RoboCup Rescue

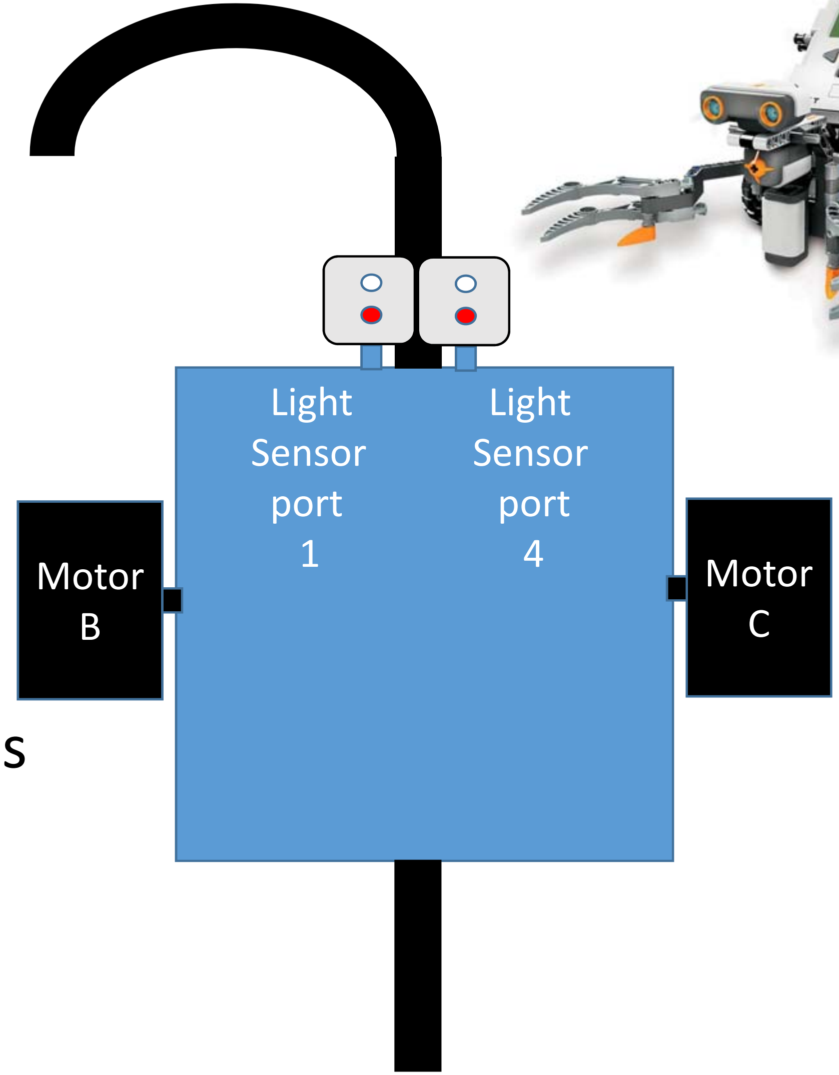
EV3 Workshop

Part 2

Rescue with RobotC

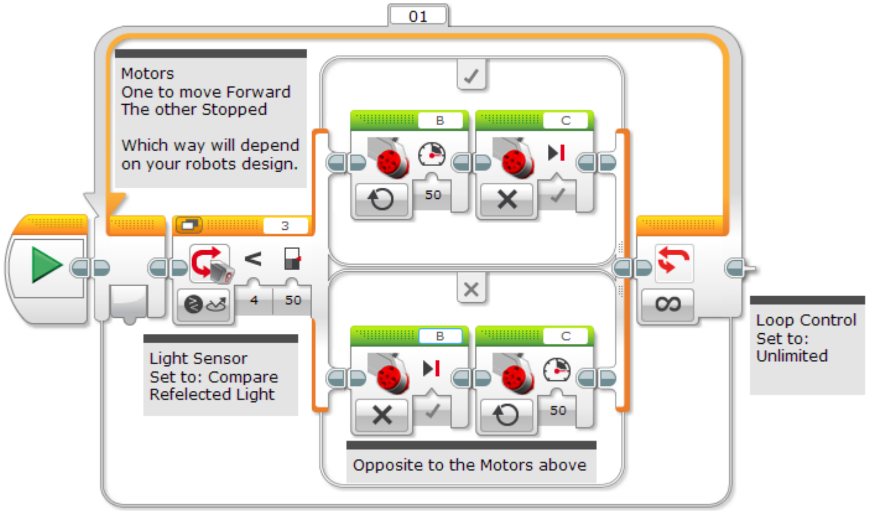
Robot Layout

- We will use a basic Rescue Robot layout for this workshop
- Light Sensors should be positioned to straddle the line or be just over the edge of the line.
  Plug into ports 1 and 4
- Motors should be plugged into ports B and C



Line Following

- Lets assume Rescue teams have some experience with programming the EV3 with the Lego EV3 software
- So lets transfer a simple Edge Following EV3 program to RobotC



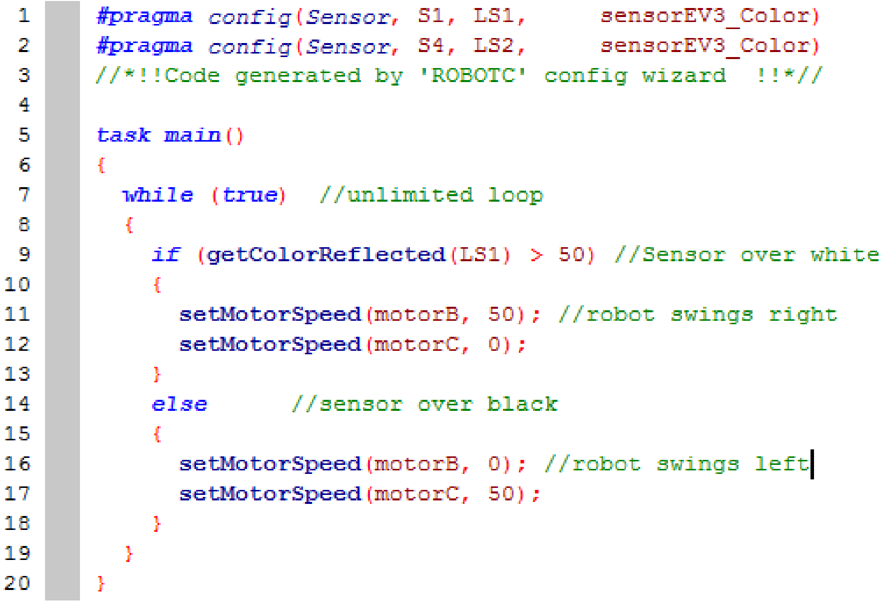
Motors
One to move Forward
The other Stopped

Which way will depend on your robots design.

Light Sensor
Set to: Compare
Refelected Light

Opposite to the Motors above

Loop Control
Set to:
Unlimited

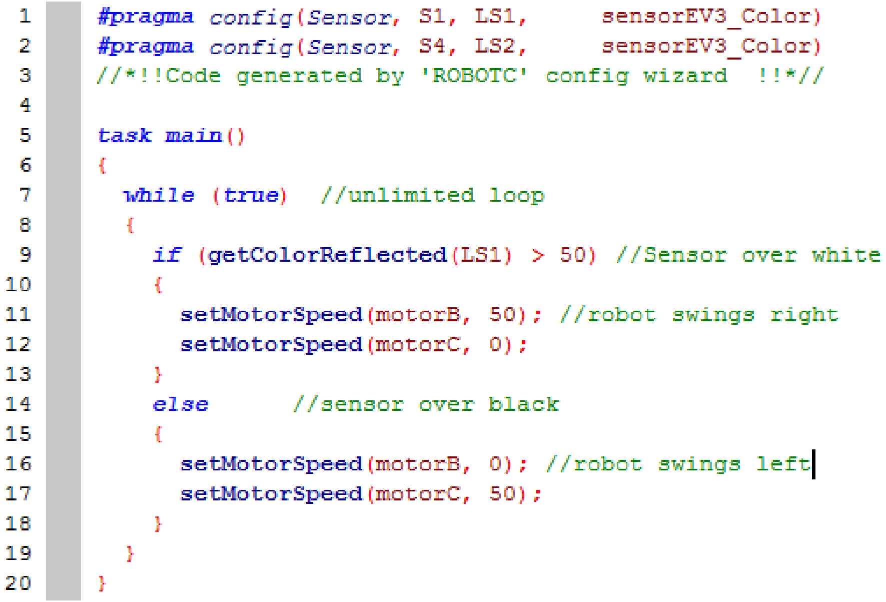


```
#pragma config(Sensor, S1, LS1,      sensorEV3_Color)
#pragma config(Sensor, S4, LS2,      sensorEV3_Color)
//*!!Code generated by 'ROBOTC' config wizard  !!*//

task main()
{
  while (true)  //unlimited loop
  {
    if (getColorReflected(LS1) > 50) //Sensor over white
    {
      setMotorSpeed(motorB, 50); //robot swings right
      setMotorSpeed(motorC, 0);
    }
    else      //sensor over black
    {
      setMotorSpeed(motorB, 0); //robot swings left
      setMotorSpeed(motorC, 50);
    }
  }
}
```

