



Agnetencollege  
Peer

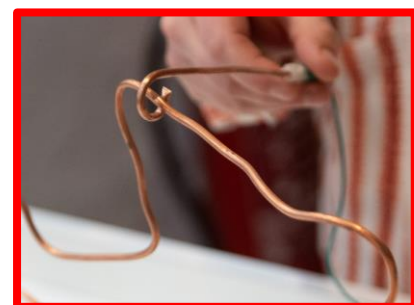
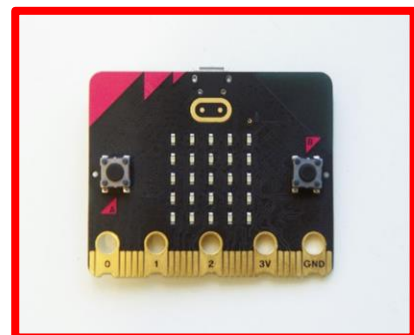
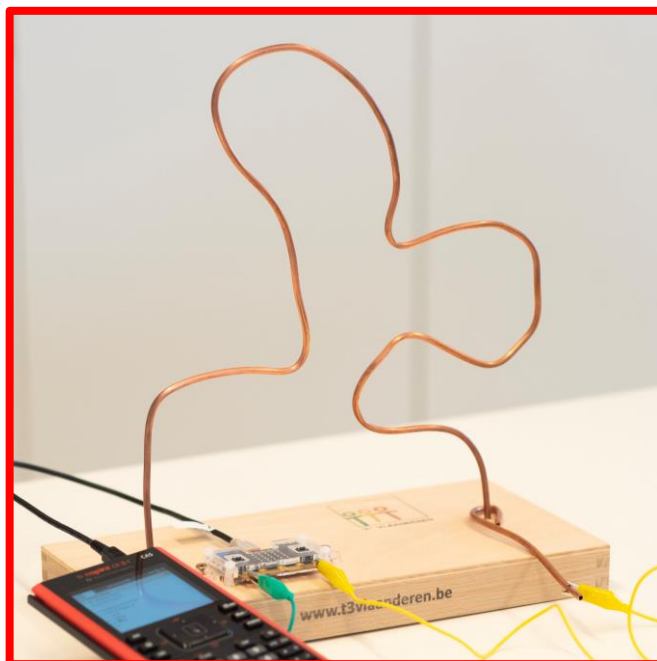
STEM-project



# Programming a Nerve spiral game with the TI-Nspire and BBC micro:bit

Teacher's bundle

*Ann-Kathrin Coenen  
& Natalie Dirckx*

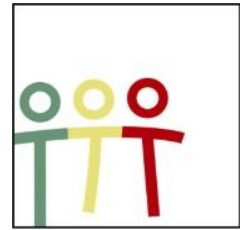


# Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
<b>T<sup>3</sup>-Flandres and T<sup>3</sup>-Netherlands</b> .....	<b>3</b>
<b>Introduction</b> .....	<b>3</b>
<b>Introducing the BBC micro:bit</b> .....	<b>4</b>
<b>Programming with the TI-Nspire CX II</b> .....	<b>5</b>
<b>The supplies</b> .....	<b>6</b>
<b>Method – build the game</b> .....	<b>7</b>
<i>Addition with an extra pin</i> .....	8
<i>Connecting TI-Nspire to micro:bit</i> .....	8
<b>Method - programming the BBC micro:bit</b> .....	<b>9</b>
<i>Game Schedule</i> .....	9
<i>Code explanation</i> .....	9
<i>Starting up Python environment</i> .....	9
Start spel.....	10
Variables game .....	11
Defining game.....	11
End of game.....	12
<b>Extension additional contact point</b> .....	<b>14</b>

## T<sup>3</sup>-Flandres and T<sup>3</sup>-Netherlands

Ann-Kathrin Coenen and Natalie Dirckx are science teachers at Agnetencollege Peer. They also belong to the teacher network of T<sup>3</sup> Flanders that works closely with the Netherlands. T<sup>3</sup> 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.



**T<sup>3</sup> VLAANDEREN**

Figure 1:

[www.t3vlaanderen.be](http://www.t3vlaanderen.be)

## Introduction

A nerve spiral or copper spiral is a familiar agility game from childhood. A stick with a ring is moved along a copper wire from one end to the other without touching the copper wire. If the wire is touched anyway, points are deducted from your total score.

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

A link can be made to the teaching of electrical systems within physics. The students learn to break the game into small parts to design a code. And the game can be used to apply the concepts of voltage, amperage and current sense to this project.

## Introducing the BBC micro:bit

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

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.

