Co-funded by the
Erasmus+ Programme
of the European Union

LBC LEARNING BY COMPETING

ArTeC® Robo 2.0 Simple Kit

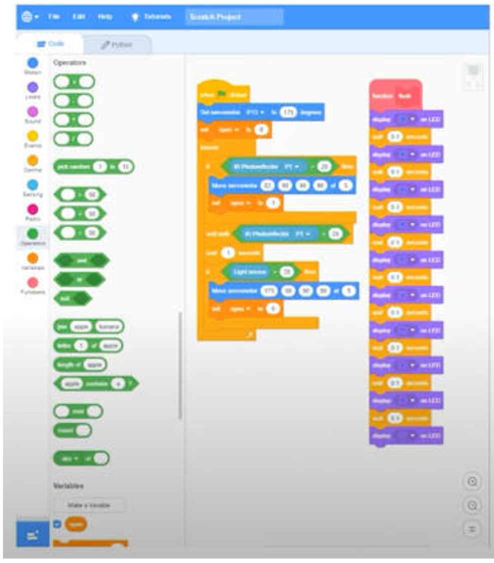
ArTeC Robo 2.0
Simple Kit
Lean to program in Python!
WARNING:
CHOKING HAZARD – Small parts.
Not for children under 3 yrs.
Artec Co., Ltd.

What's the ArtecRobo 2.0 Simple Set Python Edition?

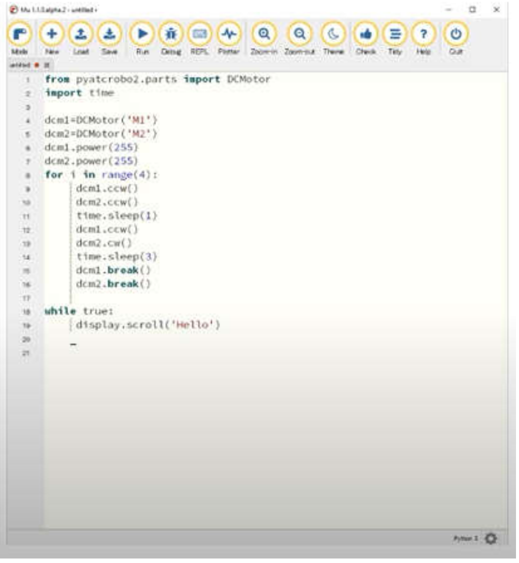
The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

GOAL :

Graphical



Text-based

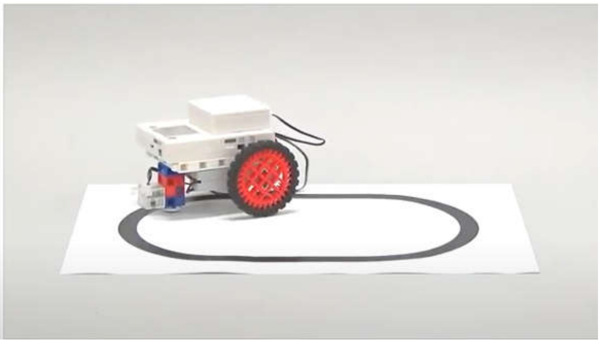


```
from pyatcrobo2.parts import DCMotor
import time

dcm1=DCMotor('M1')
dcm2=DCMotor('M2')
dcm1.power(255)
dcm2.power(255)
for i in range(4):
    dcm1.ccw()
    dcm2.ccw()
    time.sleep(1)
    dcm1.ccw()
    dcm2.cw()
    time.sleep(3)
    dcm1.break()
    dcm2.break()

while true:
    display.scroll('Hello')
```

Examples :

Traffic Light

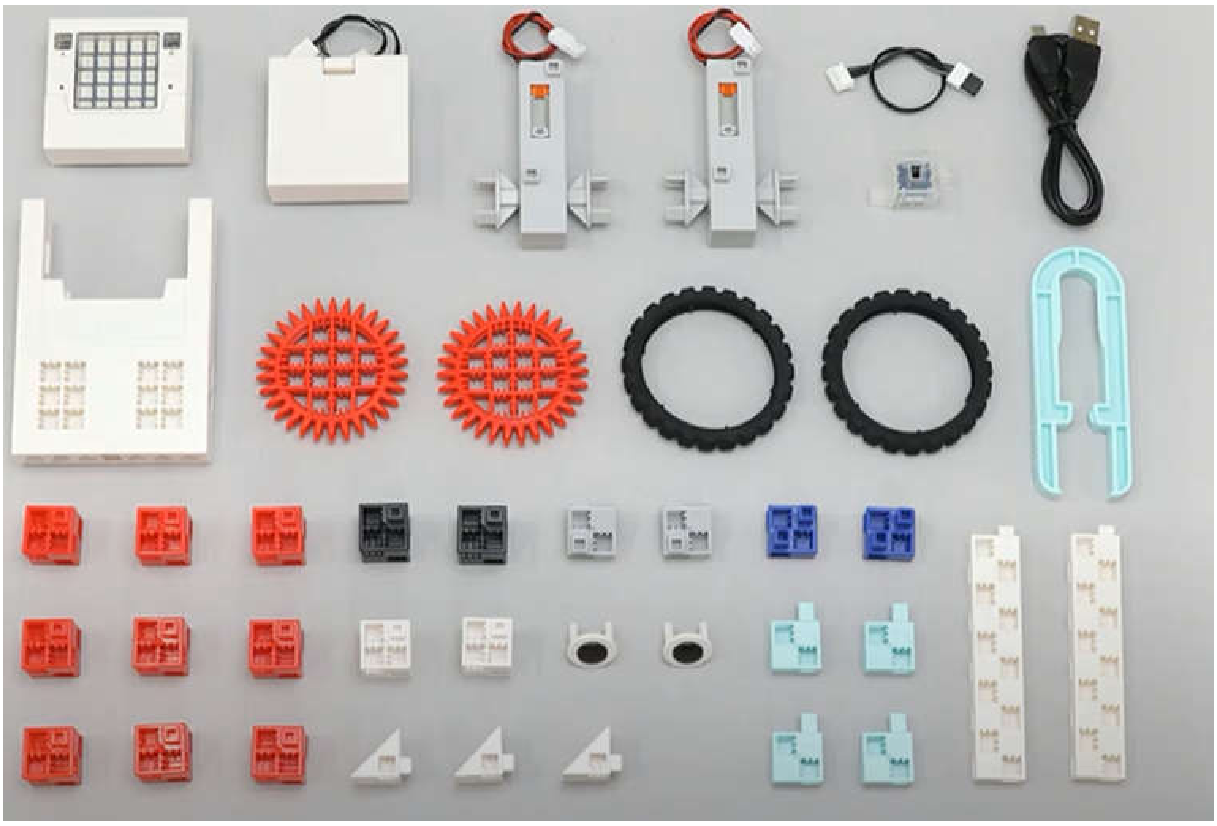
Line Tracing Car



ArTeC Robo 2.0

Simple Set Python Edition

- Contents -



- Box Content

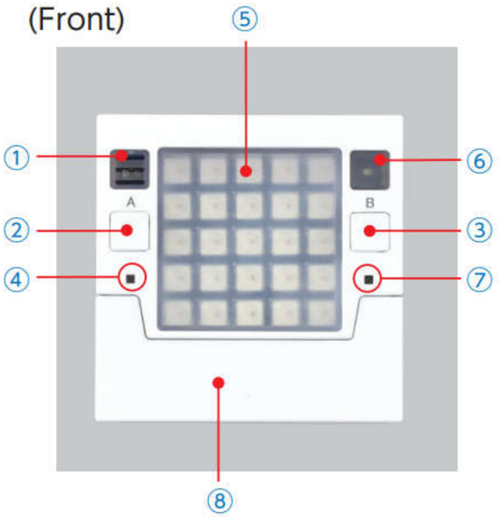
1 Studuino:bit Core Unit
1 Robot Expansion Unit
1 Battery box
2 DC motor
1 IR Photoreflector
1 USB Cable microB
1 Sensor connecting cable (3-wire, 15cm)
13 Basic cube (red, gray, white)
3 Triangle A (white)
2 Half A (light gray)
6 Half B (blue)
4 Half C (light aqua)
2 Beams
2 Gears (L)
2 Tires
2 Disks
1 Block remover

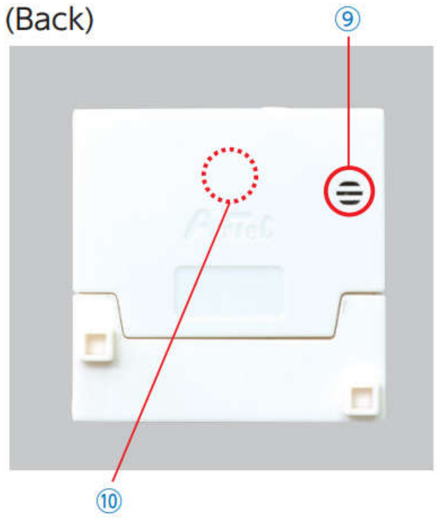
Co-funded by the
Erasmus+ Programme
of the European Union

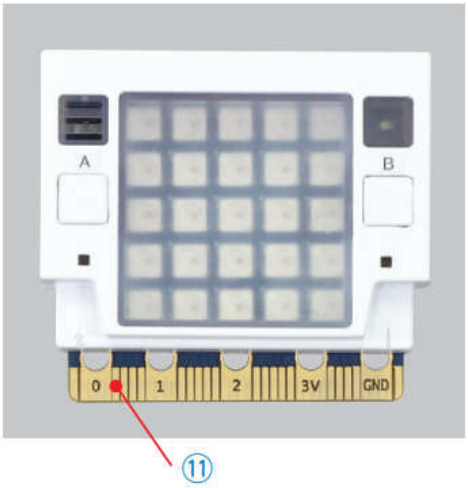
LEARNING BY COMPETING

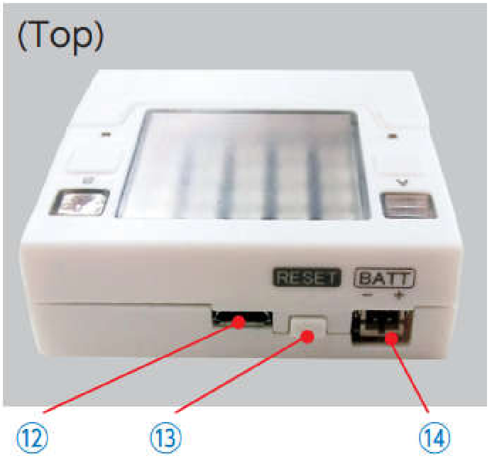
1 Parts of the Core Unit

The Core Unit is the part of the robot that runs programs.