Left Edge Line Follower

© G. Tardiani 2015

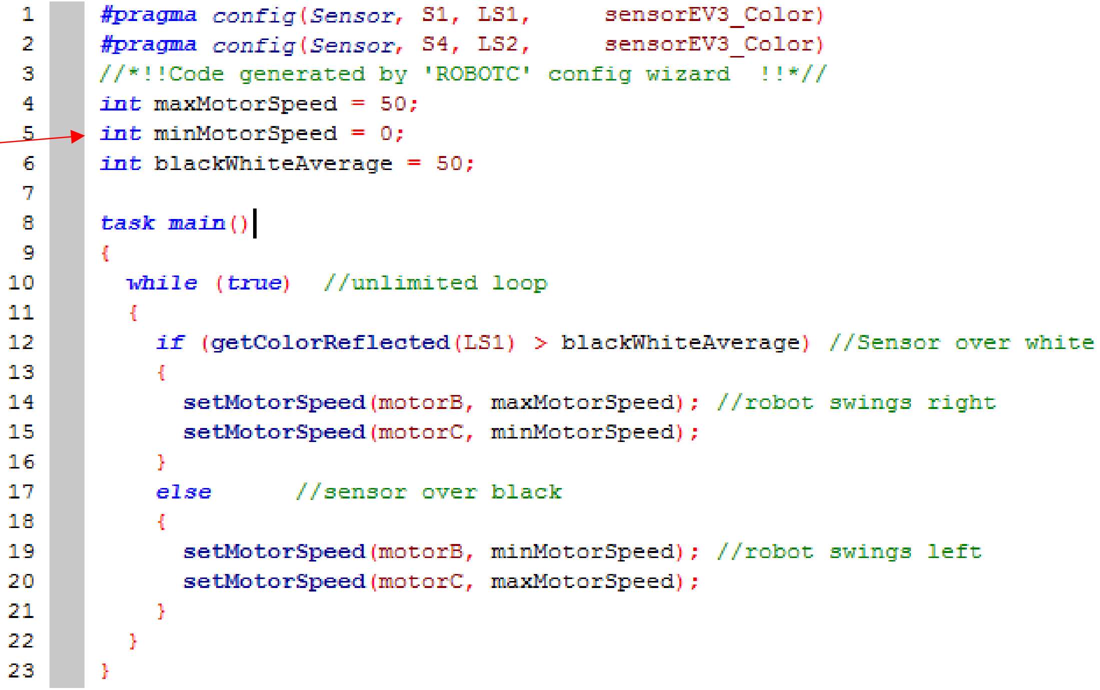
- There should be nothing new with this code
- I have used the motor & sensor setup to produce the first two lines of code as it allows me to define names LS1 and LS2 to sensors S1 and S4
- The while (true) control loops unlimited
- The IF…ELSE control allows the robot to decide whether to turn left or right depending on the value of the sensor
- Simple!



```
#pragma config(Sensor, S1, LS1,      sensorEV3_Color)
#pragma config(Sensor, S4, LS2,      sensorEV3_Color)
//*!!Code generated by 'ROBOTC' config wizard  !!*//

task main()
{
  while (true)  //unlimited loop
  {
    if (getColorReflected(LS1) > 50) //Sensor over white
    {
      setMotorSpeed(motorB, 50); //robot swings right
      setMotorSpeed(motorC, 0);
    }
    else      //sensor over black
    {
      setMotorSpeed(motorB, 0); //robot swings left
      setMotorSpeed(motorC, 50);
    }
  }
}
```

