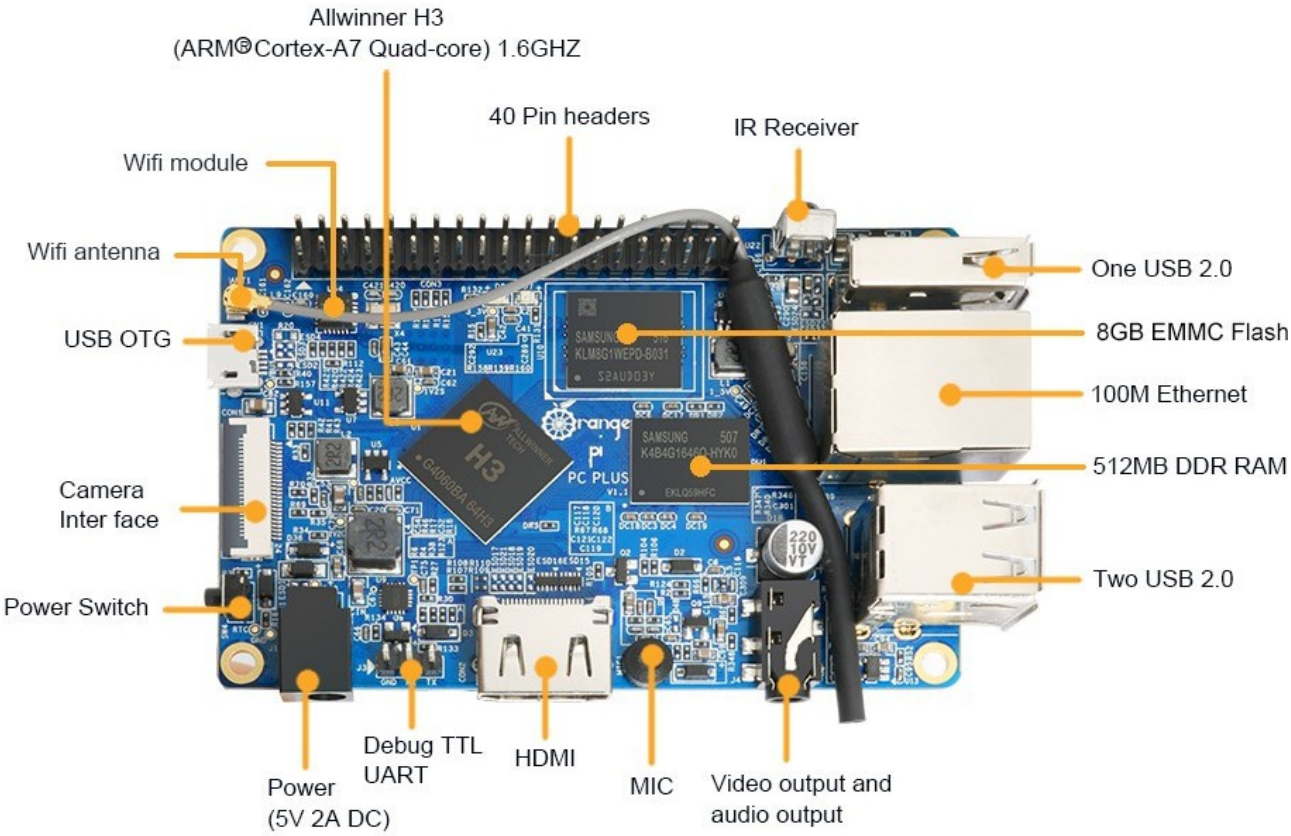
WORKSHOP ORANGE PI & PYTHON

v1.1 – 21 september 2017

In deze workshop maak je kennis met een zogenaamde bord-computer ter grootte van een bankpas. Een bord-computer kan gebruikt worden als een normale computer maar is zeer klein. Er bestaan veel verschillende soorten. De Raspberry Pi is de meest bekende, maar er zijn er meer! Wij gaan aan de slag met de Orange Pi.