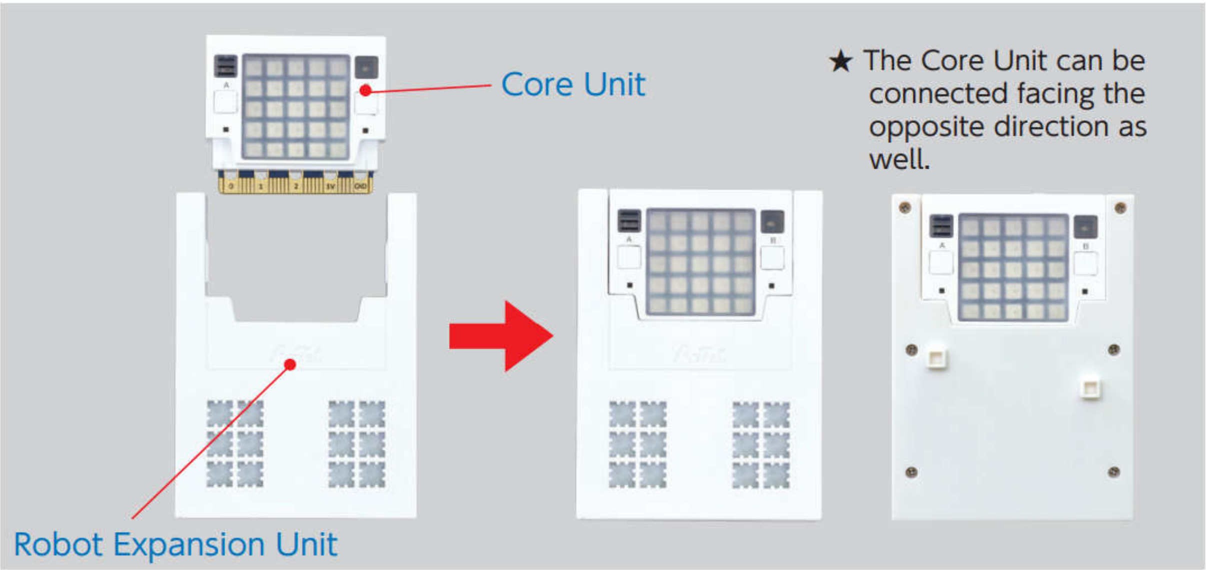


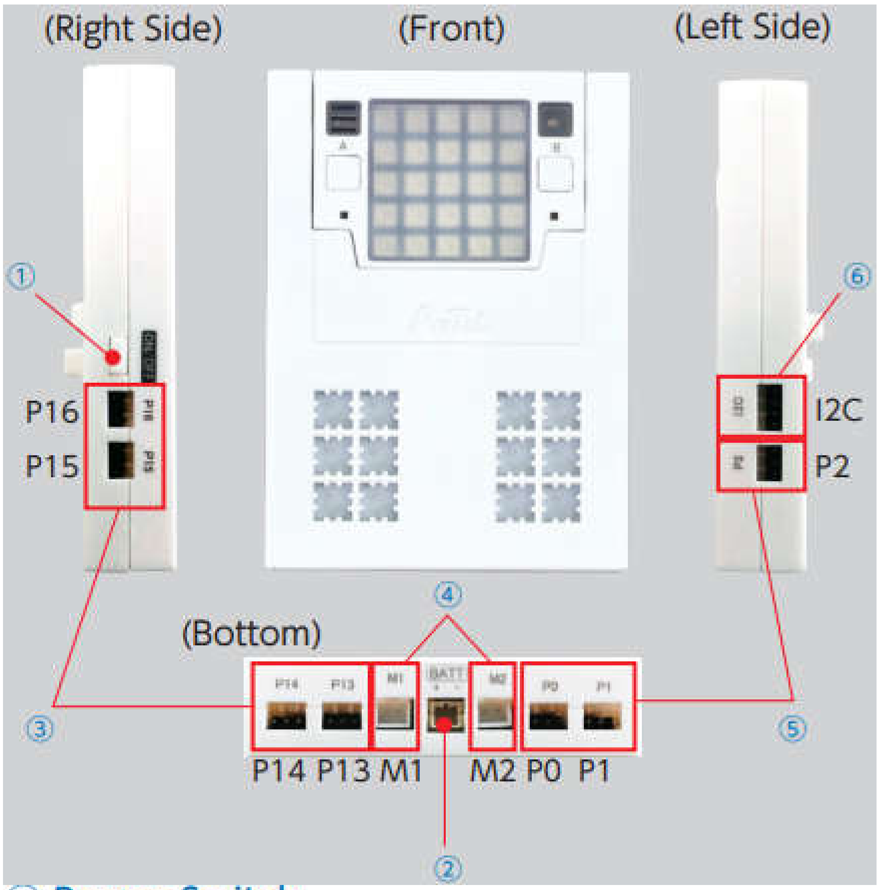


① Temperature Sensor
②/③ A/B Buttons
④ Power Light (Green)
⑤ 5 x 5 Full Color LED Matrix
⑥ Light Sensor
⑦ Connection Light (Blue)
⑧ Artec Block Connecting Cover
⑨ Buzzer
⑩ 9-Axis Sensor
⑪ Edge Connector
⑫ USB Port (microB)
⑬ Reset Button
⑭ Power Connector

Co-funded by the
Erasmus+ Programme
of the European Union
LEARNING BY COMPETING

2 The Robot Expansion Unit


Core Unit
★ The Core Unit can be connected facing the opposite direction as well.
Robot Expansion Unit



① Power Switch
② Power Connector
③ Digital Output Terminals (P13/P14/P15/P16)
④ DC Motor Terminals (M1/M2)
⑤ Analog Input Terminals (P0/P1/P2)
⑥ I2C Communication Terminal

Co-funded by the Erasmus+ Programme of the European Union
LEARNING BY COMPETING

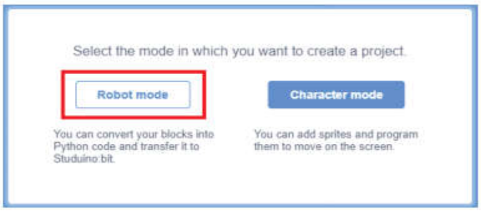
3 The Studuino:bit Software

Starting Up the Software

Follow these steps to start up the software and open the programming interface.

1. Click the **Studuino:bit** icon to start up the software.

2. Select **Robot mode** after the software starts up to enter the programming screen.



Select the mode in which you want to create a project.

Robot mode

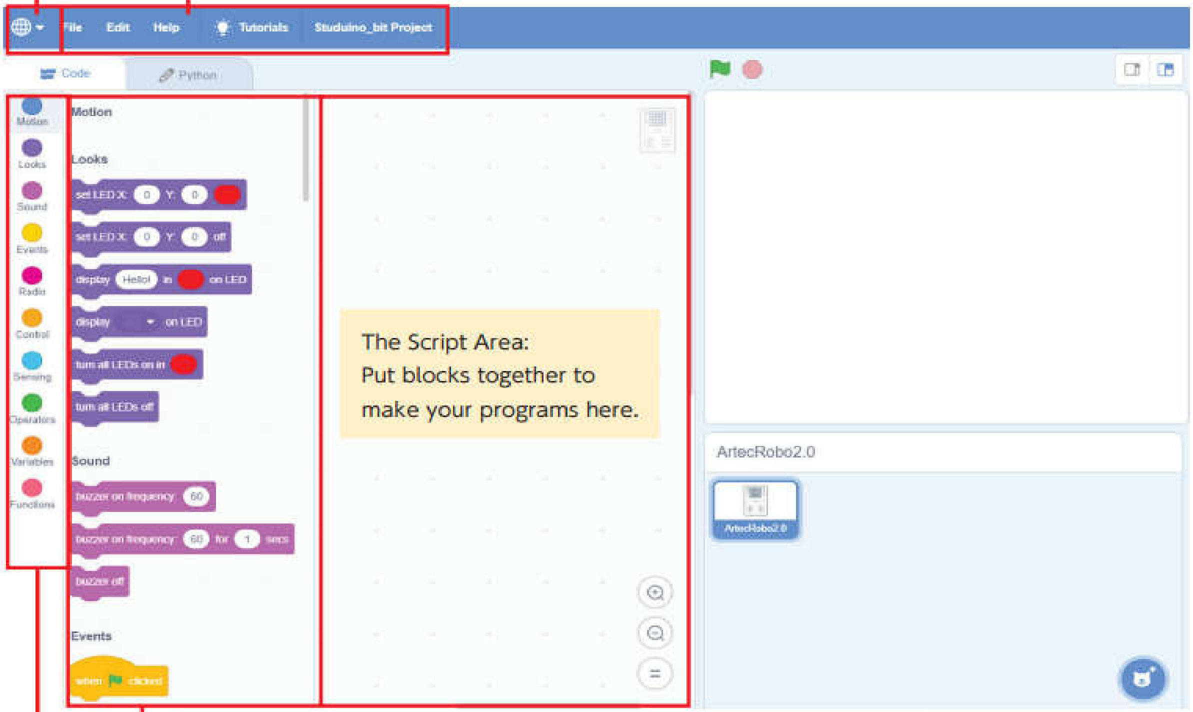
Character mode

You can convert your blocks into Python code and transfer it to Studuino:bit.

You can add sprites and program them to move on the screen.

Choose from rom English, Spanish, French, Korean, Japanese (kanji), Japanese (hiragana), Portuguese, Russian, or Chinese (traditional). ★ As of version 1.4.2.

Menu



The Script Area: Put blocks together to make your programs here.

Block Palette: Find the blocks (commands) you can use to make programs here.

Catergories: The Block Palette is divided into different categories of blocks (commands). Click on the icon for a category here to jump to its section of the Block Palette.

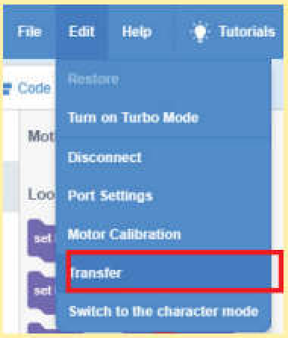
Co-funded by the Erasmus+ Programme of the European Union

LBC LEARNING BY COMPETING

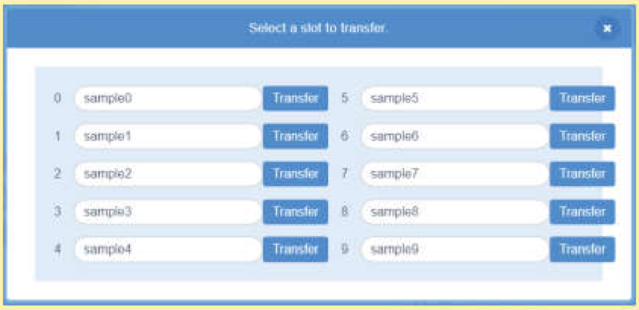
Running Programs with Transfer

1. Connect your Core Unit to your computer using a USB cable.

2. Select **Transfer** from the Studuino:bit Software's **Edit** menu.



5. Select a slot on the Core Unit from **0** to **9** to send the program to. You can also rename the program here.



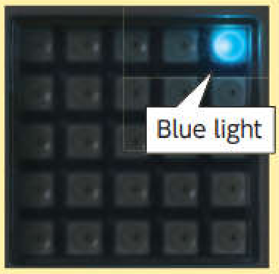
Select a slot to transfer.

0	sample0	Transfer	5	sample5	Transfer
1	sample1	Transfer	6	sample6	Transfer
2	sample2	Transfer	7	sample7	Transfer
3	sample3	Transfer	8	sample8	Transfer
4	sample4	Transfer	9	sample9	Transfer

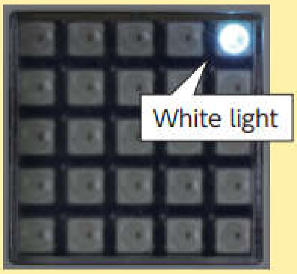
6. When the transfer is complete, the Core Unit will start running the program automatically. To restart the program from the beginning, press the **Reset** button.

If the transfer is successful, the LED in the upper right corner of the display will light up blue. If the transfer fails, the same LED will light up white instead. If this happens, press the Reset button and try again.

Successful Transfer



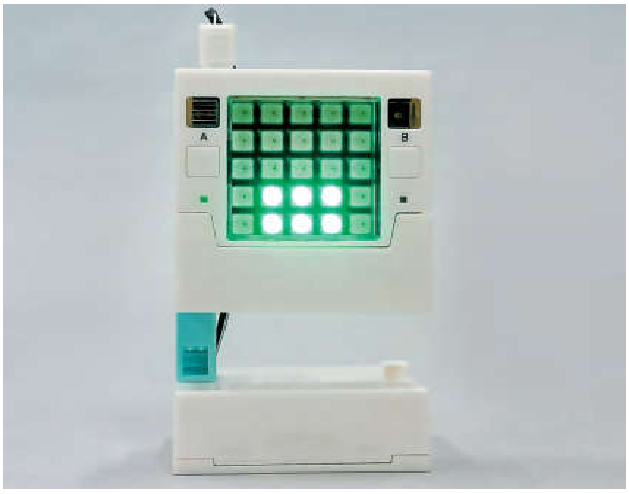
Failed Transfer



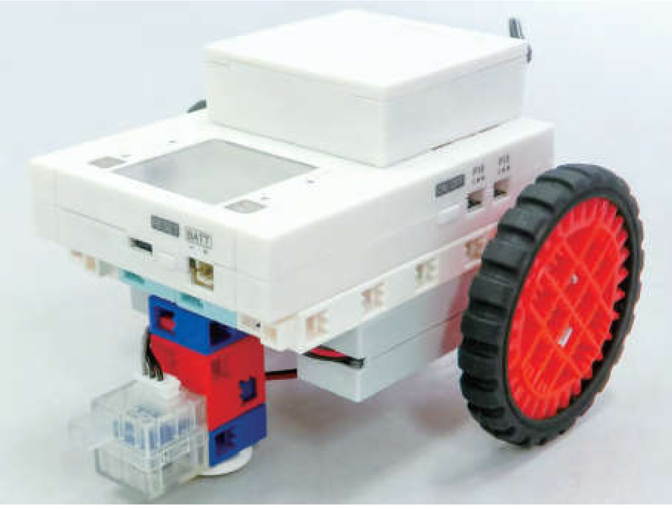
Co-funded by the Erasmus+ Programme of the European Union