Using Variables

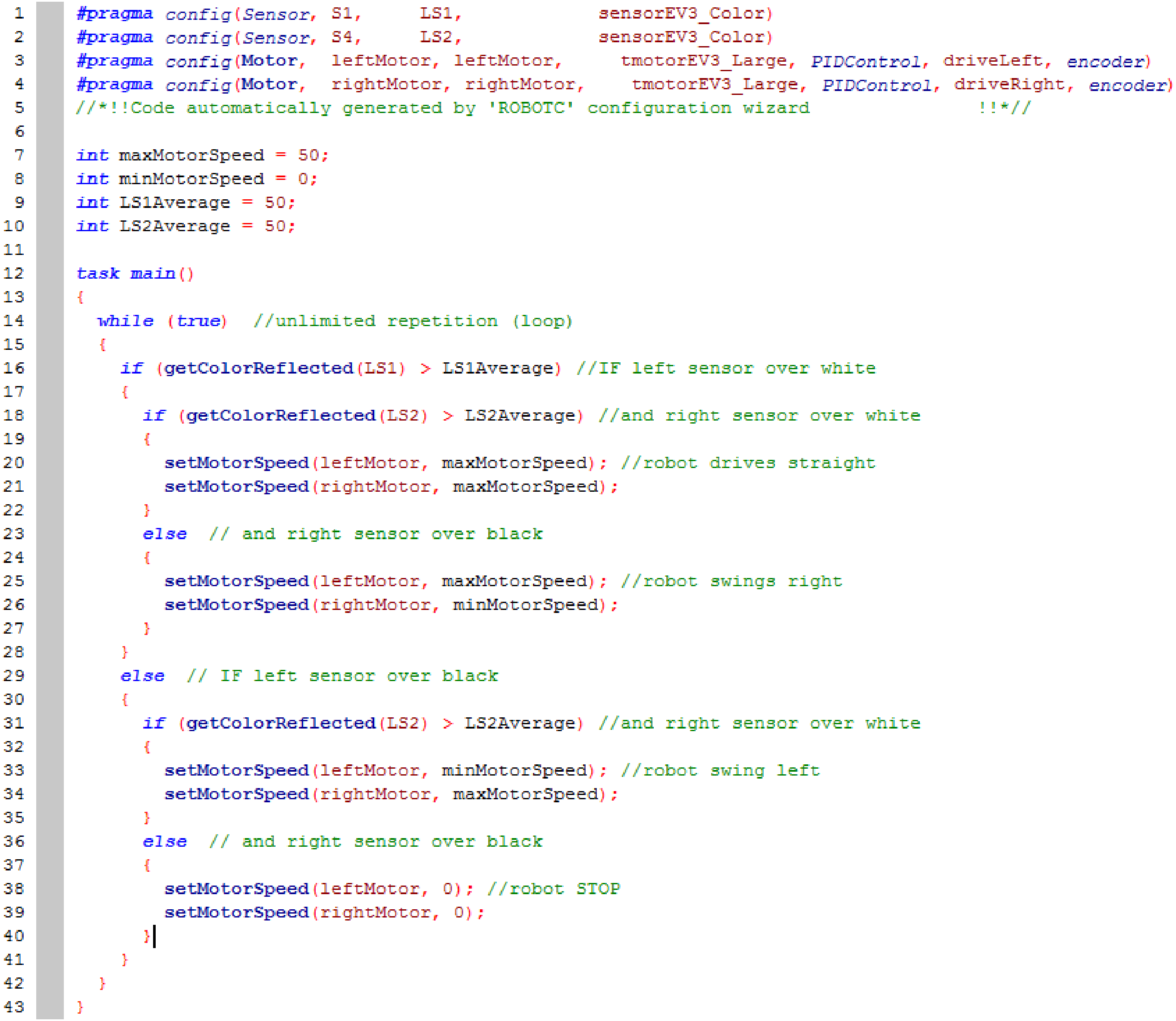
- Instead of changing every motor speed and sensor value throughout the code, which in a big program could be hundreds of instances
- Initialise the variables with - int
  - maxMotorSpeed
  - minMotorSpeed
  - blackWhiteAverage
- Now if we decide to slow the robot down, we only change the maxMotorSpeed value to 25 for example. Or change our Average to suit the conditions



```
#pragma config(Sensor, S1, LS1,      sensorEV3_Color)
#pragma config(Sensor, S4, LS2,      sensorEV3_Color)
//*!!Code generated by 'ROBOTC' config wizard  !!*//
int maxMotorSpeed = 50;
int minMotorSpeed = 0;
int blackWhiteAverage = 50;

task main()
{
  while (true)  //unlimited loop
  {
    if (getColorReflected(LS1) > blackWhiteAverage) //Sensor over white
    {
      setMotorSpeed(motorB, maxMotorSpeed); //robot swings right
      setMotorSpeed(motorC, minMotorSpeed);
    }
    else      //sensor over black
    {
      setMotorSpeed(motorB, minMotorSpeed); //robot swings left
      setMotorSpeed(motorC, maxMotorSpeed);
    }
  }
}
```

