Lesson Plan

Lesson 5: Nightlight

****

|  |  |
| --- | --- |
| **Ages:** 7 – 11 |  |
| **Programming language:** MakeCode blocks |
| **Topics**: Selection/conditionals (Programming), Input/output, Sensors (Computer systems) **Outcomes:** Students develop their use of the micro:bit’s sensors by using logic to make a simple control system, a nightlight that switches on automatically when it gets dark. |

### Key learning in this lesson

* Understand how inputs, outputs, and computer code work together to make control systems.
* Understand how logic (conditional ‘if… then… else’ instructions) is used to make different outputs happen depending on changes in data from a sensor.
* Use ‘forever’ infinite loops to keep control systems responding to changes in the environment.
* Practise testing and improving a project to make the nightlight work better in specific local lighting conditions.

### Learning objectives

* I can code a micro:bit to make a light that switches on when it gets dark using sensors and logic.
* I can explain that sensors are inputs that sense things in the real world, such as movement and light.
* I can explain that logic is how computers make decisions in code based on whether things are true or false.

# Preparation: before the lesson

### What you need

* BBC micro:bits and micro-USB cables – at least one for every two students
* At least one computer (laptop or desktop) for every two students, with internet access to the Microsoft MakeCode editor: <https://makecode.microbit.org/>
* Alternatively, you can use iPads with the micro:bit app installed. See our guide: <https://mbit.io/lessons-mobile>
* micro:bit battery packs (optional) – one per micro:bit
* PowerPoint presentation – whole class teaching slides
* Code blocks student handout (optional)

****The lesson download also includes an optional ‘.hex’ program file of the completed project, which may be useful if you have limited internet access. You can drag and drop this direct onto the MICROBIT drive when you connect a micro:bit to your computer. You can also drag ‘.hex’ files into the MakeCode editor to examine the code and test it in the simulator.

### Differentiation ideas

* If this is one of your first coding lessons with the micro:bit, it may be hard to know which students will need more support. You can use the extension ideas in the teaching section below for students who complete the task more quickly than others.

### Decide how to deliver the ‘Create’ coding activity

You’ll share the completed code on screen with your whole class from the slides. Additional options include:

* You (or selected students) model building and testing the code yourself on a large screen. The completed code is in the lesson plan and slide deck.
* Give students printed code blocks handouts to follow or cut out and assemble.
* Share a step-by-step YouTube coding video with the whole class, or individual students.
* If YouTube is blocked in your school, we also provide an animation in the slides showing how to assemble the code.
* Students can individually follow an online step-by-step tutorial.
* ****You can also choose to manage the whole class coding activity and save every student’s code using **micro:bit classroom**. Find out more at <https://mbit.io/lessons-classroom>

### Decide how to deliver the ‘Evaluate’ activity

Students download their code to real micro:bits and test the project.

You may want your students to answer the evaluation questions:

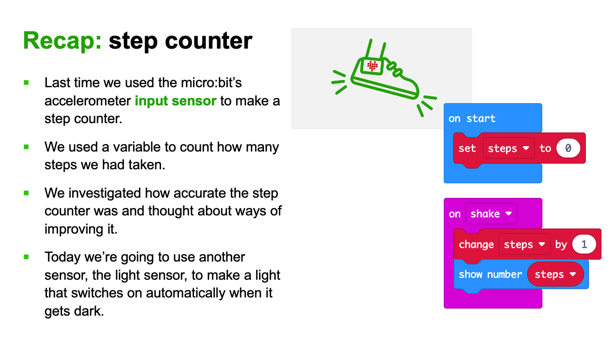
* on paper
* verbally with partners
* as part of a whole-class discussion.

### Glossary

|  |  |
| --- | --- |
| **conditionals**: | see ‘selection’ |
| **control system**: | a system that controls something depending on inputs from a sensor, such as a thermostat that turns heating on when it gets cold |
| **infinite loop**: | a loop that runs forever |
| **input**: | data sent to a computer for processing such as button presses and sensor readings |
| **LED**: | light emitting diode - the micro:bit display is made of 25 LEDs |
| **logic**: | how computers make decisions based on whether things are true or false |
| **loops**: | allow you to repeat sets of instructions without having to write them out multiple times |
| **output**: | data sent from a computer such as words shown on the display |
| **selection**: | making different things happen based on different conditions |
| **sensor**: | an input that senses things in the real world, such as movement, temperature, and light levels |

# Teaching: during the lesson

### Recap prior learning (slide 2)

Ask your students what they discovered last time, for example:

* Last time we used the micro:bit’s accelerometer **input sensor** to make a step counter.
* We used a variable to count how many steps we had taken.
* We investigated how accurate the step counter was and thought about ways of improving it.
* Explain that we’re going to be using a different sensor today, the light sensor, to make a light that switches on automatically when it gets dark.

