hogeschool vives

west-vlaanderen
de gedreven provincie

Python met Micro:Bit

Docenten Nele Moens, Michiel Van lede en Wim Dejonghe

studiegebied **Onderzoek**

campus **Kortrijk**

academiejaar **2022-2023**

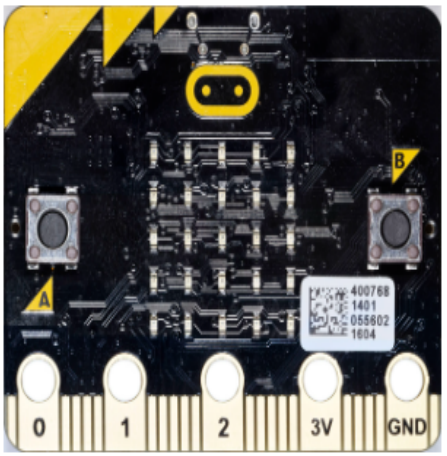
Legende van de gebruikte iconen

	Denkvraag
	Leerdoelen
	Formule
	Extra informatie
	Niet vergeten
	Opdracht/Oefening
	Presentatie (PowerPoint)
	Rekenblad (Excel)
	Samenwerking

	Studeeraanwijzingen
	Tijdsinschatting
	Toledo
	Beeld-/geluidsfragment
	Voorbeeld
	Tools/Apps
	Website
	Zelfstudie
	(Zelf)toets

Inhoudsopgave

Inleiding



De micro:bit is een klein (4x5cm) micro-computertje met knopjes voor de bediening, en een 5x5 led matrix als beeldscherm, dat met 4 verschillende programmeertalen kan worden geprogrammeerd. De meest laagdrempelige taal is de Blocks Editor, waarin het programma (script) op een grafische wijze wordt ontworpen, door blokjes te slepen en aan elkaar te hangen. Hierbij kunnen geen syntax fouten gemaakt worden, maar toch wordt de manier van denken van een programmeur aangeleerd. De praktijk laat zien dat kinderen, vanaf 10 jaar snel aan de slag gaan, met een minimum aan instructie.

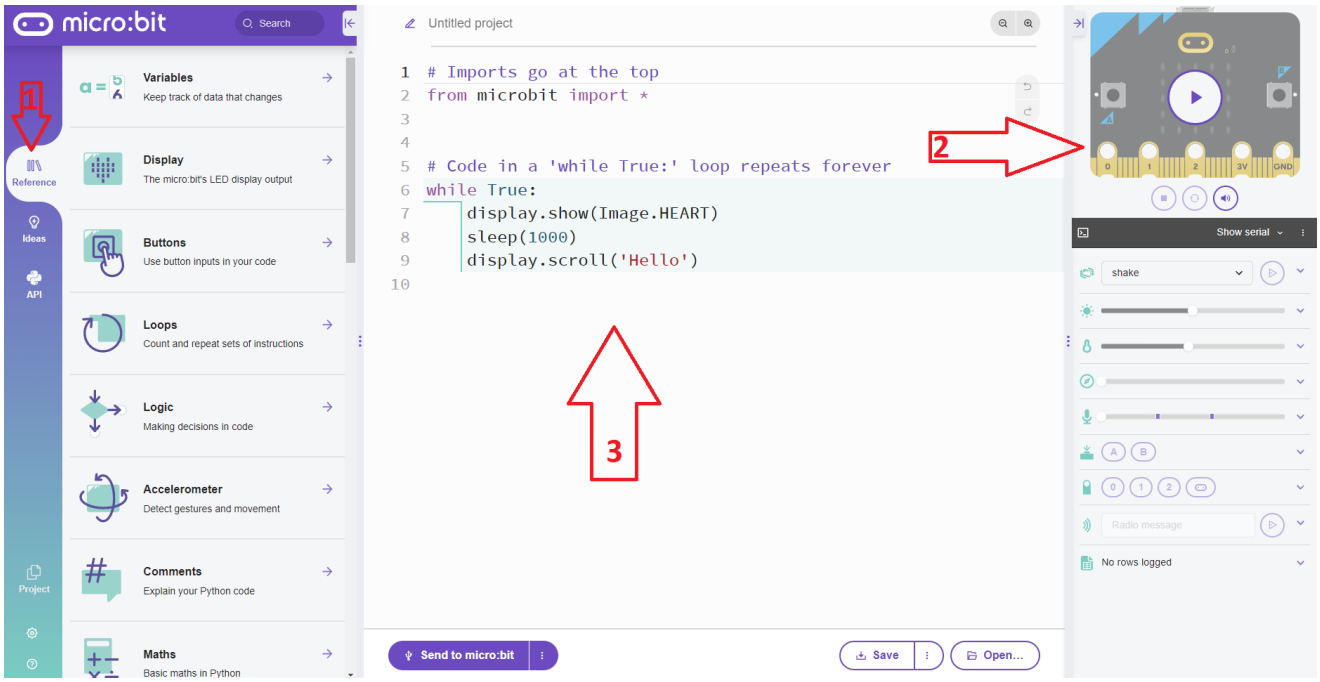
Verder beschikt de micro:bit over vele sensoren. Door gebruik te maken van de juiste Blocks kunnen temperatuur, licht, acceleratie (trillingen), oriëntatie en magneetvelden (kompasfunctie) gemeten worden en gebruikt worden in de zelf te maken toepassingen.

Daarnaast beschikt de micro:bit over draadloze communicatie met andere micro:bits, en met smart-phones en tablets.

Hiermee is micro:bit dus tevens een IoT (Internet of Things) platform dat ook voor gevorderde gebruikers interessant is. Voor de meer geavanceerdere toepassingen kan gebruik worden gemaakt van programmeertalen zoals Java-Script en Python.

Last-but-not-least is micro:bit zeer betaalbaar (± € 20), waarmee het gebruik door scholen laagdrempelig is en elk kind er in principe zelf één zou kunnen hebben.

