LESSON 99

Building and Automating the System

240 minutes (6 periods) | Applied Learning Tasks (ALTs) | Outcome 3.11 | Outcome 3.12 | Outcome 3.13 | Outcome 3.14 | Outcome 1.22 | Outcome 2.20 | Outcome 2.21

WHAT THIS LESSON DOES

In this lesson, you'll build and automate your embedded system by following clear steps. Assemble hardware, load initial code, add automation logic, test components, and refine for performance. Turn your design into a functional prototype ready for evaluation.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.11	use and control digital inputs and outputs within an embedded system
3.12	measure and store data returned from an analogue input
3.13	develop a program that utilises digital and analogue inputs
3.14	design automated applications using embedded systems
1.22	read, write, test, and modify computer programs
2.20	identify and fix/debug warnings and errors in computer code and modify as required
2.21	critically reflect on and identify limitations in completed code and suggest possible improvements

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the process of assembling hardware components for an embedded system.
- Develop skills in loading and running initial code to verify hardware functionality.
- Implement automation logic to enable systems to respond independently to inputs.
- Refine and optimise both hardware and software for improved system performance.
- Prepare a stable prototype for comprehensive testing and evaluation.

BEFORE THE LESSON

- Confirm each group's microcontroller, sensors, actuators, breadboard, jumper wires, resistors and power source are physically present and match their design plan from earlier lessons.
- Test that the development environment (Arduino IDE or your board's equivalent) opens and can upload to a board on the actual room machines, and have the correct USB cables ready.
- Make sure each group can retrieve their initial code and design sketches from earlier lessons; anything not saved to a shared location is lost time here.
- This is a 240 minute build. Plan checkpoints so groups stuck at wiring do not lose the whole session before they reach the code.

Building and Automating the System

HOW THE LESSON RUNS

Introduction
Gather Components
Assemble Hardware
Load and Run Initial Code
Add Automation Logic
Optimise and Refine
Prepare for Full Testing

TEACHING MOVES

- 2. Gather Components. Have each group lay out and tick off components against their own plan before touching a wire. A missing sensor found now is an inconvenience; found mid-wiring it stalls the group.
- 3. Assemble Hardware. Insist groups check their design sketch for which pin each part uses before connecting. Walk the room checking analogue versus digital pin choices and that connections are firmly seated, not loose in the breadboard.
- 4. Load and Run Initial Code. Have them upload the basic code first and confirm it runs before adding anything. If the LED does not blink or the sensor reads nothing, the fault is wiring or board selection, not their logic.
- 5. Add Automation Logic. Ask each group to state their automation rule aloud before coding it: what input, what threshold, what action. Then they write the if-else. This stops vague code that reacts to nothing specific.
- 6. Optimise and Refine. Prompt groups to watch their system run and name one concrete improvement: a delay to shorten, a repeated block to tidy, unnecessary polling to reduce. Optimise against observed behaviour, not guesswork.
- 7. Prepare for Full Testing. Have each group run the full system for a sustained period to catch instability, then document the final code, setup and any known issues with a photo. This documentation feeds their project evidence.

WHAT TO WATCH FOR

- Students assume that if the code compiles and uploads, the hardware must be working. A silent sensor or wrong pin gives no error but produces no behaviour. Separate the two: upload code confirms the board accepts it, observed output confirms the wiring. Get them to print raw sensor readings before trusting any automation built on them.
- Confusing assignment with comparison in the automation logic, writing if (temp = 30) instead of if (temp > 30) or if (temp == 30). Point at the condition and ask what value they are testing against and whether it should be greater, less or equal. Reserve single equals for setting a value.
- Wiring a sensor to a digital pin when it needs an analogue input, then wondering why readings are only ever 0 or 1. Send them back to the board pinout and their design sketch. Analogue sensors need analogue input pins; the reading range tells you if it is wrong.

DIFFERENTIATION

- Emerging: Pair a student whose wiring is fighting them with a group that has a working circuit, and have them compare connection by connection against the sketch. Give a single, minimal target first: get the sensor reading to appear, nothing more, before any automation.
- Developing: Ask them to explain what each line of their if-else does before adding the next rule, so they reason about the logic rather than copy a pattern.
- Proficient: Challenge them to handle an edge case or add a second condition, then justify their threshold choice as they would defend it in the Coursework Project. Have them profile the response time and rewrite one section to run faster or use less power, documenting the before and after as project evidence.