



Co-funded by
the European Union

INTEGRATED EXPLORATION: CREATIVE THINKING AND SUSTAINABLE SOLUTIONS

2024-1-PL01-KA210-SCH-000243609

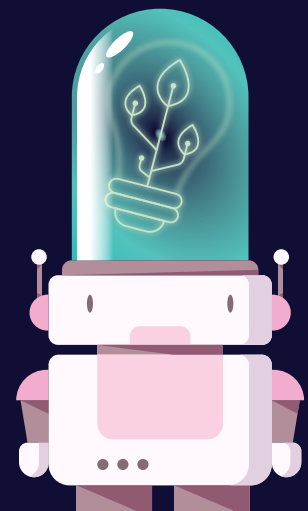
Small-scale partnerships in school education (KA210-SCH)

MODULE 1

Coding and Robotics Basics

Digital Content Guide · Student & Teacher Edition

5 Detailed Lesson Plans



Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

How to Use This Guide

This guide provides everything a teacher needs to deliver Module 1 of the Integrated Exploration curriculum — a STEAM programme designed for secondary school students aged 10–14. No prior coding or robotics experience is required from the teacher.

Each of the five lessons is self-contained and can be delivered in a standard 60-minute class period. The five-day workshop at the end of this module brings the lessons together in an intensive format — ideal for a school project week, summer programme, or cross-school event.

What This Guide Contains

-  **Five 60-minute lesson plans with full teaching notes, differentiation tips, and vocabulary support**
-  **A 5-day intensive workshop programme adaptable to any school or community setting**
-  **Pre/post knowledge quiz, performance rubric, peer feedback tools, and self-reflection sheets**
-  **Designed for culturally mixed groups — all activities work in multilingual settings**
-  **Connects forward to Module 2 (Arts & Science) and Module 3 (Sustainability)**

Module Overview

Module 1 introduces students to the world of coding and robotics through five progressive lessons. The journey begins with unplugged algorithm activities that require no technology at all, moves through block-based programming in Scratch, and culminates in students building and programming their own robots to navigate an obstacle course autonomously.

Throughout, the emphasis is on developing computational thinking — a set of problem-solving skills that are valuable far beyond coding itself. Every activity is designed to be creative, collaborative, and connected to the real world.

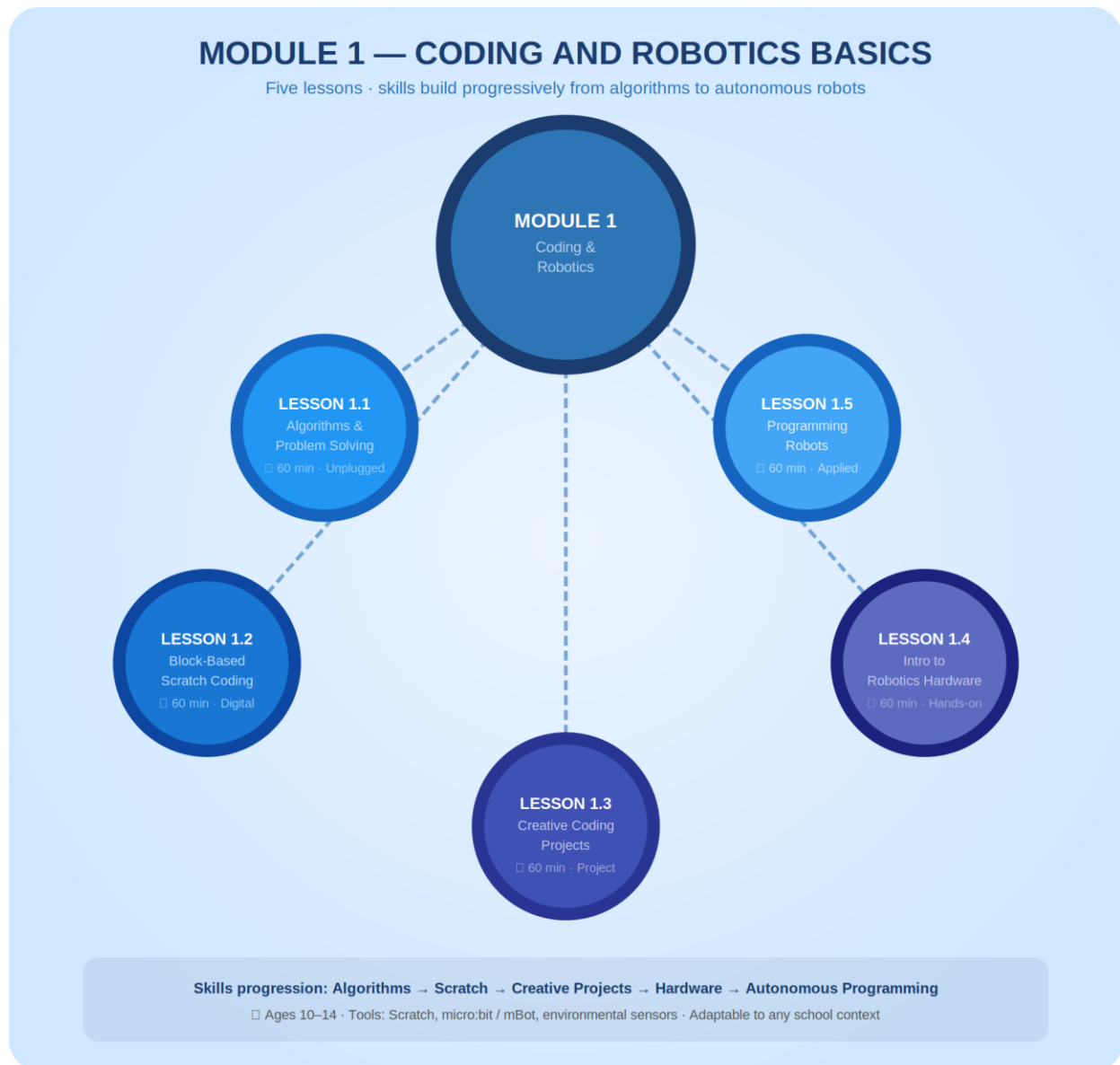


Figure 1 — Module 1: five lessons building progressively from unplugged algorithms to autonomous robots

