ALPER-EU Curriculum Framework for Agile Learning of Programming through Educational Robotics

Year levels / age range: K6-K12
Language: English (translations available in Greek, Maltese, Portuguese, Lithuanian and Croatian)
Version + date: v18 / 31-05-2026
Authors / contributors: ALPER-EU Consortium
ALPER Partner Consortium: Hellenic Mediterranean University, University of Malta, Institute of Educational Policy of Greece, Directory of Digital Literacy and Transversal Skills of Malta, Agrupamento de Escolas Teras do Ave of Portugal, Siauliu Dermes Mokykla of Lithuania, Osnovna Scola Pecine of Croatia, 47th Primary School of Heraklion Greece
Curriculum type: teacher-facing framework + classroom implementation guide

1. Rationale and Context

1.1. Educational Context

Educational need. Children need opportunities to understand and create programmable technologies as well as use them. Globally, digital change increases the importance of computational thinking, problem-solving, collaboration and adaptability. At school level, differences in access to equipment, prior experience and teacher confidence can limit participation. ALPER-EU addresses these needs through a coherent primary-school pathway that connects programming with practical construction, teamwork and reflection.

Vision and learner profile. The curriculum aims to develop confident, curious and responsible learners who can analyse a problem, propose a solution, build and program a robot, test evidence and explain improvements. Technical competence and social-emotional development are pursued together. A successful learner contributes to a team, listens to different perspectives, responds constructively to feedback and develops the project's "Can Do" mentality through achievable challenges and supported persistence.

European and project context. ALPER-EU is an Erasmus+ school-education partnership linking universities, educational authorities and schools in Greece, Malta, Portugal, Croatia and Lithuania. Its curriculum comprises 60 cumulative agile robotics sprints integrating robot mechanics, systems understanding, and algorithm design and programming. The application connects curriculum quality with planning, implementation, assessment, review and revision, drawing on EQAVET and relevant EQF/NQF frameworks. These references guide teacher-facing design and quality assurance; they do not assign higher-education qualification levels to children.

Local school context. Participating schools need adaptable teaching sequences, accessible resources and practical guidance for mixed-ability groups. Teachers also need support with robotics, programming, team organisation and assessment. Schools should identify available equipment, staff expertise, learner support requirements and timetable opportunities before implementation. The shared framework provides continuity across settings while allowing teachers to adapt task complexity, pacing and support to their own learners and national curricula.

Why educational robotics. A physical robot gives learners a shared, observable goal and makes relationships between commands, sensor inputs and actions visible. Constructing, predicting, programming and testing connect abstract ideas with direct experience. Robotic challenges can integrate measurement, forces, design and creative expression while providing opportunities to discuss alternative solutions. The **needs-analysis review** identifies promising benefits for programming, motivation and collaboration, while recognising resource demands, teacher-training needs and limited evidence for the combined robotics–Scrum approach.

Why Agile learning. Robotic solutions usually require repeated attempts rather than a single correct answer. Age-appropriate Scrum routines organise this process into manageable micro-sprints: agree a goal and roles, build and code, test, share the result, and reflect on the next improvement. Short check-ins make progress and difficulties visible. Reviews develop explanation and feedback skills, while retrospectives help teams improve their collaboration. Routines are simplified and scaffolded according to age and gradually support greater learner autonomy.

Inclusion from the outset. Inclusion is a design requirement across all sprints. Tasks should offer accessible instructions, different ways to participate and demonstrate understanding, flexible pacing and appropriately supported roles. Teachers should plan for learners with ADHD, sensory or communication needs, mobility limitations and other support requirements without assuming that a diagnosis determines ability. Role rotation, visual supports, predictable routines and multiple entry points help every learner contribute meaningfully to the same shared challenge.

1.2. Intended learners

The programme serves primary learners aged 6–12 across three developmental levels. Sprints 1–20 target ages 6–8; Sprints 21–40 target ages 8–10; and Sprints 41–60 target ages 10–12. Each level contains two cycles of ten sprints. Core ideas recur with increasing complexity: early sequences and cause-and-effect relationships develop into loops, conditions and structured debugging, then variables, more complex sensor-based logic and a supported transition from block-based programming to Python.

No prior robotics or programming experience is assumed at entry. For later levels, teachers should check prerequisite concepts and provide bridging activities rather than assume that

age alone indicates readiness. Early learning uses concrete demonstrations, visual or block-based commands, oral explanation and guided teamwork; scaffolding is reduced as learners become more independent.

Delivery can draw on computing or ICT, mathematics, science, technology and design, with arts and language activities supporting creative modelling and communication. Schools should identify a coordinating teacher and agree how subject time and assessment evidence are shared. Sessions need sufficient time for planning, construction, programming, testing, review and reflection; a sprint may extend across more than one lesson. Summer clubs and after-school activities provide additional delivery settings identified in the application. Exact lesson allocations should follow local timetables, learner needs and the demands of each sprint.

1.3. Principles underpinning the curriculum

The following eight principles guide the selection of tasks, teaching routines, resources and assessment across all 60 sprints.

Equity and accessibility. Access to learning should not depend on prior coding experience, home technology or speed of task completion. Provide shared equipment, clear multimodal instructions, accessible workspaces and low-technology or unplugged preparation where useful. Adjust support and ways of demonstrating learning while maintaining meaningful challenge.

Inclusion and respect for diversity. Treat differences in ability, language, experience and perspective as resources for learning. Plan flexible participation, accessible communication and supported team roles from the outset. Monitor who participates and whose ideas influence decisions, ensuring that every learner can make a valued contribution.

Developmentally appropriate spiral progression. Revisit concepts through increasingly demanding problems across the three age levels. Move from concrete action and supported visual programming towards abstraction, independent reasoning and textual code. Introduce complexity gradually, check understanding and provide consolidation so that progression reflects readiness as well as age.

Active inquiry and creative STEAM learning. Learners develop understanding through asking questions, predicting, making, testing and explaining. Integrate science, technology, engineering, arts and mathematics where these connections serve the challenge. Encourage alternative designs and reasoned experimentation, with teacher guidance that supports discovery without removing learner agency.

Authentic and responsible problem-solving. Connect robotics to understandable needs such as assistive devices, environmental care, transport or everyday automation. Discuss whom a solution helps, its limitations and its effects on others. Embed safe equipment use, fairness, sustainability and responsible handling of any classroom data.

Agile collaboration and shared responsibility. Use clear goals, manageable tasks and rotating roles to make collaboration purposeful. Teams plan together, communicate progress

and share responsibility for results. Age-appropriate Scrum routines support negotiation, accountability and collective decisions, with teachers helping learners manage disagreements constructively.

Feedback reflection and growth. Treat errors as information for improvement. Combine robot-test evidence with teacher feedback, peer discussion, self-assessment and team retrospectives. Learners explain what changed and why, practising metacognition, persistence and emotional regulation alongside technical debugging. Teachers use their own reflections to adapt subsequent teaching.

Coherent outcomes and adaptable implementation. Every sprint should connect a programming or robotics objective with an explicit soft-skill objective and suitable evidence of learning. Maintain cumulative progression while adapting context, pacing, materials and scaffolding to the school. Review implementation evidence regularly so that local flexibility supports the programme's shared educational aims.

Basis for this section: ALPER-EU project application, especially pp. 4, 34, 47–56 and 68–80; DWP2A1 Needs analysis and current state of the art in K12 agile software and soft skills training; and the confirmed sprint-to-age allocation for this curriculum.

2. Aims and Learning Outcomes

2.1. Overall aims

The programme aims to enable learners to:

- Develop computational literacy by analysing problems, designing algorithms and creating increasingly independent programming solutions, progressing from concrete and block-based tasks towards structured Python programming.
- Connect robot mechanics, sensors and actuators with program behaviour, enabling learners to construct, test and improve purposeful automated systems through creative, interdisciplinary STEAM challenges.
- Apply age-appropriate Agile/Scrum routines to set shared goals, break challenges into manageable tasks, organise roles, review progress and adapt plans using evidence from testing.
- Strengthen collaboration, communication and responsible leadership through equitable participation, constructive disagreement, shared accountability and clear explanations of technical decisions.
- Cultivate curiosity, a "Can Do" mentality and emotional resilience by treating errors as feedback, persisting through debugging and using self-assessment and reflection to improve learning strategies.
- Promote inclusive and responsible technology use by considering different users' needs, accessibility, safety, fairness and environmental impact when evaluating robotic solutions.

2.2. Learning Outcomes

The following cumulative competencies describe what learners should demonstrate at the end of each level. They integrate programming, robotics and soft skills through observable actions. Teachers may accept demonstrations, oral or visual explanations, annotated programs and team reflections as evidence, with accessible alternatives and age-appropriate support.

Level 1 ages 6 to 8 Sprints 1 to 20

By the end of Level 1, students can, with guided support:

- Analyse a simple robotic challenge by identifying its goal and arranging the necessary actions into a logical sequence, using objects, pictures or spoken instructions.
- Create and modify a short block-based program that produces an intended light, sound or movement response, and predict how changing one command or parameter will affect it.
- Construct and test a simple robot or mechanism, relating a motor or sensor to an observable action and comparing the result with the team's prediction.
- Locate a mismatch between an intended and an observed action, test a change to the construction or code, and explain whether it improved the result.
- Collaborate in a short guided sprint by agreeing a shared goal, taking turns in builder, programmer or tester roles, listening to peers and contributing to the review.
- Reflect on one success and one difficulty, propose a next improvement and demonstrate safe equipment use, persistence and respect for different ways of participating.

Level 2 ages 8 to 10 Sprints 21 to 40

By the end of Level 2, students can, with progressively reduced scaffolding:

- Decompose a robotic challenge into manageable tasks and design a sequenced algorithm using suitable representations such as pseudocode, diagrams or blocks.
- Create and revise programs using loops, conditions and variables or lists where appropriate, and justify how these structures control the robot's response to inputs.
- Integrate motors and sensors into a purposeful system, analyse measurements or recorded observations, and use them to adjust movement, thresholds or other parameters.
- Compare alternative designs or algorithms against agreed criteria, conduct structured tests, and justify an improvement using evidence rather than trial alone.
- Plan and manage a team sprint by allocating tasks, rotating roles, reporting progress and negotiating changes while ensuring that all team members can contribute.
- Present and evaluate a solution's function, usefulness and accessibility, then use peer feedback and self-assessment to identify improvements to both the product and the team's working process.

Level 3 ages 10 to 12 Sprints 41 to 60

By the end of Level 3 and the programme, students can, with increasing autonomy:

- Analyse a user or mission requirement, identify constraints and success criteria, and design a robotic solution that connects mechanical construction, sensing, actuation and algorithmic control.
- Translate familiar block-based logic into structured Python and create programs using variables, loops, conditions, lists and functions as appropriate to the task, explaining the relationship between algorithm and implementation.
- Diagnose syntax, runtime and logic errors through systematic testing and recorded observations, explaining the cause of an error and evaluating the effectiveness of its correction.
- Evaluate and optimise a program or mechanism against explicit criteria such as accuracy, reliability, clarity or task-completion time, using measurements or repeated tests to justify decisions.
- Coordinate an Agile/Scrum sprint through task decomposition, role rotation, progress checks, review and retrospective, adapting plans and resolving disagreements through reasoned discussion and shared responsibility.
- Critically evaluate a solution and its development process, communicate evidence-based conclusions, consider accessibility, safe use and environmental implications, and propose how the learning could transfer to a new problem.

Attainment is judged through the learner's contribution and demonstrated reasoning as well as the team's product. Teachers should adapt communication formats, tools and scaffolding without removing the intended competency. These are curriculum targets to be assessed through implementation, rather than claims of outcomes already achieved.

3. Curriculum Framework and Organisation

3.1. Organising frameworks

The curriculum combines a spiral progression in programming and robotics with competence-based outcomes and age-appropriate Agile/Scrum routines. The four strands in Section 3.2 organise learning across the whole programme; they are integrated within sprints rather than taught as four separate courses. The outcomes in Section 2 guide teaching and assessment, while subsequent progression maps and unit plans provide more detailed expectations.

Agile learning and adapted Scrum. Each sprint follows a recurring sequence: establish the challenge and success criteria, plan tasks and roles, build and program, test and improve, present the result, and reflect on learning and teamwork. The classroom routine can be expressed as Connect → Plan → Build → Code → Test → Improve → Share → Reflect. Short check-ins help teams identify progress and obstacles. Scrum terminology,

documentation and role responsibilities are simplified for younger learners; completing formal Scrum procedures is not an end in itself.

Competence-based spiral progression. Knowledge, practical skills and dispositions develop together through increasingly complex problems. Sequences, loops, conditions, variables and debugging recur in new contexts, with progressively greater independence and abstraction. Each sprint identifies a technical outcome, a soft-skill outcome and observable evidence. Assessment considers the individual learner's contribution and reasoning alongside the team's robotic product.

European competence references. DigComp 2.2 provides a reference for connecting programming and digital creation with problem-solving, collaboration, safety and responsible technology use. The programme also develops social-emotional and learning-to-learn capabilities through planning, feedback and reflection. These references inform curriculum design; a detailed mapping should identify the relevant competence and evidence for each outcome before formal alignment is claimed.

National curriculum connections. Schools map the three age levels to their national year groups and subject outcomes in computing or ICT, mathematics, science, technology and design, arts and language. Age bands provide the common project structure, while national grade labels and teaching time remain locally determined. Teachers record the links between selected sprints, national outcomes and assessment evidence, and provide bridging activities where prior learning differs.

Quality assurance and improvement. The project's EQAVET-inspired cycle of planning, implementation, assessment, review and revision guides curriculum improvement. Relevant EQF/NQF references concern the teacher-facing framework and professional learning, rather than a qualification level assigned to children. Classroom evidence, learner feedback, teacher reflection and pilot evaluation inform revisions to resources, scaffolding and delivery.

3.2. Curriculum structure

The 60 cumulative sprints are organised into three levels, each comprising two cycles of ten. The numbered sequence is fixed within this curriculum. Teachers may extend a sprint across lessons, vary scaffolding and provide consolidation, while retaining the intended learning progression and recording any adaptations.

Level 1 ages 6 to 8. Sprints 1–20 comprise Cycle 1 (1–10) and Cycle 2 (11–20). Concrete exploration, simple sequences, cause and effect, basic motor and sensor responses, guided testing and turn-taking establish the foundations. Visual or block-based programming and physical demonstrations are the principal entry points; the availability of Python files does not make Python an attainment requirement at this level.

Level 2 ages 8 to 10. Sprints 21–40 comprise Cycle 1 (21–30) and Cycle 2 (31–40). Learners develop loops, conditions, variables and lists where appropriate, purposeful sensor-based systems, measurement and more systematic debugging. Teams increasingly plan tasks, compare alternatives and use recorded evidence, with structured guidance and optional extension appropriate to readiness.

Level 3 ages 10 to 12. Sprints 41–60 comprise Cycle 1 (41–50) and Cycle 2 (51–60). Learners integrate mechanical design, sensors, algorithms and increasingly structured Python programming. More complex missions, reusable code, data handling and optimisation support greater autonomy in planning, testing and evaluation. Block-based examples and familiar models provide bridges into textual programming when required.

The four integrated strands

Strand 1 Computational Thinking and Programming. Develop problem decomposition, sequencing, logical reasoning, events, loops, conditions, variables, lists, modularity and debugging. At Level 1, learners predict and modify short sequences; at Level 2, they design and test structured algorithms; at Level 3, they translate familiar logic into Python and evaluate increasingly reusable and efficient solutions. Evidence includes algorithm representations, programs and explanations of revisions.

Strand 2 Robotics Systems and Physical Computing. Develop construction skills and understanding of movement, sensors, actuators, inputs, outputs and control. Level 1 emphasises safe assembly and observable responses; Level 2 combines components with measurement and calibration; Level 3 integrates mechanisms and control logic to meet mission requirements reliably. Testing records and comparisons between predicted and observed behaviour make systems understanding visible.

Strand 3 Agile Collaboration and Reflection. Develop shared goal-setting, role allocation and rotation, task planning, progress communication, review, retrospective and iteration. Learners move from guided turn-taking to negotiated responsibilities and increasingly autonomous coordination. Reflection examines both the solution and how the team worked, including accountability, constructive disagreement, persistence and responsiveness to feedback.

Strand 4 Responsible Inclusive and Real World Technology Use. Develop awareness of users, accessibility, safe equipment use, fairness, sustainability and responsible digital practices. Learners progress from respectful sharing and simple safety choices to evaluating design decisions and their effects on different users. Projects should connect with understandable real-life needs and invite consideration of limitations, resource use and appropriate handling of data.

Cross cutting themes

Inclusion and accessibility. Provide multiple entry points, accessible instructions and alternative ways to participate and demonstrate learning. Review participation within each team and adjust tools, roles, pacing or support to remove barriers.

Student agency and wellbeing. Offer bounded choices about designs, strategies and roles. Use predictable routines, manageable challenges and supportive feedback so that learners can take initiative, request help and regulate frustration.

Creativity and making. Encourage original constructions, alternative algorithms and expressive applications. Value reasoned exploration and improvement alongside successful completion.

Evidence and reflection. Use predictions, measurements, observations, test records and learner explanations to support decisions. Include individual self-assessment and team reflection in each sprint.

Communication and literacy. Develop age-appropriate technical vocabulary, diagrams, pseudocode, explanations and presentations. Support oral, written, visual and other accessible communication formats, including multilingual learners.

Safety privacy and responsible digital use. Teach careful handling of equipment and appropriate use of devices and software. If images, recordings or engagement data are used, follow the school's safeguarding, consent and data-protection procedures. Engagement information supports teacher judgement and must not be treated as a diagnosis or a direct measure of ability.

Sustainability and ethical design. Discuss durable constructions, reuse of materials, energy and resource choices, and whom a solution benefits or may exclude. Connect environmental and social questions to the robot's actual purpose and design.

For unit planning, teachers record the level and cycle, challenge, relevant strand outcomes, national curriculum links, technical and soft-skill objectives, resources, inclusion arrangements, sprint activities and assessment evidence. Cross-cutting themes are selected where they are meaningful to the task and revisited across the programme.

3.3. Pedagogical models

Teaching blends inquiry, project-based problem-solving, making, creative STEAM activity and user-centred design within the shared sprint routine. The choice of approach follows the learning objective and developmental level. Teachers retain an active role in modelling, questioning, safety and feedback while progressively transferring decisions to learners.

Inquiry and project based learning. Begin with an understandable question or challenge. Learners predict behaviour, investigate relevant mechanisms or code, create a solution and compare results with success criteria. Short inquiry tasks build prerequisite ideas; longer missions combine these ideas into a purposeful product. Review focuses on the evidence supporting a decision, not only whether the robot completed its task.

Making creative STEAM and design thinking. Use construction and programming to connect disciplinary ideas with a tangible outcome. Where a task serves a user, learners identify the need, generate alternatives, prototype, test and revise. Artistic expression, storytelling, movement and real-world contexts can strengthen engagement when they contribute to the learning goal. Unplugged sequencing, role-play and physical models prepare learners before digital implementation.

Universal Design for Learning. Plan different ways to engage with a challenge, access explanations and express understanding. Combine demonstration, images, spoken

instructions, readable code and manipulatives as appropriate. Provide adjustable supports and accessible assessment formats, while keeping the core learning intention clear. Adaptations respond to observed barriers and learner needs rather than fixed assumptions about ability.

Scaffolding at Level 1. Use worked examples, teacher think-alouds, visual task cards, short command sequences and guided role rotation. Model a prediction–test–explain routine and introduce one change at a time. Learners make small choices and explain outcomes through demonstration, pictures or speech, with close support for safe construction and collaboration.

Scaffolding at Level 2. Move from complete examples to partially completed algorithms, planning prompts, test tables and structured peer feedback. Learners choose between strategies, justify changes and take responsibility for defined tasks. Teachers use questions and targeted mini-lessons to address difficulties, then reduce prompts as competence becomes secure.