Two Sensor Line Follower

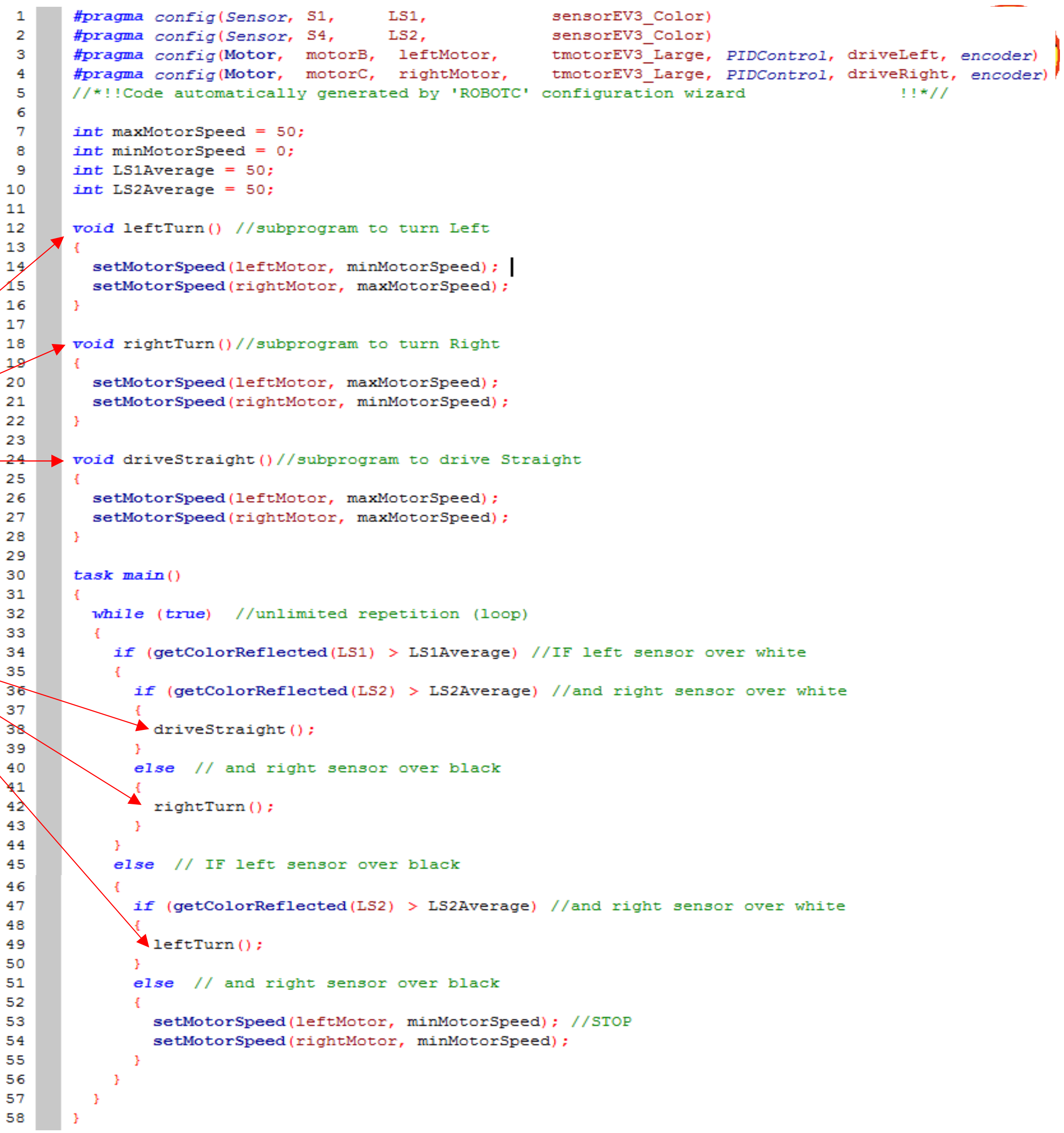
- This code allows us to easily change the max and min values of the motors and sensors
- We can also easily change the sensor averages easily
- Note: Use the Fix Formatting, to neatly indent your code for easy reading



```
#pragma config(Sensor, S1,     LS1,            sensorEV3_Color)
#pragma config(Sensor, S4,     LS2,            sensorEV3_Color)
#pragma config(Motor,  leftMotor, leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  rightMotor, rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

int maxMotorSpeed = 50;
int minMotorSpeed = 0;
int LS1Average = 50;
int LS2Average = 50;

task main()
{
  while (true)  //unlimited repetition (loop)
  {
    if (getColorReflected(LS1) > LS1Average) //IF left sensor over white
    {
      if (getColorReflected(LS2) > LS2Average) //and right sensor over white
      {
        setMotorSpeed(leftMotor, maxMotorSpeed); //robot drives straight
        setMotorSpeed(rightMotor, maxMotorSpeed);
      }
      else  // and right sensor over black
      {
        setMotorSpeed(leftMotor, maxMotorSpeed); //robot swings right
        setMotorSpeed(rightMotor, minMotorSpeed);
      }
    }
    else  // IF left sensor over black
    {
      if (getColorReflected(LS2) > LS2Average) //and right sensor over white
      {
        setMotorSpeed(leftMotor, minMotorSpeed); //robot swing left
        setMotorSpeed(rightMotor, maxMotorSpeed);
      }
      else  // and right sensor over black
      {
        setMotorSpeed(leftMotor, 0); //robot STOP
        setMotorSpeed(rightMotor, 0);
      }
    }
  }
}
```

Using Functions

- By using functions you can create re-usable code
- Here we have created a leftTurn, rightTurn and driveStraight function
- The task main() uses these functions when needed
- In this example we only used them once each, but in bigger programs this method of using Functions is very efficient.