Op deze bord-computers draait een volledig besturingssysteem. Een bekend besturingssysteem is Windows. Om de kosten laag te houden wordt gebruik gemaakt van een Linux besturingssysteem. Linux is de basis van bijvoorbeeld Android (smartphones). Hier gebruiken we Armbian.

Met de bord-computer kun je E-mailen, internet browsen, tekst verwerken etc. In deze workshop gaan we aan de slag met programmeren. Programmeren kan in verschillende talen. Misschien heb je wel eens gehoord van Scratch. Hier gaan we aan de slag met Python.

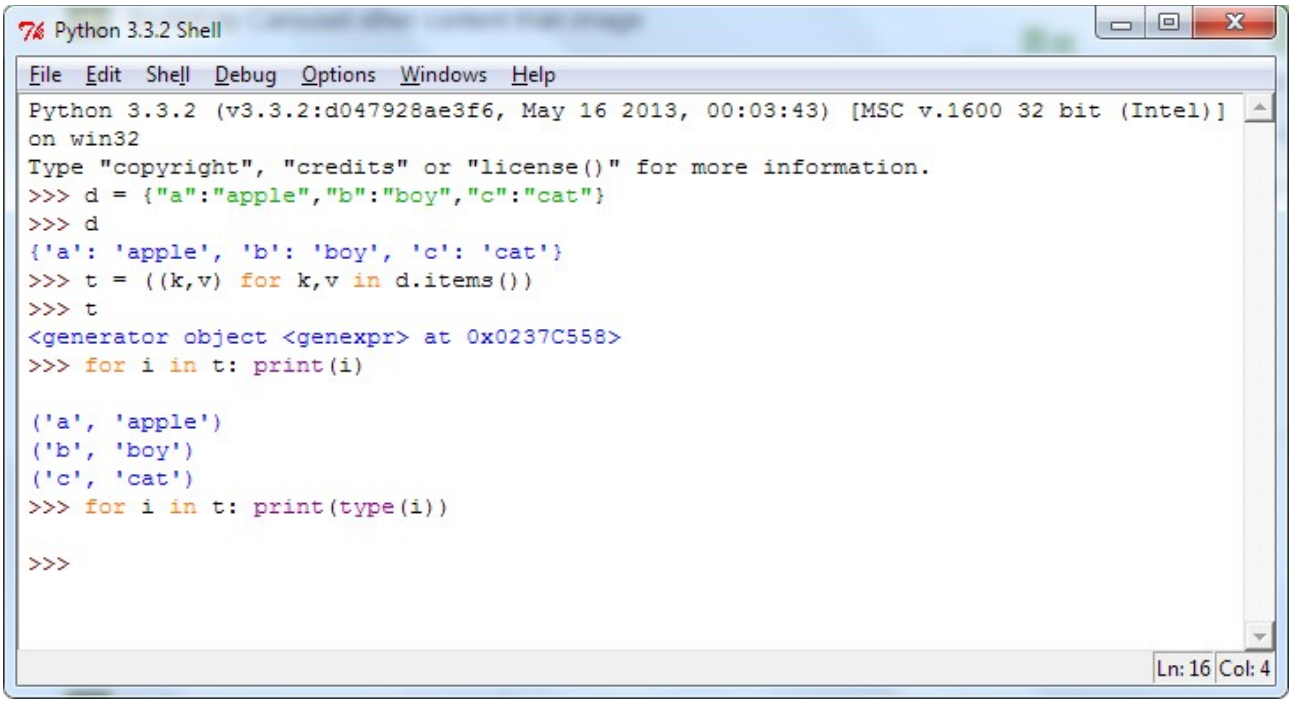
Python is een Open-source programmeertaal. Je mag hem dus gratis gebruiken.