Scaffolding at Level 3. Use open-ended briefs, explicit constraints and learner-defined test plans. Support the transition to Python through comparisons with familiar block-based logic, annotated examples and progressively reduced starter code. Teams plan and evaluate more independently, with teacher conferences and focused support for debugging, accessibility and project coordination.

Responsive progression. Scaffolding is adjusted through formative assessment rather than withdrawn solely because of age or sprint number. Reintroduce models or guided practice when needed, and offer extensions through more demanding constraints, alternative mechanisms or optimisation. Reviews and retrospectives identify the next technical learning step and the next improvement in collaboration.

4. Strands and Progression Map (High-Level)

The four strands develop together through 60 cumulative sprints. Each level contains two cycles of ten sprints; concepts recur with increasing complexity and learner independence.

Strand	Level 1 ages 6–8 Sprints 1–20 Cycles 1–10 and 11–20	Level 2 ages 8–10 Sprints 21–40 Cycles 21–30 and 31–40	Level 3 ages 10–12 Sprints 41–60 Cycles 41–50 and 51–60
1 Computational Thinking and Programming	Sequence actions and predict cause and effect. Create and modify short block-based programs using events and simple repetition. Explore sensor-triggered actions and identify errors through guided testing.	Decompose problems and design algorithms using loops, conditions, variables and lists where appropriate. Compare predictions with results and debug systematically using diagrams, pseudocode and block-based code.	Translate familiar logic into Python. Combine variables, loops, conditions, lists and functions as appropriate. Diagnose syntax, runtime and logic errors; evaluate and optimise code against clear success criteria.
2 Robotics Systems and Physical Computing	Assemble simple mechanisms safely. Explore motors, light displays and sensors; connect an input or command to an observable output. Test movement and change one parameter at a time.	Combine motors and sensors in purposeful systems. Measure distance, movement or other sensor data; adjust thresholds and calibrate behaviour. Use recorded observations to compare and improve constructions.	Integrate mechanical design, sensing and control for complex missions. Select suitable components and parameters, analyse repeated test data, and improve accuracy, reliability and performance within stated constraints.
3 Agile Collaboration and Reflection	Follow a guided sprint with a shared goal. Take turns as builder, programmer or tester. Describe a result, listen to feedback and identify one improvement with teacher support.	Plan manageable tasks, rotate roles and report progress. Negotiate changes and share responsibility. Use reviews and retrospectives to evaluate the product, individual contributions and team collaboration.	Coordinate sprint planning, task allocation and progress checks with increasing autonomy. Justify decisions, resolve disagreements constructively, adapt plans and use evidence-based reviews and retrospectives to improve team performance.
4 Responsible Inclusive and Real World Technology Use	Share materials fairly, respect different ideas and follow safety routines. Recognise whom a simple robot helps. Participate through accessible roles and communicate using speech, pictures or demonstration.	Consider different users and practical needs. Compare solutions for usefulness, accessibility and resource use. Apply safe and responsible digital practices and ensure that all team members can contribute.	Evaluate accessibility, ethical implications, sustainability and safe use. Consider privacy when collecting or presenting data. Justify user-centred design choices and propose how a solution could transfer to another context.

5. Learning Outcomes by Level (Core Section)

This section specifies assessable outcomes for the four strands at each of the three curriculum levels. Codes 6, 8 and 10 indicate the starting age of each band, not a national school grade: LO 6A1 belongs to Level 1 (ages 6–8). National year-group equivalents are assigned locally. Outcomes are cumulative and assessed by the end of the relevant 20-sprint level.

Sprint lengths below are indicative planning ranges, not additional time for each strand. The four strands operate together within the same sprint. A 45-minute lesson is the reference unit; teachers should follow the individual lesson scenario and adjust time for building, accessibility, testing and reflection. A longer mission may span several sessions. Accept oral, visual, written or demonstrated evidence while recording each learner's contribution.

Level 1 ages 6 to 8

Sprints 1–20; Cycle 1: 1–10; Cycle 2: 11–20.

Guided participation with concrete models, visual instructions and frequent teacher feedback.

By the end of this level, learners can demonstrate the following outcomes. The statements are expressed in the learner's voice.

Strand A Computational Thinking and Programming

LO 6A1 I can arrange commands in a logical sequence, predict what my robot will do and compare my prediction with its actions.

LO 6A2 I can create and change a short block-based program using movement, light or sound, and use repetition when a pattern repeats.

LO 6A3 I can test my program, identify an action that does not match my plan and explain how a change improves it.

Teacher support needed: Model sequences and one change at a time; provide command cards, starter programs and oral prompts. Do not require Python.

Typical sprint length: 45 minutes for introductory component tasks; typically 90 minutes for an integrated build-and-code challenge.

Strand B Robotics Systems and Physical Computing

LO 6B1 I can build a simple mechanism safely and show how a motor changes its movement.

LO 6B2 I can program a robot to move towards a measured target and explain how changing a movement setting affects the distance.

LO 6B3 I can use a sensor to trigger an action and explain how stopping after a set movement differs from stopping when the sensor is pressed.

Teacher support needed: Demonstrate safe assembly and connections; support measuring and help learners distinguish a command from a sensor input.

Typical sprint length: 45–90 minutes within an integrated sprint; allow extra construction time when necessary.

Strand C Agile Collaboration and Reflection

LO 6C1 I can help my team agree a small goal and take a defined builder, programmer or tester role.

LO 6C2 I can listen, share equipment and report a difficulty so that my team can decide what to try next.

LO 6C3 I can contribute to the sprint review, show what we changed and name one way to improve our teamwork.

Teacher support needed: Lead brief planning and check-ins; use role cards, turn-taking cues and simple review questions.

Typical sprint length: Planning, check-ins, review and reflection are embedded in the same 45–90-minute sprint.

Strand D Responsible Inclusive and Real World Technology Use

LO 6D1 I can explain a safety choice and show how I use and put away the equipment responsibly.

LO 6D2 I can describe whom my robot could help and suggest a change that makes it easier for someone to use.

LO 6D3 I can help a teammate participate, respect different ideas and explain why we ask permission before sharing someone's picture or work.

Teacher support needed: Use concrete examples, accessible roles and adult-supervised sharing; prompt learners to notice participation barriers.

Typical sprint length: Integrated across the same 45–90-minute sprint; discuss safety before building and users during review.

Level 1 ages 6 to 8 minimum evidence

Collect a small portfolio across the level, sampling both ten-sprint cycles. The items below are the minimum evidence set for the level, not a requirement to produce every item in every sprint. Shared products must be accompanied by identifiable individual evidence.

Concepts introduced: Sequence, cause and effect, commands and parameters, introductory events and repetition, motor output, sensor input, prediction, testing, debugging and guided sprint roles.

Artifacts produced: A shared working robot or mechanism, a saved or captured block-based program, and a picture or simple sequence plan. Record each learner's role and contribution; one team artifact can support several individual demonstrations.

Performance tasks: Across the level, each learner demonstrates a programmed action, changes a movement parameter, triggers a sensor response and tests one improvement. Use a

practical comparison of programmed-distance stopping and force-sensor stopping with teacher prompts.

Reflection and logs: A brief individual oral, pictorial or written reflection identifying one success, one difficulty and one next step; a simple team review record showing a change to the robot, code or collaboration.

Level 2 ages 8 to 10

Sprints 21–40; Cycle 1: 21–30; Cycle 2: 31–40.

Structured practice with partial examples and prompts that are reduced as learners demonstrate readiness.

By the end of this level, learners can demonstrate the following outcomes. The statements are expressed in the learner's voice.

Strand A Computational Thinking and Programming

LO 8A1 I can decompose a robotic challenge and represent my algorithm with a diagram, pseudocode or a clear sequence of blocks.

LO 8A2 I can create a program using loops and conditions, and use a variable or list when the task needs stored information.

LO 8A3 I can compare predicted and actual results, trace a faulty step and justify a code change using test evidence.

Teacher support needed: Provide partial algorithms and focused demonstrations; use questions to support tracing, conditions and data structures.

Typical sprint length: Typically 90 minutes across two lessons; allow a further lesson for complex algorithms or consolidation.

Strand B Robotics Systems and Physical Computing

LO 8B1 I can combine motors and sensors so that my robot responds appropriately to an object, a line or a measured value.

LO 8B2 I can record measurements, adjust a threshold or movement setting and explain how the adjustment changes behaviour.

LO 8B3 I can compare two designs or control strategies against agreed criteria and use the results to improve reliability.

Teacher support needed: Support sensor setup, measurement and calibration; provide test tables and gradually transfer decisions to the team.

Typical sprint length: Typically 90 minutes; construction, calibration and repeated tests may extend a sprint to 135 minutes.

Strand C Agile Collaboration and Reflection

LO 8C1 I can help plan a sprint by breaking the goal into tasks, agreeing roles and checking what we need to complete.

LO 8C2 I can report progress, negotiate a change and carry out my responsibilities while helping others contribute.

LO 8C3 I can use peer feedback and a retrospective to evaluate our solution and propose a specific improvement to our teamwork.

Teacher support needed: Use planning templates, rotating roles and structured feedback prompts; intervene when participation or disagreement needs support.

Typical sprint length: Planning, check-ins, review and retrospective run within the same 90–135-minute integrated sprint.

Strand D Responsible Inclusive and Real World Technology Use

LO 8D1 I can compare how useful and accessible two solutions are for different users and justify a design choice.

LO 8D2 I can apply safe equipment and digital practices, and identify information that should not be shared without permission.

LO 8D3 I can suggest a way to reuse materials or reduce waste and explain how my team makes participation fair.

Teacher support needed: Prompt discussion of users, access, resource use and privacy; provide examples and alternative communication formats.

Typical sprint length: Integrated within the same 90–135-minute sprint, with ethical and inclusion questions in planning and review.

Level 2 ages 8 to 10 minimum evidence

Collect a small portfolio across the level, sampling both ten-sprint cycles. The items below are the minimum evidence set for the level, not a requirement to produce every item in every sprint. Shared products must be accompanied by identifiable individual evidence.

Concepts introduced or extended: Decomposition, loops, conditions and combined conditions, variables, introductory lists, sensor thresholds, calibration, data recording, structured debugging, task allocation and peer feedback.

Artifacts produced: A purposeful sensor-based robot, a saved program, an algorithm representation and a small test or measurement record. Include an example using stored information and evidence of an accessibility or resource-use decision across the level.

Performance tasks: Each learner explains an algorithm, tests a sensor-controlled response, interprets recorded results and justifies an improvement. The team compares alternatives using a stated criterion and demonstrates the revised solution.

Reflection and logs: An individual account of the learner's contribution and a debugging decision; a team plan or task record; and a review/retrospective identifying one product improvement and one collaboration improvement, informed by feedback.

Level 3 ages 10 to 12

Sprints 41–60; Cycle 1: 41–50; Cycle 2: 51–60.

Increasing independence with teacher conferences and targeted technical or accessibility support when needed.

By the end of this level, learners can demonstrate the following outcomes. The statements are expressed in the learner's voice.

Strand A Computational Thinking and Programming

LO 10A1 I can translate familiar block-based logic into Python and explain how variables, loops and conditions control the program.

LO 10A2 I can design a solution using a list to organise data and a function to reuse actions, explaining why each structure is useful.

LO 10A3 I can diagnose syntax, runtime and logic errors and evaluate a corrected program for clarity, correctness and efficiency.

Teacher support needed: Use annotated Python examples, comparisons with blocks and error-tracing prompts; reduce starter code as competence grows.

Typical sprint length: Typically 90 minutes; allow 135–180 minutes for an extended program, debugging investigation or integrated mission.

Strand B Robotics Systems and Physical Computing

LO 10B1 I can analyse a mission's requirements and integrate a mechanism, sensors and actuators into a suitable robotic solution.

LO 10B2 I can select and adjust control parameters, collect repeated measurements and explain how they support reliable performance.

LO 10B3 I can evaluate and optimise my robot against success criteria such as accuracy or completion time and justify a design trade-off.

Teacher support needed: Confer on feasibility, safe construction and test design; provide targeted guidance for calibration and data interpretation.

Typical sprint length: Typically 90–135 minutes; complex missions with attachments or optimisation may require up to 180 minutes.

Strand C Agile Collaboration and Reflection

LO 10C1 I can coordinate a sprint plan with tasks, rotating responsibilities and success criteria, and adapt it when evidence changes.

LO 10C2 I can communicate progress, explain a technical decision and help resolve disagreement through reasoned discussion.

LO 10C3 I can evaluate my contribution and our team's process using review evidence and agree a realistic improvement for the next sprint.

Teacher support needed: Coach rather than direct routine decisions; monitor equitable role rotation and support constructive conflict resolution.

Typical sprint length: Planning, check-ins, review and retrospective are embedded in the same 90–180-minute integrated sprint.

Strand D Responsible Inclusive and Real World Technology Use

LO 10D1 I can evaluate how my design meets different users' needs and implement an improvement to accessibility or usability.

LO 10D2 I can identify safety, privacy or fairness concerns in collecting and presenting project data and explain an appropriate response.

LO 10D3 I can justify a sustainable design choice, evaluate a limitation of my solution and propose how to adapt it to a new context.

Teacher support needed: Use questions about users, evidence and consequences; retain adult oversight of data collection and sharing.

Typical sprint length: Integrated within the same 90–180-minute sprint, with user, safety and data considerations revisited at review.

Level 3 ages 10 to 12 minimum evidence

Collect a small portfolio across the level, sampling both ten-sprint cycles. The items below are the minimum evidence set for the level, not a requirement to produce every item in every sprint. Shared products must be accompanied by identifiable individual evidence.

Concepts introduced or consolidated: Structured Python, variables, loops, conditional logic, functions and parameters, lists, integrated sensing and actuation, systematic error diagnosis, repeated testing, optimisation and autonomous sprint coordination.

Artifacts produced: A functioning integrated robotic system, a saved and readable Python program, an algorithm or system plan and a test record comparing results against explicit criteria. The evidence set includes list-based data handling, reusable functions and a documented user or accessibility decision.

Performance tasks: Each learner explains code and the relationship between inputs and outputs, diagnoses an error and justifies an improvement with test evidence. The team demonstrates a mission, evaluates a trade-off and proposes a transfer or adaptation to another problem.

Reflection and logs: An individual debugging and decision record; a shared sprint plan with role rotation; a review comparing outcomes with success criteria; and a retrospective identifying evidence-based next steps for technical quality and collaboration.

Assessment and progression

Judge attainment from the quality of the learner's demonstration, explanation and contribution with the support recorded. Do not infer competence solely from a team's completed robot or from engagement metrics. Revisit prerequisites or provide an extension as needed. Use the outcomes alongside Section 2 competencies and the Section 4 progression map; national grades and reporting arrangements remain locally determined.

6. Pedagogy and Learning Activities

6.1. Approaches used

Teaching combines practical investigation, purposeful projects and explicit instruction within age-adapted Agile/Scrum routines. Teachers select approaches that serve the sprint's technical and soft-skill outcomes, model unfamiliar ideas and then transfer decisions to learners. Activities connect the four strands rather than treating construction, coding, teamwork and responsible use as separate lessons.

Inquiry and problem based learning. Introduce a question that can be explored through predictions and evidence, such as how a robot can stop reliably before an obstacle. Learners identify what they need to know, investigate relevant commands or components, and compare results. Short investigations prepare learners for longer project-based missions with an observable product and agreed success criteria.

Making and creative STEAM. Learners design and construct mechanisms, program behaviour and test their ideas. Mathematics supports measurement and data interpretation; science supports motion, force and sensing; arts and language support expression and explanation. Combine disciplines where they clarify the challenge. Allow alternative designs and creative solutions while retaining a common learning intention.

Collaborative and student led learning. Small teams share a goal and rotate roles such as builder, programmer, tester and coordinator. Teachers explicitly model listening, requesting help and constructive disagreement. Learners gradually take greater responsibility for planning and testing. Bounded choices about designs, strategies and presentation formats support agency while teachers monitor participation and safety.

Unplugged learning and explicit modelling. Use command cards, physical role-play, hand-drawn flowcharts and teacher think-alouds before or alongside device work. Trace a small algorithm together, then offer partial examples and independent variations. Unplugged tasks provide access to the same reasoning goal; they complement, rather than automatically replace, later evidence of physical robot or programming competence.

Design thinking and authentic challenges. When a task serves a user, learners explore the need, generate ideas, make a prototype, test it and revise the solution. Discuss accessibility and environmental impact alongside function. Possible contexts include assistive mechanisms, package delivery and environmental indicators. Use fictional scenarios or appropriate consultation without asking children to disclose personal support needs.

Agile learning and formative feedback. Organise work into short, cumulative sprints with planning, implementation, review and retrospective. Use brief check-ins and visible progress records to make the next action clear. Feedback from robot tests, peers and teachers drives revisions; reflection addresses both the solution and the team's process. Scaffolding follows the age-level expectations in Sections 3–5.

6.2. Classroom routines

Teachers use a common routine across contributing subjects, with one shared challenge, a small set of success criteria and explicit technical and collaboration outcomes. In classroom terms, the ALPER sprint cycle is:

Connect → Plan → Build → Code → Test → Improve → Share → Reflect

This eight-step routine enacts the overarching phases of Planning, Implementation, Review and Retrospective. Connect introduces the challenge; Plan establishes the sprint; Build, Code, Test and Improve form iterative implementation; Share is the sprint review; Reflect is the retrospective. Build and Code may overlap, and Test and Improve repeat as needed. A brief team check-in occurs during implementation and after a significant obstacle. Reflection informs the next sprint plan.

Timing and organisation

The example below allocates 90 minutes: Connect 8, Plan 10, Build 15, Code 15, Test 12, Improve 12, Share 10 and Reflect 8. This is a flexible planning example, not a fixed project requirement. For two 45-minute lessons, pause at a safe point, save the program, photograph or label the build if appropriate, and record the next action. A 45-minute introductory sprint uses a smaller or prepared build; longer missions may use 135–180 minutes as indicated in Section 5. Retain a short review and retrospective even when implementation takes longer.

Connect

Suggested time: 8 minutes.

Teacher role: Introduce the context, activate prior learning and show the intended challenge without prescribing the solution.

Learner role: Ask questions, make an initial prediction and identify whom the solution helps.

Artefacts and assessment: A question or prediction; assess prior ideas through explanation or demonstration.

Inclusion prompt: Use an object, image and spoken explanation; provide vocabulary or symbol choices.

Plan

Suggested time: 10 minutes.

Teacher role: Clarify success criteria, available resources and safety; support feasible tasks and role allocation.

Learner role: Agree a shared goal, sketch an approach, divide tasks and assign rotating responsibilities.

Artefacts and assessment: A sketch or algorithm, success criteria and task/role record; assess reasoning and equitable participation.

Inclusion prompt: Offer pictorial planning cards, accessible roles and a choice of ways to record ideas.

Build

Suggested time: 15 minutes.

Teacher role: Model unfamiliar connections or mechanisms and monitor safe equipment use.

Learner role: Construct or adapt the mechanism, check connections and explain design choices.

Artefacts and assessment: A prototype or system sketch; assess the relationship between components and intended behaviour.

Inclusion prompt: Provide sorted parts, enlarged instructions, reachable workspaces or a partially assembled model.

Code

Suggested time: 15 minutes.

Teacher role: Teach a focused command or concept when needed; ask learners to trace rather than simply copy.

Learner role: Create or modify the algorithm, predict the output and save a named version.

Artefacts and assessment: A program and predicted behaviour; assess sequencing, control logic and explanation.

Inclusion prompt: Use starter blocks, annotated Python, a worked example or paired input with roles rotated.

Test

Suggested time: 12 minutes.

Teacher role: Prompt comparison with success criteria and help teams design a fair, safe test.

Learner role: Run the robot, observe or measure results and identify a mismatch or error.

Artefacts and assessment: A short test record; assess the link between the prediction, observation and diagnosis.

Inclusion prompt: Use visual test checklists and accessible recording formats; allow repeated demonstrations.

Improve

Suggested time: 12 minutes.

Teacher role: Ask what the evidence suggests and provide targeted support without taking over the decision.

