LESSON 20

Creating a Microbits Alarm System

60 minutes | Microbit Masterclass

WHAT THIS LESSON DOES

Follow this guide to build your own alarm system with Microbits. Start by creating a new project in the MakeCode editor, set up threshold variables for sound and light, and programme the alarm to arm, disarm, and trigger based on sensor inputs.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand and apply the concept of variables in Microbits programming.
- Develop skills to use input functions for button presses on the Microbit device.
- Learn to create and utilise functions for specific tasks in coding.
- Gain knowledge on using sensor inputs (sound and light) to trigger events.
- Apply testing and debugging skills to ensure the functionality of the Microbits alarm system.

BEFORE THE LESSON

- Open https://makecode.microbit.org/ on the board and test it loads through the school filter; the simulator runs in-browser with no login.
- One device per pupil or pair, with sound on so the simulator alarm can be heard.
- If using physical micro:bits, check they are charged and pupils know how to download and transfer a project to them.

Creating a Microbits Alarm System

HOW THE LESSON RUNS

Create a New Project
Setup Threshold Variables
Arm the alarm
Disarm the alarm
Define Alarm Function
Setup Alarm Triggers
Test Your Alarm System

TEACHING MOVES

- 1. Create a New Project. Model creating and naming the project on the board so everyone starts from the same blank editor. Point out the toolbox blocks on the left and the simulator on the right.
- 2. Setup Threshold Variables. Explain a variable as a named box holding a value. Show that 128 is a starting point, not a fixed rule, and say they can tune it later once they see how sensitive the alarm is.
- 3. Arm the alarm. Stress that armed starts false and only becomes true when A is pressed. Ask pupils why an alarm should start switched off.
- 4. Disarm the alarm. Contrast B with A: same button-press structure, opposite effect. Have pupils predict what happens if armed is false when a sensor fires.
- 5. Define Alarm Function. Name the pattern: a function is a job you write once and call whenever you need it. Point out it only runs when something else calls soundAlarm, not on its own.
- 6. Setup Alarm Triggers. Walk through the forever loop as a constant check. Highlight that the condition needs both armed to be true and a threshold crossed before soundAlarm is called.
- 7. Test Your Alarm System. Have pupils drive the sound and light sliders in the simulator to trip and reset the alarm. Ask them to arm, trigger, then disarm and confirm each step behaves.

WHAT TO WATCH FOR

- Pupils think the soundAlarm function runs the moment they write it, and are confused when nothing happens. Show that a defined function is dormant until something calls it. Point to the forever loop as the only place soundAlarm gets called.
- Pupils expect the alarm to sound whenever the sensor crosses the threshold, forgetting the armed check. Have them read the if condition aloud: both armed and the sensor reading must be true. Press A to arm, then re-test.
- Pupils treat the threshold value 128 as a correct answer that cannot be changed. Get them to lower and raise the value and re-test in the simulator, so they see it as a setting they control.

DIFFERENTIATION

- Emerging: Give the variable and button blocks as a pre-started file or a printed reference so getting started does not stall them. Pair them with a partner who can help locate blocks in the toolbox.
- Developing: Ask them to change one threshold value and predict, then test, how the alarm responds before moving on. Have them explain their arm and disarm logic back to you in their own words.
- Proficient: Challenge them to add a second alarm response, such as a different icon or a warning tone before the full alarm. Ask them to make A and B also flash a confirmation icon so a user knows the system armed.