STEM ARDUINO PBL — PROJECT 4 | TIER 2: INTEGRATION

Smart Home Mini System

Full PBL Lesson Plan, Assessment Rubric & Teaching Resources

Grades 8–10 • 3 Weeks • SDG 7, SDG 9

PROJECT SNAPSHOT	
Driving Question	"How can we design an integrated home automation system that genuinely improves energy efficiency and convenience for a real household — and what trade-offs does that require?"
Real-World Angle	Intro to IoT and home automation — connecting remote control, occupancy sensing, relay switching, and real-time status display into a single integrated system rather than four separate circuits
Grade Level	Grades 8–10 (requires Projects 1–3 as prerequisites: IR control, sensor reading, state machines, and LCD display are all assumed knowledge)
Duration	3 weeks (15 × 45-minute lessons, or equivalent block periods)
SDG Connections	SDG 7 — Affordable & Clean Energy | SDG 9 — Industry, Innovation & Infrastructure
Key Components	Arduino UNO · IR remote + receiver · 5V relay module · PIR motion sensor · 16×2 LCD (I2C) · LEDs + resistors · (Optional: DHT22 sensor for Upgrade C)
Core Skills	Systems integration · Multi-input logic · Relay control · State machine design · LCD status display · Mode switching · Energy-saving auto-off logic
21st C. Skills	Systems thinking · Trade-off analysis · Client-centred design · Technical documentation · Oral presentation to a non-technical household client
Public Audience	A household 'client' — ideally a parent, school facilities manager, or a member of the public unfamiliar with Arduino
Entry Products	Household audit report + system architecture diagram + feature prioritisation document
Final Products	Working integrated Smart Home prototype + system architecture diagram + energy savings estimate + client demonstration

CURRICULUM STANDARDS COVERAGE	
Technology / ICT	Systems integration: combining multiple independent subsystems into a coherent whole · Multi-input logic: handling simultaneous inputs from different sensor types · State machine design with multiple modes · Relay switching: controlling mains-equivalent loads safely · I2C communication revisited with system-level context

Science / Physics	Electrical switching and relay operation (electromagnetic actuation) · Energy consumption and conservation · Occupancy-based control as a strategy for demand reduction · Signal types: digital (IR, PIR, relay), analogue (optional: temperature), I2C (display)
Mathematics	Energy consumption calculations: power (W) × time (h) = energy (kWh) · Cost of electricity per kWh · Estimating annual savings from automated switching · Proportional reasoning: what fraction of standby energy could automation eliminate?
Design Technology	Iterative subsystem integration (build one subsystem, test, add the next) · Failure mode analysis across a multi-component system · Client brief interpretation and feature prioritisation · Trade-off documentation: which features were excluded and why
Environmental Sci.	Standby power consumption as a contributor to household energy waste · IoT automation as an energy-reduction strategy · Comparison: smart home systems vs behaviour change vs nothing
Literacy	System architecture documentation · Client-facing demonstration script · Technical trade-off justification · Energy savings report written for a non-technical household

Why Project 4 Is Where the Real Engineering Begins

Projects 1–3 built one subsystem at a time. Project 4 is the first project in the sequence that requires students to **integrate multiple subsystems simultaneously** — and to manage the design complexity that integration creates. This is a qualitative shift in difficulty. The individual components (IR receiver, PIR sensor, relay, LCD) are all familiar from earlier projects. What is new is the question: **how do you make them work together without breaking each other?** That question is the heart of systems engineering, and it cannot be answered by good component knowledge alone.

PBL ELEMENT	HOW IT APPEARS IN THIS PROJECT
Centrality	The household automation challenge is the entire unit. Students learn multi-input logic, relay control, and state machine complexity because those tools are necessary to answer the driving question for a real household client — not as exercises.
Driving Question	The question names both the outcome (energy efficiency and convenience) and the constraint (trade-offs). Students must choose which features to include and justify the exclusions — this is a professional engineering skill, not just a build challenge.
Constructive Inquiry	What happens when the IR remote sends a command while the PIR has auto-triggered the lights? Who wins? Students cannot answer this without designing and testing the mode logic — the inquiry is built into the integration challenge.
Student Autonomy	Teams choose their household scenario, decide which modes to implement, determine the auto-off timeout duration, and design their own LCD layout. The teacher defines integration as the goal; students define what integrated means for their client.
Realism / Authenticity	The household client is real. The energy waste problem is documented (UK households leave lights on for an average of 4 hours per day in unoccupied rooms).

	The energy savings calculation is done with actual electricity tariff data. The client demonstration is not a classroom exercise.