4 Running the Sample Programs

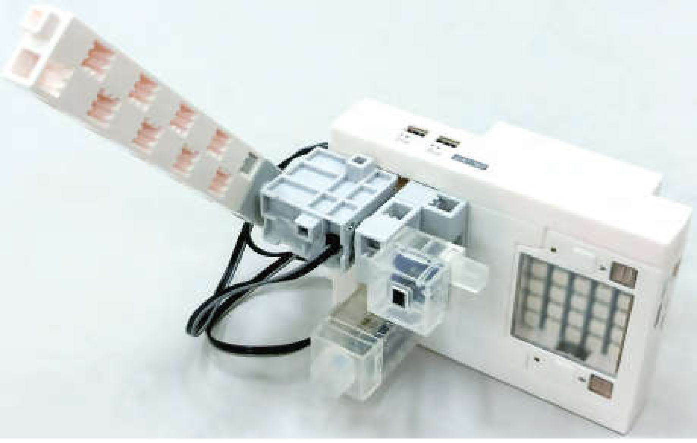
Exercise 1: Traffic Signals



Exercise 2: Robot Car



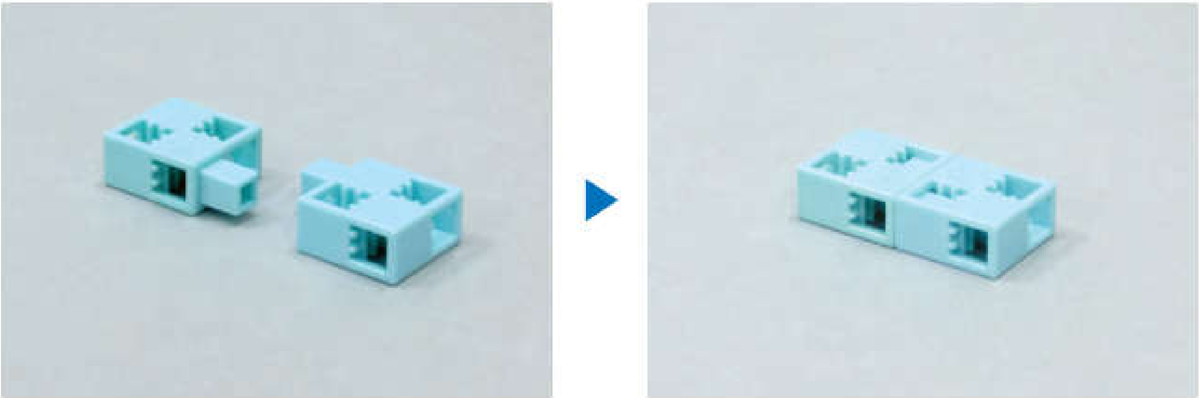
Exercise 3: Gate System



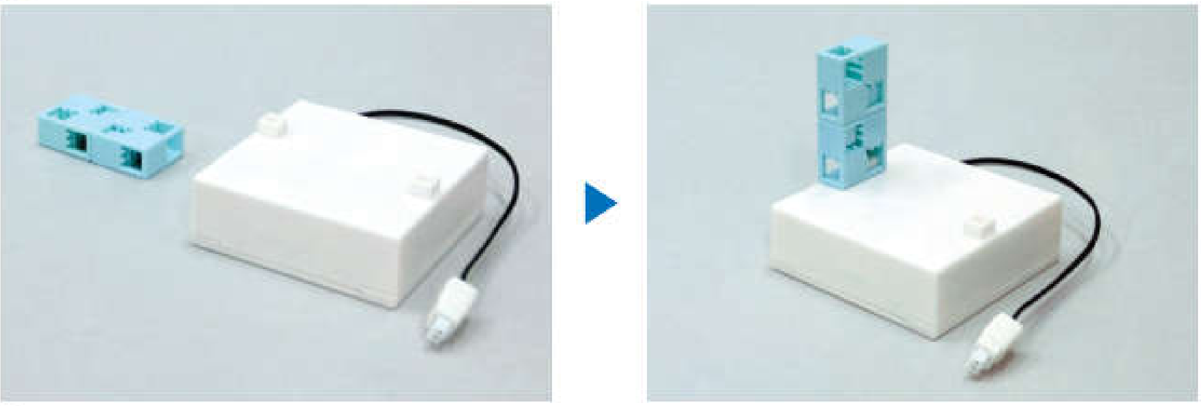
Exercise 1: Traffic Signals

Step 1: Build it

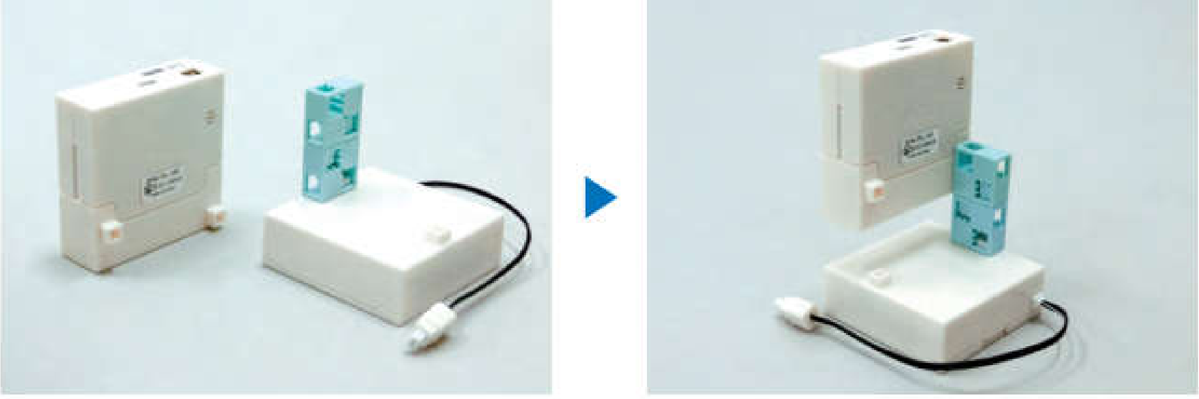
①



②



③

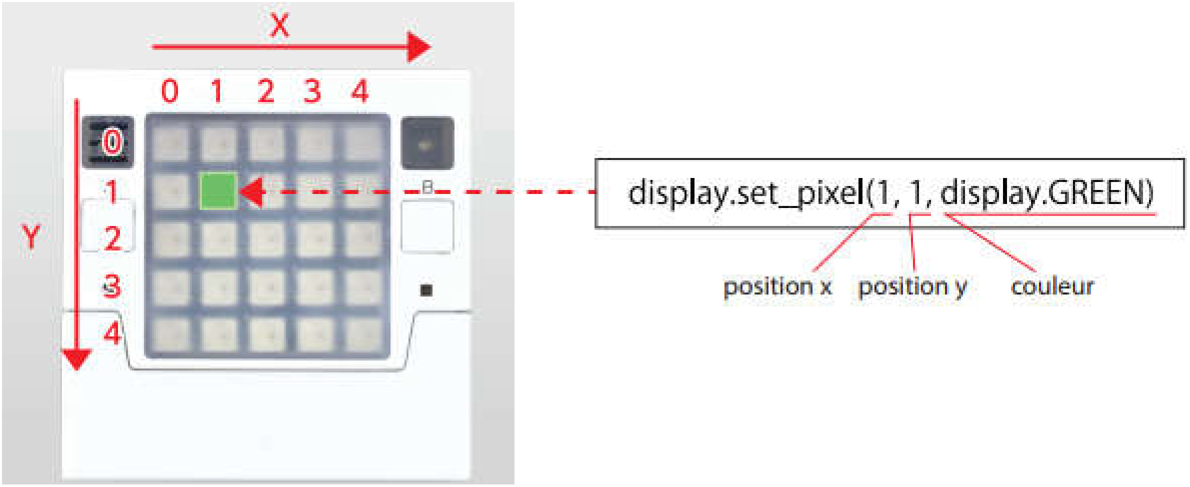


Co-funded by the
Erasmus+ Programme
of the European Union

LEARNING BY COMPETING

Step 2: Code it

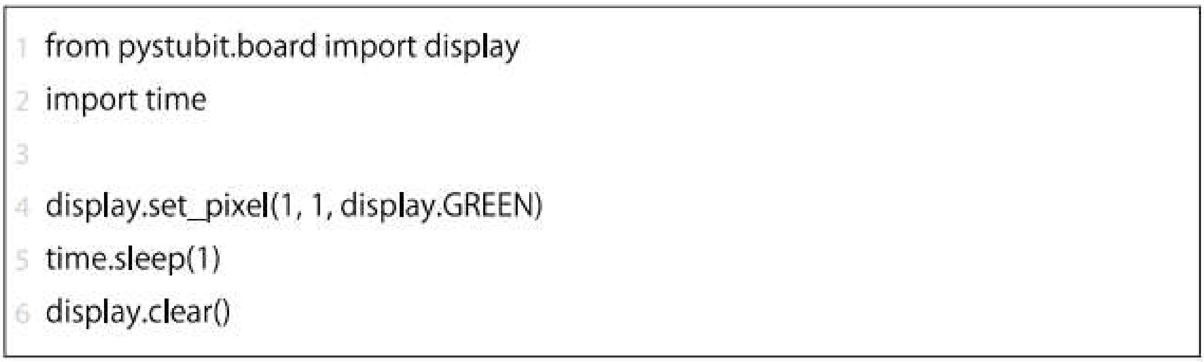
The LED display is made up of 25 individual LEDs in a 5 x 5 grid. You can make each and every one light up in any color you want. Each individual LED has specific X and Y coordinates to describe its position, as shown below.



```
display.set_pixel(1, 1, display.GREEN)
```

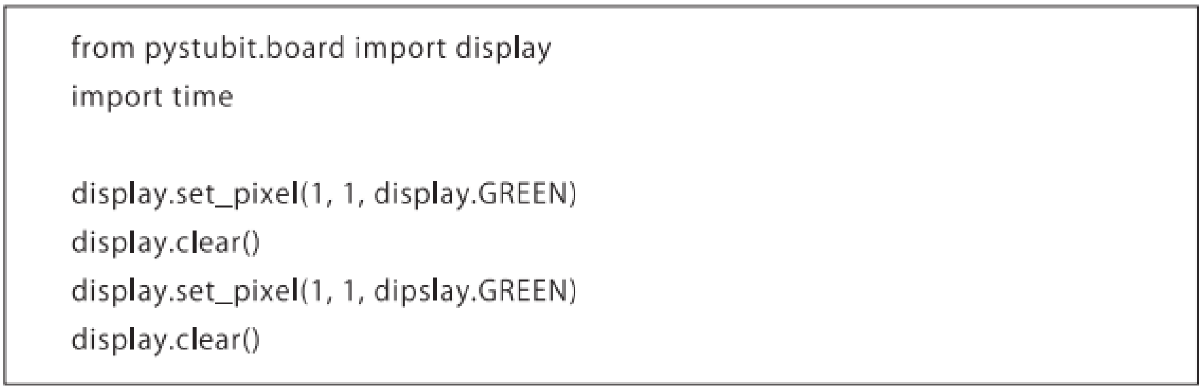
Let's code…

Code n°1 :



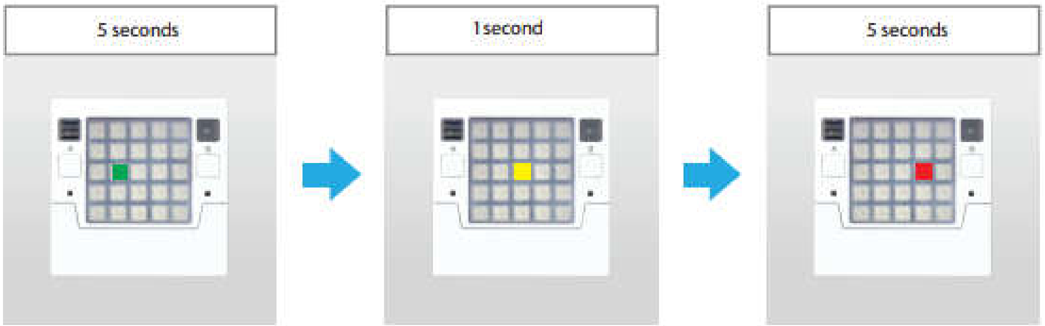
```
from pystubit.board import display
import time

display.set_pixel(1, 1, display.GREEN)
time.sleep(1)
display.clear()
```