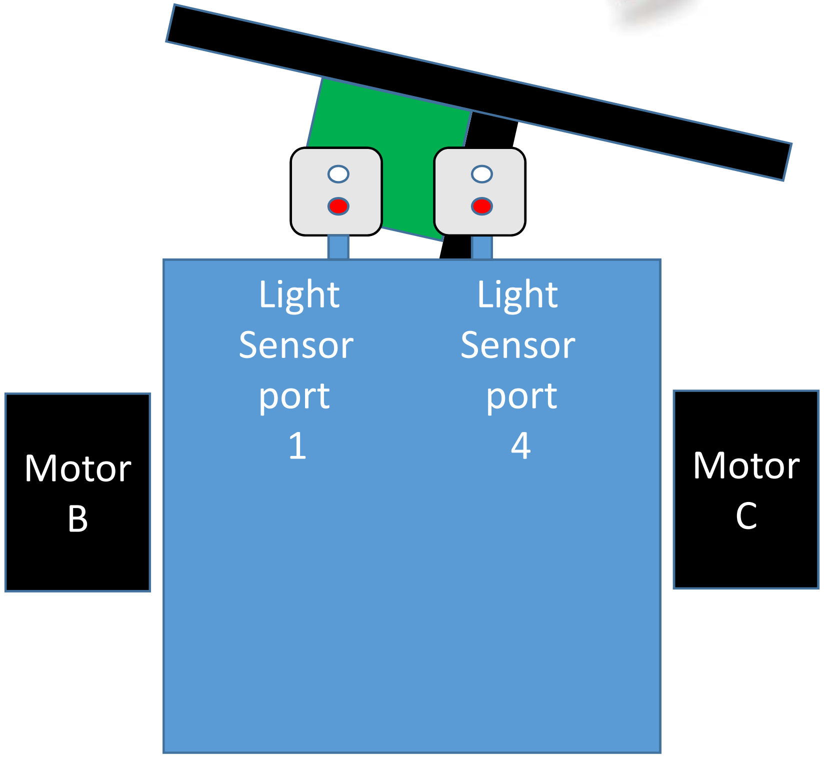
```
#pragma config(Sensor, S1,     LS1,             sensorEV3_Color)
#pragma config(Sensor, S4,     LS2,             sensorEV3_Color)
#pragma config(Motor,  motorB,  leftMotor,      tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC,  rightMotor,     tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

int maxMotorSpeed = 50;
int minMotorSpeed = 0;
int LS1Average = 50;
int LS2Average = 50;

void leftTurn()  //subprogram to turn Left
{
  setMotorSpeed(leftMotor, minMotorSpeed);
  setMotorSpeed(rightMotor, maxMotorSpeed);
}

void rightTurn()//subprogram to turn Right
{
  setMotorSpeed(leftMotor, maxMotorSpeed);
  setMotorSpeed(rightMotor, minMotorSpeed);
}

void driveStraight()//subprogram to drive Straight
{
  setMotorSpeed(leftMotor, maxMotorSpeed);
  setMotorSpeed(rightMotor, maxMotorSpeed);
}

task main()
{
  while (true)  //unlimited repetition (loop)
  {
    if (getColorReflected(LS1) > LS1Average)  //IF left sensor over white
    {
      if (getColorReflected(LS2) > LS2Average)  //and right sensor over white
      {
        driveStraight();
      }
      else  // and right sensor over black
      {
        rightTurn();
      }
    }
    else  // IF left sensor over black
    {
      if (getColorReflected(LS2) > LS2Average)  //and right sensor over white
      {
        leftTurn();
      }
      else  // and right sensor over black
      {
        setMotorSpeed(leftMotor, minMotorSpeed);  //STOP
        setMotorSpeed(rightMotor, minMotorSpeed);
      }
    }
  }
}
```

Detect the Intersection

© G. Tardiani 2015

- As the robot moves from side to side along the line it will stop at an Intersection using the 2 sensor code.
- Compare the values of each sensor
- IF LS1 < LS2, **Turn Left**
  left is less than right sensor