### Screenshot of slide 3. Introducing the light sensor (slide 3)

* Optionally show the light sensor introduction YouTube video:  
  <https://mbit.io/lessons-light-video>
* The micro:bit’s LEDs, as well as acting as an output, also can work as an input, sensing how dark or light it is.
* We can use this to make projects that react when it gets dark or light.

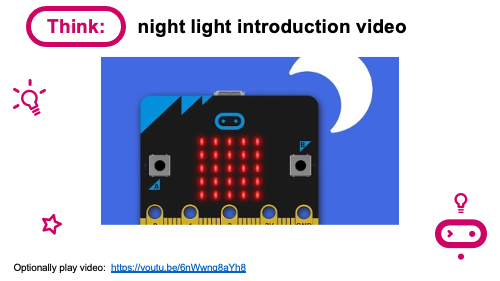
## Think: starter activity

### Screenshot of slide 4. Learning objective (slide 4)

* I can code a micro:bit to make a light that switches on when it gets dark using sensors and logic.
* I can explain that sensors are inputs that sense things in the real world, such as movement and light.
* I can explain that logic is how computers make decisions in code based on whether things are true or false.

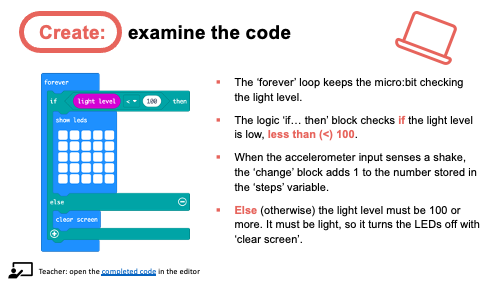
Explain that **Sensors** are inputs that measure things outside a computer, like light, or movement.  
**Logic** is a key idea in computing. It helps machines to make decisions, like turning a light on **if** it’s dark.

**Nightlight introduction video   
(slide 5)**

Optionally play the project introduction video: <https://mbit.io/lessons-nightlight-intro-video>   
  
This explains that the micro:bit light sensor works in a range from 0 (very dark) to 255 (the brightest it can go).

## Create: coding activity

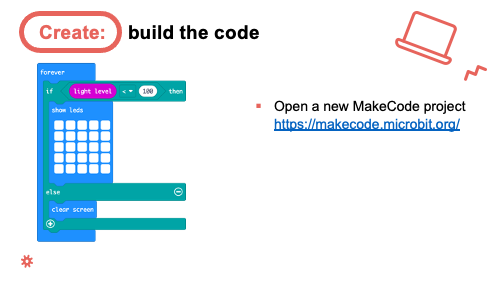
### Examine code with students (slide 6)

Explain:

* The ‘forever’ loop keeps the micro:bit checking the light level.
* The logic ‘if… then’ block checks **if** the light level is low, **less than (<) 100**.
* If the light level is less than (<) 100, it must be dark, so it lights all the LEDs.
* **Else** (otherwise) the light level must be 100 or more. It must be light, so it turns the LEDs off with ‘clear screen’.

You can also follow the link in slide 6 to open the completed code in the editor and model testing it in the simulator by dragging the yellow line in the circle up and down: <https://mbit.io/lessons-nightlight-code>

### Model building the code (slides 7-10)

* You can open a new MakeCode project from slide 7 and model building the code.
* Optionally share the YouTube coding video on slide 8 with your class: <https://mbit.io/lessons-nightlight-code-video>
* Or share the coding animation on slide 9 if YouTube is blocked in your school.

