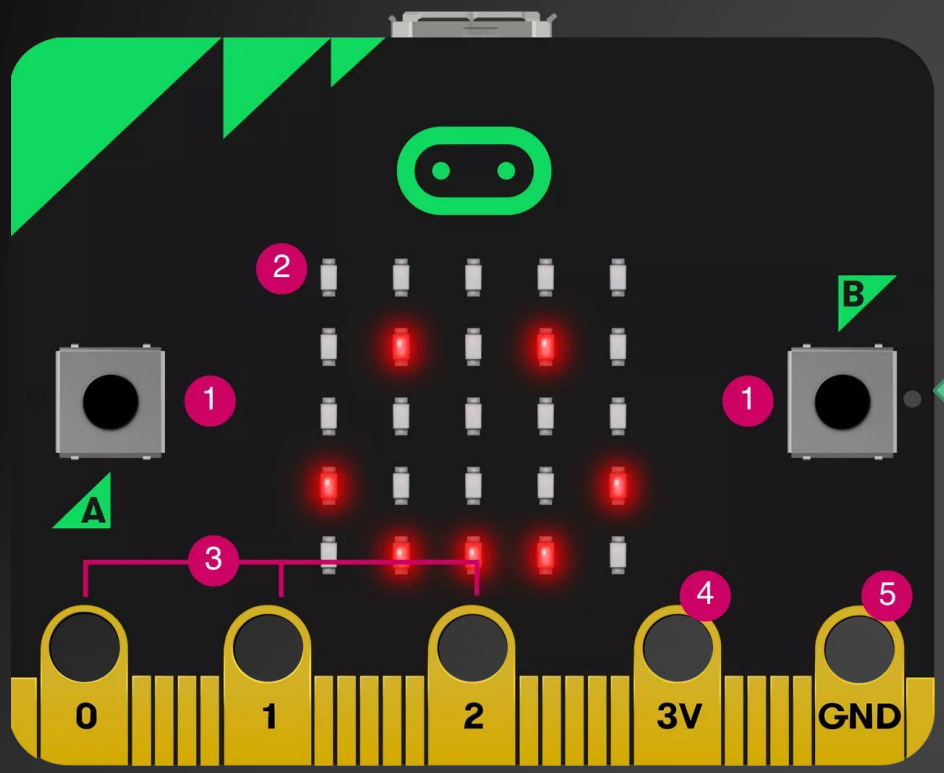
PYTHON ROBOTICS

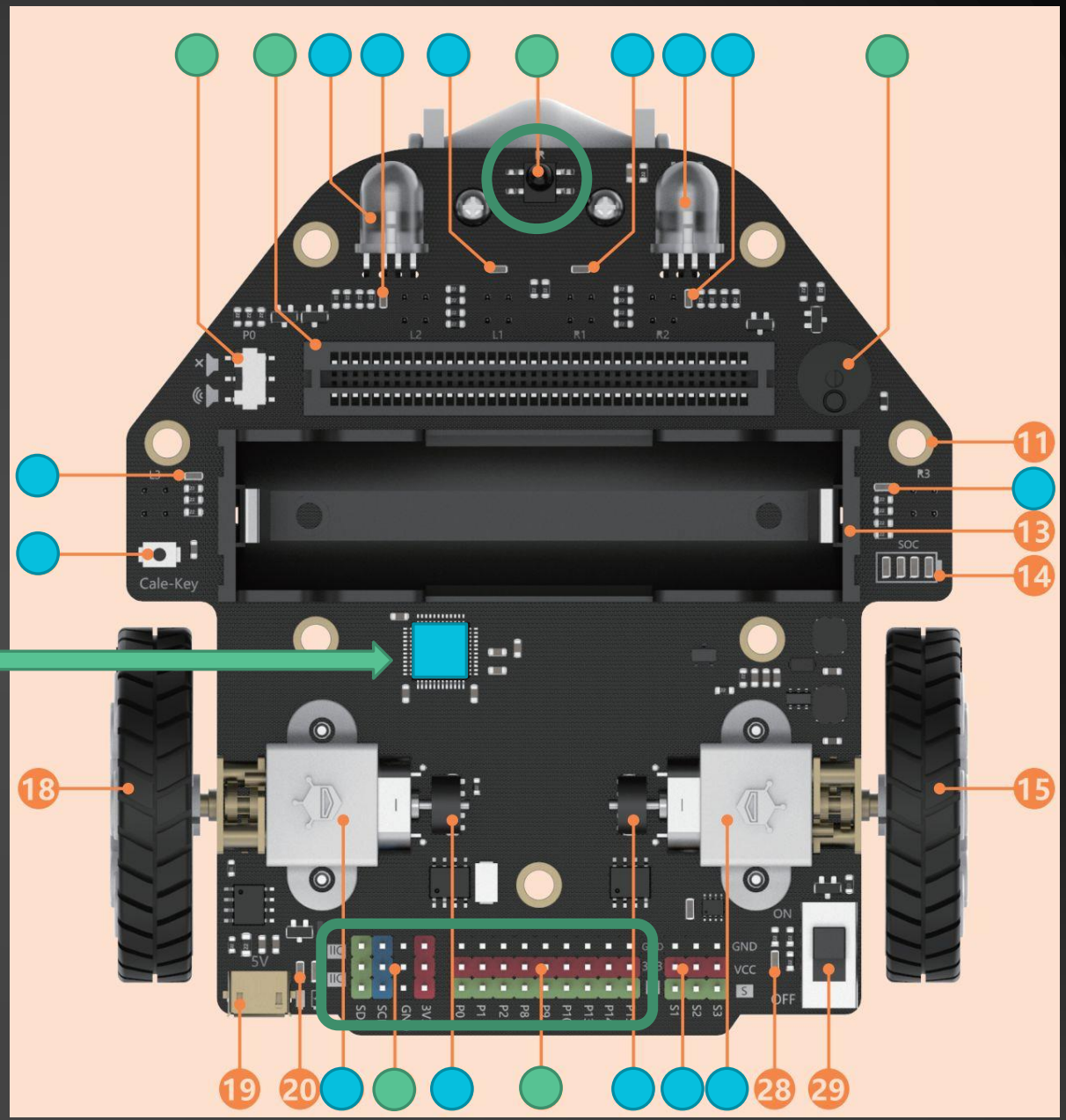
WORKSHOP 2

MAQUEEN PLUS (V1.0)

- 3.7V-18650 lithium battery power supply
- 5V charging voltage
- 900mA charging current
- 4 LEDs battery indicator
- N20 motor 260rpm drive motor
- Buzzer
- RGB-LED x 2
- P0 P1 P2 P8 P12 P13 P14 P15 P16 GPIO expansion ports
- I2C expansion ports x 3
- Servo expansion ports x 3
- Line tracking sensors x 6
- Analog + digital line tracking sensor output data