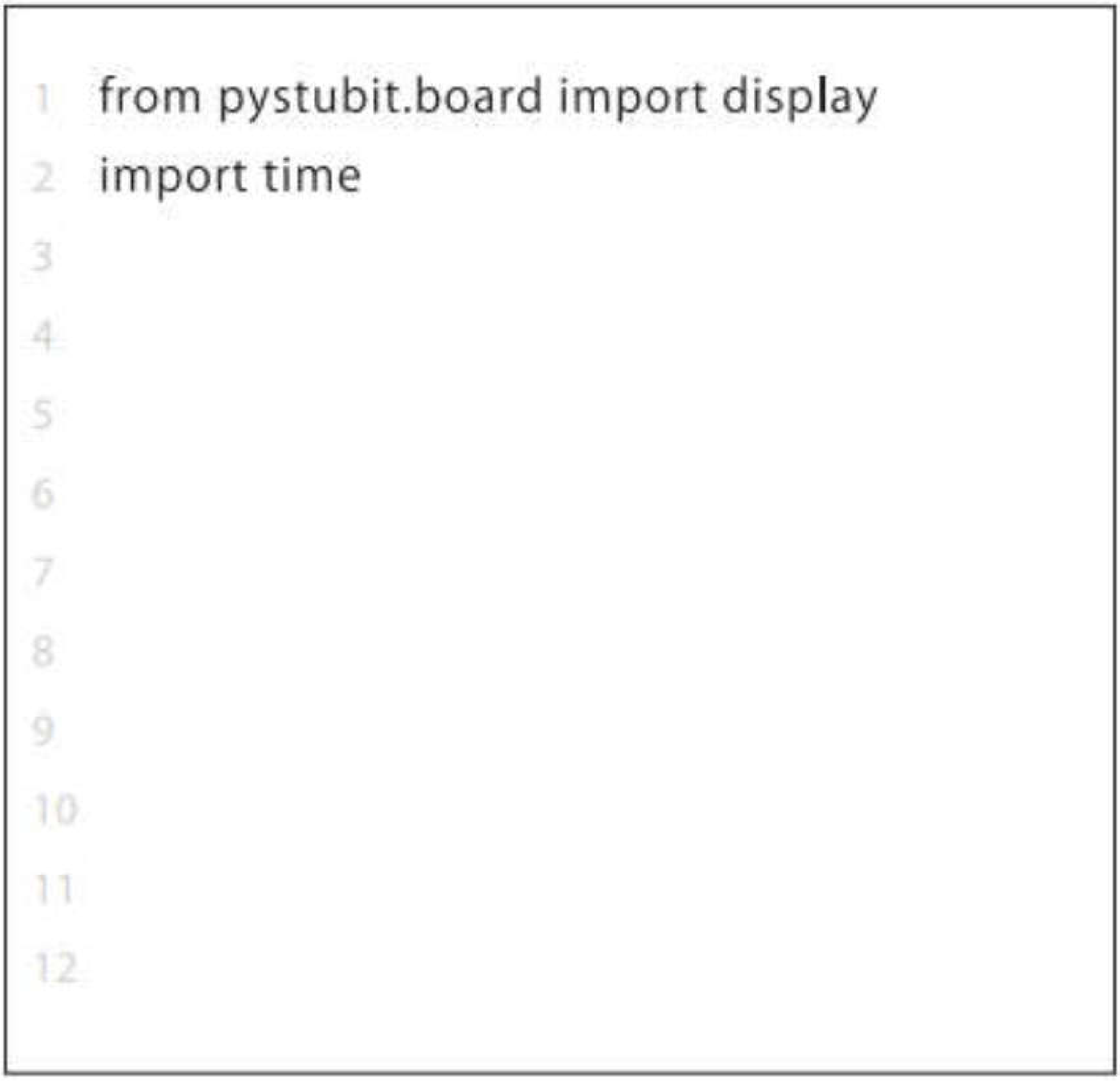
Code n°2 :



```
from pystubit.board import display
import time

display.set_pixel(1, 1, display.GREEN)
display.clear()
display.set_pixel(1, 1, dipslay.GREEN)
display.clear()
```

Step 3: do it by yourself

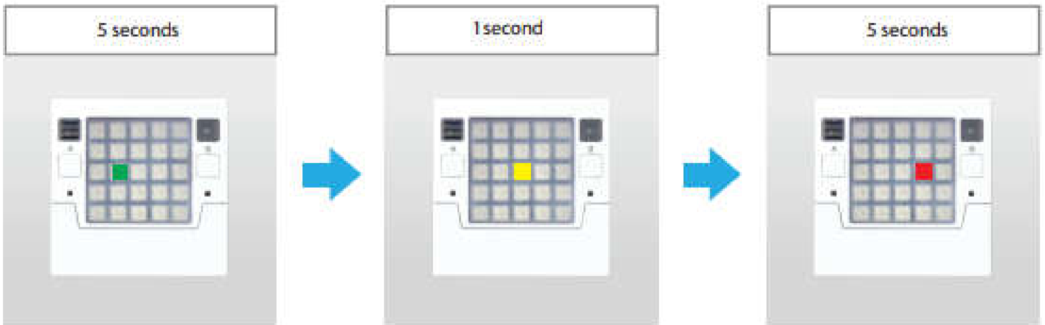


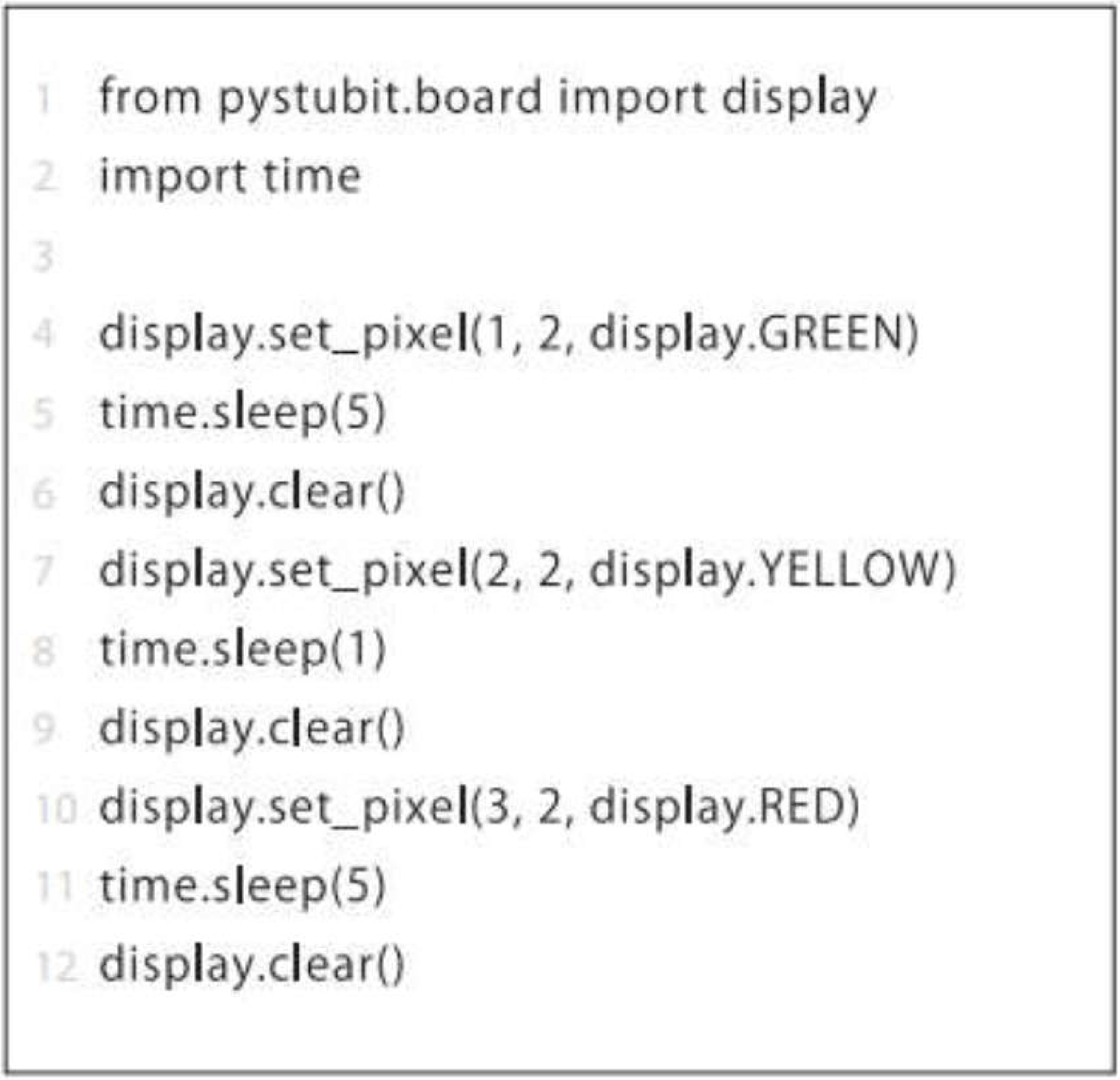


```
from pystubit.board import display
import time
```

Solution on the next page…

Co-funded by the
Erasmus+ Programme
of the European Union



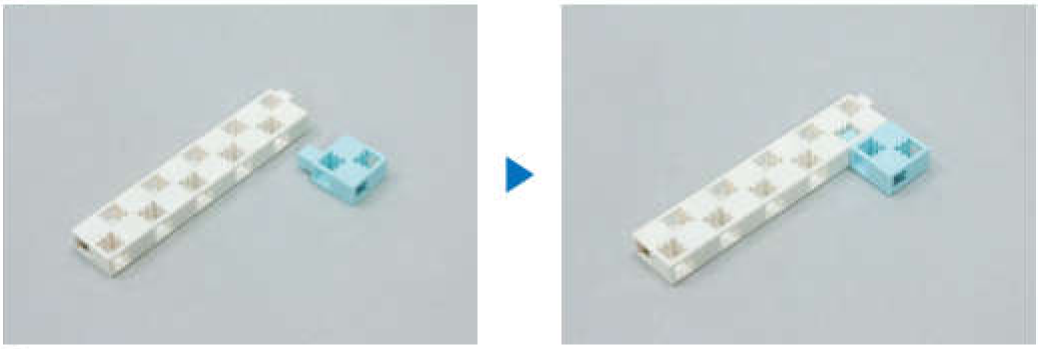


```
from pystubit.board import display
import time

display.set_pixel(1, 2, display.GREEN)
time.sleep(5)
display.clear()
display.set_pixel(2, 2, display.YELLOW)
time.sleep(1)
display.clear()
display.set_pixel(3, 2, display.RED)
time.sleep(5)
display.clear()
```