Week-by-Week Lesson Calendar

WEEK 1: ENTRY EVENT, ARCHITECTURE & SUBSYSTEM INQUIRY			
Lesson	**PBL Phase**	**Learning Activities**	**Teacher Facilitation Notes**
L1	**Entry Event**	• Launch: Show real data — UK households waste £150/year in standby and forgotten lighting; smart home market projected at $170B by 2025 • Demo: show a commercial smart home product (Google Nest, Amazon Alexa, Philips Hue) — reveal the price point (£100–500+) • Pose the Driving Question. Challenge: 'Could we build a system that does the core functions for under £20?' • Household audit activity: students audit their own home — where are lights left on? Which devices have standby power? How many rooms are empty when lights are on?	→ The price gap between commercial and DIY is the entry hook — make it explicit: £300 smart home kit vs £15–20 Arduino equivalent → Household audit connects the project to students' lived experience immediately → Do NOT show the components yet — let the audit identify what the system needs to do first
L2	**Entry Event**	• Household audit debrief: teams share findings — what are the biggest energy waste patterns? • Feature prioritisation: given a hard budget of £20 materials, what must the system do? what would be nice? what is out of scope? • Introduce the client brief: teams will present their final system to a household 'client' in Week 3 • Entry product: Household Audit Report + Feature Prioritisation Document (MoSCoW: Must have, Should have, Could have, Won't have)	→ MoSCoW prioritisation is a real project management tool — name it and explain why it exists → Push back on wish lists: 'You want 8 features. You have 3 weeks. Which 3 features matter most to your client?' → Formative: feature prioritisation quality — are trade-offs justified with client reasoning?
L3	**Inquiry — Architecture**	• Mini-lesson: what is a system architecture? How do engineers plan integration before building? • Students draw their system architecture diagram: inputs, Arduino, outputs, signal types • Identify: where does the relay fit? Why can't the Arduino control a real lamp directly? • Plan the build sequence: which subsystem first? Why does order matter for integration?	→ Architecture diagram on paper BEFORE any component is touched — this is non-negotiable for integration projects → Ask: 'If you built all four subsystems simultaneously and nothing worked, how would you debug it?' — answer: you couldn't. Build sequence is a diagnostic strategy. → Formative: architecture diagram — is signal flow labelled? Is relay justification present?
L4	**Inquiry — Relay Module**	• Mini-lesson: what is a relay? How does an electromagnetic coil switch a separate electrical circuit? • Wire relay module to Arduino: coil side (5V logic from Arduino), switch side (the load — use an LED as a safe stand-in for a lamp) • Test: digitalWrite(RELAY_PIN, HIGH) → relay clicks → LED on; LOW → relay releases → LED off	→ Safety first: in this project, the relay switches only 5V LEDs — never mains voltage in a classroom → The relay 'click' sound is a teaching moment — students can

		• Investigate: what is the difference between normally-open (NO) and normally-closed (NC) relay contacts? When would you use each?	hear the electromagnetic switching → Key question: 'Why does the relay need a 5V coil signal to switch the load? Why not connect the load directly to the Arduino pin?'
L5	**Inquiry — Mode Logic**	• Introduce the core design challenge: 'Your system has two control inputs — IR remote and PIR sensor. They can conflict. Who wins?' • Define two modes: MANUAL (IR remote controls relay directly) and AUTO (PIR controls relay, IR remote can override) • Students draw the mode state machine: MANUAL → AUTO toggle, relay behaviour in each mode • Write pseudocode for the mode logic before writing Arduino code	→ The mode conflict question has no single right answer — different design decisions produce different valid systems → Ask: 'If you're in manual-OFF mode and someone walks in — should the light turn on automatically?' This surfaces the design values, not just the logic → Formative: mode state machine diagram + pseudocode — is every mode transition accounted for?

WEEK 2: INTEGRATION BUILD & ITERATIVE TESTING

Lesson	PBL Phase	Learning Activities	Teacher Facilitation Notes
L6	**Build — Subsystem 1**	• Build Subsystem 1: IR receiver → Arduino → relay → LED (manual control only, no PIR yet) • Test: does pressing button A turn the relay on? Does button B turn it off? • Add LCD: display 'Mode: MANUAL' and 'Device: ON/OFF' updating in real time • First milestone check: Subsystem 1 working independently before adding Subsystem 2	→ Milestone check is deliberate — students who skip it and add all subsystems simultaneously cannot debug effectively → LCD layout design: ask 'If the homeowner glances at this from across the room, what must they see instantly?' → Formative: Subsystem 1 test evidence — IR → relay → LED chain verified before proceeding
L7	**Build — Subsystem 2**	• Add Subsystem 2: PIR sensor → auto mode logic • Implement mode switching: one IR button toggles MANUAL ↔ AUTO • Test: in AUTO mode, does motion trigger the relay? Does the LCD update to 'Mode: AUTO — Motion: YES'? • Integration test: switch between modes with PIR active — does the system behave as designed?	→ Expected problem: IR and PIR inputs need to be checked in the same loop without either blocking the other — introduce non-blocking pattern → Ask: 'What happens if you receive an IR signal and a PIR trigger at exactly the same millisecond?' This surfaces the single-threaded nature of Arduino → Formative: Subsystem 2 integration test — mode switching verified with both inputs active
L8	**Build — Full Integration**	• Full system integration: all subsystems running simultaneously • Extended integration test: run 20 scenarios from the test matrix (see Handout 3)	→ The 20-scenario test matrix is the most important quality assurance step in this project — do not skip

		• Identify failures: which scenarios fail? What is the root cause? Document in debugging log • Peer system review: partner team runs 5 scenarios — can they break the system?	→ Common failure: LCD not updating when mode changes because update code is inside an if-block that only runs on certain conditions → Formative: completed test matrix with pass/fail for each scenario and identified root causes for failures
L9	**Upgrade Challenge**	• Introduce differentiated upgrade options: • Upgrade A: Auto-off timer — if device stays on in AUTO mode with no new motion for 5 minutes, relay switches off automatically (uses millis()) • Upgrade B: Activity log — LCD cycles through last 3 events (e.g. 'IR: ON 14:23', 'PIR: trigger 14:31', 'Auto-off 14:36') with timestamps using millis() • Upgrade C: Temperature-triggered mode — add DHT22; if temperature > threshold, auto-on the relay (e.g. fan control); display temperature on LCD alongside device status • Teams select upgrade, plan architecture extension before building	→ Upgrade A is the most practically useful — auto-off addresses the core energy waste problem from the entry event → Upgrade C adds a third sensor type and brings the system closer to the Tier 3 multi-sensor dashboard → Formative: upgrade architecture extension — how does the new subsystem connect to the existing state machine?
L10	**Upgrade & Refinement**	• Build and test chosen upgrade • Refinement: address any remaining failures from the Lesson 8 test matrix • Energy savings calculation: using real electricity tariff data, calculate annual saving from auto-off • Peer critique: partner team uses the system as the household client would — what is confusing? What works well?	→ Energy calculation must use real numbers: find the local electricity tariff (Qatar: approximately 0.028 QAR/kWh; UK: approximately 24p/kWh) → Ask: 'Does the Arduino itself use energy? How much? Does it consume more energy than it saves?' → Formative: energy savings calculation with assumptions documented; peer critique form