De Orange Pi heeft naast de gebruikelijke computerpoorten, zoals USB, ethernet en video, ook een speciale poort: de GPIO-poort. GPIO staat voor “General Purpose Input Output”, oftwel besturingspoort voor algemeen gebruik. Deze poort kan gebruikt worden als ingang en als uitgang. De poort bestaat uit een aantal pennen, elke aansluitpen kan individueel geprogrammeerd worden.

De pennen kunnen gebruikt worden voor digitale signalen. Een digitaal signaal kent twee “logische” toestanden, een toestand wordt een niveau genoemd.

Niveau	Codering	Spanning op de pen	Aangesloten LED
Hoog niveau	Logische “1”	+3,3 Volt op de pen	LED staat aan
Laag niveau	Logische “0”	0 Volt op de pen	LED saat uit

Deze pennen (hardware) kunnen bestuurd worden vanuit een programma, de software. De software bestaat uit commando's. Deze commando's kunnen ingevoerd worden met behulp van een “Editor”. De editor is een tekst-verwerker, net zoiets als Word, maar dan speciaal voor het invoeren van programma's. Voor Python heet deze editor “ Idle”.



Idle kan gestart worden vanuit het Start-menu (Applications>Development>Idle). In Idle kan nu het programma geschreven worden, het programma wordt vervolgens opgeslagen. Om het programma uit te voeren wordt een zogenaamde “terminal” geopend. Dit kan vanuit het start-menu.

Als je een programma wilt starten type je bij de Command-prompt: sudo python <programma-naam>.py (LET OP, dit hoef je nu nog niet te proberen!)

Het commando “sudo” wordt gebruikt om een programma uit te voeren als “root”. De root is de gebruiker met alle rechten op een systeem. “sudo” is een afkorting voor Super User DO.

Het password is “scout”.

Programma's die geschreven worden in Python kunnen gebruik maken van het beeldscherm van het

systeem en het toetsenbord, maar ook van de GPIO-pinnen. Zo kan een progamma dus ook externe componenten aansturen (LEDs) en uitlezen (bijvoorbeeld drukschakelaars).

VARIABELEN

Er is eigenlijk geen programma wat geen gebruik maakt van “variabelen”. Een variabele is een plaats in het geheugen van de computer. De variabele heeft een naam en een waarde (getal). Bijvoorbeeld:

Naam	Waarde
a	2
b	4
resultaat	0

De variabelen kunnen vervolgens gebruikt worden in het programma, bijvoorbeeld:

resultaat = a x b → resultaat = 2 x 4 = 8 → resultaat heeft nu waarde 8 gekregen

of:

resultaat = a + b → resultaat = 2 + 4 = 6 → resultaat heeft nu waarde 6 gekregen

Probeer zelf eens:

resultaat = b – a = …...

resultaat = 2 x a = ……

Deze berekeningen kun je ook op de Orange Pi uitvoeren, direkt in Idle. Start Idle via “Applications>Development>Idle”.

```
Type:           b = 4 <enter>
Type:           a = 2 <enter>
Type:           b-a <enter>

Type:           2*a <enter>
```

Een programma communiceert met de gebruiker via het scherm en het toetsenbord. Resultaten kunnen weergegeven worden via het scherm. Hiervoor maken we gebruik van het commando “print”. Achter commando print geef je aan wat er op het scherm gezet moet worden. Zet je dit tussen aanhalingstekens, dan komt op het scherm te staan wat tussen de aanhalingstekens staat.

print “resultaat” → geeft op het scherm: resultaat

Op het moment dat je geen aanhalingstekens gebruikt, dan geeft het commando print de waarde van de variabele:

print resultaat → geeft op het scherm 4

In de voorbeelden hierboven zijn alle getallen van het type “integer”, dit betekend dat het een geheel getal is. Bijvoorbeeld het aantal scouts in een patrouille. Je kunt ook gebroken getallen invoeren, dit wordt een “real” genoemd. Voorbeeld hiervan is bijvoorbeeld het aantal gelopen kilometers in een hike, bijvoorbeeld 6,75 km. Maar let op, geven we dit aan in meters, dan is het weer een integer (geheel getal): 6750 meter. Of een getal een real of een integer is, bepaald Python zelf, door te kijken hoe de getallen aangegeven zijn; staat er een komma in, dan wordt het een real, anders een integer.

