MindMetrix LABS

An Education Initiative of
MindMettrix Techno Private Limited

THINK BUILD INNOVATE

FUTURE SKILLS EDUCATION SERIES

ROBOTICS & IoT STUDENT WORKBOOK

GRADE 8

CBSE CSL ALIGNED

Aligned with NEP 2020 & NCF-SE 2023

SMARTSPACE

Build a Connected & Intelligent Space



ROBOTICS
Build intelligent machines

IoT & SMART SYSTEMS
Sense, connect and control

CODING
Program and automate

ENGINEERING DESIGN
Design, build and test

INNOVATION
Create solutions for tomorrow

Student Name	: _____	**School**	: _____
Class	: _____ **Section** : _____		_____
Roll No.	: _____		_____

THINK • BUILD • CONNECT • INNOVATE

www.mindmetrixlabs.com

FRONT MATTER

Copyright & Publication Information

Title	Robotics & AI Student Workbook
Grade	8
Course	Electronics & Embedded Systems Fundamentals
Edition	First Edition
Published by	MindMetrix Techno Private Limited
Website	www.mindmetrixlabs.com
Email	info@mindmetrixlabs.com

© Copyright Notice

© 2026 MindMetrix Techno Private Limited. All rights reserved. No part of this publication may be reproduced, stored, or transmitted in any form without prior written permission from the publisher, except for classroom use by licensed institutions.

Disclaimer

This workbook is designed for educational purposes. Activities should be performed under the guidance of a trained teacher using appropriate safety practices.

FRONT MATTER

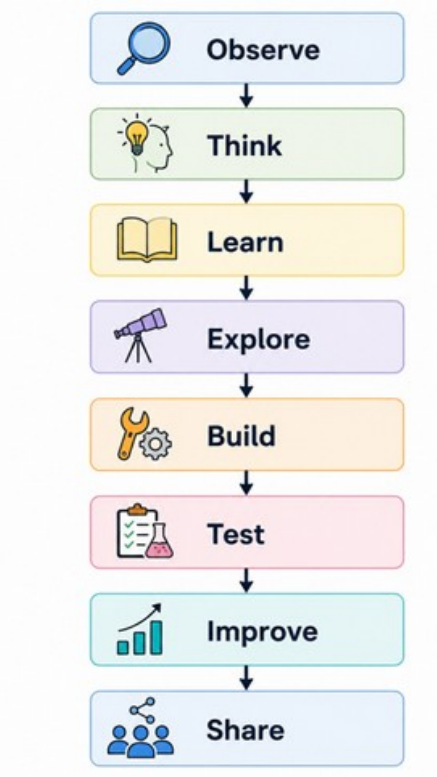
About MindMetrix Labs

Building Future Innovators Through Robotics, AI & Emerging Technologies

MindMetrix Labs is an education initiative of MindMetrix Techno Private Limited, dedicated to developing future-ready skills through experiential, hands-on learning in Robotics, Artificial Intelligence, Electronics, Embedded Systems, Coding, IoT, and other emerging technologies.

THINK • BUILD • INNOVATE

Our Learning Approach

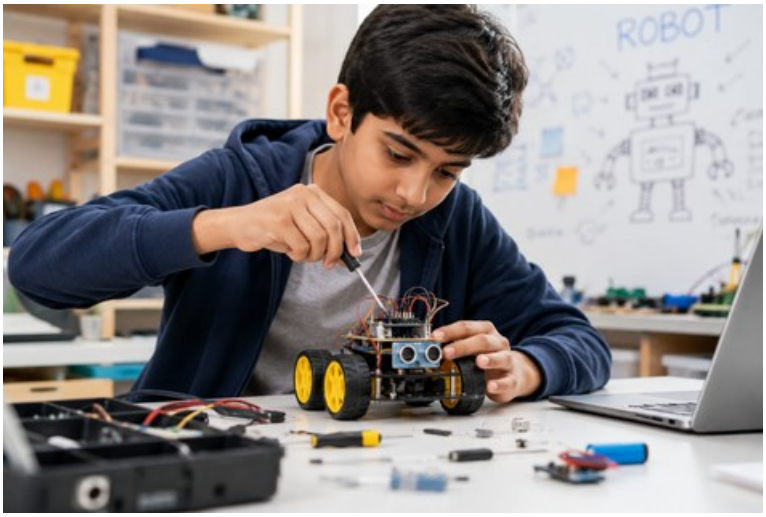


FRONT MATTER

Welcome, Future Innovator!

Your Robotics & AI Journey Starts Here!

You are about to explore the exciting world of **robots, electronics, coding, and intelligent systems.**



In this workbook, you won't just read about technology — you will:

👀 **OBSERVE** Look closely. Notice how things work.	💭 **THINK** Ask questions. Make predictions.	🔧 **BUILD** Connect, assemble, and create.
🧪 **TEST** Try your ideas and see what happens.	🚀 **IMPROVE** Find what can work better.	💡 **INNOVATE** Create your own solutions.

> 🎯 **Your Mission**
>
> **Be curious. Try things. Make mistakes. Learn from them. Try again.**
>
> There is not always one right answer.
>
> **Your ideas matter.**

THINK ● BUILD ● INNOVATE

Let's begin!

FRONT MATTER

How to Use This Workbook

Learn by Doing

This workbook is your guide during every Robotics & AI lab session. You will **explore, build, test, and improve** instead of simply reading.

For every session:

1	**Read the Mission** Know what you are going to make or discover.
2	**Check What You Need** Collect the components and tools.
3	**Follow the Activity** Build, connect, program or experiment.
4	**Record What You Discover** Write, draw or tick your observations.
5	**Try the Challenge** Change something or find another solution.
8	**Complete Your Self-Check** Think about what you learned.

Your Workbook Has These Sections

🎯	**Mission**	What are you going to achieve?
🧰	**What You Need**	Components and tools required.
🔧	**Activity**	What you will do.
👀	**Observe**	What did you notice?
💡	**Think**	Why did it happen?
🚀	**Challenge**	Can you go further?
✅	**Self-Check**	How did you do?

One Important Rule

Don't just follow the instructions. Be curious.

If something doesn't work, **don't immediately ask for the answer.** Check your connections, think about what happened, try again, and learn from the result.

Your ideas, observations, and experiments are all part of the learning.

✅ Ready to Build?

Before you start your first module, make sure you:

- ☐ Have read your Mission for the session
- ☐ Have checked what you need and collected it
- ☐ Know and understand the Lab Safety rules
- ☐ Are ready to observe, try, and try again

ORIENTATION · LAB SAFETY

Lab Safety

WORK SAFELY

Before You Start

- 🥽 Wear safety glasses when instructed.
- 🔌 Check that power is OFF before changing connections.
- 🧰 Use tools only as instructed.
- 🧹 Keep your workspace clean and organized.

While You Build

- 🔋 Check polarity before connecting power.
- 🔗 Check wires and connections before switching ON.
- ✋ Never force a component into the circuit.
- 💧 Keep electronics away from water.
- 🙋 Ask your teacher if you are unsure.

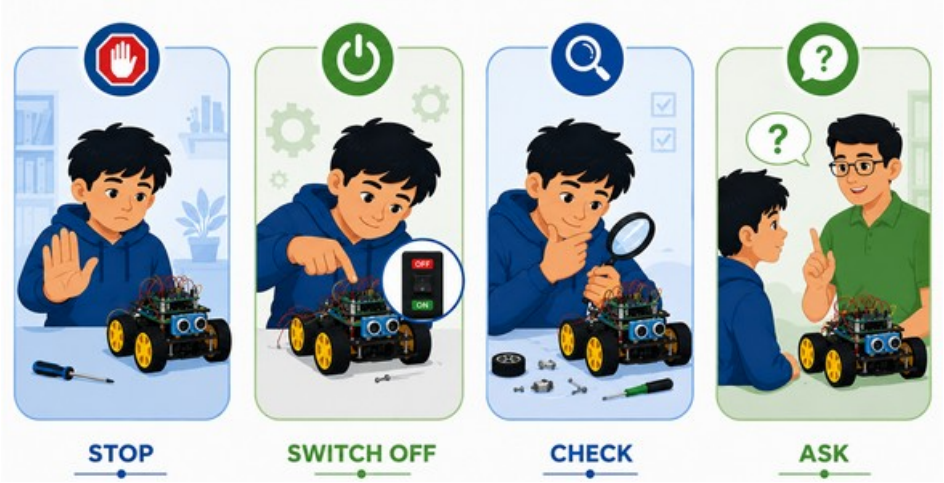
When You Finish

- 🔌 Switch OFF and disconnect power.
- 📦 Return components to their correct places.
- 🧹 Leave your workspace ready for the next team.

🚨 **Remember**

STOP → SWITCH OFF → CHECK → ASK

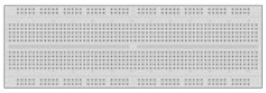
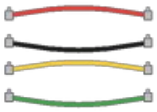
Never experiment with a powered circuit if you are unsure about the connection.



Meet Your Robotics Lab

Used in Class 6 — familiar robotics parts

One SMARTSPACE team • Find each part, name it and explain what it did in Class 6.

Picture	Name and quantity	Use
	Arduino UNO R3 — 1	Our first controller. This year we will use ESP32 instead.
	Breadboard — 1	Holds the start-button circuit; not motor power.
	Momentary push button — 1	Starts the mission. Use the four-leg button in the activity; the pictured module is an identification example.
	DC BO motors — 2	Geared DC motor; later used as a small model fan with an approved driver.
	Jumper wires — 1 set	Male–male and male–female leads carry signals.
	USB A-to-B data cable — 1	Uploads code from the computer to UNO.

Pictures identify parts; follow the activity wiring diagrams and the labels on your actual module. Teacher checks all power connections before use.

🤔 Think and discuss

Which two familiar components could become part of SMARTSPACE?

My idea: _____

⚠️ **SAFETY CHECK • These pictures are for identification, not wiring. Power must stay OFF during kit inspection.**

Meet Your Robotics Lab

Used in Class 6 — power and connections

One SMARTSPACE team • Find each item, say its name and explain its use.

Picture	Name and quantity	Use
	4-AA holder, switch and 4 AA cells	Class 6 motor supply. Do not connect these AA cells directly to the ESP32.
	Prepared motor leads and insulated connectors	Helped with safe motor wiring and insulated power junctions; teacher prepares them.
	Clips, cable ties and insulating pads	Secure boards and keep wiring away from moving parts.

Pictures identify parts; follow the activity wiring diagrams and the labels on your actual module. Teacher checks all power connections before use.

🤔 Think and discuss

Why can we not assume the Class 6 battery holder powers the ESP32 safely?

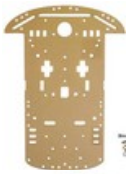
My idea: _____

⚠️ **SAFETY CHECK • ESP32 GPIO uses 3.3 V logic; never connect a 5 V output to a GPIO pin.**

Meet Your Robotics Lab

Used in Class 7 — Autonomous Mission Rover

Remember our rover • Identify each part and recall how it helped movement.

Picture	Name and quantity	Use
	ProBots clear acrylic 2WD chassis — 1	Body of our Class 7 moving rover; not needed for the fixed SMARTSPACE model.
	Drive wheels — 2	Transferred motion from the BO motors to the rover.
	Caster wheel — 1	Supported the moving rover on the route.
	Brackets, screws, nuts and spacers — 1 set	Secured parts and supported the robot body.

Pictures identify parts; follow the activity wiring diagrams and the labels on your actual module. Teacher checks all power connections before use.

🤔 Think and discuss

How is a fixed SMARTSPACE model different from a moving rover?

My idea: _____

⚠️ **SAFETY CHECK • Do not use rover batteries or driver connections until the Grade 8 diagram tells you to.**

Meet Your Robotics Lab

Used in Class 7 — motors and sensing

Find the familiar modules • Some can be reused for SMARTSPACE with new wiring.

Picture	Name and quantity	Use
	TB6612FNG motor driver — 1	Controlled the rover motors. Reuse only if approved for the SMARTSPACE fan circuit.
	Digital IR reflectance modules — 2	Sensed the rover track. Helps us recall input signals and careful testing.
	HC-SR04 — 1	Measures distance; may be reused for a SmartSpace doorway after voltage-level checking.
	Regulated 5V USB power bank — 1	Previously supplied Arduino USB power. ESP32 power must be checked separately.

Pictures identify parts; follow the activity wiring diagrams and the labels on your actual module. Teacher checks all power connections before use.

🤔 Think and discuss

Which Class 7 sensor could help our SMARTSPACE notice someone near a door?

My idea: _____

⚠️ **SAFETY CHECK • HC-SR04 echo may be 5 V. Use a teacher-checked level shifter/divider before ESP32 GPIO.**

Meet Your Robotics Lab

New in Class 8 — controller and sensing

One SMARTSPACE team • Find each part in your kit. Say its name, check its label and explain its use.

Picture	Name and quantity	Use in SMARTSPACE
	**ESP32-WROOM-32 development board** 1	The connected brain: runs our program and provides built-in Wi-Fi and Bluetooth.
	**DHT11 temperature & humidity module** 1	Reads room temperature and humidity for the status display.
	**HC-SR501 PIR motion sensor** 1	Notices movement in the model room; it does not identify people.
	**LM393 LDR light sensor module** 1	Detects light changes to help control the model room light.

Identify the real module by its printing and shape. These are reference pictures, not wiring diagrams.

🤔 Think and discuss

Which two sensors would help SMARTSPACE save energy? Explain your choice.

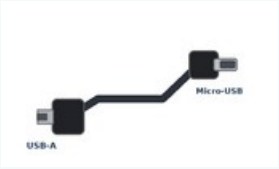
My idea: _____

⚠️ **SAFETY CHECK • Keep power OFF. ESP32 GPIO is 3.3 V logic; never connect an unverified 5 V signal to a GPIO.**

Meet Your Robotics Lab

New in Class 8 — display and connections

Match each item to its picture before attempting any wiring.

Picture	Name and quantity	Use in SMARTSPACE
	**16×2 LCD with I2C interface** 1	Displays messages such as LIGHT ON, DOOR CLOSED and room readings.
	**4-channel 3.3 V ↔ 5 V logic level converter** 1	Interfaces suitable signals between different logic voltage levels; your teacher checks its correct use.
	**ESP32 USB data cable** 1	Connects ESP32 to the computer to upload code. Use Micro-USB or USB-C as required by your board.
	**Regulated 5 V / 2 A supply** 1 if needed	Provides approved external power where required. Reuse a suitable existing supply after teacher checking.

Check the USB socket and I2C board markings on your actual kit: the connector style and pin labels can differ.

🤔 Think and discuss

Which item transfers a program? Which one displays information? Which one handles voltage compatibility?

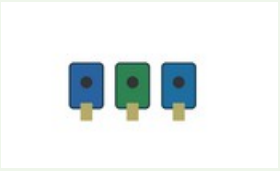
My idea: _____

⚠️ **SAFETY CHECK • Do not power motors/servos from an ESP32 GPIO. Teacher checks the 5 V source, logic levels and common ground.**

Meet Your Robotics Lab

SmartSpace building materials and familiar parts

We are making ONE stationary model. Reuse Class 6–7 electronics wherever they are suitable.

Picture	Name and quantity	Use in SMARTSPACE
	**Foam board / cardboard structure** 1 set	Makes rooms, walls and a light door for the SMARTSPACE prototype.
	**Insulated power connectors / terminal block** 1 set	Makes approved power junctions safer. Teacher prepares and checks these connections.
	**SG90 servo motor** 1, reused from Class 6	Moves the miniature door through a controlled angle; NOT a new purchase.

ALSO REUSE • Breadboard • jumper wires • LEDs + resistors • buzzer • push button • HC-SR04 (if used) • motor + driver (if approved).

Optional model fan: use a guarded low-voltage fan or a teacher-approved model output, rather than exposed rotating blades.

🤔 Think and discuss

Which part helps us BUILD the room? Which moves the door? Which Class 7 sensor could notice someone approaching?

My idea: _____

⚠️ **SAFETY CHECK • No mains wiring or soldering. Keep liquids away. Disconnect all power before moving, mounting or reconnecting parts.**

SMARTSPACE

Build a Connected & Intelligent Space

SENSE • CONNECT • CONTROL • AUTOMATE

Imagine our finished project



Project illustration • A classroom prototype, not a wiring guide. Actual components and placement may differ.

About our project

Imagine a room that can NOTICE what is happening and RESPOND. Your team will build a miniature classroom using an ESP32 as its controller. Sensors provide information; your program decides what to do; lights, a small door, a display and other low-voltage outputs respond.

ONE PROJECT — NOT MANY EXPERIMENTS

Start with one working light. Add a new feature in each mission, until the same SMARTSPACE can sense, connect and respond.

Your SMARTSPACE build journey



🤔 What should SMARTSPACE do when a classroom becomes dark?

_____

⚠️ **SAFETY CHECK**

Today we are only exploring the finished model. Keep the kit unpowered; the activity pages will show checked connections.

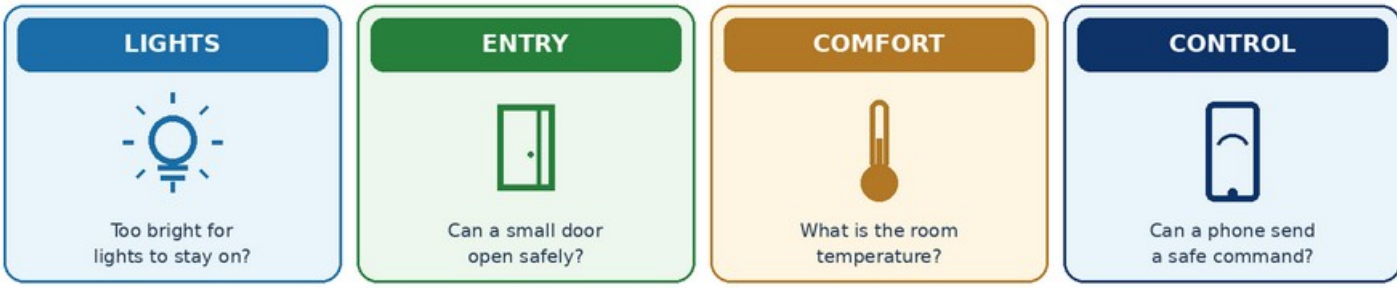
Why build SMARTSPACE?

YOUR BIG QUESTION

Can a connected classroom use information to become more useful, comfortable and energy-conscious?

Spot the problem around you

Study the four situations. For each one, discuss what your model could SENSE and what it could DO.



Turn observations into engineering ideas

EVERYDAY SITUATION	WHAT MIGHT WE SENSE?	POSSIBLE SMARTSPACE RESPONSE
Sunlight fills the room	Light sensor reading	Switch model LED OFF if enough light
Someone approaches a door	Presence or distance information	Test a safe servo-door response
The room feels uncomfortable	Temperature reading	Show temperature on the LCD
A person wants local control	Command from phone/laptop	Change an output after confirmation

Think like a designer

Every automated decision needs a human-designed rule. Our first idea may not work perfectly — that is why engineers test and improve.

🤔 Which feature could reduce unnecessary energy use? How would you test it?

_____

🤔 Why might we still need a manual switch or override?

_____

Our team's first idea • Choose ONE problem

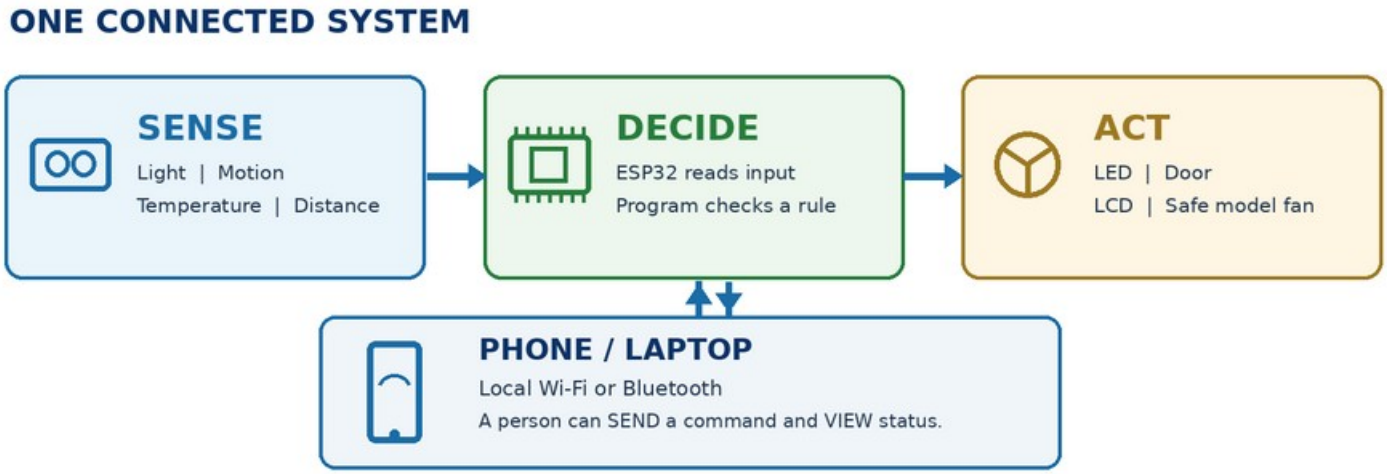
Draw or write here

⚠️ **SAFETY CHECK**

This is a classroom-scale, low-voltage model. It is not a real building electrical installation. Never connect to mains electricity.

Picture our SMARTSPACE mission

Your model will carry out three jobs: notice a change, apply your programmed rule and respond. Later, you will control some features from a phone or laptop.



FUNCTIONAL DIAGRAM • This explains the idea, not the wiring. Use only the later checked pin and power diagrams for assembly.

Plan where the parts will go

MODEL AREA	WHAT COULD WE NOTICE?	WHAT MIGHT WE CONTROL?
Near the window	Light level	Room LED
Near the entrance	Movement or distance	Lightweight model door
Inside the model room	Temperature / humidity	LCD status, model fan

Your turn • Sketch the first layout

Draw or write here

Label the ESP32, one sensor, one output, the door and a place to route wires safely.

🤔 Which part will be the “brain”? Which parts give it information?

_____

⚠️ **SAFETY CHECK**

Power OFF while planning. ESP32 pins use 3.3 V logic: never guess a connection or attach 5 V outputs directly to GPIO. Teacher checks every powered build.

Teacher verification: _____ **Date:** _____ **Team:** _____

Activity 1 • Build the Brain

Our first mission: make one light obey our ESP32

> 🎯 **YOUR MISSION**
> Can your team safely connect an LED, upload a short program, and make the SMARTSPACE light turn ON and OFF? You will add this same light to the final model.

Today you will use



Recognise each part in your kit before touching a wire. 1 ESP32 • 1 breadboard • 1 LED • 1 × 330 Ω resistor • 2 jumper leads • 1 USB data cable.

What success looks like



🤔 What must change inside the controller to make the LED blink?

_____

Three 45-minute sessions

SESSION 1 • EXPLORE	SESSION 2 • BUILD	SESSION 3 • TEST
Find parts and read labels Predict the light pattern Sketch an unpowered circuit	Connect LED + resistor Teacher checks power Upload first ESP32 code	Observe ON / OFF Change one value Debug and answer five questions

> ⚠️ **SAFETY CHECK**
> No wire changes while USB is connected. The teacher checks the exact board, USB cable and every connection before the first upload.

ACTIVITY 1 • LOOK CLOSELY

Meet the brain and the light

1 Find these labels on your ESP32

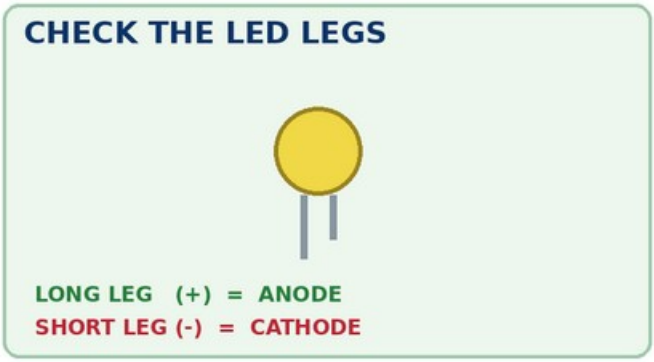
PART / LABEL	WHY IT MATTERS
ESP32-WROOM-32 board	The main controller that runs our instructions.
USB connector	Connects the board to a computer using a data cable.
GPIO23 (or 23)	The output pin we will use for our first light.
GND	The ground / return connection for the circuit.

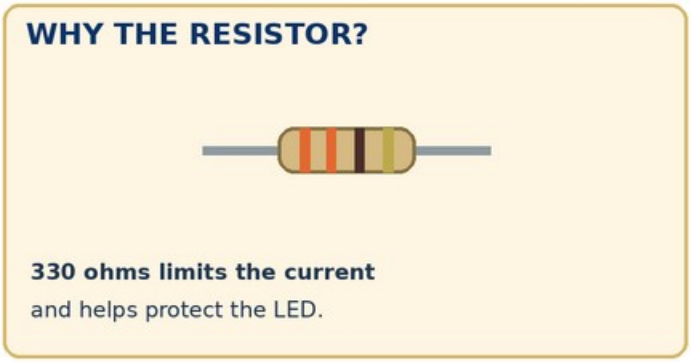
Board models vary. Find the actual printed “23” and “GND” labels on your teacher-approved 30-pin ESP32 — do not rely on pin position in a photograph.



ESP32 example photo for part recognition only — not a pinout.

2 Meet the LED and resistor





A SMALL DETAIL MATTERS

The LED has a positive side (anode) and a negative side (cathode). A 330 Ω resistor in series limits current. If the LED legs are trimmed, ask your teacher to identify the cathode from the flat edge or markings.

Predict: if we turn the LED around, will it light in the same circuit?

_____

SAFETY CHECK

ESP32 GPIO operates at 3.3 V logic. NEVER connect this LED directly to an ESP32 pin without the resistor. Do not use the 5V/VIN pin as a substitute for GPIO23.

Build it: one wire at a time

Our first circuit is very small. The best engineers build carefully, stop, check and then test.

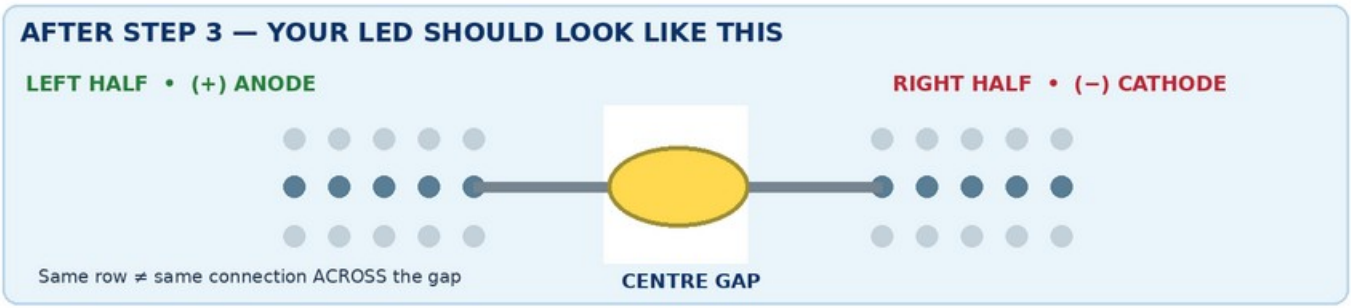


1 **POWER OFF — Prepare your work area**
Disconnect the ESP32 USB cable. Keep the board on a dry, non-metallic surface beside your breadboard.

2 **FIND THE BREADBOARD CENTRE GAP**
Locate the middle channel. Holes in a row on the left are connected to each other, but NOT across the centre gap.

3 **PLACE THE LED — Check the legs**
With power OFF, put its longer (+) leg in row 12 on the LEFT of the gap and shorter (−) leg in row 12 on the RIGHT.

🔎 **LOOK CLOSELY**
Two LED legs must not be pushed into holes that are already connected together. In our drawing, they sit on opposite sides of the centre gap.



🤔 Why must the two LED legs be on different, unconnected groups of holes?

_____

⚠️ **SAFETY CHECK**
Never force components into holes. Do not connect the ESP32 to USB yet. Ask for help if the LED legs are unclear or damaged.

Finish the circuit — then check

4 **ADD THE 330 Ω RESISTOR**
Place one resistor leg in row 10 LEFT and the other leg in row 12 LEFT. The resistor is now in series with the LED.

5 **CONNECT GPIO23**
Use a jumper lead from the ESP32 pin marked "23" / "GPIO23" to row 10 LEFT of the breadboard.

6 **CONNECT GND**
Use another jumper from an ESP32 pin marked "GND" to row 12 RIGHT, where the LED shorter (−) leg sits.

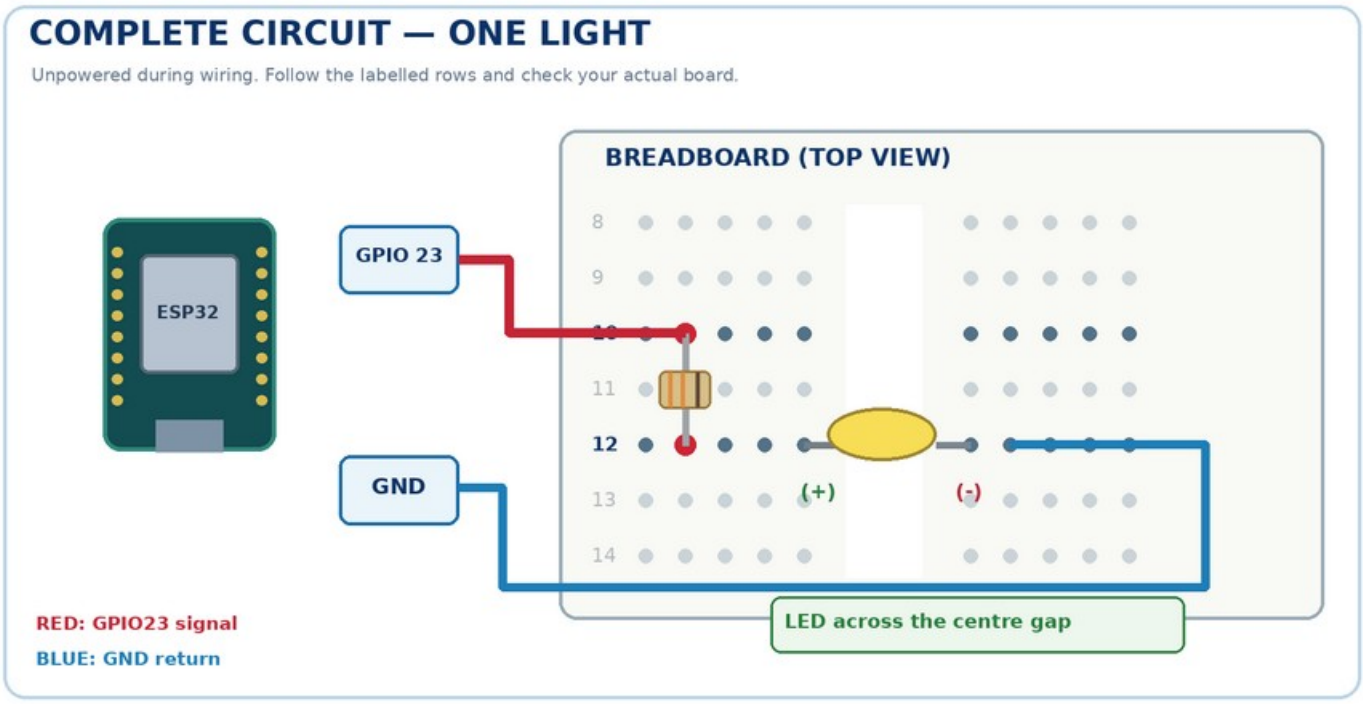


Illustration uses row 10 and row 12 as an example. Rows on your breadboard may have different numbers; maintain the SAME electrical connections.

FROM	TO	CHECK
ESP32 GPIO23	Breadboard row 10 LEFT	Signal wire
Row 10 LEFT	330 Ω resistor → row 12 LEFT	Series resistor
Row 12 LEFT	LED longer (+) leg	Anode
LED shorter (−) leg	Row 12 RIGHT → ESP32 GND	Return wire

✅ **TEACHER POWER CHECK**
☐ USB disconnected ☐ GPIO23 identified ☐ LED polarity correct ☐ 330 Ω resistor in series ☐ GND connected ☐ No bare wires touch

⚠️ **SAFETY CHECK**
Do not connect USB until the teacher verifies the circuit. If anything becomes hot, smells unusual, or behaves unexpectedly, disconnect power immediately and tell the teacher.

ACTIVITY 1 • MAKE IT THINK

Tell the brain what to do

Your ESP32 needs instructions. We will ask it to turn the light ON for one second, and OFF for one second, again and again.



Follow these Arduino IDE steps

1 TEACHER SETUP (ONCE)
Open Arduino IDE. Install “esp32 by Espressif Systems” through Boards Manager if it is not already installed.

2 CHOOSE BOARD AND PORT
Select the correct ESP32 model (commonly “ESP32 Dev Module” for this board) and the USB serial port that appears when connected.

3 OPEN A NEW SKETCH
Read the starter code below. Guess which line controls ON/OFF time before uploading.

Read → Predict → Upload

CODE POINTER | ANNEXURE A1 • Open the complete Activity 1 Arduino sketch at the back of this book. Copy it as ONE sketch, not separate pieces. Teacher checks board, libraries, pin labels and power before uploading.

Upload your program

With teacher permission, connect the USB data cable. Choose Verify / Upload and observe the LED. If uploading fails, do not unplug or rewire until the teacher checks the selected port and connection.

🧠 **CODE HINT**
HIGH = output ON • LOW = output OFF • delay(1000) = wait 1,000 milliseconds = 1 second.

⚠️ **SAFETY CHECK**
Teachers prepare and approve software installs/downloads. Students use only classroom-approved devices and do not share passwords or personal data.