- Support line sensor calibration
- Infrared receiving sensor
- URM10 ultrasonic sensor
- 50cm x 50cm map size

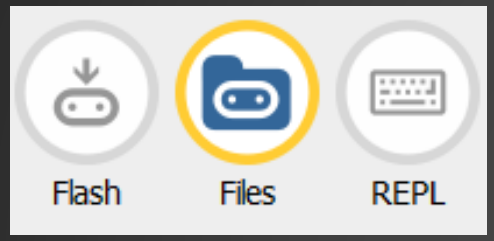

2
B
1
1
A
3
4
5
0
1
2
3V
GND

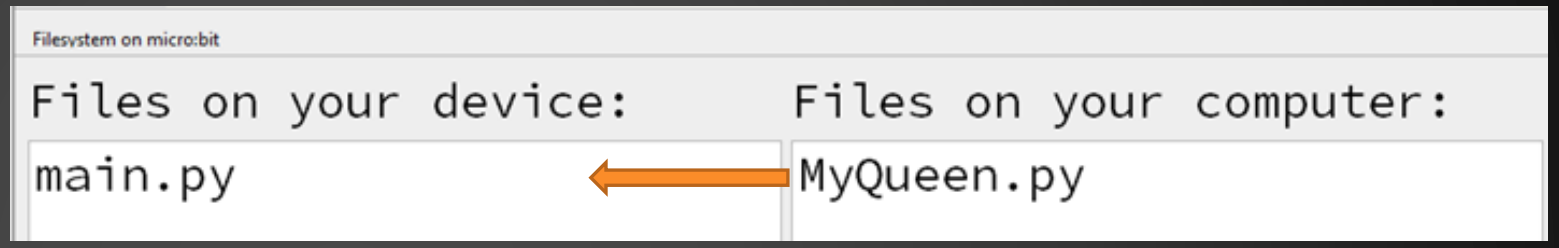
I2C


P0
L2
L1
R1
R2
L3
R3
SOC
Cale-Key
11
13
14
18
15
19
20
28
29
5V
SD
SC
GND
VCC
ON
OFF

OEFENING: RGB LED

- Zet 'MyQueen.py' in '%userprofile%\mu_code'
- Library overzetten met 'Files'





- Code hier rechts invoeren in Mu
- Code overzetten met 'Flash'

