Agnetencollege
Peer

STEM-project

T³ VLAANDEREN

Programming an operation game with the TI-Nspire and BBC micro:bit

Teacher's bundle

Ann-Kathrin Coenen
& Natalie Dirckx

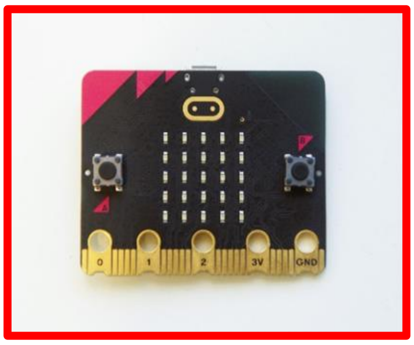


Table of contents

T^3-Flanders en T^3-Netherlands

Ann-Kathrin Coenen and Natalie Dirckx are science teachers at Agnetencollege Peer. They also belong to the teacher network of T^3 Flanders that works closely with the Netherlands. T^3 stands for Teachers Teaching with Technology. The goal of this organization is to promote the professionalization of teachers in the field of ICT and technology in education using technology from Texas Instruments.

Figure 1:
www.t3vlaanderen.be

Introduction

An operation game is a familiar skill game from childhood. A patient lies on the operating table and has swallowed various (foreign) objects. In this case, your dog has eaten foreign objects. You are the veterinarian and will operate on your dog. With tweezers you gently remove the objects. But be careful! Only touch the object, otherwise you lose points

This project can be divided into a designing part and a programming part. Building an operation game and programming the game with Python on the TI-Nspire fits ideally within STEM, biology, physics or engineering classes.

A link can be made to the teaching of electrical systems within physics. The students learn to break the game into small parts to write the code. And the game can be used to apply the concepts of voltage, amperage and current sense to this project. In addition, the game also tests the players eye-hand coordination and motor skills.

Introducing the BBC micro:bit

The BBC micro:bit is a popular pocket-sized computer of microcontroller. It is an interface for collaboration between software and hardware.

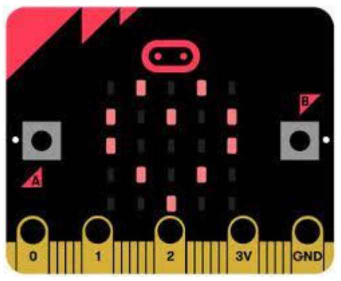


Figure 2: BBC micro:bit

The micro:bit has a 5X5 LED light display, push buttons A and B, touch input buttons, built-in microphone and speaker. In addition,this microcontroller itself contains many sensors including for temperature, light, motion and a compass. Finally, interaction with other devices or the Internet is also possible through a Bluetooth connection.

The micro:bit performs actions after programming instructions. These instructions are written in the Python programming language on the TI-84 Plus CE-T Python Edition or TI-Nspire CX II (Figure 4). In this bundle, the Nspire was chosen to code this project.

Figure 3: python programming language

Python is an open source programming language that is simple and straightforward yet widely applicable in technologies. Programming in python is recommended for beginners, which makes this programming language very suitable for students.

Figure 4: TI-Nspire CX II

Programming with the TI-*Nspire* CX II

The TI-Nspire CX II is a graphing calculator with hands-on learning tools for both math and science classes. It can be used via software as well as handheld.

The python module will be used for this project. The code can be programmed on either the laptop or the handheld. You write new python code by creating a new document in the home menu and then selecting "Add Python" (Figure 5). Using the menu button, it is possible to add partially prescribed pieces of code. Both the micro:bit and the handheld must be fitted with a module before it is possible to program for the micro:bit.