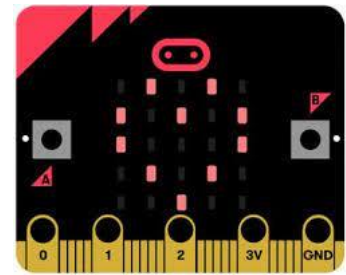


Figure 2: BBC micro:bit



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.

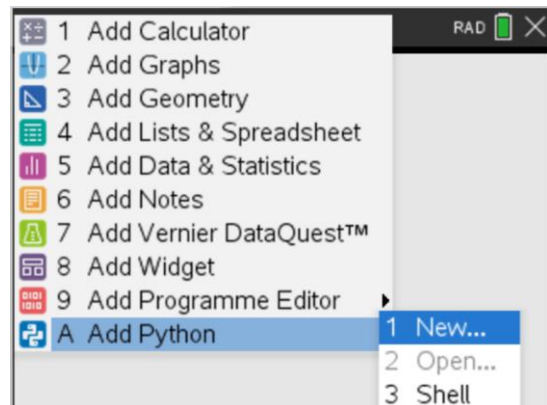


Figure 5: adding a new Python page

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.

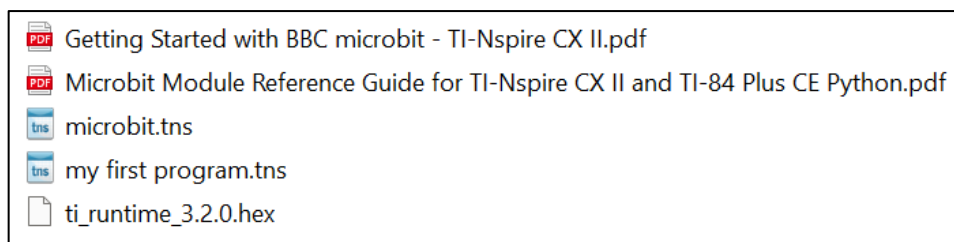


Figure 6: microbit.tns contains the module for the handheld, ti\_runtime.hex is the module for the micro:bit

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.

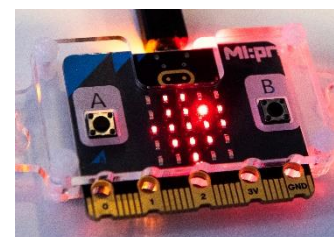
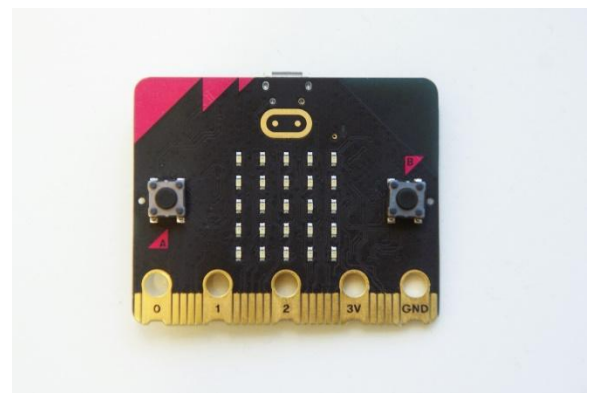
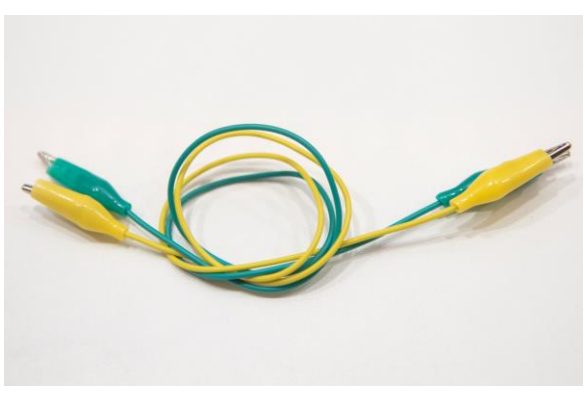




Figure 7: Texas Instruments logo on display of the micro:bit

## The supplies

One nerve spiral requires the following supplies:

- TI-Nspire CX II
- Micro:bit v2
- Mini-USB to micro-USB cable
- Flexible copper tube  $\varnothing 4\text{mm}$  70cm
- Wooden box of minimum size 15 cm X 15 cm 'Art Panel' ([www.hobbyshop-online.nl](http://www.hobbyshop-online.nl))
- 2 (3) Alligator clips

