Raspberry Pi Recepten - Deel 7

PWM op het menu

Tony Dixon
(Verenigd Koninkrijk)

Tot nu toe hebben we in deze serie gekeken naar allerlei digitale signalen: GPIO, Seriële UART, SPI en I²C. We hebben ook analoge signalen (via SPI) gezien. Wat is er nog meer? Dit keer gaan we PWM-functionaliteit toevoegen aan onze Raspberry Pi.

PWM-interfaces in hardware

Een PWM-signaal bestaat uit een blokgolf waarvan de pulsbreedte kan worden aangepast of, in technische termen, gemoduleerd. Door die verandering varieert ook de gemiddelde waarde van de golfvorm. PWM wordt vooral gebruikt om elektrische apparaten zoals lampen en motoren te regelen. Het systeem-op-een-chip (SoC) dat het hart van de Pi vormt, de BCM2835 van Broadcom, heeft twee PWM-kanalen in de hardware. De ene wordt door het systeem gebruikt voor het genereren van audio en de andere, PWM0, is vrij ter beschikking. Die is te vinden op pen 12 (GPIO18) van de uitbreidingsconnector van de Raspberry Pi (zie **tabel 1**).

Dimmen van een LED

We kunnen PWM0 gebruiken om een LED te dimmen. Om het dimmen van een LED via PWM te demonstreren sluiten we een LED in serie met een weerstand van 330 Ω aan tussen PWM0 (GPIO18/pin 12) en massa. Normaal gesproken zouden we nu Python opstarten en met de RPi.GPIO-library interessante dingen doen met de PWM-uitgang, maar vreemd genoeg biedt RPi.GPIO geen

ondersteuning voor PWM0. Daarom gaan we *wiringPi* installeren, een library die is geschreven door Gordon Henderson [1]. *wiringPi* is een C-library voor de Raspberry Pi en is te gebruiken vanuit veel verschillende programmeertalen, ook vanuit Python. *wiringPi* lijkt sterk op *wiring*, het software-platform voor Arduino.
Om te beginnen moeten we nog wat software installeren. Installeer eerst de Python Package Installer (PiP) met:

```
sudo apt-get install python-dev
python-pip
```

Download en installeer daarna *wiringPi* voor Python met:

```
sudo pip install wiringpi2
```

Start dan IDLE, de Python IDE, en voer de code uit **listing 1** in.
Dit Python-programma stelt GPIO18/PWM0 in als een digitale output. Daarna komt het programma in een lus waarin de modulatie van PWM0 stap voor stap wordt vergroot. Naarmate het PWM-signaal toeneemt, gaat de LED feller branden.

PWM-interfaces in software

Goed, we hebben dus een hardware-PWM op de uitbreidingsconnector, maar als we nu eens meer dan één PWM-signaal willen hebben? Dan kunnen we een PWM-chip zoals de 16-kanaals 12-bits PCA9685 van Texas Instruments gebruiken om extra PWM-kanalen te maken. Of we kunnen met software een PWM-signaal maken, zonder geld uit te geven.
Een kleine waarschuwing: Normaal gesproken is het geen probleem om een software-PWM te gebruiken op een embedded platform zoals een Arduino, waar de processor alleen de toepassingscode draait. Maar als we een software-PWM gebruiken op een universele computer zoals de Raspberry Pi, moeten we er rekening mee houden dat daarop een volledig besturingssysteem (OS) draait, dat allerlei verschillende taken en programma's tegelijk uitvoert. Daardoor kan de PWM-timing in de problemen komen. Dat kan leiden tot een te lage resolutie en teveel jitter, omdat het OS het PWM-programma kan onderbreken om

Tabel 1. Pinbezetting van de uitbreidingsconnector

Naam	Functie	Alternatief	RPi.GPIO
P1-02	5,0V	-	-
P1-04	5,0V	-	-
P1-06	GND	-	-
P1-08	GPIO14	UART0_TXD	RPi.GPIO8
P1-10	GPIO15	UART0_RXD	RPi.GPIO10
P1-12	GPIO18	**PWM0**	RPi.GPIO12
P1-14	GND	-	-
P1-16	GPIO23		RPi.GPIO16
P1-18	GPIO24		RPi.GPIO18
P1-20	GND	-	-
P1-22	GPIO25		RPi.GPIO22
P1-24	GPIO8	SPI0_CE0_N	RPi.GPIO24
P1-26	GPIO7	SPI0_CE1_N	RPi.GPIO26

Naam	Board Revision 1		Board Revision 2	
	Functie	Alternatief	Functie	Alternatief
P1-01	3,3V	-	3,3V	-
P1-03	GPIO0	I2C0_SDA	GPIO2	I2C1_SDA
P1-05	GPIO1	I2C0_SCL	GPIO3	I2C1_SCL
P1-07	GPIO4	GPCLK0	GPIO4	GPCLK0
P1-09	GND	-	GND	-
P1-11	GPIO17	RTS0	GPIO17	RTS0
P1-13	GPIO21		GPIO27	
P1-15	GPIO22		GPIO22	
P1-17	3,3V	-	3,3V	-
P1-19	GPIO10	SPI0_MOSI	GPIO10	**SPI0_MOSI**
P1-21	GPIO9	SPI0_MISO	GPIO9	**SPI0_MISO**
P1-23	GPIO11	SPI0_SCLK	GPIO11	**SPI0_SCLK**
P1-25	GND	-	GND	-

Opmerking: I2C0_SDA, I2C0_SCL (GPIO0 & GPIO1), I2C1_SDA en I2C1_SCL (GPIO2 & GPIO3) hebben pullup-weerstanden van 1k8 naar 3,3 V.

Listing 1. LED dimmen

```
#!/usr/bin/python

import wiringpi2 as gpio
import time

#set up gpio
gpio.wiringpiPiSetupGpio ()
gpio.pinMode (18,2)

while True:
  gpio.pwmWrite (18,0)
  for n in range (0,1024):
    gpio.pwmWrite (18,n)
    time.sleep (0.01)
```

andere dingen te draaien. Dat kan lastig zijn als we grote nauwkeurigheid en weinig jitter nodig hebben. Gelukkig heeft de Pi meerdere Direct Memory Access (DMA) kanalen in hardware, die kunnen helpen om een hardware-timing te creëren voor onze software-PWM. Omdat de DMA onafhankelijk van de CPU werkt, kunnen we de timing toch besturen met hardware en is de kans op onderbreking door het OS kleiner. Op die manier zijn de nauwkeurigheid en de jitter te verbeteren.

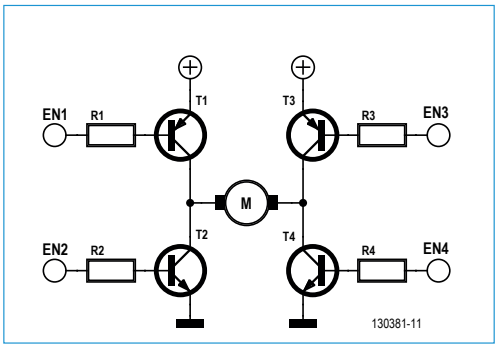
Besturing van gelijkspanningsmotoren

Een andere bekende toepassing van PWM is het besturen van de snelheid van een elektrische motor.

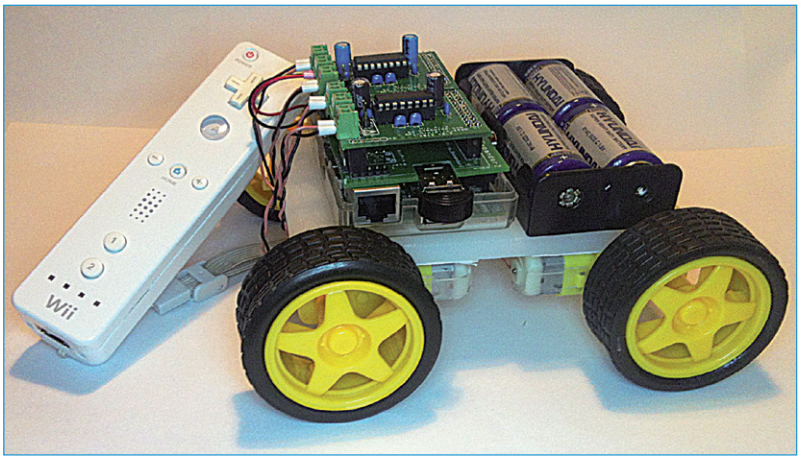
Met de H-brug-schakeling in **figuur 1** kunnen we een motor in beide richtingen laten draaien, maar als we een PWM-signaal gebruiken om één van de kwadranten aan te sturen, kunnen we ook de snelheid regelen.