Learner role: Choose a change to code or build, retest it and compare before-and-after behaviour.

Artefacts and assessment: A revised solution and recorded change; assess whether the improvement is justified by evidence.

Inclusion prompt: Offer a small set of possible changes where needed and extend through stricter constraints.

Share

Suggested time: 10 minutes.

Teacher role: Facilitate the sprint review and feedback against the agreed goal.

Learner role: Demonstrate the solution, explain a decision and a limitation, and respond to peer questions.

Artefacts and assessment: A demonstration or short presentation and feedback; assess function, reasoning and communication.

Inclusion prompt: Accept live demonstration, pictures, recording or supported speech; ensure each learner contributes.

Reflect

Suggested time: 8 minutes.

Teacher role: Lead the retrospective using supportive questions about learning and teamwork.

Learner role: Identify a success, a difficulty and one improvement; reflect on individual contribution and team participation.

Artefacts and assessment: An individual reflection and one team action for the next sprint; assess self-awareness and realistic next steps.

Inclusion prompt: Use icons, sentence starters, private responses or adult-supported communication as appropriate.

Brief check ins and cross subject continuity

Teams pause for a one- or two-minute check-in within implementation: What have we completed? What is our next task? What is blocking us? A simple To do / Doing / Done board or pictorial equivalent can support coordination. Teachers in different subjects use the same goal, vocabulary, role record and success criteria so that work continues coherently. Review examines the product and its evidence; retrospective examines how learners worked and learned.

Recording evidence without unnecessary workload

Keep a compact sprint record: goal, roles, algorithm or program, one meaningful test, the resulting change and a brief reflection. Not every stage needs a separate worksheet. Sample individual explanations and demonstrations alongside shared products, and use the existing student worksheet, self-assessment and teacher-reflection resources where suitable. Assess

participation by contribution and reasoning, not speed, amount of speech or the team's finished robot alone.

6.3. Inclusion and accessibility

Plan accessibility before the lesson, then adjust support through observation and learner feedback. Keep the core learning intention clear while varying materials, prompts, pacing and ways of demonstrating understanding. Discuss adjustments with the learner and relevant school staff where appropriate. No diagnosis should determine a fixed role, level of challenge or expectation of participation.

Multiple entry points and scaffolding. Offer the same challenge through concrete demonstrations, visual sequences, partial builds, starter code or a more open brief. At ages 6–8, emphasise modelling, short instructions and concrete choices; at ages 8–10, use partial algorithms and test prompts; at ages 10–12, reduce starter code and encourage independent test plans. Reinstate support whenever readiness or task demands require it.

Language literacy and dyslexia support. Use concise instructions, clear layouts, numbered steps, familiar vocabulary and diagrams that carry meaning alongside text. Explain technical terms with examples and offer bilingual glossaries where useful. Allow instructions to be read aloud or accessed through available assistive tools. Avoid unnecessary copying; accept speech, diagrams and demonstration when reading or writing fluency is not the assessed outcome.

Differentiated roles with equitable responsibility. Agree roles with learners and rotate them across tasks and sprints. Separate intellectual decisions from a particular physical action: a learner may direct a construction or explain a program while a partner handles parts or enters commands. Ensure that supported roles retain genuine decision-making and prevent permanent assignment of the most confident learner as programmer or another learner as observer.

Low technology and unplugged access. Use paper algorithms, floor grids, physical models and role-play when devices are limited or learners need preparation. Organise shared-device turns or stations so that every learner later demonstrates the required digital or physical skill. If an activity is completed only in simulation or unplugged form, record that evidence accurately and plan access to a robot where the outcome requires it.

Sensory movement and attention considerations. Provide predictable routines, visible stage cues, manageable task steps and agreed check-in points. Offer movement breaks and quieter working options where feasible, and avoid unnecessary noise or flashing displays. Arrange safe movement space, reachable materials and flexible seating. Adjust these supports to observed needs and preferences rather than assuming the same arrangement suits every learner with ADHD or a sensory need.

Communication differences. Use explicit turn-taking, visual role cards, extra processing time and alternatives to speaking in front of the class. Accept pointing, typed responses, communication boards, supported speech or other established communication methods.

Invite questions without requiring eye contact, and check understanding through actions or explanation rather than silence or verbal fluency.

Assistive and adaptive options. Use available adapted input devices, screen magnification, text-to-speech, keyboard access or stable building supports where compatible with the task. Check physical and software access in advance with the learner and specialist staff. Preserve learner ownership of decisions when a peer or adult assists with input or construction. Follow school procedures for safe use of equipment and assistive resources.

Extension pathways. Increase challenge through more precise success criteria, additional users, alternative mechanisms, more complex conditions, reusable functions or stronger evidence of optimisation. Ask learners to compare approaches and justify trade-offs. Extensions should deepen reasoning without requiring faster completion, excessive extra work or routine replacement of teacher support for classmates.

Safe evidence and responsive teaching. Collect only evidence needed for learning and follow school procedures for permissions, recordings and data handling. Provide alternatives to being photographed or recorded. Engagement indicators can inform a conversation about access and support; they do not establish understanding, disability or motivation on their own. Review who had opportunities to decide, build, code, test and share, then adapt the next sprint accordingly.

7. Assessment Framework

7.1. Assessment Values

Assessment supports improvement and provides a defensible account of what each learner can explain, create, test and evaluate. Use the Section 5 outcome codes to connect tasks and evidence to the four strands, and the Section 4 progression map to interpret expectations at ages 6–8, 8–10 and 10–12. Assess against the relevant level's outcomes rather than comparing pupils across age groups.

Balanced evidence. Assess conceptual knowledge through prediction and explanation; practical skills through construction, programming and testing; and collaboration, communication and responsible decisions through observed actions and reflection. A working robot is valuable evidence but cannot alone demonstrate understanding, individual contribution or ethical reasoning. Equally, an incomplete product may contain strong evidence of reasoning and improvement.

Dispositions and values. Recognise observable persistence, responsiveness to feedback, shared responsibility and care for users and resources. Describe actions and strategies rather than grading personality, confidence, enthusiasm or compliance. A learner may demonstrate collaboration through supported communication or careful testing without speaking frequently or taking a leadership role.

Inclusion and diversity. Keep the assessed competency stable while adjusting access, equipment, communication format, time and scaffolding. Accept oral, visual, practical, typed or assisted responses as appropriate, and record support used. Assistive access is not a reason

to lower attainment. Distinguish assistance that enables participation from help that supplies the reasoning being assessed; seek a further demonstration when evidence is unclear.

Individual and shared achievement. Use team products alongside identifiable individual explanations, demonstrations and reflections. Rotate opportunities to build, code, test and review so that attainment is not inferred from a dominant teammate's work. Record insufficient opportunity separately from an assessed difficulty. Engagement analytics, speed of completion and volume of speech are not proxies for learning.

7.2. Assessment types and use of evidence

Use a small, purposeful combination of diagnostic, formative and summative evidence. Establish prior understanding at the beginning of a level or new concept; gather brief evidence during sprints; review progress at the end of each ten-sprint cycle; and make an end-of-level judgement after 20 sprints. Assessment should fit within the classroom routine, not become a separate task after every activity.

Quizzes and knowledge checks. Use short predictions, code-tracing questions, matching tasks or oral explanations to uncover understanding before and during teaching. A brief end-of-cycle check can contribute summative evidence. Include explanation or application where appropriate, and offer accessible response formats when reading fluency is not the intended competency.

Performance tasks and projects. Observe learners plan, construct, program, test and improve a robotic solution. During a sprint, use errors and decisions formatively; at cycle or level end, judge a demonstration against published criteria. Ask each learner to explain or modify a relevant part of the solution. Assess targeted outcomes rather than rewarding model complexity or appearance unless these are explicit criteria.

Portfolios. Keep a selective collection of programs, algorithm sketches, test records, design decisions and reflections from both cycles. Use drafts and revisions to show progress formatively and a small final selection to support summative judgement. Follow the level-specific minimum evidence in Section 5; do not require every artifact in every sprint.

Observations and checkpoint questions. During Build, Code, Test and Improve, sample individuals using a short checklist or note linked to an outcome code. Ask questions such as "What do you predict?", "What caused the mismatch?" or "Why is this change better?" Observe responsibilities, reasoning and use of evidence, then provide a prompt or mini-lesson. Rotate the focus so every learner has an opportunity to demonstrate learning.

Reflection logs. Use brief oral, pictorial or written records after meaningful tests and retrospectives. Ask learners to identify a difficulty, the evidence they used, the change they made and a next step. Treat logs primarily as formative evidence; triangulate claims with the program, test record or demonstration rather than grading the sophistication of writing.

Peer and self assessment. Introduce age-appropriate criteria and model specific, respectful feedback. Learners compare their work with the goal, identify evidence and choose an improvement. Peer ratings do not determine another learner's grade or status; teachers moderate conclusions and check that feedback is fair and relevant.

Teacher reflection. After selected sprints, record which concepts or routines were difficult, whose participation was limited and which scaffolds helped. Use these notes to adapt teaching and resources. Teacher reflection evaluates implementation and access; it is not itself evidence that a pupil has attained an outcome.

Team reflection. At the retrospective, compare planned and completed tasks, examine role distribution and identify one achievable change to the working process. Include contributions from every learner through accessible formats. Keep this distinct from the review, which examines the product and success criteria.

Weightings and reporting decisions. The shared curriculum does not prescribe percentage weightings or a single overall score. Report a profile across all four strands so that technical achievement cannot conceal unmet collaboration or responsible-use outcomes. Where local policy requires marks or weights, agree and communicate them before assessment, map them to outcomes and avoid counting the same evidence repeatedly. Retain the strand profile alongside any required aggregate.

7.3. Rubrics

Use the common rubric categories below with task-specific success criteria and the relevant Section 5 outcomes. The four descriptors are interpreted at the learner's age level: "Secure" at ages 6–8 includes the guided support specified for that level, while older learners demonstrate progressively greater independence. "Advanced" means deeper or more transferable achievement within the level; it does not require completing older pupils' content.

Common rubric descriptors

Category	Beginning	Developing	Secure	Advanced
Programming and algorithmic reasoning	Recognises some commands but cannot yet explain their sequence or effect.	Creates a partly effective algorithm and explains some choices with prompts.	Creates and explains a suitable algorithm using the level's expected structures.	Compares algorithms and justifies a clearer or more effective approach.
Robotics testing and improvement	Connects some components; predictions and tests need substantial guidance.	Builds a partly effective system and records tests with support.	Demonstrates intended behaviour and uses test evidence to justify a change.	Designs systematic comparisons and evaluates reliability, calibration or trade-offs.
Communication and explanation	Shares an observation but links to the task are not yet clear.	Describes steps or results with prompts and emerging technical vocabulary.	Explains decisions and evidence clearly through an accessible communication format.	Adapts explanation to the audience and responds thoughtfully to questions.

Category	Beginning	Developing	Secure	Advanced
Ethical reasoning and responsible use	Identifies a rule or user need with substantial prompting.	Applies some safe or inclusive choices and gives a simple reason.	Justifies a relevant safety, accessibility, fairness or sustainability decision.	Weighs competing needs or consequences and improves a design responsibly.
Teamwork and reflection	Participates in a role with frequent support; next steps need guidance.	Carries out tasks and identifies a useful contribution or improvement with prompts.	Fulfils responsibilities, listens and uses reflection to improve shared work.	Supports equitable participation and helps the team adapt using reflective evidence.

Use “Not yet assessed” when evidence or opportunity is missing; do not automatically label the learner “Beginning”. Apply only categories relevant to the task, while ensuring that the evidence set across each cycle covers all four strands. Record support and opportunity separately from the achievement descriptor.

Interpreting evidence across ages

At ages 6–8, accept concrete demonstrations, pictures and brief supported explanations; for example, show how a changed command affects movement. At ages 8–10, look for an explained algorithm, recorded tests and a justified change. At ages 10–12, expect increasingly independent diagnosis, comparison and justified optimisation, alongside reasoned consideration of users and team process. Specialist vocabulary and written fluency are assessed only where explicitly part of the outcome.

Moderation and consistency

Before a cycle, teachers agree what the targeted outcomes look like and review a small number of sample artifacts or demonstrations. At cycle end, compare selected judgements, discuss differences and note any revised interpretation. Use several complementary observations where possible; neither a single failed attempt nor one polished team product should determine the level judgement. Allow further evidence after feedback.

7.4. Feedback and reporting

Feedback should identify the goal, what the evidence shows and one manageable next action. Give it while learners can use it, then provide time to revise or demonstrate again. Use clear, age-appropriate language and discuss technical reasoning and collaboration without attaching fixed labels to the learner.

During the sprint. Offer brief questions or specific comments during implementation. In Share, compare the demonstrated solution with success criteria and invite focused peer feedback. In Reflect, ask what the learner or team will change next. Record only significant evidence and decisions rather than every interaction.

Progress tracking. Maintain a concise learner record with date, sprint and cycle, Section 5 outcome code, evidence reference, current descriptor, support or access arrangements,

feedback and next review opportunity. A shared artifact may appear in several records, but each entry identifies the learner's own demonstrated contribution. Use the most recent relevant evidence alongside earlier work to show growth.

Cycle and level reports. At the end of Sprints 10, 20, 30, 40, 50 and 60, review the evidence and provide a short strand profile. After Sprints 20, 40 and 60, summarise end-of-level attainment against the relevant outcomes. Report strengths, a specific example, a priority next step and any outcome not yet assessed. Do not treat an end-of-cycle review as proof that every end-of-level outcome must already be secure.

Communication with learners and families. Share criteria before assessment and use accessible examples during learner conferences. Explain progress to families in plain language, with a small selection of evidence where permitted. Include learner voice and describe support constructively. Team achievements can be celebrated publicly, but individual attainment, access arrangements and sensitive information are communicated privately.

Example of an evidence-based comment. "LO 8B2: You used the test readings to adjust the stopping threshold and explained why the robot stopped earlier. Next, repeat the test and compare the results to check whether the change is reliable." For teamwork, a next step might be to rotate who leads the test and ensure that each member records or explains a result.

Responsive teaching and responsible records. Use class patterns to plan reteaching, regrouping, additional access or extensions. Keep assessment evidence proportionate and follow school procedures for access, retention and sharing. Engagement information may prompt a support conversation, but does not establish attainment or diagnose a learning need. Record adaptations and reassess when inadequate access prevented a fair demonstration.

8. Resources, Tools, and Safety (if relevant)

8.1. Required resources (minimum viable)

All 60 ALPER-EU agile sprints were developed using the LEGO SPIKE Prime Set as the minimum viable robotics platform for all three age groups: Sprints 1–20 for ages 6–8, Sprints 21–40 for ages 8–10, and Sprints 41–60 for ages 10–12. Progression is achieved through task complexity, programming structures and teacher support, not through a requirement to purchase a different set at each level. The LEGO SPIKE Prime Expansion Set is useful but is not required.

Core robotics equipment. Provide access to a complete, working SPIKE Prime Set, including its Hub, motors, sensors, building elements and the charging or connection accessories needed for the planned activities. Small teams should have scheduled access to a set and programming device; one set per small team is the preferred classroom arrangement. Where sets are shared between teams, use planned rotations and ensure that every learner can build, program and test. Check the specific sprint's component needs before the lesson.

Programming device and software. Provide a school-managed computer or tablet with a suitable SPIKE programming environment, tested with the school's device, Hub and

connection method. Confirm that the chosen setup can open the supplied projects and support the programming mode needed for the lesson. Early levels primarily use visual or block-based programming; the oldest level requires access to the Python activities. Check saving, file transfer, permissions and accessibility before delivery rather than relying on a device's general availability.

Digital teaching materials. Make the relevant ALPER-EU sprint resources available to teachers and learners: lesson scenarios, teacher guidance, student worksheets, self-assessment and reflection sheets, construction instructions, code examples and supporting images or videos where supplied. Use the matching Scratch/block-based or Python project files as appropriate. Digital construction models and code screenshots can support preparation; their presence does not make a separate modelling application a compulsory learner tool.

Files connectivity and preparation. Store accessible working copies of the resources, starter programs and any required sample data in an agreed school location. Prepare the chosen software, connections and any downloads in advance. Internet access is needed when a selected activity depends on an online resource or external data; prepare a saved example or teacher-provided data set where it can meet the same learning objective. Do not assume that every sprint needs live connectivity, an individual account or a personal device.

Offline materials and workspace. Provide clear worktables, organised parts trays, safe test space, paper, pencils, visual role cards and a simple task board. Measurement and route activities may require rulers or measuring tapes, removable floor or table markers, printed lines, targets or other inexpensive materials identified in the lesson scenario. Prepare accessible instructions and ways to record predictions, tests and reflections. Keep moving robots and loose materials away from walkways.

Teacher readiness and equitable access. Before each sprint, confirm equipment completeness, Hub charge, motor and sensor connections, program loading and a simple test run. Prepare an example or partial build where needed, especially for younger learners. Agree how teams save work and share equipment, and provide school-based access for any essential preparation. The assessment outcomes must not depend on families owning robotics kits or purchasing optional resources.

8.2. Recommended resources (enhancements)

Optional Expansion Set. The LEGO SPIKE Prime Expansion Set can broaden construction choices, support alternative mechanisms and enrich extension challenges. It is an enhancement rather than a condition of participation, progression or assessment. Teachers should identify a valid route using the core SPIKE Prime Set for every required task; if an optional redesign uses expansion elements, offer an equivalent core-set challenge assessing the same outcome.

Additional classroom resources. Useful enhancements include spare or additional core sets, replacement components, extra charging provision, larger test surfaces and reusable storage. These can reduce waiting and support parallel testing, but are not mandatory curriculum

requirements. Additional measurement tools or presentation equipment can make demonstrations and comparison of results easier.

Digital and accessibility enhancements. A classroom display, document camera or optional digital modelling/viewing tool can support shared analysis and demonstrations. Available assistive resources may include magnification, text-to-speech, adapted input, visual timers or construction supports. Select tools in response to an identified teaching or access need and check compatibility. An optional resource may become essential for a particular learner's access; plan that support without presenting it as a universal equipment requirement.

Selection and purchasing decisions. Prioritise reliable access to the core set, suitable programming devices, maintenance and teacher preparation before optional purchases. Evaluate an enhancement for its learning purpose, accessibility, ease of use and ongoing support. Keep extra hardware and paid services separate from the minimum viable resource list, and avoid making the sophistication of a model or the cost of its components an assessment advantage.

8.3. Safety, privacy, safeguarding

Teachers establish clear routines for safe, fair and responsible use before learners handle equipment. Apply the school's existing safeguarding, device-management and data-handling procedures, together with the relevant equipment instructions. Adapt supervision to the learners, mechanism and setting; younger learners need particularly explicit demonstrations and close support.

Physical safety. Check for damaged parts, cables or connections and remove faulty equipment from use. Keep small components controlled, prevent mouthing of parts, and keep hands, hair and clothing away from moving mechanisms. Stop a robot before adjusting its build or handling moving attachments. Use a clear test zone, conservative movement settings and an agreed stop procedure. Avoid launching objects towards people, testing near edges or obstructing exits and accessible routes.

Power charging and care. Use charging equipment and connection methods intended for the device and follow the manufacturer's handling instructions. Adults oversee charging arrangements, inspect cables and keep liquids away from electronics. Do not improvise electrical connections or attempt classroom battery repairs. Switch off, save work and store components safely at the end of the session; report faults to the responsible adult.

Reliability and controlled testing. Verify Hub connections, program selection, sensor readings and the expected stopping behaviour before a moving test. Make one manageable change at a time where possible, repeat tests and allow space for unexpected behaviour. A sensor or program can fail; adult supervision and physical test boundaries remain necessary. Prepare a fallback demonstration or unplugged task when equipment fails, recording which practical outcomes still need direct evidence.

Tool choices and accounts. Use school-approved software and storage, with access permissions appropriate to the activity. Test required project files and accessibility functions before lessons. Avoid introducing unnecessary services, personal accounts or public profiles.