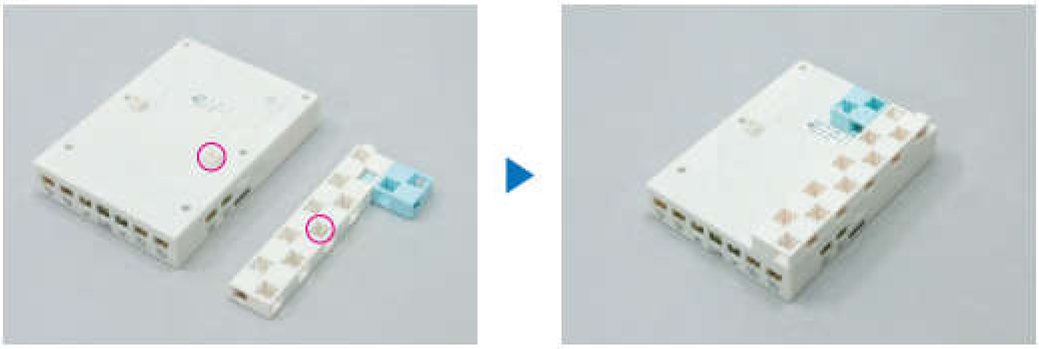
Exercise 2: Robot Car

Step 1: Build it

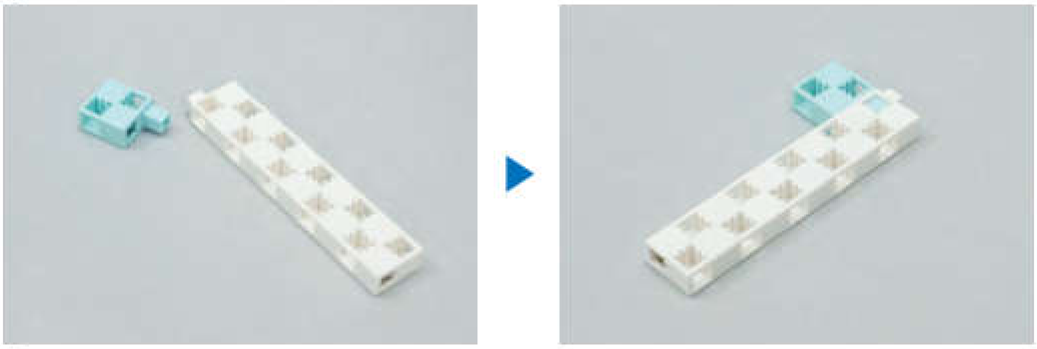
①



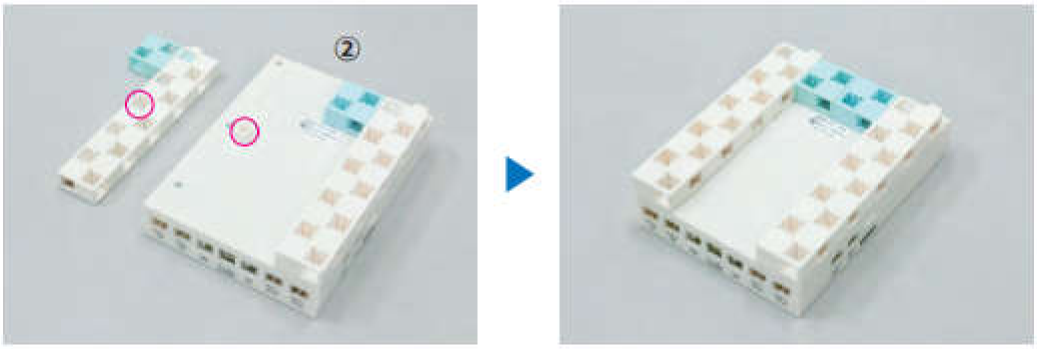
②



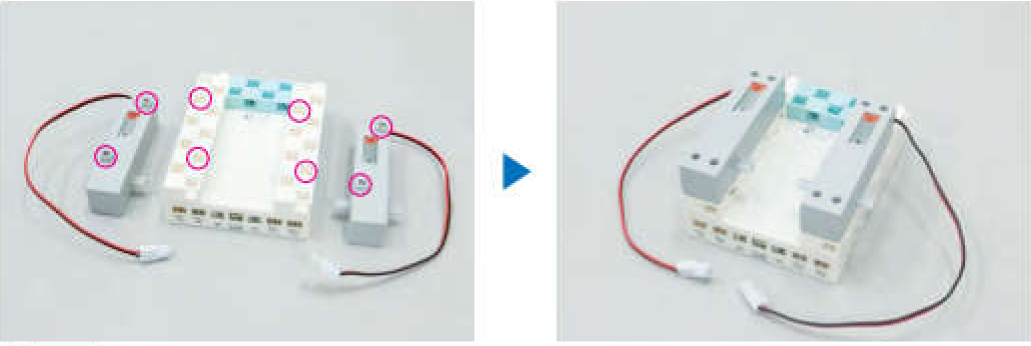
③



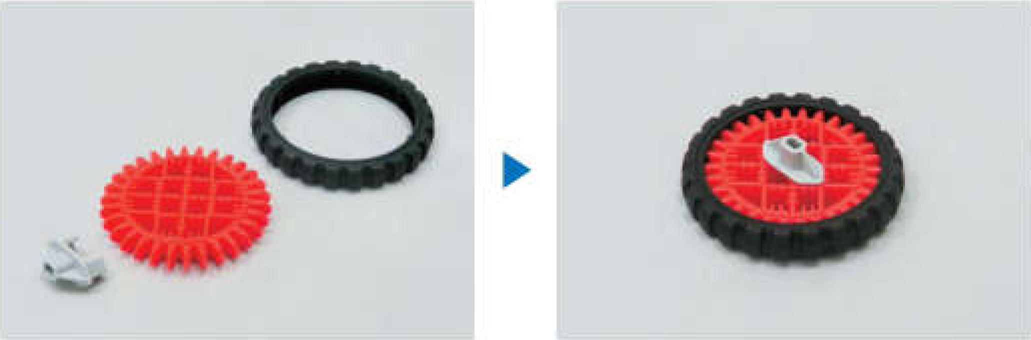
④



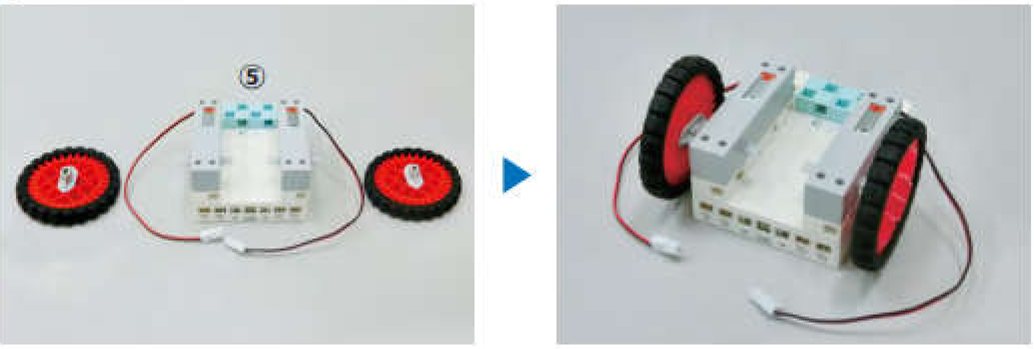
⑤



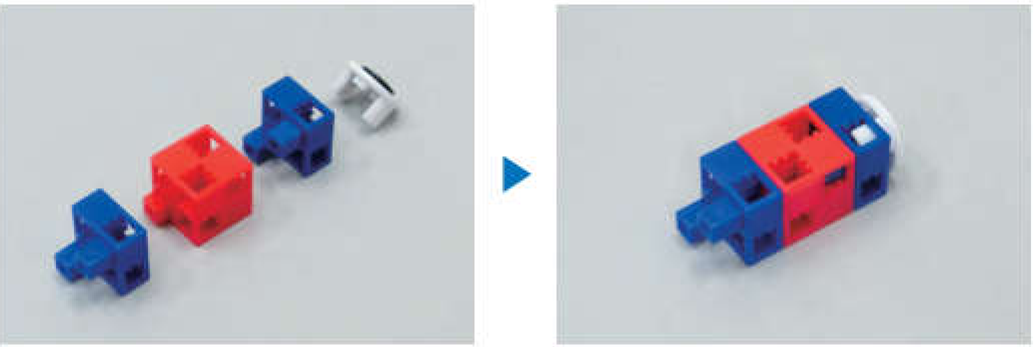
⑥ ×2



⑦



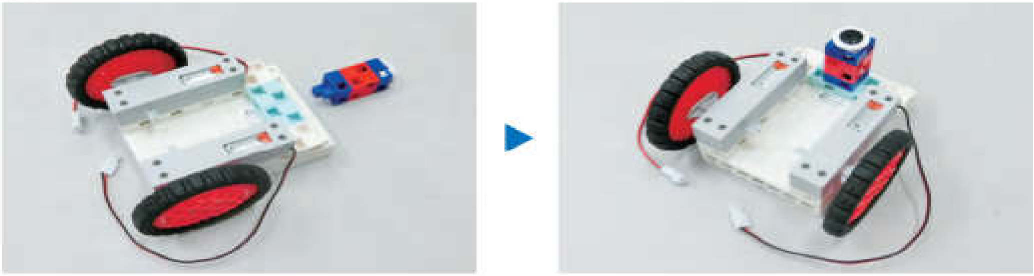
⑧



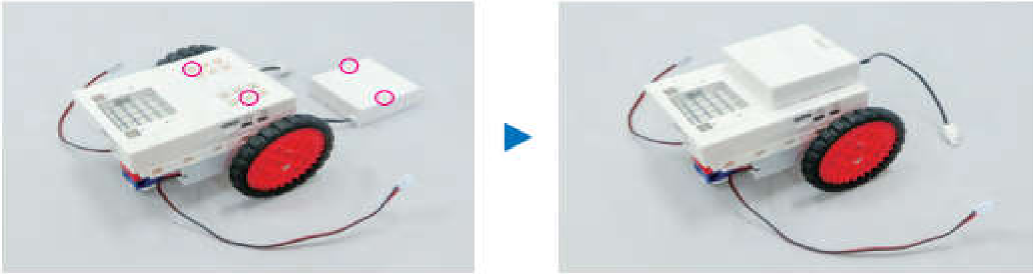
Co-funded by the
Erasmus+ Programme
of the European Union