Problemen?
Probeer eerst
'hello world' uit
workshop 1

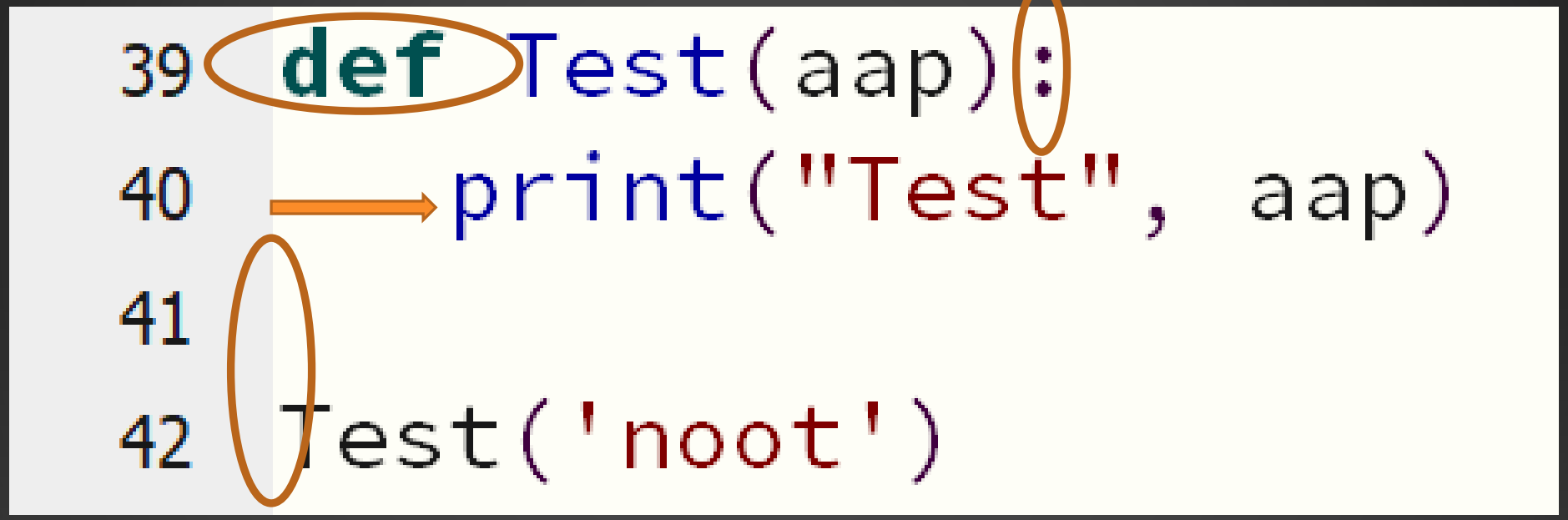
```
1  from MyQueen import *
2
3  while True :
4      Mq.RGB(1, 1)
5      sleep(500)
6      Mq.RGB(0, 0)
7      sleep(500)
```

RGB I2C CODE

```
Mq.RGB(1, 1)
```

```
89      # ---------------------------------------------------
90      # RGB - Maqueen Plus - set RGB leds (color range 1~7)
91      def RGB(self, colourL, colourR):
92          buf = bytearray(3)
93          buf[0] = 0x0b
94          buf[1] = colourL
95          buf[2] = colourR
96          i2c.write(self.I2caddr, buf)
```