Test it like an engineer



First test • Observe what really happens

1 PREDICT
Before upload, point to the code that says HIGH and LOW. Predict how many seconds the LED remains ON.

2 RUN
Ask the teacher to confirm the circuit; then upload the code. Watch at least five complete blink cycles.

3 CHANGE ONE THING
In the code, change BOTH delay(1000) values to delay(250); upload again, and compare the light pattern.

Engineer's observation table

TEST	OUR PREDICTION	WHAT WE OBSERVED
1000 ms ON / 1000 ms OFF	_____	_____
250 ms ON / 250 ms OFF	_____	_____
Difference I noticed	_____	_____

🐞 Debug Detective — if something goes wrong

SYMPTOM	FIRST SAFE CHECK
The LED never lights	Power OFF; inspect polarity, LED legs, resistor and GND connection.
Upload does not begin	Ask teacher to check USB data cable, selected board and serial port.
LED lights but does not blink	Check that the entire code uploaded and the loop has both HIGH and LOW.

🔧 **IMPROVE YOUR BUILD**
Try 500 ms ON and 1000 ms OFF. Sketch or describe the new pattern BEFORE uploading. Change the time values only — leave wiring unchanged.

⚠️ **SAFETY CHECK**
Switch off or disconnect USB BEFORE touching any connection. Never adjust wiring while the board is powered.

ACTIVITY 1 • THINK, REFLECT AND VERIFY

What did our ESP32 learn?

A working light is only the beginning. Your observations show how a programmed output can control a physical device.

🤔 Five thinking questions

1. Why do we use a 330 Ω resistor with the LED instead of connecting it directly?

_____

_____

2. What might happen if the LED is placed the wrong way round?

_____

_____

3. When delay changes from 1000 to 250, what changes in the blink pattern? What evidence did you see?

_____

_____

4. The LED stays OFF. Which TWO checks would you make first, after disconnecting power?

_____

_____

5. Where could SMARTSPACE use this light, and what rule might control it in a real classroom?

_____

_____

📓 My engineer's log

OUR BUILD	OUR EVIDENCE	OUR IMPROVEMENT
I connected an LED to: _____	It blinked: ☐ Yes ☐ No	We changed: _____
One problem we solved: _____	We observed: _____	Next I would try: _____

4CG learning evidence • Tick what your team demonstrated

☐ CG-1 We understood the mission. ☐ CG-2 We handled tools/components safely.

☐ CG-3 We tested, observed and improved. ☐ CG-4 We explained and shared our work.

🏆 **ACTIVITY COMPLETED**

Our ESP32 can control its first physical output! In the next mission, we will learn how to send a command from another device — without connecting a control wire.

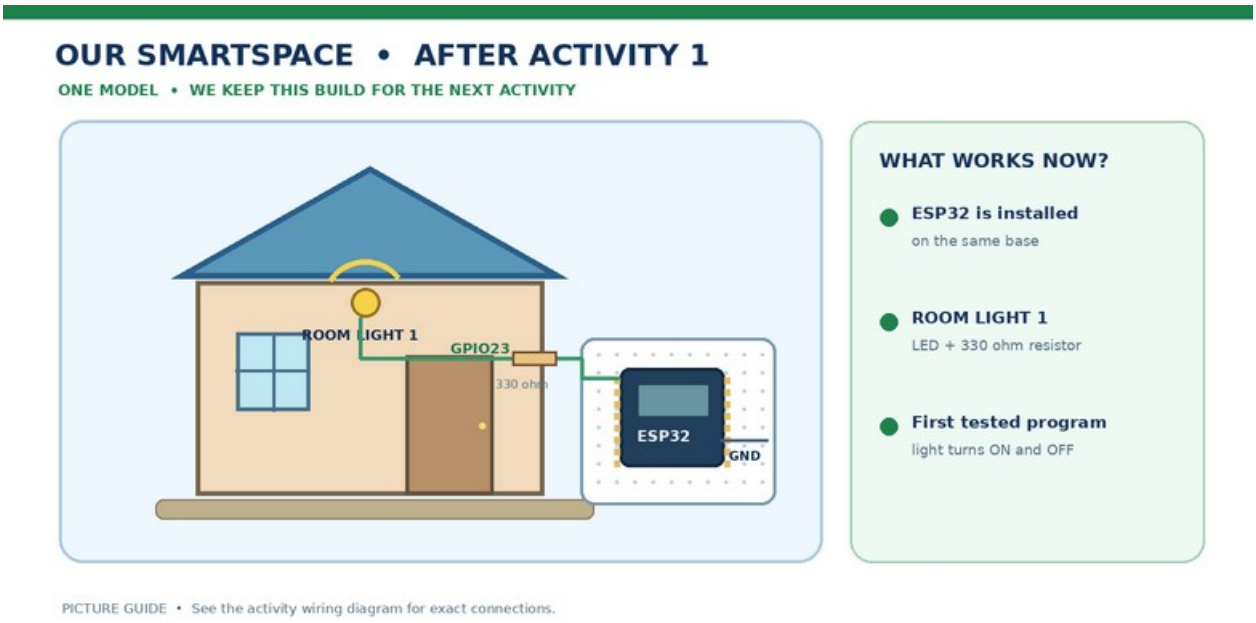
⚠️ **SAFETY CHECK**

Finish safely: disconnect USB, return parts to the kit and store the LED, resistor and breadboard safely. Do not leave the circuit powered unattended.

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 1 FINISHED • LOOK AT OUR COMPLETED MODEL

One SMARTSPACE ● Keep everything assembled for the next mission



CHECK OUR MODEL ☐ Same ESP32 ☐ Same ROOM LIGHT 1 ☐ Teacher checked our demonstration

NEXT: We will control this same light using Bluetooth, without adding another room or ESP32.

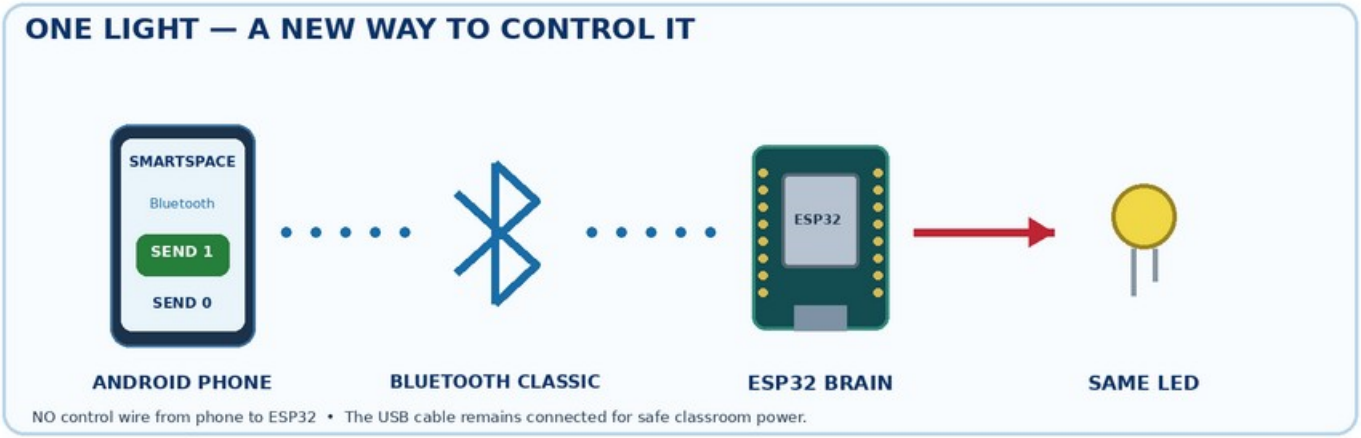
Activity 2 • Control Without Wires

Your mission: make the SAME SMARTSPACE light respond to a phone

🎯 THE CHALLENGE

In Activity 1 you made an LED blink using a timed program. Now keep the SAME ESP32, LED and breadboard, but replace the timer with two wireless commands: send “1” to turn the light ON and “0” to turn it OFF.

Picture our new mission



What you need • Reuse, do not rebuild

ESP32-WROOM-32 (Bluetooth Classic capable) • the LED + 330 Ω resistor and breadboard from Activity 1 • USB data cable • school computer with Arduino IDE • one teacher-approved Android phone/tablet with a Bluetooth Classic serial terminal app.

Watch your SAME project grow

ALREADY BUILT • ACTIVITY 1	NEW ABILITY • ACTIVITY 2	COMING NEXT • ACTIVITY 3
ESP32 + resistor + LED. The light followed a timer.	SAME light. SAME wiring. Phone sends ON / OFF commands.	SAME ESP32 + light. Add Wi-Fi connectivity later.

Three 45-minute sessions

SESSION 1 • PLAN	SESSION 2 • CONNECT	SESSION 3 • TEST
Recall the LED circuit Read commands 1 and 0 Predict the response	Upload Bluetooth code Pair the school Android Check the connection	Send 1 / 0 repeatedly Record observations Debug, reflect and explain

🤔 Will the phone need a physical wire to turn ON the LED? Explain your prediction.

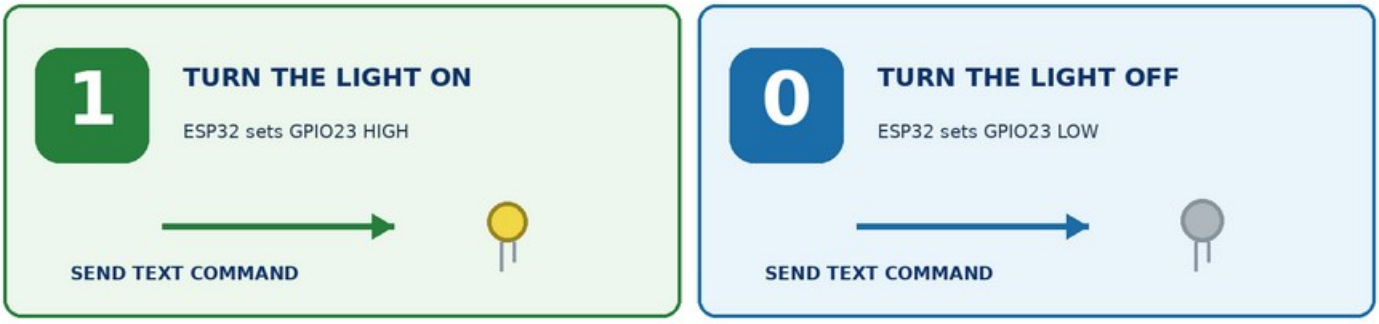
_____

⚠️ SAFETY CHECK

Power OFF before touching wires. Keep the ESP32 powered only by the teacher-approved USB supply while testing. No motor or servo is connected in this activity.

How can a phone talk to a light?

Follow the information, not a wire



1 **THE PHONE SENDS A COMMAND**
An approved Android device sends one text character through Bluetooth Classic: “1” or “0”.

2 **THE ESP32 READS THE CHARACTER**
Our program checks which character has arrived. The ESP32 does not guess or understand a spoken instruction.

3 **THE SAME LED RESPONDS**
For “1”, ESP32 sets GPIO23 HIGH. For “0”, it sets GPIO23 LOW. The 330 Ω resistor stays in the circuit.

🧠 **JUST ENOUGH TO KNOW**
Bluetooth connects nearby devices by radio. Our ESP32-WROOM-32 uses Bluetooth Classic Serial Port Profile (SPP) for this demonstration. This works with a compatible Android serial-terminal app; iPhones and BLE-only terminal apps do not use this same SPP method.

Think like a designer

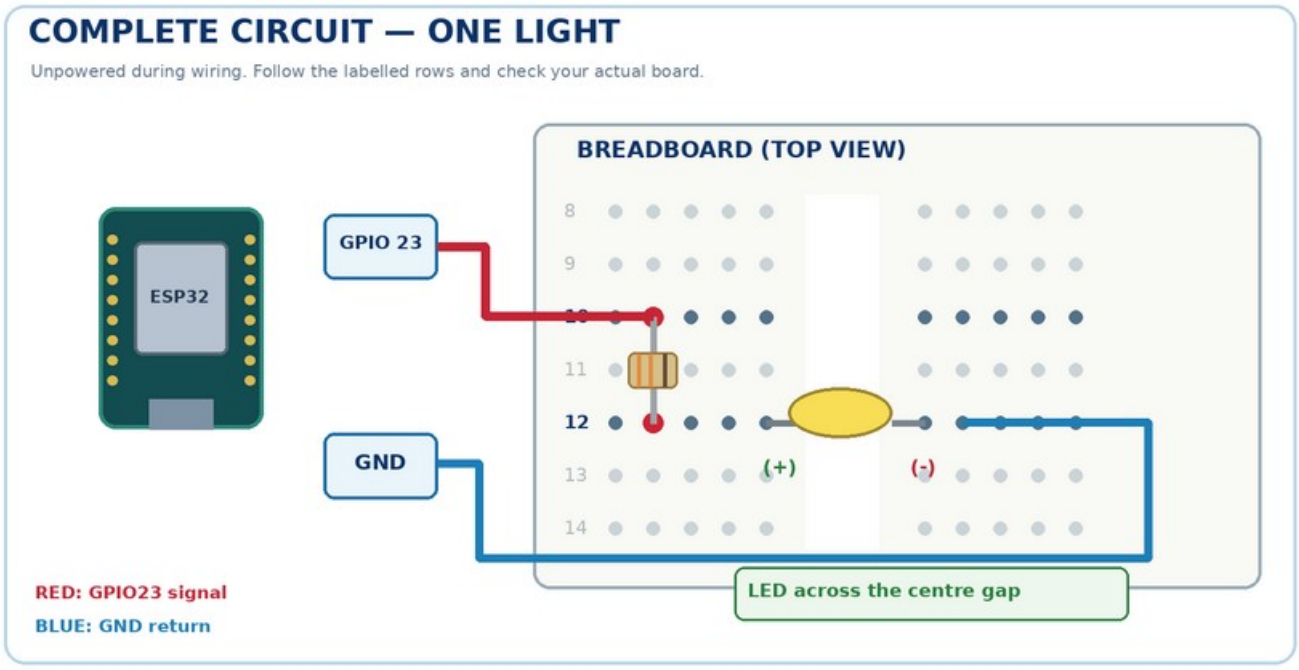
QUESTION	OUR FIRST PREDICTION
If the phone sends “1”, the LED will…	_____
If the phone sends “0”, the LED will…	_____
If no command arrives, the LED will…	_____

⚠️ **SAFETY CHECK**
Bluetooth pairing is allowed only with teacher-approved classroom devices. Never accept requests from unknown devices or share personal information or passwords.

Keep the circuit. Change the code.

The SAME wiring from Activity 1

No extra sensor or Bluetooth module is needed. Use the exact existing LED circuit: ESP32 GPIO23 → 330 Ω resistor → LED anode (+); LED cathode (−) → ESP32 GND.



Wiring guide: recognize your real ESP32 labels. The board is powered OFF while you inspect or adjust the circuit.

Before any upload, follow these steps

1 DISCONNECT USB
Power OFF and compare your circuit with Activity 1. Confirm the resistor, LED polarity, GPIO23 and GND.

2 PLACE YOUR BUILD IN SMARTSPACE
Keep the SAME LED circuit together. With power OFF, place the breadboard and ESP32 on your model base using removable insulating pads. Mark the LED position ROOM LIGHT 1. Do not change any connections.

3 OPEN ARDUINO IDE
Use the teacher-approved computer with the ESP32 board package installed. Choose your correct ESP32 board and serial port.

✅ **TEACHER CHECKPOINT**
☐ ESP32-WROOM-32 Bluetooth Classic model ☐ Activity 1 LED circuit checked ☐ Resistor in series ☐ Correct USB data cable ☐ Compatible approved Android device available

⚠️ **SAFETY CHECK**
ESP32 pins use 3.3 V logic. Never connect the light directly without its resistor. Do not connect the USB cable until the wiring check is complete.

Write the Bluetooth light program

Follow the code from top to bottom. Read the highlighted command characters before uploading.

CODE POINTER | ANNEXURE A2 ● Open the complete Activity 2 Arduino sketch at the back of this book. Copy it as ONE sketch, not separate pieces. Teacher checks board, libraries, pin labels and power before uploading.

Read → Predict → Upload

CODE CLUE	WHAT IT DOES
SerialBT.begin("MMX_SMARTSPACE")	Gives the classroom device its visible Bluetooth name.
cmd == '1'	Turns the light ON (GPIO23 HIGH).
cmd == '0'	Turns the light OFF (GPIO23 LOW).

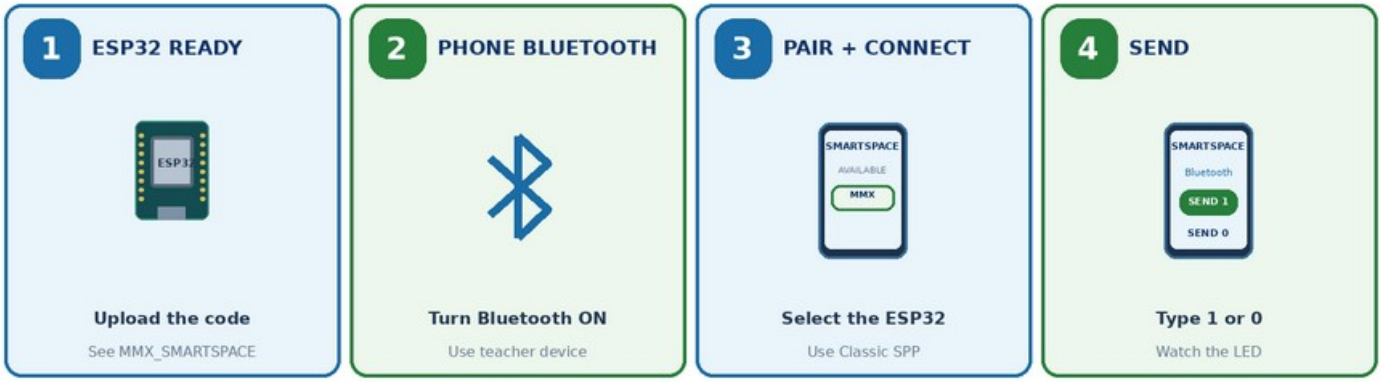
After the teacher checks the circuit, connect USB. Click Verify, then Upload. The LED should remain OFF until a Bluetooth command arrives.

⚠ **SAFETY CHECK**

Use only the approved code and app. Never download unknown files or accept pairing requests from another group. Leave the ESP32 connected to its classroom USB supply.

Connect the phone — safely

This example is for a teacher-approved Android device with Bluetooth Classic SPP; menus can look different on different phones and apps.



1 ASK FOR THE CLASSROOM DEVICE
Use a school-approved Android phone/tablet. Ask your teacher to open the approved Bluetooth Classic serial terminal app.

2 TURN BLUETOOTH ON
Open Android Bluetooth settings or the app connection menu. Look for the nearby device named “MMX_SMARTSPACE”.

3 PAIR / CONNECT
Choose the exact classroom ESP32. A teacher checks any pairing prompt. In the serial terminal, choose the Bluetooth Classic/SPP connection.

4 CHECK “CONNECTED”
The app should show a connected state. If not, do not change ESP32 wires: ask the teacher to verify the app mode and board type.

🔎 **PICTURE HINT**
Look for the exact Bluetooth name “MMX_SMARTSPACE”. This is a radio connection between a nearby phone and the ESP32. You do NOT need school Wi-Fi or mobile Internet for this experiment.

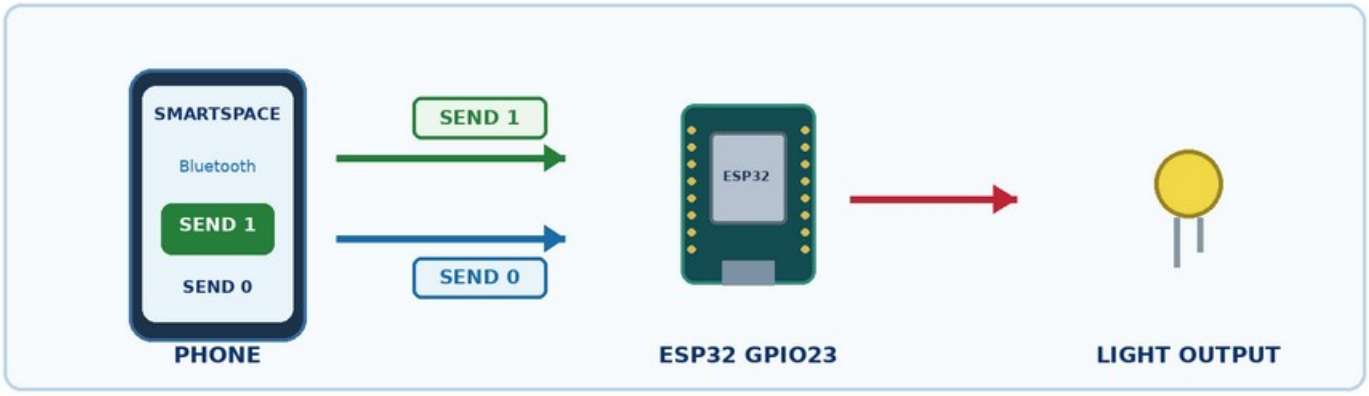
🤔 What happens if the phone finds another group’s ESP32 with a similar name?

_____

⚠️ **SAFETY CHECK**
Only pair with the device identified by the teacher. Do not change passwords, privacy settings or download a new app without permission. Disconnect Bluetooth after the lesson.

ACTIVITY 2 • SEND, TEST & OBSERVE

Send a command. Watch the light.



1 **SEND "1"**
In the approved terminal app, type 1 and press Send. The LED should switch ON. Keep the ESP32 and circuit still.

2 **SEND "0"**
Type 0 and press Send. The same LED should switch OFF.

3 **TEST REPEATEDLY**
Send 1, 0, 1, 0 in order. Observe whether each command causes the expected response. The LED keeps its state until another valid command.

Record what REALLY happened

COMMAND	OUR PREDICTION	WHAT WE OBSERVED
Send 1	_____	_____
Send 0	_____	_____
Send 1 again	_____	_____
Send x (unexpected)	_____	_____

💡 **HINT • WHAT ABOUT "x"?**
The program only recognises "1" and "0". An unrecognised character does not change the light. Think about why this is safer than accepting any character as a valid command.

⚠️ **SAFETY CHECK**
Do not let students wander outside the supervised test area or continue trying to control other groups' devices. Power OFF before changing any wire.

Debug Detective: find the clue

You do not need to be an electronics engineer. Check one simple cause at a time and use the evidence from your test.

🐞 What if something does not work?

WHAT YOU NOTICE	FIRST THING TO CHECK SAFELY
Phone cannot find MMX_SMARTSPACE	ESP32 is powered; the code uploaded; Bluetooth Classic is supported; the device name matches exactly.
App sees the device but cannot connect	Teacher checks Android permissions and whether the app uses Bluetooth Classic SPP rather than BLE.
App connects but LED stays OFF	Try sending just the character 1; if still OFF, disconnect USB and inspect Activity 1 LED wiring.
LED responds to 1 but not to 0	Inspect the two if/else command conditions in the uploaded code.
Another group controls our LED	Stop testing; disconnect and ask teacher to identify/reassign devices before further pairing.

🔧 Design a small improvement

CHALLENGE • WHO CAN CONTROL OUR LIGHT?

Imagine this light in a real classroom. Should ANY nearby phone be allowed to switch it ON and OFF? Discuss one rule to keep classroom control safe. This experiment has no user authentication, so it is only for supervised lab use.

🤔 One design rule our team would add before using wireless control in a real building is...

_____

_____

Add this ability to the project record

SMARTSPACE ROOM LIGHT 1: ☐ Timer-controlled (Activity 1) → ☐ Phone-controlled (Activity 2). Our physical model and light remain the SAME.

Your team roles — rotate!

DEVICE OPERATOR	CODE READER	TEST RECORDER
Sends the 1 / 0 command.	Explains the matching code.	Records expected and actual results.

⚠️ SAFETY CHECK

Disconnect the approved Android device from Bluetooth at the end. Do not leave a remotely controllable student project powered unattended.

ACTIVITY 2 • THINK, REFLECT & VERIFY

What changed without a wire?

You used wireless commands to control the SAME SMARTSPACE light. Your final mission is to explain exactly who sent the instruction and what caused the light to respond.

🤔 Five thinking questions

1. Which device SENDS the command, and which device makes the LED change?

_____

_____

2. Why does our Activity 2 program need BluetoothSerial when Activity 1 did not?

_____

_____

3. What happens when you send 1, 0, and then an unexpected character such as x? Why?

_____

_____

4. What would you check first if the phone connects but the LED stays OFF?

_____

_____

5. Should any nearby phone control the lights in a real classroom? Suggest one safety rule.

_____

_____

📓 My engineer's log

OUR BUILD	OUR TEST	OUR NEXT IDEA
Bluetooth name: _____	Send 1 → LED: _____	I would improve: _____
Send 0 → LED: _____	One problem: _____	One safety rule: _____

4CG evidence • Tick the work you really did

☐ CG-1 We understood the wireless-control mission. ☐ CG-2 We reused the kit and worked safely.

☐ CG-3 We predicted, tested and debugged. ☐ CG-4 We communicated results and shared roles.

🏆 MISSION COMPLETED

Your SAME SMARTSPACE model now has TWO tested behaviours: Activity 1 light blinking and Activity 2 local Bluetooth ON/OFF control. Keep the ESP32, LED and breadboard together on the model base. Activity 3 will add Wi-Fi to this system — not start a different project.

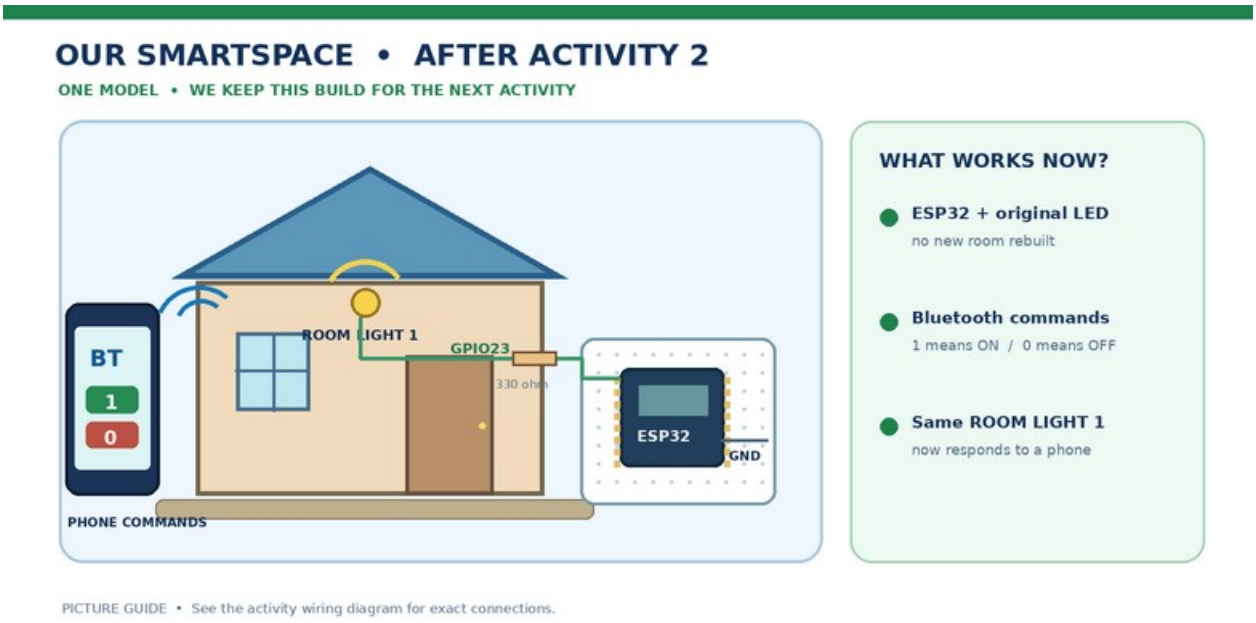
⚠️ SAFETY CHECK

End safely: disconnect the Android device from Bluetooth, then switch OFF USB power. Keep the SAME labelled ROOM LIGHT 1 circuit intact for the next SMARTSPACE mission. Never leave the remote-controlled model powered unattended.

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 2 FINISHED • LOOK AT OUR COMPLETED MODEL

One SMARTSPACE ● Keep everything assembled for the next mission



CHECK OUR MODEL ☐ Same ESP32 ☐ Same ROOM LIGHT 1 ☐ Teacher checked our demonstration

NEXT: We will connect this same SMARTSPACE to a private classroom Wi-Fi network.

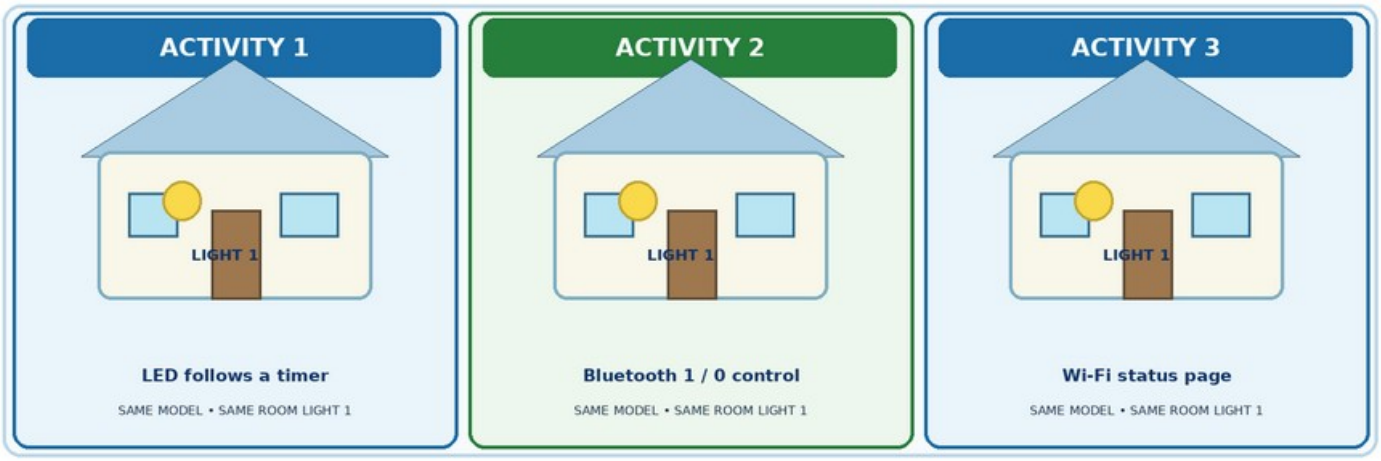
Activity 3 • Bring SMARTSPACE Online

Same model • Same light • New Wi-Fi connection

🎯 TODAY'S MISSION

Keep ROOM LIGHT 1 and your ESP32 from Activities 1–2. Connect that SAME controller to the teacher-approved classroom Wi-Fi. A browser will show "SMARTSPACE ONLINE" and whether the light is ON or OFF. Your earlier Bluetooth control must still work.

Watch ONE project grow



What stays / what changes?

KEEP FROM ACTIVITIES 1 & 2	ADD IN ACTIVITY 3
ESP32 + same LED + 330 Ω resistor + breadboard + labelled model base. Phone Bluetooth commands 1 and 0.	Wi-Fi connection on the ESP32 + its local address + a simple browser STATUS page. No new electronic module.

Our mission in three 45-minute sessions

SESSION 1 • INVESTIGATE	SESSION 2 • PROGRAM	SESSION 3 • TEST
Explain Wi-Fi vs Bluetooth. Confirm the unchanged circuit. Predict what a local address does.	Read and combine code Parts A and B. Teacher enters network details. Verify, upload and inspect Serial.	Open the local status page. Use Bluetooth 1 / 0 again. Observe, debug and improve.

🤔 How can a laptop learn whether our SAME LED is ON without connecting a new wire to it?

_____

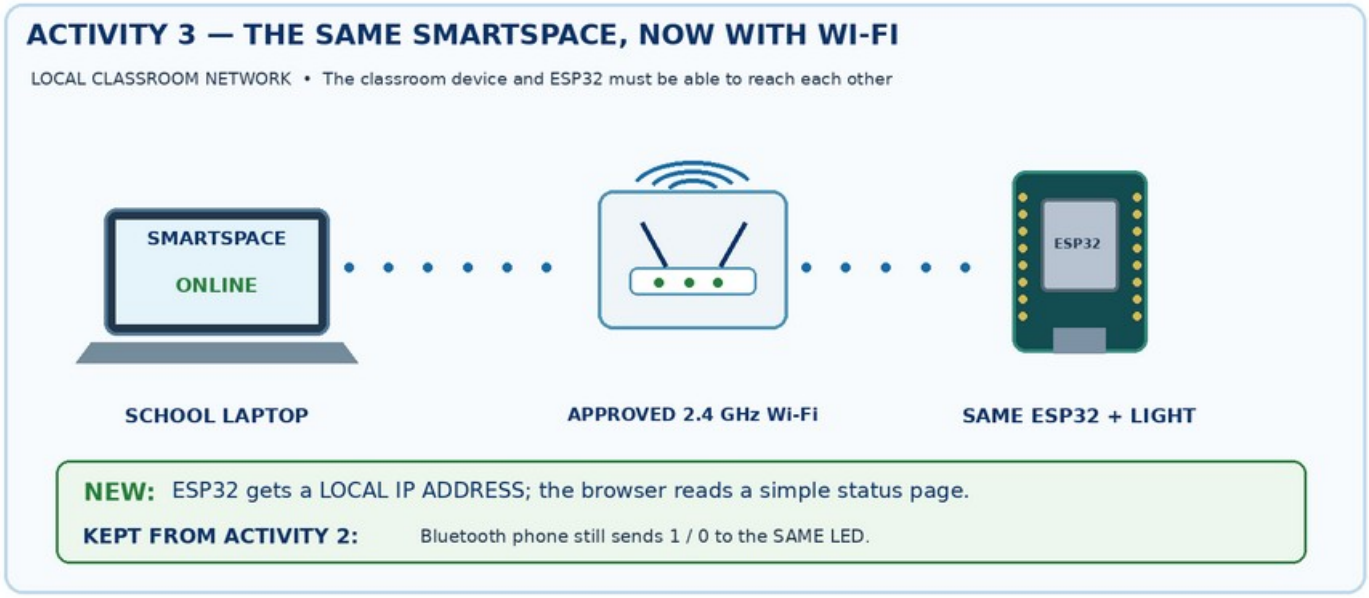
Your engineering decision • What should stay on the model?