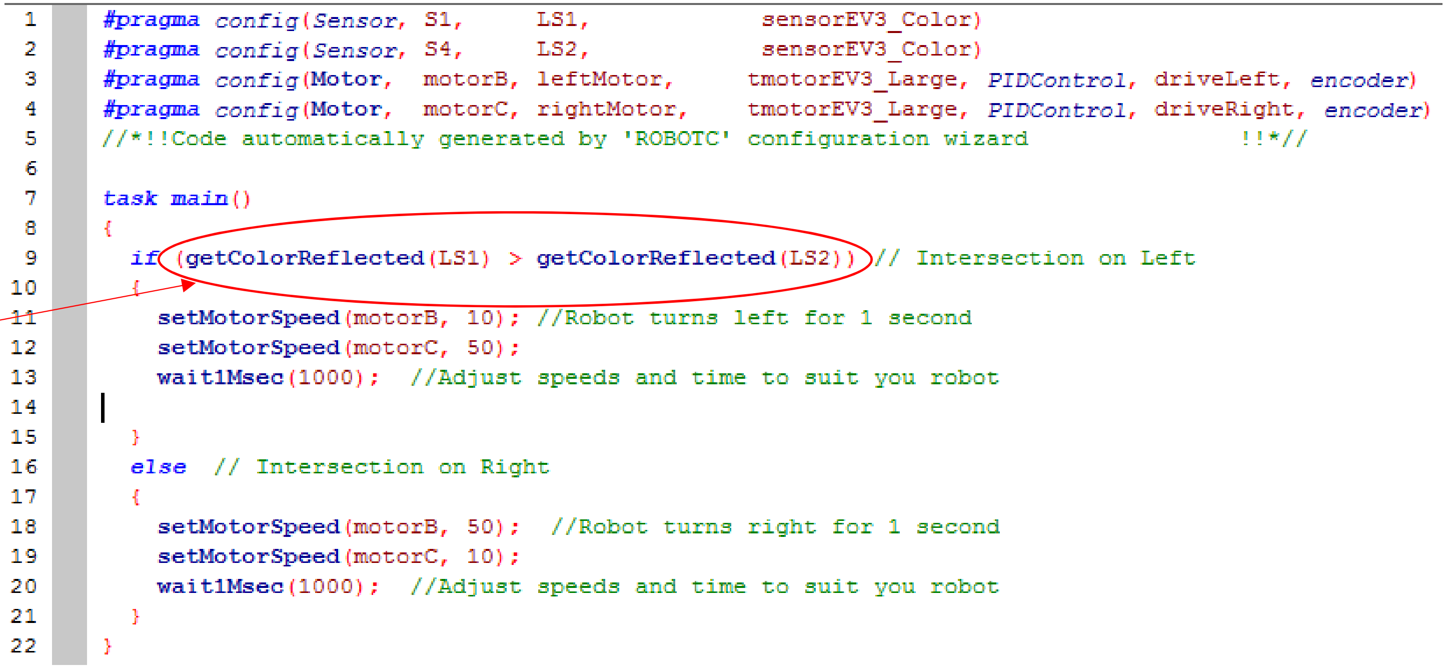
  IF LS2 < LS1, **Turn Right**
  right is less than left sensor