FUNCTIE MAKEN
{ ZONDER ACCOLADES }



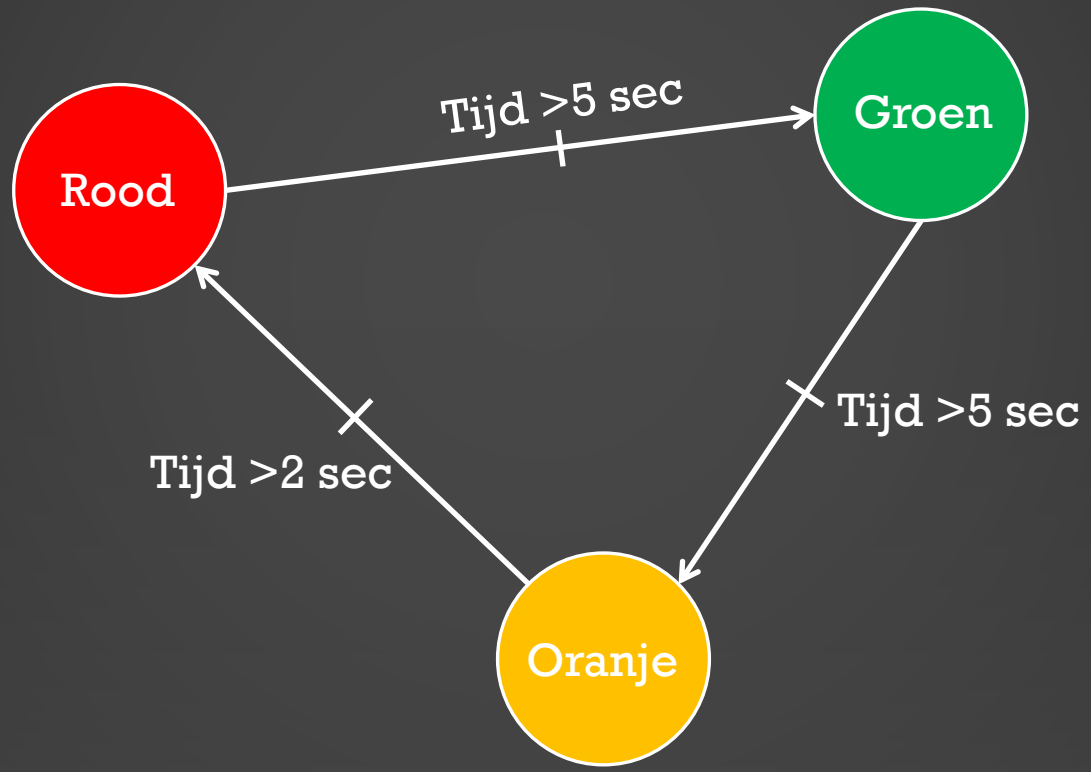
VERKEERSLICHT - STATES

state {znw.}

staat · situatie · stand · toestand · gesteldheid · constellatie

```
1  from MyQueen import *
2
3  while True :
4
5      print('StateRood')
6      Mq.RGB(1, 1)
7      time.sleep_ms(5000)
8
9      print('StateGroen')
10     Mq.RGB(2, 2)
11     time.sleep_ms(5000)
12
13     print('StateOranje')
14     Mq.RGB(3, 3)
15     time.sleep_ms(2000)
```

VERKEERSLICHT (TIJD)



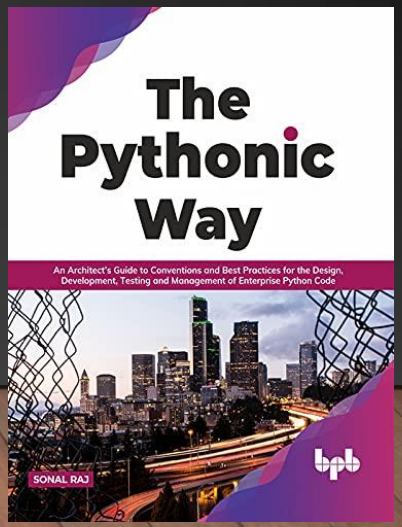
CAR VS PYTHON ROBOTICS

- States hebben nummer
- Toestand van de hele statemachine in één variabele
- Iedere state is een ‘case’ in een ‘switch’ statement
- Alle states van een statemachine in één functie.

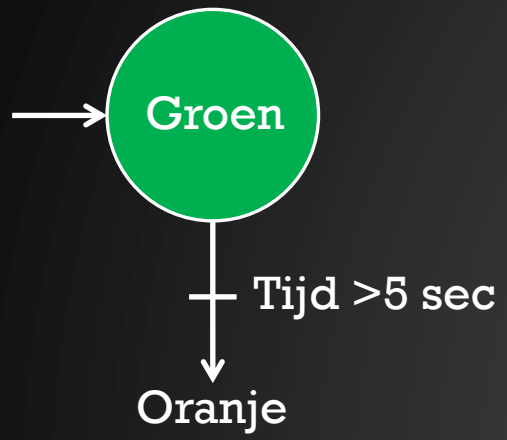
➢ Alle states bij elkaar

- States hebben naam
- StateMachine class vervangt deze variabele en het switch-statement.
- Iedere state is een functie

➢ States zijn functies die herbruikbaar zijn in andere statemachines



VERKEERSLICHT – STATEFUNCTIE



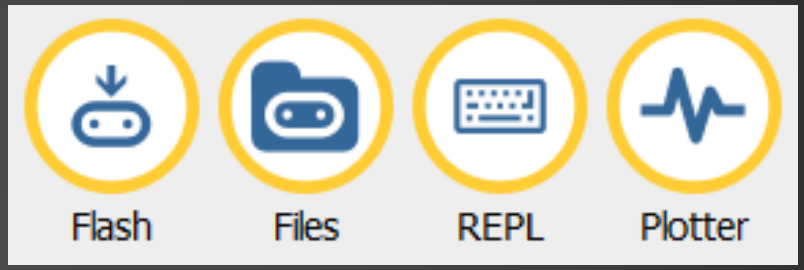

```
8  def StateGroen(S):
9      if S.IsNewState('StateGroen') :
10         Mq.RGB(2, 2)
11
12     if S.StateTime(5000) :
13         S.Goto(StateOranje)
```

