

```
/*
GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

Permissions beyond the scope of this license may be available at
http://www.robopartans.com
*/

/*
VERSION: 1.0.0

This is the software program written in NXC for Ship Nikita - a light moving robot
developed in School of robotics "Robopartans". You could find
further information, materials and videos of the robot on our site
http://www.robopartans.com with direct link:
http://www.robopartans.com/robots/ship/
*/

#define THRESHOLD 40
#define LEFT_SENS IN_2
#define RIGHT_SENS IN_4
#define BACK_SENS IN_3
#define MIN_POWER 20
#define MIN_PERCENTAGE_FROM_RANGE 20 // the minimum percentage that the new value must be greater than
the initial to init movement.

int initialValue = 0;

int valuesRange = 100;

sub config() {
    Wait(200);
    SetSensorType(BACK_SENS, SENSOR_TYPE_LIGHT_INACTIVE);
    SetSensorMode(BACK_SENS, SENSOR_MODE_PERCENT);

    SetSensorType(LEFT_SENS, SENSOR_TYPE_LIGHT_INACTIVE);
    SetSensorMode(LEFT_SENS, SENSOR_MODE_PERCENT);

    SetSensorType(RIGHT_SENS, SENSOR_TYPE_LIGHT_INACTIVE);
    SetSensorMode(RIGHT_SENS, SENSOR_MODE_PERCENT);
}

int average() {
    int result = 0;
    result += Sensor(LEFT_SENS);
    result += Sensor(BACK_SENS);
    result += Sensor(RIGHT_SENS);
    return result/3;
}

int max() {
    int max = Sensor(LEFT_SENS)>Sensor(BACK_SENS)? Sensor(LEFT_SENS):Sensor(BACK_SENS);
    return max>Sensor(RIGHT_SENS)? max:Sensor(RIGHT_SENS);
}
```

```

sub init() {
    Wait(50);
    int rep = 3;
    initialValue = max();
    valuesRange = 100 - initialValue;
}

sub valueOut(int line, string text, int value) {
    TextOut(0, line, text);
    NumOut(60, line, value);
}

sub displayValues() {
    int l = Sensor(LEFT_SENS);
    int b = Sensor(BACK_SENS);
    int r = Sensor(RIGHT_SENS);
    valueOut(LCD_LINE1, "Initial:", initialValue);
    valueOut(LCD_LINE2, "Left:", l);
    valueOut(LCD_LINE3, "Back:", b);
    valueOut(LCD_LINE4, "Right:", r);
}

int getPercentageFromRange(int value) {
    return (value*100)/valuesRange;
}

int getPower() {
    int m = max();
    int power = m - initialValue;
    int percent = getPercentageFromRange(power);
    if(percent > MIN_PERCENTAGE_FROM_RANGE) {
        return percent + MIN_POWER;
    }
    return 0;
}

int getTurnLeftToRight() {
    int l = Sensor(LEFT_SENS);
    int b = Sensor(BACK_SENS);
    int r = Sensor(RIGHT_SENS);
    int multiplier = 1;
    if(l>r) { //turn right because the wind is from the left is stronger then the wind from the right
        return l>b?(l-b)*multiplier:0;
    } else {
        return r>b?(-1)*(r-b)*multiplier:0;
    }
    return 0;
}

task main()
{
    config();
    init();
    int newPower = 0;
    int oldPower = 0;
    int finalPower = 0;
    int turn = 0;
    while(true) {
        displayValues();
        newPower = getPower();
        turn = getTurnLeftToRight();
        finalPower = (newPower + oldPower)/2;
        valueOut(LCD_LINE5, "New power:", newPower);
        valueOut(LCD_LINE6, "Old power:", oldPower);
        valueOut(LCD_LINE7, "Power:", finalPower);
        valueOut(LCD_LINE8, "Turn:", turn);
    }
}

```

```
    oldPower = finalPower;
    if(finalPower>MIN_POWER)
        OnRevSync(OUT_BC, finalPower, turn/2);
    else
        Float(OUT_BC);
    Wait(1500);
    ClearScreen();
}
}
```