Een kleine tutorial voor BrickPi.

Versie 1.0	Jan Zuurbier	19-2-2018

1. BrickPi3.

De BrickPi3 is een bordje dat je aan je RaspberryPi kunt koppelen. De BrickPi3 kan de motoren van een Legorobot aansturen en de sensoren uitlezen. Zodoende kan je op de RaspberryPi programma's schrijven om een Lego-robot te besturen. In deze cursus zullen wij daartoe de taal C++ gebruiken.

De BrickPi wordt geleverd met een behuizing waar de BrickPi en de RaspberryPi in passen. De behuizing is geschikt om er Lego-stukjes aan vast te maken.

Er is ook een voeding waar 8 batterijen in passen. In plaats daarvan kan je ook een 12V powerpack gebruiken. Voor de RaspberryPi zelf is een 5V voeding voldoende. Maar als je motoren wilt laten draaien heb je een voeding van 10V of 12V nodig. Je sluit de voeding aan op de BrickPi. De BrickPi voorziet dan de RaspberryPi van spanning (5V) als de schakelaar op 'on' staat.

2. De BrickPi3 gereed maken voor gebruik.

2.1 Raspian for Robots:

Je dient gebruik te maken van het operating system ‘Raspian for robots’. Dit is een versie van ‘Raspian for RaspberryPi’ en bevat enkele toevoegingen die nuttig zijn voor de BrickPi3.

https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card/

Gebruik Win2Diskmanager om image op SD-cart te branden. (Het programma Etcher for PC) werkte niet bij mij.

2.2 Koppel de BrickPi3 aan een Raspberry Pi.

Volg de instructies op:

https://www.dexterindustries.com/BrickPi/brickpi3-getting-started-step-1-assembly/

Er is een klein schoevendraaiertje nodig. Als je er zelf geen hebt is er wel een in het TI-lab te vinden.

2.3 Raspian configureren.

Koppel een monitor via HMDI en een toetsenbord en muis via USB.

Middels ‘`sudo raspi-config`’ kan je raspian configuren.

De volgende stappen zijn nuttig.

- Expand Filesystem.
- Verder is het wellicht nuttig om toetsenbord configuratie aan te passen. Standaardmodel 105 staat meestal al ingesteld en hoef je niet te wijzigen.
- Password kan je wijzigen maar onthoud het wel. (Het default password is robots1234, de username pi)
- Als locale kan je Nederlandse locale instellen zowel ISO-8859 als UTF-8.
- Pas de tijdzone aan.

1.4 Wifi instellen.

Een draadloze verbinding is fijn en nodig als je een rijdende robot gaat maken.

Er is een grafisch programma om wifi te configureren dat is het gemakkelijkst. Het programma heet `wpagui`. Als je een grafische desktop hebt dan is er een icon waar je op kunt klikken. Om verbinding te maken met eduroam moet je de volgende gegevens invoeren.

SSID: `eduroam`
Authentication: `WPA2-Enterprise (EAP)`
Encryption: `CCMP`
EAP method: `PEAP`
Identity: `<voornaam.achternaam>@student.hu.nl`
Password: `**********`
CA certificate: <leeg laten>
Inner auth: `EAP-MSCHAPV2`

Voor een ander network is het meestal eenvoudiger en dien je alleen je wachtwoord in te vullen bij PSK (preshared key). Maar mogelijk is er een andere Authentication-methode en Encryption-methode, dit hangt af van jouw router.

Zonder grafische tool kan het ook. Zie https://raspberrypi.stackexchange.com/questions/37920/how-do-i-set-up-networking-wifi-static-ip-address

Check eerst de inhoud van `/etc/network/interfaces` . Die zou er als volgt moeten uitzien. En hoeft dan niet aangepast te worden.

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Open nu de file `/etc/wpa_supplicant/wpa_supplicant.conf`. Om de file aan te passen dien je root-rechten te hebben. Gebruik de editor nano:

`sudo nano /etc/wpa_supplicant/wpa_supplicant.conf`.

Je kunt ook de editor vi gebruiken maar die is moeilijker te bedienen.