	
BBC micro:bit	Alligator clips
	
Wooden box (Art Panel)	Flexible Copper tube $\varnothing 4\text{mm}$

## Method – build the game

Building the nerve spiral is done as follows:

1. Take a wooden board and drill a 4-mm hole on both ends (Figure 8).
2. Cut 20 cm off the copper wire. Take the piece of copper wire for the game stick and fold it into a ring at one end. This will be the game stick that the player uses to move around the wire.
3. Bend the remaining 50 cm into a fun and exciting shape for the spiral.
4. Insert both ends of the coil through the holes in the board (Figure 9). Make sure there is about a 1 cm piece of copper wire sticking out at the bottom to attach your alligator clips (Figure 11).
5. Attach an alligator clip to one of the ends of the coil and connect it to pin 0 of the BBC micro:bit (Figure 12).
6. Attach an alligator clip to the game stick and connect it to the ground pin (GND) of the BBC micro:bit (Figure 12). This clip may well come loose from the game stick, so it is best to attach a little extra with tape.

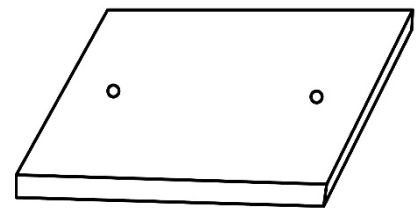


Figure 8: wooden box with holes

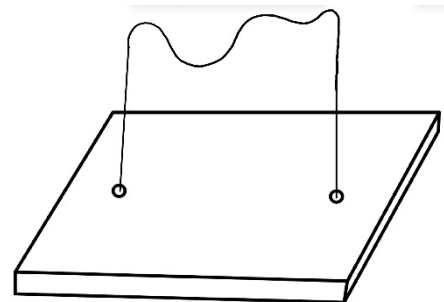


Figure 9: fixing the spiral to the box

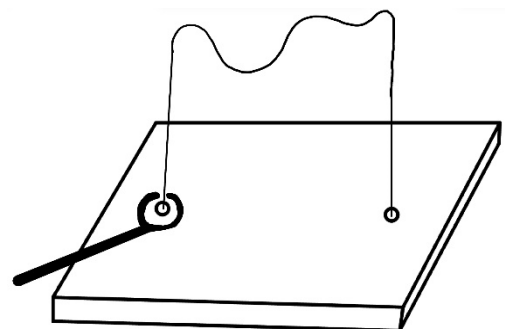


Figure 10: nerve spiral with game stick



Figure 11: crocodile clip can be attached at the bottom to the copper tube inserted through.

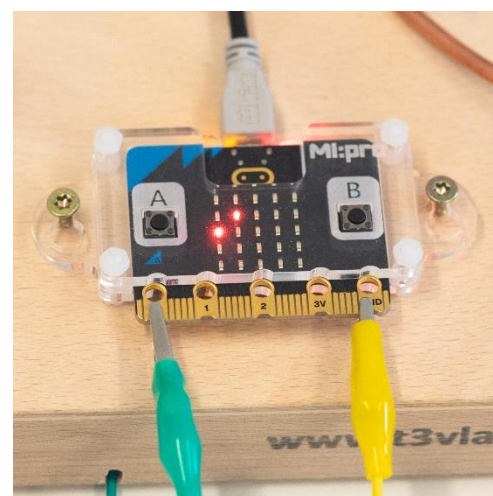


Figure 12: The alligator clip of the coil is attached to pin 0 and the terminal of the game stick is attached to GND.

## Extension with an additional contact point

The game can be expanded with an additional pin. If this contact point is touched with the game stick after traveling down the spiral then time stops running. For this contact point you need to drill an extra hole of 4 mm at 0.5 cm distance from the spiral. Here you put a small piece of copper tube (Figure 13) through so that you can also attach an alligator clip at the bottom of this (Figure 14). You attach this alligator clip on the other side to pin 1 of the micro:bit (Figure 15).



Figure 13: extra pin top

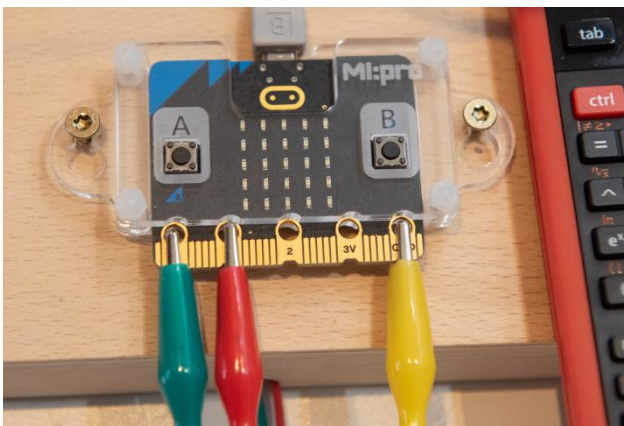


Figure 15: the alligator clip of the additional contact point are connected on the micro:bit to pin 1.