KEEP PHYSICALLY ATTACHED	A NEW DIGITAL ABILITY
☐ ESP32 ☐ Light 1 ☐ 330 Ω resistor ☐ Same breadboard	☐ Network name + safe Wi-Fi connection ☐ Local browser status

⚠️ SAFETY CHECK

Keep all hardware exactly as checked in Activity 2. Disconnect USB before inspecting any connection. Use only the teacher-approved 2.4 GHz classroom Wi-Fi; never reveal its password.

Follow the information through Wi-Fi



Look at the two kinds of wireless connection

BLUETOOTH • ACTIVITY 2	WI-FI • ACTIVITY 3
Approved Android phone sends 1/0 directly to the ESP32. The light changes.	Classroom laptop or phone requests a status page through the local Wi-Fi network.
Short-range device-to-device connection.	Devices use a router / access point to communicate within an allowed network.

🧠 **JUST ENOUGH TO KNOW**

A local IP address works like a temporary classroom "device location" on the network. It is not a password and not automatically a public Internet address. Your teacher will show you where it appears on the computer.

PREDICT FIRST • Tick your idea

☐ We must remove Bluetooth first ☐ We can try Wi-Fi while keeping the earlier light and Bluetooth program

🤔 Why might two devices connected to different or isolated Wi-Fi networks fail to reach each other?

_____

_____

Draw a tiny network route • Add the missing arrows

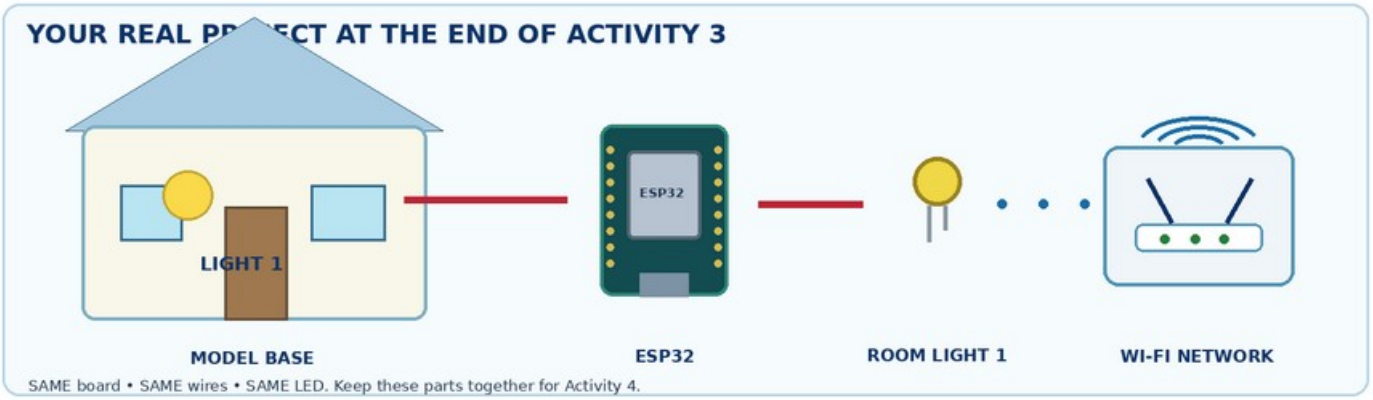
LAPTOP _____ **CLASSROOM ROUTER** _____ **ESP32** _____ **ROOM LIGHT 1**

How is this path different from the DIRECT Bluetooth control path? _____

⚠️ **SAFETY CHECK**

Use a network that allows classroom devices to communicate locally. Do not change router settings, share credentials, use a public hotspot or open control ports on the Internet.

No new wires. More useful connections.



Follow these numbered preparation steps

1 SAVE THE PREVIOUS PROGRAM
Save Activity 2 as “SmartSpace_A2_Bluetooth”. Do not overwrite it. This is our tested working version.

2 CHECK THE UNPOWERED MODEL
Unplug the USB cable. Inspect the SAME GPIO23 → 330 Ω resistor → LED (+) → LED (−) → GND connections. Keep ROOM LIGHT 1 labelled.

3 PREPARE A CLASSROOM NETWORK
Ask the teacher to provide an approved 2.4 GHz Wi-Fi network that allows local device access. The teacher keeps its credentials private.

4 OPEN A NEW ARDUINO SKETCH
Save a new file “SmartSpace_A3_WiFi_BT”. You will combine the code on the next two pages in ONE sketch.

✅ **READY?**
☐ Previous Bluetooth sketch saved ☐ Same circuit checked ☐ ESP32 board/USB verified ☐ Teacher-approved 2.4 GHz network available

🤔 Which hardware component was added between Activity 2 and Activity 3? Explain.

_____

Sketch where the existing brain and light sit in your SMARTSPACE

Draw / write / label here

⚠️ **SAFETY CHECK**
No powered rewiring. The teacher enters Wi-Fi credentials privately; passwords must not appear in students’ shared workbooks, photos or screenshots.

ACTIVITY 3 • MAKE IT THINK | CODE PART A

Programming SMARTSPACE — Complete sketch in Annexure A3

Keep the earlier Bluetooth light and add the Wi-Fi status page. Open Annexure A3 for the complete integrated sketch.

CODE POINTER | ANNEXURE A3 ● Open the complete Activity 3 Arduino sketch at the back of this book. Copy it as ONE sketch, not separate pieces. Teacher checks board, libraries, pin labels and power before uploading.

Read → Predict → Connect the ideas

CODE / CLUE	WHAT IT MEANS
#include <WiFi.h>	ESP32 can connect to classroom Wi-Fi.
#include <WebServer.h>	ESP32 can show a short browser response.
BluetoothSerial BT;	Keeps the Bluetooth commands from Activity 2.
ROOM_LIGHT = 23	Keeps the SAME light and GPIO23 connection.
statusPage()	Reports whether ROOM LIGHT 1 is ON or OFF.

✏️ **YOUR CODE EDIT**

The teacher replaces CLASS_WIFI_2G and TEACHER_ONLY with the approved network details privately. Students do not copy or photograph real Wi-Fi passwords.

🤔 Code detective • Predict before running

IF YOU CHANGE THIS...	WHAT MIGHT CHANGE?
ROOM_LIGHT = 23 becomes another number	_____
Wi-Fi is not available	_____

⚠️ **SAFETY CHECK**

Do not upload Part A alone; it is incomplete until you append Part B. Ask the teacher to check the sketch before entering credentials and connecting USB.

Before uploading • three checks

☐ Parts A+B are in ONE sketch ☐ Teacher reviewed network details ☐ Board and USB port match the real ESP32

✅ **WHAT THIS PROGRAM PRESERVES**

Bluetooth “1” / “0” still controls ROOM LIGHT 1. Wi-Fi adds a local status page without adding any new wires. This is one evolving SMARTSPACE program.

⚠️ **SAFETY CHECK**

Never share the finished sketch if it contains a live Wi-Fi password. Use only a teacher-managed classroom network; do not put this unprotected test page onto the public Internet.

Connect, upload and read the result



1 VERIFY AND UPLOAD
After the teacher checks the wiring, connect USB. In Arduino IDE choose your ESP32 board/port, click Verify, then Upload.

2 OPEN THE SERIAL MONITOR
Choose 115200 baud. Press the ESP32 RESET/EN button if needed. Look for “OPEN: http://” followed by the local IP address.

3 RECORD THE REAL ADDRESS
Write YOUR device’s assigned address below. Do not invent or copy the example address in the picture.

4 USE AN APPROVED BROWSER
On a teacher-approved laptop/tablet on the SAME permitted local network, type http:// followed by the actual address shown in Serial Monitor.

My real test observations

WHAT WE CHECKED	WE OBSERVED
ESP32 Wi-Fi joined the classroom network	☐ Yes ☐ No
Serial Monitor displayed an address	☐ Yes ☐ No
Address shown on OUR device	http:// _____
Browser message	_____

💡 WHAT IF THE MONITOR SAYS “NO WI-FI”?
Do not pull wires out. The Bluetooth light still works. Ask the teacher to check the Wi-Fi name, 2.4 GHz support, router permission and password entry before a supervised retry.

🤔 What can we still test through Bluetooth when the Wi-Fi status page cannot open?

_____

⚠️ SAFETY CHECK
Do not share actual credentials or display them in screenshots. Do not change router settings or make the local status page available outside the classroom network.

A local status page — one light



1 OPEN THE STATUS PAGE
When Wi-Fi is connected, visit the local address in your approved browser. It should show "SMARTSPACE ONLINE" and ROOM LIGHT 1: OFF (initial state).

2 SEND BLUETOOTH "1"
With the approved Android Bluetooth Classic serial app connected, send 1. The SAME LED should turn ON.

3 REFRESH THE BROWSER
Refresh the local status page. It should now report ROOM LIGHT 1: ON.

4 SEND BLUETOOTH "0"
Switch the light OFF with Bluetooth. Refresh the browser and check the updated OFF status.

Record our evidence — do not guess

TEST	PREDICTION	OBSERVED
First browser visit	Status: _____	Status: _____
Bluetooth 1 + refresh	Status: _____	Status: _____
Bluetooth 0 + refresh	Status: _____	Status: _____

🔎 **THINK ABOUT CONTROL**
The browser shows STATUS only. It cannot switch the light in Activity 3. We will build a proper IoT control dashboard later in Activity 7.

A tiny engineering challenge • Verify with evidence

What if you do not refresh the page after sending Bluetooth 1? Does the displayed status update automatically? Explain what you saw.

Our observation: _____

⚠️ **SAFETY CHECK**
Keep this unauthenticated status page on the teacher-approved local test network only. Never forward its address to outsiders or expose the ESP32 to the Internet.

Mission complete • SMARTSPACE grows

🐞 Debug Detective • Investigate before asking

SYMPTOM	FIRST SAFE CHECK
No Wi-Fi address appears	Teacher checks 2.4 GHz network, classroom access, credentials and Serial Monitor at 115200.
Address appears but browser fails	Are both devices on the same approved reachable LAN? Some guest networks block device-to-device access.
Browser says OFF after Bluetooth 1	Confirm Bluetooth is connected; light physically changed; refresh the browser page.
Bluetooth cannot pair after Wi-Fi	Check that original ESP32-WROOM-32/Classic SPP is used; try one connection at a time; ask teacher.

🤔 Five thinking questions • Write and discuss

1. Which components from Activity 2 remained unchanged in Activity 3?

_____

_____

2. Why do the ESP32 and the classroom laptop need a permitted local Wi-Fi connection?

_____

_____

3. What is the difference between sending Bluetooth "1" and refreshing the Wi-Fi status page?

_____

_____

4. A browser cannot open the local address. What two checks should your team make first?

_____

_____

5. How would you protect a real classroom SMARTSPACE from unknown network users?

_____

_____

📓 Engineer's Log + 4CG learning evidence

WHAT WORKED	WHAT WE IMPROVED	NEXT MISSION
_____	_____	Add sensors to THIS model
☐ CG-1 inquiry	☐ CG-2 safe practical work	☐ CG-3 test ☐ CG-4 share

🏆 SMARTSPACE NOW

ACTIVITY 1: timed light → ACTIVITY 2: Bluetooth ON/OFF → ACTIVITY 3: Wi-Fi status. SAME ESP32 + SAME LED. Keep it assembled; Activity 4 gives THIS model new senses.

⚠️ SAFETY CHECK

End safely: disconnect the Bluetooth phone, close the browser, unplug USB, and keep the labelled ROOM LIGHT 1 model intact. Teacher should clear the classroom network credentials before equipment leaves school.

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 3 • OUR SMARTSPACE BUILD CHECKPOINT

Look! This is OUR SMARTSPACE so far

One model. Three completed milestones. We are keeping the same build for the next mission.

What should be visible on your actual project today?

PHYSICAL PART — KEEP IT ON THE BASE	TESTED ABILITY — KEEP IT IN THE CODE
ESP32-WROOM-32, USB access, breadboard, resistor and ROOM LIGHT 1	1. Blink test from Activity 1 (saved earlier sketch).
SAME GPIO23 and GND connections — no extra module added	2. Bluetooth ON / OFF from Activity 2 (present in new sketch).
Room Light 1 labelled so teams can identify it	3. Wi-Fi status page through an allowed classroom network.

📷 **PICTURE CHECK • BUILD VS FINAL GOAL**

This drawing shows the current stage, not the final SMARTSPACE. There is no servo door, fan, LCD or environmental sensor connected yet. We will add those parts step by step, without discarding this working light.

🔎 Future feature planner • Find a place on your SAME model

NEXT COMPONENT TO ADD	WHERE COULD IT GO?
Light / motion / room sensors (Activity 4)	_____
Model door, fan and status display (later)	_____

🤔 Why is it useful to leave room for future components before attaching more wires?

_____

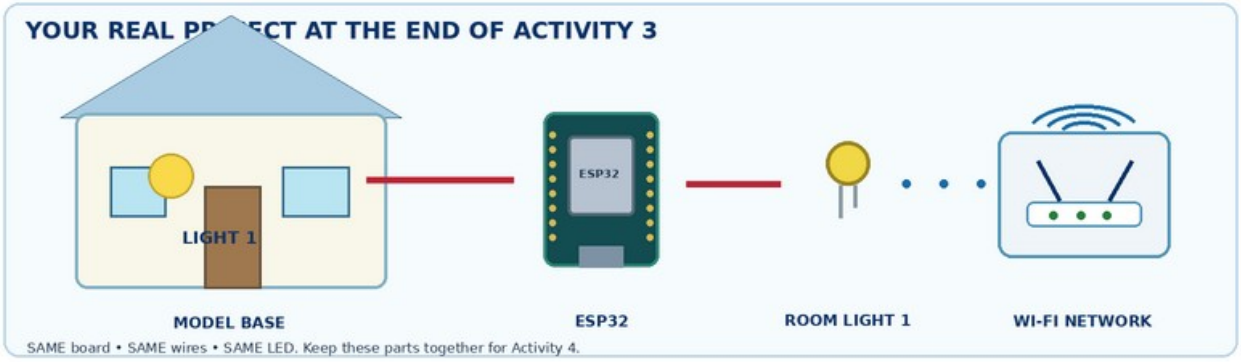
_____

⚠️ **SAFETY CHECK**

Before packing: unplug the USB power, disconnect the classroom phone from Bluetooth, and keep the current ESP32 + LED circuit labelled and intact. Never leave the kit powered unattended.

Team confirmed: _____ **Teacher initials:** _____

ACTIVITY 3 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY



Activity 4 • Give SMARTSPACE Senses

LOOK → CONNECT → READ → TEST → RECORD

Our mission today

Yesterday our ESP32 could switch ROOM LIGHT 1 through Bluetooth and show its status on classroom Wi-Fi. Today we keep those features and add three sensors so SMARTSPACE can observe its surroundings.



NEW CAPABILITY ● WHAT IS DIFFERENT NOW?

Our model will READ room brightness, temperature/humidity and motion. We will NOT automate the door or fan yet; those upgrades come later.

Before we build • look and predict

🤔 **If a room has sensors, does that automatically mean it can make good decisions? Why?**

_____

_____

Four lab sessions • 45 minutes each

SESSION 1	SESSION 2	SESSION 3	SESSION 4
Meet components + LDR wiring	Test LDR; add DHT11	Add PIR; update code	Observe + debug + 5 questions

⚠️ **SAFETY CHECK — STOP ● SWITCH OFF ● CHECK ● ASK**

KEEP the ESP32, ROOM LIGHT 1 and resistor connected exactly as before. Disconnect USB before adding any new sensor. A teacher checks every 3.3V/5V power connection.

Our three-person engineering team • rotate roles

CONNECTOR	READER + CODER	TESTER + RECORDER
Name: _____	Name: _____	Name: _____

ACTIVITY 4 • PICTURE FIRST

Meet the New Sensors

Find each part • say its name • describe its job • tick after spotting

PICTURE	NAME • QUANTITY	WHAT IT DOES IN OUR PROJECT
	**DHT11 module • 1**	Shows room temperature and relative humidity.
	**HC-SR501 PIR • 1**	Notices movement; needs teacher-approved 5V supply and warm-up.
	**LDR sensor with AO • 1**	Provides a changing analog light reading.

Three questions to discuss before wiring

🤔 **Which part should be near the door? Which part should be away from direct sunlight?**

_____

🤔 **How could a sensor tell us something without controlling any device?**

_____

Plan where each sensor belongs on YOUR model

PART	MY CHOSEN LOCATION	MY REASON
Light sensor	_____	_____
DHT11	_____	_____
PIR motion	_____	_____

📷 **PICTURE HINT**

These pictures help you RECOGNISE components. Exact board pin order and labels can differ. Read YOUR sensor markings before using the later wiring maps.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

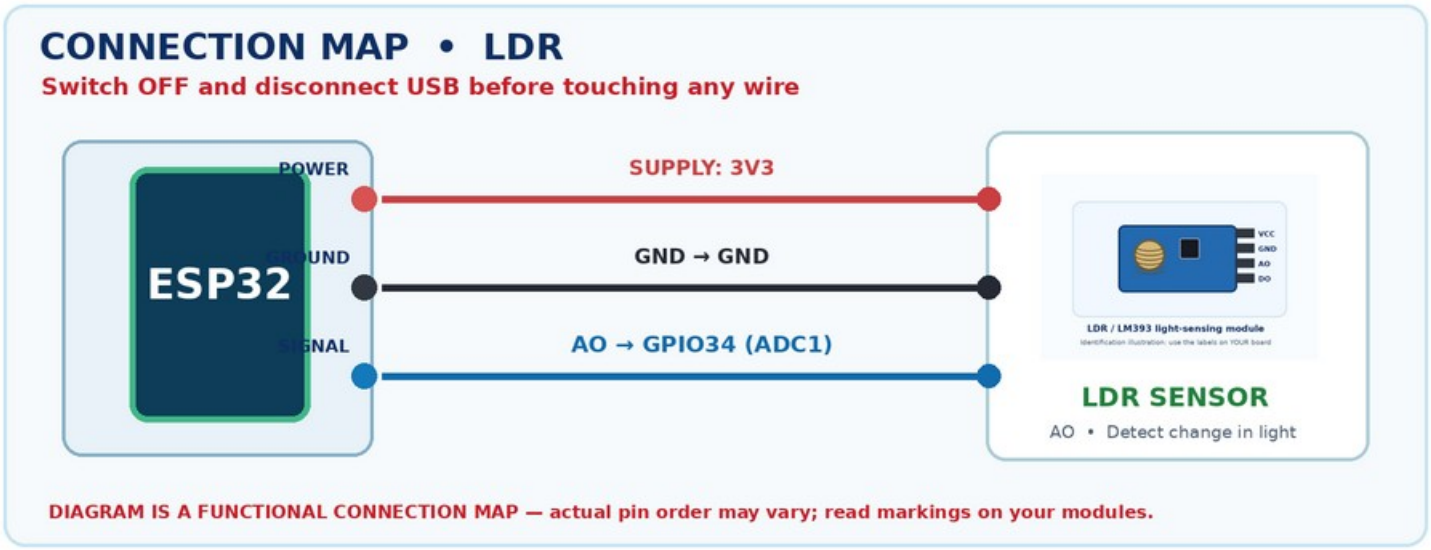
For this activity, use a DHT11 module with a fitted pull-up resistor and an LDR module that exposes an AO (analog-output) pin. Do not wire a different version by guessing.

✏️ **FIND & DRAW • Sketch the sensor pin labels you can actually see on one module.**

Step 1 • Add the Light Sensor

SAME ROOM LIGHT 1 • NEW LIGHT INPUT

The LDR measures light; it does not automatically switch anything yet.



Connect only these three new wires

LDR label (verify)	ESP32 connection	WIRE CHECK
VCC	3V3 (regulated 3.3 V)	☐
GND	GND	☐
AO (analog output)	GPIO34 / ADC1	☐

1	**POWER OFF —** Unplug ESP32 USB and any external supply.
2	**READ LABELS —** Find VCC, GND and AO on your actual light module.
3	**CONNECT SIGNAL —** AO goes to GPIO34 (ADC1 input). Keep GPIO23 connected to the original LED.
4	**CONNECT POWER —** VCC to 3V3; GND to the common ground rail.
5	**VERIFY —** Trace the three wires with a partner; ask your teacher to approve before power.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Never connect a 5V AO output to GPIO34. Power a compatible LDR module at 3.3V and check its actual labels. ESP32 analog inputs are 3.3V-domain signals.

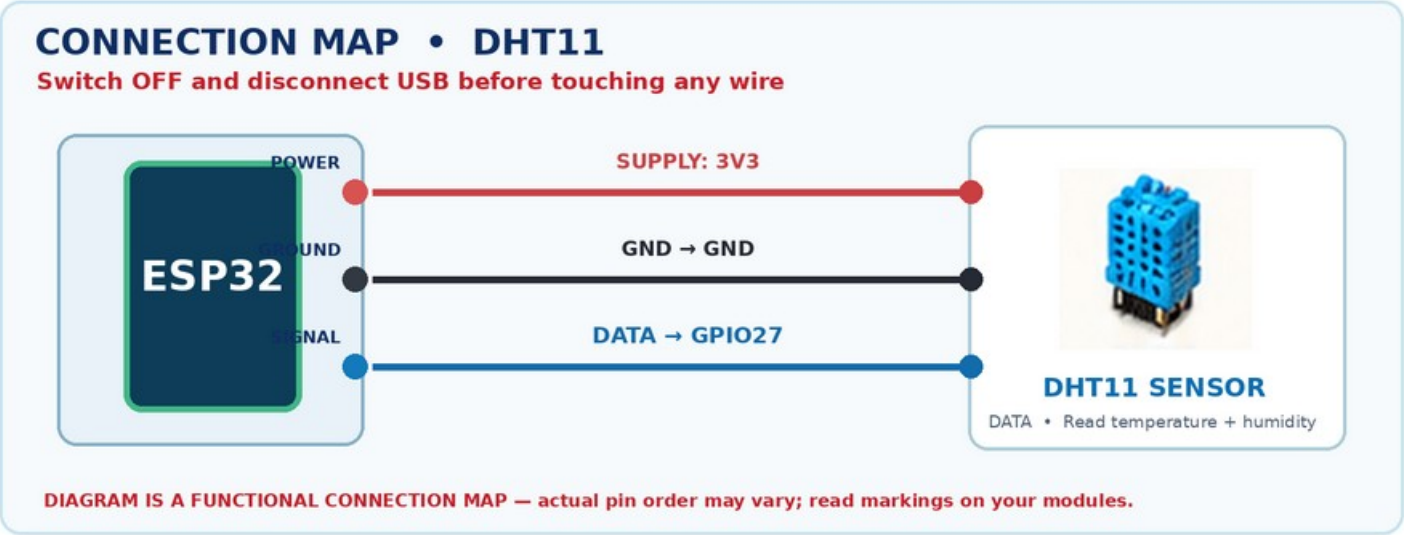
🤔 **Why did we keep ROOM LIGHT 1 on GPIO23 instead of moving it to GPIO34?**

_____

_____

Step 2 • Add Room Climate Sensing

KEEP LIGHT 1 + LDR IN THE SAME BUILD



Our next three connections

DHT11 module label	ESP32 connection	CHECK
VCC / +	3V3 (supported module)	☐
GND / –	GND	☐
DATA / S	GPIO27	☐

1	**UNPLUG USB —** Make sure the ESP32 is completely unpowered.
2	**CHECK MODULE TYPE —** Use the prepared 3-pin DHT11 breakout module. A bare 4-pin sensor needs additional wiring; do not assume it is equivalent.
3	**CONNECT DATA —** Find DATA/S on the actual module and connect to GPIO27.
4	**CONNECT SUPPLY —** Teacher confirms 3.3V-compatible VCC and common GND.
5	**POSITION & VERIFY —** Place away from the warm ESP32 or model fan. Ask for a power-on check.

Predict • what will the two numbers tell us?

🤔 **If a classmate gently warms the air near the sensor, should temperature and humidity necessarily change in the same way?**

_____

📌 **READING TIP**

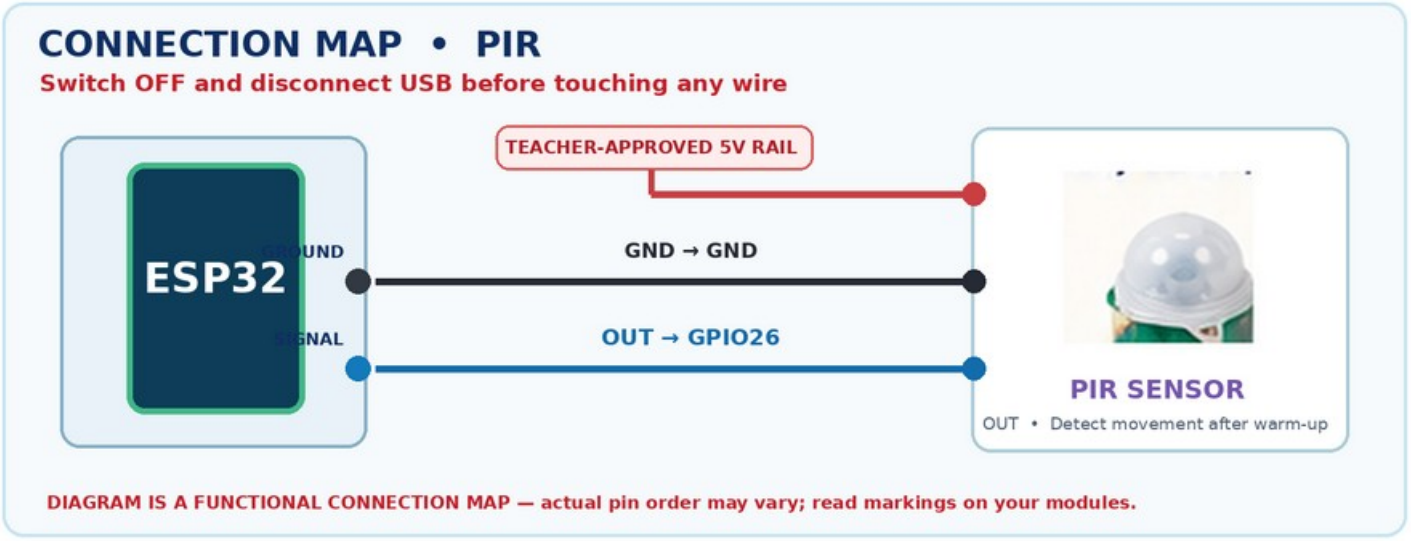
Temperature is shown in °C; relative humidity is shown in %. DHT11 updates slowly: allow a few seconds between readings. A blank/NaN reading is a troubleshooting clue.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Never guess VCC, GND or DATA positions from a product photo. Follow the actual breakout's printed labels; unplug before any correction.

Step 3 • Add Movement Detection

THE SAME MODEL NOW OBSERVES PEOPLE MOVING



PIR requires a teacher-verified 5V rail

PIR module marking	Safe connection	CHECK
VCC	Teacher-approved regulated 5V supply	☐
GND	Shared GND with ESP32	☐
OUT	GPIO26 ONLY IF measured output ≤ 3.3 V	☐

1	**POWER OFF —** Disconnect USB and teacher-controlled external power.
2	**FIND PINS —** Identify VCC, OUT, GND on the actual HC-SR501 PCB. Connector order varies by view.
3	**VERIFY THE SIGNAL —** Teacher checks that PIR OUT is safe for a 3.3V ESP32 input; do not assume 5V supply means 5V output.
4	**CONNECT —** OUT to GPIO26, common GND, and VCC to approved 5V.
5	**POSITION —** Point the white dome at the model entrance; keep it away from moving wires and direct heat.
6	**WARM UP —** After power-on, allow roughly 30–60 seconds or as the supplier specifies before judging readings.

⚠ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

NEVER put a 5V logic signal directly into GPIO26. A teacher must verify the real module output and power arrangement; use suitable level shifting if required.

🌐 **PRIVACY THINKING**

A PIR sensor only indicates movement; it cannot identify the person. We do not need photographs or names to test this classroom model.

Step 4 • Make the ESP32 Read All Sensors

KEEP Bluetooth + Wi-Fi + ROOM LIGHT 1

One upgraded sketch — Complete code in Annexure A4

WHAT WE ADD IN THE CODE

GPIO34 reads analog light, GPIO27 reads DHT11 data, and GPIO26 reads PIR movement. ROOM LIGHT 1 stays on GPIO23. The browser will display sensor readings, not control appliances yet.

CODE POINTER | ANNEXURE A4 • Open the complete Activity 4 Arduino sketch at the back of this book. Copy it as ONE sketch, not separate

pieces. Teacher checks board, libraries, pin labels and power before uploading.

🔎 CODE HINT • GPIO34 is an ADC1 input, which can be read while the ESP32 uses Wi-Fi. GPIO23 is still the original LED output.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Only use the full sketch on these three code pages as ONE program; never upload an incomplete Part A. Library install: Adafruit DHT sensor library plus its dependency, if prompted.

Step 4 • Keep Reading the Same Program

SAVE • UPLOAD • READ

TEACHER SETUP • PRIVATE CLASSROOM Wi-Fi

In the starter code, CLASS_WIFI_2G and TEACHER_ONLY are placeholders. Your teacher enters approved details locally. Never submit or share a sketch containing a real school Wi-Fi password.

Step 4 • Complete the Sensor Program

OUR MAIN LOOP • KEEP THE LIGHT CONTROLLABLE

Use the full integrated sketch in Annexure A4.

What this program does — predict first

EVERY LOOP	EVERY ~2.5 SECONDS	ON WEB PAGE
Listen for Bluetooth 1 / 0	Read LDR, DHT11 and PIR	Show latest sensor readings after refresh

🤔 **Which line changes ROOM LIGHT 1? Which lines only READ information?**

_____

✅ **SAVE • UPLOAD • KEEP THIS SKETCH**

Save the sketch as SMARTSPACE_Activity4_Integrated_Sensors.ino, then upload and perform the tests on the following pages. Do not discard your earlier working model.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

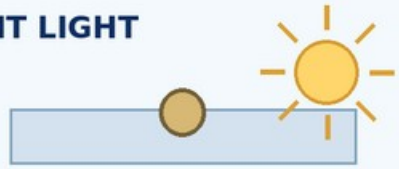
Do not upload code with a real Wi-Fi password into a shared repository or public page. Do not touch sensor wires while the board is powered.

Step 5 • Discover the Light Reading

BRIGHT OR SHADED? MEASURE BEFORE WRITING ANY RULE

TEST A

BRIGHT LIGHT



Do not use direct sunlight

ADC reading: _____

TEST B

GENTLE SHADE



Move a hand nearby

ADC reading: _____

1	**PREDICT —** Will the number go up or down when the room becomes brighter? Circle: UP / DOWN / NOT SURE.
2	**RUN THE INTEGRATED SKETCH —** Now that all three sensors are connected and the full sketch is uploaded, open Serial Monitor at 115200 baud.
3	**TEST TWO CONDITIONS —** Read the Light ADC number in ordinary room light; then make a gentle hand shadow. Do not touch the board.
4	**CHANGE JUST ONE THING —** Try a different angle or greater distance from the lamp, and compare.
5	**CHECK THE DIRECTION —** Record which condition produced the larger value. Different LDR modules can behave differently.

My real observations • no sample answers

CONDITION	LIGHT ADC READING	WHAT CHANGED?
Bright	_____	_____
Gentle shadow	_____	_____
My conclusion	_____	_____

💡 **THINKING HINT**

A sensor reports a changing number. Before we choose an automatic light rule in a later activity, we must measure our own bright and dim ranges.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Do not shine lasers or intense lamps into sensors. Keep fingers dry and do not move wires while powered.

✏️ **PLOT YOUR EVIDENCE • Draw a taller bar for whichever condition gave the higher ADC value.**

Step 6 • Test the Same SMARTSPACE

MEASURE SENSES WITHOUT LOSING CONTROL

A. Upload safely

1	**CHECK FOUR PINS —** LED stays on GPIO23; LDR AO uses GPIO34; DHT11 DATA uses GPIO27; PIR OUT uses GPIO26.
2	**LOAD THE SKETCH —** Arduino IDE → choose your exact ESP32 board/port → install DHT library if needed → upload entire program.
3	**OPEN SERIAL MONITOR —** Set 115200 baud; let the PIR stabilise.
4	**CHECK Wi-Fi / Bluetooth —** Open the classroom local address in a browser; keep teacher-approved phone paired to Bluetooth.
5	**TEST ONE CONDITION AT A TIME —** Cover LDR gently, wait for DHT updates, then walk near PIR. Refresh browser to view current readings.

B. Our observation record

TEST	WHAT I PREDICTED	WHAT I OBSERVED
Light → shade	_____	ADC: _____ → _____
Temperature / humidity	_____	_____ °C; _____ %
No movement → movement	_____	PIR: _____ → _____
Bluetooth command 1 then 0	_____	LED: _____ → _____
Reload Wi-Fi status page	_____	Page: _____

🤔 **THINK BEFORE ANSWERING**

The light can still be switched by a person over Bluetooth. Do the new sensors control it automatically yet? Look at the program logic.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Use the browser only on the approved local network. Do not port-forward, publish a local IP address online or share private Wi-Fi credentials.

🤔 **Which measurement changed most reliably? What one thing would you test again?**

_____

_____

Debug Detective • Think & Reflect

WHAT WORKED? WHAT DID WE LEARN?