Microbit.org



1. Python code bibliotheek
2. Simulatiezone
3. Programmeer zone

MicroPython op Micro:Bit

Start:

Initialisatie : Omdat we werken met een Micro:bit moeten we dit steeds aangeven bovenaan de code. Importeer steeds (*) om alle (hardware)mogelijkheden van de Micro:Bit te gebruiken. Doe dit door bovenaan in de python code te schrijven :

```
from microbit import *
#Dan de rest van de python code
```

1. Toon cijfers op LED display:

- *display.show(<nummer>)*
- *display.scroll(<nummer>)*

Opdracht :

Toon 1 cijfer

Toon getal met meerdere cijfers

2. Voer 1 keer uit / herhaal altijd (oneindige-iteratie):

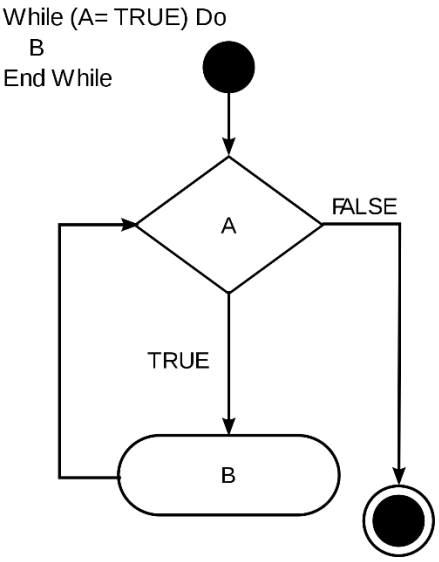
1 keer

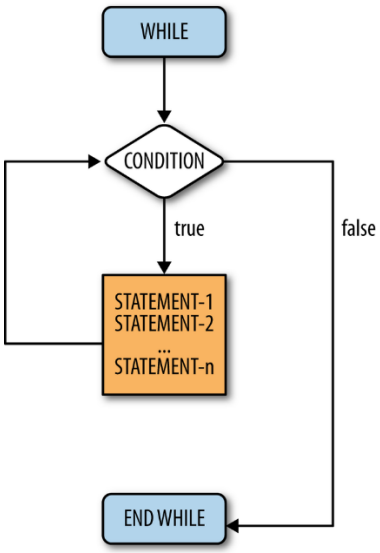
```
1  # Imports go at the top
2  from microbit import *
3
4  #volgende statements worden maar 1 keer uitgevoerd
5  display.scroll(1234)
```

Altijd – continue

while-loop met een altijd WAAR voorwaarde (True)

```
1  # Imports go at the top
2  from microbit import *
3
4  # Code in a 'while True:' loop repeats forever
5  while True:
6      display.scroll(1234)
7
```


While (A= TRUE) Do
B
End While
A
FALSE
TRUE
B


WHILE
CONDITION
true
false
STATEMENT-1
STATEMENT-2
...
STATEMENT-n
END WHILE

3. Tel af (en schrijf tekst):

Tekst die bestaat uit meerdere tekens, omsloten door enkele of dubbele aanhalingtekens wordt een **string** genoemd . (“string”)

Een getal kan een **integer** zijn (geheel getal) of een **float** (een komma getal)(**Let op!** moet een punt zijn i.p.v. een komma)

Variabele : We kunnen ook iets opslaan in het geheugen van de computer (= **declaratie** = reservatie van geheugenplaatsen). Aantal geheugenplaatsen is afhankelijk van het type variabele (int, float, string, bool). Een variabele krijgt steeds een (logische)naam die de programmeur zelf kan kiezen.

type variabele	inhoud	python syntax	voorbeeld
string	tekst (ascii)	str	naam = "John"
Integer	geheel getal	int	getal = 3
Float	komma getal	float	waarde = 27.9
Boolean	True/False	bool	actief = True

```
1  # Imports go at the top
2  from microbit import *
3
4  #volgende statements worden maar 1 keer uitgevoerd
5  display.show(4)
6  sleep(500)
7  display.show(3)
8  sleep(500)
9  display.show(2)
10 sleep(500)
11 display.show(1)
12 sleep(500)
13 display.show(0)
14 sleep(500)
15
16 # Code in a 'while True:' loop repeats forever
17 while True:
18     display.show(Image.HEART)
19     sleep(1000)
20     display.scroll('Hello')
21
```

Opdracht:

Vervang het getal 500 door een variabele (type = integer) met de naam: **wacht_tijd**

```
# Imports go at the top
from microbit import *

#declareer de variabele met de naam wacht_tijd
#en stop integer getal 500 in de variabele
wacht_tijd = 500

display.show(4)
sleep(wacht_tijd)  #lees de inhoud van de variabele
display.show(3)
sleep(wacht_tijd)
display.show(2)
sleep(wacht_tijd)
display.show(1)
sleep(wacht_tijd)
display.show(0)
sleep(wacht_tijd)


# Code in a 'while True:' loop repeats forever
while True:
    display.show(Image.HEART)
    sleep(1000)
    display.scroll('Hello')

```

Opdracht:

*Zet eens de variabele **wacht_tijd** op waarde 0, of laat de "sleep "eens weg.*

4. Tel danswedstrijd af :

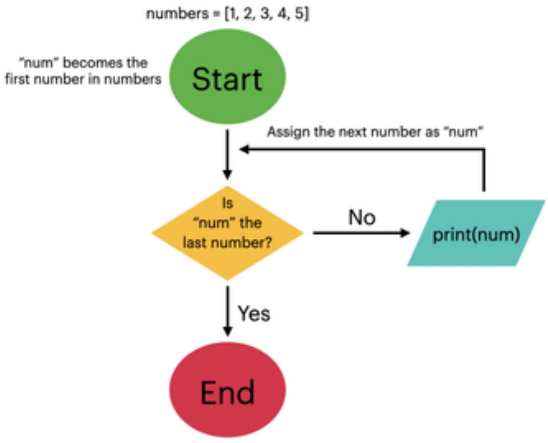
Teken een popje en laat het bewegen

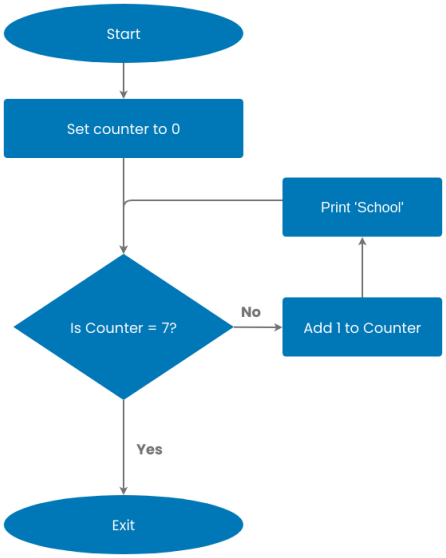
```
# Imports go at the top
from microbit import *

#Volgende statements worden maar 1 keer uitgevoerd
display.clear()
sleep(1000)

display.show(3)
sleep(500)
display.show(2)
sleep(500)
display.show(1)
sleep(500)
display.show(0)
sleep(1000)

display.scroll('Wim Dejonghe')
sleep(1000)


# Code in a 'while True:' loop repeats forever
while True:
    display.show(Image('90900:'
                       '09990:'
                       '00909:'
                       '09090:'
                       '90090'))
    sleep(300)
    display.show(Image('90909:'
                       '09990:'
                       '00900:'
                       '09090:'
                       '90009'))
    sleep(300)
```

Aftellen kan ook in een iteratie (*for-loop*) gebeuren.

For loop : maakt een variabele en laat die in een *loop* telkens eentje verhogen.

Variabele met naam **a** is van het type integer






```
for a in range(4):
    display.show(a)
    sleep(500)
```

- **Concatenatie** is het samenvoegen van twee strings aan elkaar:

  Samengevoegde string = string1 + string2

- Converteren van een type naar een ander type = **type casting**

  tekstwaarde = “6”

  naarInteger = int(tekstwaarde)

Opdracht: Zorg dat het aftellen wordt getoond!
Opdrachten: Er zitten ook standaard figuurtjes die kunnen worden gebruikt bij display.show : Zoek die eens.

- Maak een kloppend hartje met een zichtbaar interval.
- Display uw naam

5.Communicatie met de computer via USB kabel

De Micro:Bit is ook in staat om te communiceren met de computer.

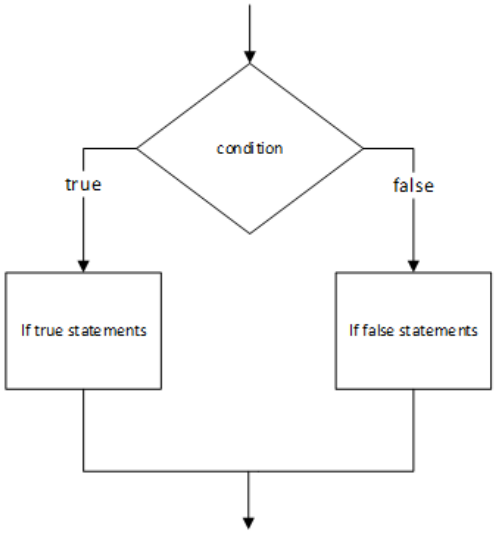
Dit kan als volgt:

```
1  # Imports go at the top
2  from microbit import *
3
4
5  # Code in a 'while True:' loop repeats forever
6  while True:
7      name = input('What is your name? ')
8      print('Hello', name)
9      display.scroll(name)
10     getal = input('Geef een getal :')
11     for i in range(int(getal)):
12         display.show(Image.HEART)
13         sleep(250)
14         display.show(Image.HEART_SMALL)
15         sleep(250)
```

Let op in de code: het inlezen (input) is steeds van het type String. Aangezien we dit als een getal willen gebruiken moet hier dus een typecasting gebeuren van een string naar een integer. Let wel, dit kan fout lopen indien er geen omzetbaar symbool wordt ingegeven.

6. Gebeurtenis op basis van drukknoppen knopA

Binnen de oneindige WHILE-loop kan er steeds nagegaan worden of een drukknop is ingedrukt. Hiervoor gebruiken we een IF-statement (keuze optie: levert altijd een True/False op).



Binnen de Micro:Bit kan gebruik gemaakt worden van:

- WAS_pressed : statements worden slechts één keer uitgevoerd (ook bij blijvend drukken)
- IS_pressed : statements worden meerdere keren uitgevoerd bij blijvend drukken

```
1  # Imports go at the top
2  from microbit import *
3
4
5  # Code in a 'while True:' loop repeats forever
6  while True:
7      if button_a.was_pressed():        #Eenmalig uitgevoerd bij press
8          display.scroll('Wim')         #Er wordt gewacht tot einde om volgend statement uit te voeren
9      if button_b.is_pressed():          #Wordt herhaaldelijk uitgevoerd bij enkele press
10         display.scroll('Dejonghe')
```

Vergelijkingsoperatoren:

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

7.Event op knop A en B

Met twee knoppen kan je dus drie situaties hebben.
Logische poorten kunnen worden gebruikt

Logische operatoren : **and or not(!)**

```
# Imports go at the top
from microbit import *


# Code in a 'while True:' loop repeats forever
while True:
    if button_a.is_pressed() and button_b.is_pressed():   #altijd eenmlig uitgevoerd als a&b worden ingedrukt
        display.scroll('A and B')
```

Drie combinaties met twee drukknoppen:

A and notB / notA and B / A and B

```
# Imports go at the top
from microbit import *


# Code in a 'while True:' loop repeats forever
while True:
    if button_a.is_pressed() and button_b.is_pressed():   #altijd eenmlig uitgevoerd als a&b worden ingedrukt
        display.scroll('A&B')
    if (button_a.is_pressed() and (not button_b.is_pressed())):
        display.show('A')
    if (button_b.is_pressed() and (not button_a.is_pressed())):
        display.show('B')
```

8. Tellen en weergeven van het aantal keren gedrukt op knop A.

Hier gaan we het aantal kliks op drukknop A bijhouden in een variabele (integer) met de naam **teller**. Telkens als er op de drukknop A wordt gedrukt moet de waarde van **teller** uit het geheugen gehaald worden (lezen), en moet daar 1 bij opgeteld worden. Daarna schrijven we het resultaat van die optelling terug naar het geheugen in dezelfde variabele **teller** (klaar voor de volgende klik).