At a Glance

-  **Target Group:** Students aged 10–14 and their teachers
-  **Lesson Time:** 5 × 60-minute lessons + 5-day intensive workshop
-  **Tools:** Scratch (scratch.mit.edu), micro:bit or mBot, environmental sensors
-  **Assessment:** Pre/post quiz, performance rubric, self-reflection, peer feedback
-  **Key Skills:** Algorithmic thinking, block-based coding, robotics, debugging, teamwork

The Computational Thinking Framework

Before any coding begins, students develop the underlying thinking skills that make coding meaningful. Computational thinking is a problem-solving approach that applies far beyond computers — in science, art, design, and everyday life.

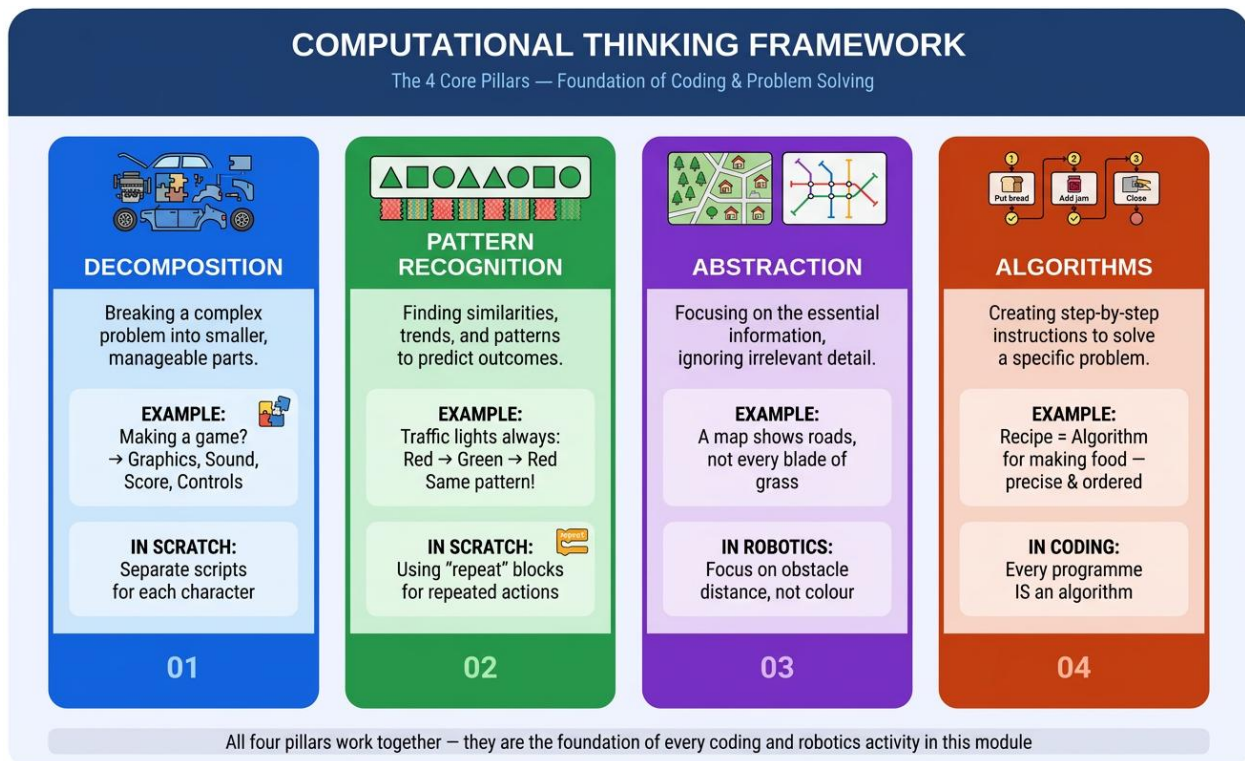


Figure 2 — The four pillars of computational thinking: the foundation for all Module 1 activities

Pillar	Applied in Lesson 1.1	Applied in Lessons 1.2–1.3	Applied in Lessons 1.4–1.5
Decomposition	Break maze challenge into single steps	Separate Scratch scripts per character	Divide robot build into sub-tasks
Pattern Recognition	Spot repeating moves in robot game	Identify loops to reuse across projects	Predict sensor patterns on course



Pillar	Applied in Lesson 1.1	Applied in Lessons 1.2–1.3	Applied in Lessons 1.4–1.5
Abstraction	Focus only on key instructions	Name sprites and variables meaningfully	Use distance threshold, ignore noise
Algorithms	Write step-by-step maze solution	Complete Scratch project script	Robot programme for obstacle avoidance

LESSON PLANS

Lesson 1.1 — Introduction to Coding & Algorithms

Duration	Materials	Grouping	Lesson Type
60 minutes	Algorithm cards, printed grid maze, whiteboard	Pairs, then groups of 4	Unplugged + Discussion

Learning Objectives



Learning Objectives:

1. Define 'algorithm' and give everyday examples
2. Write a clear sequence of instructions to solve a maze challenge
3. Identify and fix errors (bugs) in an algorithm
4. Use key vocabulary: sequence, input, output, bug, debug

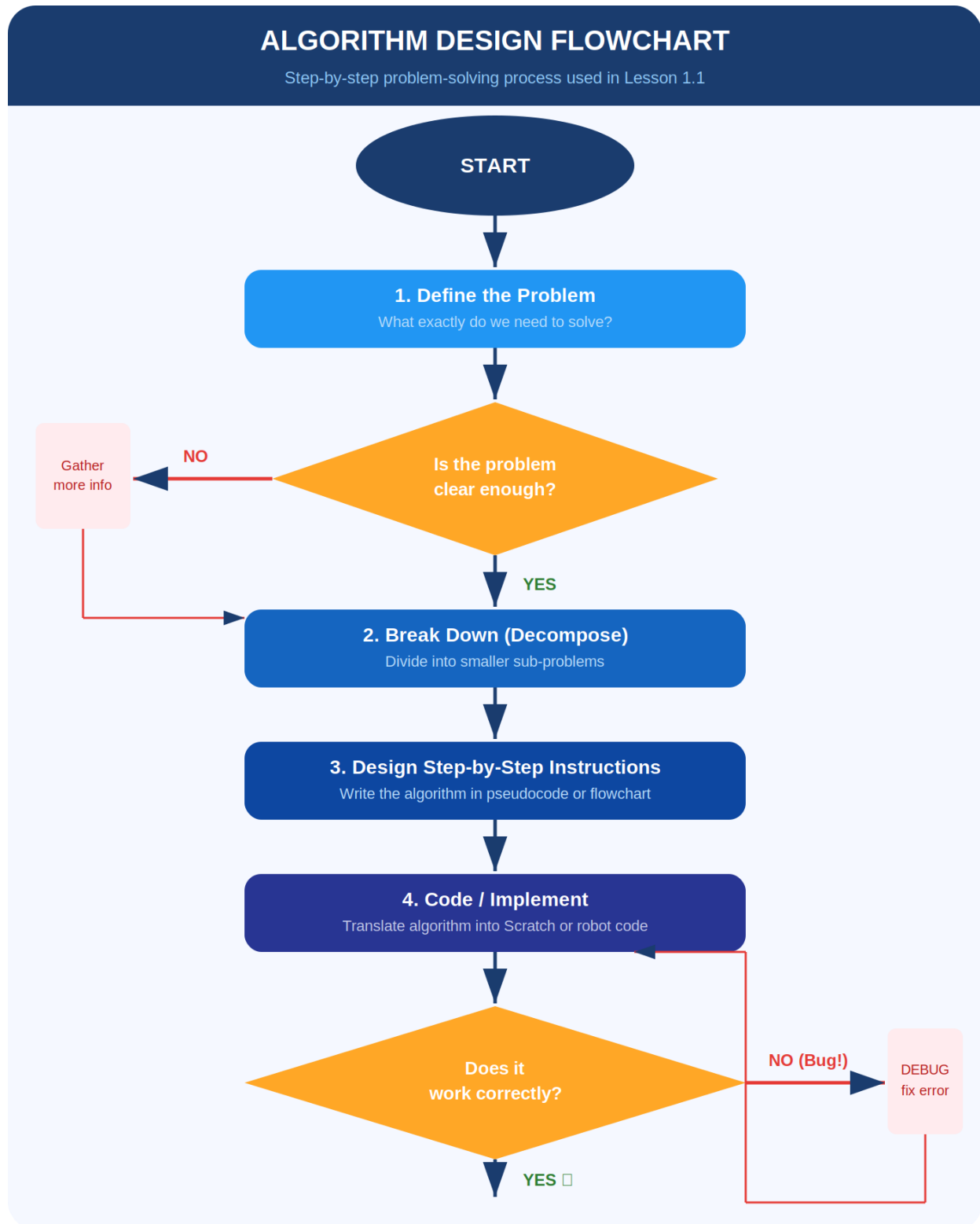


Figure 3 — Algorithm design flowchart: the problem-solving process students apply in this lesson

Lesson Flow

Part 1 — The Robot Game (15 minutes)

Divide students into pairs. One is the 'robot', one is the 'programmer'. The programmer gives precise verbal step-by-step instructions to guide the robot to a target. No touching allowed. After two minutes, swap roles. Debrief: what happened when instructions were imprecise? Introduce: BUG (mistake) and DEBUG (fix).

Part 2 — What is an Algorithm? (15 minutes)

Show three everyday examples on the board: making a cup of tea, tying shoelaces, looking up a word in a dictionary. Ask: what do these have in common? Guide students to discover: a clear start, unambiguous steps, correct order, a clear end. Key point: algorithms are everywhere — search engines, traffic lights, playlist recommendations all use them.

Part 3 — Unplugged Maze Challenge (20 minutes)

Distribute printed 6×6 grid maze cards with instruction cards (FORWARD, TURN LEFT, TURN RIGHT, PICK UP, PUT DOWN). Students write a complete algorithm to get their character to the treasure. Groups of 4 then swap cards and try to follow another group's algorithm — marking any bugs they find.

Part 4 — Vocabulary Consolidation (10 minutes)

Build a vocabulary wall together. Students complete an exit ticket: 'One algorithm I use every day without thinking is _____. A bug I fixed today was _____.'