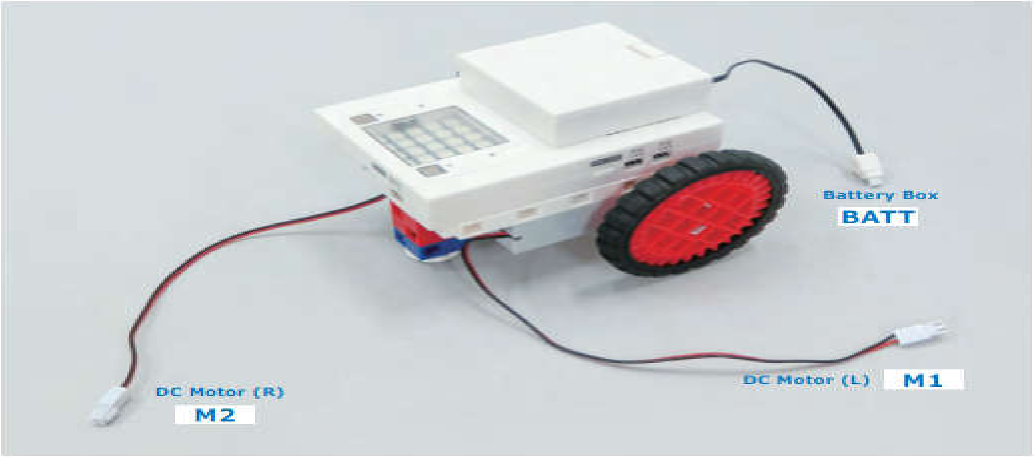
LBC LEARNING BY COMPETING

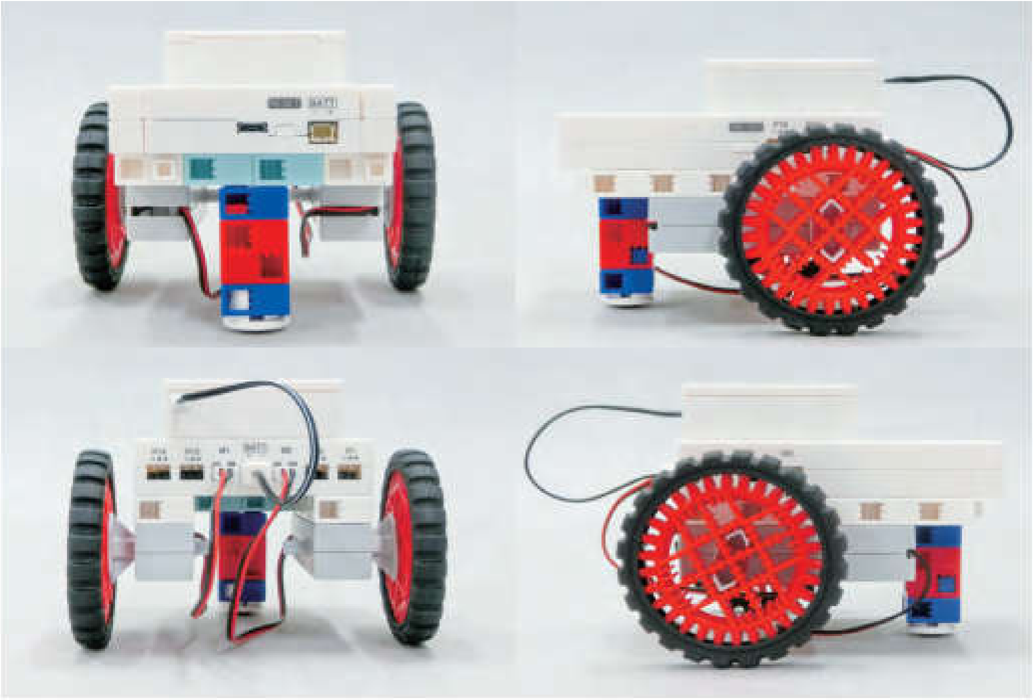
⑨



⑩

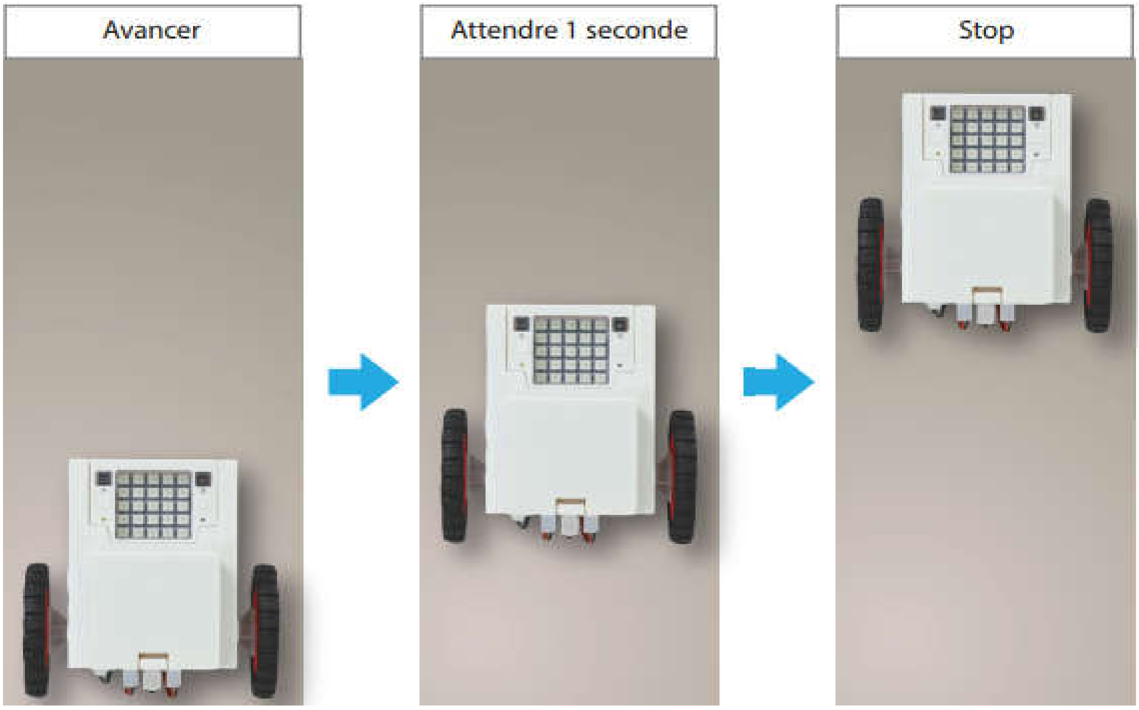




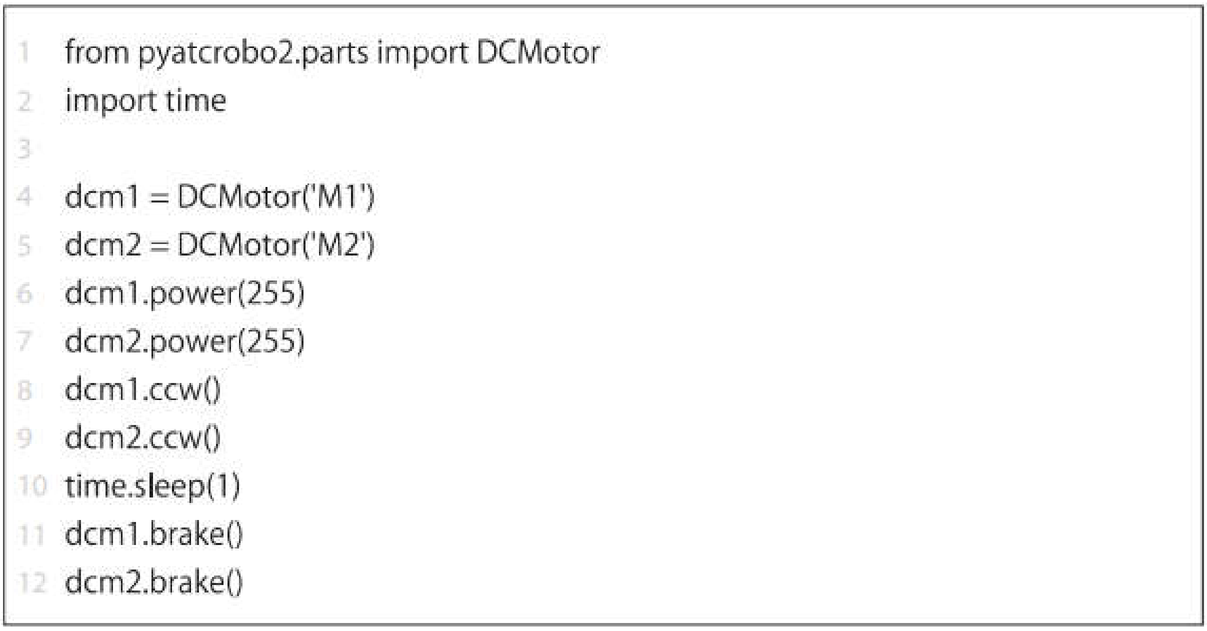


Step 2: Code it

With this program, the car will move forward for 1 second :



Let’s code…



```
from pyatcrobo2.parts import DCMotor
import time

dcm1 = DCMotor('M1')
dcm2 = DCMotor('M2')
dcm1.power(255)
dcm2.power(255)
dcm1.ccw()
dcm2.ccw()
time.sleep(1)
dcm1.brake()
dcm2.brake()
```

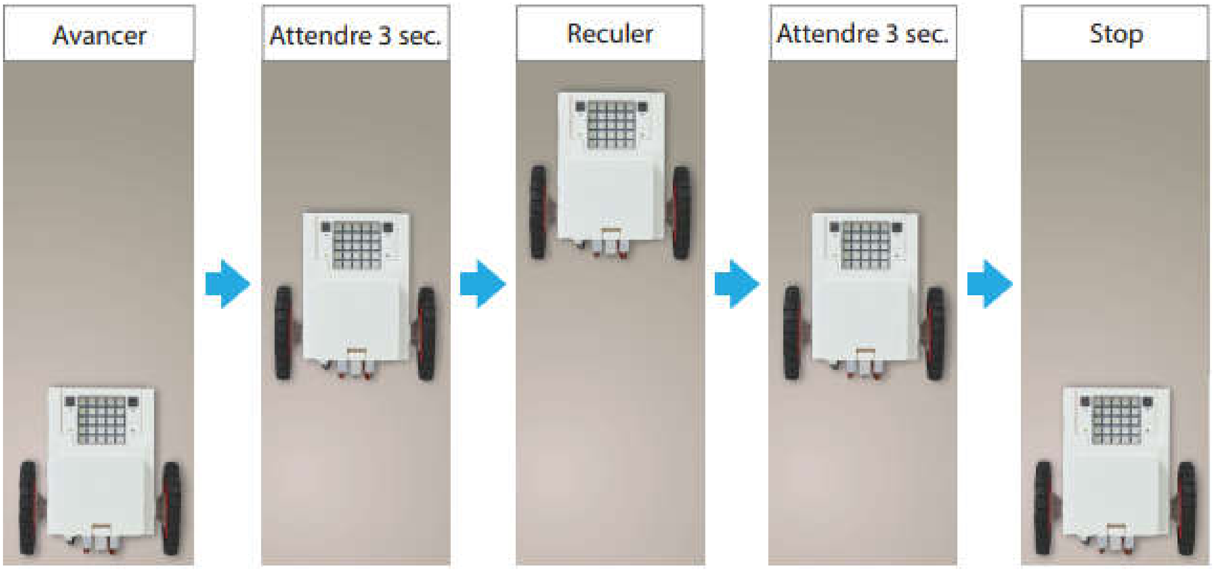
Note :

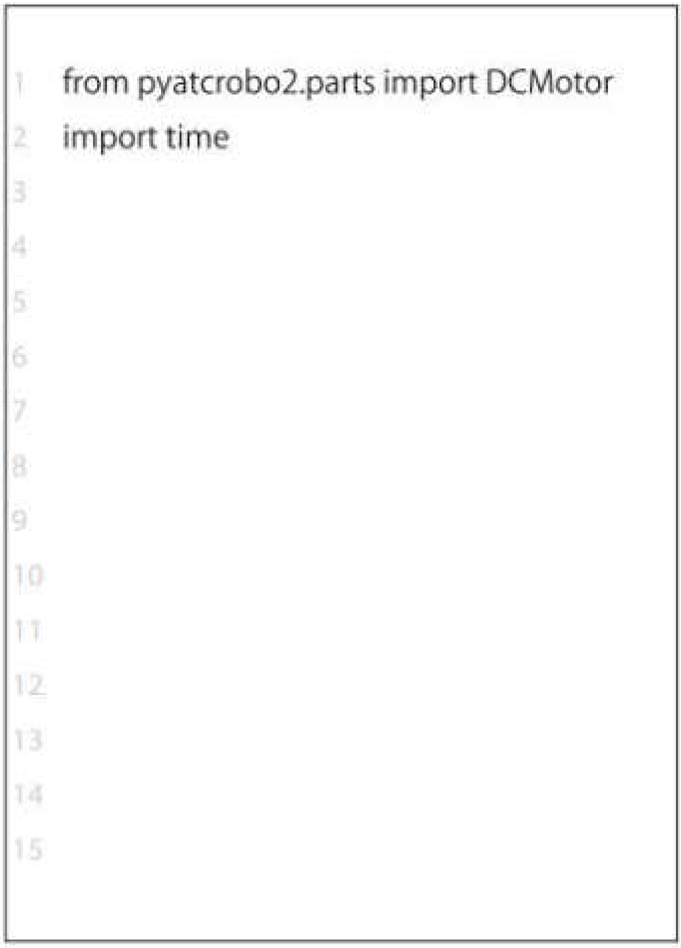
****ccw() stand for counter clockwise and makes the wheel go forward***

Co-funded by the Erasmus+ Programme of the European Union

LBC LEARNING BY COMPETING

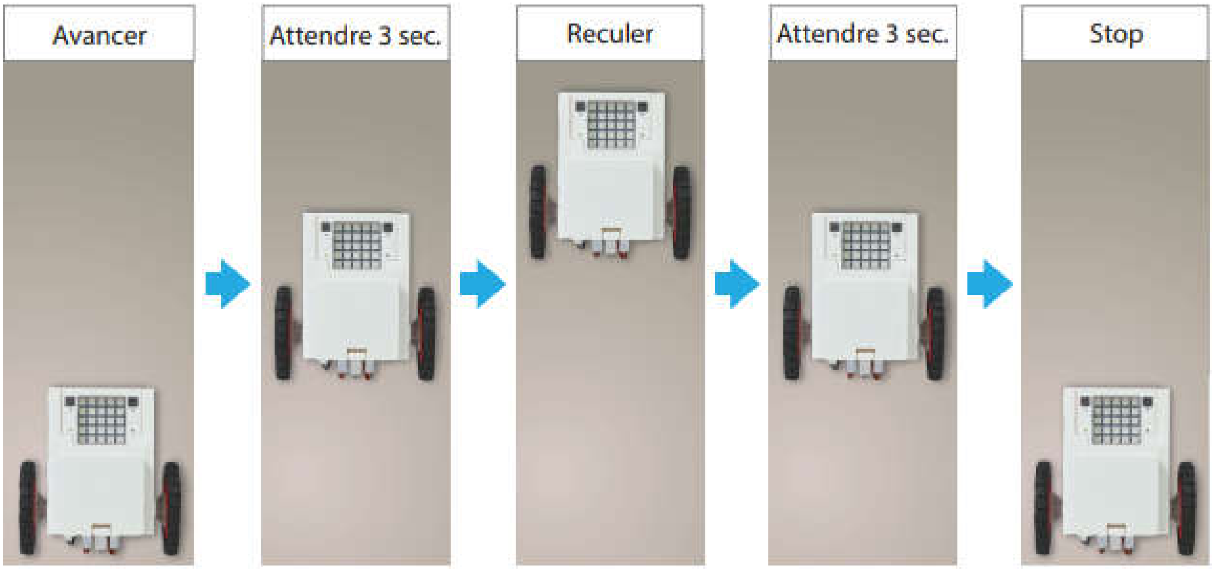
Step 3: do it by yourself





```
from pyatcrobo2.parts import DCMotor
import time
```

Solution on the next page…

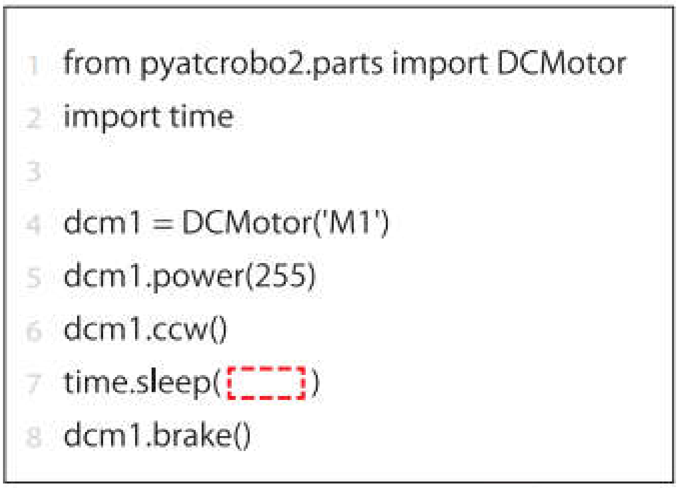


```python
from pyatcrobo2.parts import DCMotor
import time

dcm1 = DCMotor('M1')
dcm2 = DCMotor('M2')
dcm1.power(255)
dcm2.power(255)
dcm1.ccw()
dcm2.ccw()
time.sleep(3)
dcm1.cw()
dcm2.cw()
time.sleep(3)
dcm1.brake()
dcm2.brake()
```