Voeg nu een of beide van de volgende elementen aan toe:

```
network={
        ssid="eduroam"
```

```
        proto=RSN
        key_mgmt=WPA-EAP
        pairwise=CCMP
        auth_alg=OPEN
        eap=PEAP
        identity="naam@student.hu.nl"
        password="*********"
        phase2="auth=MSCHAPV2"
}

network={
        ssid="your ssid"
        psk="your password"
}
```

2.5 Updaten firmware BrickPi3

De BrickPi3 komt niet standaard met de laatste versie van de firmware. Dit komt omdat er nog regelmatig verbeteringen en nieuwe features worden toegevoegd. Voor gebruik van de C++-library is echter wel versie 1.4 nodig. Als BrickPi3 verbonden is met de RaspberryPi is updaten van de firmware eenvoudig. Met het volgende commando vanaf de linux-commandline wordt de software geupdated:

```
sudo bash /home/pi/Dexter/BrickPi3/Firmware/brickpi3samd_flash_firmware.sh
```

Zie : https://forum.dexterindustries.com/t/old-firmware-please-help/2866/2 .

2.6 Raspberry Pi benaderen via SSH:

We willen de RaspberryPi in 'headless mode' te benaderen. Dat wil zeggen zonder monitor, toetsenbord en muis, maar via het internet. Daarvoor gebruiken we SSH (secure shell). Op de RaspberryPi is SSH standaard geinstalleerd en geactiveerd. Op je laptop moet je het programma PuttY installeren.

Voordat je de monitor en toetsenbord kunt loskoppelen moet je het ipadres van de RaspberryPi weten.

Zorg eerst dat de RaspberryPi via een netwerkkabel is verbonden met je laptop. Zorg ook dat je wifi hebt ge-enabled. Open een terminal en doe '`ifconfig`'. Onthou het ip-adres van van `eth0` en van `wlan0`.

Maak een SSH verbinding naar een van de ipadressen via poort 22. (Als je `eth0` gebruikt moet je de RaspberryPi via een ethernetkabel verbinden met jouw laptop. Verbinding met `wlan0` gaat draadloos via wifi.)[1]

[1] In theorie is het mogelijk dat de RaspberryPi en jouw laptop zich in een verschillend netwerk bevinden omdat ze gebruik maken van een verschillende router. In dat geval zou je de RaspberryPi kunnen inrichten als wireless accespoint. Zie https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md .

2.7 C++ programma's.

Haal de voorbeeld programma's binnen vanaf github.

```
cd ~
git clone https://github.com/janzuurbier/piprograms.git
cd ~/piprograms
ls
```

Compileer het programma info.cpp[2]

```
g++ -o info BrickPi3.cpp info.cpp
```

Er is nu een executable met de naam info gemaakt. Deze kan je zien als je het commando `ls` geeft.

Uitvoeren gaat met

```
./info
```

Voor de ander programma's moet je eerst een robot bouwen en motoren of sensoren aansluiten….

> *Opmerking:*
> *De opzet van BrickPi3.cpp en info.cpp zoals die in de folder "Dexter/BrickPi3/Software/C/Examples" staat is niet zoals het hoort. In de C++ les heb je geleerd dat je header files dient te includen. De header file bevat de declaraties. De definities komen dan in de cpp-file die je mee complieert. Gebruik daarom in het vervolg de files BrickPi3.cpp en BrickPi3.h zoals die op github staan en niet die uit de folder Examples. Bovendien heb je geen schrijfrechten in de map Examples omdat die van 'root' is. Werk daarom in een folder die je zelf hebt gemaakt in de homedirectory van pi.*

[2] Merk op dat info.cpp in feite een C-programma is. Als je de broncode bekijkt zal je veel dingen zien die je vreemd voorkomen en die niet in de programmeerles zijn behandeld.

3. Het aansturen van de motoren.

Bouw om te beginnen een Lego-robot met wielen waarvan er twee zijn aangesloten op de motorenpoorten B en C. Zie bijvoorbeeld de file simplebot.lxf. Deze file kan je bekijken met het programma Lego Digital Designer. Maar het is natuurlijker leuker om zelf een ontwerp te maken.