Teachers control links, shared devices and any online communication; learners should know how to report an unexpected message, unsuitable content or a technical problem.

Learning evidence and privacy. Collect only the evidence needed for teaching and assessment, such as code, test records and brief reflections. Use school procedures for permissions, access, retention and sharing. Remove unnecessary personal details from file names and presentations, secure shared-device sessions and communicate individual attainment privately. Offer equivalent evidence routes when a learner cannot be photographed or recorded. Do not require public posting as proof of participation.

Engagement monitoring. The project's engagement-assessment tools are complementary evaluation resources, not equipment required to run the 60 robotics sprints. Any camera-based or other monitoring must be separately planned through the school's approved safeguarding and data-handling process, with clear information about purpose, access and alternatives. Do not treat gaze, movement or attention indicators as a diagnosis, a direct measure of learning, or a basis for awarding attainment grades.

Classroom fairness and safeguarding. Rotate meaningful roles and device access, agree turn-taking and check that each learner has opportunities to make decisions and demonstrate understanding. Do not allocate ability, responsibility or leadership by gender, disability or prior kit ownership. Model respectful feedback and intervene when exclusion, ridicule or unsafe behaviour occurs. Adapt seating, test routes, communication and sensory conditions so learners can participate safely without disclosing personal needs to peers.

Shared routines and incident response. Begin with a short safety and access check; confirm the test area, stop signal and who may start the robot. During work, pause when a fault, hazard or access barrier is identified. End with saving, switching off, counting or sorting parts and clearing the space. Learners report injuries, equipment damage and digital concerns to the teacher immediately; staff follow the school's reporting and escalation procedures.

9. Cross-Curricular Integration Plan

9.1. Where collaboration fits

Informatics or computing is a coordinating subject for all 60 sprints: each develops algorithmic reasoning, programming, systems understanding or the communication needed to design an algorithm. Science and mathematics contribute most directly to movement, sensing, measurement and data investigations. Technology/design, arts, language, physical education, environmental education and citizenship provide further meaningful links. In primary schools these contributions may be coordinated by one class teacher or shared across specialist teachers.

Basis and status of the unit links. The map below covers all 60 sprints using the previously verified lesson scenarios and summaries. It distinguishes the established lesson focus from proposed cross-curricular teaching uses; it is not a claim that every listed subject is explicitly named in each source lesson or formally mapped to every national curriculum. Original ZIP files were not available for a fresh reinspection during preparation of this section. Schools

should confirm local subject outcomes and review the relevant sprint resources before delivery.

Age levels and planning. Use Sprints 1–20 for ages 6–8, Sprints 21–40 for ages 8–10 and Sprints 41–60 for ages 10–12. Each band retains two cycles of ten sprints. Scale vocabulary, calculations and evidence to developmental readiness: early learners use concrete comparison and demonstration; middle learners record and interpret measurements; older learners justify algorithms and analyse results more independently. An optional subject extension must not become an unintended prerequisite.

Coordinating subject contributions. Agree the shared challenge, participating subjects, timetable, technical and soft-skill outcomes, resources and evidence before teaching. Informatics can lead algorithm design and debugging; science can lead predictions and fair testing; mathematics can lead measurement and data interpretation. Language and arts can support representation and communication, while physical education, citizenship or environmental education contribute where the context is authentic. Collaboration need not involve every subject in every sprint.

Reading the map. The subject links shown are additional to the informatics contribution common to the programme. Technology/design includes construction and evaluation of mechanisms. Personal/social education includes communication, collaboration and responsibility. Suggested shared tasks are planning options derived from the lesson focus; they do not replace the supplied sprint sequence or assessment outcomes.

Sprint links across the three age levels

Sprint and age band	Lesson focus	Proposed subject links	Suggested shared task
1 Ages 6–8	Light Matrix	Mathematics; art; language	Represent a symbol or number with a light pattern and explain the sequence.
2 Ages 6–8	The Motor	Science; mathematics; technology/design	Compare motor settings, direction and rotations using observations or simple measurements.
3 Ages 6–8	Colour Sensor	Science; art; mathematics	Classify coloured objects and explain the response triggered by each colour.
4 Ages 6–8	Distance Sensor	Science; mathematics	Compare measured distances with sensor readings and test an obstacle threshold.
5 Ages 6–8	Force Sensor	Science; technology/design	Compare touch and pressure responses and explain a safe start/stop mechanism.
6 Ages 6–8	Gyro Sensor	Science; mathematics; physical education	Explore orientation and tilt; connect a physical movement with a programmed response.

Sprint and age band	Lesson focus	Proposed subject links	Suggested shared task
7 Ages 6–8	Pass the Brick	Technology/design; personal/social education	Coordinate a transfer mechanism and evaluate how teams communicate and synchronise actions.
8 Ages 6–8	Ideas the LEGO Way	Art; language; technology/design	Describe a construction clearly, reconstruct a peer's idea and discuss differences.
9 Ages 6–8	What Is This	Technology/design; art; language	Repurpose a model for a different need and explain the design choice.
10 Ages 6–8	Going the Distance	Mathematics; science	Compare wheel-rotation distance estimates with touch-sensor stopping through practical tests.
11 Ages 6–8	Goal	Physical education; science; mathematics	Test how movement settings affect a shot and record successful attempts.
12 Ages 6–8	Help	Citizenship; personal/social education; technology/design	Identify an everyday need and justify whom a proposed robotic solution helps.
13 Ages 6–8	Hopper Race	Science; mathematics; physical education	Compare wheel-less mechanisms and measure performance over an agreed route.
14 Ages 6–8	Super Cleanup	Environmental education; technology/design; mathematics	Compare collection mechanisms using an agreed count, time or effectiveness criterion.
15 Ages 6–8	Design for Someone	Citizenship; technology/design; personal/social education	Design an assistive mechanism and explain an accessibility decision respectfully.
16 Ages 6–8	Broken	Science; technology/design; language	Investigate a mechanical fault and describe the evidence supporting a repair.
17 Ages 6–8	Break Dance	Music; physical education; art; mathematics	Create repeating movement patterns and evaluate timing against a chosen rhythm.
18 Ages 6–8	Repeat 5 Times	Physical education; mathematics	Represent exercise repetitions with a loop and compare commanded and observed counts.
19 Ages 6–8	Rain or Shine	Science; environmental education; art	Model a weather-related response and distinguish the model's input from actual weather measurement.

Sprint and age band	Lesson focus	Proposed subject links	Suggested shared task
20 Ages 6–8	Place Your Order	Mathematics; technology/design; social studies	Plan a delivery route and compare distance, movement settings and completion time.
21 Ages 8–10	Out of Order	Technology/design; mathematics	Debug a sorting process, classify its errors and test revised rules.
22 Ages 8–10	Driving Around	Mathematics; science	Relate rotations or movement time to distance and compare repeated route tests.
23 Ages 8–10	Playing with Objects	Science; technology/design	Use force and distance inputs to explain autonomous interaction with objects.
24 Ages 8–10	Reacting to Lines	Science; mathematics	Compare surface brightness readings and test conditions for following a line.
25 Ages 8–10	Creative Problem Solving	Technology/design; mathematics; personal/social education	Decompose a mission, compare strategies and justify a team solution.
26 Ages 8–10	Back to Back	Language; personal/social education; technology/design	Give precise construction instructions and revise them after a partner's attempt.
27 Ages 8–10	Keep It Safe	Citizenship; technology/design	Model access control and discuss why a mechanism allows or refuses entry.
28 Ages 8–10	Keep It Really Safe	Mathematics; citizenship; technology/design	Represent combined conditions and test which input combinations open the model vault.
29 Ages 8–10	Automate It	Technology/design; mathematics; social studies	Model package selection and transport, explaining rules and movement decisions.
30 Ages 8–10	Track Your Packages	Mathematics; geography; technology/design	Use coordinates and spatial instructions to locate packages on a model route.

Sprint and age band	Lesson focus	Proposed subject links	Suggested shared task
31 Ages 8–10	Code Your Moves	Language; physical education; music	Write movement pseudocode and test whether another learner or robot can follow it.
32 Ages 8–10	Wind Speed	Science; geography; mathematics	Interpret supplied wind-speed data and represent categories with a mechanical indicator.
33 Ages 8–10	Veggie Love	Science; environmental education; mathematics	Use rainfall or forecast examples to justify a model watering decision.
34 Ages 8–10	Brain Game	Mathematics; personal/social education	Store colour sequences, compare game results and discuss fair testing of performance.
35 Ages 8–10	The Coach	Physical education; mathematics; language	Design a training sequence and explain how lists or loops organise instructions.
36 Ages 8–10	Stretch with Data	Physical education; science; mathematics	Relate tilt and movement to recorded graphs; compare patterns across trials.
37 Ages 8–10	This Is Uphill	Science; mathematics; physical education	Compare measurements on different inclines and discuss force and controlled testing.
38 Ages 8–10	Time for Squat Jumps	Physical education; science; mathematics	Record height data; interpret graphs and introduce potential-energy reasoning when appropriate.
39 Ages 8–10	Watch Your Steps	Physical education; mathematics; science	Compare counted steps and estimated distance; evaluate consistency across trials.
40 Ages 8–10	Aim for It	Science; mathematics; physical education	Investigate motion data and relationships between speed, mass and kinetic energy at an appropriate level.
41 Ages 10–12	The Obstacle Course	Science; technology/design; physical education	Build moving obstacles and explain energy transfer, friction and changes in movement.
42 Ages 10–12	Advanced Driving Base	Technology/design; mathematics; personal/social education	Assemble coordinated subassemblies and evaluate the resulting platform's reliability.

Sprint and age band	Lesson focus	Proposed subject links	Suggested shared task
43 Ages 10–12	My Code Our Program	Technology/design; language; personal/social education	Combine reusable subroutines and explain interfaces between team members' code.
44 Ages 10–12	Time for an Upgrade	Technology/design; science	Design attachments, test their mechanical operation and justify parameter changes.
45 Ages 10–12	Mission Ready	Mathematics; technology/design; personal/social education	Plan mission strategy and use test evidence to improve reliable completion.
46 Ages 10–12	Start Sensing	Science; technology/design	Trace force-sensor conditions and test when a Python-controlled motor starts or stops.
47 Ages 10–12	Charging Rhino	Science; mathematics	Compare movement settings and contact stopping using safe, low-speed trials.
48 Ages 10–12	Moving Motors	Music; physical education; mathematics	Compare sequential and simultaneous movement and create a coordinated motor pattern.
49 Ages 10–12	New Moves with Motors	Mathematics; science	Compare relative rotation and absolute position while diagnosing code errors.
50 Ages 10–12	Cart Control	Science; technology/design; mathematics	Test sensor-based obstacle stopping and compare thresholds for a model delivery cart.
51 Ages 10–12	Hopper Run	Science; technology/design	Relate paired-motor control to hopping mechanics and compare movement settings.
52 Ages 10–12	Race Day	Mathematics; physical education; technology/design	Design a multi-step route and evaluate accuracy and completion time.
53 Ages 10–12	Warm Up Loop with Leo	Physical education; mathematics	Compare repeated exercise commands with a for loop and explain the repetition count.

Sprint and age band	Lesson focus	Proposed subject links	Suggested shared task
54 Ages 10–12	Setting Conditions for Yoga	Physical education; mathematics	Trace a while loop, counter and stopping condition in a model coaching routine.
55 Ages 10–12	Controlling Motion with Tilt	Science; language; art	Create an interactive story using gestures, orientation, conditions, sound and images.
56 Ages 10–12	Guess Which Color	Science; mathematics; music	Classify colour inputs, create a decision flowchart and link each case to a sound.
57 Ages 10–12	Break Dancer Break Down	Music; physical education; language	Diagnose dance-program errors and explain corrections using an error log.
58 Ages 10–12	Score	Mathematics; physical education; technology/design	Program scoring, store results and compare scores across agreed game rounds.
59 Ages 10–12	Clean Indicator	Environmental education; science; mathematics	Use colour-coded cleanliness scenarios and functions; explain that the model does not measure actual pollution.
60 Ages 10–12	Mind Games	Mathematics; personal/social education	Use lists for memory-game sequences and compare results under consistent conditions.

9.2. Shared tasks / shared rubrics

Subject teachers or the class teacher plan one coherent task with a shared product and complementary evidence, rather than repeating the same robotics activity in separate subjects. Use the same learning goal, sprint number, success criteria and evidence references across the team. Subject-specific criteria may be added, but they must be visible to learners before assessment and suitable for the age group.

Example for ages 6–8. For Sprint 10, Going the Distance, informatics contributes movement commands and comparison of stopping strategies; mathematics contributes measured distance and concrete comparison; science contributes observation of movement and contact. The shared task is a robot that moves towards a target and demonstrates two ways of stopping. Use a simple prediction, program, practical test and oral or pictorial explanation, linked to LO 6B2 and LO 6B3.

Example for ages 8–10. For Sprint 36, Stretch with Data, informatics contributes sensor data collection, mathematics contributes graphs and comparisons, and physical education or science contributes purposeful movement and orientation. The shared task relates a movement pattern to its recorded data. Use a common program or setup, a small data record

and the learner's explanation, linked to LO 8B2 and relevant communication and collaboration outcomes. Do not equate movement speed or physical performance with computing attainment.

Example for ages 10–12. For Sprint 58, Score, informatics contributes conditions, variables and stored results; mathematics contributes analysis of repeated scores; physical education or game design contributes fair rules and an accessible playing arrangement. The shared task is a functioning scoring game with a documented test. Use code, a results record, a demonstration and an individual explanation, drawing on LO 10A2, LO 10B2 and responsible-use outcomes where applicable.

Shared routines and artifacts. Keep the Connect → Plan → Build → Code → Test → Improve → Share → Reflect routine consistent across subjects. Agree task allocation, rotating roles, version naming and a compact portfolio record. A sketch, program, measurement table, graph, demonstration or retrospective may serve more than one subject when the associated outcomes are identified. Avoid requesting duplicate logs or presentations simply because several teachers contribute.

Common rubric categories. Use Section 7's categories: programming and algorithmic reasoning; robotics testing and improvement; communication and explanation; ethical reasoning and responsible use; and teamwork and reflection. Keep the Beginning, Developing, Secure and Advanced descriptors consistent within the relevant age level. Apply only categories supported by the task, and record missing opportunity as Not yet assessed rather than as low attainment.

Subject specific additions. Teachers may add criteria for a genuine disciplinary goal: measurement accuracy and graph interpretation in mathematics, prediction and controlled comparison in science, clarity of explanation in language, or rhythm and expressive intention in arts. Assess these separately and avoid allowing spelling, physical ability or artistic polish to reduce a computing judgement when they are not part of its outcome. Do not claim formal national alignment until the relevant outcomes have been mapped locally.

Individual evidence and fairness. Distinguish individual understanding from the shared team product through short explanations, demonstrations and role records. Use the same access adjustments and outcome expectations across contributing subjects. Provide alternative communication and physical participation routes without removing the intended reasoning. Extra sets, expansion elements or home resources must not create an assessment advantage.

Coordination moderation and reporting. Identify a coordinating teacher, contributing colleagues, planned lesson time and a common storage location. Review a small sample of evidence together at each ten-sprint checkpoint to calibrate judgements and agree next steps. Report one coherent learner profile with subject contributions where required, rather than counting the same evidence repeatedly towards an aggregate mark. Inform the next sprint through both product-review findings and team-retrospective actions.

10. Teacher Professional Development and Implementation

10.1. Teacher PD plan

The recommended ALPER-EU delivery model is co-teaching by an IT teacher and the learners' classroom teacher. The IT teacher leads technical instruction, programming and whole-class demonstrations. The classroom teacher provides individual and small-group support, facilitates inclusion and collaboration, and contributes to assessment. Both teachers jointly plan and review learning. This arrangement reduces the need to interrupt whole-class instruction whenever an individual learner or team needs assistance.

Starting assumptions. Python proficiency must not be assumed: few teachers may have prior experience, including teachers with an IT specialism. Technical readiness is established through practical rehearsal rather than job title or self-reported confidence alone. Professional development should prepare the two teachers for complementary responsibilities while gradually developing both teachers' understanding of the curriculum.

Joint onboarding and needs review. Before delivery, the pair reviews the four strands, outcomes and assessment framework, available equipment, learner access needs and timetable. Confirm the age allocation: Sprints 1–20 for ages 6–8, 21–40 for ages 8–10 and 41–60 for ages 10–12, with two cycles of ten per level. Identify separate development needs in programming, robotics, facilitation, inclusion and assessment, then agree training and support priorities.

IT teacher technical preparation. The lead teacher practises building and connecting SPIKE Prime models, opening and saving supplied projects, controlling motors and sensors, and diagnosing common faults using the school's actual devices. Preparation for early and middle levels focuses on block-based programming and explaining robot behaviour. Before the oldest level, provide a dedicated Python pathway covering program structure, variables, loops, conditions, lists, functions, sensor input and systematic error diagnosis. Use worked examples, code tracing and progressively less complete starter programs.

Python rehearsal and readiness. Before leading a Python sprint, the IT teacher runs the supplied program, explains the key logic, makes a purposeful modification, identifies and corrects representative errors, and prepares likely learner questions. Access to annotated examples and a technical mentor or approved asynchronous support should be arranged where available. If readiness is incomplete, schedule further preparation or qualified support; distributing a ready-made program is not a substitute for being able to explain and troubleshoot it.

Classroom teacher preparation. The classroom teacher learns the sprint's purpose, relevant commands or concepts, expected results and common difficulties sufficiently to provide first-line support. Training emphasises visual instructions, role rotation, communication, differentiated scaffolds, emotional regulation, classroom routines and evidence collection. Basic technical confidence should grow over time, but independent Python expertise is not a

prerequisite for the supporting role. Unresolved conceptual or code difficulties are referred to the IT teacher without losing the team's place in the activity.

Rehearsing co teaching. The pair rehearses a short adult learning sprint using Connect → Plan → Build → Code → Test → Improve → Share → Reflect. Practise how the IT teacher continues instruction while the classroom teacher helps a learner, how requests are prioritised, and when a recurring issue warrants a whole-class explanation. Agree a discreet handover signal and a simple support-request system. Both teachers distinguish review of the product from retrospective reflection on the learning process.

Co design inclusion and assessment calibration. Before each ten-sprint cycle, agree learning goals, subject contributions, success criteria, resources, scaffolds, roles and a compact evidence set. Prepare accessible participation and response options with specialist advice where needed. Compare sample work against Section 7 descriptors, agree who observes which learners and moderate selected judgements at the cycle checkpoint. Neither teacher should be permanently restricted to behaviour management or technical repair.

Ongoing development and support. Use short pre-sprint briefings, post-sprint debriefs, focused refreshers and the designated project or school training environment. Identify who answers technical questions and how unresolved issues reach a mentor or support contact. Review the pair's confidence after initial delivery and at every ten-sprint checkpoint. Initial orientation may be organised locally as practical half-days or shorter sessions, followed by targeted rehearsal; a short introductory course alone should not be treated as proof of Python readiness.

10.2. Implementation roles

Co-teaching is the recommended staffing arrangement across all three age groups and is particularly important during the transition to Python and in classes requiring substantial individual support. Leadership should timetable both teachers for the full practical sprint where feasible, including access to joint preparation and review time. Responsibilities are complementary, with shared accountability for learning, participation and safety.

IT teacher as instructional lead. Lead technical explanations and demonstrations, model algorithmic reasoning and debugging, verify code and equipment readiness, and address problems requiring deeper programming knowledge. During implementation, confer with teams and teach targeted mini-lessons. Lead the technical dimension of review and assessment while remaining attentive to accessibility, collaboration and learner understanding.

Classroom teacher as co teacher. Use knowledge of the class to support individuals and groups, clarify instructions, scaffold tasks, maintain meaningful role rotation and help learners manage frustration or disagreement. Check first-line difficulties using the agreed guide and collect observations and individual explanations. Support classroom organisation during demonstrations so that the IT teacher can continue teaching. Contribute to planning, disciplinary learning and attainment judgements rather than acting solely as a teaching assistant.

