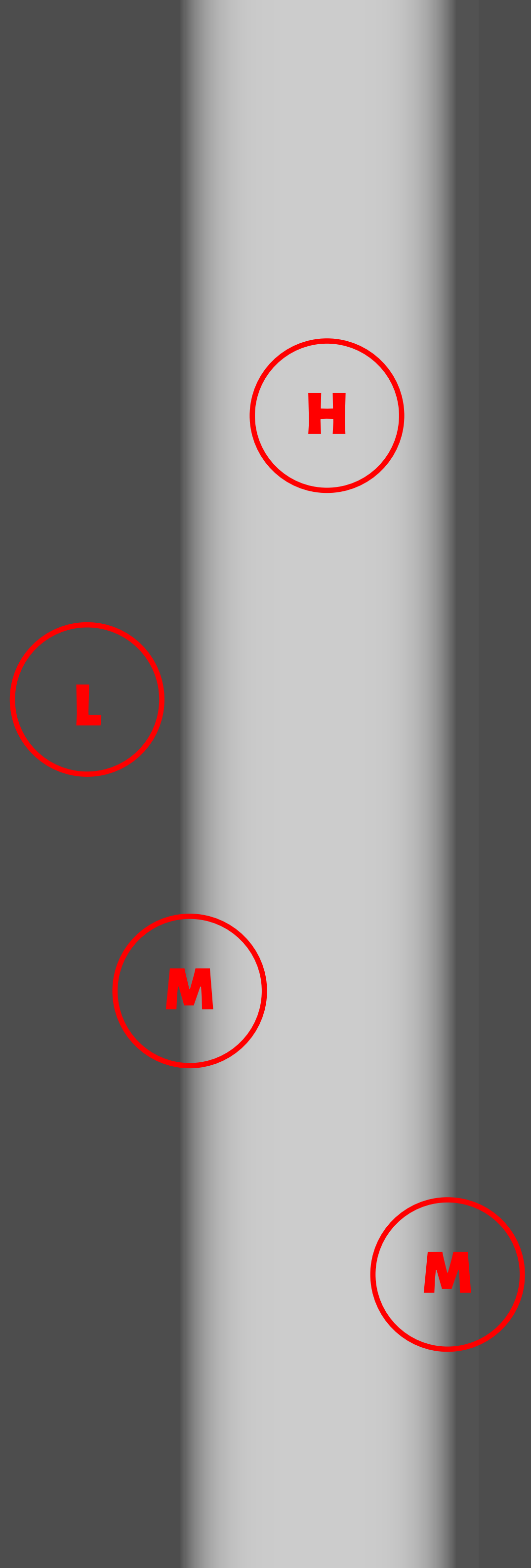


The colour sensor can shine a light at a surface and measure the amount of light reflected back at the sensor. By pointing the sensor downward, we can use it to follow a line.

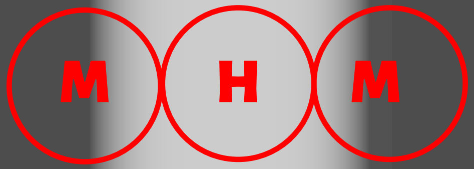
In this first example, the colour sensor is over a light-coloured region, so the level of reflected light will be relatively high. Numbers are returned as a percentage, so a range from 0-100.

In this second example, the sensor is over a dark region, so the level of reflected light will be much lower. The actual values will depend on the level of ambient light and the precise nature of the surface, so exact values are meaningless in this example. What is important is the relationship between the numbers—high versus low.

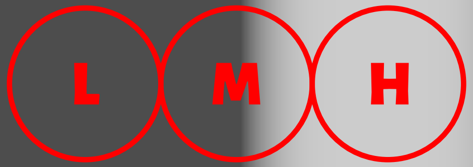
In this third example, the sensor is on the edge of the light region, so part is light and part is dark. The sensor receives just one value for the whole area sampled, so we would expect that number to be between the values for the dark region and the light region.

Here is the sample on the other side. As the balance of light and dark is the same as before, we expect the reflected light levels to be about the same as before.

So how can we use this information?