Term	Student-Friendly Definition	Everyday Example
Algorithm	A step-by-step recipe for solving a problem	Following a recipe to bake a cake
Sequence	Steps that must happen in the correct order	Get dressed: socks before shoes
Bug	A mistake or error in an algorithm	Wrong turn in the maze
Debug	Finding and fixing a bug	Going back and correcting the wrong turn
Decomposition	Breaking a big problem into smaller parts	Cleaning a room = floor + desk + shelves

💡 **Differentiation:** For students who finish early, create a harder maze. For students who struggle, reduce to a 3×3 grid with one turn. The arrow/icon versions of instruction cards support students with lower literacy levels.

Lesson 1.2 — Block-Based Programming with Scratch

Duration	Materials	Grouping	Lesson Type
60 minutes	Computers/tablets with Scratch (scratch.mit.edu), student guide sheet	Individual, paired support	Hands-on digital

Learning Objectives

🎯 Learning Objectives:

1. Navigate the Scratch interface: Stage, Sprites, Block Palette, Scripts Area
2. Use Motion, Looks, Sound, and Events blocks to create an animation
3. Start a programme with 'when green flag clicked'
4. Save and share a Scratch project

Lesson Flow

Part 1 — Guided Interface Tour (15 minutes)

Teacher-led live demonstration on screen. Walk through each area of Scratch. Then allow 5 minutes of free exploration — students click anything, drag blocks, see what happens. Deliberately allow 'chaos': this builds familiarity and reduces anxiety about making mistakes.

Part 2 — My First Animation (25 minutes)

Step-by-step tutorial written on the board and provided as a printed reference:

1. Drag 'when green flag clicked' (Events) to the Scripts Area
2. Connect 'move 10 steps' (Motion) below — click the flag and watch
3. Change 10 to 50 — what's different? Why?
4. Add 'say Hello for 2 seconds' (Looks)
5. Click the Costumes tab → choose a new look for your sprite
6. Click the Sounds tab → add and play a sound
7. Challenge: make the sprite respond to a key press (Events → 'when space key pressed')

Students are encouraged to personalise throughout: choose their own sprite, costume, and record a sound (clap, hum, word) rather than using a preset.

Part 3 — Interaction (15 minutes)

Introduce 'ask and wait' (Sensing) and the 'answer' variable. Students programme their sprite to ask the user's name and say it back. Pairs test each other's projects and leave one comment.

Part 4 — Share and Save (5 minutes)

Save to class account. Students write one comment on a partner's project: 'I noticed that your sprite...'

🌐 **Real-World Connection:** Scratch was created by MIT Media Lab and is used by over 100 million young people worldwide. The same coding concepts — events, sequences, loops — are used in professional languages like Python and JavaScript. The skills students build today transfer directly.

Lesson 1.3 — Creative Coding Projects

Duration	Materials	Grouping	Lesson Type
60 min + optional home time	Computers, Scratch, planning paper	Groups of 2–3	Project-based learning

Learning Objectives

🎯 Learning Objectives:

1. Use loops (repeat, forever) and conditionals (if/then/else) independently
2. Apply design thinking to plan a project before coding
3. Give structured peer feedback (2 Stars & a Wish)
4. Write a brief artist's statement linking creative choices to coding concepts