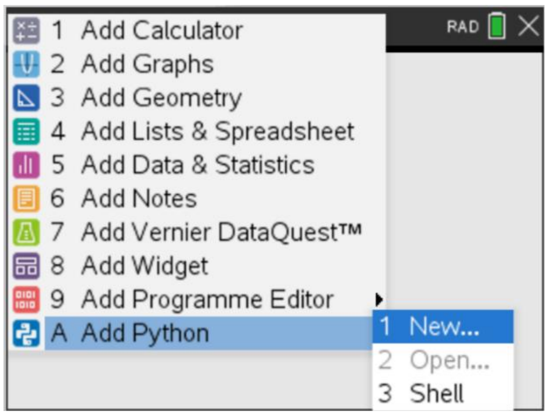
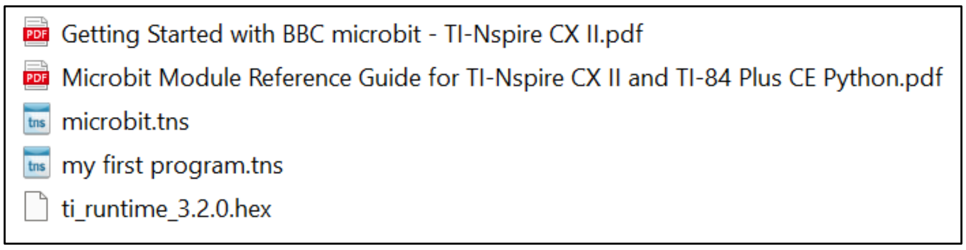


Figure 5 adding a new Python page

From the TI education website you can download the necessary files (Figure 6) as a zip. You will find all necessary files in this folder including a roadmap for installation. After installing microbit.tns, it is possible to use certain functions of the micro:bit in the Python page of the handheld.



Getting Started with BBC microbit - TI-Nspire CX II.pdf
Microbit Module Reference Guide for TI-Nspire CX II and TI-84 Plus CE Python.pdf
microbit.tns
my first program.tns
ti_runtime_3.2.0.hex

Figure 6: microbit.tns contains the module for the handheld, ti_runtime.hex is the module for the micro:bit

A hex file must be installed on the micro:bit. When the code is successfully placed on the micro:bit, the Texas Instruments logo will appear. The micro:bit can be connected to the TI-Nspire via the USB mini to micro cable.

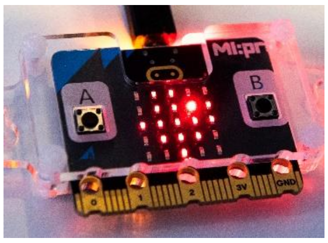
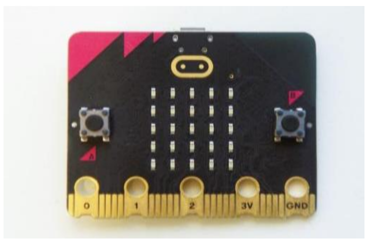
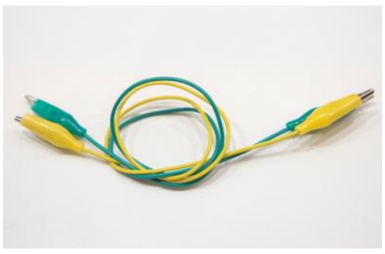


Figure 7: Texas Instruments logo on display of the micro:bit

The supplies

One operation game requires the following supplies:

- TI-Nspire CX II
- Micro:bit v2
- Mini-USB to micro-USB cable
- Aluminum foil
- Aluminium/Tinplate metal bottle caps
- Cardboard (approximately 2mm thickness) with dimensions 15 cm X 15 cm
- Tweezers
- 2 alligator clips
- Small plastic (non-conducting) objects (they have to fit into the bottle caps, for example charms for bracelets)
- A picture of the patient
- Wooden box of minimum size 15 cm X 15 cm ‘Art Panel’ (www.hobbyshop-online.nl)

		
BBC micro:bit	Alligator clips	Small plastic objects
Wooden box (Art Panel)	Aluminium/Tinplate metal bottle caps	Cardboard and aluminum foil

Method – build the game

Building the operation game is done as follows:

1. Take a picture of a patiant and determine where the foreign objects will be placed. Draw a circle on these spots in the same size as the aluminum bottle caps and cut them out.
2. Take the cardboard. If not done already, cut out a square of 15 x 15 cm. Place the picture on top of the cardboard and draw the same circles on the board. Take the picture away and cut the circles out (Figure 8).
3. Cover the cardboard completely all over the front and back with aluminum foil and glue in place. Then cut the circles from the foil as well (Figure 9).
4. Glue the picture over the aluminum foil on the cardboard so that the holes match.
5. Insert a bottle cap into each hole (Figure 10).
6. Place a small object in each bottle cap as a foreign object (Figure 12).
7. Attach an alligator clip to the tweezers and connect it to pin 0 of the BBC micro:bit. This clip may well come loose from the tweezers, so it is best to attach a little extra with tape.
8. Attach another alligator clip to the cardboard with aluminum foil and connect it to the ground pin (GND) of the BBC micro:bit (Figure 11).
9. Take the wooden box and turn it with the concave side up. Place the operation game on top of it.

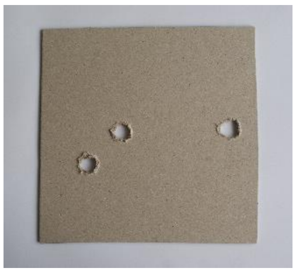


Figure 8: cardboard with cut out circles

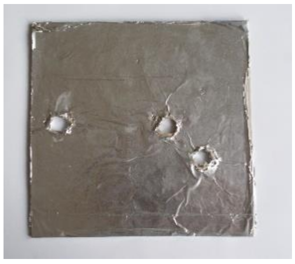


Figure 9: aluminium covered cardboard plate

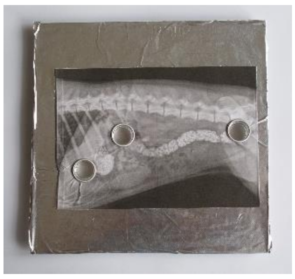


Figure 10: complete game board

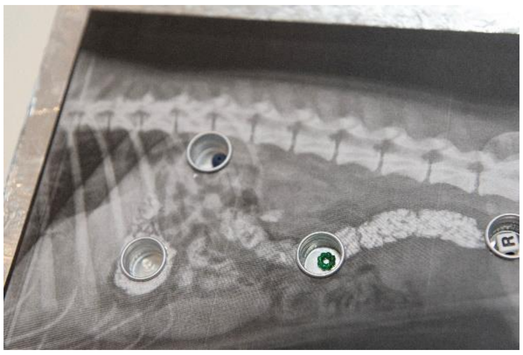


Figure 12: placement of the foreign objects

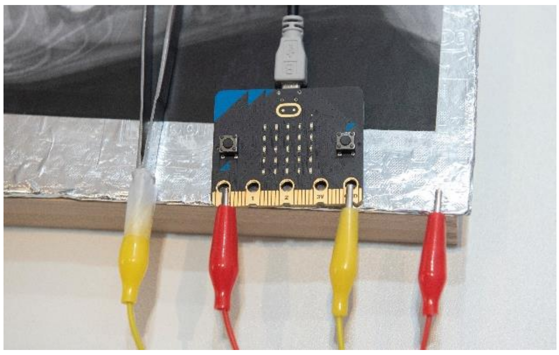


Figure 11: the alligator clip of the game is attached to pin 0 and the clip of the tweezers to GND.

Connecting TI-Nspire to micro:bit

When the operation game is finished, the micro:bit can be connected with the USB cable mini to micro according to Figure 13. Figure 14 shows a complete setup.

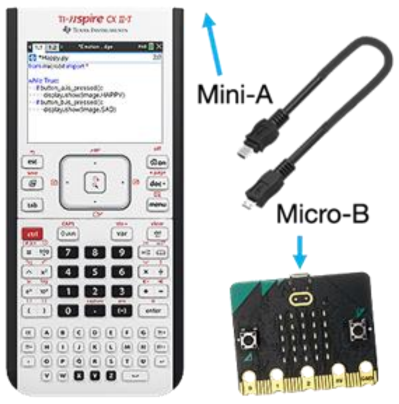


Figure 13: mini to micro usb-cable

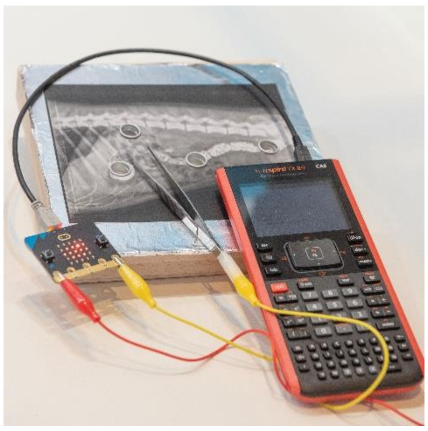


Figure 14: a finished operation game

Method – programming the BBC micro:bit

Game plan

To guide students who have never programmed before in the learning process, we break the game into small steps.