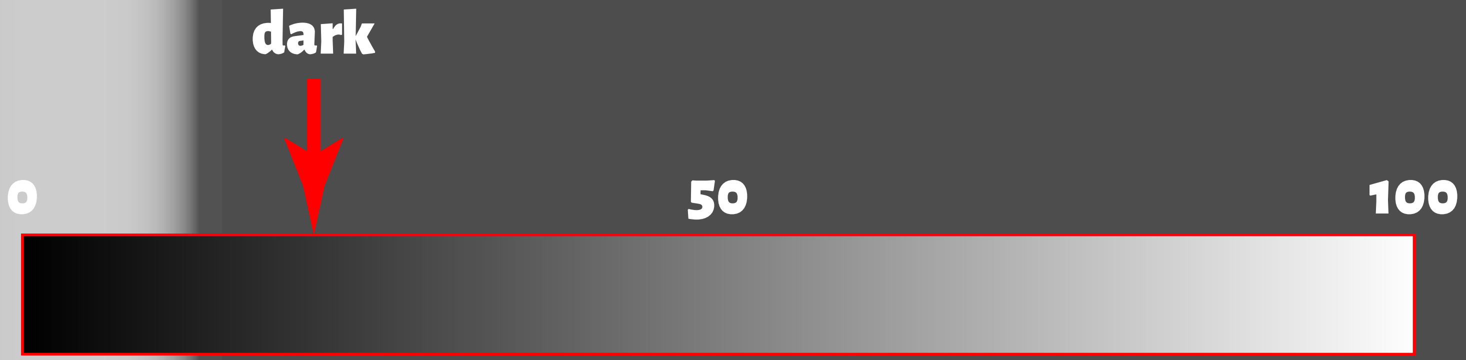


If we try to keep the robot's sensor in the middle of the line,
we are aiming to keep the reflected light level as high as we can.
We will be able to tell when the robot is drifting, as the reflected value
will gradually fall as the sensor drifts off the line. But, we won't be able to
tell which way we're drifting. We can only know that we *are* drifting.

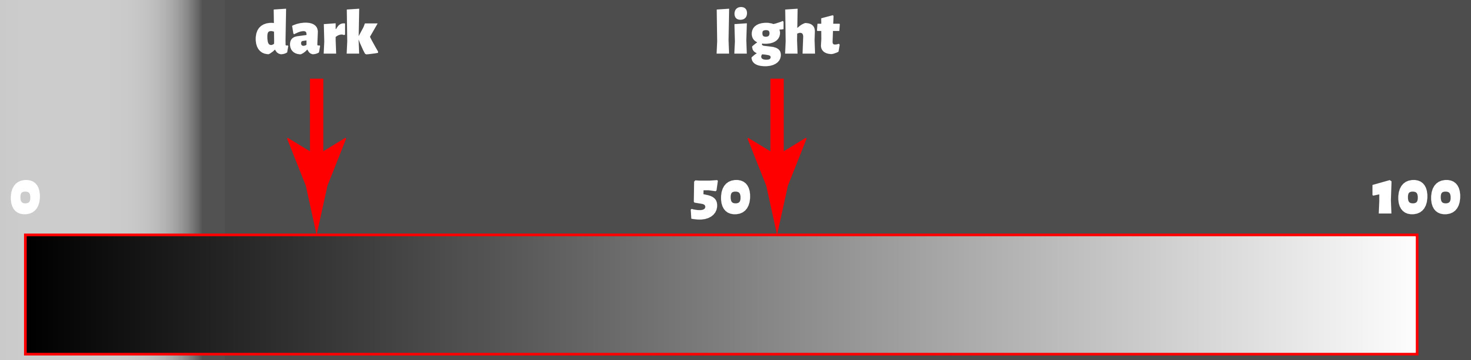


But if we aim to keep the robot's sensor on the left edge of the line,
we can make it work. The ideal value will now be “grey”, somewhere between
the light and dark values (we could average those two to figure out what
the middle value would be. If the value rises too far, we know we're drifting on
to the line, and need to veer left. If the value falls too far, we know we're drifting
off the line, and need to veer right. We could also track the right edge instead,
and reverse the directions we veer to.