Teller kan gereset worden door op drukknop B te drukken.

```
1  # Imports go at the top
2  from microbit import *
3
4  #declaratie van de teller variabele
5  teller = 0
6
7  # Code in a 'while True:' loop repeats forever
8  while True:
9      if button_a.was_pressed() == True:
10         teller = teller + 1
11     if button_b.was_pressed():
12         teller = 0
13
14     display.scroll(teller)
```

Opdracht : Analyseer de werking.
Programmeer een tafel van vermenigvuldiging. Programmeer een toepassing waarbij je met A een teller tot een bepaalde waarde brengt. Door dan op B te drukken laat je een lichtje zoveel keer knipperen.

Operatoren voor berekeningen met getallen:

```
+   optelling
-   aftrekking
*   vermenigvuldiging
/   deling
//  integer deling
**  machtsverheffing
%   modulo
```

9. Geluidje afspelen wanneer op Knop A wordt gedrukt

De Micro:Bit bevat enkele actuatoren. De Led matrix is een actuator, maar er zit ook een microfoon op de Micro:Bit die ook invers kan worden gebruikt om geluid af te spelen.

Een voorbeeldje van een sweep geluid tussen twee frequenties in een bepaalde tijd kan zo geactiveerd worden:

```
1  # Imports go at the top
2  from microbit import *
3
4  # Code in a 'while True:' loop repeats forever
5  while True:
6      if button_a.was_pressed():
7          my_effect = audio.SoundEffect(freq_start=400, freq_end=2500, duration=500)
8          audio.play(my_effect)
```

10. Parking versie1

Hier maken we een parking-display met aanduiding van het aantal bezette plaatsen.

Gegeven:

Er zijn 10 plaatsen.

Drukknop A => registratie aan de ingang
Drukknop B => registratie aan de uitgang

Gevraagd:

Maak een teller zorg ervoor dat de teller niet boven de *10 plaatsen* gaat en ook niet onder de *0 plaatsen* gaat.

```
1  # Imports go at the top
2  from microbit import *
3
4  #declaratie variabelen met toekenning van een waarde
5  max_plaatsen = 10
6  plaatsenbezet = 0
7
8
9  # Code in a 'while True:' loop repeats forever
10 while True:
11     if button_a.was_pressed():
12         if plaatsenbezet < max_plaatsen:
13             plaatsenbezet = plaatsenbezet + 1
14     if button_b.was_pressed():
15         if plaatsenbezet > 0:
16             plaatsenbezet = plaatsenbezet - 1
17     display.scroll(plaatsenbezet)
```

11. Lopende LED

display.set_pixel(x,y,helderheid)

Met dit Micro:Bit statement kan een bepaalde pixel binnen de LED matrix worden aangestuurd.

Er wordt hier gebruik gemaakt van twee geneste FOR-loops om een loop-sequentie te programmeren.

1.Python

```
1   # Imports go at the top
2   from microbit import *
3
4
5   # Code in a 'while True:' loop repeats forever
6   while True:
7       for y in range(5):
8           for x in range(5):
9               display.set_pixel(x, y, 9)
10              sleep(50)
11      sleep(1000)
12      display.clear()
13      sleep(1000)
```

Opdracht : Analyseer de werking en programmeer nog andere sequenties.

12. Parkeerplaatsen versie2

De bezetten plaatsen / vrije plaatsen kunnen we ook grafisch aanduiden op de LED-matrix. Er zijn 5*5 LED's, zo kunnen we een parkeergarage van 25 plaatsen visualiseren.

Rekenkundige bewerkingen: de normale bewerkingen spreken voor zich: + - * die kunnen zowel met integer- als met float waarden werken. Let wel : een computer kijkt steeds om een bewerking zo eenvoudig mogelijk uit te voeren.

Dwz, als de parameters integer waarden zijn , dan zal de computer de bewerking ook uitvoeren met integer waarden en het resultaat zal ook een integer zijn.

Is echter een parameter een float dan zal de computer een complexere berekening uitvoeren en zal het resultaat ook een float zijn.

Met deze redenering is er echter een probleem bij de deling / . Twee integer waarden delen zal steeds een integer opleveren. Dit kan dus tot een onvolledige deling leiden. Vb:

10/5 = 2
5/2=2 !!!! volledig behandeld met integers
5.0/2=2.5

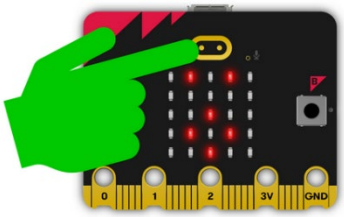
Met integers kan ook een restdeling worden uitgevoerd (modulo deling). Vb:

11%4 = 11 mod 4 = 3

Hiermee kunnen we de coördinaten van een LED x, y bepalen uit een groter getal

```
# Imports go at the top
from microbit import *

max_plaatsen = 25
plaatsenbezet = 0
yPos = 0
xPos = 0
ledWaarde = 0

# Code in a 'while True:' loop repeats forever
while True:
    if button_a.was_pressed():
        if plaatsenbezet < max_plaatsen:
            plaatsenbezet = plaatsenbezet + 1
            ledWaarde = plaatsenbezet - 1
            xPos = ledWaarde % 5
            yPos = int(ledWaarde / 5)
            display.set_pixel(xPos, yPos, 9)
    if button_b.was_pressed():
        if plaatsenbezet > 0:
            plaatsenbezet = plaatsenbezet - 1
            ledWaarde = plaatsenbezet
            xPos = ledWaarde % 5
            yPos = int(ledWaarde / 5)
            display.set_pixel(xPos, yPos, 0)
```