WEEK 3: DOCUMENTATION, ENERGY ANALYSIS & CLIENT DEMO

Lesson	PBL Phase	Learning Activities	Teacher Facilitation Notes
L11	**Documentation**	• Finalise system architecture diagram — update to reflect what was actually built (not the original plan) • Write user operation guide: how does the household client use the system day-to-day? • Document feature log: what was planned but excluded? Why? What would be added with more time? • Review: does the documentation reflect the actual system, not the intended system?	→ Architecture diagram update is important — the built system is never identical to the plan; documenting the difference is professional practice → User guide must be written for a non-technical homeowner — test it with a non-technical person → Formative: documentation completeness review against rubric criterion 6
L12	**Energy Analysis**	• Finalise energy savings report: device power consumption × hours saved per day × days per year × electricity tariff	→ Payback period is a real financial metric — name it and explain its significance

		• Calculate payback period: total component cost ÷ annual energy savings = years to break even • Compare: this system vs a commercial equivalent — what does the commercial system offer that this one does not? • Honest assessment: under what conditions does this system actually save energy? When might it not?	→ Honest assessment is important: if the Arduino runs continuously and the device is only on for 1 hour per day, the Arduino may consume more energy than it saves. Students should calculate this. → Formative: energy savings report with all assumptions explicitly stated
L13	**Client Demo Prep**	• Prepare client demonstration: open with the household energy problem, not the circuit • Structure: Problem → What our system does → Live demo (client operates it) → Energy savings in pounds/year → What we would add next • Practice Q&A: teacher plays sceptical household client ('What if the internet goes down?' 'What if I want the light on when I'm sitting still?') • Prepare: what happens if the relay fails during the demo? Have a contingency.	→ The client operates the system during the demo — not the student. This is the key design constraint for the demonstration. → Q&A preparation: 'What if I want the light on when sitting still reading?' is the most common and most valid client concern — teams must have a designed answer, not just an apology → Formative: demo run-through with teacher as client; note weakest moments
L14	**Client Demonstration**	• Client demonstration: teams present to household 'client' audience • Structure: 5-minute presentation + 3-minute client operation + 2-minute Q&A • Client completes feedback form (see Handout 5) • Audience notes: which team's system would they actually install in their home?	→ Invite a parent, school facilities manager, or community member — someone who genuinely cares about energy costs → Teacher role: facilitator only — do not rescue teams during client operation → Summative: demonstration rubric (criteria 4 + 5)
L15	**Reflection & Submission**	• Individual written reflection (see Handout 6) • Class discussion: 'What does integration complexity teach us about professional engineering?' • Preview Tier 2 progression: how does Project 4 connect to Projects 5, 6, and 7? • Final portfolio submission: architecture diagram, documentation package, energy report, reflection	→ Reflection must surface the integration challenge specifically — not just 'I learned about relays' → Tier 2 preview: Projects 5–7 each take one aspect of the Smart Home system and develop it into a standalone integration project → Summative: documentation package + reflection (criterion 6)

Detailed PBL Phase Plans

Phase 1 — Entry Event: From Individual Components to a System		
SUB-PHASE / TASK	**TEACHER ACTIONS**	**STUDENT ACTIONS + PRODUCTS**
Entry Event Launch (Lessons 1–2)	Reveal the cost gap between commercial smart home systems and a	Conduct household audit. Identify top 3 energy waste patterns. Complete

	DIY equivalent before any component is shown. Ask: 'What does a £300 smart home kit actually do? Which of those functions do households actually use?' Let students define the value proposition.	MoSCoW feature prioritisation for their specific household scenario. Product: Household Audit Report + Feature Prioritisation Document.
Architecture Planning (Lesson 3)	Introduce the architecture diagram requirement with a professional framing: 'No engineer starts building a multi-component system without a plan. Your architecture diagram is how you think before you build.' Refuse to allow component distribution until the diagram is complete.	Draw complete system architecture: inputs (IR, PIR), processor (Arduino), outputs (relay, LCD, LEDs), signal types, voltage levels. Draw mode state machine. Write pseudocode for mode logic. Product: System Architecture Diagram + Mode State Machine + Pseudocode.
Entry Assessment	Check: does the architecture diagram show signal flow, not just component names? Does the mode state machine account for all transition scenarios, including conflicts? Does the feature list match what the architecture can actually deliver?	Self-check: 'Could another team build our system from our architecture diagram without asking us any questions?' If not — what is missing?

Phase 2 — Inquiry: Understanding the New Components

SUB-PHASE / TASK	TEACHER ACTIONS	STUDENT ACTIONS + PRODUCTS
Relay Module (Lesson 4)	Ask: 'An Arduino pin can source 40mA maximum. A typical lamp draws 500–1000mA. What happens if you connect a lamp directly to an Arduino pin?' Let students reason through the problem before introducing the relay as the solution.	Wire relay module; test NO and NC contacts; document: what is the coil voltage? what is the maximum switching current? when would you choose NC over NO? Product: Relay Investigation Notes with NO/NC comparison.
Mode Logic Design (Lesson 5)	Present the design problem without a solution: 'Your system has two controllers — a remote and a sensor. They will sometimes disagree. Design the rules for who wins.' Different teams may reach different valid designs — celebrate this diversity.	Design mode logic with explicit conflict resolution rules. Write pseudocode. Test against at least 5 edge case scenarios on paper before coding. Product: Mode Logic Pseudocode + Edge Case Analysis.
Inquiry Assessment	Update Need to Know board. Ask: 'What would happen to your system if both the IR signal and a PIR trigger arrived at the same time? How does your code handle that? How did you test that it handles it correctly?'	Peer review of architecture diagrams: swap with another team. Can you identify a scenario their mode logic does not handle? Feed back as a written 'stress test' question for the other team to address.