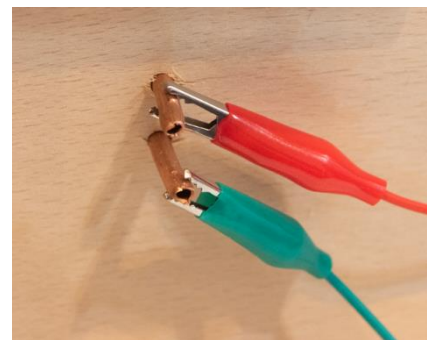


Figure 14: extra pin bottom

## Connecting TI-Nspire to micro:bit

When the spiral is finished, the micro:bit can be connected with the USB cable mini to micro according to Figure 16. Figure 17 shows a complete setup.



Figure 16: mini to micro USB-cable

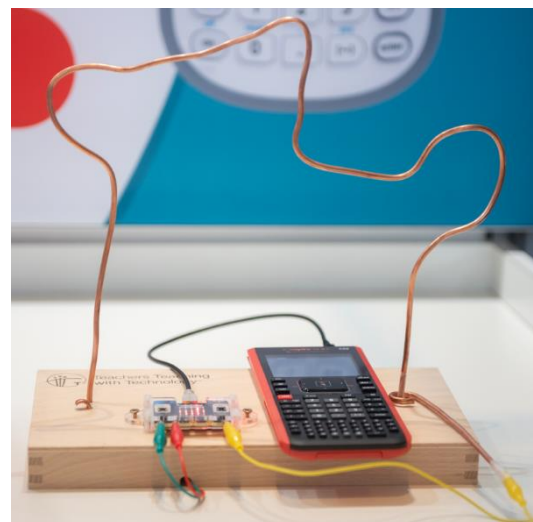


Figure 17: a finished nerve spiral



## 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, time must also begin to run. Furthermore, it must be defined how many points a player gets from the start. When touching the spiral, one point should be deducted from the remaining points.

When you run out of points, you have lost. After the time is up, if you still have points, you have won.

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 winning or touching the spiral.

We create an example diagram (Figure 18) to facilitate programming.

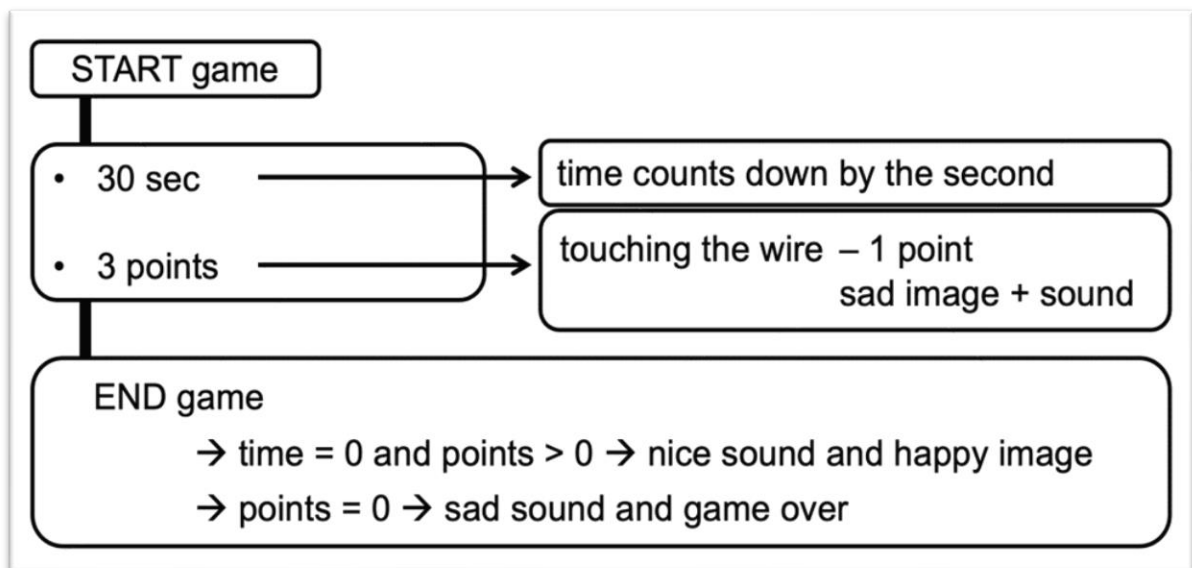


Figure 18: example of a possible game plan

### 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\*'.*

```
from time import *
```

This makes it possible to use time in the written program.