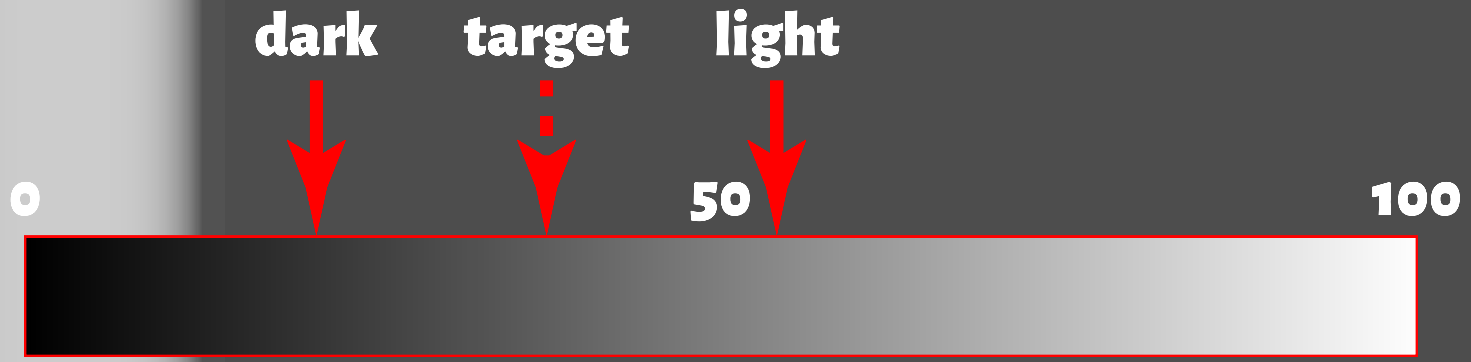
0
50
100



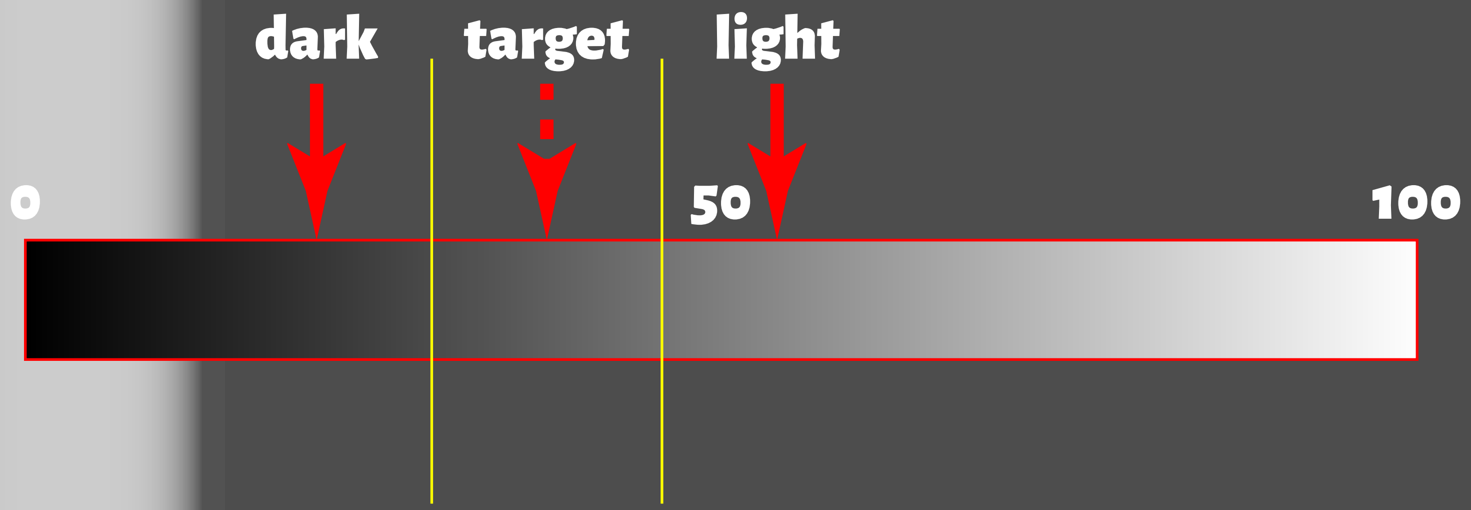
```
dark = 20
```



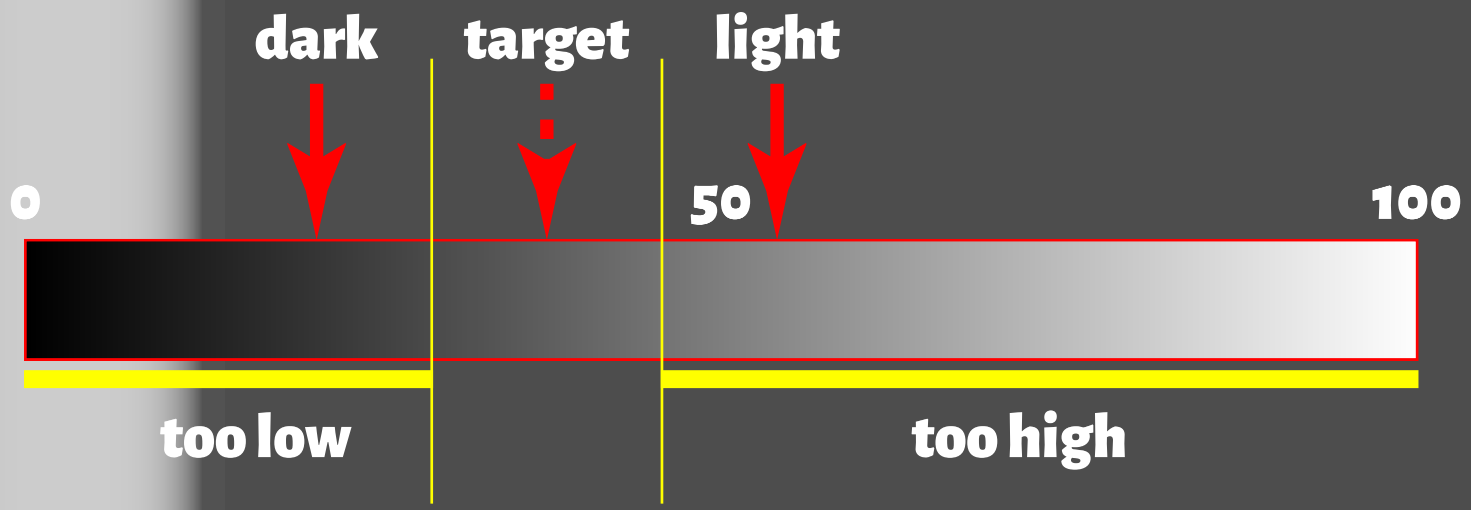
```
dark = 20
light = 56
```



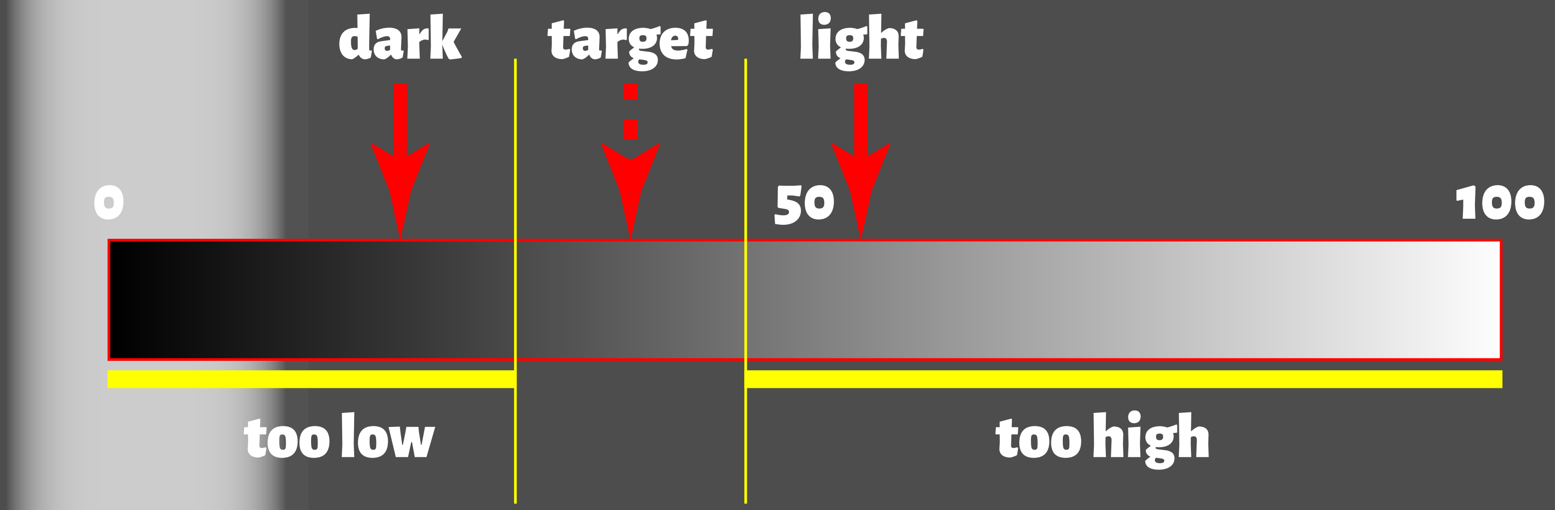
```
dark = 20
light = 56
target = (dark + light)/2
```



```
dark = 20
light = 56
target = (dark + light)/2
hiLimit = (light + target)/2
loLimit = (dark + target)/2
```



```
while True:
    if reflected light too low:
        veer right
    elif reflected light too high:
        veer left
    else:
        drive straight
```



```
while True:
    if reflected light less than loLimit:
        veer right
    elif reflected light greater than hiLimit:
        veer left
    else:
        drive straight
```