Proximity Sensor Evaluation

A wrapper class for accessing proximity sensor readings is implemented in the Zumo library in the static class `Zumo32U4ProximitySensors`.

Before it can be used, the method `initThreeSensors` needs to be called. After that, read() will read the current values and save them in local variables that can after that be accessed via the methods `countsFrontWithLeftLeds`, `countsFrontWithRightLeds`, `countsLeftWithLeftLeds`, `countsRightWithRightLeds`. Those methods return the number of brightness values that activated the corresponding sensor. For example, `countsFrontWithLeftLeds` will send out light from the left LED with different brightness values and return how often the front sensor was activated (i.e. it will be activated for high values and not for lower values). This number gives an idea of how far the sensed object will be (i.e. a value of 4 activations means the object is closer than a value of 3 activations).

We did some tests with the default brightness settings:

Distance	**Number of Activations**
> 300 cm	0
< 300 cm	1
< 250 cm	2
< 150 cm	3
< 100 cm	4
< 50 cm	5
< 10 cm	6

We conclude that these values are not very useful because we will only need readings 1,2 and 3 (the others are too far away). Also, when getting a 2, we would not know whether the object is 10cm or 50cm away. Therefore we tested around to see how those value are calculated. We set the brightness values to be 10, 20, 30 … 100 using the method setBrightnessLevels() and checked the readings for different distances:

Distance	**Number of Activations**
> 100 cm	3
< 100 cm	4
< 95 cm	5
< 90 cm	6
< 80 cm	7
< 65 cm	8
< 50 cm	9
< 30 cm	10

We saw that, other than above, the scale is not linear. This means that the brightness values used must be quadratic in order to yield linear output values (i.e. proportional to the distance). Our goal was to generate brightness values such that the outputs will be linearly 0, 1, … 10 from 0 cm to 50 cm (i.e. a 5 would mean 25 cm).

By adapting the generator formula from the library as shown below, we were able to generate the following readings which are fine for now but easily be finetuned later. The readings give us a (theoretical) proximity estimation accuracy of 5cm excluding sensor noise.

```
void generateBrightnessLevels() {
    const uint16_t numBrightnessLevels = 10;
    uint16_t defaultBrightnessLevels[numBrightnessLevels] = {};

    /* generate 10 brightness values that will scale linearly for
     * proximities from 0cm to 50cm. Anything over 50cm will return
     * proximity 0. 25cm will return proximity of 5 etc. */
    for (uint16_t i = 0; i < numBrightnessLevels; ++i) {
        double magic = (2.236 + 1.0975 * (i / 2.0f));
        defaultBrightnessLevels[i] = static_cast<uint16_t>(magic * magic * 1/4.0f);
    }
    proximitySensors->setBrightnessLevels(defaultBrightnessLevels, numBrightnessLevels);
}
```

Distance	**Number of Activations**
< 10	10/9
> 10 cm	8
> 15 cm	7
> 20 cm	6
> 25 cm	5
> 30 cm	4
> 35 cm	3
> 40 cm	2
> 45 cm	1
> 50 cm	0

Obviously this gives us a very accurate value of the distance of the object (excluding sensor noise). One strange behavior of the sensor we have to fix in the future is that the reading is different when an object suddenly appears at e.g. 20cm than when it gradually approaches 20cm.