VERKEERSLICHT – STATEMACHINE

```
30  Sm = StateMachine()
31
32  Sm.Goto(StateGroen)
33
34  while Sm.IsDone() == False:
35      Sm.Takt()
```

OEFENING - I

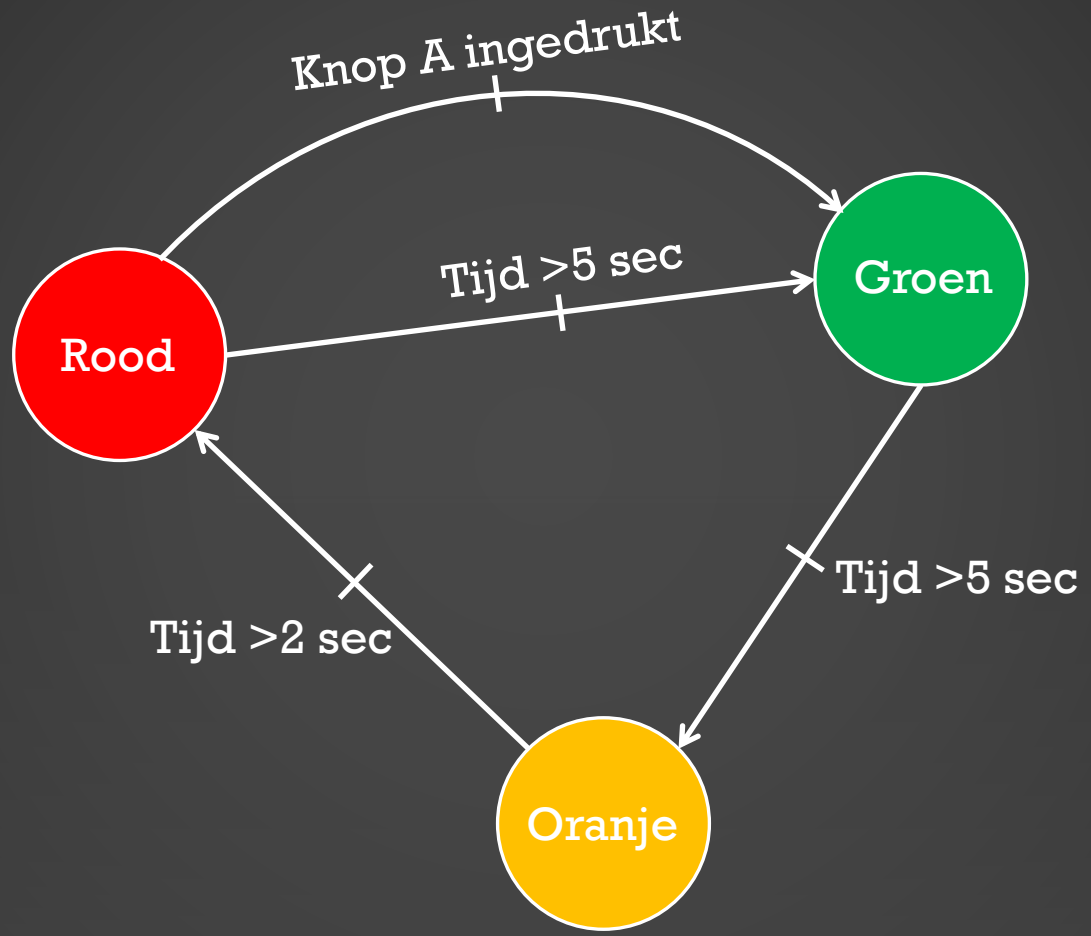
- Open “2.Verkeerslicht.py” in mu
- Flash
- Kijk naar output met REPL




```
30  Sm = StateMachine()
31
32  Sm.Goto(StateGroen)
33
34  while Sm.IsDone() == False:
35      Sm.Takt()
```

```
 8  def StateGroen(S):
 9      if S.IsNewState('StateGroen') :
10          Mq.RGB(2, 2)
11
12      if S.StateTime(5000) :
13          S.Goto(StateOranje)
```