Shared responsibilities during the sprint. In Connect and Plan, both teachers establish the challenge, roles and access arrangements. During Build, Code, Test and Improve, the IT teacher leads technical teaching while the classroom teacher provides responsive support; either may work with a small group when agreed. In Share and Reflect, both gather evidence and guide feedback. After the sprint, compare observations, record next steps and identify learners needing further opportunities or an extension.

Senior leadership and curriculum coordination. The senior leadership team allocates staffing, equipment and preparation time, oversees safeguarding and reviews implementation barriers. A named curriculum lead coordinates age-level progression, cycle plans, professional learning and the support-request process. Record a deputy and the arrangements for staff absence. One colleague may combine coordination with a teaching role.

Technical and asynchronous platform support. A designated IT support contact checks device setup, approved software, connections, storage and platform access, and provides an escalation route for persistent faults. Distinguish this technical-service function from the IT teacher's instructional role. Confirm available asynchronous resources and response arrangements locally; an online platform cannot replace active classroom support.

Inclusion assessment and resource support. Inclusion or specialist staff advise on communication, assistive access and differentiation. A nominated assessment coordinator supports moderation and reporting. A librarian, resource coordinator or designated classroom colleague manages kit booking, inventory, charging and accessible materials where available. Safeguarding and data questions follow existing school contacts and procedures; separate posts are not required for every function.

Supported alternatives when co teaching is unavailable. Schools should explicitly record staffing constraints and arrange additional trained support, smaller practical groups, staggered sessions, stations or supported delivery before adopting a single-teacher model. Any alternative must preserve supervision, access and opportunities to learn. Do not assign an unprepared teacher a Python class because a second adult is unavailable. Peer support and asynchronous training supplement staffing; they do not replace adult responsibility or technical readiness.

10.3. Common challenges and mitigations

Limited Python experience. Provide focused training, pretested code, annotated examples, common-error guidance and rehearsal before each unfamiliar activity. Establish access to a knowledgeable colleague or mentor. Use brief diagnostic practice to identify teacher learning needs and schedule further support where explanations or troubleshooting remain insecure.

Whole class teaching repeatedly interrupted. Use the agreed co-teaching roles and a visible, accessible request-for-help routine. The classroom teacher handles first-line individual support while the IT teacher maintains the lesson. If several teams encounter the same conceptual issue, pause for a short shared explanation. Urgent safety concerns always take priority over continuing instruction.

Co teaching timetable and workload pressures. Agree joint lessons and short planning/debrief windows before the cycle starts. Reuse a shared lesson record rather than producing parallel plans. Leadership should resolve recurring staffing conflicts and identify contingency arrangements. Avoid repeatedly removing the classroom teacher during practical work, since that undermines the purpose of the model.

Unclear roles or fragmented teaching. Agree who leads each phase, how a support request is handed over and who records evidence. The IT teacher retains responsibility for accessible instruction; the classroom teacher contributes to learning and assessment. Review whether either role has narrowed to troubleshooting or behaviour management, and rebalance responsibilities through the next planning discussion.

Limited equipment and technical faults. Audit and book the core SPIKE Prime Sets and devices, run a pre-lesson test and maintain a short fault log. Use rotations and unplugged preparation to reduce waiting, while ensuring later direct evidence of practical competence. Prioritise working core equipment over optional Expansion Sets. Prepare a fallback task and an agreed point for escalating a persistent fault.

Different readiness and unequal participation. Check prerequisites, provide bridging tasks and accessible scaffolds, and offer deeper constraints for advanced learners. Both teachers monitor who builds, codes, tests and explains. Rotate meaningful opportunities without assigning fixed roles by confidence, disability or previous experience. The classroom teacher's knowledge of learners informs support without predetermining attainment.

Assessment burden. Divide observation priorities between the teachers, use common outcome codes and gather a small number of meaningful explanations, tests and reflections. Reconcile evidence during debriefing and moderate at ten-sprint checkpoints. One team product can support several individual records when each contribution is identifiable; avoid duplicate worksheets and competing judgements.

Staff absence and continuity. Maintain an accessible implementation folder with plans, programs, common faults, support contacts, safety routines and current learner next steps. A substitute needs orientation and technical readiness appropriate to the activity. If qualified cover is unavailable, adapt or reschedule practical teaching rather than assume that a prepared file makes unsupervised technical delivery suitable.

Community of practice and improvement loop. After initial delivery and at each ten-sprint checkpoint, review teacher reflections, learner feedback and selected evidence. Agree a small number of improvements with a named owner and review date, then check their effect in the next sprint or cycle. Share tested adaptations and useful explanations through approved meetings or training spaces, protecting learner identities. Escalate persistent staffing, access or training barriers to leadership and feed implementation findings into Section 11's evaluation process.

11. Evaluation and Review Cycle

11.1. What will be evaluated?

Evaluation examines whether the curriculum is effective, inclusive and feasible, and identifies improvements to teaching, resources and implementation. It draws on learner assessment without becoming another grading system. Interpret findings within each age level and school context, using the established allocation of Sprints 1–20 for ages 6–8, 21–40 for ages 8–10 and 41–60 for ages 10–12.

Learner outcomes and progression. Examine progress against the Section 5 outcomes across all four strands: programming, robotics systems, collaboration and reflection, and responsible technology use. Compare early and later evidence of explanation, construction, debugging and evaluation, noting the support used and opportunities provided. Check whether prerequisite understanding is secure and whether cumulative progression is manageable. Separate outcomes not yet assessed from demonstrated difficulty.

Inclusion participation and student voice. Investigate whether all learners can access instructions, tools, team decisions and assessment. Consider role rotation, communication options, physical and sensory access, and the usefulness of scaffolds and extensions. Ask learners what helped, what created difficulty and what should change. Look beyond visible enthusiasm or task completion to understand participation; do not infer ability or motivation from gaze, movement, silence or a diagnosis.

Artefact quality and reasoning. Review selected programs, mechanisms, algorithm plans, data records and revisions for correctness, reliability and fitness for purpose at the appropriate age level. Examine whether learners explain decisions and use test evidence to improve a solution. Evaluate progress and reasoning alongside the finished product; expensive components, elaborate appearance and adult assistance should not be mistaken for stronger attainment.

Teacher experience and co teaching. Gather separate and joint feedback from the IT teacher and classroom teacher about technical readiness, Python preparation, division of responsibilities, learner support and assessment. Examine whether co-teaching reduces interruptions to whole-class instruction and provides timely individual assistance. Record whether both teachers were available, any alternative staffing arrangements, and the effect on workload, inclusion and continuity. IT specialism alone is not evidence of Python readiness.

Implementation quality and practicality. Review delivery time, equipment reliability, device access, staffing, training and resource suitability. Check whether planning, iterative implementation, review and retrospective occur meaningfully, and whether adaptations preserve core outcomes. Examine links across subjects, assessment consistency and the burden of documentation. Identify which barriers concern curriculum design and which arise from local delivery conditions.

Success criteria and interpretation. Before a pilot or cycle, agree a small set of questions and indicators with teachers and leadership. Indicators may include change in demonstrated outcomes, access to meaningful roles, lesson interruptions, support delays, teacher

preparation needs and feasibility of sprint timings. Set any numeric targets locally before examining results and report counts and context. Do not present satisfaction, participation or simple before-and-after change as proof that the curriculum alone caused an effect.

11.2. How often?

Use a recurring Plan → Implement → Gather evidence → Review → Revise → Recheck cycle. Keep routine classroom review proportionate and connect it to the ten-sprint checkpoints already established in Section 7. Larger pilot findings inform curriculum and professional-development revisions; individual safety or access concerns require immediate action rather than waiting for a scheduled review.

Before initial delivery. Record the setting, age group, prior experience, relevant outcomes, staffing and training readiness, available resources and planned access arrangements. Collect a brief baseline through an appropriate explanation or performance task and learner/teacher feedback. Agree evaluation questions, evidence sources, responsibilities and review dates. Establish how optional recordings or engagement tools will be handled through school procedures.

Initial pilot and early check. Rehearse and trial a manageable selection of the opening activities for each participating age group, preserving the relevant sequence and prerequisites. Review after the first taught sprint and again after the first few sessions to identify unclear instructions, unrealistic timings, technical faults and support needs. A small pilot informs feasibility; it does not validate the entire 60-sprint programme.

After each sprint. The co-teaching pair holds a short debrief using the learner review, retrospective and selected observations. Record only significant successes, barriers and agreed next actions. Resolve urgent hazards or exclusion immediately. Make small teaching adjustments before the next session and note changes affecting the task, evidence or degree of support.

After each ten sprint cycle. At Sprints 10, 20, 30, 40, 50 and 60, combine outcome summaries, portfolio samples, learner voice and teacher feedback. Moderate a small sample of judgements and review participation, workload and staffing. Agree which materials, scaffolds, routines or training should change, name an owner and set a date to check the effect. These checkpoints are educational milestones, not necessarily calendar terms.

At the end of each level. After Sprints 20, 40 and 60, review end-of-level attainment and evidence coverage across all four strands. Compare the level's baseline and later evidence using comparable criteria, while noting changes in support, task demand and access. Identify consolidation, transition support and extensions, and produce a concise implementation summary for leadership and the teaching team.

Wider pilot and periodic review. Where organised through the project or school network, combine findings from classroom pilots, summer or after-school delivery, teacher training and external reviews or hackathons. Keep contexts and participant groups distinguishable. At the end of the local implementation period, and before a new cohort begins, review the accumulated findings and update the curriculum resources and training plan. Confirm

project-wide reporting dates with the responsible coordinator rather than infer them from classroom milestones.

Revision and follow up. Maintain a change log stating the issue, evidence, proposed revision, owner, version/date and follow-up finding. Teachers can adjust scaffolding and organisation locally; changes to outcomes, age allocations or the shared sprint sequence require curriculum-level agreement. After a revision, examine the same question again using suitable evidence and record whether the problem improved, persisted or requires another response. Retain prior versions and explain substantive changes to staff.

11.3. Evidence sources

Triangulate evidence: combine at least complementary learner, teacher and product perspectives where feasible rather than relying on one score or survey. Use information already collected for teaching and assessment whenever it answers the evaluation question. Keep the process small enough to be sustained and avoid asking learners and staff to complete duplicate records.

Surveys and short feedback checks. Use brief age-appropriate learner and teacher questions about clarity, challenge, access, support and usefulness. Provide pictorial, oral or other accessible response options. Ask equivalent core questions at relevant checkpoints to support comparison, but allow open comments. Report the number invited and responding, and do not treat a small or incomplete response set as representative of every participant.

Focus groups and learner conversations. Invite a varied range of learners and teachers, including quieter participants and those who experienced barriers. Use neutral questions about specific experiences and possible improvements. Record themes and differing views, avoid pressure to praise the programme, and provide a private route for sensitive feedback. Participation in optional evaluation activities must follow the school's arrangements without affecting attainment judgements.

Performance tasks and portfolio evidence. Use Section 5 outcome codes and Section 7 criteria to review selected programs, demonstrations, test results and reflections. Where progress is compared, use similar demands or a common criterion and record how task difficulty or assistance differed. Identify the learner's contribution within team work. A single successful demonstration or a team's high score is insufficient evidence of all programme outcomes.

Moderation notes and observations. Keep short records of how teachers interpreted rubric descriptors, resolved differing judgements and identified missing evidence. Structured observation can sample role participation, technical reasoning and use of feedback. For co-teaching evaluation, sample interruptions, types of support and unresolved requests through brief notes, not continuous surveillance. Use these records to improve practice rather than rank teachers.

Teacher team and implementation records. Draw on co-teacher debriefs, sprint plans, attendance or staffing records, training reflections, kit-fault logs and curriculum adaptation notes. Relate difficulties to the conditions under which they occurred. Team retrospectives

reveal collaboration processes, while individual reflections help avoid assuming that every learner shared the team's experience.

Optional engagement information. Where separately approved and implemented, engagement tools can provide supplementary contextual information. Interpret indicators alongside learner explanations, observations and task evidence; they are neither a diagnosis nor a direct attainment measure. The curriculum can be evaluated without camera-based monitoring. Apply the privacy and safeguarding arrangements in Section 8 to collection, access, alternatives and reporting.

Analysis reporting and action. Summarise patterns by age level, cycle and delivery context. Report participant numbers, evidence coverage, missing information and relevant limitations. Protect identities, particularly where small groups make learners recognisable, and avoid unsupported subgroup comparisons. Share an accessible summary of strengths, barriers, revisions and next review dates with staff and leadership, and suitable feedback with learners and families. Project or external dissemination uses appropriately authorised, aggregated or de-identified evidence.

Ownership and accountability. The curriculum lead coordinates the evaluation record; the IT and classroom teachers contribute classroom evidence; an assessment colleague supports moderation; and inclusion or technical staff advise on identified barriers. Leadership confirms priorities, resources and actions. The next review checks whether agreed changes were implemented and what difference they made, closing the feedback loop rather than ending with a report.

12. Appendices (optional but useful)

A1: Full outcomes list (by level)

This appendix consolidates all 36 coded learning outcomes from Section 5, grouped by level and strand. The wording and codes are reproduced unchanged. Codes 6, 8 and 10 denote the starting age of each band, not a national school grade. These detailed outcomes elaborate the broader competencies in Section 2; teacher support, indicative timing and minimum evidence remain specified in Section 5.

Level 1 ages 6 to 8

Sprints 1–20; Cycle 1: 1–10; Cycle 2: 11–20.

Strand A Computational Thinking and Programming

LO 6A1 I can arrange commands in a logical sequence, predict what my robot will do and compare my prediction with its actions.

LO 6A2 I can create and change a short block-based program using movement, light or sound, and use repetition when a pattern repeats.

LO 6A3 I can test my program, identify an action that does not match my plan and explain how a change improves it.

Strand B Robotics Systems and Physical Computing

LO 6B1 I can build a simple mechanism safely and show how a motor changes its movement.

LO 6B2 I can program a robot to move towards a measured target and explain how changing a movement setting affects the distance.

LO 6B3 I can use a sensor to trigger an action and explain how stopping after a set movement differs from stopping when the sensor is pressed.

Strand C Agile Collaboration and Reflection

LO 6C1 I can help my team agree a small goal and take a defined builder, programmer or tester role.

LO 6C2 I can listen, share equipment and report a difficulty so that my team can decide what to try next.

LO 6C3 I can contribute to the sprint review, show what we changed and name one way to improve our teamwork.

Strand D Responsible Inclusive and Real World Technology Use

LO 6D1 I can explain a safety choice and show how I use and put away the equipment responsibly.

LO 6D2 I can describe whom my robot could help and suggest a change that makes it easier for someone to use.

LO 6D3 I can help a teammate participate, respect different ideas and explain why we ask permission before sharing someone's picture or work.

Level 2 ages 8 to 10

Sprints 21–40; Cycle 1: 21–30; Cycle 2: 31–40.

Strand A Computational Thinking and Programming

LO 8A1 I can decompose a robotic challenge and represent my algorithm with a diagram, pseudocode or a clear sequence of blocks.

LO 8A2 I can create a program using loops and conditions, and use a variable or list when the task needs stored information.

LO 8A3 I can compare predicted and actual results, trace a faulty step and justify a code change using test evidence.

Strand B Robotics Systems and Physical Computing

LO 8B1 I can combine motors and sensors so that my robot responds appropriately to an object, a line or a measured value.

LO 8B2 I can record measurements, adjust a threshold or movement setting and explain how the adjustment changes behaviour.

LO 8B3 I can compare two designs or control strategies against agreed criteria and use the results to improve reliability.

Strand C Agile Collaboration and Reflection

LO 8C1 I can help plan a sprint by breaking the goal into tasks, agreeing roles and checking what we need to complete.

LO 8C2 I can report progress, negotiate a change and carry out my responsibilities while helping others contribute.

LO 8C3 I can use peer feedback and a retrospective to evaluate our solution and propose a specific improvement to our teamwork.

Strand D Responsible Inclusive and Real World Technology Use

LO 8D1 I can compare how useful and accessible two solutions are for different users and justify a design choice.

LO 8D2 I can apply safe equipment and digital practices, and identify information that should not be shared without permission.

LO 8D3 I can suggest a way to reuse materials or reduce waste and explain how my team makes participation fair.

Level 3 ages 10 to 12

Sprints 41–60; Cycle 1: 41–50; Cycle 2: 51–60.

Strand A Computational Thinking and Programming

LO 10A1 I can translate familiar block-based logic into Python and explain how variables, loops and conditions control the program.

LO 10A2 I can design a solution using a list to organise data and a function to reuse actions, explaining why each structure is useful.

LO 10A3 I can diagnose syntax, runtime and logic errors and evaluate a corrected program for clarity, correctness and efficiency.

Strand B Robotics Systems and Physical Computing

LO 10B1 I can analyse a mission's requirements and integrate a mechanism, sensors and actuators into a suitable robotic solution.

LO 10B2 I can select and adjust control parameters, collect repeated measurements and explain how they support reliable performance.

LO 10B3 I can evaluate and optimise my robot against success criteria such as accuracy or completion time and justify a design trade-off.

Strand C Agile Collaboration and Reflection

LO 10C1 I can coordinate a sprint plan with tasks, rotating responsibilities and success criteria, and adapt it when evidence changes.

LO 10C2 I can communicate progress, explain a technical decision and help resolve disagreement through reasoned discussion.

LO 10C3 I can evaluate my contribution and our team's process using review evidence and agree a realistic improvement for the next sprint.

Strand D Responsible Inclusive and Real World Technology Use

LO 10D1 I can evaluate how my design meets different users' needs and implement an improvement to accessibility or usability.

LO 10D2 I can identify safety, privacy or fairness concerns in collecting and presenting project data and explain an appropriate response.

LO 10D3 I can justify a sustainable design choice, evaluate a limitation of my solution and propose how to adapt it to a new context.

A2: Progression rubric (beginning → developing → secure → advanced)

This appendix provides a common progression rubric for interpreting the coded outcomes in Section 5 and Appendix A1. It reproduces the five rubric categories and four descriptors from Section 7, then illustrates how they are applied at each age level. Use it to describe demonstrated learning and identify next steps, rather than rank learners or convert age into an attainment score.

Interpreting the four stages

Beginning. Evidence shows an initial or partial grasp of the targeted outcome; the learner needs substantial guidance with the reasoning or process. Use this descriptor only after a fair opportunity to demonstrate the outcome.

Developing. The learner demonstrates parts of the outcome and can complete some steps with prompts, but explanation, consistency or application remains incomplete.

Secure. The learner demonstrates the outcome consistently at the relevant age level, with the support permitted in Section 5, and can explain or show why the solution works.

Advanced. The learner demonstrates deeper reasoning, comparison, justified improvement or transfer within the same age level. Advanced performance does not require content from an older level or a more expensive construction.

Common rubric across all levels

The following descriptors are reproduced unchanged from Section 7. Apply the targeted outcome and age-level expectations before choosing a descriptor; not every category needs to be judged in every sprint.

Category	Beginning	Developing	Secure	Advanced
Programming and algorithmic reasoning	Recognises some commands but cannot yet explain their sequence or effect.	Creates a partly effective algorithm and explains some choices with prompts.	Creates and explains a suitable algorithm using the level's expected structures.	Compares algorithms and justifies a clearer or more effective approach.
Robotics testing and improvement	Connects some components; predictions and tests need substantial guidance.	Builds a partly effective system and records tests with support.	Demonstrates intended behaviour and uses test evidence to justify a change.	Designs systematic comparisons and evaluates reliability, calibration or trade-offs.
Communication and explanation	Shares an observation but links to the task are not yet clear.	Describes steps or results with prompts and emerging technical vocabulary.	Explains decisions and evidence clearly through an accessible communication format.	Adapts explanation to the audience and responds thoughtfully to questions.

