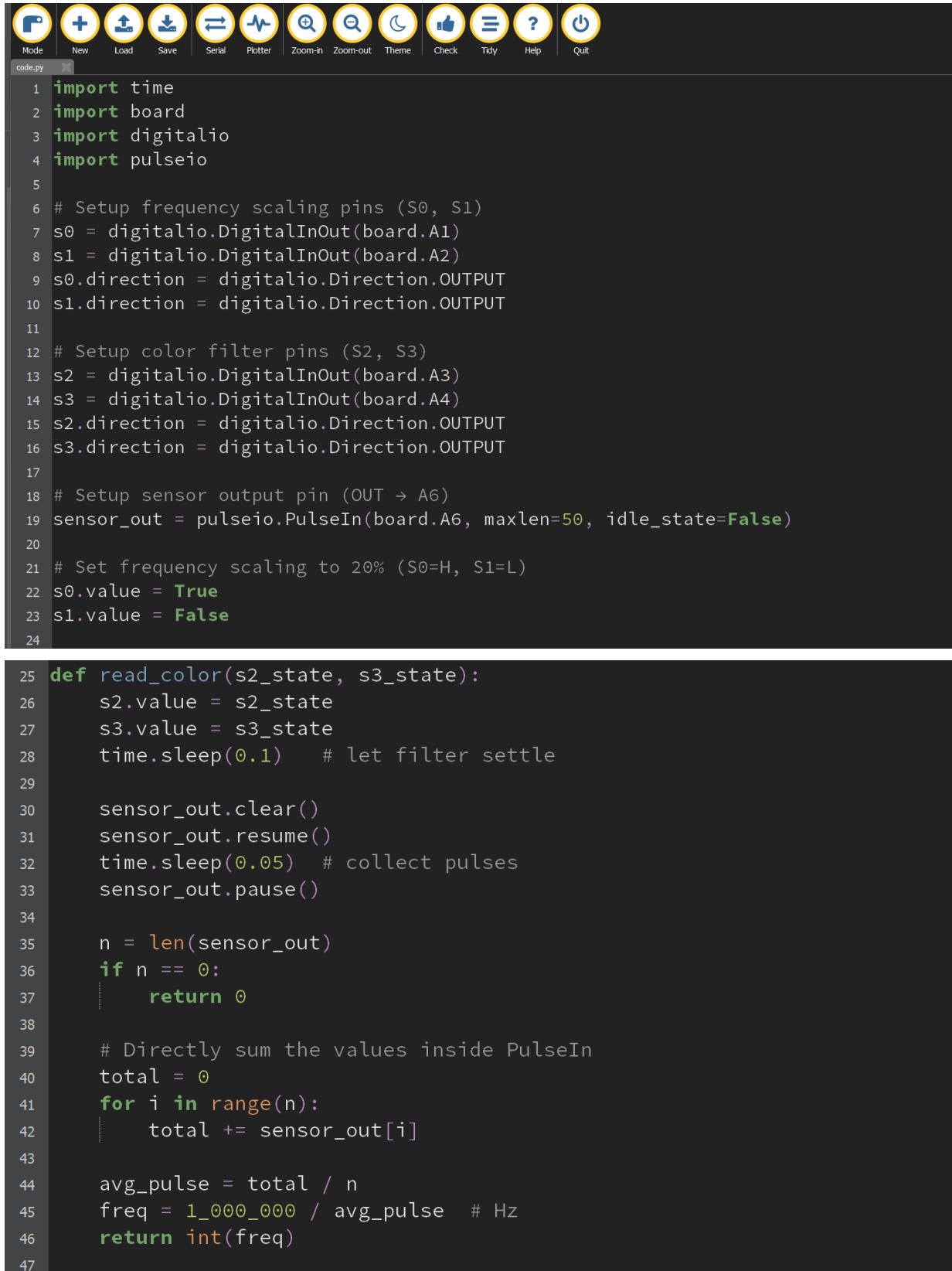


Coding part



The image shows a screenshot of a MicroPython code editor. At the top is a toolbar with various icons: Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. Below the toolbar is a code editor window with the file name "code.py" in the title bar. The code editor contains the following Python code:

```
1 import time
2 import board
3 import digitalio
4 import pulseio
5
6 # Setup frequency scaling pins (S0, S1)
7 s0 = digitalio.DigitalInOut(board.A1)
8 s1 = digitalio.DigitalInOut(board.A2)
9 s0.direction = digitalio.Direction.OUTPUT
10 s1.direction = digitalio.Direction.OUTPUT
11
12 # Setup color filter pins (S2, S3)
13 s2 = digitalio.DigitalInOut(board.A3)
14 s3 = digitalio.DigitalInOut(board.A4)
15 s2.direction = digitalio.Direction.OUTPUT
16 s3.direction = digitalio.Direction.OUTPUT
17
18 # Setup sensor output pin (OUT → A6)
19 sensor_out = pulseio.PulseIn(board.A6, maxlen=50, idle_state=False)
20
21 # Set frequency scaling to 20% (S0=H, S1=L)
22 s0.value = True
23 s1.value = False
24
25 def read_color(s2_state, s3_state):
26     s2.value = s2_state
27     s3.value = s3_state
28     time.sleep(0.1)    # let filter settle
29
30     sensor_out.clear()
31     sensor_out.resume()
32     time.sleep(0.05)  # collect pulses
33     sensor_out.pause()
34
35     n = len(sensor_out)
36     if n == 0:
37         return 0
38
39     # Directly sum the values inside PulseIn
40     total = 0
41     for i in range(n):
42         total += sensor_out[i]
43
44     avg_pulse = total / n
45     freq = 1_000_000 / avg_pulse  # Hz
46     return int(freq)
47
```

```

48 def classify_color(r, g, b, c):
49     total = r + g + b
50     if total == 0:
51         return "None"
52
53     rn, gn, bn = r / total, g / total, b / total
54
55     # Debug print
56     print("Norm → R:", round(rn, 2), "G:", round(gn, 2), "B:", round(bn, 2))
57
58     if rn > 0.40 and rn > gn + 0.10 and rn > bn + 0.10:
59         return "Red"
60     elif gn > 0.35 and gn > rn + 0.05 and gn > bn + 0.05:
61         return "Green"
62     elif bn > 0.35 and bn > rn + 0.05 and bn > gn + 0.05:
63         return "Blue"
64     elif abs(rn - gn) < 0.05 and abs(rn - bn) < 0.05:
65         return "White"
66     else:
67         return "Unknown"
68
69 while True:
70     red    = read_color(False, False)    # Red
71     blue   = read_color(False, True)     # Blue

```

```

green = read_color(True, True)      # Green
clear = read_color(True, False)     # Clear

color = classify_color(red, green, blue, clear)
print("R:", red, "G:", green, "B:", blue, "C:", clear, "→", color)
time.sleep(0.5)

```

You can adjust the value of checking condition and modify and see the output.