*At menu, choose 'More modules', there you can choose 'Time'. The first option is 'from time 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")
points = 3
t0 = clock()
time = 0
```

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

First, we still need to define that the game starts with 3 points and the clock starts running from the time button A was pressed.

*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.*

*Type yourself 'points = 3', 't0 = clock' and 'time = 0'. Clock() can also be found at menu 'More modules', there you can select 'Time' and you will also see clock(). This creates variables that we can call later in the code.*

## Defining the game

```
while time <30 and points>0:
  time = clock()-t0
  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 time is less than 30 seconds and as long as the points are not zero.

We define time as being the current time minus the start time. When it exceeds 30 seconds, the game stops. 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 'time'. Type '<30' as well as the 'and'. At the var key now choose 'points' and then type '>0'.*

You can build the 'time =...' line analogously with the var key. 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

```
if points > 0:  
  display.show(Image.HAPPY)  
  music.play(music.ENTERTAINER)  
else:  
  music.play(music.DADADADUM)
```

When the points are still greater than zero after the time expires, a happy face and a nice sound appear on the micro:bit.

If the points do equal 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 19, you can see the code in its entirety.

```

1.1 1.2 *Nerve spiral RAD
*dd.py 1/35
from microbit import *
from time import *

print("Push button A")

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

print("Game in progress")
points = 3
t0 = clock()
time = 0

while time <30 and points>0:
    time = clock()-t0
    display.show(points)
    if pin0.is_touched():
        points = points - 1
        display.show(Image.SAD)
        music.play(music.JUMP_DOWN)
        sleep(1)

if points > 0:
    display.show(Image.HAPPY)
    music.play(music.ENTERTAINER)
else:
    music.play(music.DADADADUM)

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

display.clear()

```

Figure 19: 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.

## Extension additional contact point

In the code described above, the game will end after 30 seconds. If you complete the spiral faster, the winner music will play only after the full time has passed.

By adding an extra contact point, you can make the game end faster. When you finish the spiral and you touch the extra contact point you touch pin 1.

In the code you see that 'if pin1.is\_touched():' is now in the while loop. This is possible now because you don't have to go through the whole loop to then look at the number of points. If pin 1 is touched and your points are > 0 then you are the winner. You can indent the code by space or tab.

Furthermore, students can still adjust the number of points and time duration to add variation to the difficulty of their built spiral.

Have fun!

```
while time <30 and points>0:
  time = clock()-t0
  display.show(points)
  if pin0.is_touched():
    points = points - 1
    display.show(Image.SAD)
    music.play(music.JUMP_DOWN)
    sleep(1)
  if pin1.is_touched():
    display.show(Image.HAPPY)
    music.play(music.ENTERTAINER)
else:
  music.play(music.DADADADUM)
```

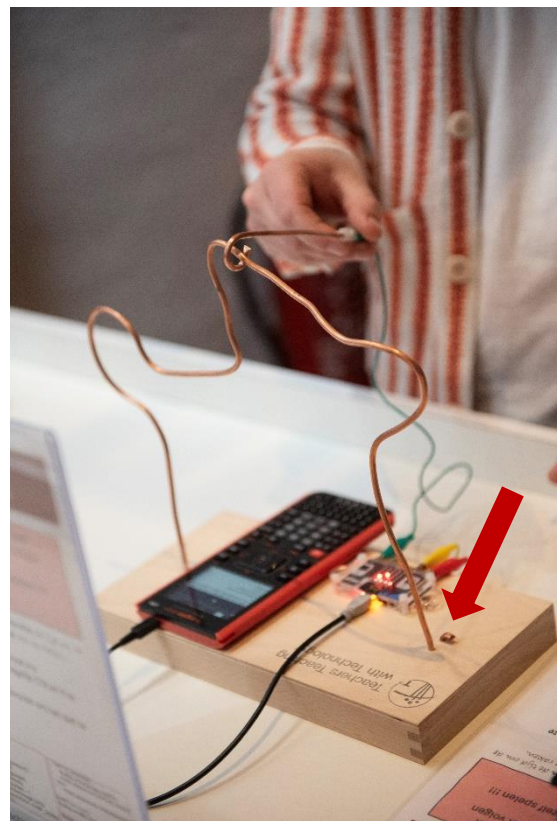


Figure 20: game with additional contact point