Phase 3 — Build & Iterate: Incremental Integration

SUB-PHASE / TASK	TEACHER ACTIONS	STUDENT ACTIONS + PRODUCTS
Subsystem 1 Build (Lesson 6)	Enforce the build sequence: IR → relay → LCD only. PIR is NOT connected yet. Ask: 'Why does order matter? What problem does incremental integration solve?' Students who understand this are doing systems engineering, not just wiring.	Build and fully test Subsystem 1 before any further integration. Document: what is the exact IR hex code for each function? What does the LCD show in each state? Product: Subsystem 1 Test Evidence (verified before L7 begins).
Subsystem 2 Integration (Lesson 7)	Before adding the PIR: ask 'What new failure modes does adding a second input create?' Students should predict at least two before they begin. Compare predictions to actual failures after integration.	Add PIR sensor; implement mode switching; run integration test. Document: which predicted failures occurred? Which were unexpected? What caused each? Product: Integration Test Log with failure mode analysis.
Full System Test (Lesson 8)	Introduce the 20-scenario test matrix (Handout 3) as a quality assurance tool. Ask: 'A professional engineer would not release a system that had only been tested in 3 scenarios. Why?' Make the test matrix a genuine quality checkpoint, not a box-ticking exercise.	Execute all 20 test scenarios. Document pass/fail for each. Identify root cause for all failures. Prioritise: which failures must be fixed before the client demonstration? Which are acceptable limitations to document? Product: Completed 20-Scenario Test Matrix with Root Cause Analysis.
Build Assessment	Observe: are students debugging systematically (isolate → hypothesise → test → confirm) or randomly changing code? Systematic debugging is a professional skill and is assessed through the debugging log, not just whether the system works.	Self-assess: 'What is the single biggest integration challenge we faced? What did it teach us about multi-component system design that we could not have learned from building one component alone?'

Phase 4 — Upgrade Challenge: Extending the Integration

SUB-PHASE / TASK	TEACHER ACTIONS	STUDENT ACTIONS + PRODUCTS
Upgrade A — Auto-Off Timer (Core+)	Connect to the energy audit: 'Your client's biggest waste was lights left on when rooms are empty. Your PIR detects motion — but what happens after motion stops and the person leaves? How long should you wait before switching off?' Let the client need drive the timer value.	Implement millis()-based auto-off timer: if AUTO mode and no new PIR trigger for N minutes, switch relay off and update LCD. N is configurable — teams choose based on their client scenario. Product: Auto-off demo with configurable timeout and energy savings recalculated with auto-off included.
Upgrade B — Activity Log (Advanced)	Introduce the concept: 'A professional building management system logs every event. Why? Who reads the log? What decisions does it enable?' Connect logging to accountability and	Implement circular buffer of last 3 events with millis()-derived timestamps. LCD displays events on button press. Serial Monitor shows full log. Product: Activity log demo with at

	the ability to optimise the system based on real usage patterns.	least 10 logged events shown; analysis paragraph: what does the log reveal about how the system is actually used?
Upgrade C — Temperature Mode (Extended)	Frame as a new client scenario: 'Your client wants to automate a fan as well as the lights. The fan should run automatically when the room is above 26°C, regardless of motion. How does this change your mode logic?' A new requirement that requires architecture revision.	Add DHT22; implement temperature-triggered mode alongside existing manual/auto modes; display temperature on LCD second line. Architecture diagram updated to reflect new subsystem. Product: Three-mode system demo (manual, auto-PIR, auto-temperature) with updated architecture diagram.
Upgrade Assessment	Upgrade A: assessed on auto-off timer accuracy (verify with stopwatch) and energy savings recalculation. Upgrade B: assessed on log accuracy and the quality of the usage analysis paragraph. Upgrade C: assessed on three-mode logic correctness and architecture diagram update quality.	Architecture diagram must be updated to show upgrade integration. Failure to update the diagram is a documentation failure, not a build failure — both matter.

Phase 5 — Client Demo & Documentation: Professional Delivery

SUB-PHASE / TASK	TEACHER ACTIONS	STUDENT ACTIONS + PRODUCTS
Energy Analysis (Lessons 11–12)	Teach: 'Your client does not care about kWh. They care about pounds per year. Convert every energy number into money.' Model the conversion: 60W bulb × 2 hours saved per day × 365 days × local tariff = annual saving in pounds.	Complete energy savings report. Calculate payback period. Write honest assessment: when does the system save energy and when might it not? Product: Energy Savings Report with real tariff data, assumptions documented, payback period calculated.
Client Demo Preparation (Lesson 13)	Run a full practice demo with teacher playing the most sceptical possible household client. Focus on the 'sitting still reading' objection — every team must have a prepared, designed response, not just an acknowledgement of the limitation.	Prepare and rehearse 5-minute presentation + 3-minute client operation + Q&A. Client operation must be genuinely client-led — the student explains, the client presses the buttons. Product: Rehearsed demonstration with contingency plan.
Client Demonstration (Lesson 14)	Introduce the client audience. Remind students: the client's experience of operating the system is the most important 3 minutes of the demonstration. Do not intervene.	Deliver client demonstration. Client operates the system. Record: what did the client find intuitive? What confused them? What did they ask that was not anticipated? Product: Client demonstration + audience feedback forms.
Reflection (Lesson 15)	10 quiet minutes for individual written reflection before any class discussion.	Complete Handout 6. Required: what integration challenge was most

	Reflection should specifically address integration complexity — what was different about building a multi-subsystem project vs a single-component project?	difficult and why; what would be different about the architecture if starting again; how does this project connect to what comes next in the sequence. Product: Individual Reflection Document.

Project 4 Assessment Rubric

Distribute on Lesson 1. *Project 4 introduces Criterion 2: Systems Thinking & Architecture Design. This criterion is new to the sequence and carries 20% weight because architecture planning is what separates integration projects from recipe-following. A student who builds a working system without planning it has done less engineering than one who planned carefully and encountered manageable failures.*

CRITERION	1 — Beginning	2 — Developing	3 — Proficient ✓	4 — Exemplary
Systems Integration & Technical Functionality (25%)	Fewer than two subsystems function; components work in isolation but cannot be combined; major integration errors prevent the system from operating as a whole.	Two or more subsystems integrated but with significant reliability issues; IR control OR PIR auto-lights work but not both simultaneously; LCD displays static text but does not update to reflect system state.	All three core subsystems integrated and functioning reliably: IR remote controls relay on/off; PIR triggers auto-light in auto mode; LCD updates to reflect current system state in real time. System handles simultaneous inputs without crashing.	Full system exceeds brief: manual and auto modes coexist with clean mode switching; LCD shows mode, device state, and activity log; system handles edge cases (PIR trigger during manual-off; IR command during auto-triggered state); energy-saving auto-off implemented with configurable timeout.
Systems Thinking & Architecture Design (20%)	No system architecture diagram produced; student cannot explain how the subsystems relate or why components were chosen; design appears to have evolved without planning.	Architecture diagram attempted but incomplete or inaccurate; student can describe individual components but not explain how signals flow between them or why the integration requires a relay rather than a direct connection.	Clear system architecture diagram produced before build begins; student explains signal flow from each input through the Arduino to each output; relay necessity justified (Arduino cannot switch mains loads directly); mode logic documented with state transition diagram.	Architecture diagram is publication quality; includes signal types (digital HIGH/LOW, I2C, PWM), voltage levels, and current flow; state machine covers all modes and transitions; design rationale document explains each component choice and what was rejected; architecture could be handed to another team and built from scratch.
Energy Efficiency Analysis (20%)	No energy analysis attempted; claim that the system 'saves energy' is asserted	Some awareness of energy savings but no quantitative analysis; student states that auto-off saves energy	Energy savings estimated quantitatively: identifies a specific household device (e.g.	Extended energy analysis: identifies multiple devices that could be automated; calculates system-

	without calculation or evidence.	without calculating how much or comparing to baseline.	60W lamp), calculates kWh saved per day if lights are off 2 extra hours due to auto-off, projects annual saving, converts to cost in local currency. Compares to energy cost of running the Arduino continuously.	level savings; compares to a commercial smart home system's claimed savings; calculates payback period (when does the cost of building the system equal the energy savings it generates?); addresses the rebound effect (do people use more energy when they think automation is managing it?).
Client-Centred Design & Trade-off Reasoning (15%)	No client perspective evident; system built to demonstrate technical capability without reference to household needs, usability, or cost constraints.	Mentions household client but design decisions not driven by client needs; features included because they are technically interesting rather than useful; no trade-offs documented.	Design decisions connected to specific household needs identified in the entry event audit; at least two features excluded with explicit justification ('we considered X but excluded it because our client needs Y'); client demonstration prepared with client's perspective, not the builder's.	Deep client-centred reasoning: household audit produced specific quantified needs (e.g. 'client leaves lights on average 3 hours per day when rooms are empty'); every feature inclusion and exclusion documented with client rationale; client demonstration shows the system from the client's perspective with no technical jargon; client feedback incorporated into at least one design revision.
Communication & Client Demonstration (20%)	Demonstration absent or unclear; technical vocabulary unexplained; no connection to household energy or convenience benefits.	Demonstration covers system functions but explained in technical terms unsuitable for a household client; benefits stated but not connected to the client's specific situation.	Clear client-facing demonstration: each function shown and explained in plain language; energy savings presented in household terms (money saved per year, not kWh); client can operate the system during the demonstration; LCD status explained accessibly.	Compelling client demonstration that tells the household story: opens with the problem (wasted standby energy, forgotten lights), shows the solution in action, quantifies the benefit in pounds/dollars per year, invites the client to operate the system, handles 'what if?' questions with evidence. Client leaves knowing exactly what the system does and why it saves them money.

System Documentation (20%)	Documentation absent or limited to a component list; no architecture diagram; no user instructions.	Documentation present but reads as technical notes rather than a client-facing document; architecture diagram incomplete; user instructions assume Arduino knowledge.	Documentation package includes: system architecture diagram (complete and accurate), user operation guide (written for a non-technical household member), energy savings calculation with assumptions stated, and a feature log (what was built, what was planned but excluded, why).	Documentation package is handover quality: architecture diagram could be used to rebuild the system; user guide tested with a non-technical reader and revised; energy savings report references real electricity tariff data; feature log includes retrospective — what would be done differently if restarting; documentation written at a register appropriate for a homeowner, not a student.