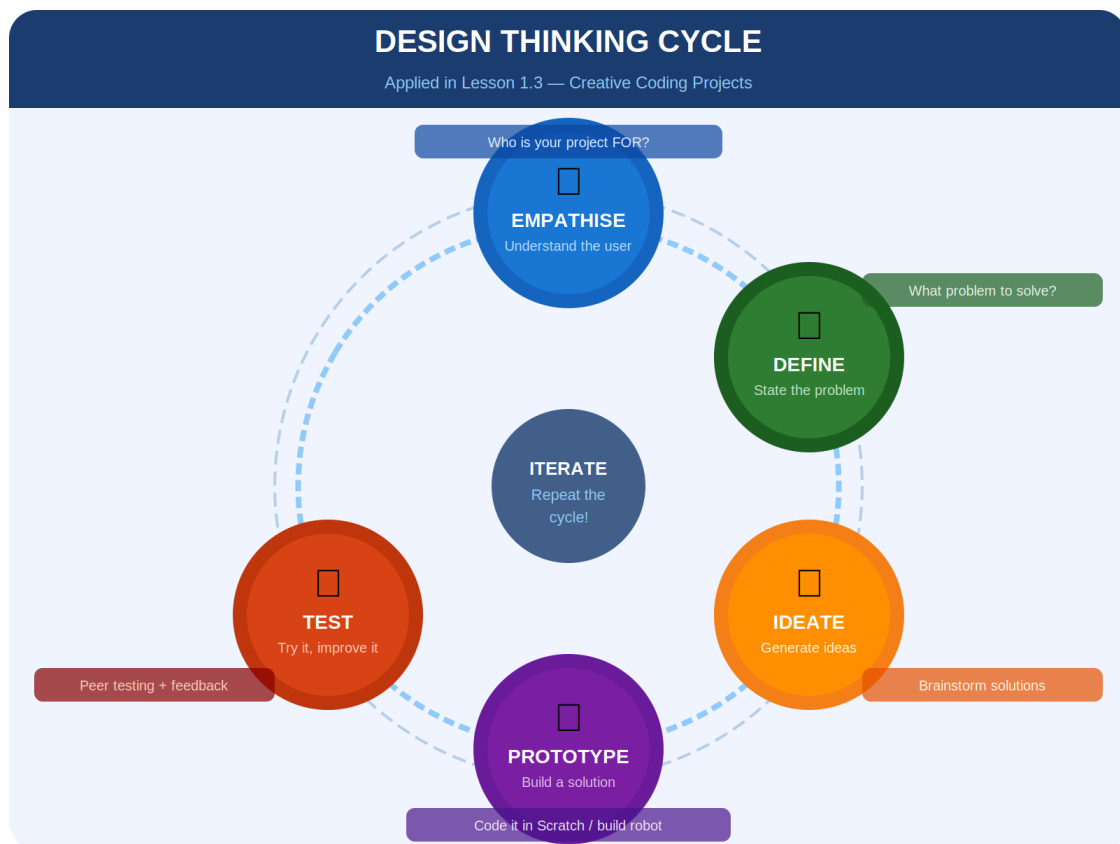


Figure 4 — The design thinking cycle applied to creative coding projects

Lesson Flow

Part 1 — New Concepts: Loops & Conditionals (15 minutes)

Live Scratch demonstration: 'repeat 4' + 'move 100' + 'turn 90' draws a square. Change 4 to 3 (triangle), then 5. Add 'forever' + 'if on edge bounce' for continuous movement. Show a score conditional: 'if score > 10 then say YOU WIN'. Students predict what each change will produce before running it.

Part 2 — Design Thinking Sprint (10 minutes)

Students answer on paper (2 minutes each): Who is this project for? What will they do or feel? List 5 ideas — circle the most exciting. Sketch a rough plan.

Part 3 — Project Build (30 minutes)

Option	Description	Required Coding Elements
A — Animated Story	Interactive story on a theme the student cares about. Characters speak, move, and change costumes.	2 sprites, 3 scenes, user interaction
B — Quiz Game	A quiz on any topic the student knows well. Player answers and receives feedback.	Score variable, 5+ questions, conditional feedback
C — Interactive Artwork	Digital artwork responding to mouse or key input. Focus on visual beauty.	Mouse/key input, loop, colour effects

Part 4 — Peer Feedback (5 minutes)

Swap computers. Complete 2 Stars & a Wish: two specific positives, one specific improvement suggestion.

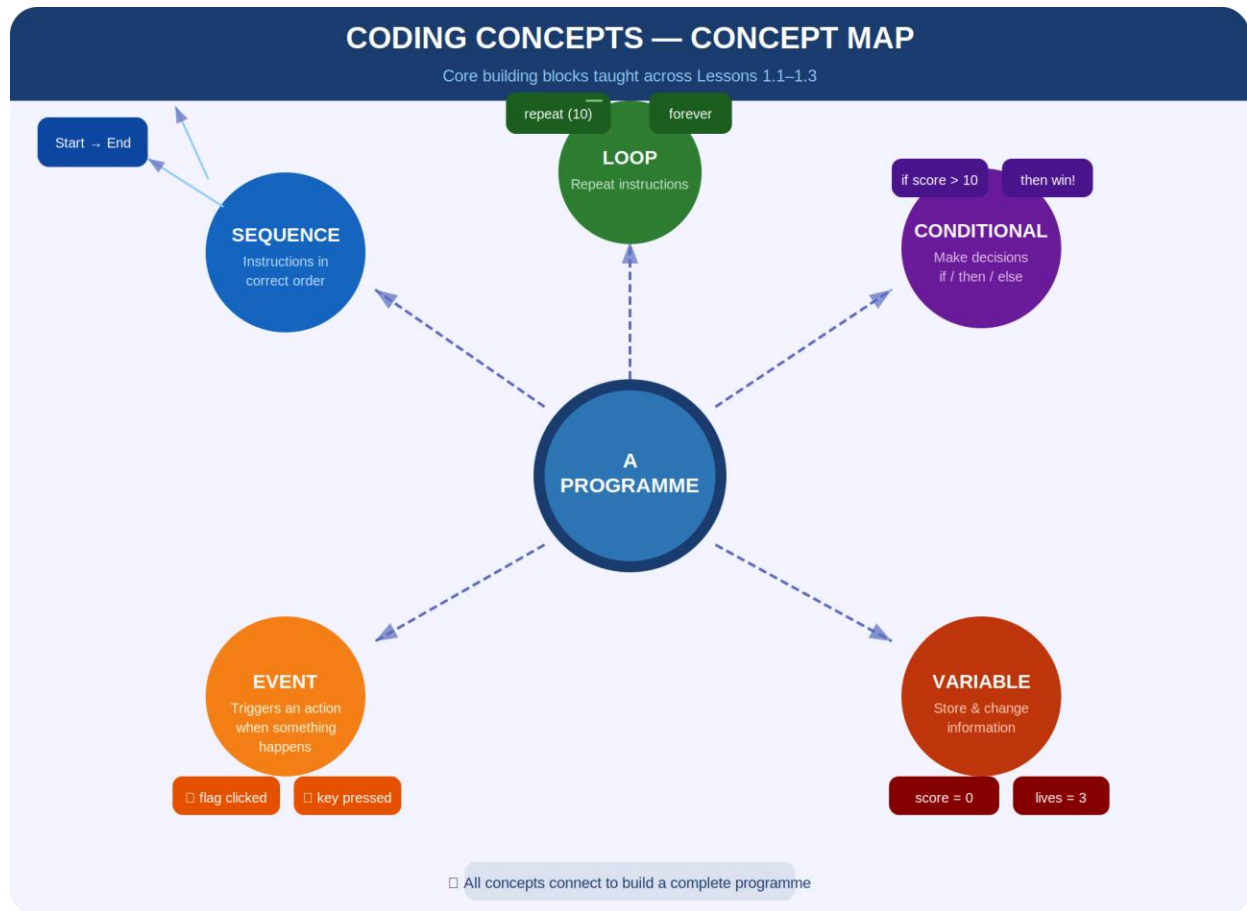


Figure 5 — Coding concepts map: all five building blocks and their Scratch implementations

Lesson 1.4 — Introduction to Robotics

Duration	Materials	Grouping	Lesson Type
60 minutes	Robotics kit (mBot, LEGO Mindstorms, or similar), component cards, worksheet	Groups of 3	Hardware exploration + assembly

Learning Objectives

🎯 Learning Objectives:

1. Define 'robot' and identify examples across diverse industries
2. Identify and label key components: sensors, controller, actuators
3. Assemble a basic wheeled robot from a kit
4. Connect and test at least one sensor

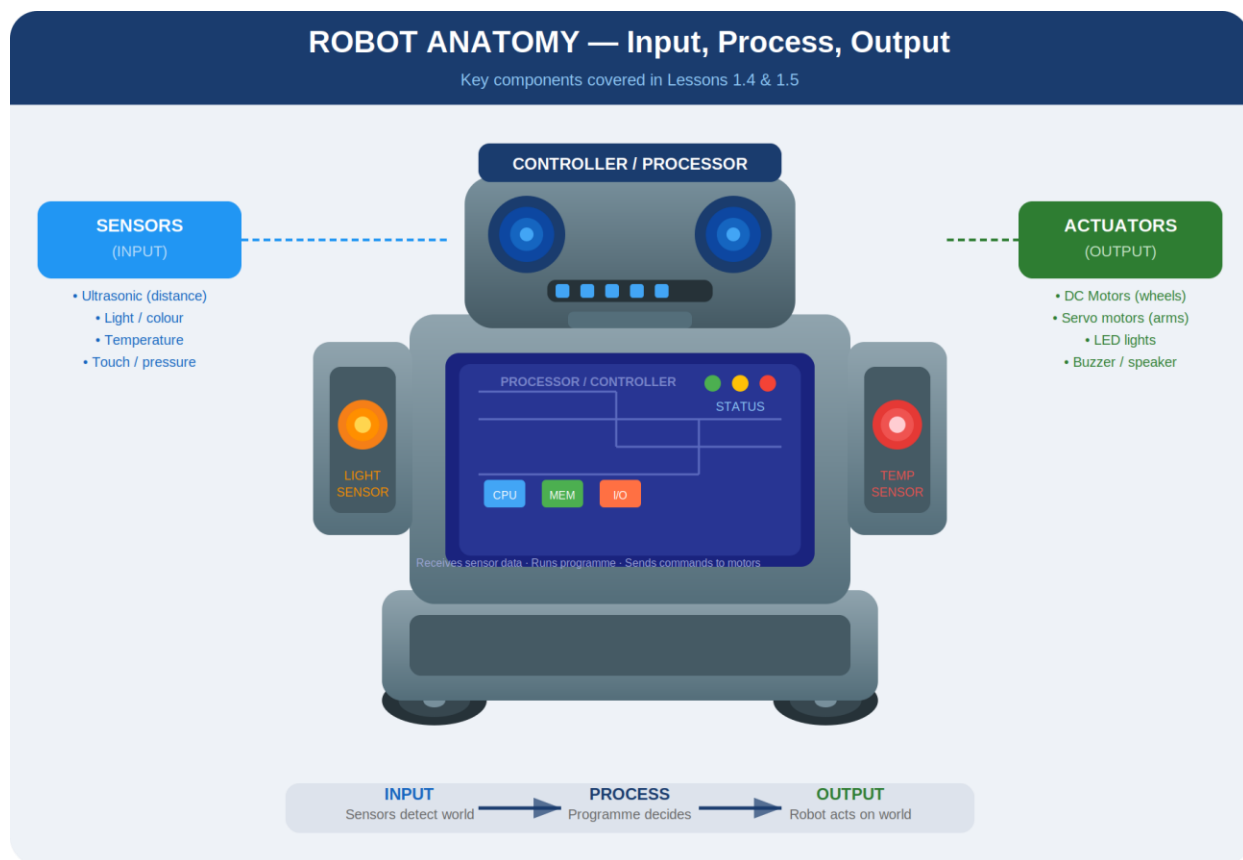


Figure 6 — Robot anatomy: input (sensors), process (controller), output (actuators)

Lesson Flow

Part 1 — What is a Robot? (15 minutes)

Show a 2-minute video montage of diverse robots (surgical, agricultural, warehouse, art robots). Students write on mini-whiteboards: what do ALL robots have in common? Guide them toward the three essentials:

All Robots Have...	Examples	Computing Term
Something to sense the world	Camera, distance sensor, touch sensor, microphone	INPUT
Something to think and decide	Microcontroller chip running a programme	PROCESS
Something to act on the world	Motor, wheel, robotic arm, speaker, LED	OUTPUT

Part 2 — Parts Identification (15 minutes)

Spread all kit components on the table. Students identify and label each on the Robot Anatomy worksheet. Key discussion: Why does the robot have TWO independent motors? What units does the ultrasonic sensor measure in?

Part 3 — Assembly (25 minutes)

Groups follow illustrated assembly cards. Teacher demonstrates uploading a test programme. Goal: get the robot moving forward and stopping. First group to make a controlled square movement wins a small recognition.

