Digital technologies - detailed explanations

Below are some detailed explanations of the content descriptions featured in this learning sequence.

Strand: Knowledge and Understanding

Content Description: Recognise and explore digital systems (hardware and software components) for a purpose (ACTDIK001).

Explanation: It is imperative that, from an early age, students understand that digital technologies systems aren't magic – that these systems behave the way they do due to a discrete set of instructions or rules that are predefined by the creator of the system. By exploring a range of systems and going through the process of developing their own sets of instructions, young children begin to understand that, with a small set of permitted rules (for example, forward, turn left, go back), quite a number of different outcomes can be achieved.

The components of the system can vary. Ultimately, though, each system comprises some kind of hardware (for example, robots), and the software that is used to program/define its movements. The way that the software is programmed could be through physical switches or buttons (or, in more complex systems, through written instructions that use a very specific grammar). However, the system will only perform the actions that the hardware and software combined will allow it to do. So, in a simple system such as that of the Bee-Bots that are being explored, the robots can only turn 90 degrees because the hardware is designed such that the instruction to 'turn left' makes the motors in the robot behave in a specific way. If either the motors didn't exist or the software instruction provided changes, the behaviour of the robot would differ.

Ideally, students would have opportunities to explore different types of systems – some physical, others virtual (that is, games or apps) – so that they can recognise how similar instructions perform in different systems and can begin to piece together an appreciation of how simulations can be used to predict outcomes in the real world. You can also provide them with opportunities to role-play the behaviour they expect to observe, and to see if their understanding of instructions is as clear as that understood by the system.

Strand: Processes and Production Skills

Content Description: Follow, describe and represent a sequence of steps and decisions (algorithms) needed to solve simple problems (ACTDIP004).

Explanation: Understanding that both hardware and software work together to make a system exhibit predictable behaviour (as explored through the Knowledge and Understanding strand) is just one aspect of appreciating the relative complexity of digital technologies systems. The reliability of a digital system is dependent on how easily the software instructions can be parsed (that is, read and interpreted). Unlike humans, who are capable of inferring additional understanding from the data provided to them, computers cannot reliably interpret ambiguous instructions provided to them. Therefore, an exploration of the importance of syntax, grammar and a permissible alphabet is key to ensuring that students understand the way data is used and processed in digital systems.

Buzzing with Bee-Bots

Development of an appropriate algorithm that lays out the steps the program will take in a clear sequence is the fundamental understanding that informs the creation of more complex algorithms in the future. A good analogue for exploring the concept of an algorithm is through recipes for cooking, where placing dry ingredients in an oven before adding wet ingredients – putting things in the wrong sequence – does not yield expected results. Another example is to take a mundane task, such as getting dressed in the morning, and explore with students what would happen if they attempted to complete the steps in an incorrect order. In a similar way, pressing the Forward button on a robot before pressing the Turn Left button does not yield the same result as pressing the buttons in the reverse order, and defining the algorithm incorrectly often means the intended behaviour of the program is not observed when it is executed.