Category	Beginning	Developing	Secure	Advanced
Ethical reasoning and responsible use	Identifies a rule or user need with substantial prompting.	Applies some safe or inclusive choices and gives a simple reason.	Justifies a relevant safety, accessibility, fairness or sustainability decision.	Weighs competing needs or consequences and improves a design responsibly.
Teamwork and reflection	Participates in a role with frequent support; next steps need guidance.	Carries out tasks and identifies a useful contribution or improvement with prompts.	Fulfils responsibilities, listens and uses reflection to improve shared work.	Supports equitable participation and helps the team adapt using reflective evidence.

Connections to the four strands: programming and algorithmic reasoning supports Strand A; robotics testing and improvement supports Strand B; teamwork and reflection supports Strand C; ethical reasoning and responsible use supports Strand D. Communication is assessed across strands through accessible explanations, demonstrations and contributions.

Age level calibration examples

These examples illustrate the degree of reasoning and support expected in each band. They are not additional outcomes or a second marking scheme. Judge the full evidence against the appropriate LO 6, LO 8 or LO 10 statement and the common rubric above.

Level and sprints	Beginning	Developing	Secure	Advanced
Ages 6–8 Sprints 1–20 LO 6 outcomes	With substantial guidance, identifies a command or component but cannot yet connect it clearly to the intended action.	With prompts, produces part of a sequence and describes a result; needs help to identify a useful change.	Using age-appropriate guidance, demonstrates a short program, explains a tested change and fulfils an agreed team role.	Compares two simple approaches, predicts a consequence and explains an improvement while supporting fair participation.
Ages 8–10 Sprints 21–40 LO 8 outcomes	Recognises a relevant structure or sensor but needs substantial support to organise steps and interpret a test.	Uses some loops or conditions and records observations with prompts; links between evidence and revisions are emerging.	Designs a suitable algorithm, interprets recorded tests and justifies a revision; shares responsibilities and uses peer feedback.	Compares strategies systematically, evaluates measurement or control choices and transfers a useful improvement to a related task.

Level and sprints	Beginning	Developing	Secure	Advanced
Ages 10–12 Sprints 41–60 LO 10 outcomes	Recognises aspects of Python or system behaviour but cannot yet trace the logic or diagnose a fault without substantial guidance.	Creates a partly effective solution and fixes some errors with prompts; justification or test consistency remains incomplete.	With increasing autonomy, explains suitable Python logic, diagnoses errors and evaluates performance while coordinating responsible shared work.	Justifies algorithmic or mechanical trade-offs, tests optimisation rigorously and proposes a reasoned adaptation for a new user or context.

Support and achievement are recorded separately. A learner can be Secure while using a communication aid, extra processing time or adapted input. Where an adult or peer supplies the reasoning being assessed, seek a further individual demonstration before concluding that the outcome is secure.

Applying and recording the rubric

Select. Identify the age level, sprint, outcome code and relevant rubric categories. Share a small number of accessible success criteria before the task.

Gather. Use individual explanations, demonstrations and observations alongside team artifacts and test records. An incomplete robot may still provide strong reasoning evidence; a successful team product does not establish every member's attainment.

Judge. Choose a descriptor for each assessed outcome or category from the available evidence. Profiles may differ across strands. Use "Not yet assessed" where evidence or opportunity is missing; do not treat this as Beginning. Do not grade personality, speed, enthusiasm or attention indicators.

Record. Keep a concise entry: date and sprint; LO code; evidence reference; descriptor; support used; feedback; and next review opportunity. Do not require a separate worksheet for every descriptor or average the four stages into a compulsory numeric score.

Improve and review. Give one actionable next step and an opportunity to apply it. At each ten-sprint checkpoint, compare selected evidence with a colleague and resolve different interpretations. At Sprints 20, 40 and 60, review the level profile against Appendix A1 and Section 5's minimum evidence requirements.

Example record

LO 8B2 — Secure. Evidence: the learner interpreted recorded distance readings, adjusted the stopping threshold and explained the changed behaviour. Support: a prepared test table and an oral prompt to repeat the trial. Next step: compare repeated results and explain whether the setting is reliable. This supports the recorded outcome; it does not automatically establish Secure attainment in every strand.

A3: Unit plans (full)

Unit organisation and gradual release of guidance

For each age group, ALPER-EU comprises two unit structures of ten Agile sprints each. The first unit is delivered with full guidance by the instructor or co-teaching team: instructors model the process, explain key concepts, demonstrate safe construction and programming, and lead planning, testing and reflection. The second unit is delivered with milder guidance, allowing and encouraging learners to demonstrate self-initiative in choosing approaches, allocating tasks, diagnosing problems and justifying improvements. This deliberate release of responsibility consolidates learning and makes learners' independent contributions visible.

Guidance is responsive rather than withdrawn on a fixed date. Full guidance includes active learner decisions and practice; milder guidance retains explicit teaching when new concepts arise, safety supervision, accessibility adjustments and timely intervention. Self-initiative is judged relative to age, prior experience and agreed support, not speed or absence of help. Review readiness at the end of the first ten sprints and retain or restore scaffolds where needed.

The six unit structures are: ages 6–8, Unit 1 Sprints 1–10 and Unit 2 Sprints 11–20; ages 8–10, Unit 1 Sprints 21–30 and Unit 2 Sprints 31–40; ages 10–12, Unit 1 Sprints 41–50 and Unit 2 Sprints 51–60. These are curriculum units, not national school grades.

Status and use of these plans

The plans below operationalise the agreed curriculum structure using the lesson focuses and proposed subject connections already recorded in Section 9. They are teacher planning guidance, not verbatim reproductions of the original lesson packs. Check each supplied sprint scenario, construction instructions and code before delivery. Suggested timing, success criteria and extensions may be adapted without changing the sprint order or age allocation. Outcome references identify opportunities to gather evidence; no single sprint establishes mastery of every linked outcome.

Common delivery arrangements

Resources and preparation. Use the LEGO SPIKE Prime Set as the minimum viable equipment in every age group; the Expansion Set is optional. Prepare a compatible device, tested programming environment, relevant sprint files, charging and connection arrangements, build instructions, a clear test area and accessible recording materials. Pretest all essential mechanisms and code. Use block-based programming for younger learners and age-appropriate Python preparation for older learners, following the supplied sprint resources. Confirm Python proficiency through rehearsal rather than assuming that IT specialism establishes readiness.

Co-teaching. The IT teacher leads technical demonstrations and programming; the classroom teacher supports individuals and teams, language, participation and observation. Both plan outcomes and review evidence. During team work they divide support responsibilities so individual troubleshooting does not repeatedly halt whole-class instruction. Where co-

teaching cannot be scheduled, apply the supported alternatives in Section 10 and adjust grouping, complexity and timing.

Standard sprint routine and indicative timing. Connect 5 minutes; Plan 10; Build 20; Code 15; Test 10; Improve 15; Share 10; Reflect 5: 90 minutes overall. Split across shorter lessons or extend construction and testing where needed. Build, Code, Test and Improve form a repeated implementation loop, not a single pass. Share is the review of the product against the goal; Reflect is the retrospective on learning and teamwork. Carry one action into the next sprint. Use brief progress check-ins during implementation.

Access and safety. Offer visual instructions, modelled vocabulary, oral or pictorial records, accessible input and communication options, low-tech algorithm rehearsal and movement breaks. Rotate meaningful roles without removing necessary adaptations. Establish safe movement limits, clear testing boundaries and an accessible stop procedure. Seek permission before sharing identifiable images or work; use anonymous or fictional data where possible. Assess reasoning separately from literacy, dexterity and physical performance.

Assessment. Select a small number of outcomes before each sprint and share accessible success criteria. Retain a plan, code or algorithm, model evidence, a test/change record and a brief individual explanation. Apply Appendix A2 and Section 7 to selected evidence, recording support separately. Observe individual contributions within team work. Use “Not yet assessed” when evidence is missing. Moderate a small sample at each ten-sprint checkpoint.

Age group 6–8 Unit 1 Sprints 1–10

Purpose and progression. Establish concrete sequences, robot components, cause and effect and cooperative routines.

Entry and outcomes. Check current experience through a short build, algorithm demonstration and discussion; do not assume prior coding. Draw on LO 6A1–A3, 6B1–B3, 6C1–C3 and 6D1–D3 in Section 5 and A1, selecting relevant statements for each sprint.

Guidance and learner agency. Instructors provide worked examples, a visible goal, stepwise demonstrations and guided test records. Learners predict before execution, carry out meaningful roles and explain a small change. Gradually invite choices within clear boundaries.

Unit evidence and review. Keep selected sprint artefacts, test records and individual explanations across all four strands. At Sprint 10, review a working or partially working solution, the reasoning behind revisions, fair participation and one retrospective action. Agree readiness and remaining scaffolds for the second unit. Do not equate an attractive construction with secure programming understanding.

Sprint 1 Light Matrix

Challenge and subject connection. Represent a symbol or number with a light pattern and explain the sequence. Links: Mathematics; art; language.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 2 The Motor

Challenge and subject connection. Compare motor settings, direction and rotations using observations or simple measurements. Links: Science; mathematics; technology/design.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 3 Colour Sensor

Challenge and subject connection. Classify coloured objects and explain the response triggered by each colour. Links: Science; art; mathematics.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 4 Distance Sensor

Challenge and subject connection. Compare measured distances with sensor readings and test an obstacle threshold. Links: Science; mathematics.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 5 Force Sensor

Challenge and subject connection. Compare touch and pressure responses and explain a safe start/stop mechanism. Links: Science; technology/design.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 6 Gyro Sensor

Challenge and subject connection. Explore orientation and tilt; connect a physical movement with a programmed response. Links: Science; mathematics; physical education.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative

solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 7 Pass the Brick

Challenge and subject connection. Coordinate a transfer mechanism and evaluate how teams communicate and synchronise actions. Links: Technology/design; personal/social education.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 8 Ideas the LEGO Way

Challenge and subject connection. Describe a construction clearly, reconstruct a peer's idea and discuss differences. Links: Art; language; technology/design.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 9 What Is This

Challenge and subject connection. Repurpose a model for a different need and explain the design choice. Links: Technology/design; art; language.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 10 Going the Distance

Challenge and subject connection. Compare wheel-rotation distance estimates with touch-sensor stopping through practical tests. Links: Mathematics; science.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Age group 6–8 Unit 2 Sprints 11–20

Purpose and progression. Apply familiar commands and mechanisms to purposeful challenges, explain tested changes and initiate team choices.

Entry and outcomes. Review the preceding unit's portfolio and individual explanations; reteach insecure essentials before increasing autonomy. Draw on LO 6A1–A3, 6B1–B3, 6C1–C3 and 6D1–D3 in Section 5 and A1, selecting relevant statements for each sprint.

Guidance and learner agency. Teams propose the plan and roles, choose a feasible strategy and gather evidence before requesting a solution. Instructors use questions, short targeted demonstrations and agreed help checkpoints. Reintroduce direct modelling for unfamiliar or insecure concepts; retain all required access supports.

Unit evidence and review. Keep selected sprint artefacts, test records and individual explanations across all four strands. At Sprint 20, review a working or partially working solution, the reasoning behind revisions, fair participation and one retrospective action. Review the level's outcomes and minimum evidence in Section 5; identify consolidation and next learning needs. Do not equate an attractive construction with secure programming understanding.

Sprint 11 Goal

Challenge and subject connection. Test how movement settings affect a shot and record successful attempts. Links: Physical education; science; mathematics.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 12 Help

Challenge and subject connection. Identify an everyday need and justify whom a proposed robotic solution helps. Links: Citizenship; personal/social education; technology/design.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 13 Hopper Race

Challenge and subject connection. Compare wheel-less mechanisms and measure performance over an agreed route. Links: Science; mathematics; physical education.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 14 Super Cleanup

Challenge and subject connection. Compare collection mechanisms using an agreed count, time or effectiveness criterion. Links: Environmental education; technology/design; mathematics.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 15 Design for Someone

Challenge and subject connection. Design an assistive mechanism and explain an accessibility decision respectfully. Links: Citizenship; technology/design; personal/social education.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 16 Broken

Challenge and subject connection. Investigate a mechanical fault and describe the evidence supporting a repair. Links: Science; technology/design; language.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO

6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 17 Break Dance

Challenge and subject connection. Create repeating movement patterns and evaluate timing against a chosen rhythm. Links: Music; physical education; art; mathematics.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 18 Repeat 5 Times

Challenge and subject connection. Represent exercise repetitions with a loop and compare commanded and observed counts. Links: Physical education; mathematics.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 19 Rain or Shine

Challenge and subject connection. Model a weather-related response and distinguish the model's input from actual weather measurement. Links: Science; environmental education; art.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 20 Place Your Order

Challenge and subject connection. Plan a delivery route and compare distance, movement settings and completion time. Links: Mathematics; technology/design; social studies.

Learning and success criteria. Target the relevant LO 6A and 6B statements for the algorithm and robot behaviour, plus LO 6C1–C3 for planning and reflection. Select an appropriate LO 6D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record a result with a picture, tally or short explanation; change one relevant feature and retest. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Age group 8–10 Unit 1 Sprints 21–30

Purpose and progression. Develop structured algorithms, sensor control and evidence-based troubleshooting with explicit modelling.

Entry and outcomes. Check current experience through a short build, algorithm demonstration and discussion; do not assume prior coding. Draw on LO 8A1–A3, 8B1–B3, 8C1–C3 and 8D1–D3 in Section 5 and A1, selecting relevant statements for each sprint.

Guidance and learner agency. Instructors provide worked examples, a visible goal, stepwise demonstrations and guided test records. Learners predict before execution, carry out meaningful roles and explain a small change. Gradually invite choices within clear boundaries.

Unit evidence and review. Keep selected sprint artefacts, test records and individual explanations across all four strands. At Sprint 30, review a working or partially working solution, the reasoning behind revisions, fair participation and one retrospective action. Agree readiness and remaining scaffolds for the second unit. Do not equate an attractive construction with secure programming understanding.

Sprint 21 Out of Order

Challenge and subject connection. Debug a sorting process, classify its errors and test revised rules. Links: Technology/design; mathematics.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated

challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 22 Driving Around

Challenge and subject connection. Relate rotations or movement time to distance and compare repeated route tests. Links: Mathematics; science.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 23 Playing with Objects

Challenge and subject connection. Use force and distance inputs to explain autonomous interaction with objects. Links: Science; technology/design.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 24 Reacting to Lines

Challenge and subject connection. Compare surface brightness readings and test conditions for following a line. Links: Science; mathematics.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 25 Creative Problem Solving

Challenge and subject connection. Decompose a mission, compare strategies and justify a team solution. Links: Technology/design; mathematics; personal/social education.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 26 Back to Back

Challenge and subject connection. Give precise construction instructions and revise them after a partner's attempt. Links: Language; personal/social education; technology/design.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 27 Keep It Safe

Challenge and subject connection. Model access control and discuss why a mechanism allows or refuses entry. Links: Citizenship; technology/design.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 28 Keep It Really Safe

Challenge and subject connection. Represent combined conditions and test which input combinations open the model vault. Links: Mathematics; citizenship; technology/design.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated

challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 29 Automate It

Challenge and subject connection. Model package selection and transport, explaining rules and movement decisions. Links: Technology/design; mathematics; social studies.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 30 Track Your Packages

Challenge and subject connection. Use coordinates and spatial instructions to locate packages on a model route. Links: Mathematics; geography; technology/design.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Age group 8–10 Unit 2 Sprints 31–40

Purpose and progression. Apply and extend algorithmic and data reasoning through team-led investigations and decisions.

Entry and outcomes. Review the preceding unit's portfolio and individual explanations; reteach insecure essentials before increasing autonomy. Draw on LO 8A1–A3, 8B1–B3, 8C1–C3 and 8D1–D3 in Section 5 and A1, selecting relevant statements for each sprint.

Guidance and learner agency. Teams propose the plan and roles, choose a feasible strategy and gather evidence before requesting a solution. Instructors use questions, short targeted demonstrations and agreed help checkpoints. Reintroduce direct modelling for unfamiliar or insecure concepts; retain all required access supports.

Unit evidence and review. Keep selected sprint artefacts, test records and individual explanations across all four strands. At Sprint 40, review a working or partially working solution, the reasoning behind revisions, fair participation and one retrospective action. Review the level's outcomes and minimum evidence in Section 5; identify consolidation and next learning needs. Do not equate an attractive construction with secure programming understanding.

Sprint 31 Code Your Moves

Challenge and subject connection. Write movement pseudocode and test whether another learner or robot can follow it. Links: Language; physical education; music.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 32 Wind Speed

Challenge and subject connection. Interpret supplied wind-speed data and represent categories with a mechanical indicator. Links: Science; geography; mathematics.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 33 Veggie Love

Challenge and subject connection. Use rainfall or forecast examples to justify a model watering decision. Links: Science; environmental education; mathematics.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO

8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 34 Brain Game

Challenge and subject connection. Store colour sequences, compare game results and discuss fair testing of performance. Links: Mathematics; personal/social education.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 35 The Coach

Challenge and subject connection. Design a training sequence and explain how lists or loops organise instructions. Links: Physical education; mathematics; language.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative

solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 36 Stretch with Data

Challenge and subject connection. Relate tilt and movement to recorded graphs; compare patterns across trials. Links: Physical education; science; mathematics.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 37 This Is Uphill

Challenge and subject connection. Compare measurements on different inclines and discuss force and controlled testing. Links: Science; mathematics; physical education.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 38 Time for Squat Jumps

Challenge and subject connection. Record height data; interpret graphs and introduce potential-energy reasoning when appropriate. Links: Physical education; science; mathematics.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic

decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 39 Watch Your Steps

Challenge and subject connection. Compare counted steps and estimated distance; evaluate consistency across trials. Links: Physical education; mathematics; science.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 40 Aim for It

Challenge and subject connection. Investigate motion data and relationships between speed, mass and kinetic energy at an appropriate level. Links: Science; mathematics; physical education.

Learning and success criteria. Target the relevant LO 8A and 8B statements for the algorithm and robot behaviour, plus LO 8C1–C3 for planning and reflection. Select an appropriate LO 8D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Age group 10–12 Unit 1 Sprints 41–50

Purpose and progression. Establish reliable integrated mechanisms and progressively develop Python reasoning through guided tracing and debugging.

Entry and outcomes. Check current experience through a short build, algorithm demonstration and discussion; do not assume prior coding. Draw on LO 10A1–A3, 10B1–B3, 10C1–C3 and 10D1–D3 in Section 5 and A1, selecting relevant statements for each sprint.

Guidance and learner agency. Instructors provide worked examples, a visible goal, stepwise demonstrations and guided test records. Learners predict before execution, carry out

meaningful roles and explain a small change. Gradually invite choices within clear boundaries.

Unit evidence and review. Keep selected sprint artefacts, test records and individual explanations across all four strands. At Sprint 50, review a working or partially working solution, the reasoning behind revisions, fair participation and one retrospective action. Agree readiness and remaining scaffolds for the second unit. Do not equate an attractive construction with secure programming understanding.

Sprint 41 The Obstacle Course

Challenge and subject connection. Build moving obstacles and explain energy transfer, friction and changes in movement. Links: Science; technology/design; physical education.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 42 Advanced Driving Base

Challenge and subject connection. Assemble coordinated subassemblies and evaluate the resulting platform’s reliability. Links: Technology/design; mathematics; personal/social education.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 43 My Code Our Program

Challenge and subject connection. Combine reusable subroutines and explain interfaces between team members' code. Links: Technology/design; language; personal/social education.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 44 Time for an Upgrade

Challenge and subject connection. Design attachments, test their mechanical operation and justify parameter changes. Links: Technology/design; science.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative

solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 45 Mission Ready

Challenge and subject connection. Plan mission strategy and use test evidence to improve reliable completion. Links: Mathematics; technology/design; personal/social education.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 46 Start Sensing

Challenge and subject connection. Trace force-sensor conditions and test when a Python-controlled motor starts or stops. Links: Science; technology/design.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 47 Charging Rhino

Challenge and subject connection. Compare movement settings and contact stopping using safe, low-speed trials. Links: Science; mathematics.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against

the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 48 Moving Motors

Challenge and subject connection. Compare sequential and simultaneous movement and create a coordinated motor pattern. Links: Music; physical education; mathematics.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 49 New Moves with Motors

Challenge and subject connection. Compare relative rotation and absolute position while diagnosing code errors. Links: Mathematics; science.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 50 Cart Control

Challenge and subject connection. Test sensor-based obstacle stopping and compare thresholds for a model delivery cart. Links: Science; technology/design; mathematics.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Model the task and one key example; elicit a prediction and jointly agree a small goal, roles and test. Revisit prior learning and agree what result would count as success.

