Autonomous bots!

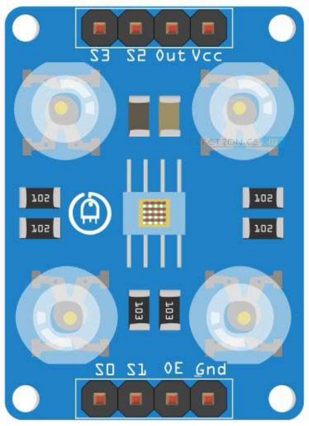
Challenge: **to program a robot so it can automatically complete a course in the shortest possible time. The robot will need to:**

- **Be able to sense the colour of the ground**
- **Search for a line to follow (the SEARCH state)**
- **Follow a line (the FOLLOWING state)**
- **Navigate around obstacles (the OBSTICLE AVOIDANCE state)**
- **Start and stop using a button**

Part 1: Introducing the colour sensor

We're going to give our bot the ability to sense colour. This will allow us to send commands to the motors based on the colour it detects and, ultimately, program it to follow a line.

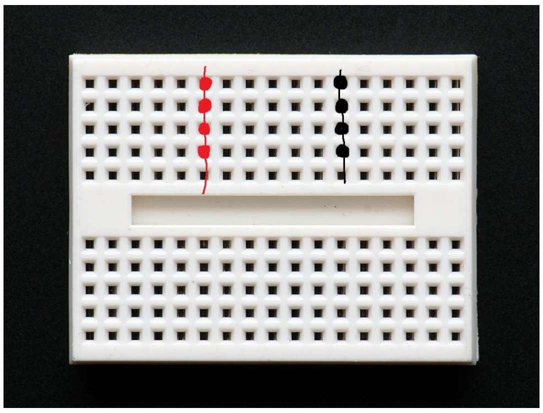
This is the TCS3200 colour sensor. There are four white LEDs which shine light onto the surface to be reflected back to the sensor in the centre. The sensor has an 8x8 array of photodiodes which convert light into electrical current. Some of these have a red filter and they can only see red light. Others have green, blue and clear filters. By measuring the electrical current generated by each of the photodiodes we can determine the colour of the reflected light in terms of red, green and blue (RGB) light.



Wiring the sensor

Using the BLUE wire, connect the OUT pin to GPIO 21
Using the ORANGE wire, connect the S2 pin to GPIO 20
Using the YELLOW wire, connect the S3 pin to GPIO 16

Using your small breadboard create a power rail from a 5v pin and a GND pin on your Pi. Note that the small boards don't have separate power rails so you must be pay attention to how the holes connect (else you'll fry the Pi by creating a short circuit).
From a 5V pin on the Pi, connect to the top row of a column on the board as below using a GREEN wire. The four pins underneath are now receiving 5V from the pi. Repeat this with the GND using a BLACK wire.



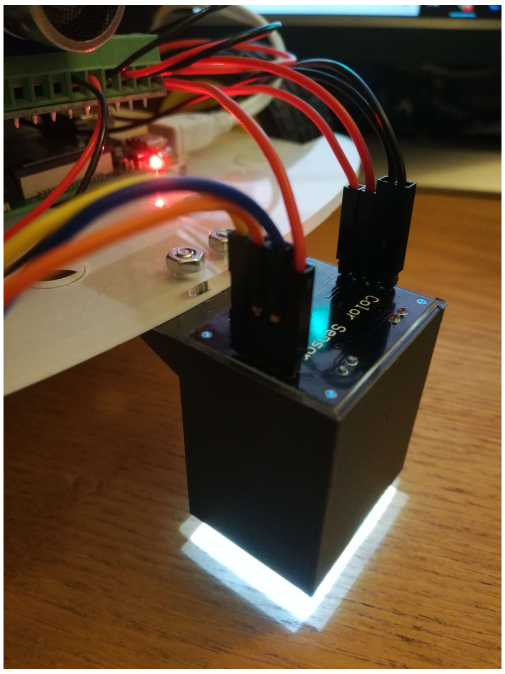
Using the GREEN wires, connect VCC S0 and S1 pins on the sensor to your 5V power rail
Using the BLACK wires, connect OE and GND pins on the sensor to your GND rail on the breadboard.

See how we're using the different colour wires for positive, negative and signals. This makes it easier to tell if we're hooking it up correctly, and makes it less likely we break something.

If everything went OK you should see the four LEDs light up on the sensor!

Installing the sensor

We need to mount the sensor on the front of our PiBot. Use the sensor holder and two of your spare nuts and bolts like so:



Setting up our Python project

To make things a bit easier I've set up a project on GitHub which we'll need to download onto the Pi.