The game should receive a signal to start. From the start, it must be defined how many points a player gets from the start. In this case there are 3 points at the start. When touching the edge of the board or bottle cap, one point should be deducted from the remaining points.

When you run out of points, you have lost.

The micro:bit has a display on which you can show figures and the micro:bit can also play sounds. You can combine this with touching the edge or bottle cap.

We create an example diagram (Figure 15) to facilitate programming.

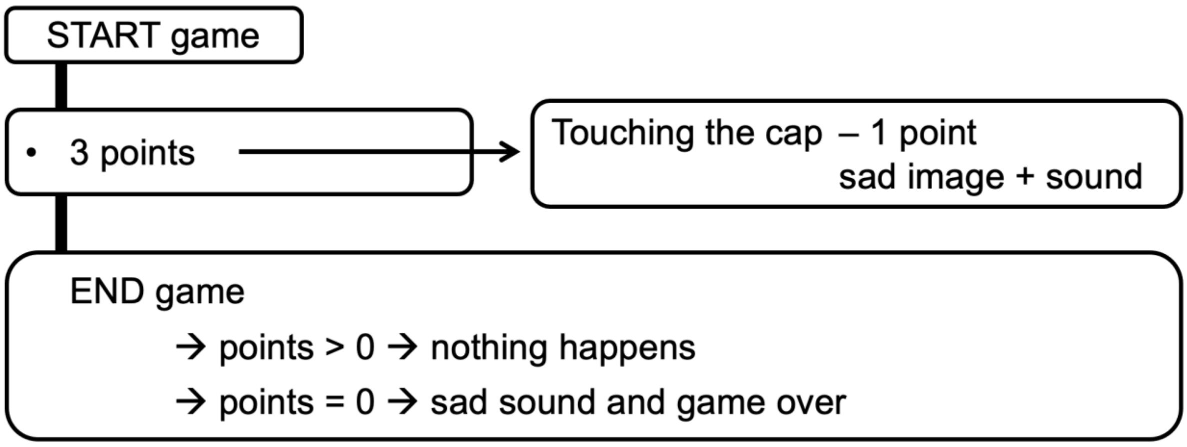

START game
• 3 points
Touching the cap – 1 point
sad image + sound
END game
→ points > 0 → nothing happens
→ points = 0 → sad sound and game over

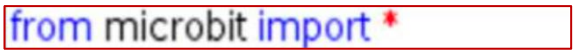
Figure 15: example of a possible game plan

Code explanation

Starting up a Python environment

Go to the home screen and choose 'new'. Here you can choose 'Add Python'. After that, choose 'New'. Choose a name for your program without using a space. Via the menu button you can add the necessary functions. It is also always possible to type in the code yourself.

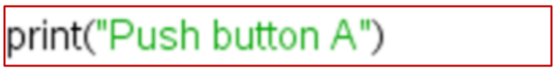
Starting the game



```
from microbit import *
```

This makes it possible to use the library code for micro:bit.

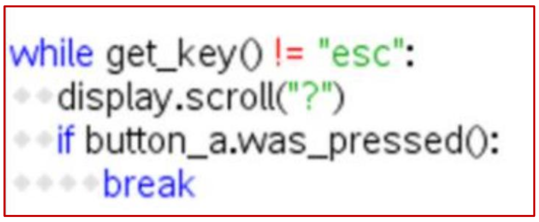
At menu choose 'More modules', there you can choose 'BBC micro:bit'. The first option is 'from microbit import'.*



```
print("Push button A")
```

On the handheld's screen, we want "Push button A" to appear.