🐞 Debug Detective • troubleshoot with evidence

SYMPTOM	CHECK BEFORE CHANGING ANYTHING
Light always reads the same	Is AO really connected to GPIO34? Is the module receiving 3.3V?
Temperature shows “warming up”	Has enough time passed? Are DHT DATA and the module pull-up correct?
PIR never changes	Has it warmed up? Is the 5V supply correct? Is OUT safe for GPIO26?
Bluetooth works but browser does not	Is the 2.4 GHz Wi-Fi connection active? Are devices on the same allowed local network?

🤔 Five thinking questions • write your own answers

1. What does the LDR measure? Why must we test it in both bright and shaded conditions?

_____

_____

2. What is the difference between the information from the DHT11 and the PIR sensor?

_____

_____

3. Why did we choose GPIO34 for the light sensor instead of another analog pin?

_____

_____

4. A movement sensor reports HIGH, but the door does not open. Is that a problem at this stage? Explain.

_____

_____

5. Imagine designing a classroom that saves energy. Which two sensor readings would help, and why?

_____

_____

📓 Engineer’s Log • 4CG evidence

CREATE / BUILD	THINK / TEST	COLLABORATE / EXPLAIN
I added _____.	I checked _____.	Our team changed _____.

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 4 • PROJECT CHECKPOINT

Look! Our SMARTSPACE Now Has Senses

ONE MODEL • FOUR ACTIVITIES COMPLETED

What stays assembled and ready for Activity 5?

PHYSICAL BUILD • KEEP IT IN PLACE	WORKING FEATURE • KEEP IN THE CODE
ESP32 + ROOM LIGHT 1 + resistor (GPIO23)	Bluetooth commands 1 / 0 for the same light
LDR → GPIO34 (ADC1)	Live light value
DHT11 → GPIO27; PIR → GPIO26	Temperature, humidity and movement readings
Existing Wi-Fi connection and status page	Browser sensor report after manual refresh

🚀 **OUR NEXT UPGRADE • ACTIVITY 5**

Now SMARTSPACE can SENSE three things. Next we add outputs—a lightweight servo door, a safely powered fan, an alert and an LCD display—to make it RESPOND. We will keep all today's sensors in the same model.

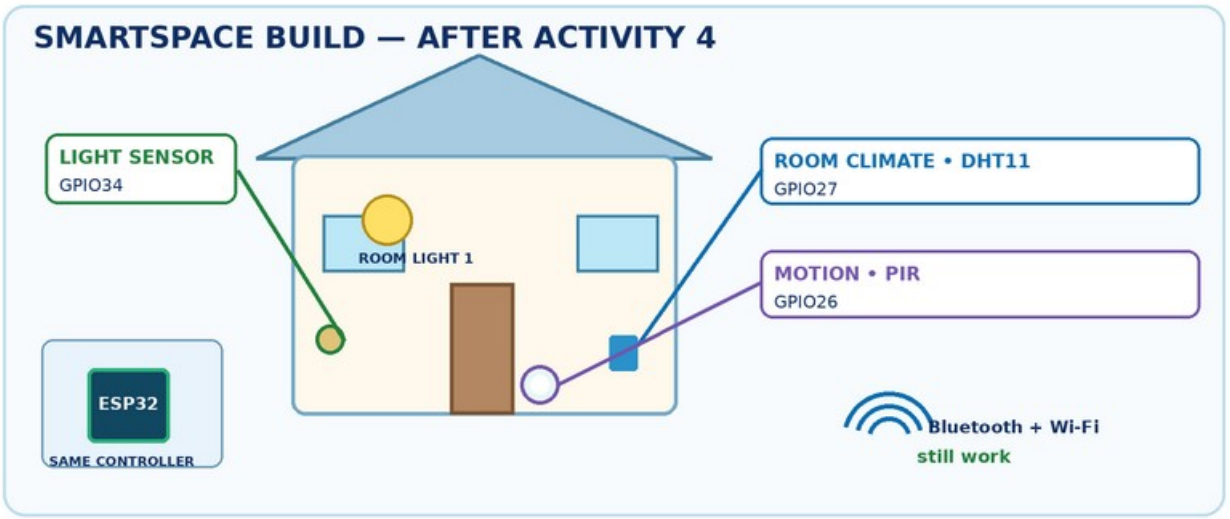
🤔 **Where should the model door, fan and LCD fit so the sensors and wires remain accessible?**

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Finish safely: disconnect power, protect sensor wiring, label GPIO23/26/27/34, and store the single assembled model for the next lesson.

✏️ **NEXT BUILD • Sketch where a moving door, safe model fan and LCD will fit WITHOUT removing these sensors.**

ACTIVITY 4 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY

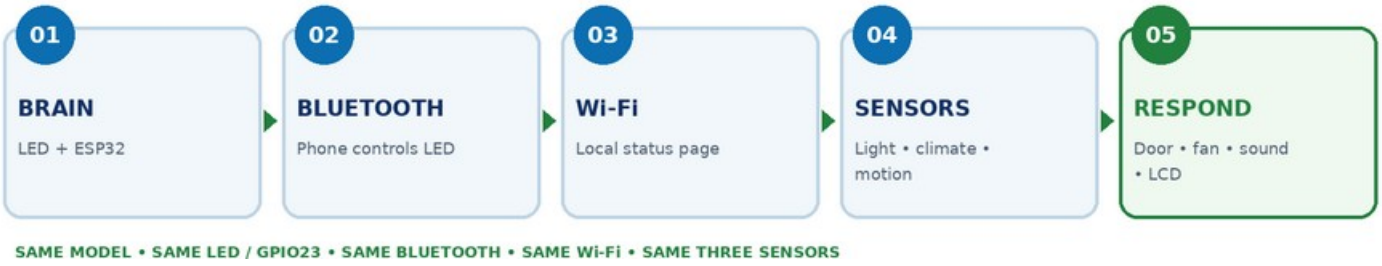


Activity 5 • Make SMARTSPACE Respond

LOOK → CONNECT → CONTROL → TEST → IMPROVE

Our mission today

In Activity 4, our SMARTSPACE could already measure light, movement, temperature and humidity. Today we do NOT dismantle it. We add four output features — a small door, a model fan, an alert and a status display — while ROOM LIGHT 1, Bluetooth and Wi-Fi keep working.



TODAY = MANUAL RESPONSE ● LATER = AUTOMATION

We will give a command from the phone and watch one device act. In Activity 6 we will connect SENSOR DATA to automatic decisions. Keep these ideas separate.

🤔 Wonder before we build

If the motion sensor notices someone, does the servo door open automatically already? What else must we program?

_____

_____

Five 45-minute sessions • teacher may divide complex wiring into shorter periods

SESSION 1	SESSION 2	SESSION 3	SESSION 4	SESSION 5
Door build + servo	Driver + model fan	Buzzer + LCD	Integrated sketch	Test + 5 questions

⚠️ SAFETY CHECK — STOP ● SWITCH OFF ● CHECK ● ASK

STOP and switch OFF before any connection change. NEVER power a servo, motor or 5V LCD directly from an ESP32 GPIO or 3.3V pin. All power rails must be teacher-verified.

ACTIVITY 5 • PICTURE FIRST

Find the New Outputs

USE THE SAME CLASS 7 PICTURE → NAME → USE PATTERN



What we already have • what we add now

KEEP FROM ACTIVITIES 1–4	ADD TO OUR SAME MODEL
ESP32 • ROOM LIGHT 1 • LDR • DHT11 • PIR • Bluetooth • Wi-Fi	SG90 servo • compatible motor driver + guarded fan/BO motor • active buzzer • I2C LCD + logic level shifter

🔍 **PICTURE HINT**

Look for the servo's three labeled leads, driver input/output markings, buzzer input, and LCD I2C backpack. Product pictures are for RECOGNITION only; labels on your own modules govern wiring.

🤔 Design before connecting

Where will you put the door? Can the LCD be read without touching the sensor wires?

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Our motor is a miniature classroom demonstration. Use a guarded 5V fan or an enclosed motor indicator; never attach exposed or sharp fan blades to the BO motor.

Step 1 • Add the Automatic Door

REUSE THE SAME ESP32 • ADD ONE SG90 SERVO



Find the three leads • check YOUR servo markings

WIRE / SERVO PIN	CHECKED CONNECTION	TICK
Control signal (often orange/yellow)	ESP32 GPIO18	☐
V+ (often red)	Regulated external 5V suitable for SG90	☐
GND (often brown/black)	Shared ground with ESP32	☐

1	**POWER OFF —** Disconnect ESP32 USB and switch OFF the external supply.
2	**BUILD THE DOOR —** Make a lightweight cardboard door and hinge. Avoid heavy loads and sharp corners.
3	**ATTACH SERVO —** Mount SG90 securely; do not force its shaft or arm.
4	**WIRE SIGNAL —** Connect verified servo control lead to GPIO18.
5	**CHECK POWER —** Teacher connects rated 5V power and a common GND; NOT to 3.3V GPIO.
6	**TEST SLOWLY —** Only after uploading the completed sketch, command D for OPEN then d for SHUT. Reduce angles if the hinge strains.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

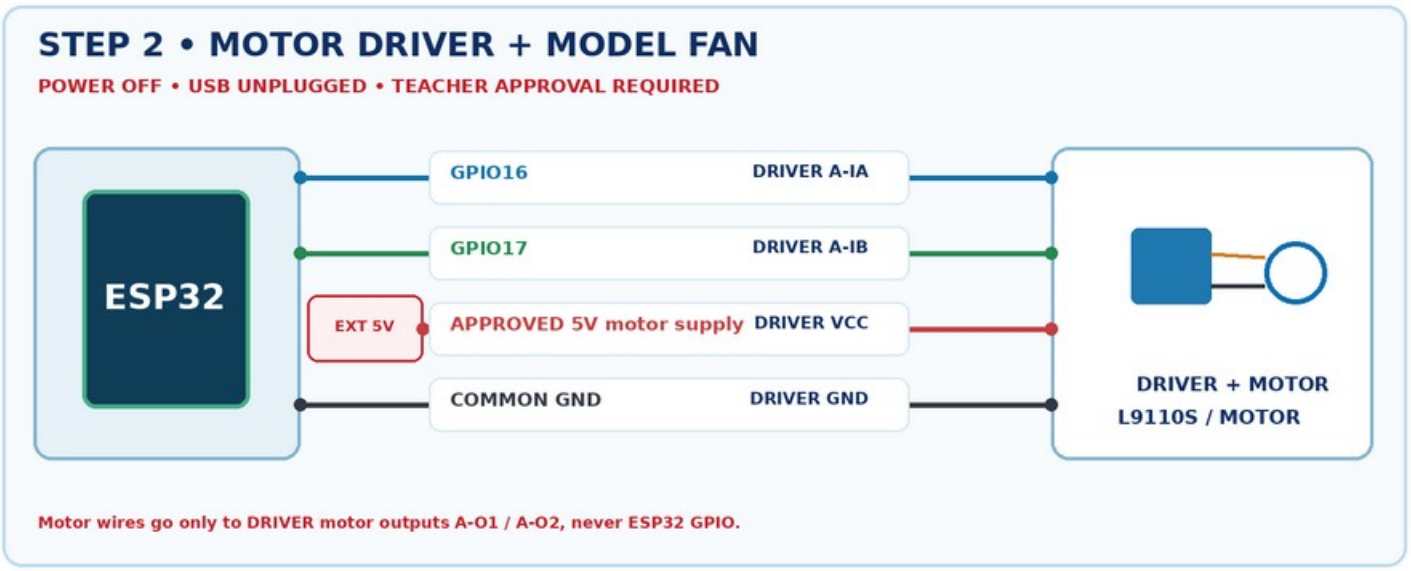
The servo may draw high current when moving. Use a suitable external 5V regulated supply and common ground; do not use the ESP32 3.3V pin. Do not trap fingers in the door.

🤔 **What would happen if the door hinge were too tight for the servo?**

_____

Step 2 • Add the Model Fan

KEEP THE DOOR AND ROOM LIGHT WIRED



L9110S driver mapping • verify the module markings

ACTUAL DRIVER PIN	CONNECT TO	CHECK
A-IA and A-IB (control)	ESP32 GPIO16 and GPIO17	☐
VCC / VM (motor supply)	Approved regulated 5V motor rail	☐
GND	Common GND shared with ESP32	☐
A-O1 and A-O2 (motor outputs)	Two insulated motor leads	☐

1	**UNPLUG ALL POWER —** Do not change motor leads with power connected.
2	**IDENTIFY DRIVER —** Read the physical L9110S input/output labels. Some boards are marked differently.
3	**CONNECT SIGNALS —** GPIO16 → A-IA; GPIO17 → A-IB.
4	**CONNECT MOTOR —** Attach the two DC motor leads to driver A-O1 and A-O2, not ESP32.
5	**VERIFY POWER —** Teacher checks driver supply voltage, current and common ground.
6	**TEST & STOP —** After uploading the combined sketch, F runs the model fan; f stops it. Observe for 3 seconds then stop.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

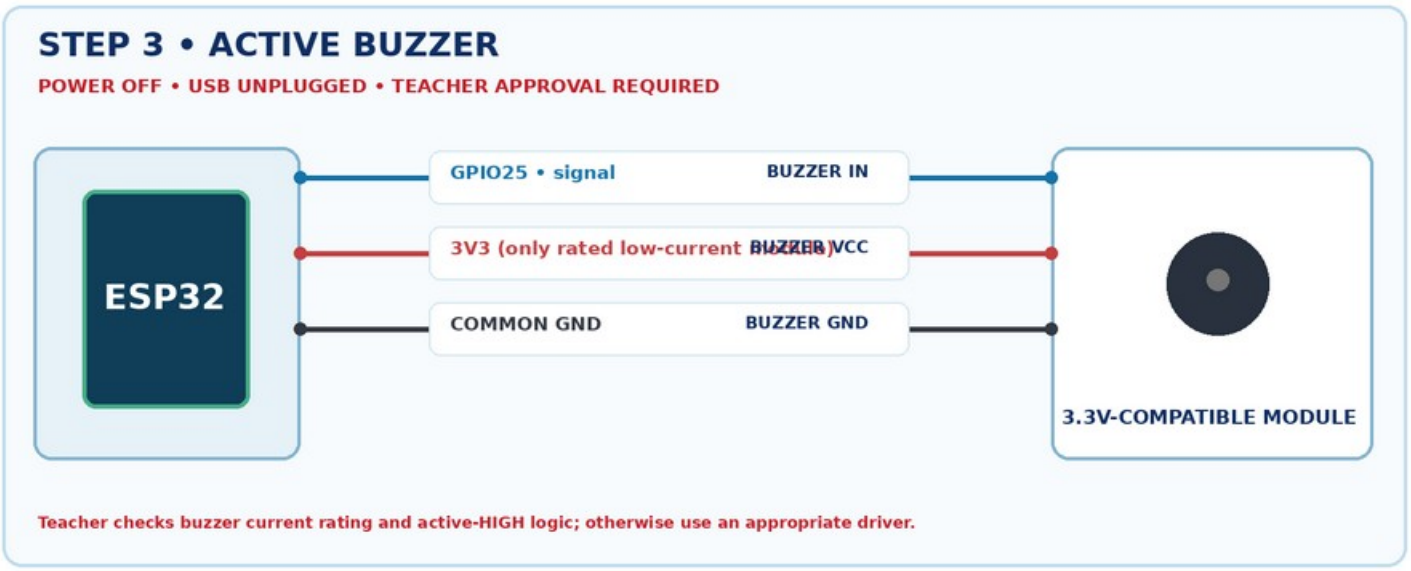
Never connect the motor directly to GPIO16/17. Keep wires, hair and fingers away from the rotating shaft; use a guarded classroom fan or an enclosed visible motor indicator.

💡 **THINKING HINT**

A motor driver is a power interface: the ESP32 decides, but the driver supplies controlled power to the motor.

Step 3 • Add a Short Alert

THE SAME MODEL NOW HAS SOUND



Use a 3.3V-compatible active buzzer module

MODULE MARKING	CONNECTION	CHECK
IN / SIG (active HIGH)	ESP32 GPIO25	☐
VCC	3V3 only for a specifically rated low-current module	☐
GND	Common GND	☐

1	**POWER OFF —** Disconnect USB and external power.
2	**READ THE LABEL —** Check whether the module is active-HIGH or active-LOW; this example assumes active-HIGH.
3	**VERIFY CURRENT —** Teacher checks that the buzzer input and power draw are ESP32-compatible; otherwise uses a suitable driver.
4	**CONNECT —** IN → GPIO25; VCC and GND to the verified supply rails.
5	**TEST ONCE —** After upload, send B from the phone. Listen for one short beep, not a continuous alarm.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

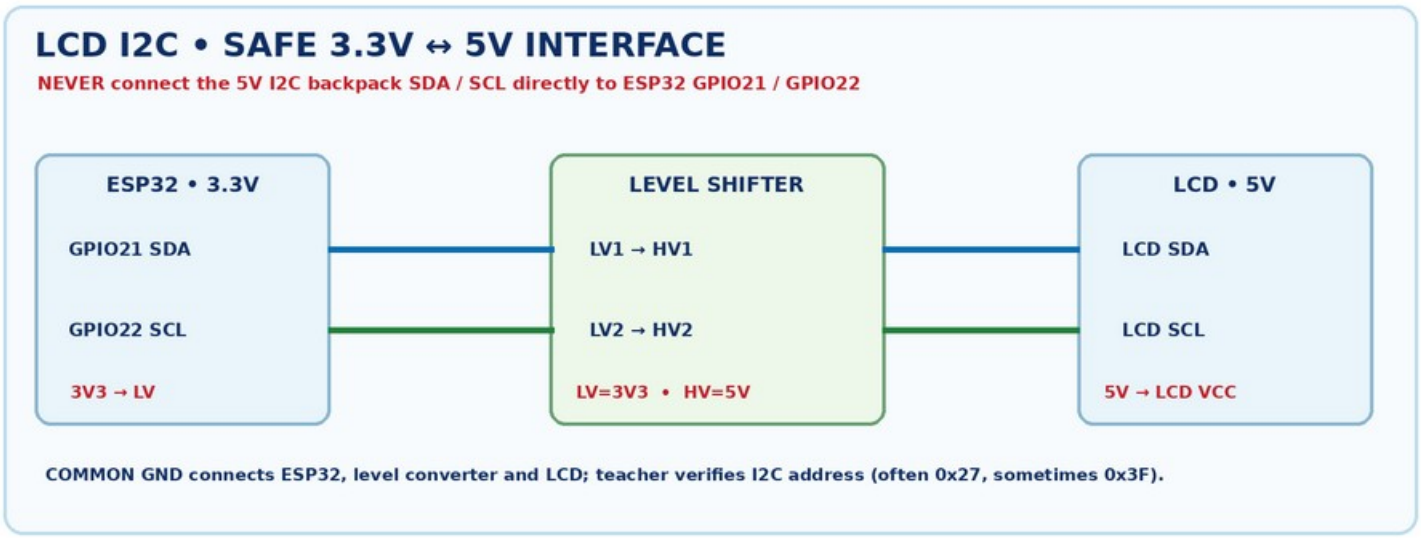
Do not drive an unknown 5V buzzer or a high-current sounder directly from the ESP32. Verify polarity, operating voltage and driver requirements. Keep sound levels comfortable.

🤔 **Why is a short beep more suitable for a school demonstration than a continuous loud alarm?**

_____

Step 4 • Add the Status Display

KEEP ALL THE EARLIER PARTS IN THEIR PLACES



Teacher-verified 4-channel logic level interface

LOW SIDE • ESP32	LEVEL SHIFTER	HIGH SIDE • LCD
GPIO21 • SDA	LV1 ↔ HV1	LCD SDA
GPIO22 • SCL	LV2 ↔ HV2	LCD SCL
3V3 + GND	LV + GND	HV 5V + GND
Approved 5V rail	HV • 5V	LCD VCC • 5V

1	**POWER OFF —** Unplug ESP32 USB and turn off all teacher-approved supplies.
2	**FIND SDA + SCL —** Check the four-pin I2C backpack markings on YOUR LCD.
3	**WIRE LOW SIDE —** GPIO21 → LV1; GPIO22 → LV2; low-side supply LV at 3.3V.
4	**WIRE HIGH SIDE —** HV1 → LCD SDA; HV2 → LCD SCL; high-side HV at 5V.
5	**CHECK GROUND —** Use common ground between ESP32, converter and 5V display.
6	**FIND ADDRESS —** Teacher checks LCD I2C address (commonly 0x27 or 0x3F), updates sketch and runs the status test.

⚠ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Many I2C LCD backpacks pull SDA/SCL up to 5V. NEVER connect such lines straight to ESP32 3.3V pins. Correct level conversion is essential; teacher verifies all rails and labels.

Step 5 • Plan Before We Code

ONE INTEGRATED ARDUINO SKETCH • NOT FOUR DISCONNECTED PROGRAMS

Input → program → output • our command map

PHONE COMMAND	WHAT CHANGES?	WATCH FOR
1 / 0	ROOM LIGHT 1 ON / OFF	Same LED from Activity 1
D / d	Door OPEN / SHUT	Servo arm moves gently
F / f	Model fan ON / OFF	Motor turns / stops
B	Short buzzer alert	One short beep
Browser refresh	Read status only	Door/fan/light + sensor data

New code parts to notice

1	**NEW PIN NAMES —** SERVO_PIN=18; FAN_A=16; FAN_B=17; BUZZER_PIN=25; LCD SDA=21, SCL=22.
2	**OUTPUT FUNCTIONS —** setDoor(open), setFan(on), startBeep(), updateDisplay().
3	**OLD PROGRAM PRESERVED —** Bluetooth Classic, Wi-Fi status page, sensors and ROOM LIGHT 1 stay in the same sketch.
4	**AUTOMATION COMES LATER —** Today the inputs are phone commands. We do not open doors or run fans automatically from sensor readings.

📚 **LIBRARIES NEEDED**

Arduino IDE with original ESP32-WROOM-32 board support; DHT sensor library (and dependency), ESP32Servo, and a compatible LiquidCrystal_I2C library. Library method names may vary: the teacher checks the chosen version.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

This is a classroom starter program, NOT a certified building-control system. Do not use for mains electricity, real doors, emergency alerts or unattended operation.

CODE POINTER | ANNEXURE A5 • Open the complete Activity 5 Arduino sketch at the back of this book. Copy it as ONE sketch, not separate pieces. Teacher checks board, libraries, pin labels and power before uploading.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Do not paste your real school Wi-Fi password into a public or shared file. The teacher enters it locally.

Step 6 • Test One New Feature at a Time

USE THE SAME MODEL • CHANGE ONE OUTPUT ONLY

Safe test protocol

1	**CHECK BEFORE POWER —** Teacher reviews every VCC, GND, motor output, servo signal, buzzer IN and LCD level shifter connection.
2	**UPLOAD COMPLETE SKETCH —** Use Arduino IDE, original ESP32-WROOM-32, correct board/port and 115200 Serial Monitor.
3	**CHECK THE OLD LIGHT —** Send 1 then 0. If it fails, STOP and debug before moving on.
4	**TEST DOOR —** Send D then d; watch for free, slow movement with no hinge strain.
5	**TEST FAN —** Send F, let the guarded motor spin briefly, then send f.
6	**TEST BEEP + SCREEN —** Send B once; see a short beep. Read LCD status and refresh the browser page.
7	**RECHECK SENSORS —** Read LDR, DHT11 and PIR values, and confirm Bluetooth/Wi-Fi still report.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

If anything heats up, stalls, smells unusual or resets unexpectedly, switch OFF external power and unplug USB. Do not continue tests until a teacher checks the circuit.

🤔 My prediction

Which output will need the most power — an LED, a buzzer, a servo or a motor? Explain how you could test this safely.

_____

Our Observation Lab • Change One Thing

RECORD WHAT HAPPENED, NOT WHAT YOU HOPED TO SEE

TEST COMMAND	EXPECTED RESULT	WHAT WE OBSERVED	PASS?
1 then 0	LED ON, then OFF	_____	☐
D then d	Door opens, then shuts	_____	☐
F then f	Motor runs, then stops	_____	☐
B	One short alert	_____	☐
LCD + local Wi-Fi	Latest status is visible	_____	☐
LDR / DHT11 / PIR	Earlier sensors still report	_____	☐

Mini-engineering experiment

Door Test A: angle _____ Door Test B: angle _____

Which setting opens enough without pressing the cardboard hinge? _____

When the motor starts, did the ESP32 restart? ☐ NO ☐ YES Did you stop and check the supply? ☐

💡 **HINT • ONE CHANGE AT A TIME**

If the servo seems blocked, first switch off power and inspect the hinge. Adjust only the safe servo angle, not the wires and door simultaneously.

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Measure only low-voltage circuits using teacher-approved tools. NEVER test mains-powered fans, switches or doors in this school activity.

ACTIVITY 5 • THINK BEFORE ASKING

Debug Detective • Find the Cause

READ THE SYMPTOM → CHECK SAFELY → RETEST

SYMPTOM	INVESTIGATE (POWER OFF FIRST)
Door jitters or does not move	Servo external 5V? Shared GND? Signal GPIO18? Door hinge binding?
Motor moves but door stops	Motor load / supply voltage drop? Did shared supply reset ESP32?
Motor never starts	Driver motor supply? A-IA GPIO16 / A-IB GPIO17? Motor on A-O1/A-O2?
Buzzer silent or stuck on	Is module 3.3V-compatible? Is IN active-HIGH? GPIO25?
LCD lit but blank	I2C address, SDA/SCL direction, level shifter rails, contrast setting?
Bluetooth works but Wi-Fi page missing	Is 2.4 GHz classroom Wi-Fi connected? Check local IP; never public internet.
Sensor values disappear	Reconnect OLD sensor pins (GPIO34, GPIO27, GPIO26); check libraries and power.

What did our team fix?

Our symptom was...

_____

We tested _____ first. Our evidence was...

_____

🌐 **INTERNET SAFETY**

Use only teacher-approved Bluetooth pairing and local Wi-Fi. Do not share network credentials, private addresses or students' names/screenshots publicly.

ACTIVITY 5 • APPLY & REFLECT

🤔 Five Thinking Questions

SAY WHY • USE EVIDENCE • SUGGEST AN IMPROVEMENT

1. Why do we need a separate motor driver and suitable external power rather than connecting the fan motor to an ESP32 GPIO?

_____
_____

2. Your servo door opens halfway and stops. What should your team check before changing the code?

_____
_____

3. How is the information displayed on the LCD different from the action carried out by the servo?

_____
_____

4. If Bluetooth still controls the light but the fan does not run, which two connections or components would you inspect first?

_____
_____

5. Our PIR senses movement but our buzzer does not sound by itself yet. Why is this correct at the end of Activity 5? What rule could we add in Activity 6?

_____
_____

📓 My Engineer's Log • 4CG evidence

CREATE	THINK / TEST	COLLABORATE	COMMUNICATE
I assembled: ____	I debugged: ____	My team role: ____	I explained: ____

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 5 • PROJECT CHECKPOINT

Look! SMARTSPACE Can Now Respond

THE SAME MODEL HAS GROWN • SAVE IT FOR ACTIVITY 6

Our progression checklist

ALREADY BUILT	NEW IN ACTIVITY 5	KEEP FOR ACTIVITY 6
ROOM LIGHT 1 • Bluetooth • local Wi-Fi • LDR • DHT11 • PIR	Servo door • model fan / driver • buzzer • LCD	ESP32 GPIO23/34/27/26 plus new GPIO18/16/17/25 and I2C GPIO21/22

NEXT MISSION • ACTIVITY 6

We will use real sensor readings to create tested IF / THEN automation. The same door, fan, light and buzzer will respond to CONDITIONS, not only phone commands.

🤔 **Which automatic response should we make FIRST, and how will we test its rule before trusting it?**

_____

_____

⚠️ **SAFETY CHECK — STOP • SWITCH OFF • CHECK • ASK**

Leave the model UNPOWERED between sessions; cover exposed electrical connections and store the light door and fan safely. Teacher checks it before the next activity.

ACTIVITY 5 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY

OUR SAME SMARTSPACE — AFTER ACTIVITY 5

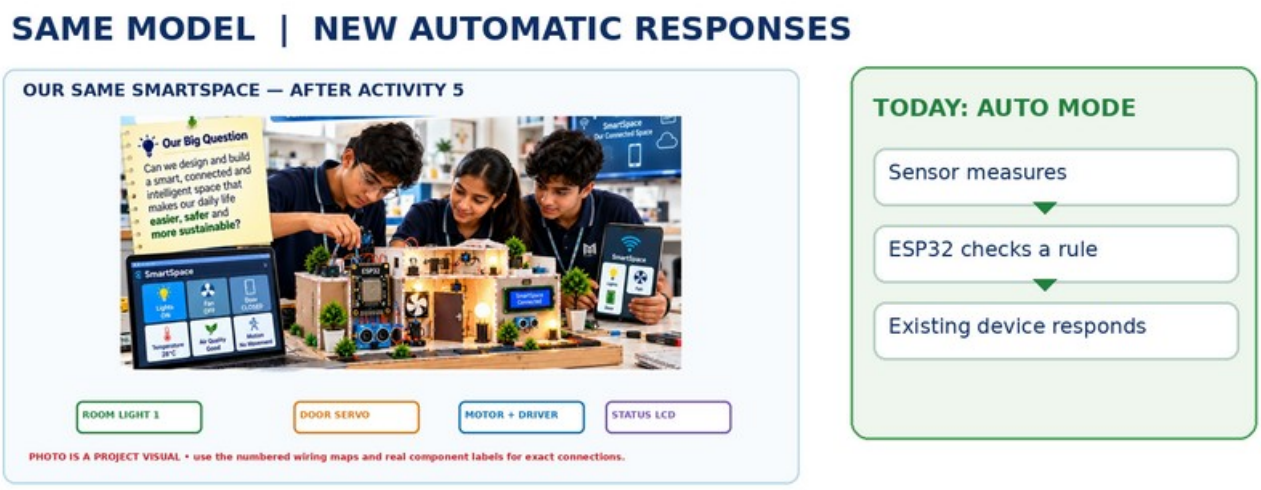


ROOM LIGHT 1 | DOOR SERVO | MOTOR + DRIVER | STATUS LCD

PHOTO IS A PROJECT VISUAL • use the numbered wiring maps and real component labels for exact connections.

Activity 6 • Make SMARTSPACE Automatic

SENSE → DECIDE → ACT • OUR SAME MODEL GETS A NEW ABILITY



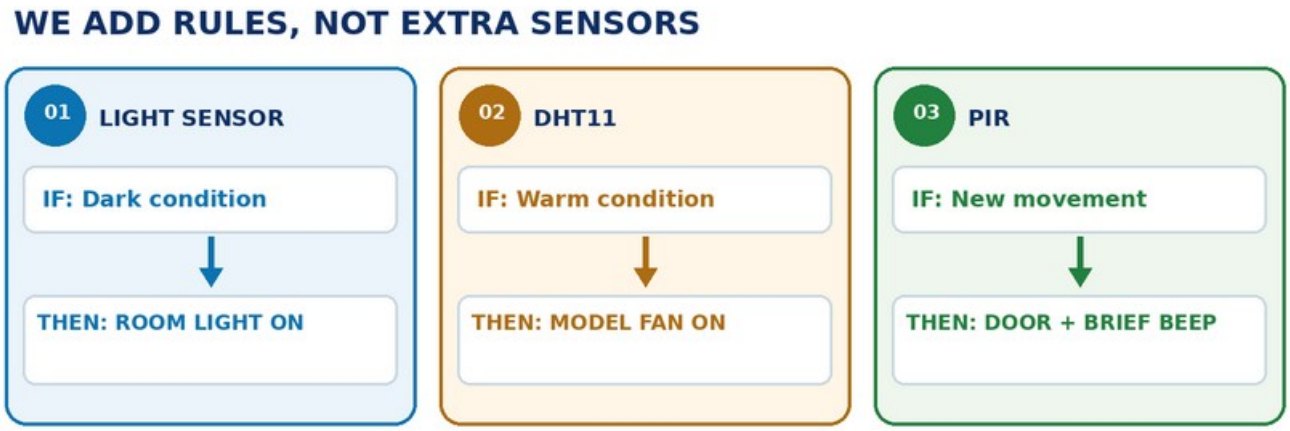
Our mission today

In Activity 5 you could command the light, door, model fan, buzzer and LCD. Today these same parts will respond to carefully tested sensor rules. No new components and no new wires are required.

Keep your existing build

- ☐ ESP32 + ROOM LIGHT 1 + Bluetooth + Wi-Fi still work
- ☐ LDR on GPIO34, DHT11 on GPIO27 and PIR on GPIO26 are already connected
- ☐ Servo door, motor driver, buzzer and I2C LCD remain as in Activity 5

Three new abilities



🤔 **THINK FIRST What is the difference between a sensor noticing movement and the program deciding to open a door?**

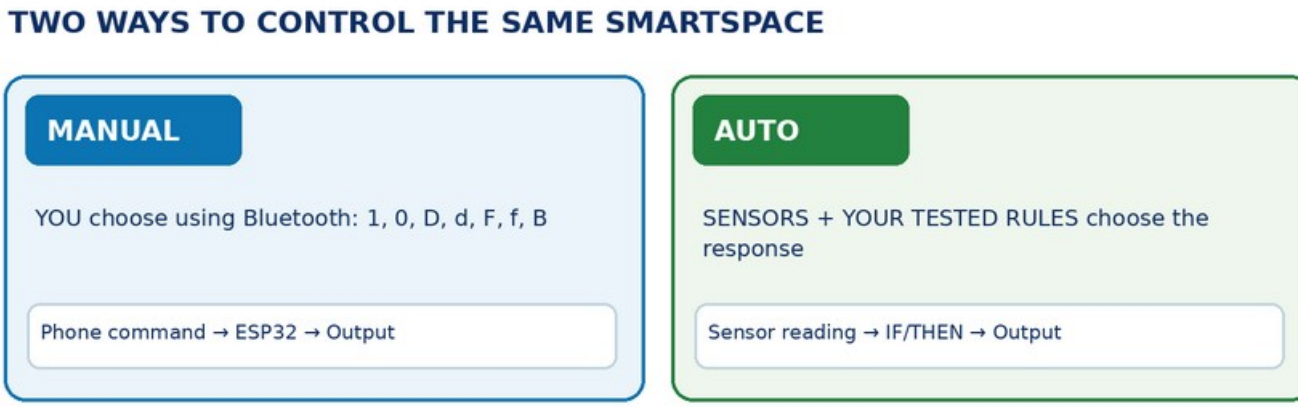
_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
All circuits remain low voltage. Keep moving parts guarded and unplug BOTH USB and external power before changing any connection.