Students recreate the code, testing it in the simulator. They can either:

* Copy the code from slide 6.
* Follow printed code blocks handouts.
* Individually follow a step-by-step online tutorial:  
  <https://mbit.io/tutorial-night-light> - you can share the link from slide 10.
* If you are using [micro:bit classroom](https://classroom.microbit.org/), start a new session and ask your students to join your lesson. You can also open a session with completed code to edit and share with students: <https://mbit.io/lessons-nightlight-classroom>

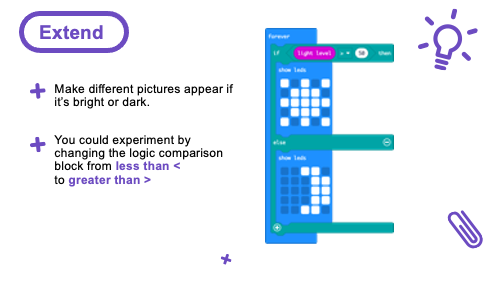
## Screenshot of slide 11.Evaluate: (slide 11)

Students transfer code to their micro:bit and test it. They can cover and uncover the micro:bit, shine a light on it, move it closer and further away from a light source, or cover it with different materials.

**Questions:**

* Does it work as you expect?
* Change the number 100 to a smaller number if the LEDs switch on too easily.
* Change the number 100 to a larger number if it’s hard to make them switch on.
* Transfer the code to the micro:bit and test again.
* How good is the project?
* Would you recommend it to a friend?
* How could you improve it?
* Could it have other uses?
* How does it work?
* Encourage students to think about how it works when holding it in their hands.

## Extend: (optional, slide 12)

* Make different pictures appear if it’s bright or dark. (Sun and moon example on slide).
* Experiment by changing the logic comparison block from less than < to greater than > (Opportunity to reinforce mathematics learning about ‘less than’ and ‘greater than’ symbols.)

## Share: revisit learning objectives (slide 13)

* I can code a micro:bit to make a light that switches on when it gets dark using sensors and logic.
* I can explain that sensors are inputs that sense things in the real world, such as movement and light.
* I can explain that logic is how computers make decisions in code based on whether things are true or false.

 **Ask**:

* What are sensors? How did you use them? (The LEDs work as an input, sensing light levels).
* What is logic? How did we use it today? (We used logic to make a control system: if the light level is low, then we turn the lights on, else we turn them off.)
* How did you test or improve your code? (Depending on the lighting conditions in your classroom, students may have had to use larger numbers to make the lights more sensitive – turn on when it’s brighter – or use smaller numbers to make them turn on only when it’s very dark.)

## Screenshot of slide 14.Next steps (slide 14)

* Today we used the micro:bit light sensor and logic to turn LED lights on automatically when it gets dark.
* Next time, we’ll use micro:bit sensors and logic to make a classic game of chance.

# Assessment: after the lesson

When assessing students’ work you may find it helpful to ask these questions:

|  |  |
| --- | --- |
|  | How well did the student follow instructions to code and test an automatic nightlight? |
|  | Can they explain what sensors are and that the light sensor is used to make the nightlight work? |
|  | What is their understanding of how logic is used to make the light turn on or off depending on the light level reading from the light sensor? |

**Here are some guiding criteria you might want to include when assessing your students’ work:**

### WORKING TOWARDS the learning objective

|  |  |
| --- | --- |
|  | Student tested a nightlight project in the simulator with assistance or with pre-prepared code. |
|  | The student may know a light sensor is used but can’t link the light sensor to other sensors like the accelerometer or explain what sensors are in general terms (inputs that sense things in the real world, such as movement, temperature, and light levels). |
|  | The student may be able to use but cannot describe how the ‘if… then… else’ blocks work using natural language. |

### MEETING the learning objective

|  |  |
| --- | --- |
|  | Student independently coded and tested a nightlight project in the simulator and on a real micro:bit. |
|  | The student knows a light sensor is used and can link the light sensor to other sensors like the accelerometer and can explain what sensors are in general terms (inputs that sense things in the real world, such as movement, temperature, and light levels). |
|  | The student can describe how the ‘if… then… else’ blocks work using natural language. |

### EXCEEDING the learning objective

|  |  |
| --- | --- |
|  | Student independently coded and tested a nightlight project in the simulator and on a real micro:bit and made improvements to their code following testing and downloaded and tested improved code on to a micro:bit. |
|  | The student knows a light sensor is used and can link the light sensor to other sensors like the accelerometer and can explain what sensors are in general terms (inputs that sense things in the real world, such as movement, temperature, and light levels), and give other examples of sensors being used in control systems, for example a thermostat in a heating system. |
|  | The student can describe how the ‘if… then… else’ blocks work using natural language and describe them using the terms ‘selection’ or ‘conditionals’. They may also explain that logic works on the basis of certain statements being true or false. The student may also explain that a ‘forever’ or infinite loop is needed so the micro:bit can constantly keep checking the light level and react accordingly. |