Under menu, choose "Built-in functions" and go to "I/O. The first option is print(). You need to type the text "Push button A" yourself with the letters at the bottom.



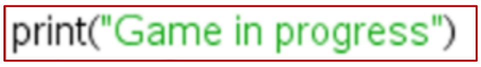
```
while get_key() != "esc":
  display.scroll("?")
  if button_a.was_pressed():
    break
```

We want the code to keep running until "esc" is pressed. For this we use a while loop. On the screen of the micro:bit we want to see a "?" scroll. And when button A is pressed on the micro:bit it stops and the game starts. The code continues to the game. This pictured piece of code can also be left out, but then the game starts immediately when you let the handheld run the program. So now you build in an additional start signal (button A) on the microbit.

In the microbit module, you choose 'Commands'. And there you will find as the second option 'while get_key()="esc"'. After this you see that the code is indented. So everything at this tab belongs to this while loop.

In the microbit module choose 'Commands' and then 'if <>:break. The code appears on the page and immediately in the microbit module under 'Display' choose 'scroll(text or number)'. Now you see that this code is completed between the <>. You have to type the "?" between the ().

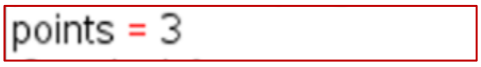
Variables in the game



```
print("Game in progress")
```

When button A is pressed on the micro:bit, we want the game to start. On the handheld "Game in progress" is then displayed.

Under menu, choose "Built-in functions" and go to "I/O. The first option is print(). You need to type the text "Game in progress" yourself with the letters at the bottom.

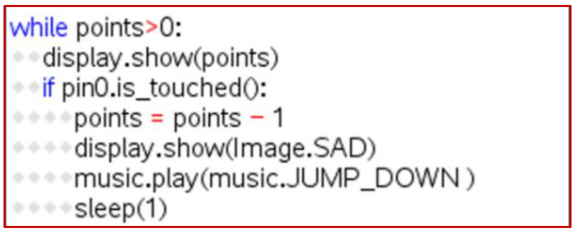


```
points = 3
```

First, we still need to define that the game starts with 3 points.

Type yourself 'points = 3'. This creates a variable that we can call later in the code.

Defining the game



```
while points>0:
  display.show(points)
  if pin0.is_touched():
    points = points - 1
    display.show(Image.SAD)
    music.play(music.JUMP_DOWN )
    sleep(1)
```

We create a new while loop that keeps running as long as the points are not zero.

On the display of the micro:bit, we want to see the points.

In the if function we program that the points should decrease by 1 value each time pin 0 (circuit closed) is touched. Here we link a picture and a music on the micro:bit.

After this, 1 second of rest (sleep) is built in before the code continues.

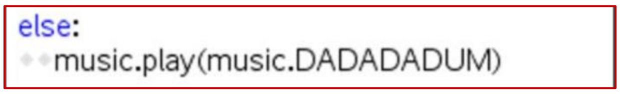
At menu, choose 'Built-in functions' and go to 'built-in functions'. Under the second option 'control' choose 'while'. When you press the var key, a selection list appears and choose 'points' and then type '>0'. To show the points on the display of the micro:bit, again go to 'Display' in the microbit module.

'If' is found in the built-in functions 'Control'. Touching pin 0 must be placed after this if. Go to the microbit module and under 'I/O Pins' select the 'Digital'. Below you will see 'pin.is_touched()'. You then choose pin0.

The points are rebuilt with the var key. In the microbit module you can choose an 'Image' at 'Display' and a tune at 'Music'.

After this you build in a 'sleep(1)'. You can find this in the microbit module under 'Commands'.

End of the game

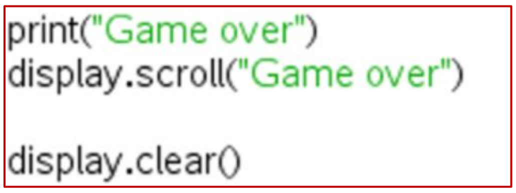


```
else:
  music.play(music.DADADADUM)
```

If the points are equal to zero a sad music will play.