ACTIVITY 6 • THINK BEFORE CODING

Step 1 • Become the SmartSpace Controller

ACT OUT THE RULES WITH YOUR TEAM BEFORE USING ELECTRONICS



Human-role simulation • NO WIRES

1	One student acts as the LDR, another as the ESP32, and another acts as ROOM LIGHT 1.
2	"Sensor" says DARK or BRIGHT. "Controller" applies the agreed IF/THEN rule.
3	"Light" student raises a card when ON and lowers it when OFF. Swap roles.
4	Repeat with DHT11 → model fan and PIR → model door. Use only pretend movements.

Rule-writing grid

SENSOR SAYS...	OUR RULE SAYS...	OUTPUT DOES...
It is dark	IF _____ THEN _____	Light: _____
Temperature is high	IF _____ THEN _____	Fan: _____
A NEW movement occurs	IF _____ THEN _____	Door + beep: _____

🤔 **THINK FIRST Does the machine "learn" this rule, or does it follow rules created by people?**

_____

Step 2 • Calibrate Before Automation

THE NUMBERS IN SAMPLE CODE ARE NOT YOUR CLASSROOM ANSWERS



A. Find YOUR light sensor values

1	Keep the same circuit from Activity 5. Teacher checks power and restarts the model.
2	Open Serial Monitor at 115200 baud. Read LDR ADC with the room well lit.
3	Shade the light sensor gently without touching its wires. Read again.
4	Write both readings. Decide whether a smaller number or larger number means “darker” on YOUR module.

Condition	LDR ADC reading	My observation
Bright room	_____	_____
Partially shaded	_____	_____
Covered / dark	_____	_____

B. Check room temperature

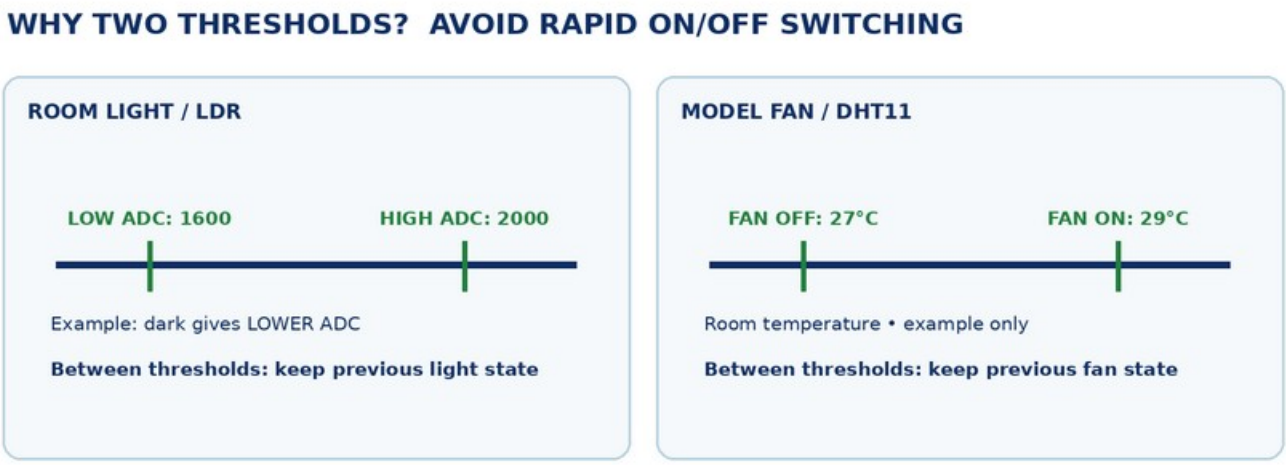
DHT11 temperature now: _____ °C Our safe demo ON setting: _____ °C OFF setting: _____ °C

💡 PICTURE HINT Choose two limits with a gap. Do not use hairdryers, heaters or hot surfaces to force temperature changes.

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Do NOT use a heat source near the sensor. To demonstrate without heating, the teacher can adjust the code thresholds around the current room temperature; write those chosen values in your log.

Step 3 • Let Darkness Control ROOM LIGHT 1

KEEP THE SAME LED AND GPIO23 CONNECTION



Why does our rule use ON and OFF limits?

Light readings can change slightly even if the room looks the same. Two limits create a "quiet zone" so the light does not flicker ON/OFF when the sensor reading hovers near one number.

Check your rule

1	Use your page 73 readings to decide whether dark means LOWER or HIGHER LDR ADC.
2	Teacher updates DARK_WHEN_LOW and the two light thresholds in the Activity 6 program.
3	Upload the integrated sketch, enter AUTO mode using the Bluetooth command A.
4	Shade the existing LDR. Observe ROOM LIGHT 1. Unshade it and observe again.
5	Write what happened. Try MANUAL mode (M), then command light ON (1) and OFF (0).

Condition	Prediction	Actually observed
Bright	Light: _____	Light: _____
Dark / shaded	Light: _____	Light: _____
Return bright	Light: _____	Light: _____

🤔 **THINK FIRST If the light flickers when you move your hand, should you change a wire first or improve your tested thresholds?**

_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Never rewire the LDR while powered. Keep the circuit unchanged; modify only the indicated software values with teacher guidance.

Step 4 • Let Temperature Control the Model Fan

THE SAME DHT11 • THE SAME MOTOR DRIVER • NO NEW FAN WIRING

Our fan rule

IF temperature reaches the high limit → FAN ON. IF temperature drops to the low limit → FAN OFF. Between those limits, remember the last fan state.

Example setting	Value	What it means
FAN ON threshold	29°C	At/above this value, turn fan ON
FAN OFF threshold	27°C	At/below this value, turn fan OFF
Between 27°C and 29°C	No switch	Keep the previous state

Test without heating

1	Read the classroom temperature on the existing LCD or Serial Monitor.
2	Teacher chooses a safe demonstration pair around your actual reading.
3	Write the two values and upload the sketch. Send A to enter AUTO.
4	Observe whether the fan is ON/OFF; adjust only one threshold at a time after switching back to MANUAL.
5	Restore useful classroom settings after the demonstration and record them.

Trial	Temp °C	Fan ON limit	Fan OFF limit	Fan result
1	_____	_____	_____	_____
2	_____	_____	_____	_____
3	_____	_____	_____	_____

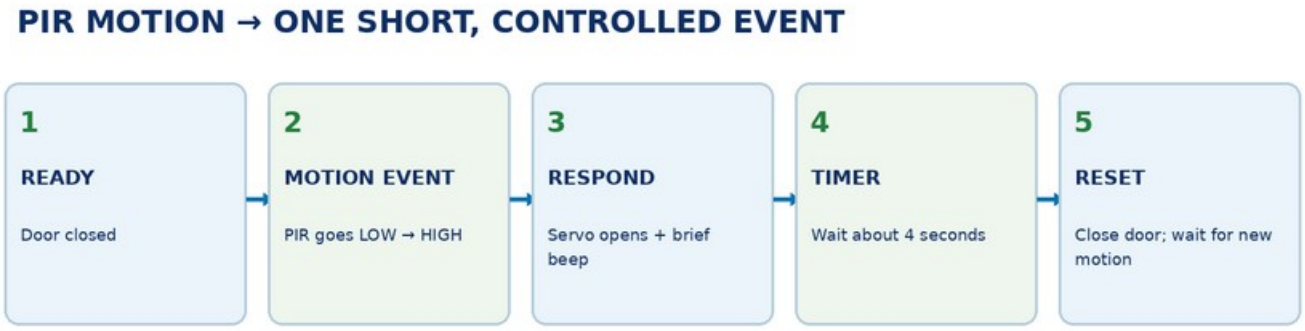
💡 PICTURE HINT A sensor measures conditions; a rule written by people makes the fan decision.

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Do not touch rotating blades. Use only a guarded, lightweight fan or a safely coupled motor indicator; the motor must remain on its driver and external supply.

ACTIVITY 6 • THIRD AUTOMATIC RESPONSE

Step 5 • Let Motion Trigger a Safe Door Demo

MOTION EVENT → BRIEF BEEP → MODEL DOOR OPENS → CLOSES



Build on the same PIR + servo + buzzer circuit

1	Confirm the servo arm moves freely and the door does not jam. Keep fingers away.
2	Wait for PIR stabilization after power-up; the module may report movement while settling.
3	Use Bluetooth A to enter AUTO mode. Step away, then move into the PIR viewing area.
4	Watch the door open and listen for the short beep. Wait for the door to close.
5	Check whether the system waits for a NEW movement event before opening again.

Our test	Prediction	Observation
Walk through PIR area	Door: _____	Door: _____
Wait about 4 s	Door: _____	Door: _____
Motion stays HIGH	Repeat? _____	Repeat? _____

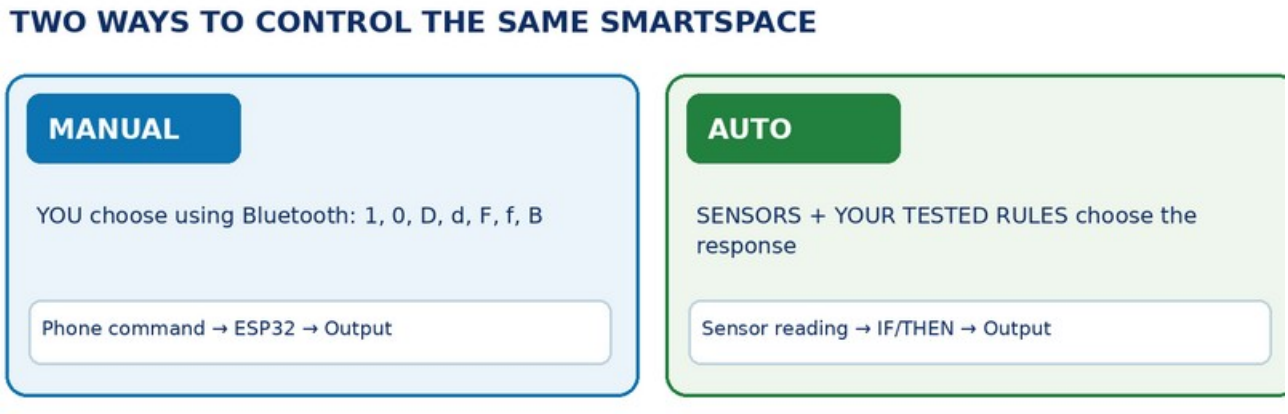
🤔 **THINK FIRST Why should the door close by a timer rather than stay open until someone moves again?**

_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
This is a SCHOOL MODEL, not an emergency-door or security system. PIR reports movement, not identity or guaranteed occupancy. Teacher must supervise all moving parts.

Step 6 • Switch Between MANUAL and AUTO

KEEP BLUETOOTH + WI-FI FROM ACTIVITIES 2 AND 3



Bluetooth command guide • one character at a time

Command	Purpose	What student checks
A	Enter AUTO mode	Rules now control outputs
M	Enter MANUAL mode	Outputs safely reset; manual control resumes
1 / 0	Light ON / OFF	MANUAL mode only
D / d	Door OPEN / SHUT	MANUAL mode only
F / f	Fan ON / OFF	MANUAL mode only
B	Short test beep	MANUAL mode only
S	Report current mode	Both modes

Test your dashboard and LCD

Refresh the Activity 3 local browser status page using teacher-approved Wi-Fi. It now displays MODE, light, fan, door and sensor readings. The LCD displays the mode and status too.

🤔 **THINK FIRST Why is a local status webpage different from publishing your SMARTSPACE on the open internet?**

_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Keep passwords out of the workbook and screenshots. Use a classroom-approved Wi-Fi network. Never expose the device or its control page to the public internet.

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Never modify wires while USB or external power is ON. Teacher verifies wiring from Activity 5 before the new upload.

CODE POINTER | ANNEXURE A6 • Open the complete Activity 6 Arduino sketch at the back of this book. Copy it as ONE sketch, not separate pieces. Teacher checks board, libraries, pin labels and power before uploading.

→ → READ → PREDICT → MODIFY RUN OBSERVE
→ → READ → PREDICT → MODIFY RUN OBSERVE
→ → READ → PREDICT → MODIFY RUN OBSERVE
→ → READ → PREDICT → MODIFY RUN OBSERVE

🔎 READ → PREDICT → MODIFY → RUN → OBSERVE → DEBUG

ACTIVITY 6 • TEST EACH RULE SAFELY

Step 8 • Test, Record and Compare

MANUAL FIRST → AUTO NEXT → ONE CHANGE AT A TIME

Our full progressive test route

1	With power OFF, check all connections against Activity 5. Teacher checks servo, fan driver and 5V/3.3V interface.
2	Power on; allow the PIR and DHT11 to settle. Test Bluetooth M and one safe manual command.
3	Check LCD and local Wi-Fi page. Existing sensors must show reasonable readings.
4	Enter AUTO (A). Test light rule, then fan rule, then motion/door rule in turn.
5	After each test, record the result. If wrong: send M, stop motion, power OFF before checking wiring.

Feature	Expected	What happened?	PASS?
Bluetooth / MANUAL	Commands work	_____	☐
Wi-Fi status	Local page opens	_____	☐
AUTO dark light	Light responds	_____	☐
AUTO temperature	Fan responds	_____	☐
AUTO motion	Door + short beep	_____	☐
LCD status	Mode + readings	_____	☐

Our debugging decision

No response? Check mode → read Serial Monitor → verify sensor reading → check your thresholds → check output supply (POWER OFF). Change ONE thing, then retest.

⚠ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
If the ESP32 resets when a motor starts, switch OFF and ask the teacher to inspect the external supply and shared ground. Do not keep retrying a faulty power connection.

ACTIVITY 6 • THINK • EXPLAIN • IMPROVE

🤔 Five Thinking Questions

USE EVIDENCE FROM YOUR OWN WORKING SMARTSPACE

1. The LDR reading changes from 1750 to 1800 near the chosen limit. Why could one threshold make the light flicker, and how do two thresholds help?

_____
_____

2. You send F to start the fan, but AUTO mode is active. What should happen, and why?

_____
_____

3. A student waves continuously in front of the PIR. Why might the model door not open again immediately after closing?

_____
_____

4. The DHT11 reads 28°C; your fan settings are ON at 29°C and OFF at 27°C. Should the fan change state? Explain.

_____
_____

5. Is SMARTSPACE using AI when it follows your IF/THEN rules? Explain the difference and suggest one way a later AI activity could learn from data.

_____
_____

📓 My Engineer's Log + 4CG evidence

What we built	What we tested	What we improved
_____	_____	_____
_____	_____	_____

4CG evidence: CG-1 ☐ CG-2 ☐ CG-3 ☐ CG-4 ☐ | 4C skills: Think • Create • Collaborate • Communicate

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 6 • PROJECT CHECKPOINT

Look! SMARTSPACE Can Now Decide by Rules

SAME MODEL • SIX ACTIVITIES COMPLETED • READY FOR THE IoT DASHBOARD

What is working now?

- ☐ Activity 1 • ESP32 and ROOM LIGHT 1
- ☐ Activity 2 • Local Bluetooth manual commands
- ☐ Activity 3 • Wi-Fi connection and browser status
- ☐ Activity 4 • Light, motion, temperature and humidity sensors
- ☐ Activity 5 • Servo door, model fan, buzzer and LCD
- ☐ Activity 6 • MANUAL / AUTO with three tested IF/THEN responses

Plan for Activity 7 — Our IoT Control Centre

The browser currently shows status only. Next, we will make a simple local control page so a supervised user can see SMARTSPACE values and request actions through the same network.

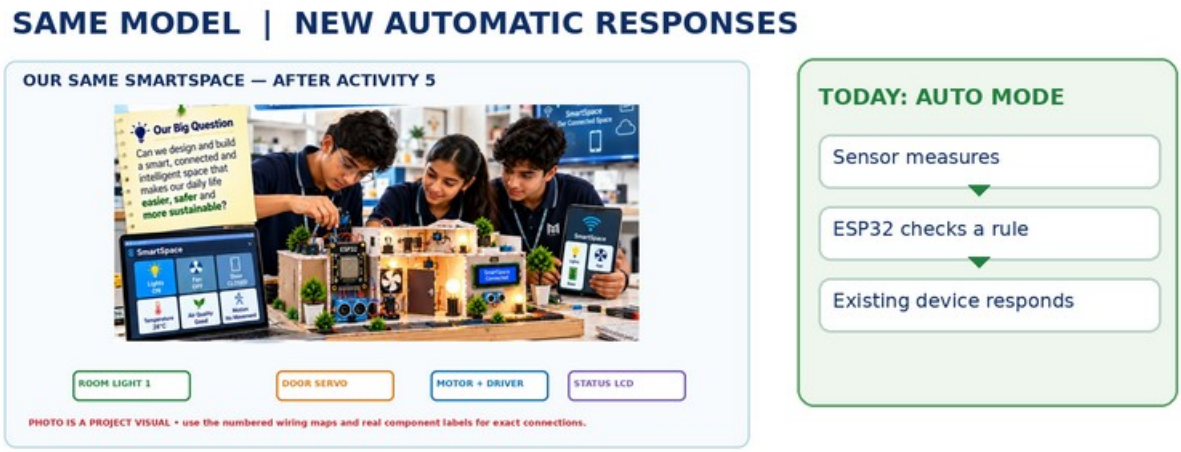
🤔 **THINK FIRST Which controls should appear on the browser dashboard? Which safety controls should stay physical or with the teacher?**

_____

_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Save your current integrated Arduino sketch and lab log. Leave the same model assembled; switch off USB and external motor power before storing.

ACTIVITY 6 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY



Activity 7 • Build the IoT Control Centre

SAME SMARTSPACE • SAME WIRES • ONE NEW WEB CONTROL FEATURE



Our mission today

Until Activity 6, the browser showed status while Bluetooth could send commands. Today our classroom browser becomes a LOCAL control centre. It will display live sensor values and offer safe MANUAL controls without removing any earlier feature.



What stays exactly the same?

- ☐ ESP32, ROOM LIGHT 1, LDR, DHT11 and PIR
- ☐ Servo door, model fan, buzzer and LCD
- ☐ Bluetooth commands and tested automatic IF/THEN rules

🤔 **THINK FIRST What is different between a page that only shows data and one that also sends a command?**

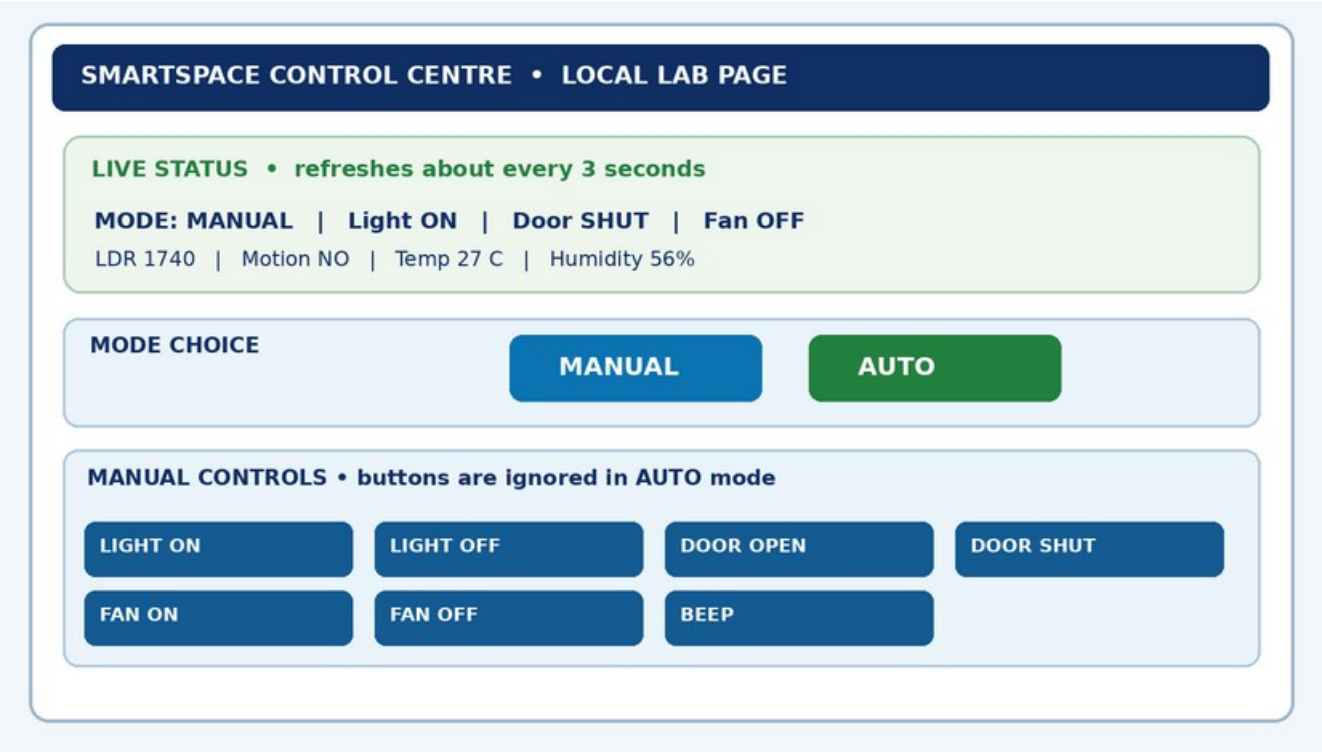
_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
No new wiring is needed. Keep the existing model assembled. The classroom browser must stay on a teacher-approved LOCAL network, never public internet.

ACTIVITY 7 • DESIGN BEFORE CODE

Step 1 • Plan Your SMARTSPACE Dashboard

LOOK → PLAN → CHOOSE WHICH ACTIONS ARE SAFE



Your design challenge

1	Point to the STATUS area: which sensor and output readings should be visible?
2	Find the MODE buttons: why might MANUAL and AUTO need different rules?
3	Find the manual commands: which should stop working when AUTO is active?
4	Draw one improvement on a paper dashboard before opening the real page.

Our dashboard needs...	**Our team's idea**
Three live readings	_____
Two safe operating modes	_____
One important warning	_____

🤔 **THINK FIRST Should students be able to bypass the teacher's power-off check using a browser button? Explain.**

_____

Step 2 • Keep Every Existing Connection

NO NEW SENSOR • NO EXTRA Wi-Fi MODULE • CHECK WHAT IS ALREADY BUILT

DO NOT REWIRE • THESE CONNECTIONS STAY AS BEFORE

ROOM LIGHT 1 Activity 1	GPIO23	Servo door Activity 5	GPIO18
LDR Activity 4	GPIO34 (ADC1)	Fan driver inputs Activity 5	GPIO16 / GPIO17
DHT11 Activity 4	GPIO27	Buzzer Activity 5	GPIO25
PIR Activity 4	GPIO26	LCD I2C Activity 5	GPIO21 SDA / GPIO22 SCL

Before power is connected

1	Switch OFF USB and external motor/servo power; inspect all existing leads.
2	Check GPIO numbers against the teacher-verified Activity 6 build; do not move any wires.
3	Confirm the external motor/servo supply, suitable logic-level connections and shared ground where required.
4	Teacher signs the checklist; only then reconnect power for the software upgrade.

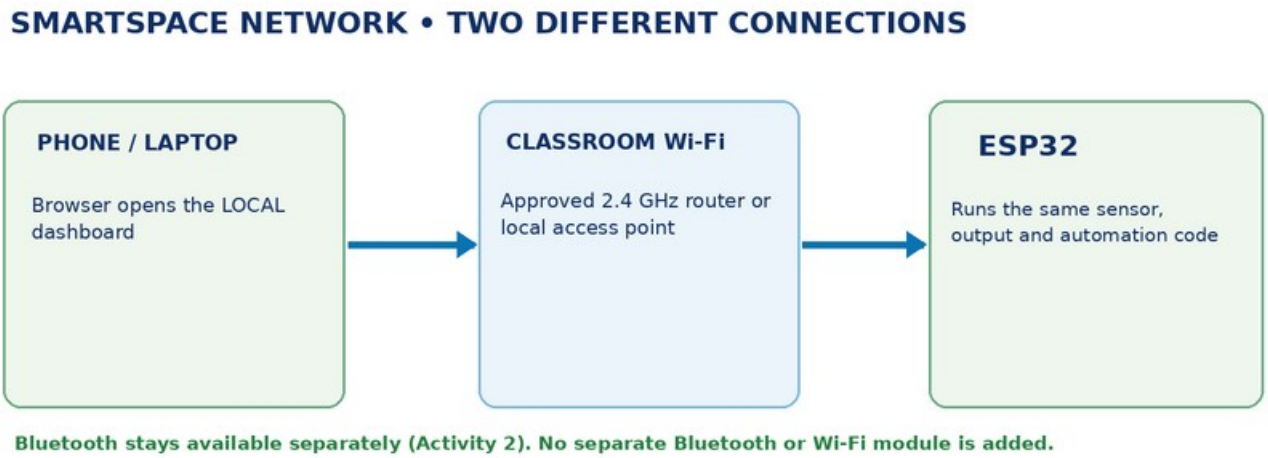
Teacher power-off verification: _____ **Date:** _____

⚠ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
ESP32 GPIO is 3.3 V logic. Never feed 5 V signals directly to GPIO. Do not power the servo or motor from an ESP32 GPIO pin. Use the existing protected interfaces.

ACTIVITY 7 • CONNECTION MAP

Step 3 • How Does the Browser Reach SMARTSPACE?

LOCAL 2.4 GHz Wi-Fi • NOT THE PUBLIC INTERNET



Follow the arrows

1	Teacher verifies a permitted, isolated classroom 2.4 GHz Wi-Fi network.
2	ESP32 joins that network using settings prepared privately by the teacher.
3	Laptop or phone joins the SAME approved local network.
4	Browser opens the ESP32 local address printed in Serial Monitor.
5	Bluetooth remains a separate nearby connection; no hardware is removed.

Connection	Already used for	New in Activity 7
Bluetooth	Manual commands from Activity 2	Still works
Wi-Fi	Local status from Activity 3	Live dashboard + supervised controls

🤔 **THINK FIRST If the classroom has Wi-Fi but no internet connection, can a local dashboard still work? Why?**

_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Never share Wi-Fi passwords or dashboard login details in the workbook, screenshots or public messaging. Do not configure port forwarding.

Step 4 • Upgrade Our Integrated ESP32 Sketch

📎 CODE POINTER | See Annexure A7: Arduino reference code for teacher review. Do not type a long sketch from this activity page.

KEEP ALL PREVIOUS FEATURES IN ONE ARDUINO IDE FILE

Today we change software, not the circuit

1	Open the saved Activity 6 sketch; use Save As to create “SMARTSPACE_Activity7”.
2	Open the supplied Activity 7 integrated .ino in Arduino IDE and compare the unchanged GPIO pins.
3	Teacher enters network details and changes the temporary dashboard username/password privately.
4	Check that Bluetooth, Wi-Fi, LDR, DHT11, PIR, servo, fan, buzzer and LCD code still exists.
5	Select the correct ESP32 board/COM port. Teacher reviews, then upload and open Serial Monitor at 115200.

Spot the three NEW webpage routes in the program

Page/route	What it does	Student observation
/	Shows HTML dashboard	Open in browser
/status	Returns live sensor + output values	Read updated status
/do	Receives an approved POST command	Test in MANUAL mode

Small code window • Read, do not memorise

A route in our program	Its job
web.on("/", HTTP_GET, dashboardPage);	Shows the dashboard
web.on("/status", HTTP_GET, dashboardStatus);	Supplies live readings
web.on("/do", HTTP_POST, dashboardAction);	Processes buttons

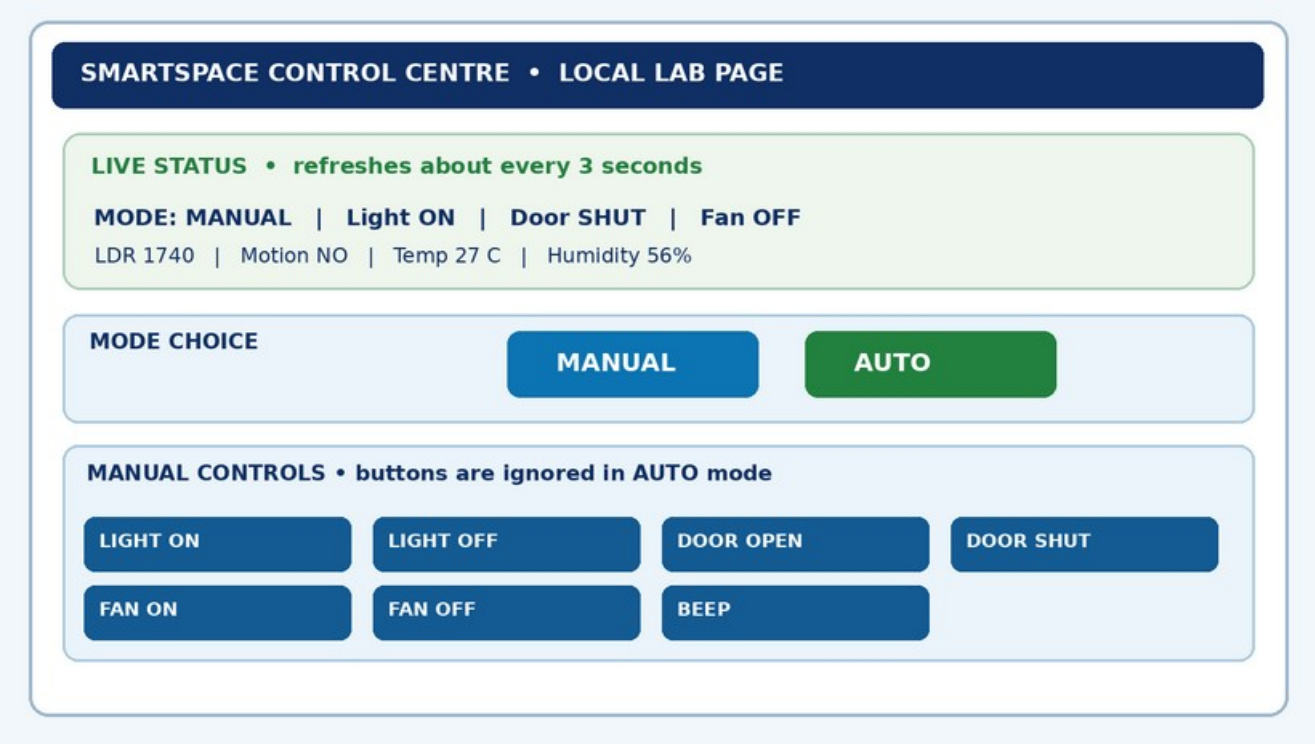
💡 PICTURE HINT The complete reference sketch is printed in Annexure A7. Do not copy only this table; the board needs the complete integrated program.

⚠️ SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK
The example uses local HTTP with Basic Authentication, which does NOT encrypt passwords. Use it only on a private isolated classroom network, under teacher supervision—not for real building security.

ACTIVITY 7 • OPEN THE LOCAL PAGE

Step 5 • Find and Open Your Dashboard

FIRST TEST: SAME NETWORK • LOCAL IP ADDRESS • TEACHER LOGIN



Follow each step carefully

1	After upload, teacher checks the Serial Monitor for “Local dashboard: http://...” and writes the local address on a classroom card.
2	Connect an approved laptop/phone to the same classroom Wi-Fi. Avoid public or guest networks that isolate devices.
3	Type the local address into the browser; a login prompt may appear. Teacher completes it privately.
4	Observe MODE, light, fan, door, LDR, motion, temperature and humidity.
5	Wait about 3 seconds. Confirm at least one reading updates without rebuilding the circuit.

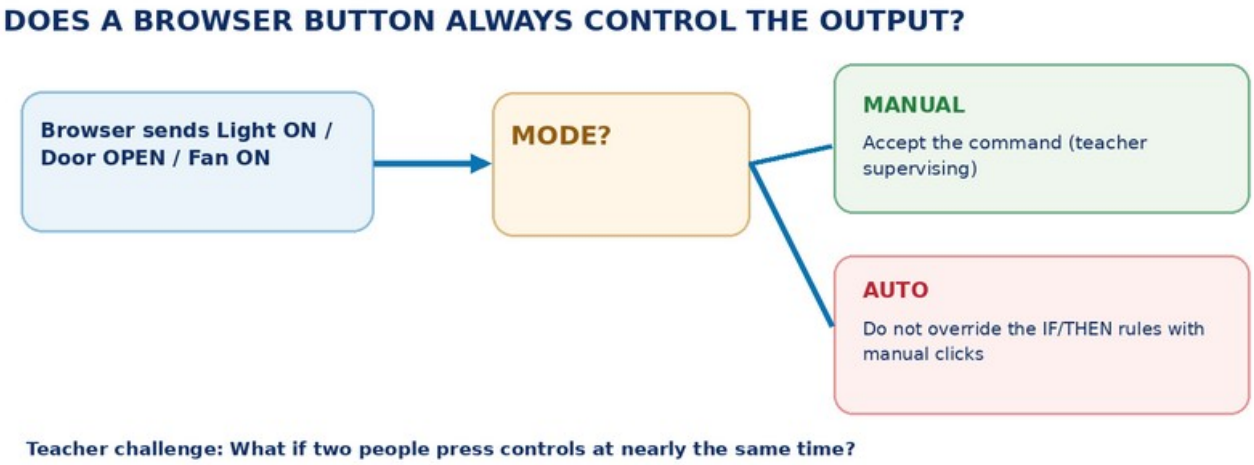
Watch this value	First reading	Next reading	What changed?
LDR	_____	_____	_____
Temperature (°C)	_____	_____	_____
Light state	_____	_____	_____

🤔 **THINK FIRST Is an IP address the same as a Wi-Fi password? Why must we treat them differently?**

_____

Step 6 • Try Manual Buttons One at a Time

MODE DECIDES WHETHER A BUTTON MAY CONTROL A DEVICE



Test the existing outputs — not a new model

1	Select MANUAL on the browser; verify the LCD or Bluetooth S message confirms MANUAL.
2	Use Light ON, then Light OFF. Compare both the physical ROOM LIGHT 1 and the webpage.
3	With the door mechanism clear, press OPEN then SHUT while observing the servo.
4	Test the safely guarded model fan ON/OFF and one short buzzer beep.
5	Select AUTO. Try Light ON; the page should NOT override the automatic rule.

