LESSON 105

Building the Embedded System

120 minutes (3 periods) | Coursework Project Preparation | Outcome 1.1 | Outcome 1.3 | Outcome 1.19

WHAT THIS LESSON DOES

Please note: this module follows the 2026 project brief, whose theme was forests. The State Examinations Commission publishes a new brief each year, with its own theme and its own dates. These lessons will be updated once this year's brief is published. Check the current brief on the SEC website before you begin. In this lesson, you will build a functional embedded system for your forest-themed project. Connect at least one digital and one analogue input, configure an output, implement code for automatic data collection and process simulation (e.g., wildfire risk), and test thoroughly. By the end, your system will meet all 2026 brief requirements, with evidence saved for your report.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.1	describe a systematic process for solving problems and making decisions
1.3	solve problems by deconstructing them into smaller units using a systematic approach in an iterative fashion
1.19	identify features of both staged and iterative design and development processes

WHAT STUDENTS SHOULD BE ABLE TO DO

- Construct a functional embedded system incorporating at least one digital input, one analogue input, and one primary output.
- Implement automated operation following manual calibration to collect and store real environmental data related to a forest theme.
- Develop code to simulate real-world processes, such as canopy cover, drought cycles, or wildfire risk, based on sensor inputs.
- Test and verify system responses to varying environmental conditions and inputs.
- Document the system build, including wiring, code, data logs, and evidence of functionality.

BEFORE THE LESSON

- Test one full build yourself first: a digital input, an analogue input and an output on the class board, and confirm the storage method actually writes a CSV file. Flash and card failures eat the whole class.
- Have spare cables, at least one spare sensor of each type, and confirmed working microSD cards or USB drives. This is a two-hour practical and a single dead cable stalls a student for the period.
- Set up the shared Artefact and Embedded_System_Evidence folder structure students save into, and confirm they can save main.py where you can reach it.

Building the Embedded System

HOW THE LESSON RUNS

undefined

Exact Basic Requirements You Must Meet

Hardware Checklist, Confirm Before Starting

Core Code Structure You Should Follow

Practical Building Time

Quick Testing Checklist

What to Save and Photograph

Checklist Before Next Lesson

TEACHING MOVES

- 2. Exact Basic Requirements You Must Meet. Read the six basic requirements aloud from the brief and have students tick which they can already meet today. Stress that all six are non-negotiable minimums, not a menu, and that the project is the graded 30%, not an ALT.
- 3. Hardware Checklist, Confirm Before Starting. Make everyone complete the hardware column against their own kit before touching code. A student missing an analogue sensor now will fail requirement two later. Fix supply gaps before anyone starts wiring.
- 4. Core Code Structure You Should Follow. Walk the five sections as a skeleton: imports, pin definitions, one-time calibration, then the automatic loop. Emphasise calibration runs once, the loop runs forever. Let them adapt pins and language, but every section must be present.
- 5. Practical Building Time. Insist on the stated order: test each sensor with printed values before adding anything else. Students who wire everything at once cannot tell which part failed. Only add logging once both inputs print sensible numbers.
- 6. Quick Testing Checklist. Circulate with this list. Do not accept a tick on trust: watch the output fire under the correct condition and open the data file to confirm multiple real rows, not one repeated line.
- 7. What to Save and Photograph. Reserve the last fifteen minutes for evidence capture so it is not rushed at home when the rig is packed away. Check the CSV screenshot shows several changing readings and the video shows the system responding to a real change.

WHAT TO WATCH FOR

- Students treat an analogue reading that never changes as a working sensor, so their simulation logic never triggers. Have them print raw values and physically change conditions: wet then dry soil, cover the light sensor. If the number does not move, the wiring or pin is wrong before the code is.
- The simulation is hard-coded to always report high risk, so it looks impressive but proves nothing. Ask them to demonstrate the output switching off. A system that cannot reach a safe state has not simulated a real-world process and fails the testing requirement.
- Calibration code is placed inside the main loop, so the system re-prompts forever and never runs automatically. Point to requirement six: automatic operation after a manual start. Move the calibration or button-wait above the loop so it executes exactly once.

DIFFERENTIATION

- Emerging: Give them a single sensor to get printing correctly before anything else, and pair them with someone whose board is already reading data. One working input is progress worth banking. Provide the pin-definition and calibration sections as a starting skeleton so they spend their time on wiring and logic rather than boilerplate.
- Developing: Have them explain out loud what each line of their loop does before you sign off the testing checklist, so they can reason about the code and not just run it.
- Proficient: Push them toward robust logging: timestamps, error handling if the storage fails, and averaging noisy analogue readings. This is exactly the evidence quality a strong Coursework Project write-up needs. Ask them to add a second simulated process, for example both drought cycle and wildfire risk from the same inputs, and document how the two interact.