'If...else' is found in the built-in functions under 'Control'. At the var key you find the points back.

In the microbit module, under "Display" and "Music," you will find the pictures and the tunes.



```
print("Game over")
display.scroll("Game over")

display.clear()
```

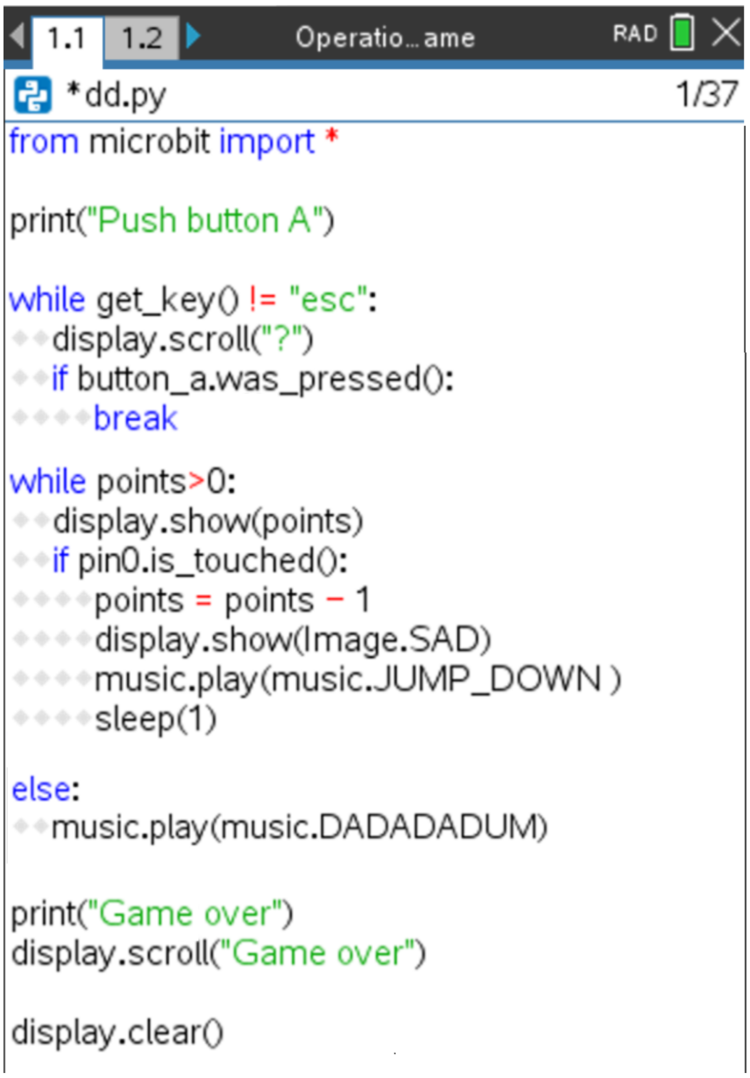
The handheld displays the message "Game over" and the micro:bit displays "Game over."

The handheld displays the message "Game over" and the micro:bit displays "Game over."

From menu, choose 'Built-in functions' and go to 'I/O'. Choose 'print()' and type in the text itself. The code for the display can be found in the microbit module at Display.

If you want to type for yourself a comment to the code that is not executed you can note it after a # character.

In Figure 16 *you can see the code in its entirety*



```
from microbit import *

print("Push button A")

while get_key() != "esc":
  display.scroll("?")
  if button_a.was_pressed():
    break

while points>0:
  display.show(points)
  if pin0.is_touched():
    points = points − 1
    display.show(Image.SAD)
    music.play(music.JUMP_DOWN )
    sleep(1)

else:
  music.play(music.DADADADUM)

print("Game over")
display.scroll("Game over")

display.clear()
```

Figure 16: full code

Once the code is written, it can be run. You do this by choosing 'execute' in 'menu'. This can also be done by pressing 'Ctrl'+'R'. A Shell will open in a new page. With 'Ctrl' + left or right arrow you can switch between pages. So if you want to change something in your code, you will have to make your change on the previous page and run the program again in the shell.