Manual action	Predicted result	Actually observed
Light ON/OFF	_____	_____
Door OPEN/SHUT	_____	_____
Model fan ON/OFF	_____	_____
AUTO: click Light ON	_____	_____

⚠ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Keep fingers away from moving parts. The web dashboard is not an emergency stop. Physical power isolation and teacher supervision are required.

Step 7 • Watch AUTO Rules on the Same Dashboard

OBSERVE SENSOR → RULE → OUTPUT → LIVE STATUS

Predict what the dashboard will report

1	From the browser, choose AUTO and wait for sensor readings to settle.
2	Gently shade the existing LDR without touching wires. Check the LED and LDR number.
3	Unshade it; observe whether the two-threshold rule changes the LED.
4	Observe room temperature and model fan according to your Activity 6 thresholds.
5	Walk briefly through the PIR area; watch door/alert, then view the new motion status.

Condition	Sensor value	Actual output	Dashboard updates?
Bright	LDR _____	Light _____	☐ YES ☐ NO
Shaded	LDR _____	Light _____	☐ YES ☐ NO
Normal room	Temp ____°C	Fan _____	☐ YES ☐ NO
Motion event	PIR _____	Door _____	☐ YES ☐ NO

Record one difference between reading and action

A sensor gives a reading. The program applies a rule. The dashboard shows both the reading and the present output state.

🤔 **THINK FIRST Why can the light remain ON when the sensor reading falls between two thresholds?**

_____

_____

Step 8 • When the Dashboard Does Not Work

CHECK ONE CAUSE AT A TIME • KEEP ALL POWER RULES



What happened?	First safe check	Next step
Local page does not open	Check ESP32 IP and classroom Wi-Fi	Check both devices on same network
Login repeatedly requested	Ask teacher to verify classroom credentials	Do not share or guess passwords
Readings stay unchanged	Check Serial Monitor and sensor updates	Wait for DHT/PIR settling
Button does nothing	Check MANUAL/AUTO mode	Test Bluetooth to isolate the issue
ESP32 resets when motor starts	STOP and disconnect power	Teacher checks external supply and ground

If there is no classroom Wi-Fi

The teacher can demonstrate using an isolated 2.4 GHz classroom access point. Otherwise, use the printed mock dashboard to plan the interface and return to the live test when approved local connectivity is available.

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
Do NOT connect the ESP32 to a public hotspot or expose its control page outside the classroom. HTTP Basic authentication is not encrypted. Students must not enter personal information.

ACTIVITY 7 • THINK • RECORD • EXPLAIN

🤔 Five Thinking Questions + Engineer's Log

USE OBSERVATIONS FROM YOUR OWN SMARTSPACE, NOT GUESSES

1. How is the new browser dashboard different from the status-only page we made in Activity 3?

_____

_____

2. Why should Light ON, Door OPEN and Fan ON be blocked when AUTO mode is active?

_____

_____

3. The dashboard shows an LDR reading but the light does not switch. Which TWO checks would you make before moving any wire?

_____

_____

4. If the ESP32 and laptop use the same Wi-Fi but there is no internet connection, what should still work? Explain.

_____

_____

5. Which two changes would you make to improve safety and privacy before showing SMARTSPACE at a school exhibition?

_____

_____

My Engineer's Log

We kept working	We added	We tested / improved
_____	_____	_____
_____	_____	_____

4CG evidence: CG-1 ☐ CG-2 ☐ CG-3 ☐ CG-4 ☐ | 4C skills: Think • Create • Collaborate • Communicate

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 7 • PROJECT CHECKPOINT

Look! Our SMARTSPACE Has a Control Centre

SAME PROJECT • SEVEN ACTIVITIES COMPLETE • READY TO REFINE MODES

Check what your team has now

- ☐ Activities 1–2: ESP32 + working light + Bluetooth
- ☐ Activities 3–4: Classroom Wi-Fi + LDR / DHT11 / PIR sensors
- ☐ Activity 5: Servo door + guarded model fan + buzzer + LCD
- ☐ Activity 6: MANUAL / AUTO tested rules
- ☐ Activity 7: LOCAL browser dashboard with live status and supervised buttons

What will we improve next? • Activity 8

We will refine the difference between manual, remote and automatic operation. Students will test priority rules and decide what should happen when two control requests disagree.

🤔 **THINK FIRST Which mode should have priority if there is an unsafe situation or a teacher needs to stop the model?**

_____

_____

⚠️ **SAFETY CHECK • STOP → SWITCH OFF → CHECK → ASK**
After testing, use MANUAL and switch the outputs OFF. Disconnect both USB and external supply before storage. Keep the same SMARTSPACE model intact for Activity 8.

ACTIVITY 7 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY

SMARTSPACE • ACTIVITY 7 COMPLETE

OUR SAME SMARTSPACE — AFTER ACTIVITY 5



ROOM LIGHT 1 | DOOR SERVO | MOTOR + DRIVER | STATUS LCD

PHOTO IS A PROJECT VISUAL • use the numbered wiring maps and real component labels for exact connections.

NOW IT CAN...

- Sense and show readings
- Control through Bluetooth
- Connect through Wi-Fi
- Run MANUAL/AUTO rules
- Show a local browser dashboard

ACTIVITY 8 • MISSION LAUNCH

Three Ways to Control SMARTSPACE

LOCAL • REMOTE • AUTO | Same ESP32 • Same model • New control choices



Our mission today

Your SMARTSPACE already senses, connects and responds. Now we will add one LOCAL button and make its three modes easy to compare. We will not remove the light, fan, servo, LCD, sensors, Bluetooth or Wi-Fi.

Project progression Activity 7 had REMOTE commands and AUTO rules. Activity 8 adds a physical button and clearer LOCAL / REMOTE / AUTO control priority.

Session 1 • 45 minutes	Session 2 • 45 minutes
Explore modes • 10 min Power-off wiring • 20 min Teacher check • 15 min	Upload + test • 15 min Conflict challenge • 15 min Think, log, tidy • 15 min

🤔 **THINK If you can switch the light from a phone, why might a real button still be useful?**

_____

⚠️ **SAFETY • RED CHECK** Keep the model unpowered while touching the button wiring. The physical power switch belongs to the teacher and always comes before software control.

ACTIVITY 8 • WONDER & PLAN

Who Should Control Our SMARTSPACE?

Look • Predict • Discuss • Choose



Try human logic before writing code

Work in teams of three. One learner acts as the sensor, one as the phone, and one as the person next to the model. Only the chosen mode may decide what the output should do.

Mode	Who makes the decision?	What should happen?
LOCAL	Person next to SMARTSPACE	Press the new button to control ROOM LIGHT 1.
REMOTE	Person using Bluetooth or Wi-Fi	Commands may operate the outputs.
AUTO	ESP32 program using sensors	IF / THEN rules respond without direct commands.

Prediction board

Situation	My prediction
Classroom Wi-Fi is OFF. Someone presses LOCAL.	_____
AUTO is active. Someone taps Fan ON.	_____
REMOTE is active. An LDR reading changes.	_____

🤔 **THINK Which method can still work when classroom Wi-Fi is unavailable? Why?**

_____

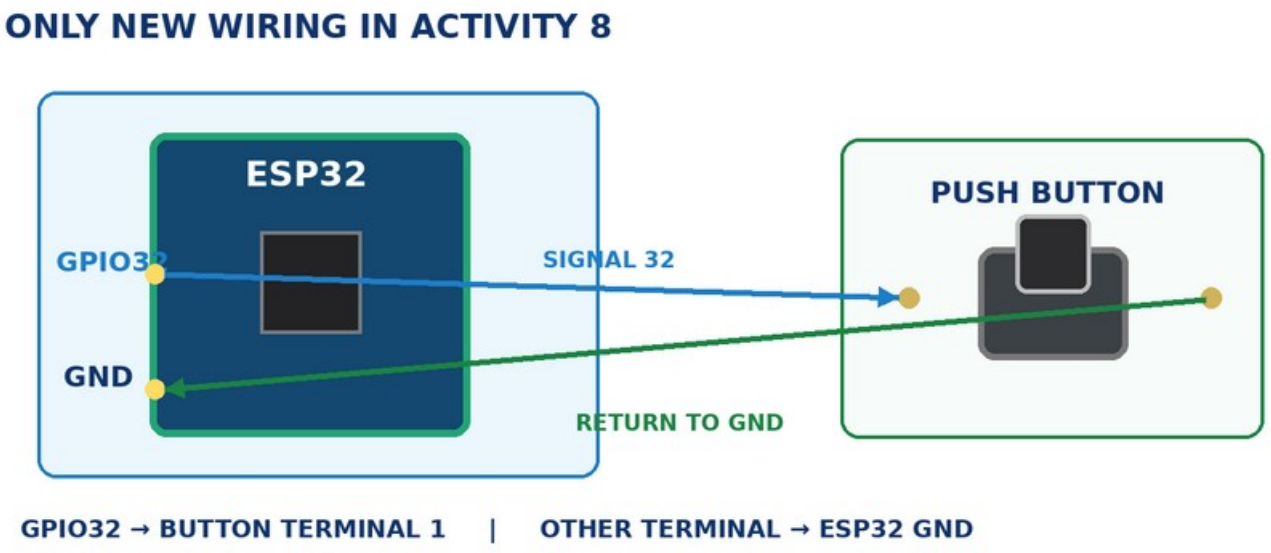
ACTIVITY 8 • BUILD STEP 1

Add the LOCAL Push Button

Only ONE new input • Keep all previous connections

Today we need

1 push button from the Class 6 kit • 2 jumper wires • existing ESP32 + breadboard • checked classroom power arrangement.



Connection guide: one terminal to GPIO32, the other to ESP32 GND. No external voltage on this button.

Step	What your team does
1	Switch OFF USB and external motor/servo power. Ask a teacher to confirm both are disconnected.
2	Place the button on the accessible wall/panel of the SAME SMARTSPACE model.
3	Read the actual button legs. On a 4-leg tactile button, the teacher checks which two terminals are switched.
4	Connect the chosen terminal to ESP32 GPIO32 and the other to an existing GND point.
5	Check that the wire cannot touch 3V3, 5V, motor or servo power. Secure it neatly.
6	Teacher verifies the actual pins and the existing circuits before any power returns.

⚠️ **SAFETY • RED CHECK** Do not connect a push button directly between 3V3 and GND. Keep USB and external power OFF when moving wires. ESP32 GPIO is 3.3V logic.

🤔 **THINK Why do we connect a simple push button to GND rather than to an external 5V supply?**

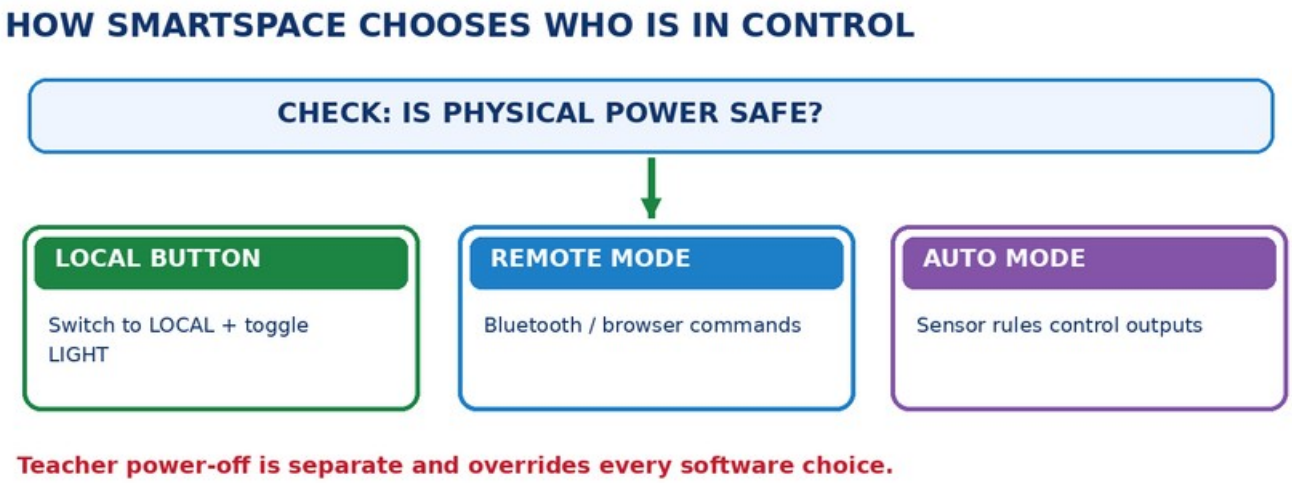
_____

ACTIVITY 8 • CODE STEP 2

Upgrade Our Existing Program

📎 CODE POINTER | See Annexure A8: Arduino reference code for teacher review. Do not type a long sketch from this activity page.

One program keeps Activities 1–7 working



The three important ideas in our updated code

New idea	Code you can recognise	Meaning
Input	pinMode(32, INPUT_PULLUP);	A press reads LOW safely.
Local press	handleLocalButton();	Read the button repeatedly.
Mode selection	LOCAL / REMOTE / AUTO	Choose one source of output control.
AUTO rule	if (controlMode == MODE_AUTO)	Apply sensor rules in AUTO only.

Follow these steps in Arduino IDE

→ Open your Activity 7 sketch (do not delete your saved backup).

→ Open the integrated sketch in Annexure A8. Compare the changed lines with the table above.

→ Verify ESP32 board selection and required libraries. Do NOT type classroom Wi-Fi credentials into your workbook.

→ Click Verify; discuss any error message before clicking Upload.

→ With the teacher's permission, reconnect USB. Observe the Serial Monitor (115200 baud).

HINT • Look for a change A mode switch first closes the model door, stops the motor and switches the light OFF. Then the chosen mode starts from a predictable state.

⚠️ **SAFETY • RED CHECK** Upload only after an unpowered wiring inspection. A working program is not a substitute for turning OFF the actual power when unsafe.

🤔 **THINK What do you predict will happen to the fan when the mode changes from AUTO to LOCAL?**

_____

ACTIVITY 8 • TEST STEP 3

Test LOCAL and REMOTE

Predict → Choose mode → Try → Record

A • Physical button test

1 Reconnect the checked model and open the Serial Monitor. Start in LOCAL.
2 Press and release the new button once. Observe ROOM LIGHT 1.
3 Press and release again. Observe whether the light toggles.
4 Select AUTO on the approved dashboard, then press the LOCAL button once. Which mode is now shown?

B • Remote test

1 Select REMOTE using the dashboard or by sending R via the paired Bluetooth phone.
2 Tap Light ON and Light OFF; try one door command with hands clear.
3 Return to LOCAL. Try the same remote Light ON command. Observe whether it is blocked.

Test	Predict	Observed	Mode displayed
LOCAL: first press	_____	_____	_____
LOCAL: second press	_____	_____	_____
REMOTE: Light ON	_____	_____	_____
LOCAL: remote command	_____	_____	_____

HINT • Testing fairly Change just ONE control source at a time. Confirm the mode shown on the LCD or dashboard before judging a command.

⚠️ **SAFETY • RED CHECK** Keep hands clear of the servo door and fan. If a motor behaves unexpectedly, ask the teacher to switch OFF its supply before changing anything.

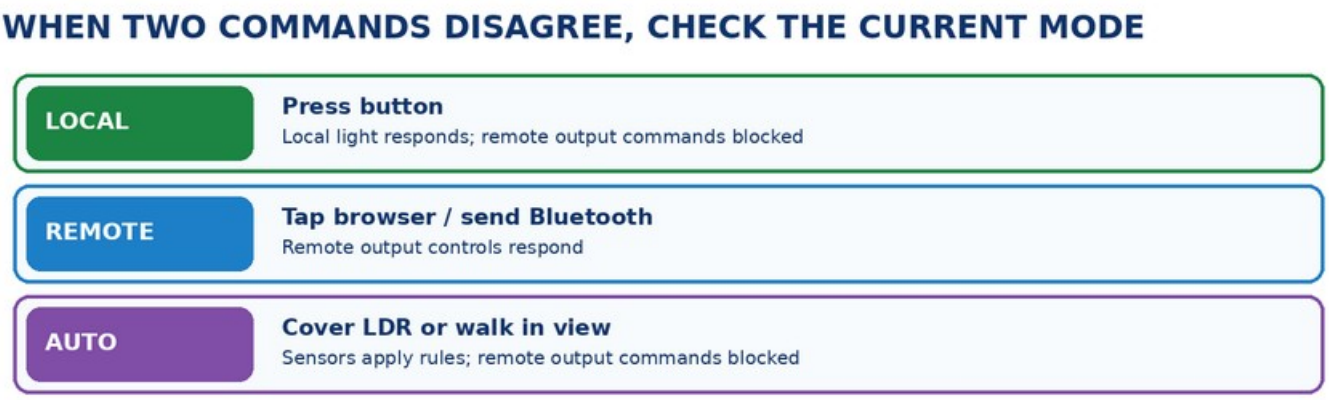
🤔 **THINK Why is the current mode as important as the command you send?**

_____

ACTIVITY 8 • TEST STEP 4

What If Commands Disagree?

Test the decision rules • No extra wiring



Conflict challenge: the mode decides

Starting mode	Challenge	Expected rule
AUTO	Send Bluetooth “F” (fan ON).	IGNORE remote output command.
AUTO	Cover the LDR for a moment.	Run the tested AUTO light rule.
AUTO	Press the new LOCAL button.	Switch to LOCAL and toggle the light.
REMOTE	Cover the LDR for a moment.	Sensor may be read, but AUTO outputs do not run.
LOCAL	Try browser Light ON.	REMOTE output command is blocked.

Write ONE test you conducted: _____

The expected rule was: _____

What I actually saw: _____

Important design decision LOCAL always takes over when the physical button is pressed. This is a convenient control fallback, not a certified emergency stop. Teacher power-off remains essential.

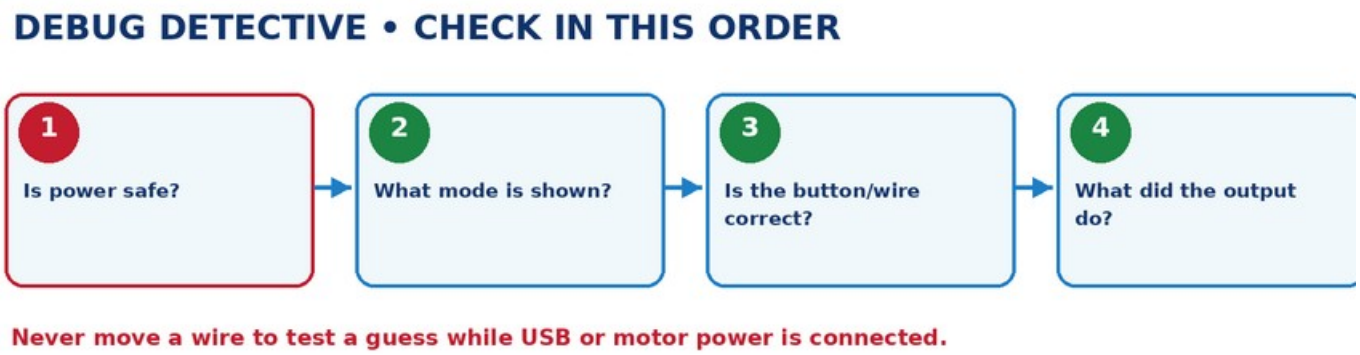
🤔 **THINK Which source should decide the light in AUTO: the phone or the tested LDR thresholds? Explain.**

_____

ACTIVITY 8 • DIAGNOSE & PROTECT

Debug Detective + Digital Safety

When something fails, check safely and in order



Never move a wire to test a guess while USB or motor power is connected.

Problem	First safe checks
Button does not toggle	Is LOCAL displayed? Does GPIO32 read LOW when pressed? Check two distinct button terminals with power disconnected.
Bluetooth commands ignored	Is REMOTE selected? Is phone paired with the right model?
Browser command blocked	Mode may be LOCAL or AUTO; remote output commands work only in REMOTE.
AUTO action surprises us	Review measured thresholds, sensor position and the latest log; do not change wiring while powered.
Wi-Fi unavailable	Local button still works. Teacher may restore the approved access point; do not use unknown networks.

Real-world responsibility

- Use only the teacher-approved isolated classroom Wi-Fi and paired device. Never publish the local controller or port-forward it.
- If the model door, motor or fan behaves unsafely, teacher must physically remove the motor/external power.
- Do not share network passwords, a student's personal details or classroom photographs through the dashboard.

⚠️ **SAFETY • RED CHECK** A software mode change or on-screen STOP is NOT a physical emergency disconnect. Teacher switch-off and disconnected supplies take priority.

🤔 **THINK If Wi-Fi stops working, what should a student check before touching any electronics?**

_____

ACTIVITY 8 • THINK, RECORD & EXPLAIN

Five Thinking Questions + Engineer's Log

Answer from YOUR experiments, not copied answers

1. Why is a physical LOCAL button useful even when Bluetooth and Wi-Fi already control SMARTSPACE?

_____

_____

2. In AUTO, the light changes with the LDR but the browser Light ON command is blocked. Why is that desirable?

_____

_____

3. A student presses the LOCAL button while AUTO is controlling the fan. What should happen first, and why?

_____

_____

4. One of the four legs of a push button is connected incorrectly. What safe check could identify the correct pair?

_____

_____

5. How would you improve the control modes for an older person or someone who cannot use a smartphone?

_____

_____

My Engineer's Log

We kept working	We added	We tested / improved
_____	_____	_____
_____	_____	_____

4CG evidence: CG-1 ☐ CG-2 ☐ CG-3 ☐ CG-4 ☐ | 4C skills: Think • Create • Collaborate • Communicate

Teacher verification: _____ **Date:** _____ **Team:** _____

ACTIVITY 8 • SAME PROJECT CHECKPOINT

Our SMARTSPACE Now Has Three Control Methods

EIGHT activities complete • ONE connected system

Check the SAME model — nothing has been dismantled

- ☐ ROOM LIGHT 1 still works with ESP32 GPIO23.
- ☐ Bluetooth and the private Wi-Fi dashboard still connect.
- ☐ LDR, DHT11 and PIR readings still appear.
- ☐ Door, model fan, buzzer and LCD still respond safely.
- ☐ LOCAL push button (GPIO32 → GND) works.
- ☐ REMOTE commands work in REMOTE mode; AUTO sensor rules still run.

Next Activity 9 — AI for SMARTSPACE: Can AI Help Us Save Energy?

We will train and test a small model with open-source Orange, then try a teacher-supervised gesture demo using MediaPipe. We will verify ideas on the SAME SMARTSPACE.

⚠️ **SAFETY • RED CHECK** When finished, return to LOCAL with outputs OFF. Teacher disconnects USB and external power before storage. Keep the SAME SMARTSPACE model assembled for Activity 9.

🤔 **THINK What data would you record to show that automatic lighting really reduces unnecessary use?**

_____

ACTIVITY 8 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY

SMARTSPACE • ACTIVITY 8 CHECKPOINT



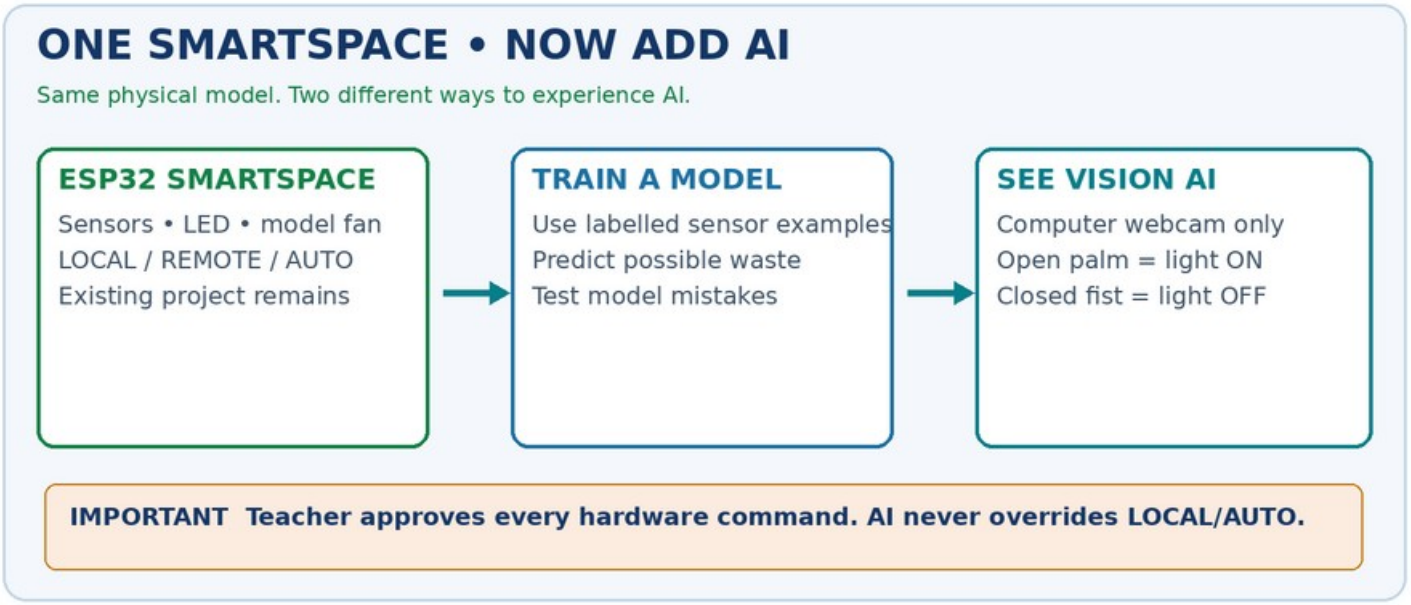
NOW YOU CAN...

- LOCAL: physical button
- REMOTE: phone + browser
- AUTO: measured rules
- Keep all 7 previous
- features working

The push button belongs on the SAME MODEL — no new robot or second ESP32.

AI for SMARTSPACE

4 hours | TRAIN • RECOGNISE • PREDICT • VERIFY • IMPROVE



MOTTO "I train AI, test its predictions, notice its mistakes and use it responsibly to improve SMARTSPACE."

Our two practical missions

1 Train a decision-tree model to identify examples that deserve an energy-use review, using readings from our SAME SMARTSPACE model.

2 Test a pretrained gesture-recognition model on a teacher-controlled computer and use a recognised hand gesture to control ROOM LIGHT 1.

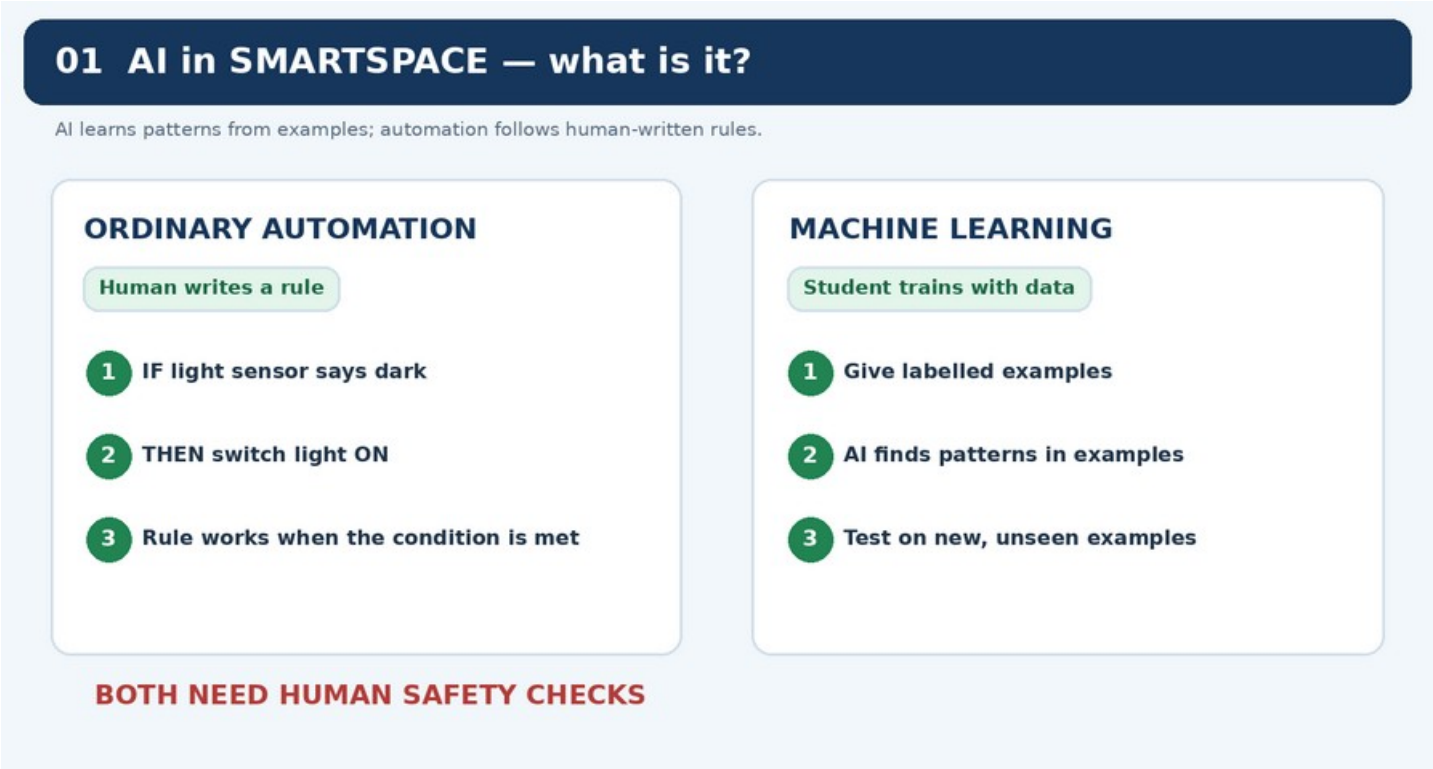
STOP • STUDENT SAFETY Keep the original low-voltage SMARTSPACE assembled. The AI activity does not introduce new ESP32 wiring. Teacher checks the fan guard, servo movement and Wi-Fi security before starting.

In my own words, a smart system is...

_____

What Is AI? What Will We Learn?

AI finds patterns and makes predictions; it does not always know the correct answer.



ARTIFICIAL INTELLIGENCE (AI) is a way to build computer systems that can recognise patterns or produce useful predictions from data. MACHINE LEARNING is a type of AI in which a model learns patterns from examples instead of following only a rule written by a person.

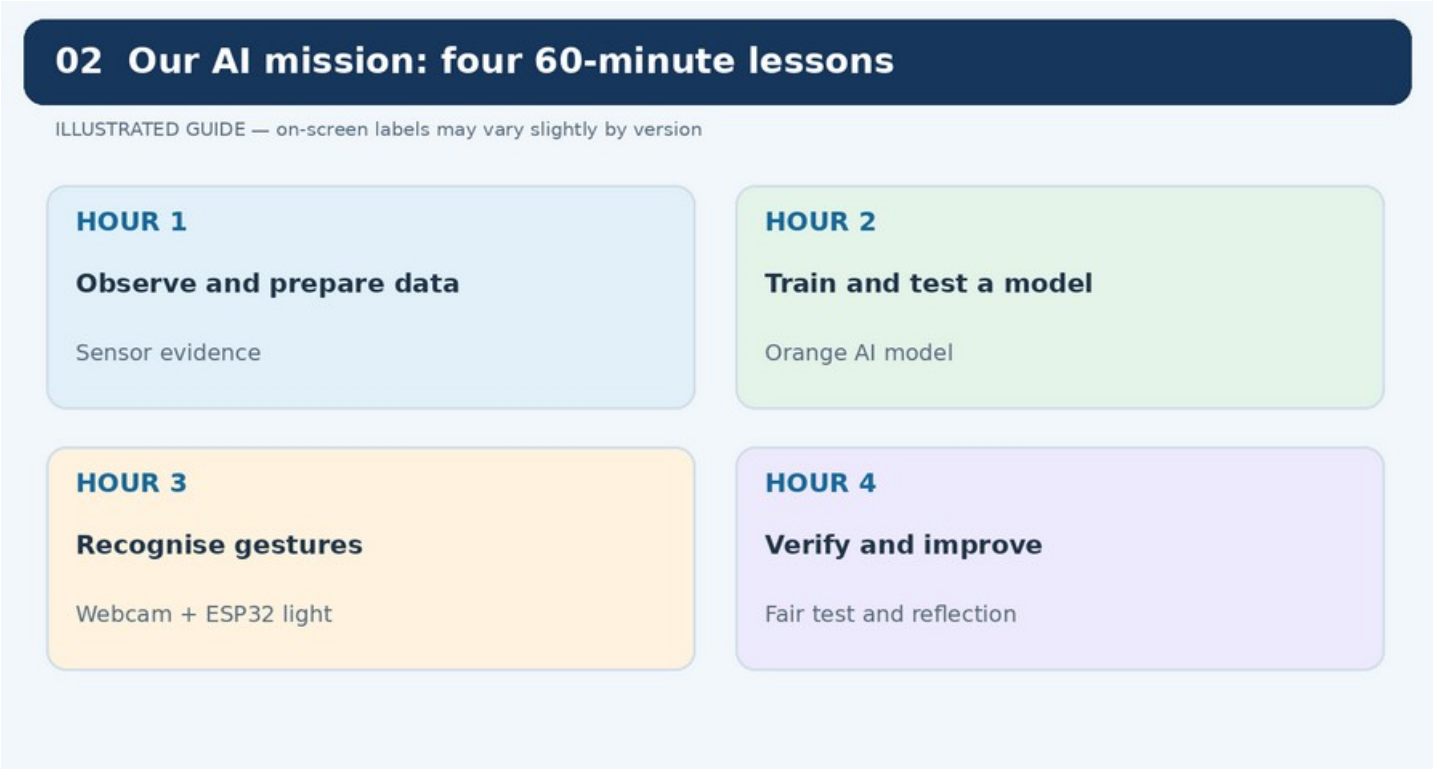
By the end, I can...

- ☐ Explain the difference between ordinary IF/THEN automation and machine learning.
- ☐ Collect and label SMARTSPACE sensor observations without personal information.
- ☐ Train and TEST a model on different datasets, then discuss one error.
- ☐ Demonstrate a PRETRAINED gesture recogniser controlling the same model LED.
- ☐ Judge whether an AI-supported suggestion should be adopted after a fair test.

IMPORTANT DISTINCTION Hour 2: YOU train an energy-review decision tree. Hour 3: you TEST an existing pretrained gesture model; we do not claim to have trained its camera model.

What Will We Achieve in Four Hours?

Every session produces something you can show a teacher.



Time	Learning goal	Evidence produced
Hour 1 • 60 min	Understand AI and prepare data	My sensor-reading sheet + labelled CSV
Hour 2 • 60 min	Train and test AI in Orange	A model workflow + test results
Hour 3 • 60 min	Recognise gestures and explain results	4 gesture trials + safety check
Hour 4 • 60 min	Investigate improvements fairly	Before/after record + final reflection

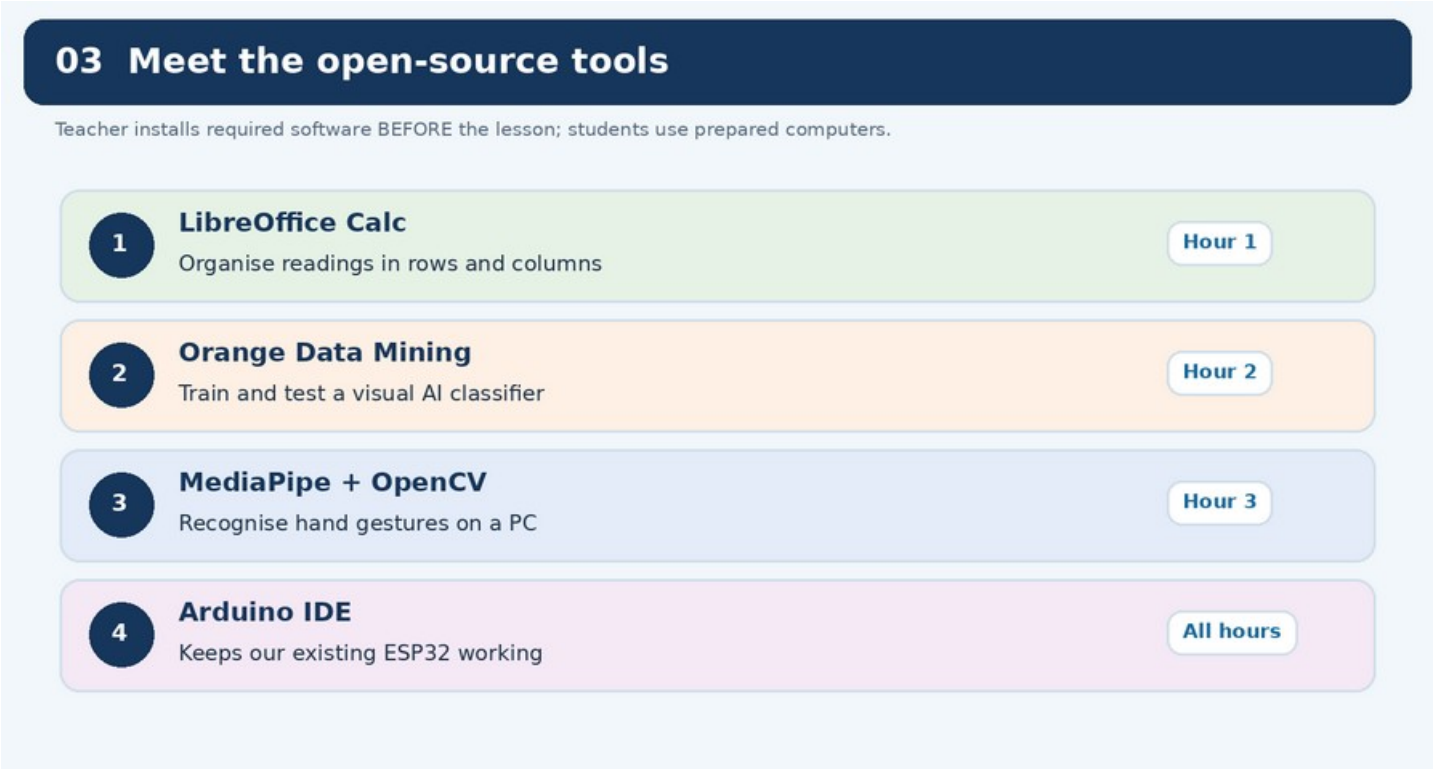
ONE PROJECT All four hours connect to the original SMARTSPACE. Orange runs on the classroom PC; the ESP32 still runs ordinary automation rules.

The most exciting thing I want to test is...

_____

Which Software Do We Use — and Why?

Every tool is open source. All teacher installations happen before the lesson.



1 LIBREOFFICE CALC (Hour 1): a free spreadsheet programme used to organise readings and open/save CSV files.

2 ORANGE DATA MINING (Hour 2): free visual AI software. Instead of typing difficult code, we connect on-screen widgets to train and test a decision tree.

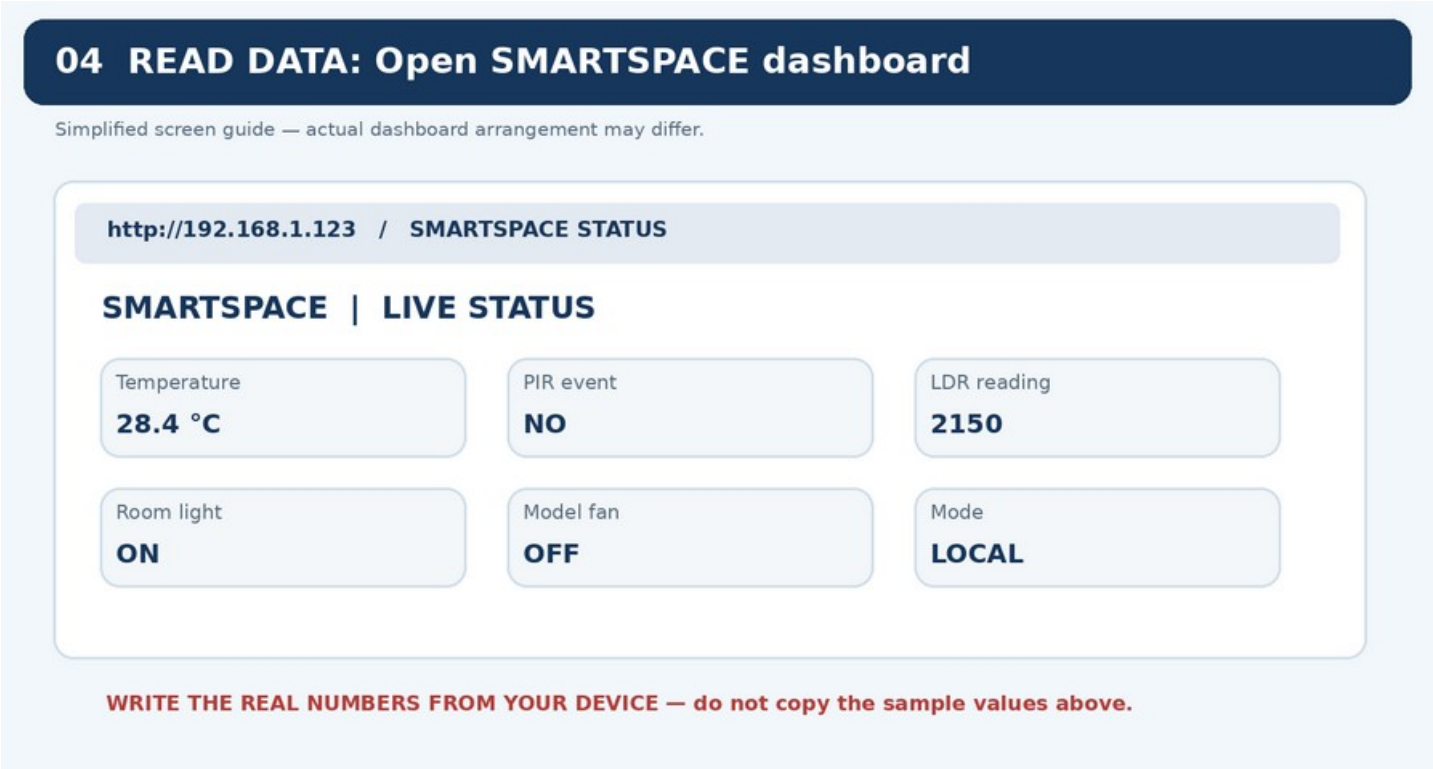
3 MEDIAPIPE + OPENCV (Hour 3): open-source Python libraries used by the teacher's computer to recognise hand gestures from a webcam.

4 ARDUINO IDE (existing): keeps the ESP32 programme working. It is not the AI model-training application.

TEACHER CHECK Teacher prepares Orange, LibreOffice, a compatible Python environment, webcam permissions and a separately licence-checked pretrained gesture model. The AI lesson must not begin with students troubleshooting installations.

STEP 1 — Read Your SMARTSPACE Sensors

Use the existing IoT status page or Serial Monitor.



1 Ask your teacher to switch ON the checked SMARTSPACE. Keep it in LOCAL mode while you record observations.

2 On the classroom laptop, connect to the teacher-approved local Wi-Fi. Open the IP address of your existing SMARTSPACE dashboard in the browser.

3 Read the current temperature, PIR event, LDR reading and device ON/OFF status. These are measurements, not AI predictions.

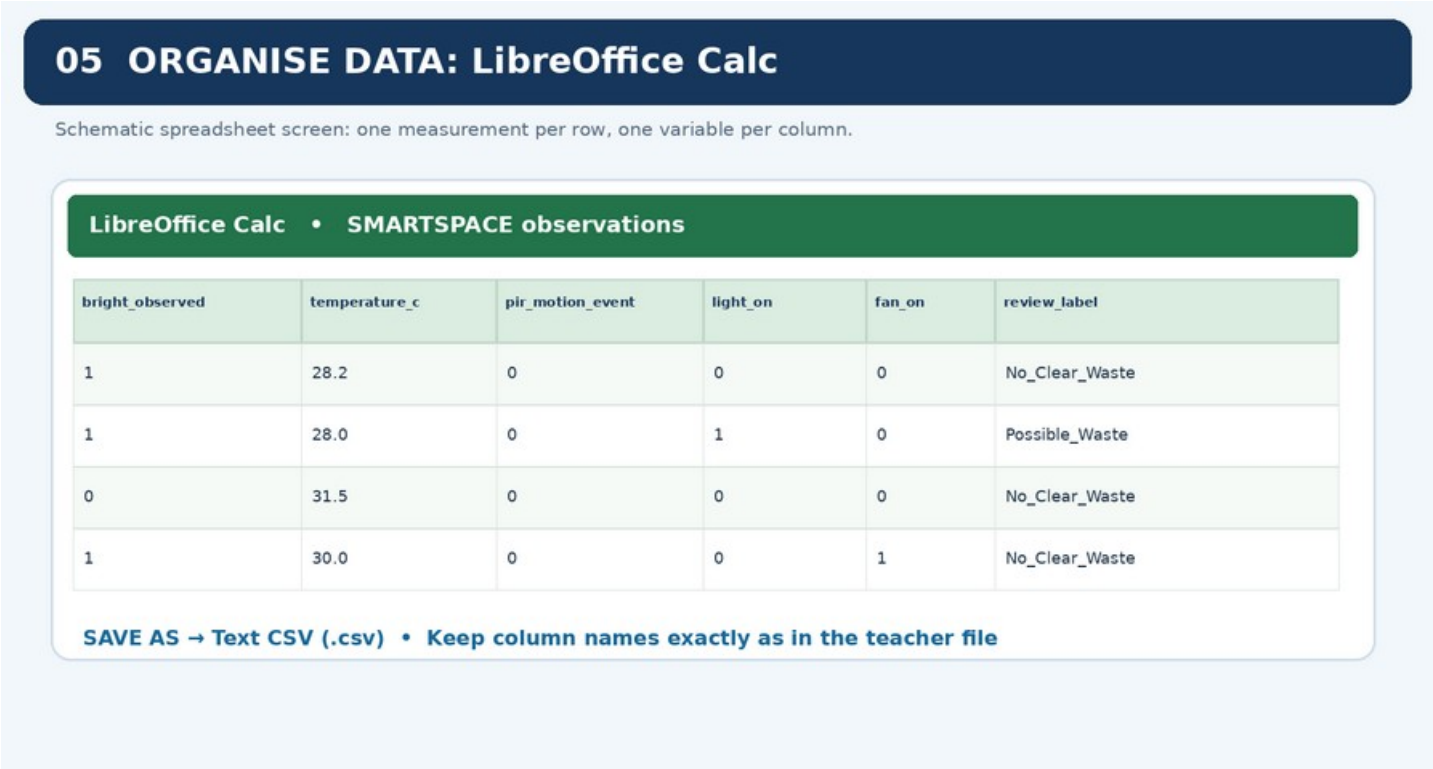
4 Copy the ACTUAL values below. The numbers in the picture are only an example.

Measured at	LDR value	Temp °C	PIR	Light	Fan
Time _____	_____	_____	___	___	___

RED CHECK A PIR motion event is NOT proof that the room is occupied or empty. Do not use this classroom model for real security or access control.

STEP 2 — Organise Data in LibreOffice

This is how sensor observations become a dataset.



1 Open LibreOffice Calc using its computer icon or the desktop Applications menu. Teacher provides the blank observation CSV from the kit.

2 Choose File → Open; select SMARTSPACE_student_observation_blank.csv. If a Text Import window appears, choose comma as the separator and confirm.

3 Enter one new observation on each row. Keep the exact column headings from the original CSV. Do not enter student names or pictures.

4 Use 1 for “Yes/ON” and 0 for “No/OFF” where indicated. Write the observed temperature as a number, for example 28.4.

5 Choose File → Save As → Text CSV (.csv). Confirm “Use Text CSV Format” if asked and save to your teacher-approved folder.

CHECKPOINT Can your partner explain what one ROW represents and what a COLUMN represents? If yes, you are ready to label data.

ACTIVITY 9 • HOUR 1 • NEXT 20 MIN

STEP 3 — Give Data Meaningful Labels

How could unnecessary energy use appear in the observations?



1 Discuss one example: the classroom is visibly bright while ROOM LIGHT 1 is ON. It may deserve an energy-use REVIEW.

2 Label examples carefully as Possible_Waste or No_Clear_Waste. These are teacher-guided judgements, not measured watts or kWh.

3 Write WHY you chose the label. Think about whether sufficient natural light is available and whether the lamp is needed for the current task.

4 Never assume the absence of a PIR event means the room is empty. It could simply mean no movement was detected.

Example	What we observed	Our label and reason
1	Room bright; LED ON	_____
2	Room dark; LED ON	_____
3	Room bright; LED OFF	_____

GOOD DATA HABIT A model trained on mistaken labels will learn those mistakes. Labels are examples of human judgement, not perfect truth.

STEP 4 — Meet the Training and Test Files

Before a model can be trusted, it must face examples it did not learn from.

YOUR TRAINING DATA • EXAMPLE ROWS

ILLUSTRATIVE ONLY • generated practice data, not a real energy meter

Bright?	Temp °C	PIR event	Light ON	Fan ON	Review label
1	27.0	0	1	0	Possible_Waste
0	29.5	1	1	1	No_Clear_Waste
0	23.0	0	0	1	Possible_Waste
1	28.0	1	0	0	No_Clear_Waste

Teacher replaces these examples with real, approved observations during discussion.

1 Find SMARTSPACE_train_simulated.csv in the teacher kit. It has 40 labelled PRACTICE rows, prepared only for classroom learning.

2 Find SMARTSPACE_test_simulated.csv. It has 16 different PRACTICE rows that we keep for testing only.

3 Compare the file headings. Both files have five Features and the review_label Target. Keep your real classroom observation data separate.

4 Predict why it would be unfair to train AND test only on the same examples.

My explanation: test rows must be different because...

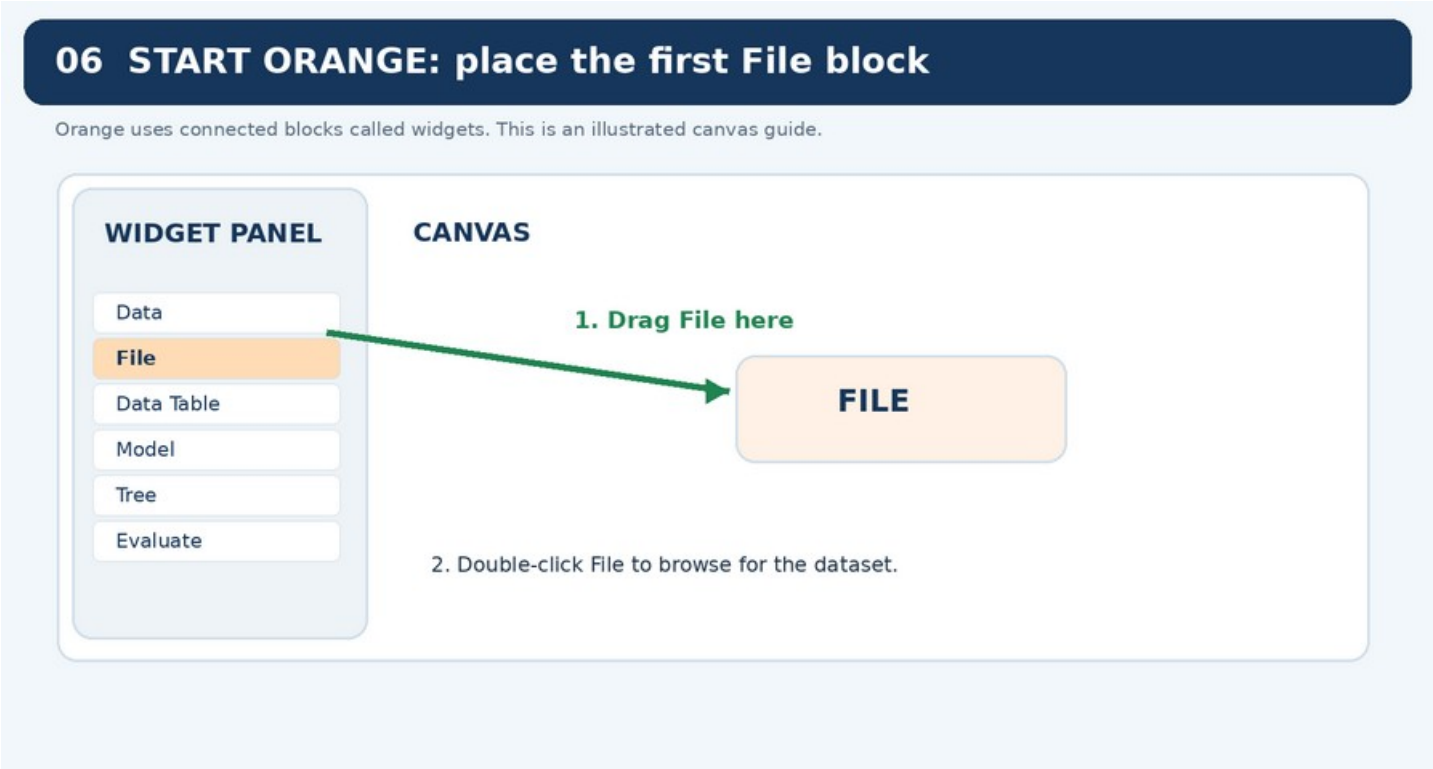
_____

_____

HONEST EVIDENCE Both supplied datasets are SIMULATED teaching examples. They cannot establish real-world model accuracy or actual school electricity savings.

STEP 5 — Open Orange Data Mining

Find the visual building blocks on your computer.



1 Double-click the Orange Data Mining desktop icon. Wait for the empty workflow canvas.

2 Look at the widgets panel on the LEFT. Find the category called Data and locate the File widget.

3 Drag File to the large canvas on the RIGHT. The widget should appear as one connected-block icon.

4 Double-click the File widget. A window for selecting a dataset should open.

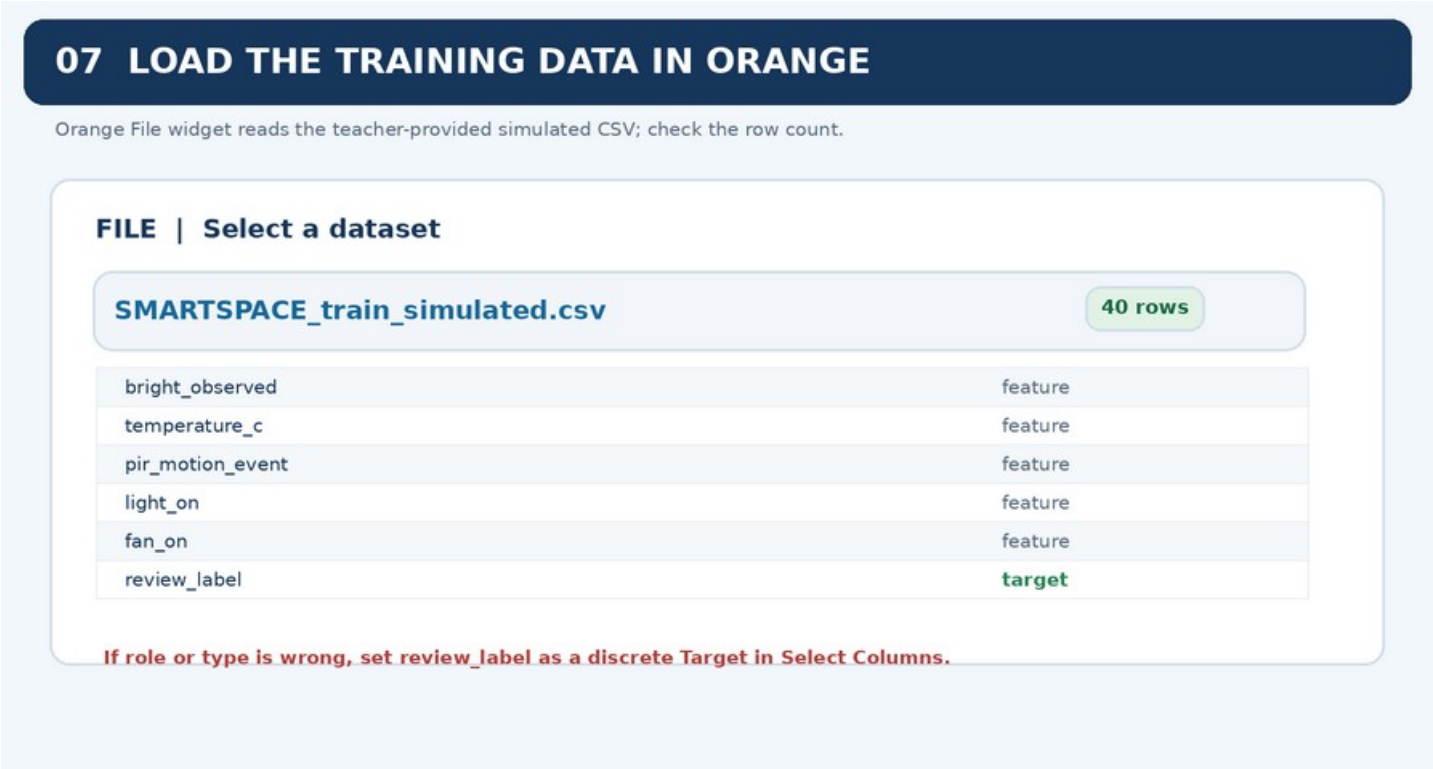
IF YOU CANNOT FIND IT Ask the teacher to check the Orange installation and whether the widgets panel is visible. Do not install software from unknown websites.

I found the File widget under this category...

_____

STEP 6 — Load Our Training Examples

Open the supplied CSV in the File widget.



1 In the File window choose Browse (folder icon). Select SMARTSPACE_train_simulated.csv from the teacher-provided folder.

2 Check the filename. Orange should show 40 rows and six columns (five inputs and one class label).

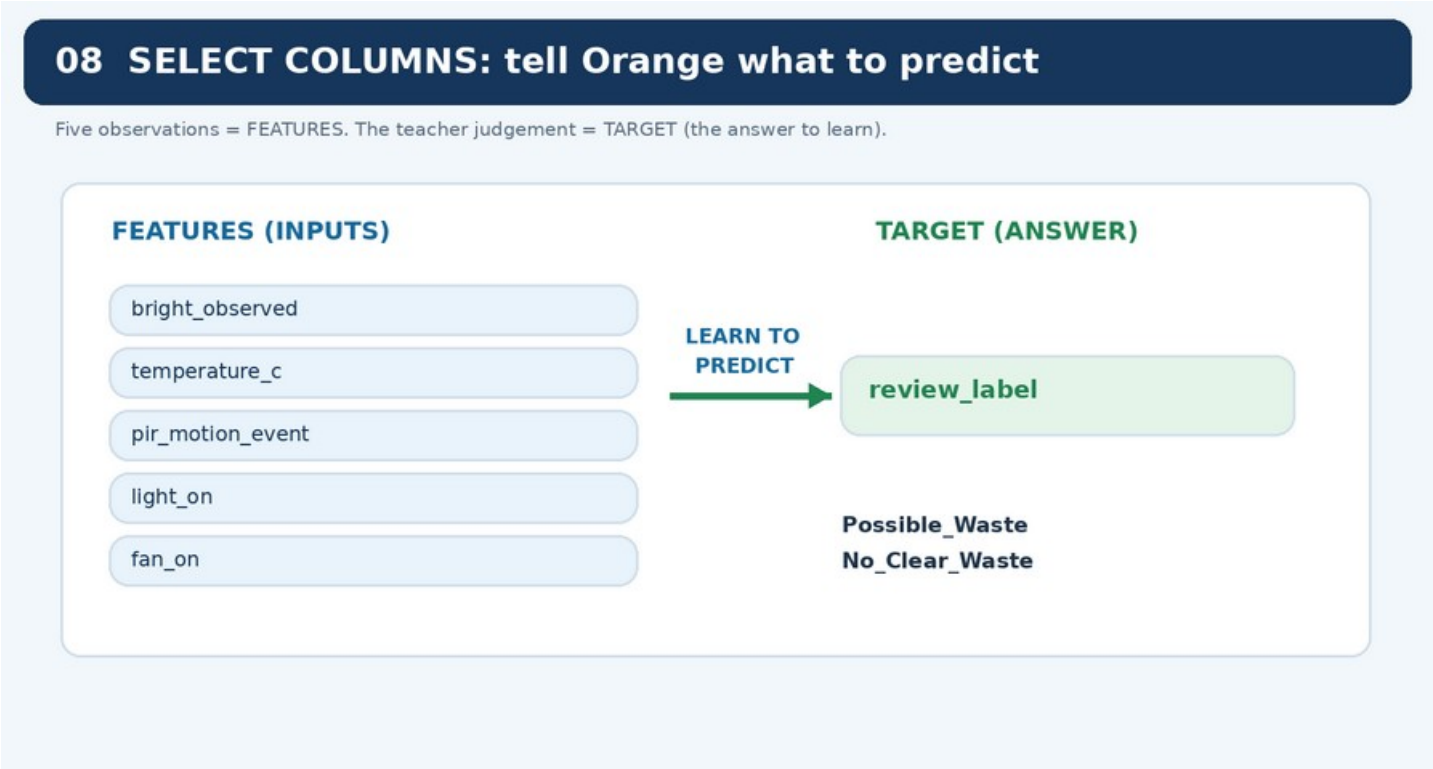
3 Examine the names: bright_observed, temperature_c, pir_motion_event, light_on, fan_on and review_label.

4 If the programme reports an unexpected file type or no data, close the widget and check the CSV with your teacher.

PICTURE GUIDE Our guide shows a simplified Orange screen, not a captured screenshot. Labels and button positions can differ slightly between releases.

STEP 7 — Tell AI the Correct Answer Column

The Target is the label that our model should predict.



1 From Transform widgets, drag SELECT COLUMNS to the canvas. Connect File → Select Columns.

2 Double-click Select Columns. Put these five fields under FEATURES: bright_observed, temperature_c, pir_motion_event, light_on, fan_on.

3 Move review_label to TARGET (or Class). It should have the two categories Possible_Waste and No_Clear_Waste.

4 If Orange sees review_label as free text, ask the teacher to change its variable TYPE to Discrete/Nominal, then set it as Target.

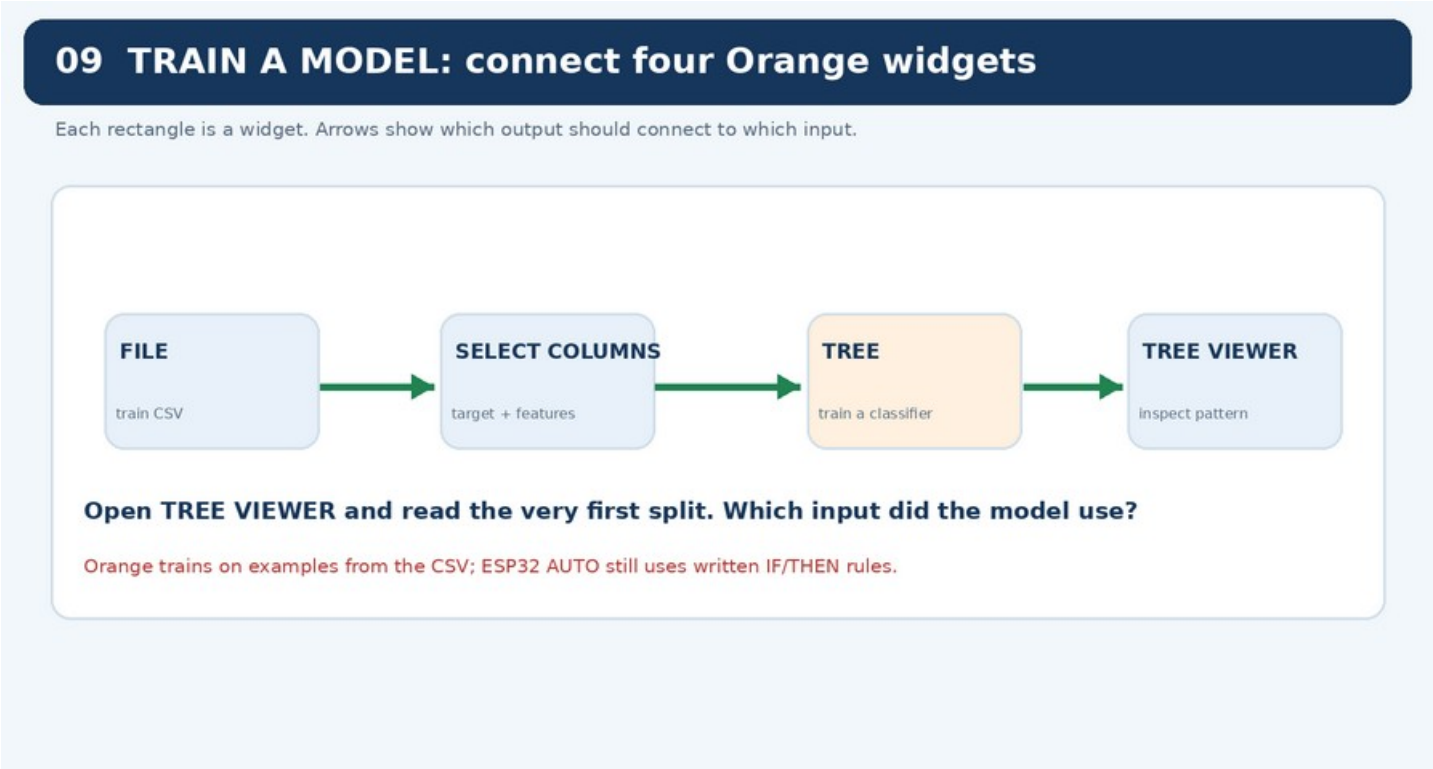
5 Check again: review_label must NOT also appear as a Feature. Otherwise the model could "cheat" by reading the answer.

AI DESIGN RULE Feature = information AI is allowed to inspect. Target = answer AI is trying to learn from the labelled training examples.

ACTIVITY 9 • HOUR 2 • NEXT 12 MIN

STEP 8 — Train Our First AI Model

Orange connects our examples to a small Decision Tree.



1 Drag the TREE widget onto your canvas under the Model category. Connect Select Columns → Tree (Data input).

2 Orange trains the decision tree using the supplied labelled examples. In most versions this happens automatically.

3 Find TREE VIEWER in the Visualize area and connect Tree → Tree Viewer using the trained Model output.

4 Double-click Tree Viewer to inspect the branches. Find the first decision at the top of your tree.

5 In one sentence, describe what information the first branch uses to divide the examples.

My model's first question is...

