en richting. De Arduino kan dan dit ROS bericht ontvangen en ontleden zodat hij dit kan gebruiken om de motor controller aan te sturen.
De Arduino wordt via de USB verbinding aangesloten op het Pandaboard. Eerst testen we de communicatie tussen de Arduino en de MD03. Zie TestLink code in GitHub. Daarna gaan we de motor aansturen via de Arduino maar de Arduino met zijn waarde uit een ROSNode. Deze Node versturen we via ROS met een bepaalde waarde en de Arduino gaat dan kijken wat deze waarde is en zo zijn motor aansturen met deze waarde. Zie code ROSARDUINO in GitHub

In ros geven we de volgende commands in:

deze commands moeten elk in een aparte terminal ingegeven worden.

```
roscore  // om ros op te starten
```

```
rosrun rosserial_python serial_node.py _port:=/dev/ttyUSB0 //om de communicatie met de arduino op te starten.
```

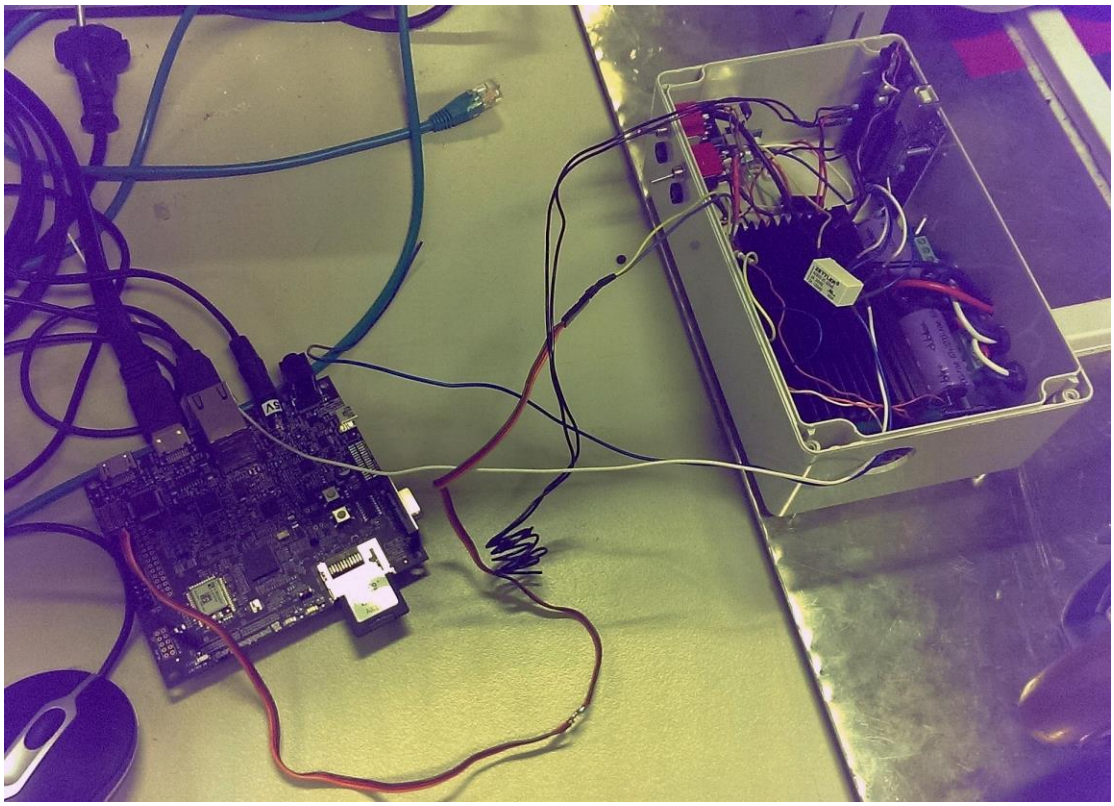
In een nieuw terminal gebruiken we rostopic pub om de snelheid van onze motor te beheren. We kunnen hier een snelheid ingeven die dan naar de Arduino verstuurd wordt.

```
rostopic pub motor std_msgs/UInt16  <speed>
```

Conclusie

We hebben ondervonden dat we vaak verkeerde informatie aan het bekijken waren omdat wij met een ARM werkten en niet met een desktop. Bijvoorbeeld we hebben enkele labo's gezocht naar de juiste Ubuntu versie die nog geïnstalleerd kan worden op ons Pandaboard omdat dit een ARM versie moet zijn. Dan uitzoeken welke versie van ROS we konden installeren op deze Ubuntu versie en dat nog compatibel was met ons Pandaboard. En als we dan bepaalde fouten tegen kwamen was het niet altijd even gemakkelijk om deze op te lossen omdat we op de ARM versie zaten. Er zijn dan wel veel voorbeelden of projecten te vinden omdat ROS een open source omgeving is en dit maakte het voor ons makkelijk om de codes te kennen die ROS gebruikt. Maar omdat er zoveel informatie te vinden is wordt het moeilijk om de juiste terug te vinden en dit hebben we ondervonden door de vele pagina's die we af zijn gegaan voor oplossingen. Dit opzoekwerk zorgde dat we vaak tijd verloren en dan extra uren konden besteden aan ons project.

Bronvermelding



http://www.ros.org/

http://wiki.ros.org/hydro

http://wiki.ros.org/hydro/Installation/UbuntuARM

http://pandaboard.org/

http://pandaboard.org/content/resources/references

http://www.microtechnica.tv/support/manual/pbes-man.pdf

http://www.ubuntu.com/

https://wiki.ubuntu.com/ARM/OmapDesktopInstall

http://jbohren.com/articles/roscpp-hello-world/

http://www.robot-electronics.co.uk/htm/md03tech.htm

http://www.robot-electronics.co.uk/files/i2cdrv.c

http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup

http://pharos.ece.utexas.edu/wiki/index.php/Attaching_the_MD03_Motor_Controller_to_the_Arduino_Mega_-_09/16/2011

http://nl.wikipedia.org/wiki/I%C2%B2C-bus