CRITERION	WEIGHTING	ASSESSED WHEN
Systems Integration & Technical Functionality	25%	Lessons 6–10 (test matrix) + Lesson 14 (client demo)
Systems Thinking & Architecture Design	20%	Lesson 3 (diagram) + Lesson 11 (updated diagram)
Energy Efficiency Analysis	20%	Lessons 10–12 (report) + Lesson 14 (presentation)
Client-Centred Design & Trade-off Reasoning	15%	Lesson 2 (MoSCoW) + Lesson 13 (demo prep) + Lesson 14
Communication & Client Demonstration	20%	Lesson 13 (practice) + Lesson 14 (summative)
System Documentation	20% (see note)	Lesson 15 final submission
TOTAL	**120% → normalised to 100%**	**Portfolio submitted Lesson 15**

Note: Weightings sum to 120%. Divide raw total by 1.2 to get a percentage score out of 100.

Student Handouts

Handout 1 — 'Need to Know' Starter Questions

WHAT DO I NEED TO KNOW TO ANSWER THE DRIVING QUESTION?

Project 4 is the first integration project. You need to know how each subsystem works AND how to make them work together. Both halves matter equally.

About the System Design Challenge:

- What is a system architecture diagram? What does it show that a circuit diagram does not?
- What is a state machine? How does it handle conflicting inputs from two different sources?
- What is incremental integration? Why do engineers build and test subsystems before combining them?
- What are the failure modes that appear only when multiple subsystems are integrated — not when they are tested individually?

About the New Components:

- What is a relay module? How does an electromagnetic coil switch a separate electrical circuit?
- What is normally-open (NO) vs normally-closed (NC) in a relay? When would you use each?
- Why can the Arduino not control a real household lamp directly — what is the current limit of an Arduino output pin?
- How does the relay module's IN pin work — does HIGH turn the relay ON or OFF? (check your specific module — some are active-low!)

About Energy and the Client:

- How much energy does a typical household lamp use in 1 hour? (look up the wattage of a standard bulb)
- What is the electricity tariff in Qatar (or your local context) per kWh?
- How many hours per day do typical households leave lights on in unoccupied rooms? (research this — do not guess)
- What is a payback period? How do you calculate it for a home improvement?
- Does the Arduino itself use electricity? How much? Is this more or less than the energy it saves?

About the Integration Challenge:

- What happens in Arduino code when an IR signal arrives at the same time as a PIR trigger? How does the loop() function handle simultaneous events?
- How do you write Arduino code that checks multiple inputs without any single check blocking the others?
- What is a non-blocking code pattern? How is it different from using delay()?

Add your own questions:

1. _____
2. _____
3. _____

Handout 2 — Household Audit & MoSCoW Feature Prioritisation

HOUSEHOLD AUDIT + FEATURE PRIORITISATION

Part 1: Household Audit

Identify your household scenario (your own home, or a defined fictional household). Answer:

- Where are lights most often left on when rooms are empty? _____
- Which devices are left in standby mode unnecessarily? _____
- Which household member would benefit most from automated lighting? Why? _____
- What times of day is automation most needed? _____
- Estimate: how many hours per day are lights on in unoccupied rooms? _____

Part 2: MoSCoW Feature Prioritisation

List all the features you could build. Then sort them into the four categories below. You have 3 weeks and a £20 materials budget — be ruthless.

CATEGORY	FEATURES IN THIS CATEGORY — AND WHY
MUST HAVE (core function — system fails without these)	
SHOULD HAVE (important but not critical)	
COULD HAVE (nice to have if time allows)	
WON'T HAVE THIS TIME (explicitly out of scope)	

Trade-off justification (required): Name one feature in 'Could have' or 'Won't have' that you originally wanted but chose to exclude. Explain the trade-off in one sentence using client reasoning, not technical reasoning.

_____

_____

Handout 3 — Integration Test Matrix

20-SCENARIO INTEGRATION TEST — complete this in Lesson 8

A system that passes 5 tests is not the same as a system that passes 20. Professional engineers test edge cases, not just the happy path. Run every scenario below and record the result honestly.

#	Test Scenario	Expected Result	Actual Result	Pass / Fail / Root Cause
1	IR button A pressed — device is OFF	Relay ON, LCD: Device ON		
2	IR button B pressed — device is ON	Relay OFF, LCD: Device OFF		
3	Mode toggle pressed — system in MANUAL	Mode: AUTO, LCD updates		
4	Mode toggle pressed — system in AUTO	Mode: MANUAL, LCD updates		
5	Motion detected — system in AUTO mode	Relay ON, LCD: Motion YES		
6	Motion stops — system in AUTO mode	Relay OFF after timeout, LCD: Motion NO		
7	Motion detected — system in MANUAL OFF mode	Relay stays OFF (manual overrides)		
8	IR ON command — system in AUTO, relay already ON	Relay stays ON, mode: MANUAL		
9	IR OFF command — system in AUTO, relay ON (motion active)	Relay OFF, mode: MANUAL		
10	Rapid IR button presses (5 in 2 seconds)	Last command wins, no crash		
11	PIR warmup period — motion in first 30 seconds	No false trigger during warmup		
12	Mode toggle during active PIR trigger	Mode changes, relay behaviour updates		

13	IR command during PIR-triggered relay ON (AUTO mode)	IR command takes precedence, mode switches to MANUAL		
14	Power cycle — system restarts	System boots to defined default state		
15	LCD readable from 2 metres in normal room lighting	All text visible without squinting		
16	System runs for 10 minutes with no input	No crash, no spurious relay triggers		
17	PIR trigger followed immediately by IR OFF	IR OFF wins, relay turns off		
18	Auto-off timer expires (Upgrade A only)	Relay off, LCD: Auto-off		
19	Temperature exceeds threshold (Upgrade C only)	Relay ON in AUTO mode, LCD shows temp		
20	All inputs active simultaneously (IR, PIR, timer)	System behaves according to priority rules without crashing		

Summary: Scenarios passed: ___ / 20 Critical failures (must fix before demo): ___ Accepted limitations (document in feature log): ___

Handout 4 — Energy Savings Calculation Guide

ENERGY SAVINGS REPORT — use real numbers, not estimates

Every figure in this report must be sourced. State where each number came from.

1. Baseline energy waste (without automation)

- Device controlled: _____ (e.g. 60W bedside lamp)

- Device power consumption: ___ W (source: _____)
- Hours left on unnecessarily per day (from household audit): ___ hours
- Daily wasted energy: ___ W × ___ hours = ___ Wh = ___ kWh
- Annual wasted energy: ___ kWh × 365 = ___ kWh/year
- Annual cost of waste: ___ kWh × ___ (local tariff per kWh) = ___ £/$/QAR per year

2. Energy saved by automation

- Estimated hours saved per day by auto-off: ___ hours (justify this estimate from your household audit)
- Daily energy saved: ___ W × ___ hours saved = ___ kWh
- Annual energy saved: ___ kWh × 365 = ___ kWh/year
- Annual cost saved: ___ kWh × ___ (local tariff) = ___ £/$/QAR per year

3. Energy cost of running the Arduino

- Arduino UNO power consumption: approximately 0.25W when active
- Hours per day the Arduino runs: ___ hours (does it run 24/7 or only when needed?)
- Annual Arduino energy cost: 0.25W × ___ hours × 365 = ___ kWh × tariff = ___ £/$/QAR per year
- Net saving (automation saving minus Arduino cost): ___ £/$/QAR per year

4. Payback period

- Total component cost of the system: £___ / $___
- Annual net saving (from section 3): £___ / $___
- Payback period: total cost ÷ annual saving = ___ years
- Interpretation: Is this a good investment? Compare to a commercial smart plug (£15–30) with the same function.

5. Honest assessment

- Under what conditions does this system save the most energy? _____
- Under what conditions might it save less than calculated? (e.g. always-on PIR warmup, false triggers keeping lights on) _____
- What is the single most important limitation of this energy analysis? _____

Handout 5 — Client Audience Feedback Form

CLIENT FEEDBACK FORM (for household client audience)

Team name / number: _____ Date: _____

1. Did the system do what a household needs?

Did the team understand your energy problem? Does their system address the actual waste patterns they identified?

Rating: 1 — Needs work 2 — Adequate 3 — Good 4 — Excellent

Comments: _____

2. Could you operate the system yourself?

During the demonstration, could you operate the system without technical guidance? What was confusing?

Rating: 1 — Needs work 2 — Adequate 3 — Good 4 — Excellent

Comments: _____

3. Is the LCD display useful?

Could you understand the system status from the LCD in a real room? What would you want it to show that it currently does not?

Rating: 1 — Needs work 2 — Adequate 3 — Good 4 — Excellent

Comments: _____

4. Is the energy saving claim credible?

Did the team calculate savings using real numbers? Does the claimed saving seem plausible for a real household?

Rating: 1 — Needs work 2 — Adequate 3 — Good 4 — Excellent

Comments: _____

5. Would you install this in your home?

Honestly — would you install this system? What would need to change to make you say yes?

Rating: 1 — Needs work 2 — Adequate 3 — Good 4 — Excellent

Comments: _____

Overall: Would you install this system? Yes / No / With modifications

If 'With modifications': what specifically? _____

Handout 6 — Individual Reflection Prompts

REFLECTION — integration is harder than it looks: prove you understand why

Complete individually in the final lesson. Reference specific integration challenges, test results, or client feedback from your project.

The hardest integration challenge:

Describe the most difficult integration failure you encountered — the one where two subsystems worked individually but not together. What was the root cause? How did you find it?

_____

_____

_____

Architecture diagram value:

Did your architecture diagram turn out to be accurate? What changed between the plan and the build? What does that teach you about planning complex systems?

_____

_____

_____

The 'sitting still' problem:

A client asks: 'What if I'm sitting reading quietly and the PIR doesn't detect me — will the light turn off on me?' How did your team design for this? Is your answer satisfying? What would you add with more time?

_____

_____

_____

Energy honesty:

After completing the energy savings calculation: did the numbers surprise you? Is your system a good energy investment compared to just buying a £15 smart plug? What does your honest assessment conclude?

_____

_____

_____

Client operation moment:

Describe the moment during the client demonstration when the client operated the system. What happened? What did their reaction tell you about the design that testing with your team could not have told you?

_____

_____

_____

Tier 2 synthesis:

Project 4 is the first integration project. Projects 5, 6, and 7 each take one aspect of this system further. Which subsystem from Project 4 do you most want to develop? What would you do differently in that development if you started from your experience today?

_____

_____

_____

Teacher Reference: Common Misconceptions & Facilitation Moves

Project 4 surfaces a new category of misconception that does not appear in single-component projects — misconceptions about integration itself. These are often more persistent than component-level errors because they are about how students think about systems, not just about individual circuits.

STUDENT MISCONCEPTION	CORRECT UNDERSTANDING	TEACHER FACILITATION MOVE
'I'll build everything at once and debug at the end'	Integrating all subsystems simultaneously before testing any of them individually makes debugging nearly impossible — when something fails (and it will), there is no way to isolate which subsystem is the cause. Incremental integration — build one subsystem, test it, then add the next — is the professional approach because it isolates failure modes.	*Before distributing any components: 'Describe your build sequence. What will you test first? How will you know Subsystem 1 is working before you add Subsystem 2?' Refuse to allow the next component until the team can demonstrate the previous subsystem working. This feels slow but saves an entire lesson of unsolvable debugging.*
'The relay module works — just write HIGH to the pin'	Some relay modules are active-HIGH (HIGH = relay ON), while others are active-LOW (HIGH = relay OFF). This depends on whether the onboard transistor inverts the logic. Students who assume active-HIGH and have an active-LOW module will spend an entire lesson debugging what appears to be a code problem but is actually a hardware characteristic.	*Ask: 'Before you write a single line of code, test your relay module manually: connect the IN pin to 5V and observe. Connect it to GND and observe. Which state closes the relay? Write this down before you code.' This empirical first step prevents the entire confusion.*
'My loop() runs fast enough to handle everything simultaneously'	Arduino loop() runs sequentially — it processes IR decoding, then PIR reading, then LCD update, one at a time. At 16MHz this is fast, but some library functions (particularly IRremote) block for significant periods. The perception that the Arduino handles simultaneous inputs is an approximation — understanding the sequential reality prevents subtle timing bugs.	*Ask: 'What happens if an IR signal arrives while your code is inside a 500ms delay() call? Does the signal get missed?' Students almost always say yes. Then: 'What does this tell you about where you can and cannot put delay() in an integration project?'*
'The system works in testing so it will work for the client'	Testing in familiar conditions with deliberate, careful inputs does not represent real household use. A household client will press buttons in unexpected sequences, walk past the sensor at unexpected speeds and angles, and try to do things the system was not designed for. The 20-scenario test matrix and the peer system review are specifically designed to surface failures that self-testing misses.	*Before the client demonstration: 'Your partner team just tested your system and found two scenarios where it behaves unexpectedly. What are they? Have you addressed them? If not — document them as known limitations in your feature log so you can answer honestly when the client discovers them.'*

'Energy savings = the amount of energy the device saves'	A complete energy analysis must account for the energy consumed by the automation system itself. An Arduino UNO running continuously consumes approximately 0.25W — or about 2.2 kWh per year. For a 10W LED that would otherwise be left on for 1 extra hour per day (3.65 kWh/year saving), the Arduino overhead is significant. Net savings, not gross savings, is the honest number.	*Ask: 'Does your Arduino use electricity? How much? Calculate the annual energy cost of running your Arduino. Now subtract that from your claimed saving. Is the system still a good investment?' Students who have not done this calculation often discover the net saving is surprisingly small — and that is a valid finding.*
'The feature was too hard so we cut it — that's a trade-off'	A trade-off is a deliberate design decision made to better serve the client — not an admission of a technical failure. 'We couldn't get the LCD to update in real time' is a failure. 'We chose not to implement activity logging because our client's primary need is reliability, not monitoring, and logging would add code complexity that could introduce instability' is a trade-off. The distinction matters for the rubric and for professional engineering practice.	*During MoSCoW: 'Read me your Won't Have list and explain each item as a trade-off — using client language, not technical language.' If a student says 'we couldn't get it to work' — ask: 'Is that a trade-off or a failure? What's the difference? How would you document each one honestly?'*

Differentiation Strategies

LEARNER PROFILE	SUPPORT SCAFFOLDS	EXTENSION CHALLENGES
Learners needing additional support	Pre-drawn architecture diagram template with component boxes and signal type labels provided · Relay module pre-wired to LED load · Mode logic pseudocode provided as a fill-in-the-blank · 10-scenario test matrix (not 20) · Energy calculation template with formula pre-written · Client demo script with sentence starters	Two subsystems only (IR + relay; PIR optional) · MANUAL mode only (no AUTO mode switching) · Basic LCD: device ON/OFF status only · 10-scenario test matrix · Energy calculation for one device · Client demo with 3-minute presentation
On-track learners	Standard project brief · BIE phase check-ins · Architecture diagram required before build · 20-scenario test matrix · All 6 rubric criteria targeted	All three subsystems (IR + PIR + relay + LCD) · MANUAL and AUTO modes with clean switching · One upgrade · Energy savings report with payback period · 5-minute client demonstration with client operation of the system
Advanced learners	Minimal scaffolding · Self-directed architecture design · Research relay specifications from actual datasheet · Design own test matrix beyond the provided 20 scenarios	All three upgrades (auto-off + activity log + temperature mode) · Extended energy analysis with rebound effect consideration · Full documentation package with retrospective · Peer-teach relay module concept to on-track team · Client demonstration with energy savings

		report presented as a household investment case

Materials List & Approximate Costs

COMPONENT	QTY (per team)	UNIT COST (USD approx.)	NOTES
Arduino UNO	1	$4–8	UNO recommended for Project 4 — larger board makes multi-subsystem wiring easier to inspect and debug
IR receiver module (TSOP38238)	1	$0.50–1.50	Reuse from Project 1 if available; same module
IR remote control (NEC protocol)	1	$1–3	Reuse from Project 1; ensure students know which hex codes they are using
5V relay module (1-channel)	1	$0.80–1.50	Pre-built module strongly recommended — safer, includes flyback diode; check whether active-HIGH or active-LOW before class
HC-SR501 PIR motion sensor	1	$1–3	Reuse from Project 3 if available; allow 60-second warmup in all sessions
16×2 LCD with I2C backpack	1	$2–4	Reuse from Project 2 if available; note I2C address (0x27 or 0x3F)
Red LED (5mm)	1	$0.10–0.20	For relay status indicator
Green LED (5mm)	1	$0.10–0.20	For safe/off status indicator
220Ω resistors	2	$0.05 each	For LEDs
DHT22 sensor (Upgrade C only)	1	$3–5	Optional; for temperature-triggered mode upgrade
Breadboard (full-size recommended)	1	$2–4	Full-size breadboard strongly recommended for Project 4 — more components require more space
Jumper wires (large assorted pack)	30+	$1.00 pack	More wires needed than previous projects; colour-coding is essential — use consistent colour conventions
TOTAL per team (core kit)	—	≈ $13–22	If reusing components from Projects 1–3, core kit cost

			drops to approximately $6–10 for new components only