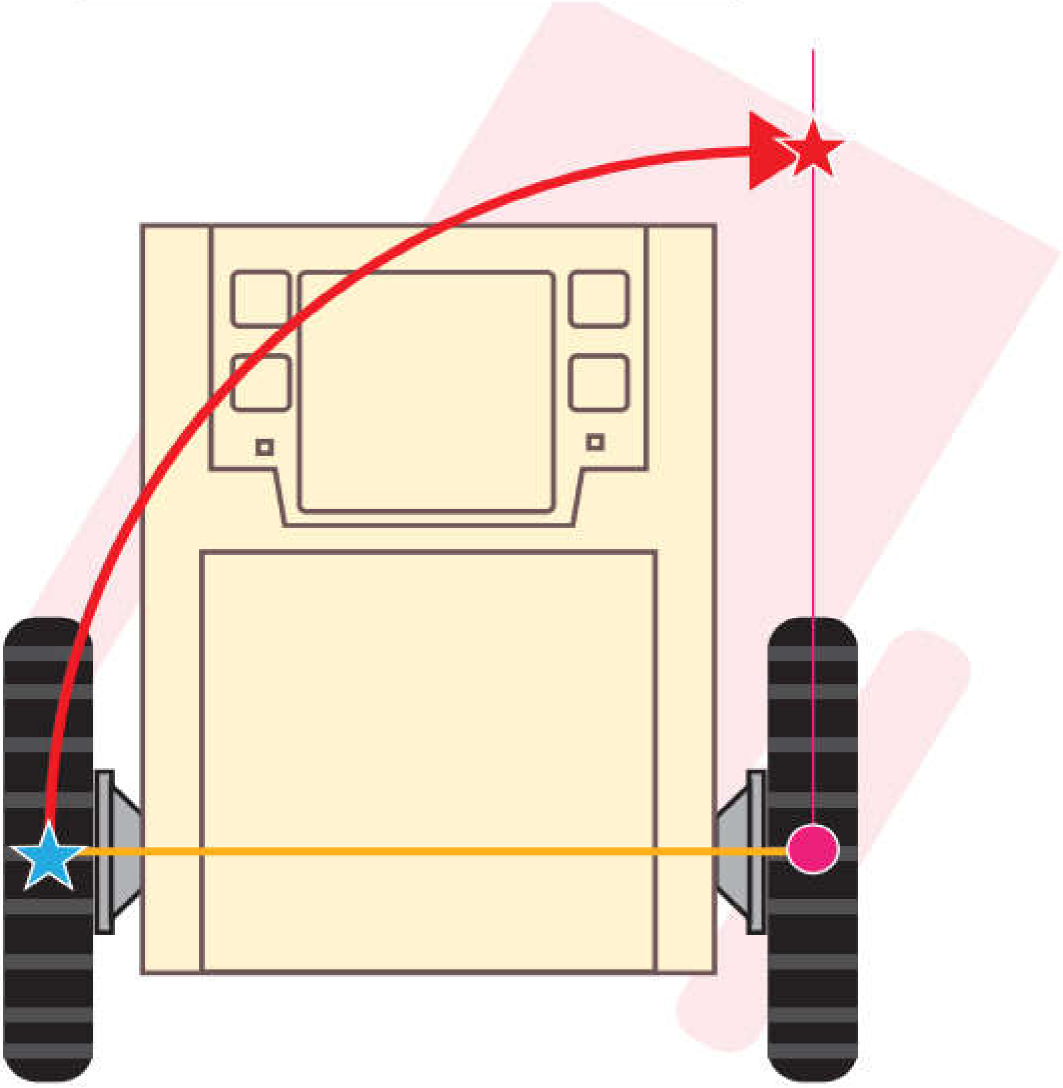
Co-funded by the
Erasmus+ Programme
of the European Union

Step 4:

Find the missing number to turn 90° to the right

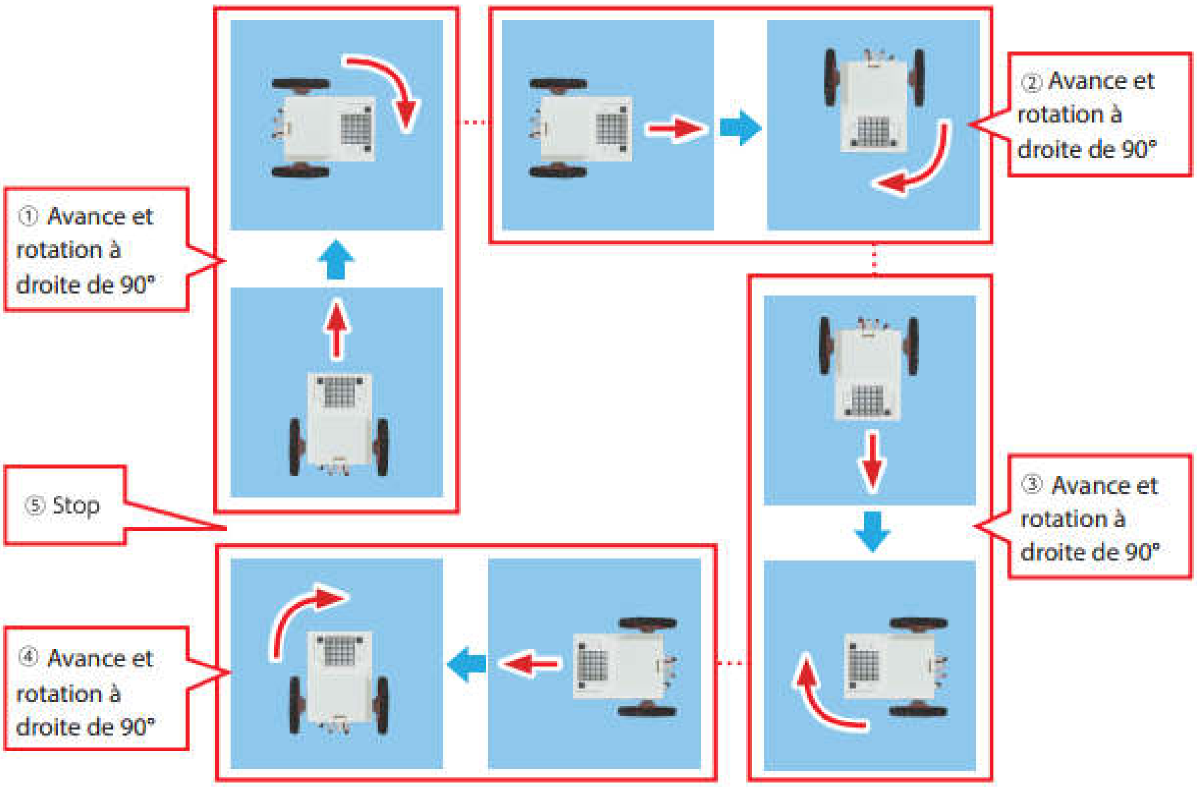


```
from pyatcrobo2.parts import DCMotor
import time

dcm1 = DCMotor('M1')
dcm1.power(255)
dcm1.ccw()
time.sleep(□)
dcm1.brake()
```



Co-funded by the
Erasmus+ Programme
of the European Union

Step 5: for the bravest students…



① Avance et rotation à droite de 90°

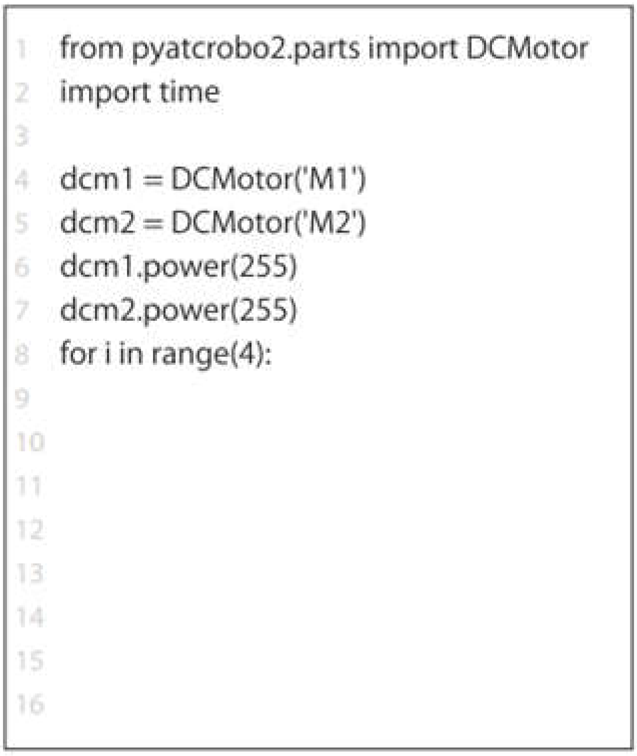
② Avance et rotation à droite de 90°

③ Avance et rotation à droite de 90°

④ Avance et rotation à droite de 90°

⑤ Stop

Let's code…

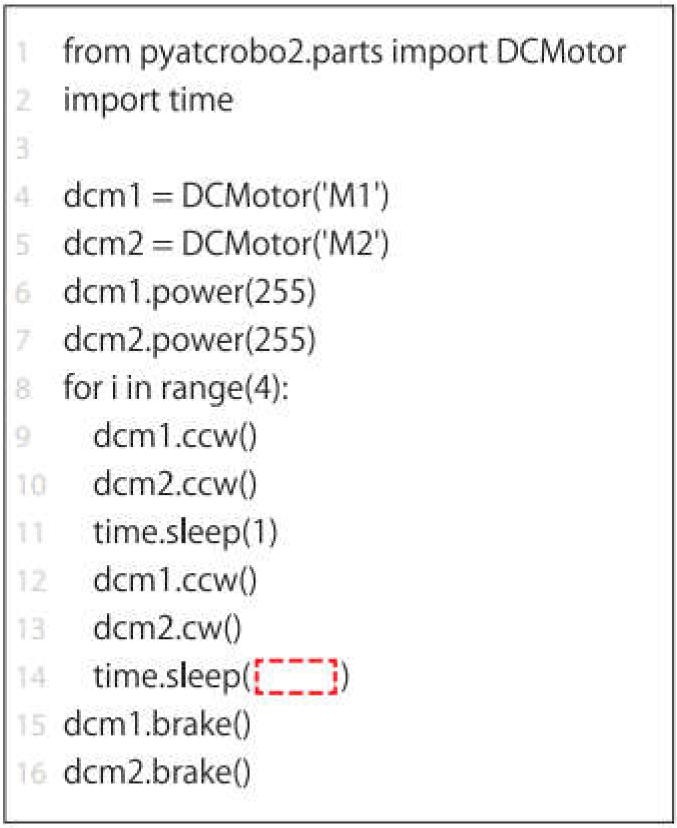


```
from pyatcrobo2.parts import DCMotor
import time

dcm1 = DCMotor('M1')
dcm2 = DCMotor('M2')
dcm1.power(255)
dcm2.power(255)
for i in range(4):
```

Co-funded by the Erasmus+ Programme of the European Union

LEARNING BY COMPETING

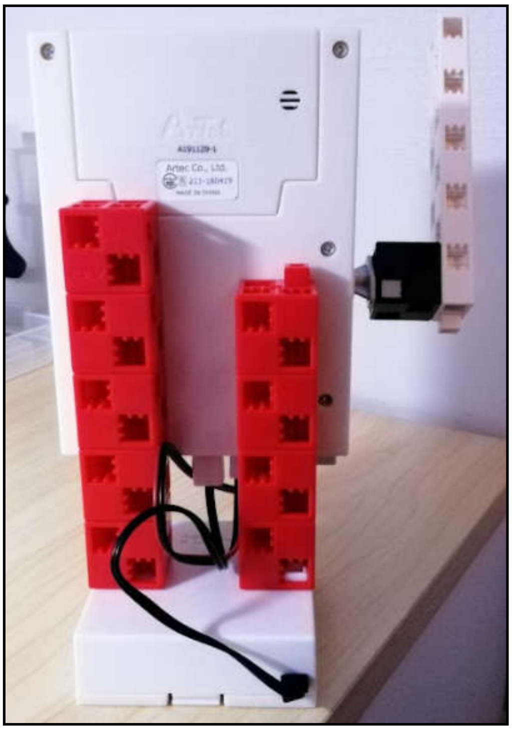
Solution :

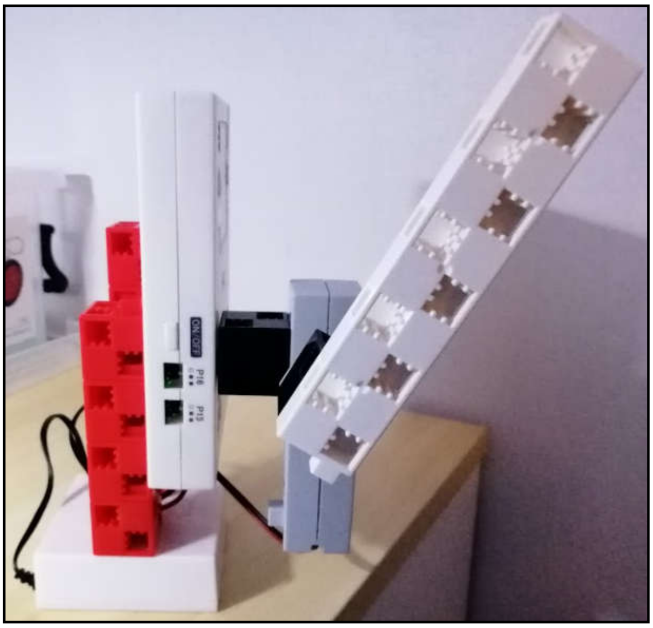


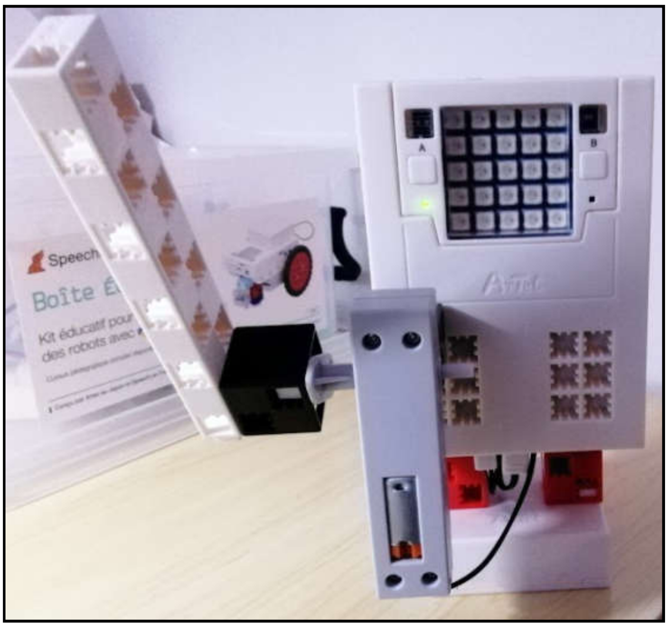
```
from pyatcrobo2.parts import DCMotor
import time

dcm1 = DCMotor('M1')
dcm2 = DCMotor('M2')
dcm1.power(255)
dcm2.power(255)
for i in range(4):
    dcm1.ccw()
    dcm2.ccw()
    time.sleep(1)
    dcm1.ccw()
    dcm2.cw()
    time.sleep(    )
dcm1.brake()
dcm2.brake()
```

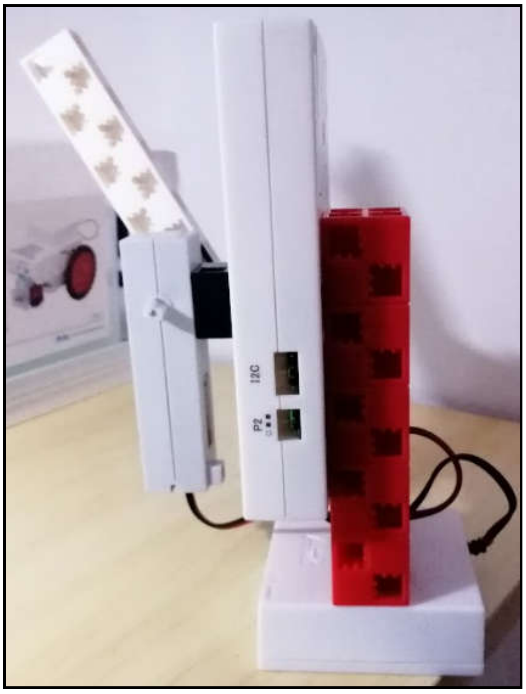
Exercise 3: Gate System

Step 1: Build it





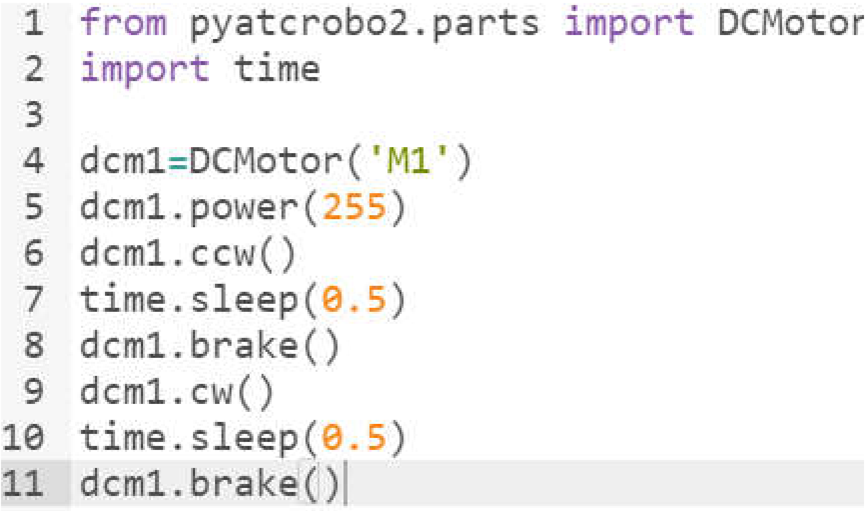




Step 2: Code it

This program will make the gate go up and down

Let’s code…

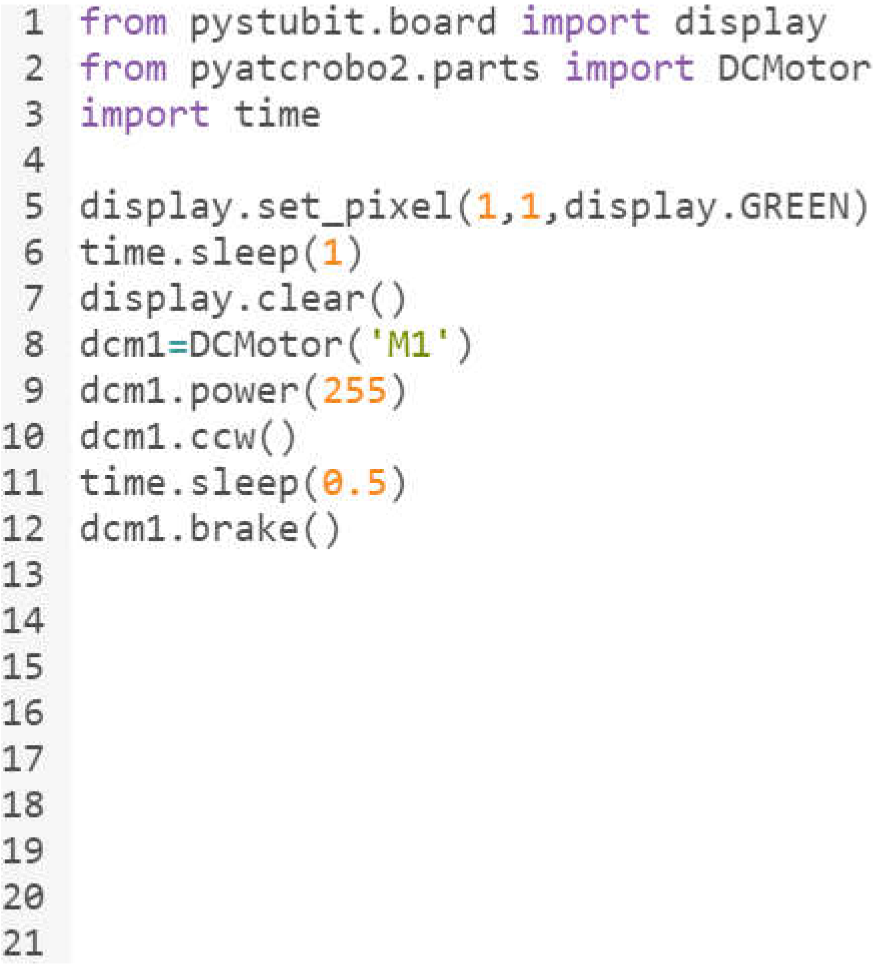


```python
from pyatcrobo2.parts import DCMotor
import time

dcm1=DCMotor('M1')
dcm1.power(255)
dcm1.ccw()
time.sleep(0.5)
dcm1.brake()
dcm1.cw()
time.sleep(0.5)
dcm1.brake()
```

Step 3: do it by yourself

Complete the program to…

1. **make the gate go up**
2. **make a green light appear for 1 second**
3. **make a yellow light appear for 1 second**
4. **make the gate go down**
5. **make a red light appear for 1 second**

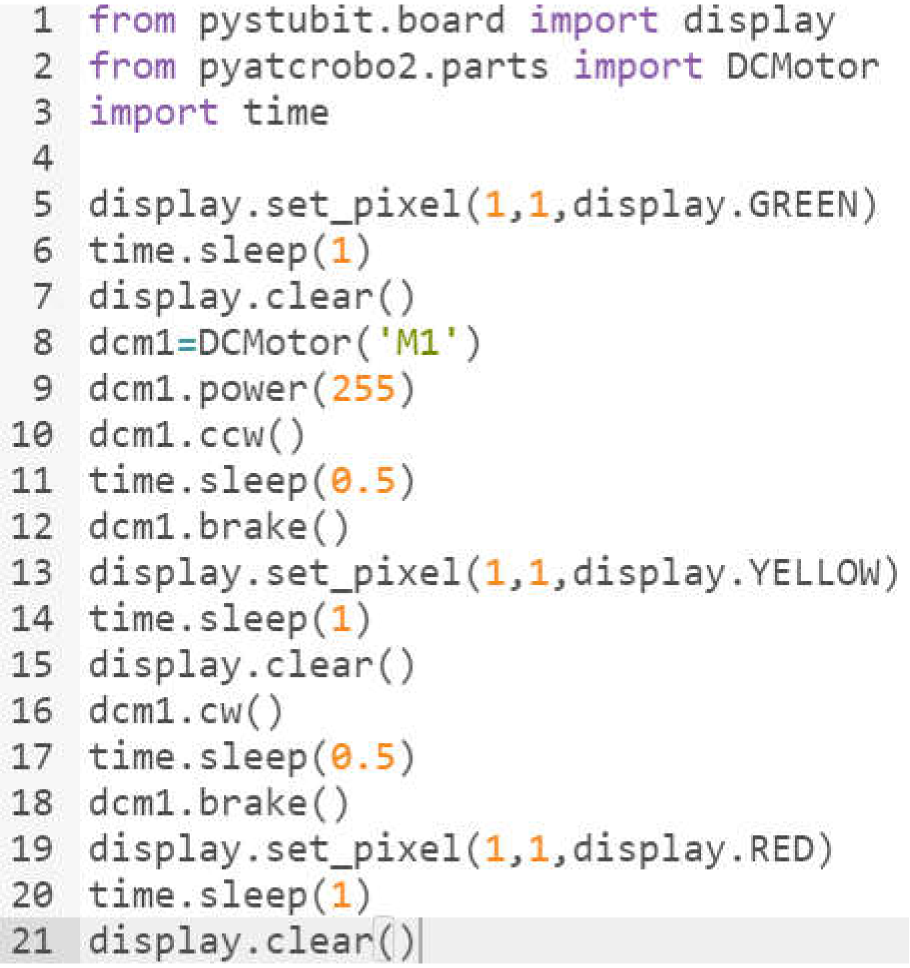


```
from pystubit.board import display
from pyatcrobo2.parts import DCMotor
import time

display.set_pixel(1,1,display.GREEN)
time.sleep(1)
display.clear()
dcm1=DCMotor('M1')
dcm1.power(255)
dcm1.ccw()
time.sleep(0.5)
dcm1.brake()
```

Solution on the next page…

Co-funded by the
Erasmus+ Programme
of the European Union

LBC LEARNING BY COMPETING



```python
from pystubit.board import display
from pyatcrobo2.parts import DCMotor
import time

display.set_pixel(1,1,display.GREEN)
time.sleep(1)
display.clear()
dcm1=DCMotor('M1')
dcm1.power(255)
dcm1.ccw()
time.sleep(0.5)
dcm1.brake()
display.set_pixel(1,1,display.YELLOW)
time.sleep(1)
display.clear()
dcm1.cw()
time.sleep(0.5)
dcm1.brake()
display.set_pixel(1,1,display.RED)
time.sleep(1)
display.clear()
```