OEFENING - II




```
34      if pin5.read_digital() == False:
```

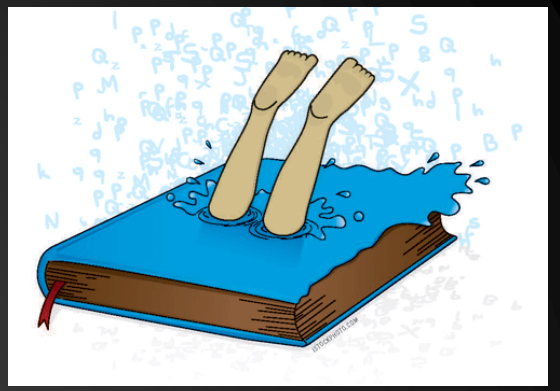
```
3  # ------------------------------------
4  def StateRood(S):
5      if S.IsNewState('StateRood') :
6          Mq.RGB(1, 1)
7
8      if S.StateTime(5000) :
9          S.Goto(StateGroen)
10
11     if pin5.read_digital() == False:
12         S.Goto(StateGroen)
13
```

STATEMACHINE API

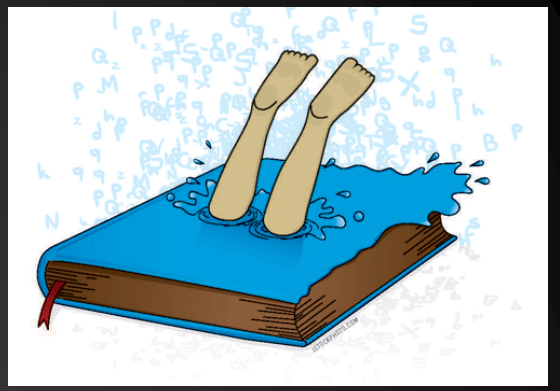
- Reset() Klaar voor een schone start
- Goto(NextState) Ga naar state of states
- Return() Einde van (serie van) states
- IsNewState(StateName) Geeft True terug bij eerste aanroep
- StateTime(Delay) Geeft True terug als we Delay miliseconden in deze state zijn.
- IsDone() Geeft True terug als alle states zijn afgehandeld (zie return).
- Takt() Voer actieve state uit

```
28  # ------------------------------------
29  def SequenceWachtA(S):
30      if S.IsNewState('SequenceWachtA') :
31          Mq.RGB(7, 7)
32          Mq.Stop()
33
34      if pin5.read_digital() == False:
35          S.Goto(Sequence1s)
36
37  # ------------------------------------
38  def Sequence1s(S):
39      if S.IsNewState('Sequence1s') :
40          Mq.RGB(5, 5)
41
42      if S.StateTime(1000) :
43          S.Return() # einde van s
```

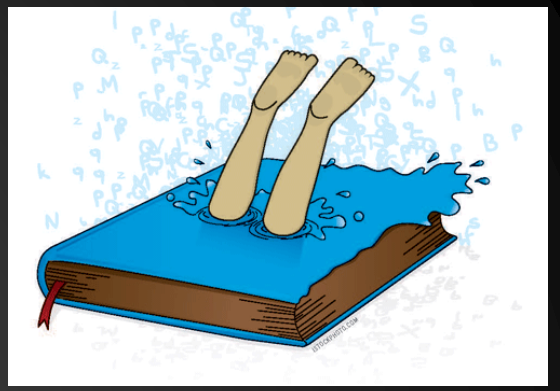
```
52  Sm.Goto(SequenceWachtA)
53  ...
54  Sm.Goto(StateGroen)
```



```
>>> a=[1, 2, 3]
>>>
>>> a
[1, 2, 3]
>>>
```

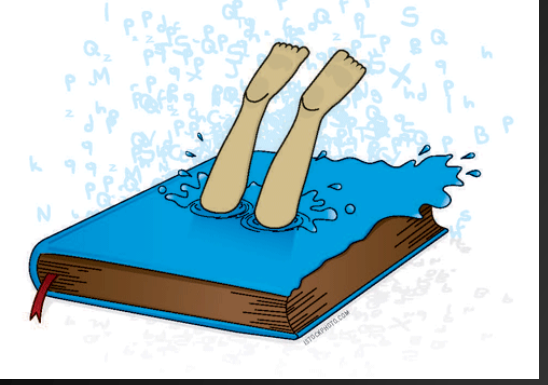


```
>>>
>>> a=[1, 2, 3]
>>>
>>> a
[1, 2, 3]
>>>
>>> a.append(StateGroen)
>>> a
[1, 2, 3, <function StateGroen at 0x20005820>]
>>>
>>>
```

