```
 1 # plasma_rgb.py - last steps in the
   implementation
 2 # of oldskool plasma effect
 3 # see http://www.bidouille.org/prog/
   plasma
 4 #
 5 # The plasma is basically a function on
   2D space created
 6 # by adding together a few sinusoids.
 7 #
 8 # By combining different types of sines
   and adding
 9 # a time component the illusion of
   motion is achieved.
10 #
11 # 2017_0122 PePo new for Open Dag WF,
   using default library neopixel
12 #
13 # Sources: Youtube https://www.youtube.
   com/watch?v=QcyuYvyvOEI&index=14&list=
   PLuuAy8GJr5z1WoOJAFh1adr_yjCMJQ2Yl
14 # Tony Dicola source: https://gist.
   github.com/tdicola/
   6fe1fbc173dcd49de3a95be5fd9594f6
15
16 # ##################################### 
   ###############
17 # setup machine configuration
18 import machine
19 import math
20 import neopixel
21 import time
22
23 # LED matrix: 8 * 8 pixels
24 PIXEL_WIDTH = 8
25 PIXEL_HEIGHT = 8
26 MAX_BRIGHT = 50.0 #100.0
27
```

```
28 np = neopixel.NeoPixel(machine.Pin(13),
   PIXEL_WIDTH*PIXEL_HEIGHT)
29
30 # Clear all the pixels and turn them off
   .
31 np.fill((0,0,0))
32 np.write()
33
34 # ##################################### 
   ###############
35 # stap 7: add color ....
36 # To preserve the organic, fluid look of
    the plasma,
37 # the color scheme should not have
   discontinuities.
38 # However after adding our sines
   together,
39 # the plasma value is not necessarily
   constrained
40 # in a nice known interval like [0, 1].
41 # An easy way to solve this problem is
   to take the
42 # sinus again of the value we obtained
   at the end
43 # of our plasma function, and use it to
   create the
44 # RGB components of the color.
45 # In the examples below r, g and b are
   the red, green and
46 # blue components of the color, with -1
   being the lowest
47 # intensity (black), and 1 the highest (
   fully saturated
48 # color component).
49 # r = sin(v*pi)
50 # g = sin(v*pi + 2*pi/3)
51 # b = sin(v*pi + 4*pi/3)
52 def stap7():
```

```
53      while True:
54          np.fill((0, 0, 0))  # start met alle leds uit...
55          current = time.ticks_ms() / 1000.0  # tijd in seconden
56          for x in range(PIXEL_WIDTH):  # voor alle pixels langs x-as
57              for y in range(PIXEL_HEIGHT):  # en alle pixels langs de y-as
58                  v = 0.0
59                  v += math.sin(x + current)
60                  v += math.sin(1.0 * (x * math.sin(current / 0.5) + y * math.cos(current / 3.0)) + current)
61                  cx = x + 5.0 * math.sin(current / 5.0)
62                  cy = y + 3.0 * math.cos(current / 3.0)
63                  v += math.sin(math.sqrt((math.pow(cx, 2.0) + math.pow(cy, 2.0)) + 1.0) + current)
64                  v = (v + 3.0) / 6.0  # v in range: 0..1
65                  # calculate color
66                  # let op: r,g,b hebben negatieve waarden! -> zeer heldere intensiteit (255)
67                  # schaling of math.fabs() maakt er een positief getal van.
68                  # r = math.fabs(math.sin(v * math.pi))
69                  # g = math.fabs(math.sin(v * math.pi + 2.0 * math.pi / 3.0))
70                  # b = math.fabs(math.sin(v * math.pi + 4.0 * math.pi / 3.0))
71                  r = math.sin(v * math.pi)
```

```
72                  r = (r + 1.0) / 2.0  # r in range [0..1]
73                  g = math.sin(v * math.pi + 2.0 * math.pi / 3.0)
74                  g = (g + 1.0) / 2.0  # g in range [0..1]
75                  b = math.sin(v * math.pi + 4.0 * math.pi / 3.0)
76                  b = (b + 1.0) / 2.0  # b in range [0..1]
77                  np[y * PIXEL_WIDTH + x] = (int(MAX_BRIGHT * r),
78
        int(MAX_BRIGHT * g),
79
        int(MAX_BRIGHT * b))
80          np.write()
81 stap7()
82
83 # ########################################################
84 # stap 8: add color  r = g = b = sin(v*5*pi)
85 # B/W pattern
86 def stap8():
87     while True:
88         np.fill((0, 0, 0))  # start met alle leds uit...
89         current = time.ticks_ms() / 1000.0  # tijd in seconden
90         for x in range(PIXEL_WIDTH):  # voor alle pixels langs x-as
91             for y in range(PIXEL_HEIGHT):  # en alle pixels langs de y-as
92                 v = 0.0 # start met 0
93                 v += math.sin(x + current)
94                 v += math.sin(1.0 * (x
```

```
 94 * math.sin(current / 0.5) + y * math.
    cos(current / 3.0)) + current)
 95                     cx = x + 5.0 * math.sin
    (current / 5.0)
 96                     cy = y + 3.0 * math.cos
    (current / 3.0)
 97                     v += math.sin(math.sqrt
    ((math.pow(cx, 2.0) + math.pow(cy, 2.0)
    ) + 1.0) + current)
 98                     v = (v + 3.0) / 6.0  #
    v in range: 0..1
 99                     # color
100                     r = math.sin(v * 5.0 *
    math.pi)
101                     r = (r + 1.0) / 2.0 # r
     in range 0..1
102                     g = b = r
103                     np[y * PIXEL_WIDTH + x]
     = (int(MAX_BRIGHT * r),
104
        int(MAX_BRIGHT * g),
105
        int(MAX_BRIGHT * b))
106             np.write()
107 #stap8()
108
```