Open up a terminal and run the following:

```
git clone https://github.com/mattstoneham/PiBot
```

Using File Explorer, check you now have a PiBot directory under home/pi/

Using the colour_sensor module to get the RGB values

Import modules
Start Python 3 (IDLE) and create a new file. Save as detect_colours.py in /home/pi/PiBot/project.
In order to get the RGB values we need to import the ColourSensor class from the colour_sensor module:

```
from PiBot.lib.colour_sensor import ColourSensor
```

You will also need to import time module

Define main function, and code to execute it
Define a main function, for now just add a debug print statement so we can test our modules are loading and the function is being called.

You will need to add the following two lines of code at the bottom of your script. This tells Python to call the main function when we run our code in the Python shell.

```
If __name__ == '__main__':
      main()
```

Test it's working!
Before we go any further, try executing your script (F5 key) to make sure that the modules are loading, your main function is being called and printing out your debug print.

Everything working? Great, let's continue!

Whoa there! Before we rush into this, let's think about what we want our code to do!
Our main function needs to do the following:

- Create an instance of the ColourSensor class which we use to return RGB values from the sensor
- In a While True loop:
  - call the appropriate function in the class instance to get the RGB values from the sensor
  - Store the returned data in a variable
  - Parse the data and print the red, green and blue values to screen
  - Use the time module to pause script execution for half a second

Write it in pseudocode first!
 Using comments, write your main code functionality in pseudocode. This allows you to focus on the main structure and logic of your code without having to worry about much about programming conventions or syntax. This means we can get our ideas down very quickly and solve some of the bigger problems without having to write and re-write lots of code as things change.

Hints:
Use the following code to instance the ColourSensor class:
```
myColourSensor = ColourSensor()
```

To get the RBG values your code should look something like:
```
RGBvalues = myColourSensor.???????????
```
To find what function to call, have a look at the code in the module we're importing (\lib\colour_sensor.py) and you should easily find out what function to call.

Questions and things to think about in your code:
What type of data does the function return? e.g list, string, set, dictionary

How do we work with the returned data type to print out the RGB values nicely to the screen?

How do we query just the RED value that is returned?

Testing, and fixing any bugs!
So now we should have our first version of the code written, and now it's time to test it out, will our PiBot be colour savvy or colour blind???

Extending the functionality of our code

So, now we're getting our RGB values from the sensor. Let's extend this code to do the following:

- Add some auto calibration, so that we store the darkest and brightest reflected total RGB values (this gives us a range: darkest →lightest that the sensor has seen)
- Display the current reading as a percentage of the range (so white paper should read close to 100%, black paper close to 0%)
- Based on the RGB values returned from the sensor, print out what colour we are seeing:
    - RED
    - GREEN
    - BLUE
    - WHITE
    - BLACK

Auto calibration and percentage display
To store the darkest and brightest reflected light value and display the percent we'll need some variables in our main function. We'll deliberately initialise them with out-of-range values:

minimum = 9999
maximum = 100
percentage = 0
total = 0

Add up the returned red, green and blue values from the sensor and store the total in the variable 'total'

Now, we need to make sure that our minimum variable contains the <u>darkest</u> reading the sensor has ever seen, and the maximum variable contains the <u>brightest</u> reading the sensor has ever seen.

Your code should look something like this:

```
if current total RGB reading is brighter that our our maximum variable:
      print something to say we've auto calibrated our max intensity
      update our maximum variable with the current total

if current total RGB reading is darker than our minimum variable:
      print something to say we've auto calibrated our min intensity
      update our minimum variable with the current total
```

Once we've got our min/max variables updated we can work out the percentage of the current brightness using the following code:

```
percentage = (100 / (maximum-minimum)) * (total - minimum)
print('Light percentage: {0}'.format(percentage))
```

Time to test and fix any bugs!
Execute your code.
Place the sensor on white paper and allow the code to auto calibrate, then do the same on black paper.
After calibration to our lightest and darkest surfaces, black paper should display a close to 0%, and white close to 100%
A mid-grey should give around 50%

Displaying the colour the sensor is seeing

Now, based on the RGB values the sensor is returning we want to print out 'RED' if it's red, BLUE if it's blue etc. etc.

Run your code and look at the RGB values printed out for the red, green, blue, white and black test cards. Take note of the approx values for each.

Add code to print out the appropriate colour, it should look something like this:

```
if ReturnedRedValue > number and ReturnedGreenValue < number and
ReturnedBlueValue < number:
      print('--RED--')
```

Figure out what the numbers should be, and implement code to print out GREEN, BLUE, WHITE and BLACK