Part 4 — Robotics Careers (5 minutes)

Field	How Robots Are Used	Connection to Students
Healthcare	Precision surgery, rehabilitation, dispensing	Biology + Engineering
Agriculture	Harvesting drones, soil analysis, irrigation	Science + Computing
Environmental	Ocean monitoring, reforestation drones	This module's theme
Arts	Animatronics, drone light shows, interactive art	Module 2 connection

Lesson 1.5 — Programming Robots

Duration	Materials	Grouping	Lesson Type
60 minutes	Assembled robots from L1.4, computers with IDE, challenge course tape	Groups of 3	Applied programming + challenge

Learning Objectives

🎯 Learning Objectives:

1. Write a programme to control robot movement (forward, turn, stop)
2. Use sensor data in a conditional to create obstacle-avoiding behaviour
3. Debug a robot programme through testing and iteration
4. Explain what makes a robot 'autonomous'

Connecting Scratch to Robots

Scratch Concept	Robot Equivalent	Example
Events (when flag clicked)	When power on / button pressed	Robot starts on button press
Motion blocks (move 10 steps)	Motor power + duration	Move forward at 80% for 1 second
Loop (forever)	Main programme loop	Continuously check sensors
Conditional (if/then)	If sensor value > threshold	If distance < 20cm → turn right
Variable (score)	Stored sensor reading	distance = sensor.read()

Lesson Flow

Part 1 — Movement Programming (20 minutes)

Teach the motor control blocks in the IDE. Live code: make the robot draw a square on the floor. Students first predict what the code needs, then write and test it. Most groups need 3–5 iterations — celebrate the debugging process, not just the final result.

Part 2 — Sensor-Based Autonomy (25 minutes)

Introduce the ultrasonic distance sensor. Programme: IF distance < 20 cm THEN stop and turn right. Test with a hand as obstacle. Extension challenges:

8. Obstacle on left → turn right; obstacle on right → turn left
9. Reverse and turn if blocked straight ahead (wall-following behaviour)
10. Add LED colour changes based on sensor readings (bridge to Module 2)

Part 3 — Challenge Course (15 minutes)

Achievement	Points	Note
Robot moves without crashing	10	Basic movement working
Robot avoids at least one obstacle	20	Sensor conditional working
Robot completes full course	40	Full autonomous navigation
Bonus: incorporates LED art or sound	15	Cross-module creativity bonus
Bonus: logs environmental sensor data	15	Bridge to Module 3

💡 Assessment note: A robot that partially succeeds with a well-reasoned, clearly explained approach should score as highly as one that completes the course accidentally. The rubric rewards thinking, not just outcome.

5-DAY INTENSIVE WORKSHOP PROGRAMME

Five-Day Workshop — Coding and Robotics Basics

This workshop brings together all five lessons in an intensive format over five consecutive days. It is ideal for a school project week, a cross-school collaboration, a community coding camp, or a holiday programme. It works equally well within a single school or with mixed groups from multiple schools.

Groups should be mixed across year groups, gender, and ability wherever possible. Mixed groups consistently produce more creative outcomes and develop stronger interpersonal skills than homogeneous ones.

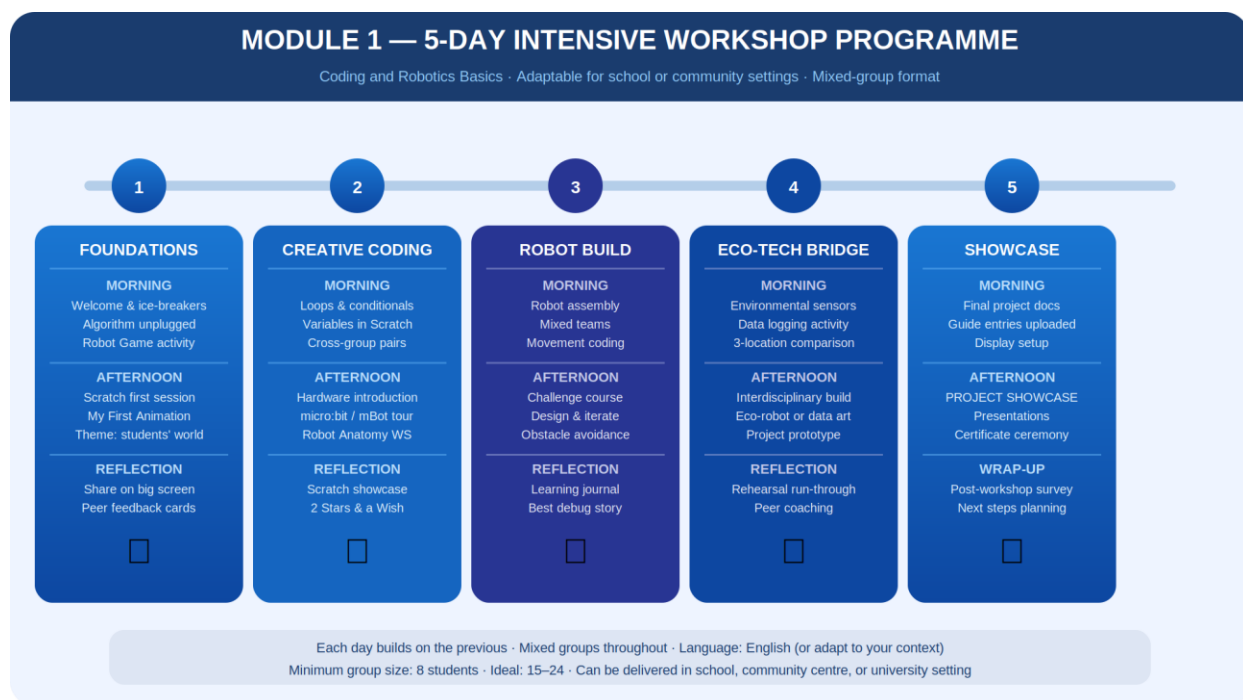


Figure 7 — Five-day workshop programme: one thematic day builds on the next

Workshop Logistics

Element	Recommendation
Group size	8–24 students; groups of 3–4 for all collaborative tasks
Facilitators	1 lead teacher per 8 students; coding knowledge helpful but not required
Session length	Morning session 09:00–12:30, afternoon 13:30–17:00 (can be adapted)
Language	Can be delivered in any language; multilingual settings encouraged