De BrickPi3 kent vier poorten waarop je motoren kunt aansluiten. Deze aansluitingen zijn genummerd A tot en met D. In het programma heten deze poorten PORT_A, PORT_B, PORT_C en PORT_D. Om de motoren te laten draaien zijn er een aantal functies. Hieronder staan de declaraties zoals die in de headerfile `BrickPi3.h` staan.

```
// Set the motor PWM power
int     set_motor_power(uint8_t port, int8_t power);

// Set the motor target position to run to
// Set the absolute position to run to (go to the specified position)
int     set_motor_position(uint8_t port, int32_t position);

// Set the relative position to run to (go to the current position plus the
specified position)
int     set_motor_position_relative(uint8_t port, int32_t position);

// Set the motor speed in degrees per second. As of FW version 1.4.0, the
algorithm regulates the speed, but it is not accurate.
int     set_motor_dps(uint8_t port, int16_t dps);

// Set the motor limits. Only the power limit is implemented. Use the power
limit to limit the motor speed/torque.
int     set_motor_limits(uint8_t port, uint8_t power, uint16_t dps);
```

PWM betekent pulse width modulation. Bij gebruik van de functie `set_motor_power` krijgt de motor niet continu spanning maar krijgt telkens een korte puls spanning. De tijdsduur van zo'n puls noemt men ook wel de breedte van de puls. De breedte van de puls bepaald hoeveel procent van de tijd de motor spanning krijgt en dus met hoeveel kracht de motor draait. De tweede paramater van de functie `set_motor_power` is een 8 bits integer. Deze kan waarden aannemen van -128 tot 127. Deze integer is de breedte van de pulse. Als de integer gelijk is aan 0 dan krijgt de motor in het geheel geen spanning. Als de de integer gelijk is aan 127 dan krijgt de motor voortdurend spanning en draait hij dus op maximale kracht. Bij negatieve waarden draait de motor de andere kant op. Hierbij is -128 een speciale waarde. De motor krijgt dan geen stroom, maar gaat vrij draaien. Dit in tegenstelling tot de waarde 0. Als je een motor_power 0 geeft dan remt de motor.

De functie `set_motor_dps` laat je ook de motorsnelheid regelen. Hier wordt de snelheid gereguleerd op het aantal graden per seconde (degrees per second). Zodoende is de snelheid onafhankelijk van het voltage

van de batterij. Het schijnt echter dat dit niet accuraat is. Deze functie zou je kunnen gebruiken om een tweewielig voertuig in een rechte lijn te laten rijden zonder navigatielijn. Je dient dan beide wielen namelijk dezelfde snelheid te geven.

De functie `set_motor_position` draait de motor naar een bepaald hoek. Bijvoorbeeld `set_motor_position(360)` zorgt dat er precies één omwenteling wordt gemaakt. Als je daarna doet `set_motor_position(360)` dan draait de motor niet en blijft in dezelfde positie omdat hij eerder als 360 graden is gedraaid.

De functie `set_motor_position_relative(360)` rekent altijd vanuit de huidige positie. Als je dan doet `set_motor_position(360)` en daarna `set_motor_position_relative(360)` dan zal hij de tweede keer weer een volledige omwenteling maken.

Bij het gebruik van `set_motor_position` of `set_motor_position_relative` draait de motor in principe op volle kracht. Je kunt dat beperken met de functie `set_motor_limits`.

Het programma `simplebot1.cpp` toont het gebruik van deze functies.

Het editen van code op de RaspberryPi.

De code kan je bekijken in een editor op je laptop. Gebruik daartoe het programma WINSCP in explorer-mode. Maak verbinding met de RaspberryPi. Ga nu naar de map piprograms waar je gedownloade files staan. Klik met de rechtemuisknop op het bestand simplebot1.cpp. Kies vervolgens op 'openen met....' en kies je favoriete editor.

Als je het programma wilt editen met een teksteditor op de RaspberryPi dan kan je de editor nano gebruiken of de commandline editor vi. Mogelijk kan je ook een grafische editor op de RaspberryPi installeren. En dan heb je een VPN-verbinding nodig om de editor vanaf je laptop te bedienen.

