Subject: Computing	Year: 6 – Summer 2 – **Programming B – Sensing**

National Curriculum objectives

- Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts;
- Use sequence, selection, and repetition in programs; work with variables and various forms of input and output;
- Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs;
- Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information.

To begin this unit, the children should have already learnt:

Year 1 – Moving a Robot (Spring 1)

Floor robots have buttons which help us to direct them. We can use algorithms (a set of guidelines to perform a task) to program floor robots along routes.

Year 1 – Introduction to Animation (Summer 2)

Programming is when we make a set of instructions for computers to follow. *ScratchJr* is a program that we can use in order to code our own stories and animations.

Year 2

We can create simple quizzes in *ScratchJr* where the user can select an answer by clicking on a sprite. An outcome occurs when the sprite is clicked.

Year 3

We can use event and action command blocks in order to make sprites carry out actions when certain prompts take place. Algorithms (a set of instructions to perform a task) allow us to sequence movements, actions and sounds in order to program effective animations.

Year 4

Count-controlled and infinite loops can be used to create different examples of repetition in games: using repeat and loop operator blocks in *ScratchJr* can make our programs more logical and efficient.

The learning in this unit will prepare the children to learn these things in the future:

National Curriculum Objectives at KS3:

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems;
- Understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem;
- Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems;
- Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits;
- Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users;
- create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability;
- Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns.

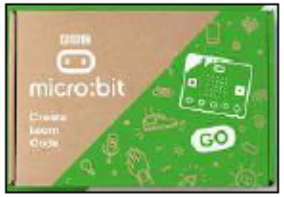
Year 5 Conditions’ can be used in programming: the ‘if… then… else...’ structure can be used to select different outcomes depending on whether a condition is ‘true’ or ‘false’.	
Key Enquiry Question What is a micro:bit? How can emulators help programmers? When might ‘if… then… else’ statements be used in the real-world? What happen if you change the value of a variable? How do operands determine the flow of a program? How tools can you use to check the success of your step counter?	**The Big Idea:** Micro:bits are small computers that perform different actions based on programs written on computer software. Programs are then downloaded to the micro:bit. Micro:bits have a range of input sensors that can be used as input triggers for different codes to run. Output devices on Micro:bits (e.g. LED displays) can be programmed to display words, pictures and numbers.
To achieve ARE, pupils will need to be secure in the following knowledge:	
By the end of this unit, children will know: • A ‘variable’ as something that is changeable; • Examples of information that is variable, e.g. a football score during a match; • A variable can be used in a program; • A program variable as a placeholder in memory for a single value; • A variable has a name and a value; • The value of a variable can be used by a program; • The value of a variable can be updated; • Variables can hold numbers or letters; • Variable can be set as a constant (fixed value); • The importance of setting up a variable at the start of a program (initialisation); • There is only one value for a variable at a given time; • If you change the value of a variable, you cannot access the previous value (cannot undo); • you read a variable, the value; • The name of the variable is meaningless to the computer; • The name of a variable needs to be unique.	**Vocabulary:** Programming; *Scratch Jr.*; command; algorithm; sprite; home; block; stage; background; app (introduced in Y1). Sequence; quiz; debugging (introduced in Y2). Code; events; motion; trialling (introduced in Y3). Logical; condition; selection (introduced in Y5). **Micro:bit; LED; accelerometer; sequence; emulator; motion.**

By the end of this unit, children will be able to do:	**Useful Resources:**
- Identify a variable in an existing program; - Experiment with the value of an existing variable; - Choose a name that identifies the role of a variable to make it more usable (to humans); - Decide where in a program to set a variable; - Update a variable with a user input; - Use an event in a program to update a variable; - use a variable in a conditional statement to control the flow of a program; - Use the same variable in more than one location in a program.	**Online training courses** **Raspberry Pi online training courses** Crumble: Physical computing kit - KS2 Crumble Primary programming and algorithms (remote) *MakeCode* Program.

Y6

COMPUTING: PROGRAMMING KNOWLEDGE ORGANISER

Overview



Using Micro:bits

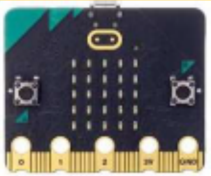
- Programming is when we make a set of instructions for computers to follow.

-Micro:bits are small computers that perform different actions based on programs written on computer software. Programs are then downloaded to the micro:bit.

-Micro:bits have a range of input sensors that can be used as input triggers for different codes to run.

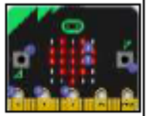
-Output devices on Micro:bits (e.g. LED displays) can be programmed to display words, pictures and numbers.

The Basics of Micro:bits



-**What is a Micro:bit?** A micro:bit is a pocket-sized computer. We can write programs on our computers which can then be transferred to micro:bits to run.
-Micro:bits have an LED light display, buttons, sensors and many input/output features that we can program.

The Parts of a Micro:bit - Front



1. A and B buttons: make things happen.

2. LED Display: shows words, pictures, numbers.

3. Light Sensor: Measures the light that falls onto the micro:bit.

4.Input and Output Pins: Connects the micro:bit to other devices.

The Parts of a Micro:bit - Rear

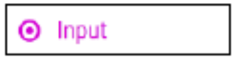
5. Temperature Sensor
6. Compass
7. Accelerometer – Detects movement
8. Radio Communication – to communicate with other micro:bits and devices.
9. USB Port – Connects device to computer. 10. Reset Button
11. Battery Socket – to power away from the computer.
12. Processor – The 'brain' of the device.

Using Micro:bit Software

-**Software Interface:** Just like other programming software, the micro:bit interface has programming blocks and a programming area. The emulator gives a simulation for testing code.

Basic

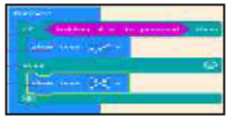
-**Basic Blocks**: Can be used to do things like display images, text and pictures on the LED display. They should be placed into the 'on start' or 'forever' blocks.



Input

-**Input Blocks:** Enables the user to create 'triggers' using different parts of the micro:bit device, e.g. 'on button ... pressed.'

Logic



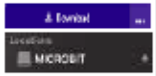
-**Logic Blocks:** Allow conditions to be set. E.g. 'If, then, else' blocks allow us to set actions for when certain conditions are met (true), and alternative actions for when they are not met (false).

pick random 1 to 6

-**Math Blocks:** Includes numbers and sums in programs. The 'pick random number' block can allow different codes to run dependent on the random number generated.

Transferring to Micro:bit

Micro:bit can be connected to the computer using a USB cable.

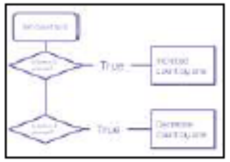


1. Select 'download'
2. Locate the file in the downloads folder.
3. Copy the file from the MICROBIT drive.
4. Run the program on the micro:bit.

-Micro:bit will only run code that has been downloaded. If code is changed in the editor, it will need to be downloaded again in order to run on the micro:bit.

Sensing Inputs

-There are a number of input sensors on micro:bits, including the buttons, light sensor, accelerometer, compass, temperature sensor and GPIO pins.



-We can create algorithms that enable different codes to run depending upon what is detected by different sensors.

-Remember to trial your programs and to debug them if there are sequence, keying, or logical errors.

Important Vocabulary

Programming	Micro:bit	LED	Sensor	Random	Condition	Accelerometer	Sequence	Emulator	Motion