PiiBOT: De deegroller

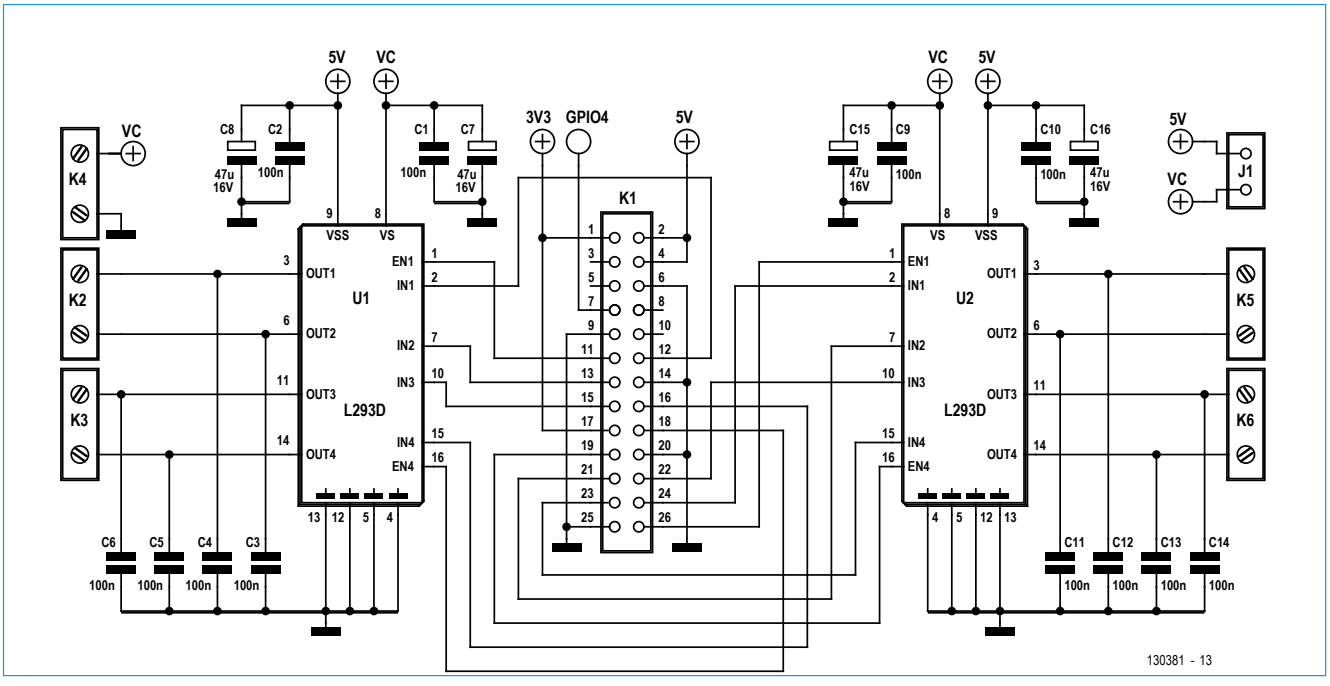
In figuur 2 zien we de PiiBOT. Dit is een robot met vier wielen, die op afstand wordt bediend met een Nintendo Wii. Deze robot heeft twee chips van het type L293D waarmee vier kleine gelijkspanningsmotoren worden aangestuurd en een Bluetooth USB-dongle voor het ontvangen van de commando's van de Wii.



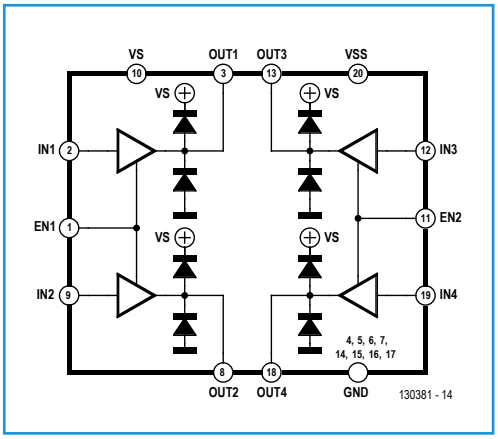
Figuur 1.
Schema van een H-brug.



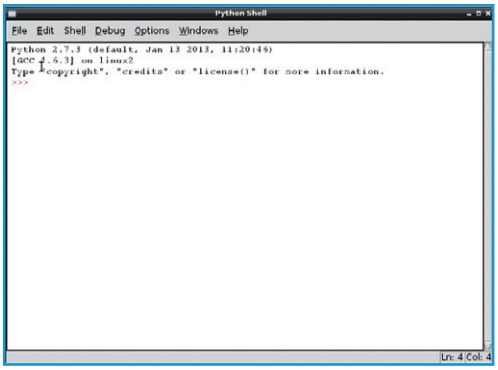
Figuur 2. De PiiBOT, een robot op vier wielen.