Element	Recommendation
Technology	Computers/tablets (1 per student ideal, 1 per pair minimum); robotics kit (1 per group of 3)
Assessment	Pre-quiz on Day 1 morning; post-quiz on Day 5 afternoon; rubric completed Day 5
Documentation	Photograph all projects; record Day 5 presentations; upload to shared platform

Day-by-Day Focus Areas

Day	Theme	Morning Focus	Afternoon Focus	Reflection Activity
1	Foundations	Welcome, ice-breakers, algorithm unplugged games	Scratch first session — My First Animation	Big-screen sharing; peer feedback cards
2	Creative Coding	Loops, conditionals, variables; cross-group pair coding	Hardware tour; Robot Anatomy worksheet	Scratch showcase; 2 Stars & a Wish
3	Robot Build	Assembly in mixed groups; movement programming	Challenge course design, test, and iterate	Learning journal: best debug story of the day
4	Eco-Tech Bridge	Environmental sensors; data logging from 3 locations	Interdisciplinary project: eco-robot or data visualisation	Rehearsal presentations; peer coaching
5	Showcase	Final project completion and documentation	Project SHOWCASE and presentations	Post-workshop survey; next steps planning

 **Multilingual facilitation:** Use visual instructions alongside written ones. Pair-programming works across language barriers — one student codes, one navigates. Accept home-language discussion; ask for English summary. Celebrate multilingualism as a strength.

 **Celebrate failure:** When a robot fails the course, ask 'What did you learn?' before asking 'How will you fix it?' The process of debugging is where most real learning happens.

 **Document everything:** Photographs and video of student work are essential. They provide evidence for assessment, content for the digital guide, and lasting memories for students.

ASSESSMENT TOOLS

Pre/Post Knowledge Quiz — Coding & Robotics

Administer the same quiz before the module begins (to establish a baseline) and again after completing the five lessons or the workshop. The change in scores is the most important measure — not the absolute score.

#	Question	Correct Answer
1	What is an algorithm?	B — A step-by-step set of instructions to solve a problem
2	In Scratch, what does the 'forever' block do?	B — Runs the script in a continuous loop
3	Which component is an INPUT sensor on a robot?	C — Ultrasonic distance sensor
4	What does 'debugging' mean?	B — Finding and fixing errors in code
5	An 'if/then' block is an example of:	C — A conditional
6	The purpose of a LOOP in coding is to:	B — Repeat instructions multiple times
7	Robots interact with the world using:	C — Both sensors AND actuators
8	To keep a score in a game, you would use a:	B — Variable
9	Environmental sensors can collect data about:	B — Air quality, temperature, and light
10	Decomposition means:	C — Breaking a problem into smaller, manageable parts

Performance Rubric — Coding & Robotics Project

Criterion	Beginning (1)	Developing (2)	Proficient (3)	Exemplary (4)
Algorithm Design	Cannot write a clear sequence	Basic sequences with frequent errors	Correct sequences with minor errors	Efficient; anticipates edge cases
Coding (Scratch)	Cannot use blocks independently	Uses basic blocks with support	Loops, conditionals, events independently	Advanced concepts; well-structured code
Robotics	Cannot assemble or programme	Assembles with help; basic movement	Programmes robot for specific task	Autonomous sensor-based behaviour
Problem Solving	Gives up; needs teacher to fix errors	Attempts with frequent teacher help	Debugs independently; systematic	Proactive; finds root causes; helps peers
Collaboration	Does not engage with group	Participates minimally	Contributes meaningfully to team	Takes leadership; includes all voices

Criterion	Beginning (1)	Developing (2)	Proficient (3)	Exemplary (4)
Communication	Cannot explain the project	Basic description of what was made	Explains decisions and challenges clearly	Engaging; uses evidence; invites questions

Student Self-Reflection — Module 1

Question	Response Format
Before this module, how confident did you feel about coding? (1–5)	Circle: 1 2 3 4 5
After this module, how confident do you feel? (1–5)	Circle: 1 2 3 4 5
What was the most exciting thing you made or programmed?	Open text (2–3 sentences)
What was the hardest moment? How did you deal with it?	Open text (2–3 sentences)
What coding concept could you now explain to a friend?	Open text (1 sentence)
How well did your group work together? (1–5)	Circle: 1 2 3 4 5
What would you like to learn next?	Open text (1 sentence)

Teacher Observation Checklist

Observable Behaviour	Yes	Partial	Not Yet	Notes
Student can define 'algorithm' and give an everyday example	☒	☐	☐	
Student writes and runs a Scratch programme independently	☒	☐	☐	
Student assembles a robot from a kit using instructions	☒	☐	☐	
Student identifies and fixes a bug without teacher direction	☒	☐	☐	

Observable Behaviour	Yes	Partial	Not Yet	Notes
Student uses loops and conditionals correctly	☒	☐	☐	
Student shows resilience when their programme doesn't work	☒	☐	☐	
Student explains their project clearly to an unfamiliar listener	☒	☐	☐	
Student connects coding/robotics to a real-world application	☒	☐	☐	