Behalve het laten draaien van de motor kan je ook opvragen hoeveel de motor gedraaid heeft. Dat is van belang als je een voertuig een gegeven afstand wil laten afleggen. En als je daarbij tijdens het rijden ook iets anders wilt doen, bijvoorbeeld opletten op obstakels of een lijn volgen. In een dergelijk geval is `set_motor_position` namelijk niet toepasbaar.

Je kunt de afgelegde draaihoek opvragen met de functie `get_motor_encoder`, waarvan er twee versies zijn. Je krijgt dan het aantal graden dat de motor heeft gedraaid sinds de vorige keer dat je de encoder hebt gereset.

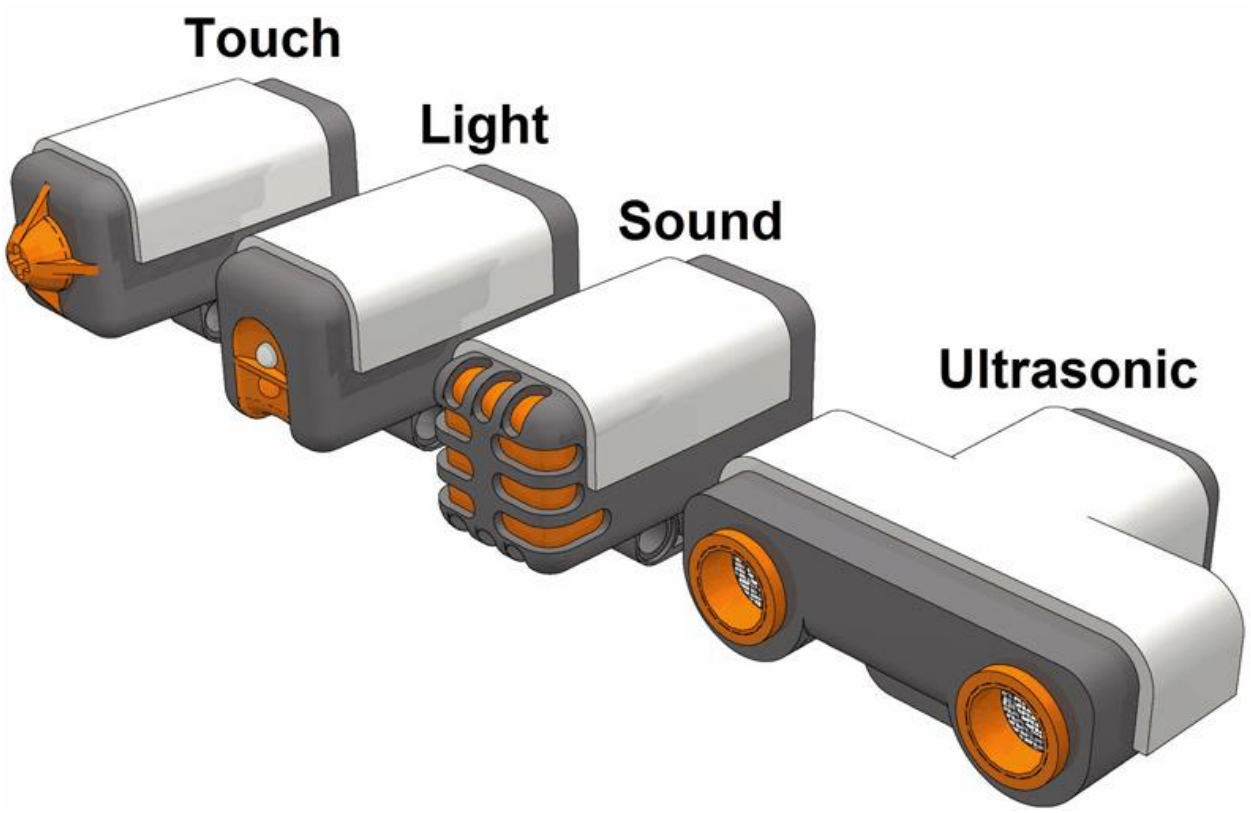
In de volgende figuur zie je de declaratie van functies die betrekking hebben op de motorencoder. Deze zijn gekopieerd uit het bestand `BrickPi3.h`.

```
// Offset the encoder position. By setting the offset to the current position, it
effectively resets the encoder value.
int     offset_motor_encoder(uint8_t port, int32_t position);

// Reset the encoder.
// Pass the port and pass-by-reference variable where the old encoder value will be
stored. Returns the error code.
int     reset_motor_encoder(uint8_t port, int32_t &value);

// Pass the port. Returns the error code.
int     reset_motor_encoder(uint8_t port);

// Set the encoder position.
// Pass the port and the new encoder position. Returns the error code.
int     set_motor_encoder(uint8_t port, int32_t value);

// Get the encoder position
// Pass the port and pass-by-reference variable where the encoder value will be
stored. Returns the error code.
int     get_motor_encoder(uint8_t port, int32_t &value);

// Pass the port. Returns the encoder value.
int32_t get_motor_encoder(uint8_t port);
```

Het programma `motors.cpp` demonstreert het gebruik van de encoder.

4. Het gebruik van de sensoren.

In combinatie met Lego NXT kan je verschillende sensoren gebruiken. Bij de basis set worden meegeleverd een touch-sensor, een ultrasoon-sensor, een light-sensor en een color-sensor en een soundsensor. Er zijn vier sensorpoorten aan de BrickPi waaraan je de sensoren kunt aansluiten. Deze poorten zijn genaamd PORT_1, PORT_2, PORT_3 en PORT_4.



Een light sensor kan je in twee manieren gebruiken. De ene manier is met het licht aan zodat het gereflecteerde licht wordt gemeten. De andere manier is met het licht uit zodat het ambient licht wordt gemeten. Het ambient licht is het licht uit de omgeving.

Een color sensor (niet getoond in de figuur) kan je op verschillende manieren gebruiken. Je kunt het groene, rode of blauwe lampje aan doen. Dan wordt het gereflecteerde licht gemeten. Vergelijkbaar met de light sensor. Je kunt alle drie de lampjes aandoen. Dan wordt ook het gereflecteerde licht meten en dan kan je nagaan wat de kleur is van het gereflecteerde licht. Je kunt ook geen van de lampjes aan doen. Dan wordt het ambient licht gemeten en kan je nagaan wat daarvan de kleur is.

Voor het gebruik van de sensoren in een C++ programma zijn twee functies belangrijk. In de file BrickPi3.h staan de decraraties van de functies. Ook kan je constanten en struct-typen vinden die van belang zijn bij het gebruik van de functies. Als eerste wordt hieronder de declaraties van de twee functies besproken.

Met de functie `set_sensor_type` kan je aangeven welk soort sensor er is aangesloten op welke sensorpoort. De eerste parameter is de poort. De tweede parameter is het type. In de file BrickPi3.h staat de enummeratie SENSOR_TYPE. Hierin staan verschillende constanten gedefinieerd die je kan gebruiken voor het type van de sensor. De functie `set_sensor_type` kent nog twee optionele parameters namelijk `flags` en`i2c_struct`. Deze parameters hoef je bij de aanroep geen waarde te geven omdat ze een default waarde hebben. In de voorbeelden zullen ze niet worden gebruikt.

De functie `get_sensor` dient om de waarde van de sensor op te vragen. De eerste parameter is de poort waaraan de sensor is aangesloten. De sensorwaarde wordt opgeslagen in de struct `sensor_val_struct.` Je dient voor het aanroepen van de functie een struct te maken en een deze struct mee te geven met de aanroep. Omdat er sprake is van call by reference kan in de functie de struct gevuld worden met relevante sensorwaarden. In de file BrickPi3.h staan verschillende struct-typen gedefinieerd voor de verschillende soorten sensor.