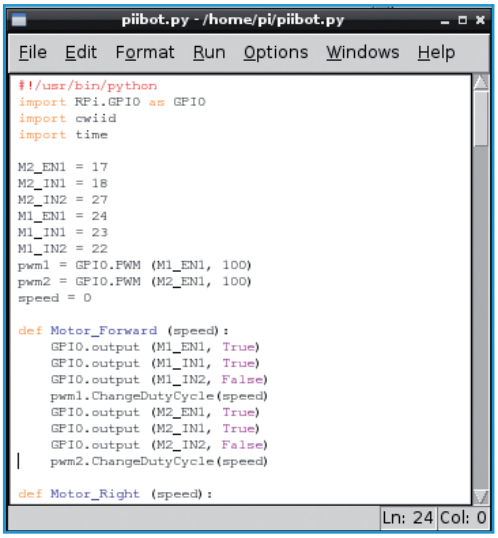
Figuur 3. Schema van de opsteekprint voor het besturen van de motoren van de PiiBOT.



Figuur 4. De L293D motorbesturingschip.



Figuur 5. De Python-shell.



Figuur 6. De IDLE-editor met het script 'piibot.py'.

Tabel 2. Gebruik van de GPIO-lijnen

Motoren 1 en 2 (U1 L293D)		Motoren 3 en 4 (U2 L293D)	
Motorfunctie	**GPIO**	**Motorfunctie**	**GPIO**
EN1	GPIO17	EN1	GPIO7
IN1	GPIO18	IN1	GPIO8
IN2	GPIO27	IN2	GPIO9
EN2	GPIO24	EN2	GPIO10
IN3	GPIO22	IN3	GPIO25
IN4	GPIO23	IN4	GPIO11

De schakeling

Onze motorregelaar is opgebouwd als een opsteekprintje voor de Raspberry Pi [2]. Het schema is te zien in **figuur 3**. Het bevat twee L293D-chips. De L293D heeft zich al bewezen in veel projecten met kleine motoren. Hij bevat twee complete H-brug-drivers, die elk 600 mA kunnen leveren (piekstromen tot 1,2 A) en is heel gemakkelijk

Installeren van de Bluetooth-drivers en de CWii-library voor Python

Om de Nintendo Wii-controller te kunnen gebruiken, moeten we eerst de Bluetooth-drivers voor de USB-Bluetooth-module installeren. We hebben alleen eenvoudige Bluetooth-communicatie nodig, daarom geven we de optie '`--no-install-recommends`' mee bij het onderstaande commando:

```
sudo apt-get install --no-install-recommends bluetooth
```

Plug de Bluetooth-dongle in de Pi en test de interface met:

```
sudo service bluetooth status
```

Als alles in orde is, krijgen we het volgende resultaat:

```
[ ok ] bluetooth is running.
```

Als de Bluetooth-drivers zijn geïnstalleerd, kunnen we CWii [4], de Python Wii-library, downloaden. Geef de volgende commando's:

```
sudo apt-get install python-cwiid
```

Als alles is geïnstalleerd, kunnen we de Wii-controller gebruiken met onze Pi, in dit geval voor de PiiBOT. Meer informatie over het lezen van data van zo'n controller is te vinden onder [5].

aan te sturen. Elke H-brug heeft een *enable*-signaal (EN1/2) en twee inputs (IN1/3 en IN2/4) die de draairichting besturen (zie **figuur 4**).
Op connector K4 kan een externe voeding (4,5 V tot 36 V gelijkspanning) worden aangesloten. Met jumper J1 kan ook worden gekozen om te motoren te voeden uit de 5-V-voeding van de Pi, maar dat is niet aan te raden. Kies liever voor een aparte voeding via K4.
Per L293D-chip zijn zes GPIO-lijnen nodig voor de besturing, dus in totaal gebruiken we 12 GPIO-lijnen van de Pi voor het besturen van de vier wielen. Een overzicht van al die signalen is weergegeven in **tabel 2**.
We sturen de *enable*-lijnen aan met PWM om de snelheid te regelen. Omdat er vier *enable*-signalen zijn, gebruiken we vier software-PWM-kanalen.