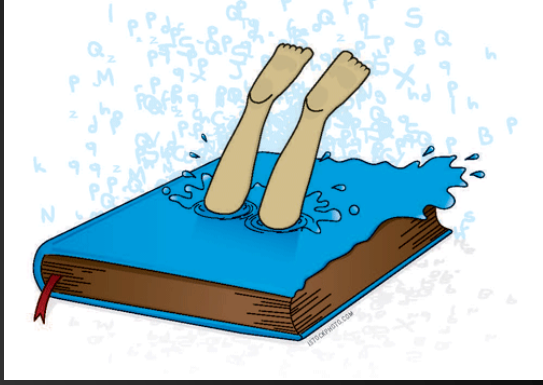


```
>>>
>>> b=[SequenceWachtA, StateGroen]
>>>
>>> b
[<function SequenceWachtA at 0x20005d90>, <function StateGroen at
0x20005d70>]
>>>
```

```
49
50  Sm.Goto([SequenceWachtA, StateGroen])
51
```

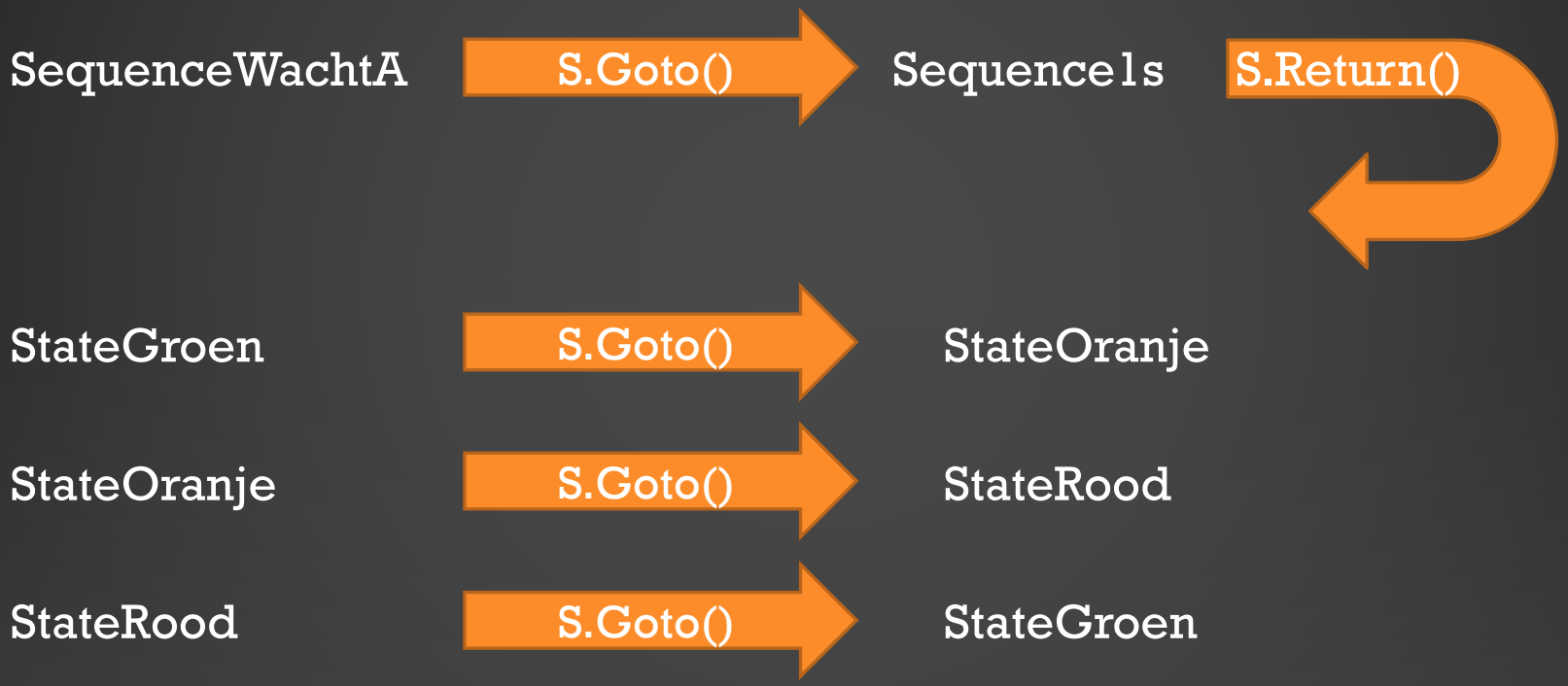


DEEP DIVE



```
22      # ------------------------------------------------------------------
23      # Goto - Execute NextState (leave current state)
24      def Goto(self, NextState):
25          print("Goto ", end='')
26          self.Return(True) # remove current active state (if any) without message
27
28          # NextState can be a function or list of functions
29          if isinstance(NextState, list) :
30              for S in reversed(NextState) :
31                  self._States.append(S)          # add new items in reverse order
32          else :
33              self._States.append(NextState)    # add new item
34
```