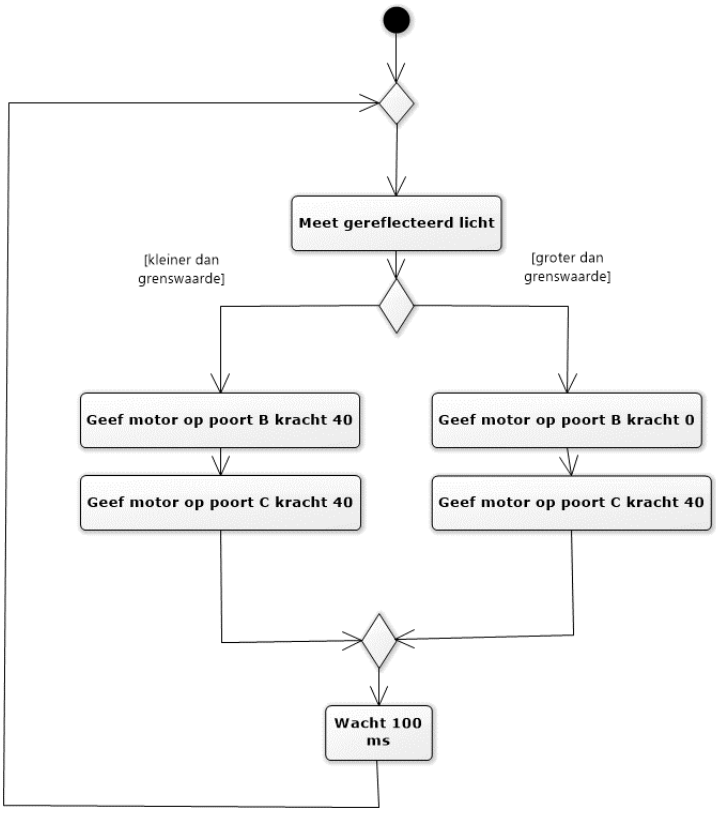
```
// Configure a sensor
// Pass the port(s), sensor type, and optionally extra sensor configurations
(flags and I2C information).
int     set_sensor_type(uint8_t port, uint8_t type, uint16_t flags = 0,
i2c_struct_t *i2c_struct = NULL);

// Get sensor value(s)
int     get_sensor(uint8_t port, sensor_val_struct_t &sensor_val_struct);
```

Het programma `sensors_nxt.cpp` demonstreert het gebruik van sensoren. Maar alleen voor NXT-sensoren en niet voor EV3 sensoren.

5. Lijn volgen.

Om een lijn te volgen kan een two-state algoritme worden toepast. Je krijgt een soort zigzag beweging zoals te zien is in https://www.youtube.com/watch?v=qLDKEjKZPDg . De twee toestanden zijn ‘vooruit’ als de robot goed boven de lijn zit. En ‘naar recht’ als de robot van de lijn dreigt te lopen. Met een lichtsensor kan je bepalen in welke van de twee toestanden de robot zich bevindt. Als er een donkere kleur wordt gemeten kan de robot vooruit. Beide motoren worden dan aangestuurd. Als er een licht kleur wordt gemeten dan moet de robot naar recht bijsturen, alleen de linker motor wordt dan aangestuurd. Het is verstandig om de grenswaarde te bepalen door twee metingen te doen: eentje op de zwarte lijn en eentje op de witte achtergrond. Het gemiddelde van deze twee waarden is dan de grenswaarde.

Het algoritme kan met een activity-diagram worden weergegeven.



Een meer vloeiende beweging krijg je door zogenaamd ‘proportioneel bijsturen’. Indien de hoeveelheid gereflecteerdlicht groter is dan de grenswaarde wordt niet gelijk de kracht van motor B gereduceerd tot 0. Maar dit gebeurt proportioneel. Als er een kleine overschrijding is dan wordt de motor kracht niet zoveel gereduceerd. Bij grote overschrijding wordt uiteindelijk wel de motorkracht tot 0 gereduceerd. Ook als de hoeveelheid gereflecteerd licht kleiner is dan de grenswaarde dan wordt er bijgestuurd. In dat geval zit de sensor te veel boven de zwarte lijn en moet hij terug naar de rand van de zwarte lijn. Nu wordt de andere

motor in kracht gereduceerd. Dit principe is toegepast in `simplebot3.cpp` . Zie ook https://youtu.be/IH7PWprLPkc .