We gaan dit proberen door een eenvoudig programma te schrijven (typ het programma over in Idle). Om het programma in te kunnen voeren, ga je in Idle naar “File” en kies je “New file”. In het nieuwe window type je dit programma:

```
# doosjes is het aantal doosjes met Camilla's eieren
# inhoud is het aantal eieren in een doosje
# in totaal komt het totale aantal eieren te staan
doosjes = 4
inhoud = 6
totaal = 0
print “aantal doosjes eieren: “ , doosjes
# in een computerprogramma betekend een * hetzelfde als keer
(vermenigvuldigen)
totaal = doosjes * inhoud
print “het totaal aantal eieren is: “ , totaal
```

Sla het programma op **(File > Save → Desktop → Camilla)**. Ga vervolgens naar het Terminal venster **(Applications > Terminal Emulator)** en start het programma **(typ: cd Desktop <enter> en vervolgens sudo python Camilla.py <enter>)**. Voer als password “**scout**” in. Wat komt er uit? Zijn doosjes en inhoud nu integers of reals?

Een programma, zoals je dat net geprobeerd hebt, heeft nog niet heel veel nut. Het wordt nuttiger als de gebruiker zelf de getallen in kan voeren.

Om getallen in te kunnen voeren gebruiken we het commando “raw_input”. Pas het programma als volgt aan:

```
# doosjes is het aantal doosjes met Camilla's eieren
# inhoud is het aantal eieren in een doosje
# in totaal komt het totale aantal eieren te staan
doosjes = 1.0
inhoud = 1
totaal = 0
print “We gaan het totale aantal eieren berekenen.”
inhoud = raw_input  (“Het aantal eieren in een doosje is: “)
inhoud = int(inhoud)
doosjes = raw_input (“Hoeveel doosjes zijn er? “)
doosjes = int(doosjes)
# in een computerprogramma betekend een * hetzelfde als keer
(vermenigvuldigen)
totaal = doosjes * inhoud
print “het totaal aantal eieren is: “ , totaal
```

Om programma's beslissingen te laten nemen zijn er ook bepaalde instrukties. Hier gaan we twee soorten beslissingen bekijken:

- *if-then-else*
- *while*

if-then-else komt, net als andere instructies, uit het Engels. *If* betekent “als”, *then* betekent “dan” en *else* betekent “anders”.

Bijvoorbeeld:
if (als) geluid = tok *then (dan)* de vogel=kip *else (anders)* vogel=geen kip

Je ziet hier een voorbeeld van *if-then-else.* Als de bewering na *if* waar is, dan gaat het programma het deel uitvoeren na *then*. Als de bewering na *if* niet waar is, dan gaat het programma verder met wat na *else* staat. Voor “waar” wordt vaak de Engelse benaming *true* gebruikt, voor niet-waar wordt *false* gebruikt.

While kan het beste vertaald worden met “zolang”. Als er wachter *while* een bewering staat die waar is, dan wordt het stuk programma dat hierachter staat uitgevoerd.

Bijvoorbeeld:
while totaal=12
    print “twee dozen eieren”

while wordt heel vaak gebruikt bij programma-stukken die veel herhaald moeten worden,

GPIO-CONTROL

We gaan het commando *while* gebruiken in een programma dat een LED aanstuurd. Ga hiervoor in Idle naar het menu “File” en kies “New File”. Type het onderstaande programma in:

```
#import library
from pyA20.gpio import gpio
from pyA20.gpio import port
from time import sleep

#initialize the gpio module
gpio.init()

#setup the port
gpio.setcfg(port.PG7, gpio.OUTPUT)

while True:
     gpio.output(port.PG7, gpio.HIGH)
     sleep(1.5)
     gpio.output(port.PG7, gpio.LOW)
     sleep(1.5)

done
```

Ga naar **File>Save**, sla het programma op op de Desktop, gebruik bijvoorbeeld de naam **test1**.