Build and Code. Demonstrate essential connections or logic in short steps, then pause for learners to construct, rehearse and modify the example. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Age group 10–12 Unit 2 Sprints 51–60

Purpose and progression. Apply and extend Python loops, conditions, lists and functions through student-initiated designs and systematic evaluation.

Entry and outcomes. Review the preceding unit's portfolio and individual explanations; reteach insecure essentials before increasing autonomy. Draw on LO 10A1–A3, 10B1–B3, 10C1–C3 and 10D1–D3 in Section 5 and A1, selecting relevant statements for each sprint.

Guidance and learner agency. Teams propose the plan and roles, choose a feasible strategy and gather evidence before requesting a solution. Instructors use questions, short targeted demonstrations and agreed help checkpoints. Reintroduce direct modelling for unfamiliar or insecure concepts; retain all required access supports.

Unit evidence and review. Keep selected sprint artefacts, test records and individual explanations across all four strands. At Sprint 60, review a working or partially working solution, the reasoning behind revisions, fair participation and one retrospective action. Review the level's outcomes and minimum evidence in Section 5; identify consolidation and next learning needs. Do not equate an attractive construction with secure programming understanding.

Sprint 51 Hopper Run

Challenge and subject connection. Relate paired-motor control to hopping mechanics and compare movement settings. Links: Science; technology/design.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 52 Race Day

Challenge and subject connection. Design a multi-step route and evaluate accuracy and completion time. Links: Mathematics; physical education; technology/design.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 53 Warm Up Loop with Leo

Challenge and subject connection. Compare repeated exercise commands with a for loop and explain the repetition count. Links: Physical education; mathematics.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 54 Setting Conditions for Yoga

Challenge and subject connection. Trace a while loop, counter and stopping condition in a model coaching routine. Links: Physical education; mathematics.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 55 Controlling Motion with Tilt

Challenge and subject connection. Create an interactive story using gestures, orientation, conditions, sound and images. Links: Science; language; art.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 56 Guess Which Color

Challenge and subject connection. Classify colour inputs, create a decision flowchart and link each case to a sound. Links: Science; mathematics; music.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against

the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 57 Break Dancer Break Down

Challenge and subject connection. Diagnose dance-program errors and explain corrections using an error log. Links: Music; physical education; language.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 58 Score

Challenge and subject connection. Program scoring, store results and compare scores across agreed game rounds. Links: Mathematics; physical education; technology/design.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 59 Clean Indicator

Challenge and subject connection. Use colour-coded cleanliness scenarios and functions; explain that the model does not measure actual pollution. Links: Environmental education; science; mathematics.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above

to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

Sprint 60 Mind Games

Challenge and subject connection. Use lists for memory-game sequences and compare results under consistent conditions. Links: Mathematics; personal/social education.

Learning and success criteria. Target the relevant LO 10A and 10B statements for the algorithm and robot behaviour, plus LO 10C1–C3 for planning and reflection. Select an appropriate LO 10D statement for safe, fair or useful design. Success means demonstrating the stated challenge, explaining a relevant code or build decision and supporting a revision with observable evidence.

Connect and Plan. Invite teams to propose a goal, algorithm and role allocation; check feasibility and ask them to justify the first test. Revisit prior learning and agree what result would count as success.

Build and Code. Teams choose and implement an approach using the supplied resources; instructors probe decisions and provide focused help when needed. Use the challenge above to determine the required mechanism, input and output; sketch or trace the algorithm before running it.

Test and Improve. Predict the expected behaviour, run a safe trial and compare observation with prediction. Record repeated results where useful, change one parameter or logic decision and compare the revised result with the baseline. Explain the cause of a mismatch rather than merely retrying until it works.

Share and Reflect. Demonstrate the challenge and explain one evidence-based change. Each learner contributes an accessible explanation or demonstration. Review the product against the goal, then discuss role-sharing and a difficulty overcome; agree one process improvement for the next sprint.

Evidence, adaptation and extension. Keep the algorithm or code, a model image or sketch, test/change evidence and an individual reflection. Offer a prepared build, visual sequence, reduced initial goal or unplugged rehearsal as needed. Extend by comparing an alternative solution or adapting for another user, with justification. Use the common timing, safety and assessment arrangements above; adjust time to the actual build and learner needs.

ALPER EU Curriculum Agile Sprint Listing

This catalogue brings together the descriptive titles, learning content, and resource inventories for all 60 agile sprints available at https://drive.google.com/drive/folders/1OjGiZr3vesl-qfGJ5XPvAdxu-5CWIudR?usp=sharing .

Included in every sprint: 10 teaching documents in DOCX format: an English lesson scenario, teacher guide, student worksheet, student self-assessment, and teacher reflection/self-assessment, plus their five counterparts translated in Greek, Maltese, Lithuanian, Croatian and Portuguese. These are included in every total below.

Additional formats include SPIKE projects (.llsp3), digital construction models (.io), images, videos, and sample spreadsheets. Descriptive titles summarize each lesson's focus.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
1 LP_SP_1	**Light Matrix: First Robot Signals.** Display patterns, symbols, and numbers on the Hub through simple command sequences.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 video; No PDF resource included. 17 files total.
2 LP_SP_2	**Motors: Speed, Direction, and Rotation.** Explore how motor speed, duration, and rotations change movement.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 video; No PDF resource included. 17 files total.
3 LP_SP_3	**Colour Sensor: Detect and React.** Recognize colours and trigger robot responses using conditions.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 video; No PDF resource included. 17 files total.
4 LP_SP_4	**Distance Sensor: Detecting Obstacles.** Measure distance and program reactions when an obstacle reaches a chosen threshold.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 video; No PDF resource included. 17 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
5 LP_SP_5	**Force Sensor: Touch-Controlled Actions.** Use pressing, releasing, and pressure as inputs for robot control.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 video; No PDF resource included. 17 files total.
6 LP_SP_6	**Gyroscope: Responding to Tilt.** Use Hub orientation and movement to trigger images, sounds, or other actions.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 video; No PDF resource included. 17 files total.
7 LP_SP_7	**Pass the Brick: Robotic Teamwork.** Build and program an object-transfer mechanism while practising coordination and shared roles.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
8 LP_SP_8	**Ideas, the LEGO Way: Build and Communicate.** Develop creativity and clear descriptions so classmates can reproduce constructions.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
9 LP_SP_9	**What Is This? Repurposing a Robot.** Imagine alternative uses for an existing construction and redesign it to solve new problems.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
10 LP_SP_10	**Going the Distance: Measure and Stop.** Relate wheel rotations to travel distance and compare calculated stopping with touch-sensor stopping.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 2 videos. 20 files total.
11 LP_SP_11	**Goal! Precision Through Motor Control.** Adjust force, timing, and mechanical design to propel a ball toward a goal.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 2 instruction PDFs; 1 video; No .io model included. 18 files total.
12 LP_SP_12	**Help! Designing for Everyday Needs.** Identify real problems and develop empathetic robotic solutions.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
13 LP_SP_13	**Hopper Race: Movement Without Wheels.** Investigate jumping and pulling mechanisms and improve a hopping robot.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
14 LP_SP_14	**Super Cleanup: Comparing Collection Mechanisms.** Design and evaluate robotic approaches to collecting objects and addressing an environmental problem.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; No PDF resource included; No video included. 16 files total.
15 LP_SP_15	**Design for Someone: Assistive Robotics.** Create mechanisms that address the needs of people with mobility limitations.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; No PDF resource included; No video included. 16 files total.
16 LP_SP_16	**Broken: Diagnose and Repair.** Investigate mechanical faults, test explanations, and restore a robot's operation.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; Broken_PrintTempalte PDF; 1 video. 22 files total.
17 LP_SP_17	**Break Dance: Timing and Synchronization.** Program a dancing robot using coordinated, repeating movement patterns.	1 Python project; 1 Scratch project; 9 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video. 26 files total.
18 LP_SP_18	**Repeat 5 Times: Exercise with Loops.** Connect fixed-count repetition in programming with physical exercise and movement.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 20 files total.
19 LP_SP_19	**Rain or Shine: Weather-Reactive Robotics.** Use environmental information and conditions to change a model's behaviour.	1 Python project; 1 Scratch project; 5 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 21 files total.
20 LP_SP_20	**Place Your Order: Automated Delivery.** Program cargo transport while exploring distance, speed, time, and movement accuracy.	1 Python project; 1 Scratch project; 6 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video. 23 files total.
21 LP_SP_21	**Out of Order: Debugging a Sorting System.** Diagnose and correct an algorithm that sorts objects incorrectly.	1 Python project; 1 Scratch project; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 19 files total.
22 LP_SP_22	**Driving Around: Accurate Robot Movement.** Compare motor rotations, movement time, and measured travel distance through repeated tests.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 20 files total.
23 LP_SP_23	**Playing with Objects: Sensor-Guided Interaction.** Use force and distance sensors to detect objects and control autonomous actions.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 20 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
24 LP_SP_24	**Reacting to Lines: Following a Path.** Interpret colour and brightness readings to guide a robot along lines.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; Reacting with Lines_Line Ideas PDF; 1 video. 21 files total.
25 LP_SP_25	**Creative Problem Solving: Robotics Missions.** Analyse competition-style challenges, plan strategies, and improve mechanical and programming solutions.	1 SPIKE project; 1 code screenshot; 1 photo or model image; 1 .io model; 3 instruction PDFs; 1 video. 18 files total.
26 LP_SP_26	**Back to Back: Clear Construction Instructions.** Turn a physical construction into precise verbal and written instructions, introducing pseudocode.	1 photo or model image; 1 .io model; 1 instruction PDF; Back To Back_Lesson Presentation PDF; No video included. 14 files total.
27 LP_SP_27	**Keep It Safe: A Programmable Vault.** Build a security mechanism that uses sensors and conditions to control access.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video. 21 files total.
28 LP_SP_28	**Keep It Really Safe: Combined Conditions.** Strengthen a vault's access control by combining conditions with logical AND.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video. 21 files total.
29 LP_SP_29	**Automate It: Warehouse Package Handling.** Build and program a mechanism that selects and transports packages.	1 Python project; 1 Scratch project; 3 code screenshots; 1 photo or model image; 1 .io model; 3 instruction PDFs; 1 video. 21 files total.
30 LP_SP_30	**Track Your Packages: Coordinates and Routes.** Use spatial orientation and X–Y movement to locate and transport packages.	1 Python project; 1 Scratch project; 8 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; Track Your Packages_ Print Template PDF; 1 video. 26 files total.
31 LP_SP_31	**Code Your Moves: From Pseudocode to Dance.** Write, test, and refine movement instructions before translating them into robot commands.	1 Python project; 1 Scratch project; 9 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; Code Your Moves_Cards and Presentation PDF; 1 video. 26 files total.
32 LP_SP_32	**Wind Speed: Turning Weather Data into Motion.** Translate external wind-speed data and Beaufort categories into mechanical and visual indications.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 3 instruction PDFs; Wind Speed_Map PDF; 1 video. 23 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
33 LP_SP_33	**Veggie Love: Predicting Watering Needs.** Combine weather forecasts, rainfall information, and variables to decide whether vegetables need watering.	1 Python project; 1 Scratch project; 3 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; Veggie Love_Printable PDF; 1 video. 21 files total.
34 LP_SP_34	**Brain Game: Memory, Lists, and Results.** Build a colour-memory game using lists to store sequences and analyse performance.	1 Python project; 1 Scratch project; 8 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video. 25 files total.
35 LP_SP_35	**The Coach: Design a Digital Training Assistant.** Build a robotic coach that guides an athletic activity. Use pseudocode, lists, and loops to organize training instructions, measure time or repetitions, and improve the routine through testing.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
36 LP_SP_36	**Stretch with Data: Graphing Movement.** Build a Yoga Ring and explore how tilt and movement appear in real-time graphs.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video; 2 sample XLSX spreadsheets. 22 files total.
37 LP_SP_37	**This Is Uphill: Comparing Incline Measurements.** Investigate how slope affects movement and required force using a training-tracker model.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video; 1 sample XLSX spreadsheet. 20 files total.
38 LP_SP_38	**Time for Squat Jumps: Measuring Height and Energy.** Record jump-height measurements and explore gravitational potential energy through data analysis.	1 Python project; 1 Scratch project; 4 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video; 1 sample XLSX spreadsheet. 21 files total.
39 LP_SP_39	**Watch Your Steps: Build a Pedometer.** Count steps, calculate distance and walking speed, and represent movement data graphically.	1 Python project; 1 Scratch project; 10 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video; 1 sample XLSX spreadsheet. 28 files total.
40 LP_SP_40	**Aim for It: Motion and Kinetic Energy.** Use a wheeled model to collect movement data and investigate speed, mass, and kinetic energy.	1 Python project; 1 Scratch project; 6 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video; 1 sample XLSX spreadsheet. 23 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
41 LP_SP_41	**The Obstacle Course: Energy in Motion.** Design moving obstacles and investigate potential energy, kinetic energy, friction, and repeated motion.	1 Python project; 1 Scratch project; 5 code screenshots; 1 photo or model image; 1 .io model; 2 instruction PDFs; 1 video. 22 files total.
42 LP_SP_42	**Advanced Driving Base: Modular Team Building.** Assemble separate robot subassemblies, integrate them, and test a versatile driving platform.	1 Python project; 1 Scratch project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
43 LP_SP_43	**My Code, Our Program: Reusable Subroutines.** Create individual My Blocks and combine them into a structured team program.	1 Python project; 1 Scratch project; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 19 files total.
44 LP_SP_44	**Time for an Upgrade: Mission Attachments.** Design two attachments and program their motors to extend the driving base's capabilities.	1 Python project; 1 Scratch project; 3 code screenshots; 2 photo or model images; 1 .io model; 1 instruction PDF; No video included. 19 files total.
45 LP_SP_45	**Mission Ready: Integrated Competition Strategy.** Combine attachments, controlled movement, pseudocode, and reusable code to complete missions efficiently.	1 Python project; 1 Scratch project; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 19 files total.
46 LP_SP_46	**Start Sensing: Force-Sensor Control in Python.** Use sensor conditions to start or stop motors and wait for input events.	3 Python projects; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 20 files total.
47 LP_SP_47	**Charging Rhino: Stop on Contact.** Control paired motors in Python and stop a moving robot when its force sensor touches an obstacle.	1 Python project; 1 code screenshot; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 16 files total.
48 LP_SP_48	**Moving Motors: Sequential and Simultaneous Actions.** Explore motor rotation parameters and coordinate multiple motors in Python.	2 Python projects; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
49 LP_SP_49	**New Moves with Motors: Precision and Debugging.** Compare relative rotation with absolute positioning and diagnose Python errors.	1 Python project; 6 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 21 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
50 LP_SP_50	**Cart Control: Stop Before an Obstacle.** Use distance-sensor conditions and Python waiting logic for automated package delivery.	1 Python project; 5 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 20 files total.
51 LP_SP_51	**Hopper Run: Synchronizing Wheel-Less Movement.** Control paired motors and investigate how mechanism design affects hopping motion.	1 Python project; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
52 LP_SP_52	**Race Day: Program a Precise Route.** Plan a multi-step race route and refine a Hopper's movement through testing and debugging.	1 Python project; 1 code screenshot; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 16 files total.
53 LP_SP_53	**Warm-Up Loop with Leo: Python For Loops.** Program repeated push-up movements and compare loops with duplicated commands.	1 Python project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 17 files total.
54 LP_SP_54	**Setting Conditions for Yoga: Python While Loops.** Use conditions and counters to repeat movement sequences and investigate loop errors.	1 Python project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 17 files total.
55 LP_SP_55	**Controlling Motion with Tilt: Gesture-Based Decisions.** Use Python conditions to react to Hub gestures and orientation in an interactive story game.	1 Python project; 2 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 17 files total.
56 LP_SP_56	**Guess Which Color: Multiple Conditional Choices.** Program a dog robot to respond to colours using if/elif/else, supported by flowcharts.	1 Python project; 2 code screenshots; 2 flowchart images; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 19 files total.
57 LP_SP_57	**Break Dancer Break Down: Find the Bugs.** Identify and fix syntax, logic, and runtime errors in a dance-robot program.	1 Python project; 1 code screenshot; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 16 files total.
58 LP_SP_58	**Score! Conditions, Variables, and Game Results.** Build a sensor-based scoring game and analyse results using conditions and lists.	1 Python project; 1 code screenshot; 1 flowchart image; 1 photo or model image; 1 .io model; 1 instruction PDF; 2 videos. 18 files total.

Sprint / ZIP	Descriptive title and learning content	Additional contents beyond the 10 teaching documents
59 LP_SP_59	**Clean Indicator: Functions with Parameters.** Use colour-coded inputs to model cleanliness levels while learning reusable functions and debugging parameter errors.	1 Python project; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.
60 LP_SP_60	**Mind Games: Memory and Data Lists.** Create a memory-and-reaction game using lists, conditions, and iterative debugging.	1 Python project; 3 code screenshots; 1 photo or model image; 1 .io model; 1 instruction PDF; 1 video. 18 files total.

A4: Assessment rubrics

Purpose and relationship to the assessment framework

This appendix provides three ready-to-use analytic rubrics for robotic performance, communication and collaboration, and responsible technology use. Together they reproduce all five categories and the Beginning, Developing, Secure and Advanced descriptors from Section 7 and Appendix A2 without changing their wording. The rubrics support assessment of the coded outcomes in Section 5 and A1; Appendix A2 explains progression and age-level calibration. They are complementary views of one assessment framework, not additional grading schemes.

How to use the rubrics

Before teaching, select the age level, sprint, one or two outcome codes and the rubric categories that the task can genuinely demonstrate. Agree task-specific success criteria and show learners accessible examples. Do not require every category to be assessed in every sprint.

Gather evidence during planning, construction, programming, testing, review and retrospective. Combine team artefacts with an individual explanation, demonstration or observation. A working team robot alone does not establish each member's understanding; a partly working model may still reveal strong reasoning.

Judge the evidence against the selected outcome and the relevant age expectations. Record one descriptor per assessed category or outcome, rather than averaging the four stages. No compulsory percentage weighting is introduced. Use Not yet assessed when evidence or opportunity is missing.

Record support separately. Communication aids, adapted input, visual instructions, processing time and other access arrangements remain available at every stage. Where an adult or peer supplies the reasoning being assessed, seek an additional individual demonstration. Do not grade personality, reading fluency, movement ability, speed or enthusiasm.

Use the first ten-sprint unit in each age group with full instructor guidance and the second with milder guidance that encourages self-initiative. Interpret achievement in that context: increased initiative is an evidence opportunity, not a reason to remove support or automatically award a higher descriptor. Revisit an outcome with a different task when necessary.

A4.1 Robotic performance and problem solving

Use for practical demonstrations, programming tasks and projects. Select relevant LO 6A/6B, LO 8A/8B or LO 10A/10B outcomes. Assess the algorithm and the testing process separately so that strength in construction does not conceal insecure code reasoning.

Category	Beginning	Developing	Secure	Advanced
Programming and algorithmic reasoning	Recognises some commands but cannot yet explain their sequence or effect.	Creates a partly effective algorithm and explains some choices with prompts.	Creates and explains a suitable algorithm using the level's expected structures.	Compares algorithms and justifies a clearer or more effective approach.
Robotics testing and improvement	Connects some components; predictions and tests need substantial guidance.	Builds a partly effective system and records tests with support.	Demonstrates intended behaviour and uses test evidence to justify a change.	Designs systematic comparisons and evaluates reliability, calibration or trade-offs.

