Upper Key Stage 2 – *Robotics and Systems* – Theme Guide

Children investigate automated systems in the wider world and the use of sensors within them. They create, test, debug and refine algorithms, pseudocode and the related programs using sequence, selection, repetition and variables. They program physical devices, controlling inputs and outputs, relating to their study of automated systems.

Learning objectives for the term
To identify automatic control systems, understanding that many have sensors and can respond to changes in conditions around them.
To review the need for efficient program design.
To understand that a variable is used in computer programming to store and retrieve data when the program is run. *Use more than one variable in a procedure, for example write a procedure to draw any polygon in Logo.*
To understand programs can control computer screen displays. *Write programs that control pen width/colour and co-ordinate this with the screen background – as for example in a screensaver.*
To use sequence, selection, repetition and variables in programming an onscreen game or activity. *Program a physical device using Scratch.*
To understand adding comments to programs aids understanding and supports future development.
To know that programs can be written to make physical automatic control systems respond to inputs from sensors.
To know we can review and refine programs to improve them.
On-going Learning Objectives
To understand the need to save drafts and act on critical review to evaluate and improve their work.
To organise their work confidently in agreed locations, using appropriate file-naming conventions and folder structures.
To understand some of the methods they can use to report concerns about content and contact.
To be proactive in keeping electronic/other data secure, protecting personal information and encouraging eSafe practice in others.

Vocabulary – see Glossary in main scheme document for definitions (for terms in blue)	
decomposition, *algorithm,* *variable (in programming),* *constant,* *abstraction,* *pseudocode,*	*program (noun and verb),* *procedure,* *sensor,* *input,* *output*

Possible resources for this theme (further resources are suggested with the explanatory notes below. Note that these are examples and not formal recommendations.)	
Installable programming software • MSW Logo • Scratch 1.4 or 2.x • eLearn and Go **Online Programming environments** • Rapid Router: https://www.codeforlife.education/rapidrouter/ • Scratch: https://scratch.mit.edu/ • Code.org: https://code.org/ • BBC Micro:Bit website http://www.microbit.co.uk	**Physical programmable devices** • Lego® Wedo and Wedo 2.0 • Sphero / Ollie • Roamer Too • InO-Bot • Pi2Go • BBC Micro:Bit • Crumble **Programming Apps** • Swift Playgrounds (iPad) • Pyonkee (iPad) • Tynker (iPad / Android) • Logotacular (iPad) • Hopscotch (iPad)

Please note that with any online platform it is essential that you review the privacy policy and terms and conditions of the service. The school is responsible for the protection of data it holds and compliance with current data protection legislation. Always assess both the data protection and safety of the service you are considering using, and ensure any necessary permissions are in place before using with pupils.

Free *Barefoot Computing* activities to support this theme.	
• **Solar System Simulation Activity.** Link Pupils use Scratch to program a solar system simulation. • **Bug in the Water Cycle**. Link Pupils find and correct errors in a Scratch program, linked to the water cycle. • **Make a Game Project**. Link Children program a game using Scratch. • **Kodu Game Selection Activity**. Link Children program a game using Kodu Game Lab. • **2D Shape Drawing Debugging.** Link Children debug algorithms used for drawing 2D shapes. • **Shapes and Crystal Flowers Repetition.** Link Children create patterns by repeating shapes in Scratch. • **Scratch Maths Quiz Selection Activity**. Link Children create a maths quiz in Scratch, using selection to control the flow. • **Variables Unplugged Activity**. Link Pupils learn about variables by keeping a score in a game. • **Maths Quiz – Adding a Score Activity (Using Variables)**. Link Pupils use variables in Scratch to create a scoring system. • **Investigating Inputs Activity**. Link Pupils learn about input, and program in Scratch, using input devices. • **Investigating Outputs**. Link Pupils learn about output and use Scratch to program a Lego® Wedo motor. • **Classroom Sound Monitor Activity**. Link Pupils program a sound monitor, in Scratch. • **Abstraction Unplugged Activity**. Link An introduction to abstraction where children create models/drawing and guess what each other is creating.	*Barefoot Computing* provides freely available resources to support teachers in delivering the computer science aspects of the 2014 Primary National Curriculum for Computing. Free registration with the *Barefoot Computing* website is required to view and download these resources. To register, visit: http://barefootcas.org.uk References to these resources and the accompanying links are provided with permission from *Barefoot Computing*.

Primary Computing Scheme online materials that are referenced in this guide can be accessed from: http://www.hertsforlearning.co.uk/user/login

You will need to be logged into your school account and have a current subscription to the Primary Computing Scheme to gain access. The materials can be accessed from the *My Resources* link at the top/right of the screen, once you are logged in.

Key learning objectives (some objectives might be used for more than one lesson)
To identify automatic control systems in the outside world, understanding that many have sensors and can respond to changes in conditions around them.

• Ask pupils for examples of automated systems that may have sensors as input devices. These could include:
  ○ Speed warning sign.
  ○ Car park barrier which 'senses' the car approaching.
  ○ Movement sensors in modern buildings, that will turn the lights off if no movement is detected, and will switch them back on when someone enters the room.
  ○ Burglar alarm system. The PIR sensor (passive infrared sensor) detects motion, and causes the alarm to be triggered if movement is detected in the area it covers.
  ○ Supermarket doors open when you walk up to them.
  ○ Metal detection security scanner in an airport.

- Pupils take one system and think about how it works, breaking down the process it follows into individual steps (decomposition) so that these can be presented as an algorithm, which could be written and as a flow chart and/or pseudocode.
- Pseudocode is a way of writing an algorithm, but it more closely follows programming conventions rather than the plain English we would use with younger learners. It is still designed to be read and understood by a person rather than a computer. Like a plain English algorithm, pseudocode is used to help us design and plan a computer program. It could be described as a step between the initial plain English algorithm and actually writing the code, though for older children, the plain English algorithm step could be skipped and you could go straight to using pseudocode. Pseudocode could also be presented as a flow chart.
- See the accompanying sheet: [*Working with pseudocode*]
- See the accompanying sheet: [*10 ideas for teaching algorithms*]
- You could use the shapes menu in Microsoft® Word to create flowcharts.
- The free iPad apps 'Pureflow' (by Aleksandr Kozlov) or 'Grafio Lite' (by Ten Touch Ltd.) may be useful for creating flow charts.
- See the free Barefoot Computing Activities (links above):
  - Investigating Inputs Activity
  - Investigating Outputs
  - Classroom Sound Monitor Activity

To review the need for efficient program design.

- Revisit the main commands in programming languages such as Logo or Scratch, for example reviewing how repetition and procedures can improve program efficiency.
- This is a good opportunity to compare two programming languages side by side, concentrating on using the same programming concepts, but in different ways. Scratch and MSW Logo are possible options for this on PCs/laptops, or iPad apps such as Hopscotch, Pyonkee or Tynker.
- Try writing similar programs in each environment, for example creating a geometric pattern by drawing and rotating a shape multiple times. Compare the differences and similarities between the two programming languages. How does knowing how to create the program in one environment help us create it in the other?
- Pupils should plan their programs on paper first, possibly using *pseudocode*. See accompanying sheet [*Working with Pseudocode*.]
- The important thing is that by understanding the underlying programming concept, such as sequence and repetition, it becomes easier to use new programming environments by learning how these concepts are applied in each one.
- Pupils share programs (and/or their algorithms/pseudocode) with each other (printed out or on screen) and predict what the program will do. They can then test the program to see if they were correct. They can suggest corrections (debugging) or improvements to each other.
- See the accompanying video guides for using MSW Logo.
- See the accompanying downloadable lesson activity: [*Debugging Activity – Scratch – Draw a Pattern*]
- Any of the Barefoot Computing Activities (links above) may be helpful for this objective.

To understand that a variable is used in computer programming to store and retrieve data when the program is run.
Use more than one variable in a procedure for example, write a procedure to draw any polygon in Logo.

- Start by teaching variables unplugged.
  1. Use a small box or container on your desk, and label this box 'Score (Variable)'.
  2. You will run a quiz with the class. (You could do some revision here and quiz the pupils on computing terms or suchlike. Try the accompanying Microsoft® Powerpoint® file: [*File Extensions Quiz*] available to download.)
  3. Before running the quiz, explain that the box is the score and that for each correct answer you will drop a counter into box, but you will remove one for each incorrect answer.
  4. Ask what the score will be at the end of the quiz. Of course, we don't know the answer to this yet, but we have created the container – the variable – to hold the score, and this will change as the quiz progresses.
  5. Then run the quiz with the class. Each time the pupils answer a question correctly, drop a counter into the variables box. Each time the pupils answer a question incorrectly, remove a counter from the variables box.
  6. At the end, see what the final score is by counting what is in the variables box. The important thing is that the variable is the container, into which different values can be added.
- Discuss how using variables can improve program flexibility and efficiency.
- As above, it is useful for pupils understanding of computing concepts if these can be applied in more than one programming language.

- In Scratch you could create a maths quiz or suchlike using a variable to keep the score.
- In MSW Logo you could create a procedure for drawing a shape, where a variable is used in place of a fixed value for the length of one side. The user can then input the value when writing the command to run.

- See the accompanying video guides on using MSW Logo.
- See the accompanying sheet: [*Introducing Variables in Scratch*]
- Start by keeping the activity relatively simple so that pupils can concentrate on using the variable, rather than trying to create an elaborate game at the same time. They will have an opportunity to make a game later in the theme (below.)
- See the accompanying downloadable lesson activity: [*Debugging Activity – Logo*]
- See the free Barefoot Computing Activities (links above):
  - Variables Unplugged Activity
  - Maths Quiz – Adding a Score Activity

To understand that programs can control computer screen displays. *Write programs that control pen width/colour and co-ordinate this with the screen background – as for example in a screensaver.*

- Changing the background colour can be programmed in a number of different environments, including Scratch and MSW Logo. Again, it is helpful for pupils to try and achieve the same objective in different programming environments.
- Suggestions of activities (try to use repetition where possible):
  - Create flashing onscreen 'disco lights.'
  - Using an external sensor, e.g. LEGO® Wedo®, program a flashing screen when an object moves close enough to the sensor.
  - Use Scratch to tell a story or explain a process, either of these linked to another area of learning. Each stage of the story or process uses a different background.

To use sequence, selection, repetition and variables in programming an onscreen game or activity. *Program a physical device using Scratch.*

- Here we can allow children time to apply the programming concepts they have learned to develop a game, using a particular programming environment.
- You might allow them to choose which language they use for their game or activity, depending on what they have previously experienced. Possible options may be Scratch, Hopscotch (iPad® app,) Tynker (iPad® app,) 2Simple 2Code, Espresso Coding, J2Code and more.
- Children create a program which includes selection and repetition and with one or more interactive elements. (For example program a sprite to respond differently to a mouse press or screen touch, or when it touches the edge of the screen or different objects or colours).
- Children should also use variables in their games, for example to keep score. This is easier in some programming environments than in others.
- See the accompanying sheet: [*Scratch Activity – Shark Attack*] as an idea for creating a game, but hundreds of ideas and tutorials are available on the www, and pupils should try and come up with their own ideas rather than just copy directly from a tutorial or guide.
- Ensure time for free play and tinkering so that the children can develop their own program ideas.
- Any of the Barefoot Computing Activities (links above) may be helpful for this objective.

To understand adding comments to programs aids understanding and supports future development.

- In Scratch, by right-clicking in the *scripts* area, comments can be made.
- If using Scratch, pupils can use their existing projects to add comments, working with a partner and explaining what different parts of the program do, and identifying programming features such as variables, selection etc.
- Also in Scratch, project notes can be added from the *File* menu. Pupils could use this feature to explain more about their program, and suggest ideas for further development / improvement.

To know that programs can be written to make physical automated systems respond to inputs from sensors.

- Refer back to the examples above of physical systems in the world around us that use input from sensors.
- Programs can be written to make physical systems respond to inputs from sensors, and this input data can be used to control output.
- What is the output that is controlled by the input in these systems?
  - Speed warning sign – lights come on to tell the driver to slow down.
  - Car park barrier which 'senses' the car approaching. A motor is activated to move the barrier up and allow the car to pass.
  - Movement sensors in modern buildings, that will turn the lights off if no movement is detected, and will switch them back on when someone enters the room. A switch is activated to turn the lights on/off.

- Burglar alarm system. The PIR (passive infrared sensor) detects motion, and causes the alarm to be triggered if movement is detected in the area it covers. A siren/alarm sounds.
- Supermarket doors open when you walk up to them. A motor is activated to open the doors and allow customers in/out of the supermarket.
- Metal detection security scanner in an airport. An audible alert is triggered in the scanner.

- Build (if available) and program a physical device with an output and input (for example a motor which starts when a touch sensor is pressed, or a light which turns on when a movement is detected).
- Highlight where their program includes selection and repetition.
- There are several physical devices/kits available which can be programmed from a computer, and the program then sent to the device. Some can integrate with Scratch and/or offer their own programming software or apps. Some examples include:
  - LEGO® Wedo 2.0 https://education.lego.com/en-gb/lesi/elementary/wedo-2
  - Pi2Go http://www.tts-group.co.uk/pi2go-raspberry-pi-programmable-floor-robot/1007814.html
  - Sphero® and Ollie® http://www.sphero.com/
  - InO-Bot http://www.tts-group.co.uk/ino-bot-scratch-programmable-bluetooth-floor-robot/1009821.html
- If no physical devices are available, simulate these onscreen. An example of software that simulates physical devices is *eLearn and Go* from Data Harvest: http://www.data-harvest.co.uk/catalogue/technology/primary/software/primary-software/3551SL
- See the free Barefoot Computing Activities (links above):
  - Investigating Inputs Activity
  - Investigating Outputs
  - Classroom Sound Monitor Activity

To know that we can review and refine programs to improve them.

- An opportunity to return to programs the pupils have previously written and see if they can improve them and/or develop them further. Review programs with a partner, debugging and improving them.
- Additionally, pupils could add comments to previously written programs, identifying key programming concepts and explaining what different sections of the program do.
- Give pupils the opportunity to work together to comment on each other's work, predicting outcomes and debugging / refining the programs.

On-going Learning Objectives

To understand the need to save drafts and act on critical review to evaluate and improve their work.

To organise their work confidently in agreed locations, using appropriate file-naming conventions and folder structures.

To understand some of the methods they can use to report concerns about content and contact.

To be proactive in keeping electronic and other data secure and protecting personal information when entering data online. Use their understanding to encourage eSafe practice in others.

Suggested independent task – any open-ended activity (2-3 sessions) enabling the children to demonstrate their computing capability around the knowledge and understanding provided in the term

- Use decomposition and algorithms/pseudocode to plan a program to control a physical/onscreen device which includes input sensors and output devices for a specific brief
- Include repetition and selection and use of variables in their program
- Add comments to their program to explain how it works.
- Refine and modify their program, updating the comments to record the process and saving drafts.
- Evaluate their program, considering efficiency and effectiveness.

Other considerations:

Does the task provide for children to work at different levels?
Is there support available for children to select if they wish?
Are there opportunities for the children to review and develop their work?
Is there an opportunity for the children to evaluate the finished task?

In this independent task, pupils have the opportunity to demonstrate their programming skills and also their understanding of the programming concepts they are using.

Depending on the resources available to you, pupils could write their program to control an actual physical device, or it could be onscreen in the form of a game or activity. You should provide a brief for the pupils so they are all working towards the same objectives.

Before creating the actual program in the chosen programming environment, pupils should create a written algorithm (either as text or as a flowchart) to show how they have broken down the process and planned their program. Comments should be added to the program, identifying where specific programming concepts have been applied, such as selection, repetition etc. This should be done within the programming environment if such a feature is present (e.g. Scratch.)

Pupils should have the opportunity to review their work and improve it as required, before writing an evaluation of the program, explaining how they have met the brief and applied programming concepts.

Please note there is an example medium term plan for this theme, donated by a Hertfordshire school, available to download from the online area.