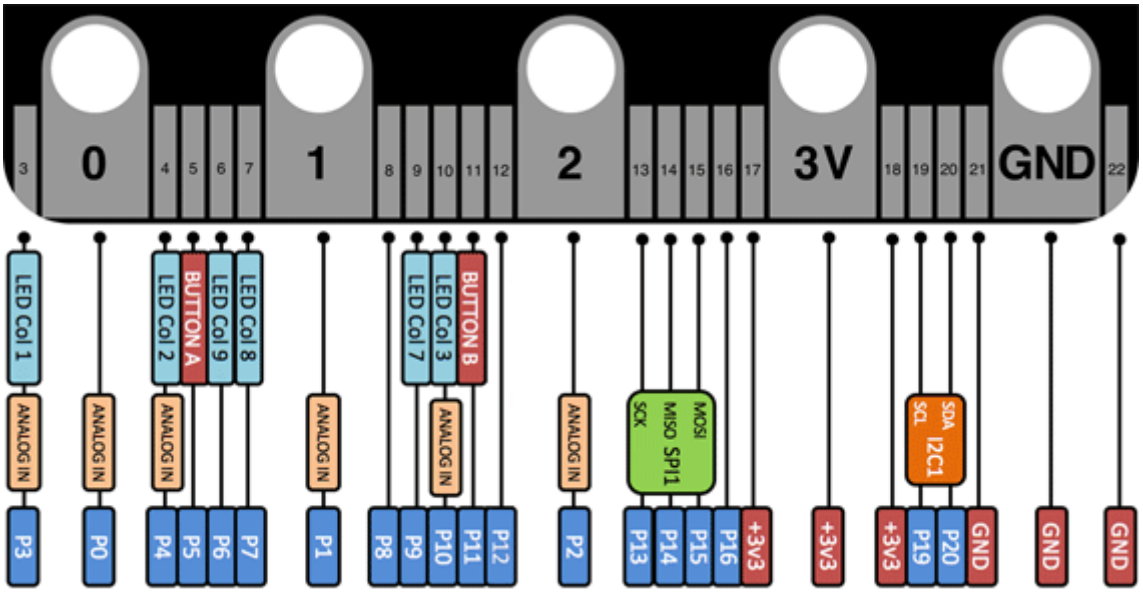
13. Touch logo



De Micro:Bit bevat ook een touch-logo. Dit is een soort drukknop die werkt op capacitieve verandering. Het volstaat om met de vinger hier dicht in de buurt te komen om de drukknop te activeren (analogie met een touch-screen)

```
1   # Imports go at the top
2   from microbit import *
3
4
5   # Code in a 'while True:' loop repeats forever
6   while True:
7       if pin_logo.is_touched():
8           display.show(Image.HAPPY)
9           sleep(100)
10          display.clear()
```

14. Externe pinnen



De meeste pinnen kunnen gebruikt worden als digitale in/uitgang. Sommige pinnen kunnen ook een als een analoge ingang worden gebruikt (om bv een analoge sensor die een analoge elektrische spanning afgeeft ifv een te meten grootheid als druk, temperatuur, geluid, licht, CO2, ….). Nog andere pinnen kunnen worden gebruikt binnen een bus-protocol standaard (SPI, I2C).

Zelf een drukknop maken kan eenvoudig door een elektrische verbinding te maken tussen de 0-pin en de GND. Dit met een elektrisch geleidende stof en niet met een isolator.

```
# Imports go at the top
from microbit import *

# Code in a 'while True:' loop repeats forever
while True:
    if pin0.is_touched():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Opdracht: maak een knop met aluminiumfoliestrookjes en krokodil-klemmen. Is een menselijk lichaam een geleider of een isolator?

15. Capacitieve Touch pin P0

Een pin kan ook veel gevoeliger worden gemaakt door die als een touch-sensor te laten werken (capacitief). Hierbij is geen verbinding nodig naar de GND als je met je vinger in de buurt komt van P0. Dit kan door éénmalig de touch_mode op die pin als capacitive in te stellen.:

```
1  # Imports go at the top
2  from microbit import *
3
4
5  pin0.set_touch_mode(pin0.CAPACITIVE)
6
7
8  # Code in a 'while True:' loop repeats forever
9  while True:
10     if pin0.is_touched():
11         display.show(Image.HAPPY)
12     else:
13         display.show(Image.SAD)
```

16. Dobbelsteen (random integer)

De Micro:Bit kan een random getal genereren tussen bepaalde grenzen. Hiervoor moet echter wel een aparte bibliotheek worden geïmporteerd. Met random.randint(ondergrens, bovengrens) kan dus een getal worden gegenereerd.

```
1  # Imports go at the top
2  from microbit import *
3  import random
4
5  # Code in a 'while True:' loop repeats forever
6  while True:
7      if button_a.was_pressed():
8          display.show(random.randint(1,6))
```

Uitbreiding: maak op de LED matrix de visualisatie van een echte dobbelsteen.

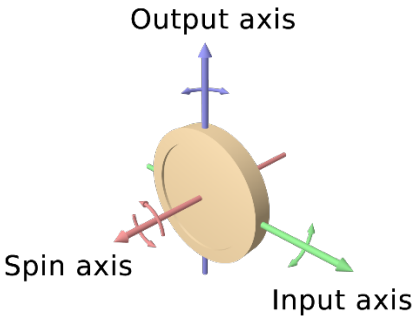
```
# Imports go at the top
from microbit import *
import random

# Code in a 'while True:' loop repeats forever
while True:
    if button_a.was_pressed():
        waarde = random.randint(1,6)
        #display.show(waarde)
        if waarde == 1:
            display.clear()
            display.set_pixel(2,2,9)
        elif waarde == 2:
            display.clear()
            display.set_pixel(0,0,9)
            display.set_pixel(4,4,9)
        elif waarde == 3:
            display.clear()
            display.set_pixel(0,0,9)
            display.set_pixel(2,2,9)
            display.set_pixel(4,4,9)
        elif waarde == 4:
            display.clear()
            display.set_pixel(0,0,9)
            display.set_pixel(0,4,9)
            display.set_pixel(4,0,9)
            display.set_pixel(4,4,9)
        elif waarde == 5:
            display.clear()
            display.set_pixel(2,2,9)
            display.set_pixel(0,0,9)
            display.set_pixel(0,4,9)
            display.set_pixel(4,0,9)
            display.set_pixel(4,4,9)
        else:
            display.clear()
            display.set_pixel(0,2,9)
            display.set_pixel(4,2,9)
            display.set_pixel(0,0,9)
            display.set_pixel(0,4,9)
            display.set_pixel(4,0,9)
            display.set_pixel(4,4,9)
```

17. Functies/Methoden met een parameter

Je kan ook een aparte functie maken om de Leds aan te sturen in python. Functies worden binnen programmeertalen ook als methoden benoemd of subroutines.

```
# Imports go at the top
from microbit import *
import random
def toonLeds(getal):
    if getal == 1:
        display.clear()
        display.set_pixel(2,2,9)
    elif getal == 2:
        display.clear()
        display.set_pixel(0,0,9)
        display.set_pixel(4,4,9)
    elif getal == 3:
        display.clear()
        display.set_pixel(0,0,9)
        display.set_pixel(2,2,9)
        display.set_pixel(4,4,9)
    elif getal == 4:
        display.clear()
        display.set_pixel(0,0,9)
        display.set_pixel(0,4,9)
        display.set_pixel(4,0,9)
        display.set_pixel(4,4,9)
    elif getal == 5:
        display.clear()
        display.set_pixel(2,2,9)
        display.set_pixel(0,0,9)
        display.set_pixel(0,4,9)
        display.set_pixel(4,0,9)
        display.set_pixel(4,4,9)
    else:
        display.clear()
        display.set_pixel(0,2,9)
        display.set_pixel(4,2,9)
        display.set_pixel(0,0,9)
        display.set_pixel(0,4,9)
        display.set_pixel(4,0,9)
        display.set_pixel(4,4,9)
# Code in a 'while True:' loop repeats forever
while True:
    if button_a.was_pressed():
        waarde = random.randint(1,6)
        toonLeds(waarde)
```

18. Gebeurtenis (event) bij schudden



De Micro:Bit bezit een gyrosensor die op de in de drie dimensies een versnelling kan waarnemen. Schudden is een actie die in de drie dimensies een versnelling kan veroorzaken. De Micro:Bit kan dit waarnemen en dit als voorwaarde worden gebruikt om een handeling uit te voeren. Is vergelijkbaar met een event op een knop maar nu niet drukken maar schudden.

Opdracht : laat de Micro:Bit een random getal genereren (0-100) bij het schudden van de Micro:Bit en geef dit getal weer op het LED-display.

```
1  # Imports go at the top
2  from microbit import *
3  import random
4
5  # Code in a 'while True:' loop repeats forever
6  while True:
7
8      if accelerometer.was_gesture('shake'):
9          waarde = random.randint(1,100)
10         display.show(waarde)
```

19. Schudden : Lovemeter :

Idem als vorige opdracht maar nu interpreteren we de random waarde naar een symbool.

- Getal tussen 0-30 : display.show(Image.SAD)
- Getal tussen 30-60 : display.show(Image.SMILE)
- Getal tussen 60-100 : display.show(Image.HAPPY)

Untitled project

```
# Imports go at the top
from microbit import *
import random

# Code in a 'while True:' loop repeats forever
while True:

    if accelerometer.was_gesture('shake'):
        waarde = random.randint(1,100)
        #display.show(waarde)
        if waarde < 30:
            display.show(Image.SAD)
        elif waarde < 60:
            display.show(Image.SMILE)
        else:
            display.show(Image.HAPPY)
```

20. Analyse sensoren

Op de Micro:Bit zitten al een aantal sensoren die in python kunnen worden uitgelezen. Onboard:

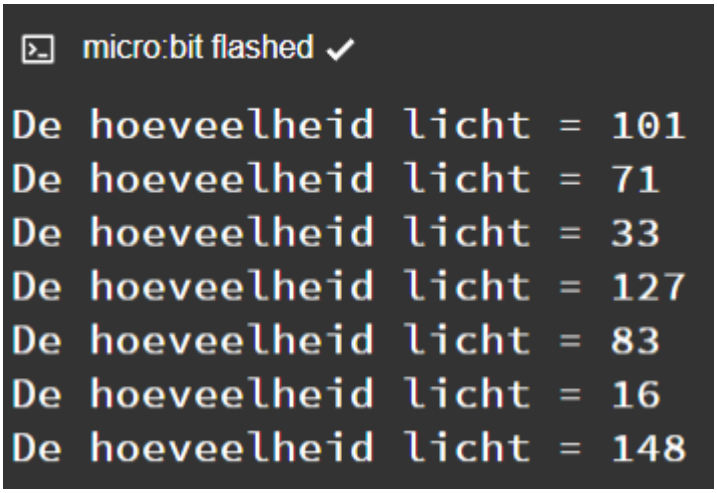
- Gyro sensor : 3-dimensionele versnelling (g-force)
- Magnetisme : 3-dimensionele magnetische veldsterkte (µ-tesla)
- Geluidsensor
- Temperatuursensor
- Lichtsensor
- Voel-sensoren (drukknoppen, en touch sensor, externe: pinnen)

Om een analyse (studie) te maken over de werking en verwerking van data binnen de Micro:Bit kunnen we gebruik maken van het print() python statement. De Micro:Bit kan zeker uitgebreid worden met externe sensoren door die juist aan te sluiten op de beschikbare pinnen.

Analyse van lichtsensor:

```
1  # Imports go at the top
2  from microbit import *
3
4  # Code in a 'while True:' loop repeats forever
5  while True:
6      tekst = 'De hoeveelheid licht = ' +  str(display.read_light_level())
7      print(tekst)
8      sleep(500)
```

Met het print commando stuur je data van de Micro:Bit naar de computer. Op de computer kan via het venster **Show Serial**



In de code zie je ook een type casting (convertie/omzetting) van een integer waarde naar een string :

str(integer)

Je ziet ook een concatenatie van twee strings.

21. Automatische verlichting

Opdracht : Zorg dat de LED matrix in drie helderheden wordt aangestuurd afhankelijk van het invallend licht. Analogie met de automatische verlichting bij een wagen die een tunnel binnen komt. Wanneer geen lichten? Wanneer kleine lichten? Wanneer grote lichten? Bepaal eerst aan de hand van uw testen de twee grenswaarden voor de lichtsensor.

```
1  # Imports go at the top
2  from microbit import *
3
4  # Code in a 'while True:' loop repeats forever
5  while True:
6      if (display.read_light_level() < 100):
7          display.show(Image('30903:'
8                             '06960:'
9                             '99999:'
10                            '06960:'
11                            '30903'))
12
13     elif (display.read_light_level() < 150):
14         display.show(Image('00300:'
15                            '03630:'
16                            '36963:'
17                            '03630:'
18                            '00300'))
19
20     else:
21         display.clear()
```

Uitbreiding: Je kan gerust meerdere niveaus gebruiken met meerdere helderheden.

22. Geluidsensor

Gelijkaardig kan je een analyse doen met de geluidsensor. Doe dit eerst door de waarde van het geluid te PRINTEN naar de Serial console. Bepaal hiermee welke waarden vertegenwoordigen stil/luid? Wat is stil? Wat is luid? Bepaal een grenswaarde en schrijf dan code om een alarm te maken wanneer er teveel lawaai is in de klas.

Meet het geluid in Python met het statement : ***microphone.sound_level()***

23. Temperatuursensor

Opdracht: Meet de temperatuur in de klas en display de temperatuur op de LEDS als getal of als figuur of met pixels. Print ook die waarde eens realtime naar de Serial Console.

Lees de temperatuur in Python met: ***temperature()***

24. Gyrosensor

Opnieuw gelijkaardig kan de gyrosensor worden geanalyseerd. Echter bezit de gyrosensor niet één enkele meetwaarde maar drie meetwaarden volgens de versnelling in de drie dimensies (X, Y en Z as).

accelerometer.get_x()

Bepaal zelf via printen naar de Serial console wat X, Y en Z is. De versnelling wordt uitgedrukt in milli-G-kracht (versnelling 1G = 9,81m/s²). Op die manier kan zwaartekracht berekend worden uit massa en versnelling F=m.a (vb m=80kg en a=9,81m/s² => F=78,48N).

Bepaal zelf wat en waar is in de x-richting de versnelling = 1000mG, waar en hoe 0, waar en hoe max positief, max negatief?

Idem voor de y-richting en de z-richting?

25. Magnetischesensor

Met de Micro:Bit kan ook een magnetisch veld worden gemeten. Opnieuw kan dit in de 3-dimensionele ruimte adhv de 3 assen (X, Y en Z). Hiermee kan een kompas worden gemaakt die volgens een as van de micro:bit (meestal X) het aardmagnetisme kan meten. Maar het kan ook magnetische velden van losse magneten meten en bepalen waar zich een NOORD of een ZUIDPOOL ligt.

Echter is het wel zo dat wanneer je deze sensor wenst te gebruiken de Micro:Bit een kalibratie procedure zal vragen die moet uitgevoerd worden op de Micro:Bit vooraleer deze kan worden gebruikt.

Experimenteer met deze sensor door gebruik te maken van het Python commando : ***compass.get_x()***

Stuur opnieuw waarden naar de Serial console na de kalibratie procedure gevolgd te hebben (zie instructies op de Micro:Bit zelf).

26. Data logging

Nu we wat ervaring hebben met de verschillende sensoren willen we veelal waarden van die sensoren gaan loggen (data wegschrijven op de Micro:Bit zelf en die bijhouden) gedurende een bepaalde tijd of een bepaalde omstandigheid om er dan nadien een analyse op uit te voeren eventueel met Excel.

We wensen dus bijvoorbeeld de lichtsterkte (zonlicht) te monitoren gedurende een volledige dag en dit door iedere minuut een lichtmeting uit te voeren.

Dit kan met de Micro:Bit in Python als volgt gebeuren.

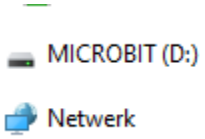
Logging_Light