Voorbeeldprogramma: piibot.py
Als de schakeling is gebouwd, zijn we bijna klaar om het besturingsprogramma te schrijven in Python. Maar eerst moeten we nog de drivers en library's voor de Nintendo Wii-controller installeren. Hoe dat gaat, wordt beschreven in een apart tekstkader.
Dubbelklik het pictogram IDLE op het bureaublad van de Pi om de Python-shell en -IDE te starten (zie **figuur 5**).
Kies nu de optie 'File' in het menu en maak een nieuw programma. Dit start de IDLE-editor (**figuur 6**). Voer hiermee het programma in **listing 2** in. Het is nogal lang, maar gelukkig kan het ook worden gedownload van de ondersteuningspagina over de Raspberry Pi-serie bij Elektor.LABS [3].
Vergeet niet het programma op te slaan na het intikken. Daarna schakelen we over naar LXTerminal. Geef het volgende commando om het programma uitvoerbaar te maken:

```
chmod +x piibot.py
```

Start dan het programma met:

```
sudo ./piibot.py
```

Het programma vraagt ons dan om de Wii-controller te 'paren' met de Pi. Druk de knoppen 1 en 2 op de Wii-controller tegelijk in om dat te doen. Als de controller is gepaard, kan onze PiiBOT aan de rol!

(130381)

Weblinks

[1] WiringPi GPIO Library: http://wiringpi.com

[2] MiniPiio Motor293D-opsteekprint: www.dtronixs.com

[3] Raspberry Pi-ondersteuningspagina's bij Elektor.LABS: www.elektor-labs.com/RPi

[4] CWiid-library voor de Nintendo Wii-controller: http://abstrakraft.org/cwiid/

[5] Nintendo Wii Remote, Python en de Raspberry Pi: http://bit.ly/RPi-Wii

Listing 2: piibot.py (download dit programma van [3])

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import cwiid
import time

M1_EN1 = 24
M1_IN1 = 23
M1_IN2 = 22
M2_EN1 = 17
M2_IN1 = 18
M2_IN2 = 27
M3_EN1 = 7
M3_IN1 = 8
M3_IN2 = 9
M4_EN1 = 10
M4_IN1 = 25
M4_IN2 = 11

speed = 40

def Motor_Setup ():
    print 'Setting up..'
```

```
    GPIO.setwarnings (False)

    # Configure GPIO
    GPIO.setmode (GPIO.BCM)
    GPIO.setup (M1_EN1, GPIO.OUT)
    GPIO.setup (M1_IN1, GPIO.OUT)
    GPIO.setup (M1_IN2, GPIO.OUT)
    GPIO.setup (M2_EN1, GPIO.OUT)
    GPIO.setup (M2_IN1, GPIO.OUT)
    GPIO.setup (M2_IN2, GPIO.OUT)
    GPIO.setup (M3_EN1, GPIO.OUT)
    GPIO.setup (M3_IN1, GPIO.OUT)
    GPIO.setup (M3_IN2, GPIO.OUT)
    GPIO.setup (M4_EN1, GPIO.OUT)
    GPIO.setup (M4_IN1, GPIO.OUT)
    GPIO.setup (M4_IN2, GPIO.OUT)

    print “ready”

def Motor_Forward (speed):
    print ‘Forward, speed = ‘, speed
    GPIO.output (M1_IN1, True)
    GPIO.output (M1_IN2, False)
    pwm1.ChangeDutyCycle(speed) # M1 EN1
    GPIO.output (M2_IN1, True)
    GPIO.output (M2_IN2, False)
    pwm2.ChangeDutyCycle(speed) # M2 EN1
    GPIO.output (M3_IN1, True)
    GPIO.output (M3_IN2, False)
    pwm3.ChangeDutyCycle(speed) # M3 EN1
    GPIO.output (M4_IN1, True)
    GPIO.output (M4_IN2, False)
    pwm4.ChangeDutyCycle(speed) # M4 EN1

def Motor_Right (speed):
    print ‘Right, speed = ‘, speed
    GPIO.output (M1_IN1, True)
    GPIO.output (M1_IN2, False)
    pwm1.ChangeDutyCycle(speed)
    GPIO.output (M2_IN1, False)
    GPIO.output (M2_IN2, False)
    pwm2.ChangeDutyCycle(0)
    GPIO.output (M3_IN1, True)
    GPIO.output (M3_IN2, False)
    pwm3.ChangeDutyCycle(speed)
    GPIO.output (M4_IN1, False)
    GPIO.output (M4_IN2, False)
    pwm4.ChangeDutyCycle(0)