© G. Tardiani 2015

Compare the Difference

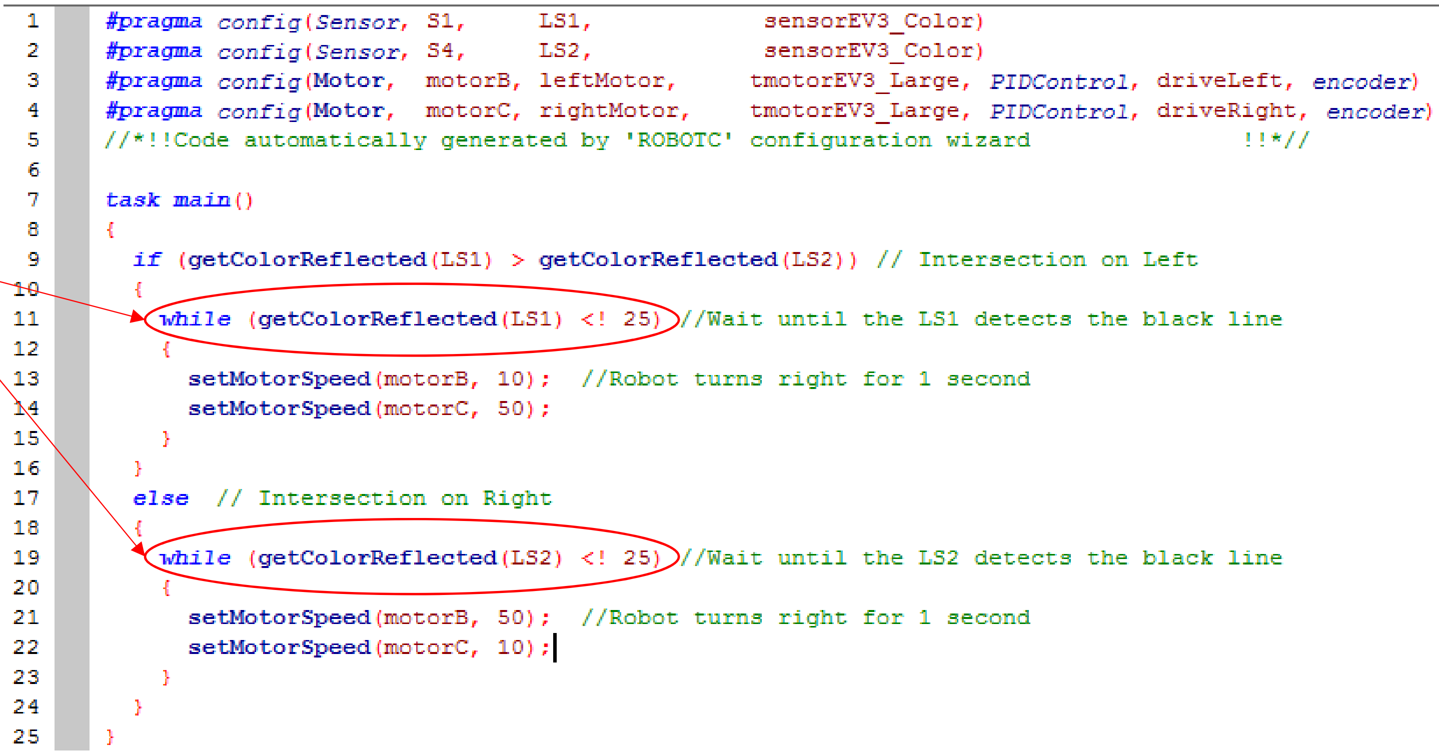
- Having stopped at an Intersection the robot will now compare the amount of reflected light of each Colour Sensor
- If the robot is over an intersection each sensor should have a different reading which the Boolean operator should differentiate



```
#pragma config(Sensor, S1,     LS1,            sensorEV3_Color)
#pragma config(Sensor, S4,     LS2,            sensorEV3_Color)
#pragma config(Motor,  motorB, leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC, rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  if (getColorReflected(LS1) > getColorReflected(LS2)) // Intersection on Left
  {
    setMotorSpeed(motorB, 10); //Robot turns left for 1 second
    setMotorSpeed(motorC, 50);
    wait1Msec(1000);  //Adjust speeds and time to suit you robot

  }
  else  // Intersection on Right
  {
    setMotorSpeed(motorB, 50);  //Robot turns right for 1 second
    setMotorSpeed(motorC, 10);
    wait1Msec(1000);  //Adjust speeds and time to suit you robot
  }
}
```

Alternative compare the Difference

- Instead of a set turn in the previous example, a more reliable solution would be Turn left or right UNTIL the left or right sensor detects the Black line.
- RobotC does not have a REPEAT…UNTIL true post-test loop, so we need to use the While loop with a Not<
- The Not< allows the robot to move while over white and green but not black when it will stop.



```
#pragma config(Sensor, S1,     LS1,            sensorEV3_Color)
#pragma config(Sensor, S4,     LS2,            sensorEV3_Color)
#pragma config(Motor,  motorB, leftMotor,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
#pragma config(Motor,  motorC, rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  if (getColorReflected(LS1) > getColorReflected(LS2)) // Intersection on Left
  {
    while (getColorReflected(LS1) <! 25)//Wait until the LS1 detects the black line
    {
      setMotorSpeed(motorB, 10);  //Robot turns right for 1 second
      setMotorSpeed(motorC, 50);
    }
  }
  else  // Intersection on Right
  {
    while (getColorReflected(LS2) <! 25)//Wait until the LS2 detects the black line
    {
      setMotorSpeed(motorB, 50);  //Robot turns right for 1 second
      setMotorSpeed(motorC, 10);
    }
  }
}
```

3rd Party Sensors and Actuators

- MTA – Everything Lego Education
  http://www.teaching.com.au
- Omni Wheels – RotaCaster designed for Lego
  http://www.rotacaster.com.au/
- Linear Actuators – Firgelli have NXT and EV3 models
  http://www.firgelli.com
- HiTechnic – Official 3rd party Lego Sensors
  http://www.hitechnic.com/
- MindSensor – Unofficial 3rd party Lego Sensors
  http://www.mindsensors.com/
- Dexter Industries – Advanced sensors
  http://www.dexterindustries.com