Code explanation

import library whatever u need

Setup frequency scaling pins (S0, S1)

s0 = digitalio.DigitalInOut(board.A1)

s1 = digitalio.DigitalInOut(board.A2)

s0.direction = digitalio.Direction.OUTPUT

s1.direction = digitalio.Direction.OUTPUT

👉 These are like “volume knobs.” S0 and S1 set how strong the sensor signal is (here we will choose 20%).

Setup color filter pins (S2, S3)

s2 = digitalio.DigitalInOut(board.A3)

s3 = digitalio.DigitalInOut(board.A4)

s2.direction = digitalio.Direction.OUTPUT

s3.direction = digitalio.Direction.OUTPUT

👉 These two pins are like glasses 🕶️. They decide which color filter (Red, Green, Blue, or Clear) the sensor looks through.

Setup sensor output pin (OUT → A6)

sensor_out = pulseio.PulseIn(board.A6, maxlen=50, idle_state=False)

👉 The OUT pin from the sensor is connected to A6.

👉 PulseIn counts how fast the light pulses are coming (that tells us the color strength).

Set frequency scaling to 20% (S0=H, S1=L)

s0.value = True

```
s1.value = False
```

👉 We set the sensor to send fewer signals (20%). This keeps the data small and easy to read.

```
def read_color(s2_state, s3_state):
    s2.value = s2_state
    s3.value = s3_state
    time.sleep(0.1)   # let filter settle
```

👉 Choose which color filter to use (Red, Green, Blue, or Clear).

👉 Wait a moment so the sensor can adjust.

```
    sensor_out.clear()
    sensor_out.resume()
    time.sleep(0.05)  # collect pulses
    sensor_out.pause()
```

👉 Start counting pulses from the sensor for a short time.

👉 Then stop and keep the data.

```
    n = len(sensor_out)
    if n == 0:
        return 0
```

👉 If no pulses came, return 0 (no color detected).

```
    # Directly sum the values inside PulseIn
    total = 0
    for i in range(n):
```

```
        total += sensor_out[i]


    avg_pulse = total / n
    freq = 1_000_000 / avg_pulse  # Hz
    return int(freq)
```

👉 Add all the pulse times together, find the average.

👉 Convert this into frequency (how fast the pulses are).

👉 Higher frequency = stronger color.

```
def classify_color(r, g, b, c):
    total = r + g + b
    if total == 0:
        return "None"
```

👉 Add Red, Green, Blue values. If all are 0, no color is seen.

```
    rn, gn, bn = r / total, g / total, b / total
```

👉 Normalize: we turn the values into percentages of total.

👉 Example: if R=500, G=250, B=250 → R=0.5, G=0.25, B=0.25.

```
    # Debug print
    print("Norm → R:", round(rn, 2), "G:", round(gn, 2), "B:", round(bn, 2))


    if rn > 0.40 and rn > gn + 0.10 and rn > bn + 0.10:
        return "Red"
    elif gn > 0.35 and gn > rn + 0.05 and gn > bn + 0.05:
        return "Green"
```

```
    elif bn > 0.35 and bn > rn + 0.05 and bn > gn + 0.05:
        return "Blue"
    elif abs(rn - gn) < 0.05 and abs(rn - bn) < 0.05:
        return "White"
    else:
        return "Unknown"
```

👉 These rules decide the color name:

- If Red is much bigger → "Red"
- If Green is much bigger → "Green"
- If Blue is much bigger → "Blue"
- If all nearly equal → "White"
- Otherwise → "Unknown"

```
while True:
    red   = read_color(False, False)   # Red
    blue  = read_color(False, True)    # Blue
    green = read_color(True, True)     # Green
    clear = read_color(True, False)    # Clear
```

👉 Take readings with different filter settings:

- (False, False) → Red filter
- (False, True) → Blue filter
- (True, True) → Green filter
- (True, False) → Clear (no filter)

```
    color = classify_color(red, green, blue, clear)
```

👉 Use our rules to decide which color it is.

```
print("R:", red, "G:", green, "B:", blue, "C:", clear, "→", color)
```

👉 Show the numbers and the final color name.

```
time.sleep(0.5)
```

👉 Wait half a second before reading again.