Night Lesson 3

Curriculum links:

- **Computing:** Algorithms (pseudocode & flowcharts), logical thinking, programming, input devices, output devices, iteration, loops, selection, variables, testing and debugging
- **Science** Day and Night / Sensors, **Design & technology** Product design, **Citizenship** Road safety

Skills: Designing, analysing, problem solving, team working

Resources: Teacher Guide: downloads: presentation, lesson plan, JavaScript Blocks editor

Background:
It is assumed that you have first completed Lesson 1 and 2 of the Night Sensor activity. This lesson assumes no knowledge of using micro:bit and can be adjusted according to your students' experience.

Introduction
In this lesson students are introduced to the JavaScript Blocks editor and program the Night Sensor algorithms they wrote in the previous lesson.

Teacher guide:

Learning objectives

- To follow an algorithm to write a program to create a 'Night Sensor'
- To use paired programming and understand why it is helpful
- To test and debug code

Agenda

- Introduction (5 minutes)
- MakeCode environment (10 minutes)
- Paired programming (15 minutes)
- Testing and refining code (15 minutes)
- Sharing learning (10 minutes)
- Wrap up (5 minutes)

Introduction

- Remind students they will be coding their 'Night Sensor' (slide 2) and recap the pseudocode and flowchart algorithms created in the last lesson (slide 3), inviting them to share how the algorithms will help them in their coding today. Share the learning objectives on slide 4 if you wish.

Main activities
The MakeCode block editor

- If your students have never used the JavaScript Blocks editor before, or need a refresher, briefly show them the interface before giving them time to experiment in pairs. Have a class 'show and tell' session where they share what they have discovered, ensuring you highlight the importance of regular code testing.

Paired programming

- Introduce paired programming to students if they have not used it already (slide 5), inviting them to share their ideas about why it is helpful. Get them to choose the initial driver and navigator, though ensure they swop roles during programming.

- Depending on their confidence level, you can work through the start of the algorithm as a class as a demonstration before setting them off, or let them get started straight away. You can build up from a basic version (e.g slide 6) to a more complex one e.g. with a start and stop button and audios (slide 7).
- You may need to show / help them discover how to create variables, as shown in the example code. Avoid giving students the code to simply copy.
- As they program, encourage students to work through the problems together (e.g. what is the 'right' level for darkness?), thinking about why the program is behaving that way (logical thinking) and to regularly test and debug their code together.
- If they encounter issues, encourage them to work with another pair to problem solve.

Testing and refining code

- As pairs complete, invite them to team up with another pair and explain their code (slide 8).
- Encourage constructive feedback and allow pairs time to refine their code as necessary.

Sharing learning

- As a class share some of the programs and invite pairs to share their learning, using the questions on slide 8 as a guide. Encourage logical thinking about 'why' particular blocks are most appropriate / different ways the program can be created and why a particular approach might be 'best'.
- Highlight the use of input and output devices, iteration and loops, selection and other terminology as appropriate.
- Discuss any common issues and highlight key learning as appropriate to your students.

Lesson wrap up

- Invite students to share their experience of paired programming.
- Explain that next lesson they will be using the knowledge, understand and skills they have been developing over the last 4 lessons to design their own innovation to address a problem relating to safety for children. Remind them of the challenge using slide 9.
- If you wish, use the learning objectives to check progress and understanding (slide 10).

Extension / homework

- Encourage students to add additional features to the Night Sensor. They can write the algorithm first, then add the additional code.
- You could ask them to work on some ideas for their own innovation for homework, refining their initial team ideas from lesson 1, or creating their own.

Differentiation
Support:

- Consider pairings carefully and encourage students to code a simple version of the Night Sensor (e.g. slide 6 or more basic depending on their level). You could also give out printed versions of the blocks to sequence before coding.

Stretch & challenge:

- Consider pairings carefully to enable challenge. Students can code additional elements from the algorithms they planned last lesson, and include other features as they wish. They can be challenged to consider the most efficient way of writing the program and to explain why in detail. If students wish, they could also use one of the other editors (e.g. python)

Opportunities for assessment:

- You could ask students to print and annotate their code for assessment purposes, or create a screen recording / video where they discuss their code.
- Informal, or more formal assessment of programs and explanations of their code.