Year 9 – Physical computing

Unit introduction

This unit applies and enhances the learners' programming skills in a new engaging context: physical computing, using the BBC micro:bit.

In the first half of the unit, learners will get acquainted with the host of components built into the micro:bit, and write simple programs that use these components to interact with the physical world. In the process, they will refresh their Python programming skills and encounter a range of programming patterns that arise frequently in physical computing applications.

In the second half, learners will work in pairs to build a physical computing project. They will be required to select and design their project purposefully, apply what they have learnt by building a prototype, and keep a structured diary throughout the process.

The Year 8 and 9 programming units are prerequisites for this unit. It is assumed that learners are already able to write Python programs that use variables and data structures to keep track of information. They are also expected to be able to combine sequence, selection, iteration, and function/method calls to control the flow of program execution.

Overview of lessons

Lesson	**Brief overview**	**Learning objectives**
1 Hello physical world	This introductory lesson is meant to get learners acquainted with the micro:bit. They will explore its hardware components, so that they develop an awareness of its capabilities. They will also write and execute their first Python programs on the micro:bit, so that they familiarise themselves with the development environment, the practicalities of flashing their programs, and some simple coding patterns. At the end of the lesson, the learners will discuss what makes physical computing different from what they have been doing so far.	• Describe what the micro:bit is • List the micro:bit's input and output devices • Use a development environment to write, execute, and debug a Python program for the micro:bit

2 Bare bones	Through the course of this lesson, learners will write programs that use the micro:bit’s 5✕5 LED display for output and some of the built-in sensors for obtaining input. This simple ‘bare bones’ setup will allow them to focus on the code and the patterns that often arise in physical computing applications. At the same time, they will get the chance to revisit some elementary programming constructs they learnt in previous units. At the end of the lesson, learners will be asked if they have had any project ideas while exploring the micro:bit. Designing and building their own project is the ultimate goal of the unit.	• Write programs that use the micro:bit’s built-in input and output devices
3 Connections	This lesson provides learners with examples of using the micro:bit’s General-Purpose Input Output (GPIO) pins to connect it to external hardware components, such as switches, speakers, and LEDs. The ability to connect the micro:bit to additional components enhances the built-in capabilities for input and output, which extends the range of projects the learners will be able to build. The lesson also demonstrates the use of the micro:bit’s radio antenna in order to transmit and receive messages wirelessly. This is one of its most versatile capabilities and opens the way for projects that involve multiple micro:bits working together. At the end of the lesson, learners will again be asked about their project ideas. This time, they will also be asked to put their ideas on paper as homework, as they will find themselves taking their first creative design steps in the next lesson.	• Write programs that use GPIO pins to generate output and receive input • Write programs that communicate with other devices by sending and receiving messages wirelessly
4 Dream it up	The first three lessons allowed learners to explore the individual physical computing components at their disposal. Starting with this lesson, they will build their own physical computing project, thus bringing together what they have learnt into a meaningful creation.	• Design a physical computing artifact purposefully, keeping in mind the problem at hand, the

		needs of the audience involved, and the available resources ● Decompose the functionality of a physical computing system into simpler features
5 Build it up	The bulk of this lesson is dedicated to developing the learner projects. In pairs, they will work on their project prototype, following the proposal they drafted in the previous lesson. Halfway through the lesson, learners will pause to receive peer feedback, evaluate it, and fill in their project diary. By the end of the lesson, the project prototypes should largely be implemented.	● Implement a physical computing project, while following, revising, and refining the project plan
6 Wrap it up	In this final lesson, learners will add the finishing touches to their projects; they will proceed to document what they have produced and reflect on the journey. Their projects will be evaluated using a rubric, and they will also take a quiz to assess the knowledge and skills they have individually acquired over the course of the unit. The lesson will conclude with a look at other existing physical computing platforms.	● Implement a physical computing project, while following, revising, and refining the project plan

Progression

Python syntax and short examples relevant to the contents of this unit are condensed into a set of Python for micro:bit cheat sheets.

Curriculum links

National curriculum links **(Computing programmes of study: key stage 3)**

Aims

- Can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms, and data representation

- Can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems

Subject content

- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- Understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- Understand how instructions are stored and executed within a computer system
- Design, use, and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems