

STUDENT WORKBOOK

STEM in Practice

with KodeKLIX®

- ✓ Define
- ✓ Plan
- ✓ Model
- ✓ Test
- ✓ Reflect
- ✓ Improve

NAME:




AISWA
www.ais.wa.edu.au

KODEKLIX
kodeklx.com

Peter Crosbie • Jan Clarke

AISWA SAMPLE

W

SECTION 1: CORE KNOWLEDGE - LEARN IT!

Consolidating, developing and reflecting on concepts introduced in the Core Activities

1a	Materials, equipment and safety with electricity	Page 4
1b	Conductors, insulators and resistance	Page 7
2	Coding and computational thinking	Page 10
3a	Inputs in a basic circuit	Page 17
3b	Coding inputs and procedures	Page 20
4a	Outputs in a basic circuit	Page 27
4b	Coding to control outputs	Page 30
5a	Circuits in series	Page 32
5b	Circuits in parallel	Page 35
6	Sensors, analogue data, transistors	Page 37
6a	Using a heat sensor in a circuit	Page 40
6b	Coding an analogue heat sensing circuit	Page 42
7a	Using a light sensor in a circuit	Page 47
7b	Coding a light/dark sensor circuit	Page 50
8	Making a sound/music coding circuit	Page 52
9	Using and coding a servo motor	Page 55

G

GLOSSARY

Page 62

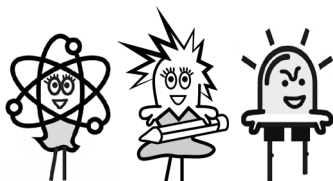
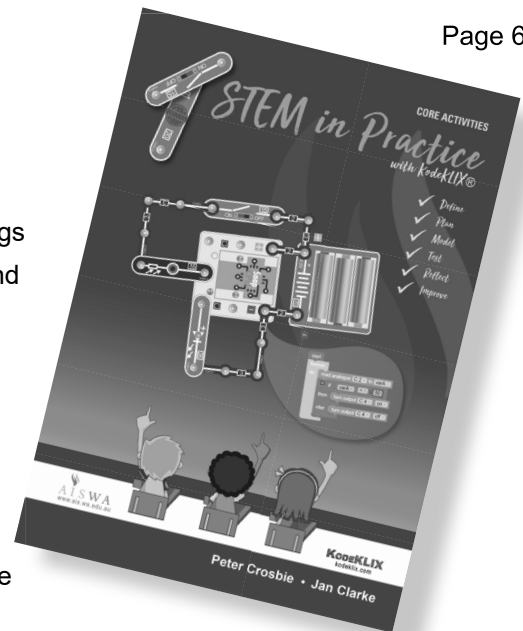
This is YOUR workbook.

It is where you can clarify (make clearer, better understand) the things you have been learning from engineering the KodeKLIX® circuits and building the Blockly code.

The activities in this Student STEM Workbook match with the activities in the blue STEM in Practice Core Activities book.

For example, if you do (3a) in the blue book then look in this book for (3a) to build on that knowledge.

This book is also a place where you can show what you have been learning and how well you can think and solve problems!



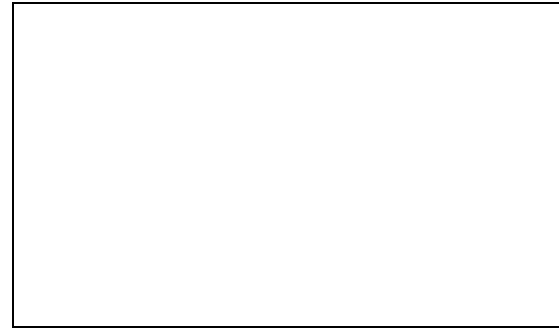
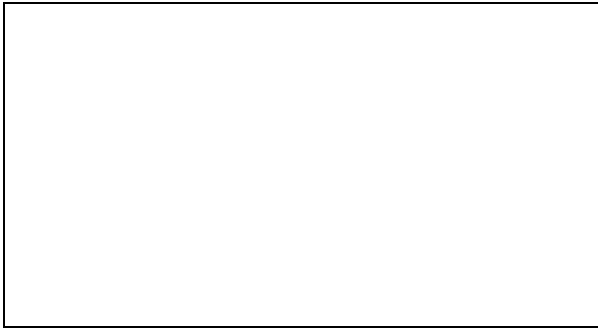
Your teacher has some other interesting extension activities that link this learning to other subjects such as Music, English, Geography, History and Language. There are puzzles and codes to solve, interesting stories to read and QR codes to follow to interactive websites. Ask for them!

MAIN IDEA: COMPONENTS, BATTERIES and SAFETY

Safety must always come first. Electricity can be dangerous if you are not sensible.



1. DRAW 2 examples of ways the kit has been designed to make it a **safe and useful** learning kit.
ANNOTATE the drawing to help explain your reason.

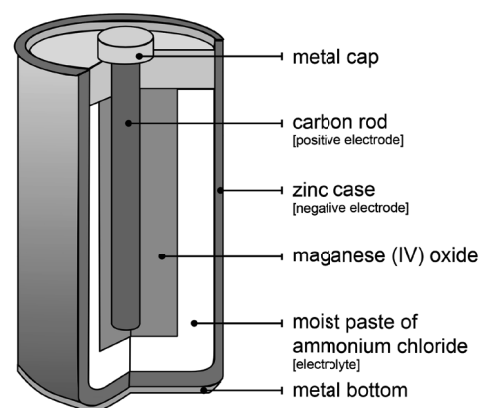


2. LIST any 6 important things about using technology **safely**.
Only use a few words for each answer. The first one is done.

1	Think. Use your brain.	4	
2		5	
3		6	

3. READ this battery diagram. What 3 **materials** inside a battery interact (work together) to make electricity?

- _____
- _____
- _____



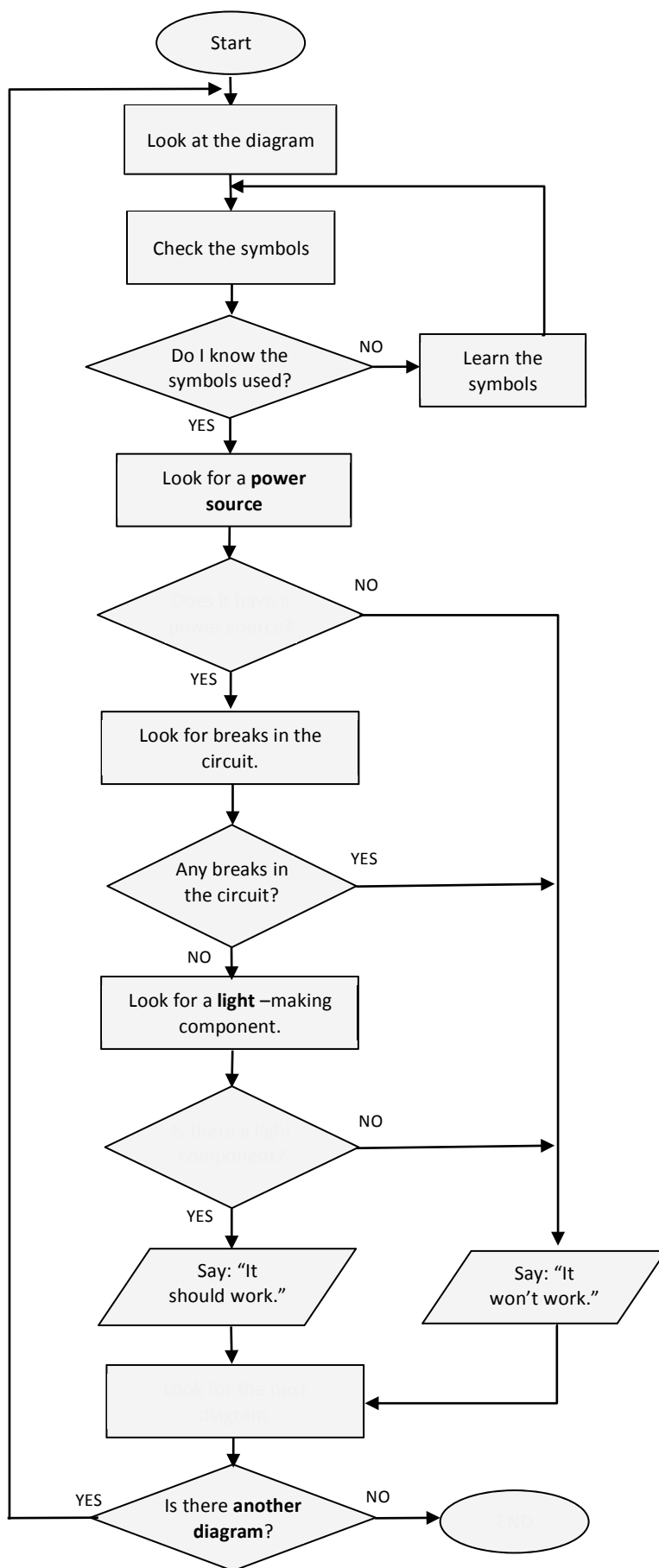
4. ADD + and – to the diagram to show the POSITIVE (+) and NEGATIVE (-) **terminal**.

The + and – show the **polarity** (opposite ends).

Knowing which end is (+) and which is (-) is important for certain components.

(Remember “polarity” by thinking of the North Pole and South Pole of the Earth!)





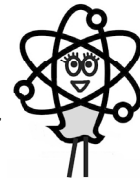
SOLVING A PROBLEM is easier if you have some steps to follow.

Programmers make **flowcharts** to help them work out the steps before they start their coding.

Here is a flowchart (algorithm) to check **whether a light is going to work** in a circuit with a switch.

It's quite easy to follow. Just answer the yes/no questions and see where the arrows take you.

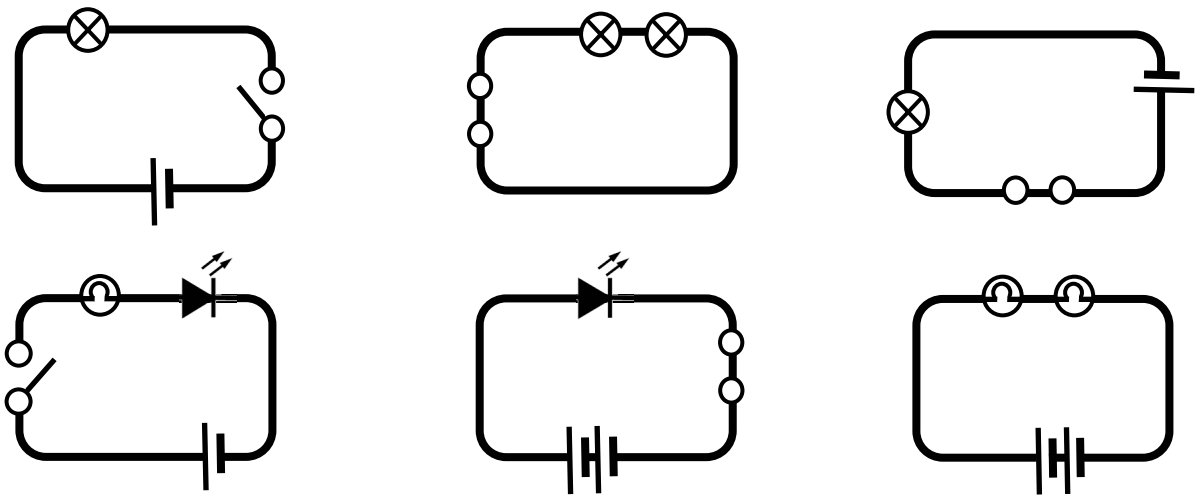
Electra says...



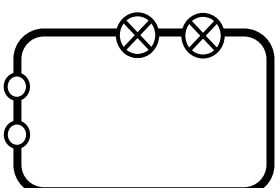
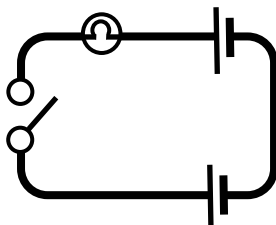
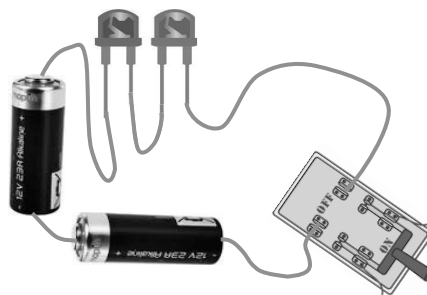
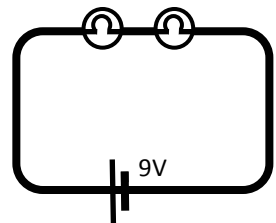
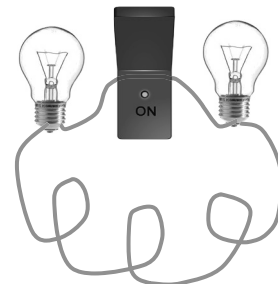
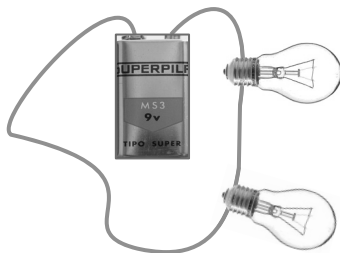
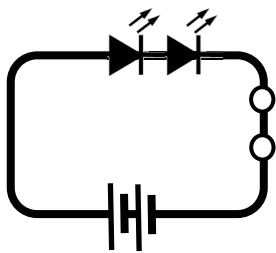
4. ADD in the **missing steps**. Choose from these 4 instructions:
 - a) Is there a **light** component?
 - b) Search for **next diagram**
 - c) End
 - d) Is there a **power source**?
5. TICK the **two messages** that are **OUTPUTS**.
6. There are **2 loops** in this flowchart. Find them. **TRACE** the loops with a coloured pencil.
7. There are 5 places where you can see **branching**. LABEL them 1 to 5.

8. In these circuit diagrams, TICK the circuits where the **light can shine**. CROSS the ones where it won't be able to shine.

Different light symbols have been used so that you learn to recognise them.
Use the checking steps from the flowchart if it helps.



9. DRAW a line to **match** the picture diagram to the correct circuit diagram.

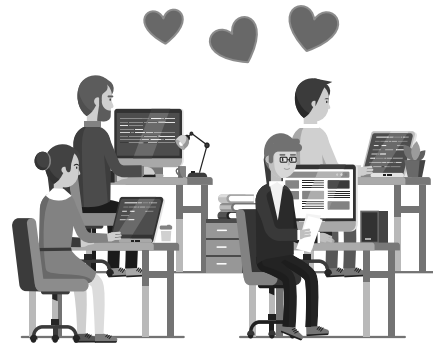


10. WHICH circuit would **never work** because the **polarity** of the batteries is not correct? LABEL it 'WRONG POLARITY'.
11. In another circuit the lamps will also not glow. FIND the problem and give it a **logical** LABEL.

PROCEDURES – mini-programs input into main programs

A PROCEDURE is a mini program inside the main program.
It is **given a name** and when it is needed it is **called** in by that name.

A procedure creates a **module** of code (a useful, re-usable block of code). Programmers love modules of code because they can use them over and over without having to write the same instructions every time.



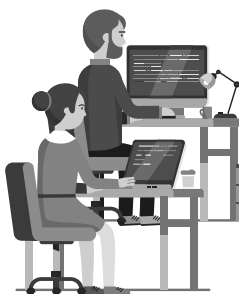
Programmers always look for **patterns of instructions** in code. When they see these patterns they save them as modules. Instead of writing the code every time they just **call** for the one they want and the whole module of instructions will run (**execute**) from one pre-programmed block! It saves such a lot of time.

Programmers like to make code collections (**libraries**) full of procedure modules. Often they **share them** with other programmers. Sometimes they have competitions to see who can write a program that does the same thing with the LEAST amount of code! Clever programmers need fewer lines of code because they make good use of ready-made modules.

The name of a PROCEDURE must be ONE WORD that briefly describes what the code does.
Programmers use the **underscore _ character** to take the place of a space between words.
Sometimes they use **capital letters with no spaces** to show where the different words start.
English teachers might not like this punctuation but it **MUST** be done this way when coding.




Sparkie says...



This **warning symbol** reminds the coder that it won't work until the code is put in.



 **to name_the_procedure_here**
The re-usable module of code goes in here.



The procedure is given a **logical name** so that a programmer knows what it does.

6. The procedure in the activity was called **FlashingLED** because that's what the code created.
RENAME these procedures so that a program could **call** them. They are not useable yet.

Saying what the procedure code does	A logical one_word name for it
play a tune using 6 notes	
make the cat run fast	FastCat
make police car lights and siren work	
draw a big square that is green	
make a motor turn on with a fan	
Make an LED flash when a sound is detected	noise_flash

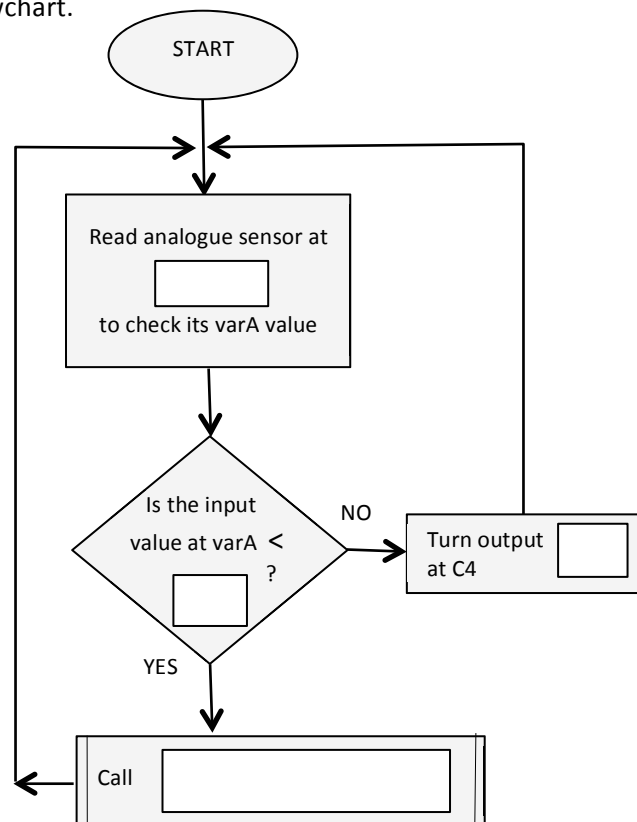
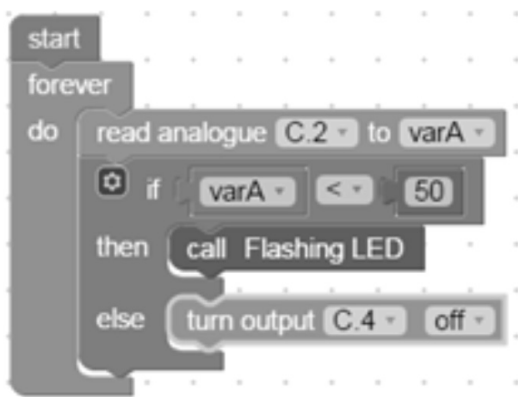
IF-THEN-ELSE BLOCKS – decisions in code

If-then-else statements can be used in coding to give more control over circuits.

If-then-else statements check to see if a condition is TRUE or FALSE.

1. The flowchart is missing some values.
ANALYSE the *Blockly* code. ANNOTATE the flowchart.

- a) FILL IN the missing values.
- b) TICK the **if-then-else decision**.
- c) TRACE and LABEL the 2 **loops**.
- d) LABEL the **procedure** (input code).

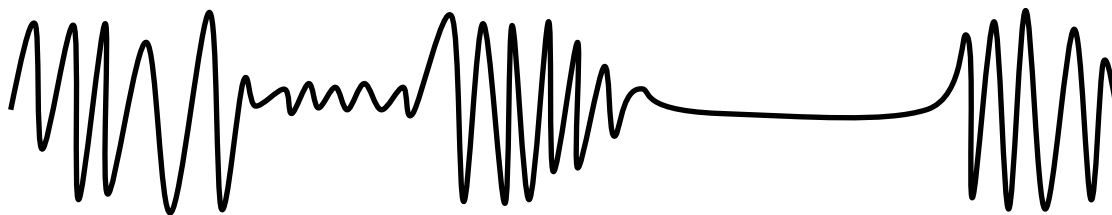


2. TICK the automatic **systems** that might use an **LDR sensor**.

A system to automatically:

- ☐ turn on street lights when it gets dark at night.
- ☐ shut off the stereo if it's too loud.
- ☐ turn on a kettle for a cup of tea at sunrise.
- ☐ open the dog food dispenser when it rains and the dog is barking.
- ☐ ring a bell if somebody walks past a light beam in a doorway.
- ☐ turn on the outside house security lights at night.
- ☐ stop the music playing when a music box lid is closed.
- ☐ turn on the headlights when you drive through a tunnel.
- ☐ make the fridge colder when you open the door and press the switch.

3. IDENTIFY and CIRCLE an area of **loud** sound (use an **L**) and **quieter** sound (use a **Q**) and **silence** (use an **S**) in this soundwave.

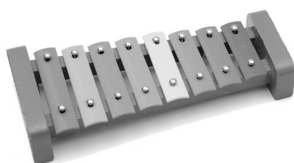


Electronics systems can make and detect waves like this. If you add **certain values** to the code and send this to a speaker you can **make exactly the sounds you want**.

Speakers are sensitive to vibrations. They have a surface that can sense the vibrations and components that turn vibrations into electrical signals. There are other components that amplify the sound (make it louder).

If you place your hand lightly on a speaker box while music is playing you can probably feel the different kinds of vibrations. Deaf people are able to 'listen' to music in this way. Your eardrums pick up sound vibrations like this too. Your skull, jawbone and cheek bones work like in-built amplifiers.

4. CIRCLE the values in the code that identify the **frequency (pitch)** of the sound you want to make with the code.



The notes of a scale on a xylophone.

```

start
forever
do
  if input C.4 is on
  then
    play note 55 for 500 ms on C.2
    pause for 500 ms
  if input C.1 is on
  then
    play note 92 for 500 ms on C.2
    pause for 500 ms

```

Here is some code that makes a musical **scale**. That means that the notes **change in pitch, step-by-step**. They are in sequence.

LOOK at the **pattern** of the **sound frequency data**.
 APPLY the knowledge from what you just read about the relationship between **pitch** (high and low sounds) and **frequency**.

5. Do you think the note sounds will be **getting higher** or **getting lower** as they play in this sequence?

6. Make a **logical guess** about what the missing values would be. ADD them in.

```

start
forever
do
  play note 86 for 500 ms on C.4
  play note  for 500 ms on C.4
  play note 95 for 500 ms on C.4
  play note 97 for 500 ms on C.4
  play note 100 for 500 ms on C.4
  play note  for 500 ms on C.4
  play note 106 for 500 ms on C.4
  play note 107 for 500 ms on C.4

```

7. BUILD this code in *Blockly*. DOWNLOAD it. PLAY it. How accurate was your guess? **Adjust the values** it if you need to make it more accurate.