Sm.Goto([SequenceWachtA, StateGroen])



HUISWERK

- Maak een statemachine voor een ‘Duits’ verkeerslicht

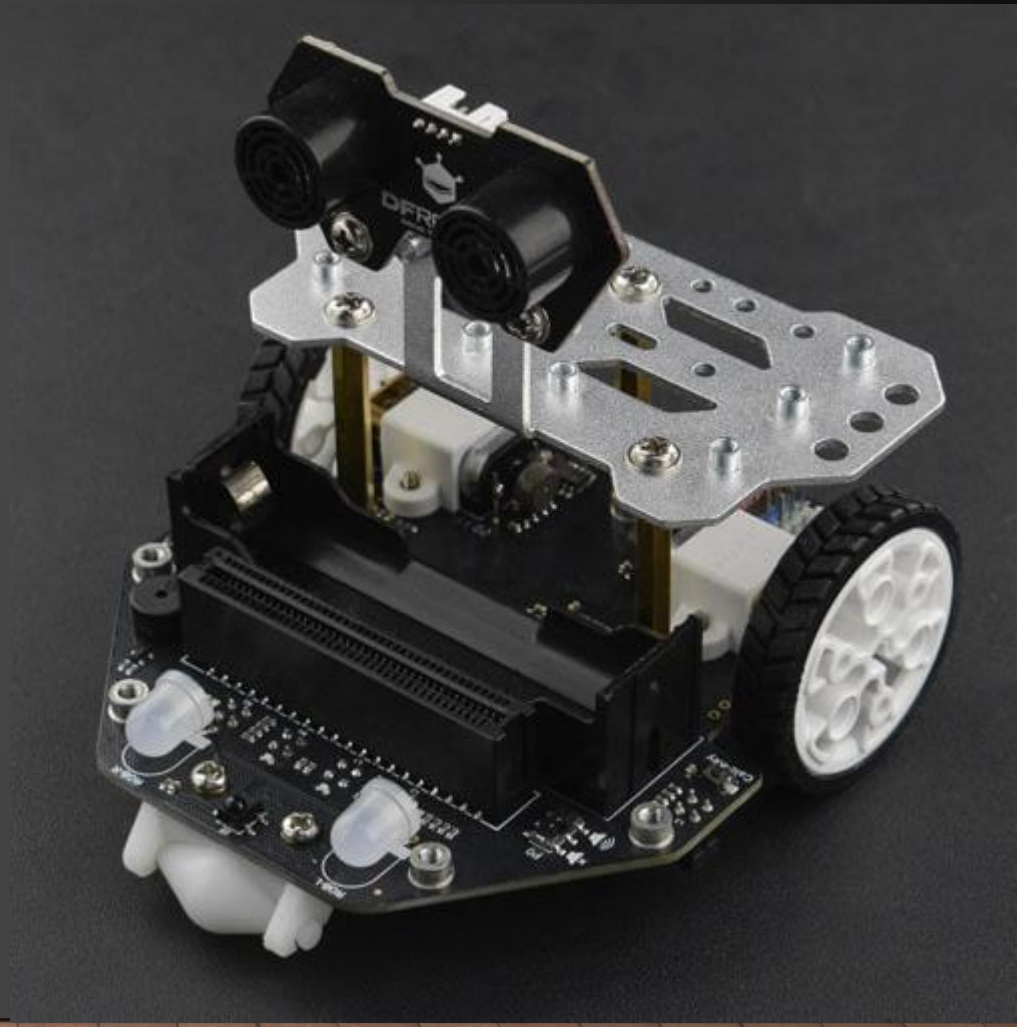
- Pas het verkeerslicht aan zodat je met knop A kunt schakelen naar ‘oranje knipperen’ en met knop B terug naar normaal gebruik.

Bonuspunten:

- De RGB leds hebben 8 kleur-combinaties. Maak een statemachine die deze om de beurt toont. Ga naar de volgende kleur als je op knop A drukt.

VOORBEREIDING WORKSHOP 3

- Plateau & ultrasoon sensor monteren
- Accu plaatsen & laden
- Kruiskop schroevendraaier meenemen



MEER UITDAGING?

```
1  from MyQueen import *
```