def Motor_Left (speed):
    print ‘Left, speed = ‘, speed
    GPIO.output (M1_IN1, False)
    GPIO.output (M1_IN2, False)
    pwm1.ChangeDutyCycle(0)
    GPIO.output (M2_IN1, True)
    GPIO.output (M2_IN2, False)
    pwm2.ChangeDutyCycle(speed)
    GPIO.output (M3_IN1, False)
    GPIO.output (M3_IN2, False)
    pwm3.ChangeDutyCycle(0)
    GPIO.output (M4_IN1, True)
    GPIO.output (M4_IN2, False)
    pwm4.ChangeDutyCycle(speed)

def Motor_Reverse (speed):
    print ‘Reverse, speed = ‘, speed
    GPIO.output (M1_IN1, False)
    GPIO.output (M1_IN2, True)
    pwm1.ChangeDutyCycle(speed)
    GPIO.output (M2_IN1, False)
    GPIO.output (M2_IN2, True)
    pwm2.ChangeDutyCycle(speed)
    GPIO.output (M3_IN1, False)
    GPIO.output (M3_IN2, True)
    pwm3.ChangeDutyCycle(speed)
    GPIO.output (M4_IN1, False)
    GPIO.output (M4_IN2, True)
    pwm4.ChangeDutyCycle(speed)

def Motor_Stop ():
    GPIO.output (M1_IN1, False)
    GPIO.output (M1_IN2, False)
    pwm1.ChangeDutyCycle(0)
    GPIO.output (M2_IN1, False)
    GPIO.output (M2_IN2, False)
    pwm2.ChangeDutyCycle(0)
    GPIO.output (M3_IN1, False)
    GPIO.output (M3_IN2, False)
    pwm3.ChangeDutyCycle(0)
    GPIO.output (M4_IN1, False)
    GPIO.output (M4_IN2, False)
    pwm4.ChangeDutyCycle(0)


# Main Program
Motor_Setup ()
pwm1 = GPIO.PWM (M1_EN1, 100)
pwm2 = GPIO.PWM (M2_EN1, 100)
```

```
pwm3 = GPIO.PWM (M3_EN1, 100)
pwm4 = GPIO.PWM (M4_EN1, 100)
pwm1.start (0)
pwm2.start (0)
pwm3.start (0)
pwm4.start (0)

Motor_Stop ()

button_delay = 0.1

print 'Press 1 + 2 on your Wii Remote'
time.sleep(1)

# Connect to the Wii Remote
try:
    wii=cwiid.Wiimote()
except RuntimeError:
    print 'Error Connecting to Wii Remote'
    quit()

print 'Wii Remote connected'

wii.rpt_mode = cwiid.RPT_BTN


# Loop
try:
    while True:

        buttons = wii.state['buttons']

        stop = True

        if (buttons & cwiid.BTN_LEFT):
            Motor_Left (speed)
            time.sleep(button_delay)
            stop = False

        if(buttons & cwiid.BTN_RIGHT):
            Motor_Right(speed)
            time.sleep(button_delay)
            stop = False

        if (buttons & cwiid.BTN_UP):
            Motor_Forward (speed)
            time.sleep(button_delay)
            stop = False

        if (buttons & cwiid.BTN_DOWN):
            Motor_Reverse (speed)
            time.sleep(button_delay)
            stop = False


        if (buttons & cwiid.BTN_MINUS):
            speed = speed - 1
            if (speed < 0):
                speed = 0
            print 'speed =', speed
            time.sleep(button_delay)

        if (buttons & cwiid.BTN_PLUS):
            speed = speed + 1
            if (speed > 100):
                speed = 100
            print 'speed =', speed
            time.sleep(button_delay)

        if (stop == True):
            Motor_Stop ()

except KeyboardInterrupt:
    pass

print "End"
GPIO.output (M1_EN1, False)
GPIO.output (M1_IN1, False)
GPIO.output (M1_IN2, False)
GPIO.output (M2_EN1, False)
GPIO.output (M2_IN1, False)
GPIO.output (M2_IN2, False)
GPIO.output (M3_EN1, False)
GPIO.output (M3_IN1, False)
GPIO.output (M3_IN2, False)
GPIO.output (M4_EN1, False)
GPIO.output (M4_IN1, False)
GPIO.output (M4_IN2, False)

pwm1.stop ()
pwm2.stop ()
pwm3.stop ()
pwm4.stop ()

GPIO.cleanup
exit (wii)
```