Open via het menu Applications (links boven) een Terminal Emulator.

Type in de Terminal: **cd Desktop**
Type vervolgens: **sudo python test1.py**

Vervolgens vraagt de Orange Pi om het password, dat is "**scout**". Kijk nu wat er gebeurt met de LED.

Stop het programma met **<CTRL> C**.

Je kan nu proberen het programma wat je net hebt gemaakt zo aan te passen dat de LED bijvoorbeeld sneller gaat knipperen. Of breid het programma uit zodat de LED een morse boodschap uitzendt.

SIGNALEN INLEZEN – REACTIE-SNELHEIDSTESTER

Je hebt nu een voorbeeld gezien van een programma dat een signaal opwekt om (bijvoorbeeld) een LED aan te sturen. Je kunt ook signalen van buitenaf inlezen, in dit geval gaan we een drukschakelaar (druktoets) uitlezen en gebruiken voor een reaktie-snelheidstester.

```
#import the library
from pyA20.gpio import gpio
from pyA20.gpio import port
from time import sleep
from datetime import datetime
import random

#initialize the gpio module
gpio.init()

#setup the port
gpio.setcfg(port.PG7, gpio.OUTPUT)
gpio.setcfg(port.PC4, gpio.INPUT)

#main programma
gpio.output(port.PG7, gpio.LOW)
print ("Reactie-snelheidstester gestart")
sleep(2.0)

while True:
     random.seed()
     sleep(random.random()*10)
     start = datetime.now()
     gpio.output(port.PG7, gpio.HIGH)
     while gpio.input(port.PC4):
          pass
     print ("Je reactive-tijd is: ", (datetime.now() –
start).total_seconds())
     sleep(2.5)

done
```

Sla het programma op, bijvoorbeeld als “reactietester”. Start het programma via de terminal en kijk eens hoe het werkt!

Er lijkt iets nog niet te kloppen met het programma….. Als je door het programma leest (hierboven) kun je dan begrijpen wat er gebeurt? Heb je een idee hoe je dit kunt oplossen?

Tip: Gaat de LED uit nadat de reactie-tijd is gemeten?

Het programma kan nog verder uitgebreid worden. Bijvoorbeeld met de vraag of je verder wilt gaan, of met een valsspeel-detector.

Iemand kan makkelijk vals spelen door de knop ingedrukt te houden. Kun je het programma zelf zo aanpassen dat het programma deze situatie herkend?

(Mocht je er niet uitkomen, op de volgende bladzijde staat een voorbeeld).

```
#import the library
from pyA20.gpio import gpio
from pyA20.gpio import port
from time import sleep
from datetime import datetime
import random

#initialize the gpio module
gpio.init()

#setup the port
gpio.setcfg(port.PG7, gpio.OUTPUT)
gpio.setcfg(port.PC4, gpio.INPUT)

#main programma
doorgaan = 0
gpio.output(port.PG7, gpio.LOW)
print (“Reactie-snelheidstester gestart”)
sleep(2.0)

while True:
     random.seed()
     sleep(random.random()*10)
     while not gpio.input(port.PC4):
          print (“Niet vals spelen!”)
     start = datetime.now()
     gpio.output(port.PG7, gpio.HIGH)
     while gpio.input(port.PC4):
          pass
     print (“Your reaction time: “, (datetime.now() –
start).total_seconds())
     sleep(2.5)
     doorgaan = raw_input  (“Druk op Enter om door te gaan. “)

done
```

=====##### END #####=====