0.1 PWM MOTOR CONTROL

There is often the situation where the power of the motors must be controlled. One convenient way to do this is that we don't power the motor full time, but we can turn off the motor for short period of time. For an example we can turn the motor on for 1 ms and turn it off for 1 ms. In this case the motor will not get 100% of power, but the motor's average power will be 50%.

Since we are changing the pulse width of logical 1 with the respect to width of logical 0, this technique is called `pulse width modulation` or shorter `PWM`.

This modulated output is controlled by the `analogWrite(pin, pwm)` function. Modulatio can be performed on pins: 3, 5 and 6 of the RobDuino modul. The value of `pwm` parameter can be on a scale of 0 - 255., where 0 is 0% and 255 is 100% of electrical power served.

0.1.1 Tasks:

1. Write new functions for driving the robot left and right with reduced power of the motors:

   - `moveLeftPWM();`
   - `moveRightPWM();`

   In one case you will might find yourself in trouble of controlling the power of the motor since both pins are not able to perform `PWM` output. In this case you can remember that the motor's power is 0 W also if both pins are in state of logical `1`.

   An example of reducing power of both motors in function `moveForwardPWM()` is here:

```
void robotForwardPWM()
{
  digitalWrite( LEFT_MOTOR_PIN_1, LOW);
  analogWrite(  LEFT_MOTOR_PIN_2, 150);
  digitalWrite( RIGHT_MOTOR_PIN_1, LOW);
  analogWrite(  RIGHT_MOTOR_PIN_2, 150);
}
```

Similar to this function you can write other functions to.

2. Change the functions `moveLeft()` and `moveRight()` in S-R-A loop with new ones with less power on motors.

Program 1: PWM motor control.

```
#include "RobotMovingFunctions.h"
const int LIGHT_SENSOR_PIN = A0;
const int SURFACE_BRIGHTNESS_REFERENCE = 400;

void setup()
{
  setIOpins();
  pinMode(LIGHT_SENSOR_PIN , INPUT);
}

void loop()
{
  int light_sensor_value = analogRead(LIGHT_SENSOR_PIN );
  if ( light_sensor_value < SURFACE_BRIGHTNESS_REFERENCE ){
      moveLeft();
  } else {
      moveRight();
  }
  delay(10);
}
```

3. Also add `analogWrite(LEFT_MOTOR_PIN_A, 0);` to function `stopTheRobot()` to stop the `PWM` control of the motor. And do similar code for the `right motor`.
4. Add a parameter `PWM_value` to each function to set the `duty cicle` of the controlled output.
   - `moveLeftPWM(int PWM_value)`
   - `moveRightPWM(int PWM_value)`
5. Save `moveRightPWM(int PWM_value)` and `moveLeftPWM(int PWM_value)` functions into header file `RobotMovingFunctions.h`

0.1.2 Questions:

1. How can we control the average power of the motor?
2. How can we control the average power of the motor in both directions if we are not able to control `PWM` both output pins of the motor?
3. Explain the purpose of programming function `analogWrite(pin, pwm)`.
4. Explain the meaning of the `pin` and `pwm` parameters in function `analogWrite`.

0.1.3 Summary:

0.1.3.1 <++> <++>

0.1.4 Issues:

0.1.4.1 <++>

<++>