Evidence and task criteria

Programming evidence: a sequence, diagram, pseudocode or program; a prediction; and an individual trace or explanation of a relevant command, loop, condition or other structure appropriate to the selected outcome. Ask: What should this part do? Why did you choose it? What would change if this value or instruction changed?

Robotics evidence: the build or mechanism, agreed target behaviour, a test record and a documented revision. Specify the intended action, safe operating conditions and relevant measure before testing. Ask: What happened compared with your prediction? What evidence supports your change? Where appropriate, repeat trials to examine reliability.

Feedback example: Your program stops the cart when the distance condition is met. Next, repeat the test from the same starting point and explain whether the threshold stops it reliably. This is an illustrative next step, not an extra outcome or a required sensor task for every sprint.

A4.2 Communication collaboration and reflection

Use for sprint planning, brief check-ins, product reviews, team retrospectives and individual explanations. Link teamwork to LO 6C, LO 8C or LO 10C outcomes and communication to the technical or responsible-use outcome being explained. Keep review of the product distinct from reflection on how the team worked.

Category	Beginning	Developing	Secure	Advanced
Communication and explanation	Shares an observation but links to the task are not yet clear.	Describes steps or results with prompts and emerging technical vocabulary.	Explains decisions and evidence clearly through an accessible communication format.	Adapts explanation to the audience and responds thoughtfully to questions.

Category	Beginning	Developing	Secure	Advanced
Teamwork and reflection	Participates in a role with frequent support; next steps need guidance.	Carries out tasks and identifies a useful contribution or improvement with prompts.	Fulfils responsibilities, listens and uses reflection to improve shared work.	Supports equitable participation and helps the team adapt using reflective evidence.

Evidence and task criteria

Communication evidence: an accessible oral, visual, written, signed or assisted explanation that connects a decision to the task and evidence. Invite questions without requiring a public performance where an individual conversation or recorded demonstration provides equivalent evidence. Assess clarity of meaning and reasoning, not accent, spelling or confidence.

Teamwork evidence: a role or task record, observed contribution, an example of listening or negotiating and an individual reflection identifying a realistic next step. Rotate meaningful responsibilities while retaining necessary adaptations. Ask: What did you contribute? How did the team decide? What will you change next time? Check individual evidence before using peer comments to inform a judgement.

Feedback example: You explained your test result clearly and listened to another solution. In the next plan, help the team allocate a meaningful task to each member and check that everyone can contribute. Avoid rewarding dominance or equating frequent speaking with effective collaboration.

A4.3 Ethical reasoning and responsible technology use

Use for design decisions, safe equipment routines, accessible game rules, data handling and real-world applications. Select relevant LO 6D, LO 8D or LO 10D outcomes. Assess a relevant decision and its explanation; a task need not cover safety, privacy, fairness, accessibility and sustainability simultaneously.

Category	Beginning	Developing	Secure	Advanced
Ethical reasoning and responsible use	Identifies a rule or user need with substantial prompting.	Applies some safe or inclusive choices and gives a simple reason.	Justifies a relevant safety, accessibility, fairness or sustainability decision.	Weighs competing needs or consequences and improves a design responsibly.

Evidence and task criteria

Evidence may include a demonstrated safety routine, a justified accessibility change, a fair rule, a privacy decision or a material-reuse choice. Ask: Who could use this solution? Who might be excluded? What information do we need? What should we avoid sharing? What

could we change responsibly? Use fictional or anonymous scenarios rather than collecting sensitive information for assessment.

Judge ethical reasoning through age-appropriate explanation and action. An isolated correct answer does not establish consistent safe practice. Intervene immediately when an action is unsafe; do not allow risk to continue to obtain evidence. Reteach the routine and arrange a safe reassessment opportunity.

Feedback example: You identified that sharing a photograph requires permission. Next, explain an alternative way to show the robot's work without identifying classmates. For advanced evidence, invite a justified comparison of options rather than expecting abstract terminology.

A4.4 Applying the same rubrics across the three age levels

Ages 6–8 Sprints 1–20 LO 6 outcomes

Use short sequences, concrete cause and effect, simple mechanisms and sensor responses where relevant. Accept demonstration, a picture sequence or a brief oral explanation. Full modelling is expected in Sprints 1–10; in Sprints 11–20 invite learners to choose a small change, propose a test or initiate a role with lighter prompts. Secure performance remains compatible with the age-appropriate guidance in Section 5.

Ages 8–10 Sprints 21–40 LO 8 outcomes

Look for decomposition, appropriate loops and conditions, recorded measurements and justified revisions. Variables or lists are assessed where the task and outcome require them. Model planning and systematic tests in Sprints 21–30; in Sprints 31–40 invite teams to organise tasks, choose control settings and use feedback with milder guidance.

Ages 10–12 Sprints 41–60 LO 10 outcomes

Look for appropriate Python reasoning, integrated mechanisms and sensing, systematic debugging and justified control or design decisions. Assess lists and functions when relevant to the selected outcome, not as compulsory features of every program. Provide technical modelling and Python support in Sprints 41–50; in Sprints 51–60 seek increasing initiative in planning, testing and evaluating trade-offs while retaining targeted instruction.

Advanced means deeper comparison, justified improvement or transfer within the same age level. It does not require older-level content, extra equipment or a larger model. Refer to Appendix A2 for the common age-level calibration examples.

A4.5 Recording evidence feedback and moderation

Use a concise assessment entry containing learner identifier, date, sprint and age level; outcome code; selected category; evidence reference; descriptor; support provided; one next step; and the next review opportunity. Keep team-product evidence linked to individual contributions. Store only necessary information under the school's access and retention arrangements.

Illustrative entry. Ages 8–10; Sprint 24; LO 8B2; Robotics testing and improvement: Secure. Evidence: the learner recorded brightness readings, changed the threshold and explained the effect on line response. Support: visual test table and one prompt to repeat a trial. Next step: compare repeated trials on two surfaces. Review: next suitable sensor task. This category judgement does not imply Secure attainment in every strand.

Learner self-assessment and peer feedback. Use accessible prompts: I can show…; My evidence is…; A change that helped was…; Next I will…. Peers describe observed evidence and suggest one helpful next step. Learners may discuss the rubric descriptors, but teachers retain responsibility for formal judgements. Do not require personal disclosures or public ranking.

Teacher and team reflection. After a sprint, note whether the task made the intended learning visible, which prompts supplied reasoning, who had limited opportunity and what must change. In co-teaching, the IT teacher and classroom teacher compare technical evidence and participation observations. Reflection informs teaching; it is not a score for pupil personality or compliance.

Moderation and reporting. At each ten-sprint checkpoint, compare a small sample of evidence with a colleague against the same outcome and descriptors. Record the reason for any revised judgement and arrange further evidence if uncertainty remains. At Sprints 20, 40 and 60 review the level's minimum evidence in Section 5. Report strengths, support used and next steps across the four strands without counting the same evidence repeatedly or imposing a compulsory numeric total.

A5: Glossary

This glossary explains key terms as they are used in the ALPER-EU curriculum. It is intended for instructors and supporting staff; definitions can be simplified, demonstrated or translated for learners. The classroom routines adapt Agile and Scrum ideas to educational robotics. They do not imply that every practice of professional Scrum is required.

Curriculum structure and learning

Age level. An ALPER-EU age band, not a national school grade: ages 6–8 follow Sprints 1–20; ages 8–10 follow Sprints 21–40; ages 10–12 follow Sprints 41–60.

Competence. The ability to apply relevant knowledge, practical skills and responsible judgement in a task or situation.

Cross-curricular integration. Coordinated learning that connects robotics and programming with relevant subjects, such as mathematics, science, language, arts or physical education. Subject links require local curriculum checking.

Learning outcome and LO code. A statement of what a learner can demonstrate. In LO 8B2, 8 identifies the starting age of the 8–10 band, B identifies the strand and 2 identifies the outcome; it is not a school-grade label.

Progression. Increasing depth, complexity and independence in learning over time. Progression depends on evidence and readiness as well as the planned sprint sequence.

Spiral progression. Revisiting important concepts in increasingly demanding contexts so that learners consolidate and extend earlier understanding.

STEAM. An approach connecting science, technology, engineering, arts and mathematics through purposeful inquiry, design and making.

Strand. A connected area of curriculum learning. ALPER-EU uses A Computational Thinking and Programming; B Robotics Systems and Physical Computing; C Agile Collaboration and Reflection; and D Responsible, Inclusive and Real-World Technology Use.

Transdisciplinary learning. Learning organised around a shared problem or context that draws on several disciplines rather than treating subject boundaries as the starting point.

Unit structure and ten-sprint cycle. A connected sequence of ten Agile sprints. Each age group has two units: the first with full instructor guidance and the second with milder guidance encouraging self-initiative. The six units cover Sprints 1–10, 11–20, 21–30, 31–40, 41–50 and 51–60.

Agile learning and teamwork

Agile learning. An iterative approach in which learners plan a manageable goal, develop and test a solution, use feedback and reflect before deciding what to improve.

ALPER Sprint Cycle. The classroom routine Connect → Plan → Build → Code → Test → Improve → Share → Reflect. Build, Code, Test and Improve are repeated as needed; the routine is not a one-pass checklist.

Backlog. A visible, prioritised list of possible tasks or improvements. A simple classroom task board can serve this purpose; a formal software tool is not required.

Check-in. A brief team conversation about progress, difficulties and the next action. It helps instructors identify support needs without repeatedly stopping all teams.

Iteration. A repeated attempt or revision informed by evidence or feedback.

Role rotation. Sharing responsibilities such as builder, programmer, tester or coordinator across time so learners gain meaningful experience. Rotation retains necessary accessibility adaptations.

Scrum. An Agile framework using goals, short work periods, inspection and adaptation. ALPER-EU uses an age-adapted learning cycle rather than requiring children to reproduce every professional Scrum role or event.

Self-initiative. Starting or proposing a useful action, strategy or improvement with increasing ownership. It can be demonstrated while using access supports or seeking appropriate help.

Sprint. A planned learning cycle with a manageable goal, practical development, testing, review and reflection. Its classroom duration is adapted to the activity and learners; it is not necessarily a single lesson.

Sprint goal. A clear statement of the result or learning challenge a team is trying to achieve during a sprint.

Sprint planning. Agreeing the goal, success criteria, tasks, responsibilities and resources before implementation.

Sprint retrospective. Reflection on how the team worked and learned, followed by an agreed improvement for the next sprint. It is distinct from reviewing the product.

Sprint review. A demonstration and discussion of the solution against the goal and success criteria, using feedback to guide further development.

Task board. A visual record of work, commonly using To do, Doing and Done. Pictures or physical cards can replace written or digital entries.

Programming and computational thinking

Algorithm. An ordered set of clear steps for solving a problem or completing a task. It may be represented through actions, pictures, pseudocode, a flowchart or code.

Block-based programming. Creating programs by arranging visual command blocks. Blocks make structure visible while learners develop sequencing and logic.

Computational thinking. Ways of analysing and solving problems through decomposition, patterns, abstraction and algorithms, including evaluating and debugging solutions.

Condition and conditional. A condition is a test that evaluates as true or false. A conditional, such as if, selects an action according to that result.

Debugging. Finding, explaining and correcting faults in an algorithm, program or robotic system, then checking the correction.

Decomposition. Breaking a complex challenge into smaller, manageable parts.

Event. An occurrence, such as a button press or sensor change, that can trigger a programmed response.

Flowchart. A diagram using connected symbols to show an algorithm's steps and decisions.

Function and parameter. A function is a named, reusable section of code. A parameter is a named input defined by a function; the value supplied when it is called is an argument.

Input and output. An input supplies information to a system, such as a sensor reading or button press. An output is an action or signal produced by it, such as motor movement, light or sound.

List. An ordered collection of values used to store or process several items, such as colours, instructions or game scores.

Loop. A programming structure that repeats instructions. A for loop commonly repeats over a sequence or range; a while loop continues while its condition is true.

Modularity and subroutine. Modularity organises a solution into manageable, reusable parts. A subroutine is a reusable section of instructions; functions are a common way to implement this idea.

Nested logic. A structure placed inside another structure, such as a condition inside a loop or one loop inside another.

Pseudocode. A clear, informal description of an algorithm using ordinary language and programming-like steps, without requiring the exact syntax of a programming language.

Python. A text-based programming language used in the older ALPER-EU age band, with explicit teaching and preparation. Learners must use commands compatible with the chosen robotics environment.

Sequence. Instructions arranged in the order in which they should be carried out.

Syntax runtime and logic errors. A syntax error breaks the language's structural rules. A runtime error arises during execution and prevents the intended operation. A logic error allows execution but produces incorrect behaviour.

Tracing. Following an algorithm or program step by step to explain changes in values, decisions and outputs.

Variable. A named reference used to hold a value that a program can read or update, such as a count, threshold or score.

Robotics systems and testing

Actuator. A component that turns a control signal into physical action; a motor is a common example.

Calibration. Checking readings or actions against a known reference and adjusting the system or its interpretation to improve accuracy.

Control parameter. A setting that influences behaviour, such as motor speed, movement duration or a stopping threshold.

Fair test. An investigation that changes a relevant factor while keeping other important conditions consistent, so comparisons are meaningful.

Hub. The programmable control unit that runs code and connects compatible motors and sensors in the robotics system.

Mechanism. Connected physical parts that transmit or transform movement and force.

Model and simulation. A model represents selected features of a real system. A simulation represents behaviour under defined rules. A classroom indicator or automation model does not automatically measure or reproduce the real-world phenomenon it represents.

Optimisation and trade-off. Optimisation improves performance against chosen criteria. A trade-off occurs when improving one feature, such as speed, affects another, such as accuracy or reliability.

Physical computing. Programming that interacts with the physical world through inputs, sensors and outputs.

Reliability. How consistently a system produces the intended result under specified conditions. Repeated trials help establish reliability.

Sensor. A component that detects a physical property or change and provides information to a program, such as colour, distance or force.

SPIKE Prime resources. The LEGO SPIKE Prime Set is the minimum viable robotics equipment for all three ALPER-EU age groups. The SPIKE Prime Expansion Set is an optional enhancement.

Test evidence. Recorded observations or measurements used to compare expected and actual behaviour and justify an improvement.

Threshold. A selected boundary value used to make a decision, such as stopping when a distance reading falls below a specified value.

Pedagogy inclusion and access

Accessibility. Designing resources, environments and interactions so learners can perceive, use and participate in them, including through assistive options.

Assistive and adaptive options. Tools or adjustments that enable participation, such as communication aids, alternative input devices or an adapted workspace.

Co-teaching. The recommended partnership in which an IT teacher leads technical instruction and the classroom teacher supports individuals and teams. Both share planning, inclusion and assessment responsibilities.

Differentiation. Adjusting task complexity, guidance, resources or response formats to learner readiness and needs while retaining meaningful learning goals.

Equity and inclusion. Equity provides the support each learner needs for a fair opportunity. Inclusion ensures learners belong, participate meaningfully and can demonstrate learning.

Gradual release of guidance. Moving from modelling and guided practice towards greater learner responsibility. ALPER-EU applies full guidance in the first unit and milder guidance in the second unit of each age group, while retaining responsive support.

Inquiry-based learning. Learning through questions, predictions, investigation, evidence and explanations.

Maker learning and design thinking. Maker learning develops understanding through constructing and revising artefacts. Design thinking uses attention to users' needs, idea generation, prototyping and testing to improve a solution.

Multiple entry points. Different accessible ways to begin the same challenge, such as a prepared build, picture algorithm or extension question, without making participation depend on one starting skill.

Project-based and problem-based learning. Project-based learning organises sustained work around producing and improving a meaningful outcome. Problem-based learning begins with a problem that drives investigation and learning. Both may contribute to ALPER-EU activities.

Scaffolding. Temporary instructional support, such as worked examples, prompts or partly completed code, adjusted as understanding grows. Necessary access arrangements are not removed merely to increase independence.

SEN. Special educational needs. Schools identify and support individual needs using their local procedures; a diagnosis alone does not specify the support a learner requires.

Student agency. Learners' meaningful influence over decisions, goals and actions within appropriate support and safety boundaries.

UDL. Universal Design for Learning: proactively providing varied ways to engage, access information and demonstrate understanding, reducing avoidable barriers.

Unplugged activity. An activity that develops computational ideas without running a computer program, such as acting out a sequence or sorting instruction cards.

Assessment evidence and responsible use

Artefact. An outcome of learning, such as a robotic model, program, algorithm diagram or test record. Artefact quality is interpreted alongside the learner's reasoning and contribution.

Assessment calibration and moderation. Calibration develops shared understanding of criteria through examples and discussion. Moderation compares judgements against agreed criteria and resolves differences using evidence.

Beginning developing secure and advanced. The four rubric descriptors used to describe demonstrated attainment against the selected age-level outcome. They are not age labels or compulsory numeric scores; Advanced means deeper reasoning within the level.

Feedback. Specific information about demonstrated learning and an actionable next step, followed by an opportunity to apply it.

Formative and summative assessment. Formative assessment uses evidence to adjust teaching and learning during the process. Summative assessment summarises achievement at a defined point. The same task can serve both purposes when used appropriately.

Minimum evidence. The required range of learning evidence specified for a level in Section 5. It is more than a finished team product and includes opportunities to show individual understanding.

Not yet assessed. A recording status used when evidence or a fair opportunity is missing. It is not equivalent to Beginning.

Performance task. A practical challenge in which learners apply knowledge and skills and explain their decisions.

Portfolio. A selected collection of artefacts, explanations, test records and reflections that shows learning and progress over time.

Privacy and data minimisation. Privacy protects people's information and choices about its use. Data minimisation means collecting and retaining only what is necessary for a stated purpose, following school procedures.

Reflection log. An oral, visual, written or assisted record of what was learned, what changed and what to try next.

Rubric. A set of assessment categories and descriptions of performance used to interpret evidence consistently. Section 7 and Appendices A2 and A4 share the same core descriptors.

Safeguarding. School procedures and actions that protect children's welfare and respond to concerns. Robotics teaching follows the school's established reporting and supervision arrangements.

Self-assessment and peer assessment. Learners use agreed criteria to reflect on their own evidence or offer constructive feedback on a peer's work. Teachers retain responsibility for formal attainment judgements.

Success criteria. Observable features that indicate whether a task's intended goal or learning outcome has been achieved.

Sustainability. Considering resource use, reuse, waste and longer-term consequences when making and evaluating design choices.

A6: Resource list + links

This document was generated with the use of AI. The following sources were used:

European Commission, Joint Research Centre. (2022). *DigComp 2.2: The digital competence framework for citizens: With new examples of knowledge, skills and attitudes*. Publications Office of the European Union. https://doi.org/10.2760/115376

Computer Science Teachers Association. (2017). *CSTA K–12 computer science standards*. https://csteachers.org/k12standards/

K–12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. https://k12cs.org/

Lego Education. (2026). *Lego Spike Prime Set*. https://education.lego.com/en-us/products/lego-education-spike-prime-set/45678/

Organisation for Economic Co-operation and Development. (2019). *OECD Learning Compass 2030*. OECD. https://www.oecd.org/en/about/projects/future-of-education-and-skills-2030.html

Organisation for Economic Co-operation and Development. (2019). *Student agency for 2030*. OECD. https://www.oecd.org/content/dam/oecd/en/about/projects/edu/education-2040/concept-notes/Student_Agency_for_2030_concept_note.pdf

UNESCO. (2022). *K-12 AI curricula: A mapping of government-endorsed AI curricula*. https://www.unesco.org/en/articles/k-12-ai-curricula-mapping-government-endorsed-ai-curricula

UNESCO. (2023). *Guidance for generative AI in education and research*. https://www.unesco.org/en/articles/guidance-generative-ai-education-and-research

European School Education Platform. (2025, February 10). *Educational robots for inclusive skills development using the RIDE method*. https://school-education.ec.europa.eu/en/learn/courses/educational-robots-inclusive-skills-development-using-ride-method

ALPER-EU. (2025, December 1). *1st press release – ALPER-EU project launched!* https://alper-eu.hmu.gr/1st-press-release-alper-eu-project-launched/