```
1  # Imports go at the top
2  from microbit import *
3  import log
4
5  log.set_labels('light')
6
7  @run_every(s=1)
8  def log_data():
9      my_effect = audio.SoundEffect(duration=10)
10     audio.play(my_effect)
11     log.add({'light': display.read_light_level()})
12
13 while True:
14     sleep(100000)
```

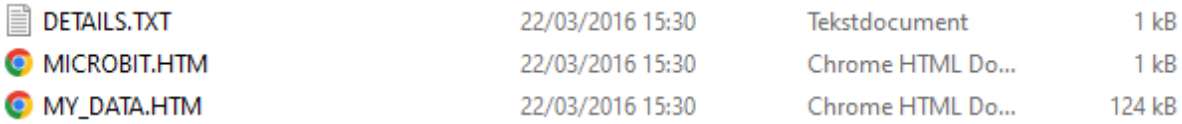
- Vooreerst zie je dat een extra import moet uitvoeren van **log**
- Daarna moet 1 keer **log.set_labels()** worden uitgevoerd die aangeeft wat de hoofding (gewoon tekst) van die reeks getallen zal inhouden. Stel dat je meerdere verschillende waarden wil loggen op het zelfde moment, dan kunnen die waarden gescheiden worden van elkaar door ze in aparte kolommen op te nemen en de kolom te voorzien van een hoofding (titel). Dit kan achteraf eenvoudig in Excel worden gelezen.
- We zien dat er in de code een oneindige lus (While True:) is opgenomen maar dat we daar niets in doen. We laten daar de Micro:Bit slapen gedurende 100000 seconden. We kunnen gerust de Micro:Bit iets anders laten doen in de oneindige lus (zie later)
- Er is een aparte functie (zie eerder) opgenomen in de code met de naam: **log_data()** zonder parameter. Deze methode of functie wordt aangeroepen (uitgevoerd) op basis van een timer: @run_every(h=1, min=20, s=30, ms=50).
- In die timer functie wordt een klein beepje afgespeeld zodat je de werking van de timer kan horen. Be werking kan natuurlijk dit wordne weggelaten.

Als je nu de gelogde data wenst te lezen dan kan je als volgt doen:

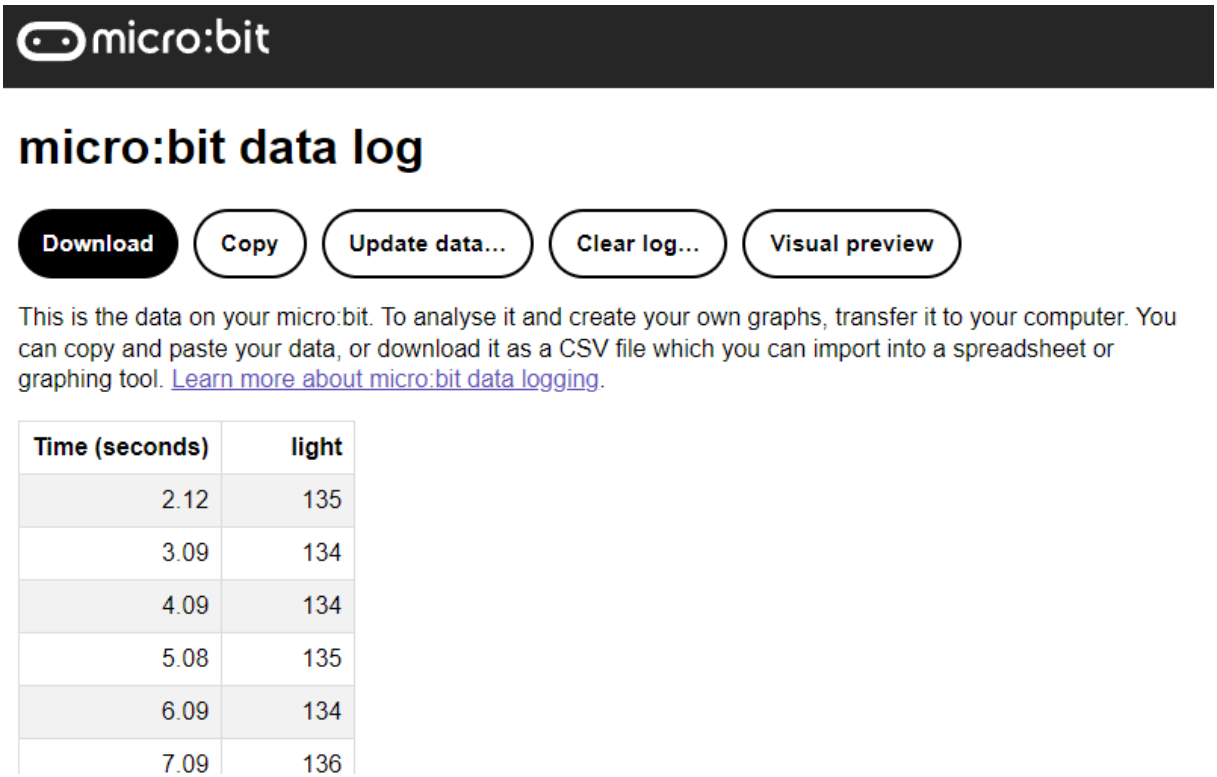
- Ontkoppel de Micro:Bit van de USB-kabel en wacht 10 seconden en koppel die terug aan (let wel de logging zal opnieuw starten, maar de oude data is bereikbaar , nieuwe logging data zal maar bereikbaar zijn na een volledige reset van de Micro:Bit => ontkoppel van USB / wacht 10sec / en koppel terug aan de computer)
- Ga naar de Verkenner van de computer en zoek naar de extra harddrive die de Micro:Bit heeft gemaakt tijdens het koppelen:



- Klik op die drive en zoek het bestand : MY_DATA.HTM



- Open dit bestand door dubbelklik:



Met copy kan je de lijst exporteren naar Excel (CSV).
Met Visual preview kan je een grafische weergave bekijken van de data.

27. Starten en stoppen van Data logging en clear de data

Soms is het handig dat …..

```
# Imports go at the top
from microbit import *
import log

log.set_labels('X','Y','Z', timestamp=log.MILLISECONDS)
logging = False

@run_every(ms=50)
def log_data():
    if logging == True:
        log.add({'X': accelerometer.get_x(),'Y': accelerometer.get_y(),'Z': accelerometer.get_z()})

# Code in a 'while True:' loop repeats forever
while True:
    if button_a.was_pressed():
        logging = not logging
        if logging == True:
            display.set_pixel(2,2,9)
        else:
            display.clear()
    if button_b.was_pressed():
        log.delete()
        display.show(Image.SQUARE_SMALL)
        sleep(500)
        display.clear()
        log.set_labels('X','Y','Z', timestamp=log.MILLISECONDS)
```