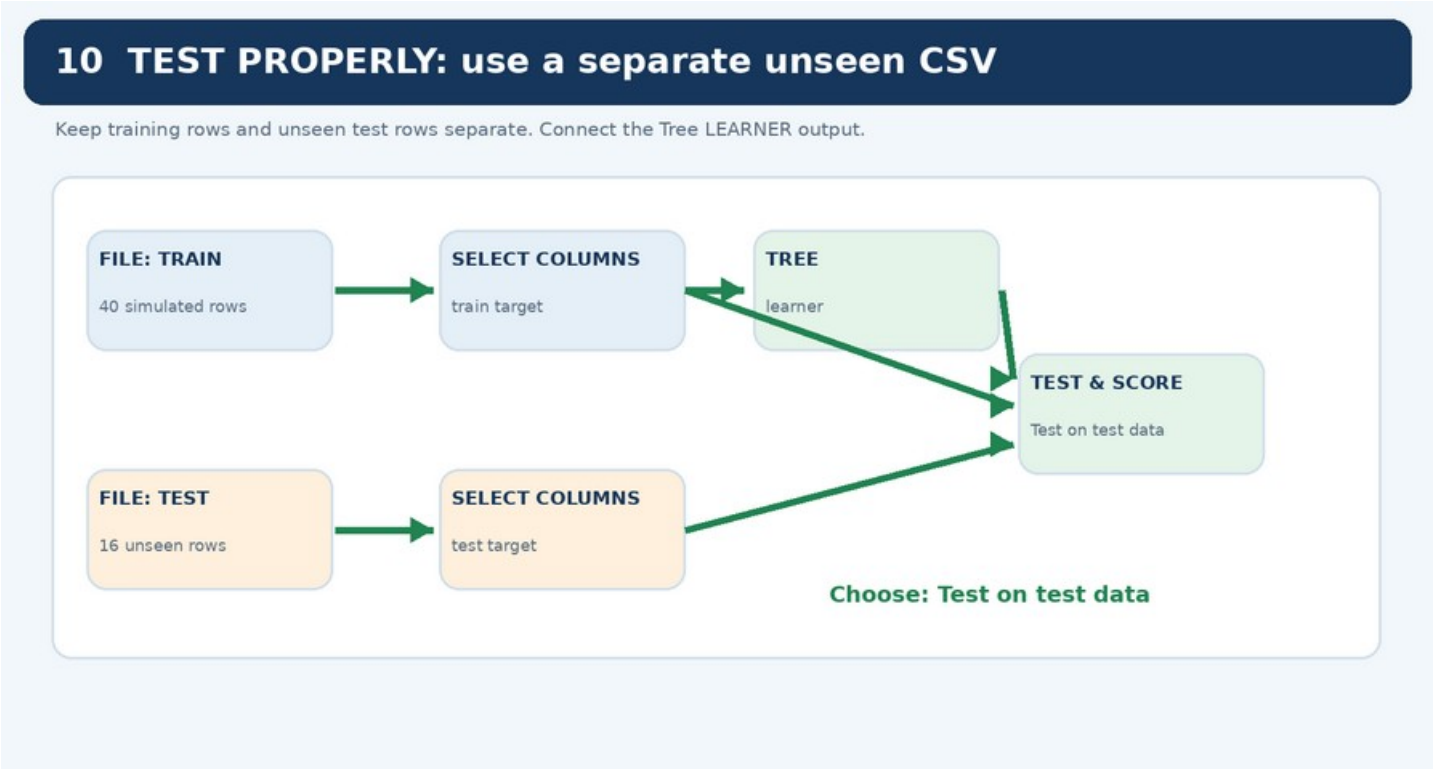
_____

UNDERSTAND IT Students TRAIN this particular model. The tree learns from classroom sample labels; it is different from the written IF/THEN rules in the ESP32 firmware.

ACTIVITY 9 • HOUR 2 • NEXT 12 MIN

STEP 9 — Test AI Using UNSEEN Data

A model must be tested fairly, not only on the examples it remembers.

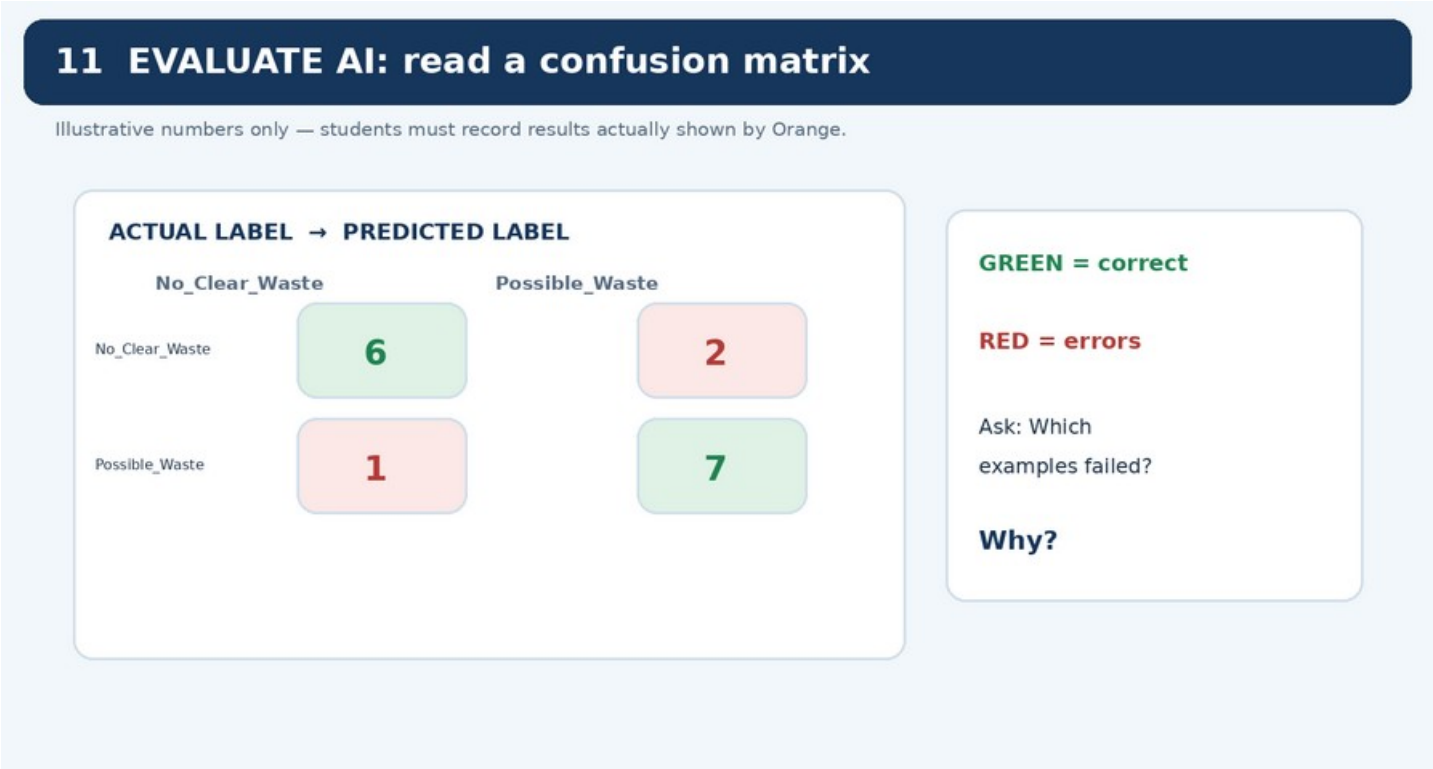


1. Add a SECOND File widget and open SMARTSPACE_test_simulated.csv. Check that it contains 16 rows.
2. Add another Select Columns widget; give the test file exactly the same five Features and review_label Target.
3. Drag TEST & SCORE from Evaluate onto the canvas. Connect TRAIN Select Columns → Data input.
4. Connect TEST Select Columns → Test Data input; connect TREE → Learner input. Set evaluation to “Test on test data.”
5. Connect Test & Score → CONFUSION MATRIX using the Evaluation Results output. Look for correct and wrong predictions.

DO NOT DO THIS Do not judge the model using “Test on train data.” It may score well simply because it has already learned from those rows.

STEP 10 — Read the AI Mistakes

Confusion Matrix helps us investigate predictions rather than trust them blindly.



1 Read the total number of new test examples and record the accuracy shown by YOUR Orange programme.

2 Find the correct predictions on the diagonal and the incorrect predictions outside it. The picture shows imaginary numbers, not actual results.

3 Choose one incorrect prediction. Can you explain why the model could have failed? Was an important feature absent?

4 Discuss how more varied examples or better review labels could help. Keep the unseen evaluation rows separate.

My test results	Fill after using Orange
Unseen test rows	16
Correct / incorrect	_____ / _____
Reported accuracy	_____ %
One possible weakness	_____

ENGINEER'S LOG Accuracy from a small synthetic CSV is a learning exercise, not proof that AI will make good decisions in every real classroom.

STEP 11 — What Is Gesture Recognition?

Now we TEST a pretrained AI model. We do NOT train it from scratch.



COMPUTER VISION allows a computer to examine images and identify visual patterns. Our teacher-prepared MediaPipe model can recognise categories such as Open_Palm and Closed_Fist. OpenCV displays a live camera preview; the video is processed on the local computer.

1 Teacher explains where the webcam is, how permission works and which students wish to participate voluntarily.

2 Teacher loads a separately reviewed, appropriately licensed MediaPipe gesture model and opens the prepared local script.

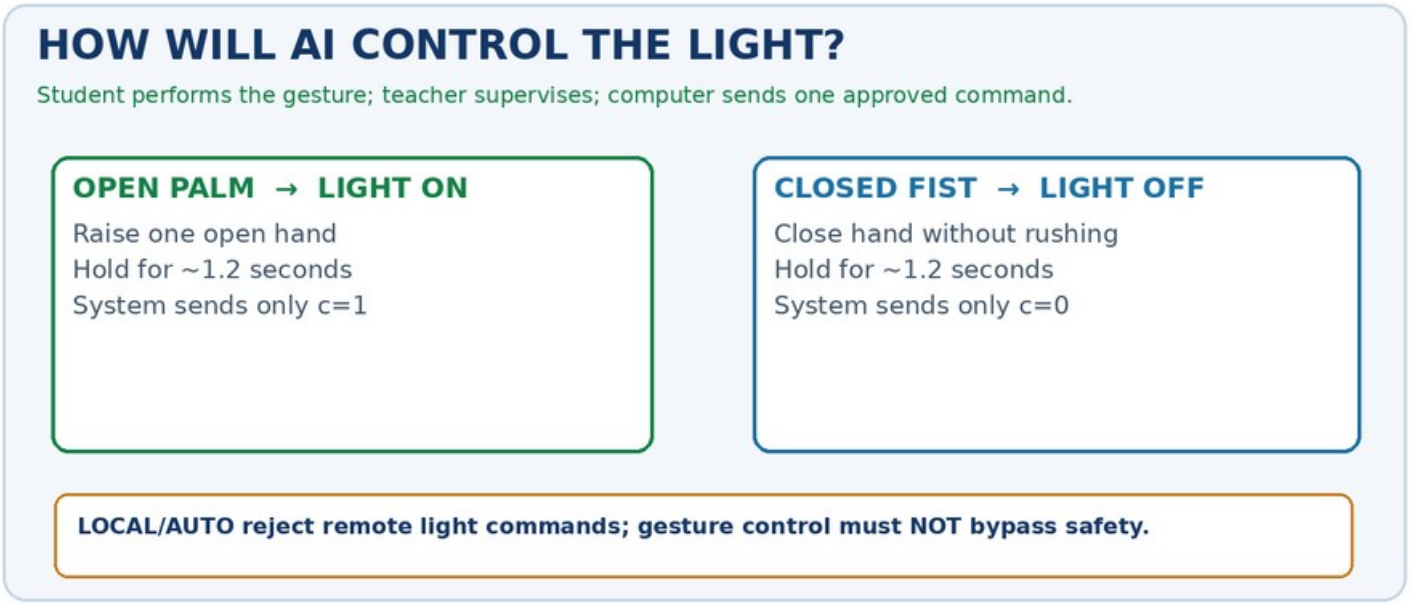
3 Show only a HAND, not a face. Volunteers try open palm and closed fist without sending commands to SMARTSPACE.

4 Record when the software correctly predicts the gesture and when it fails due to angle, background or lighting.

PRIVACY FIRST No photos, faces, names or hand videos should be recorded or uploaded. Students can select a camera-free observing role.

STEP 12 — Run a DRY-RUN on the Computer

A dry run prints what AI recognises but DOES NOT operate the ESP32.



1 Teacher opens the terminal in the folder containing gesture_bridge.py and the compatible .task model.

2 Teacher runs the command in the box below. Replace model_file.task with the real licensed filename on the teacher PC.

```
python gesture_bridge.py --model model_file.task --dry-run
```

3 When the camera preview opens, a volunteer shows Open Palm, holds it steadily and watches for the predicted label.

4 Show Closed Fist next. A valid prediction should appear; NO ESP32 light should change during this dry run.

5 Press Q to close the camera. Ask: why is testing without physical movement useful?

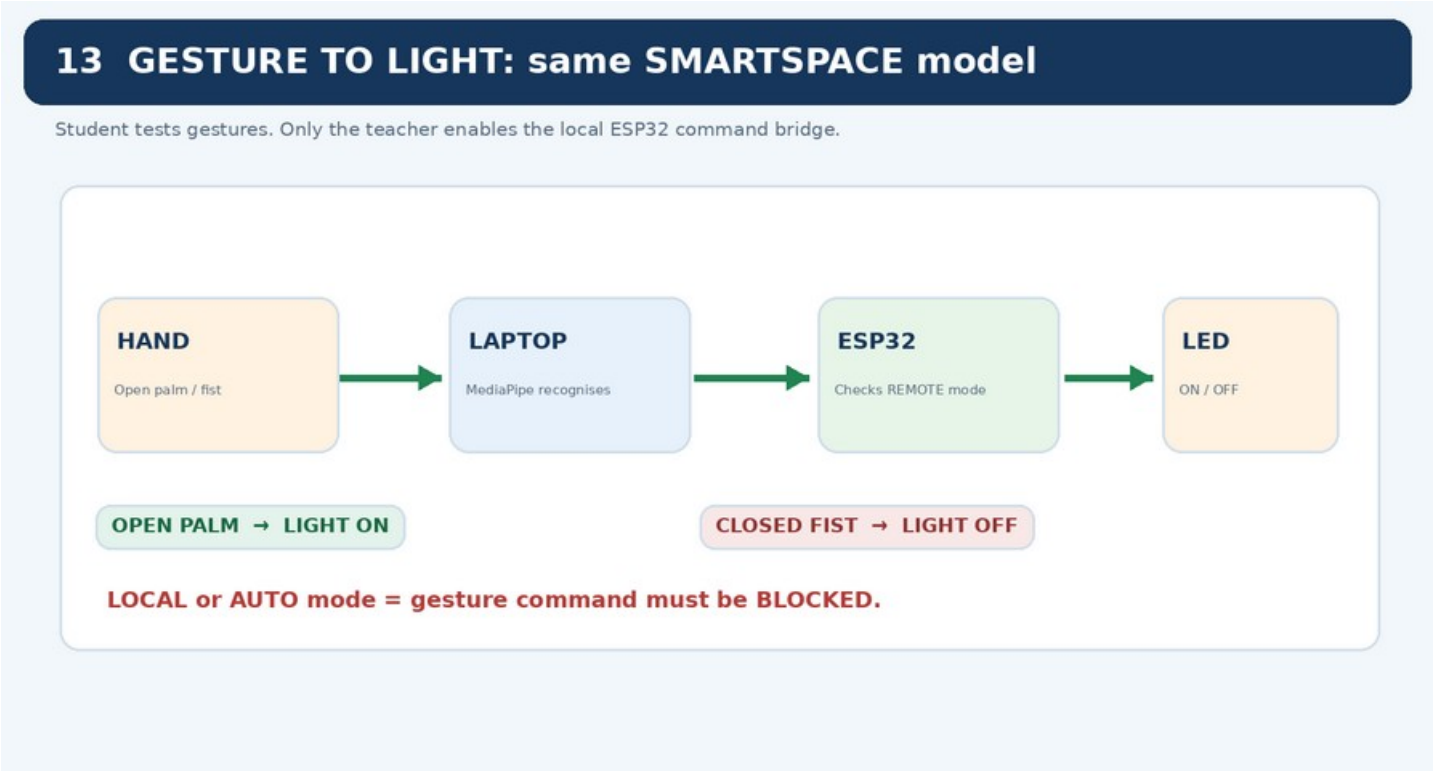
Gesture tried	AI label seen	Correct?
Open palm	_____	Yes / No
Closed fist	_____	Yes / No

TEACHER HANDLES CODE The supplied Python script is teacher-operated. This is not a student programming examination; our objective is to observe, test and explain AI predictions.

STEP 13 — Let a Gesture Control Our Light

📎 **CODE POINTER | See Annexure A9 (ESP32 reference) and Annexure G (teacher-run Python gesture bridge). The teacher operates the PC software.**

Only after teacher validation: webcam → computer → existing ESP32 light.



1 Teacher checks the isolated classroom Wi-Fi, ESP32 IP address, temporary authentication and output safety. No new wiring is required.

2 Teacher uses the SMARTSPACE dashboard to choose REMOTE mode; AI must not switch modes or control the door/fan.

3 Teacher starts the gesture bridge with the ESP32 private LAN IP address: python gesture_bridge.py --model model_file.task --host 192.168.1.123 (example address only).

4 Hold Open Palm steady. Did ROOM LIGHT 1 switch ON? Try Closed Fist to switch it OFF. Teacher watches for unexpected responses.

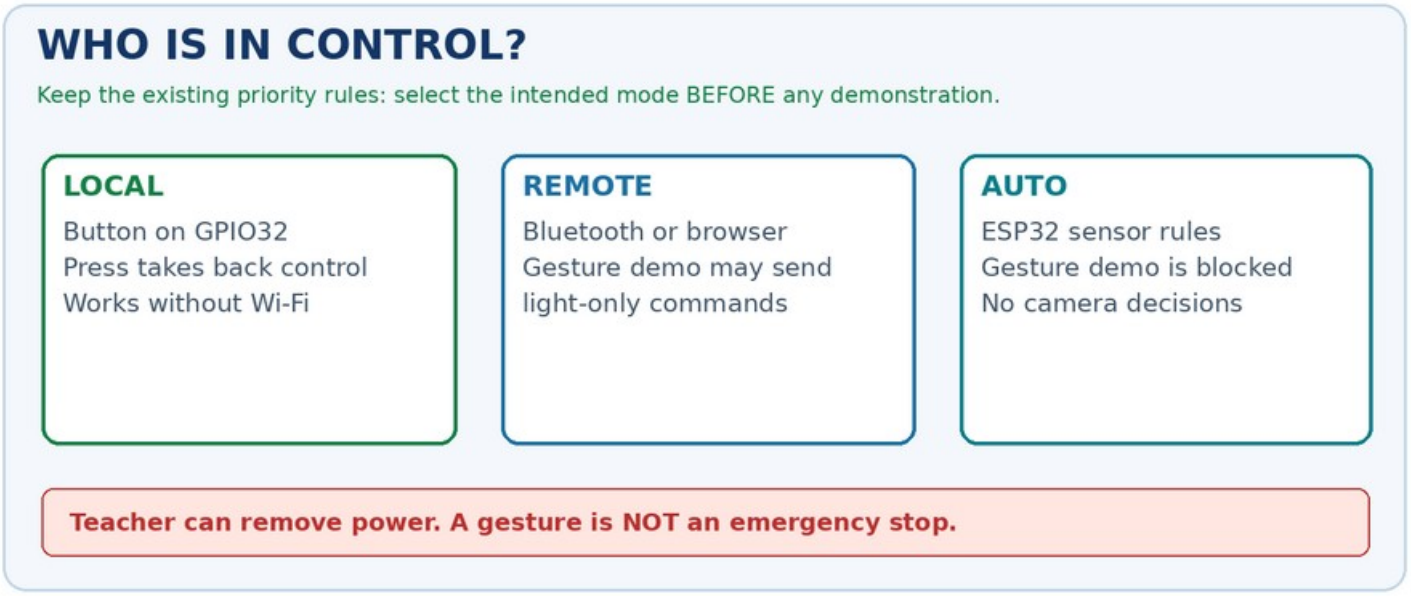
5 Test again with the model in AUTO and LOCAL modes: gesture commands must be blocked by the ESP32 programme.

Test	Expected outcome	What actually happened?
REMOTE + Open Palm	ROOM LIGHT 1 ON	_____
REMOTE + Closed Fist	ROOM LIGHT 1 OFF	_____
AUTO + gesture	Gesture BLOCKED	_____

NETWORK SAFETY The current dashboard uses basic HTTP authentication without encryption. Use an isolated teacher-controlled LAN; never expose SMARTSPACE to public Wi-Fi or the Internet.

STEP 14 — Test Control and Failure Cases

Responsible AI never overrides manual and safety controls.



1 If camera recognition is uncertain, remove the hand from view. Does the programme stop reporting a valid command?

2 Switch from REMOTE to AUTO and try a gesture. It must NOT change the model light by direct remote command.

3 Use the model's local push button. It should still function according to the previously tested LOCAL mode.

4 If Wi-Fi drops, explain which local control and physical power-disconnection safety options remain.

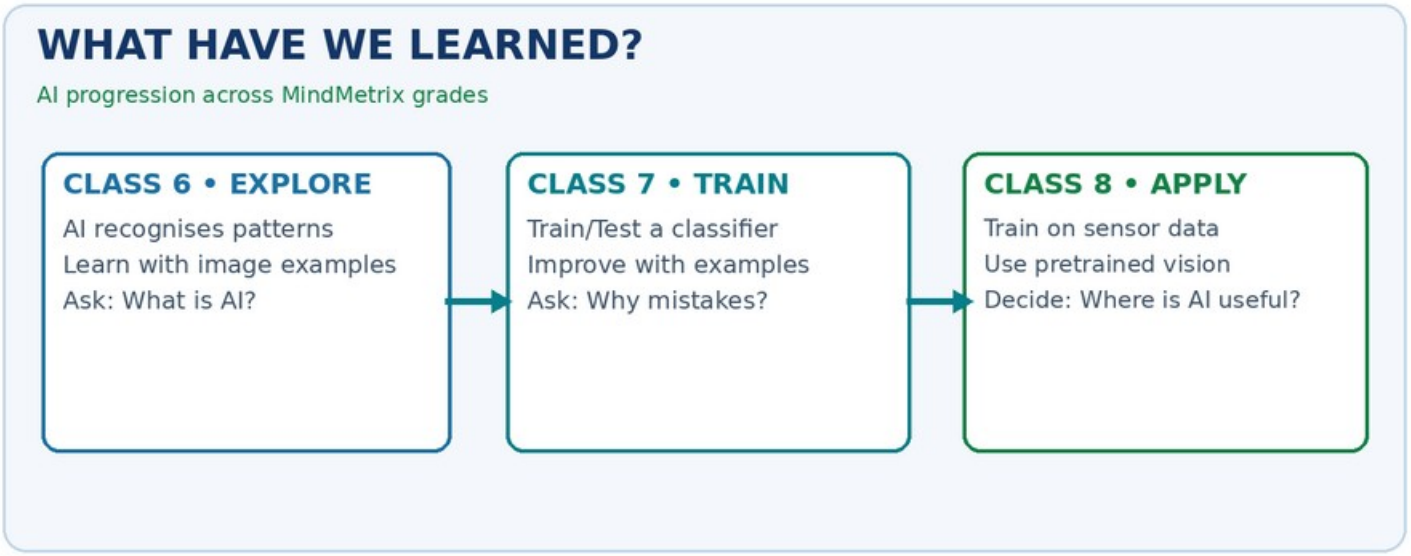
One limitation of gesture recognition is...

_____

STOP • HARDWARE Never use a gesture recogniser to operate real doors, mains devices or uncovered rotating equipment. A teacher can disconnect power whenever necessary.

STEP 15 — Choose One AI-Supported Improvement

An AI prediction is useful only when humans check it.



1 Review the possible-energy-waste examples and the errors of your Orange decision tree.

2 Choose ONE narrow question: for example, does ROOM LIGHT 1 remain ON when the model is placed under bright ambient light?

3 Compare the human IF/THEN threshold with your AI model's suggestion to review the situation.

4 Ask a teacher whether a small change to the LDR threshold is safe and useful. Do not allow AI to edit firmware or set physical outputs automatically.

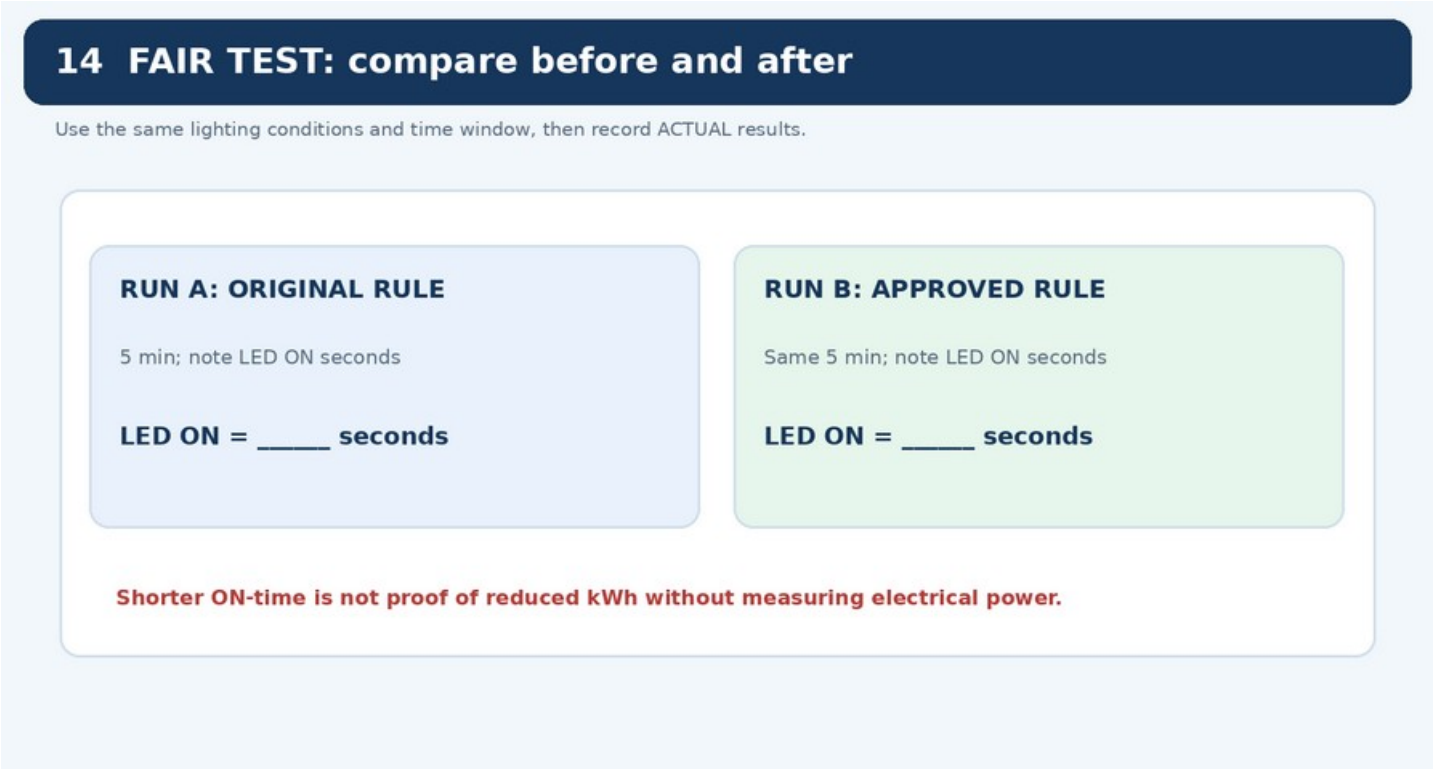
Our team wants to test...

_____

KEY SCIENCE RULE Labels like Possible_Waste do not tell us how much electrical energy was used. Never translate model accuracy directly into money or kWh saved.

STEP 16 — Compare the Original and New Rule

A FAIR TEST changes just one approved variable.



1 Run the original LDR rule for exactly FIVE minutes. Record LED ON-time with a stopwatch and note whether lighting is adequate.

2 Record the original threshold. Teacher approves one small threshold change in the existing ESP32 automation logic.

3 Repeat the same FIVE-minute scenario. Try to keep room light, model placement and timing as similar as possible.

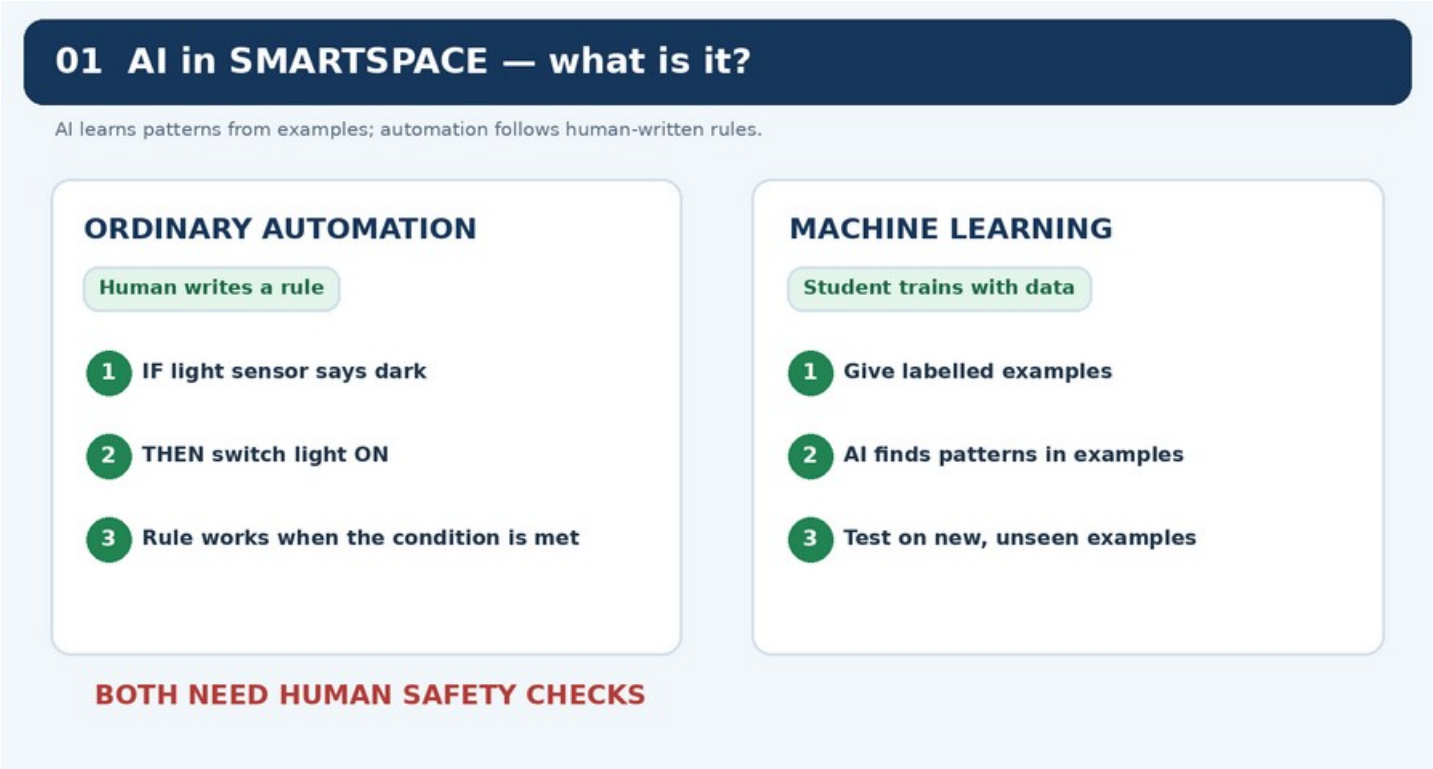
4 Record the new ON-time. If the new behaviour is unsafe or unreliable, restore the tested original programme.

Observation	Run A: original	Run B: new
Threshold	_____	_____
Light ON-time	_____ seconds	_____ seconds
Lighting adequate?	Yes / No	Yes / No
Unexpected behaviour	_____	_____

MAKE A CAREFUL CLAIM Less ON-time may be promising. It is NOT measured electricity saving unless we also measure electrical power or energy.

STEP 17 — Explain and Judge Your Results

Think like an engineer, not like an advertisement.



What did we discover?

1 MEASURE: Which sensor and light ON-time values did your team actually observe?

2 COMPARE: Was the AI classifier's review label helpful? When was it misleading?

3 VERIFY: Was the proposed LDR change tested under comparable conditions?

4 DECIDE: Should SMARTSPACE keep the original rule, use the approved change, or collect more evidence?

Evidence	Our team's conclusion
One correct AI prediction	_____
One mistaken AI prediction	_____
Our threshold decision	Keep original / Accept change / Need more data

RESPONSIBLE AI Human review and safety rules come first. The ESP32 remains a model—not a reliable occupancy detector, safety controller or validated energy meter.

ACTIVITY 9 • HOUR 4 • FINAL 15 MIN

My Five AI Thinking Questions

Write in your own words. Include one example from the activity.

1. What is the difference between an IF/THEN rule and the decision tree you trained in Orange?

_____

_____

2. Why is it important to test AI on examples that were NOT used for training?

_____

_____

3. Name one wrong AI prediction and explain what may have caused it.

_____

_____

4. Which AI model did students train themselves, and which model came pretrained?

_____

_____

5. If our LED stayed ON for fewer seconds, what can we conclude—and what do we still need to measure?

_____

_____

THINKING ICON AI can make a useful guess; humans must ask “What is the evidence?”

Engineer's Log — My SMARTSPACE AI Mission

We are ready to demonstrate our work in Activity 10.

My AI achievement checklist

- □ I can explain machine learning and ordinary automation.
- □ I can collect and label classroom sensor examples.
- □ I can open a CSV, choose the Target and train a decision tree in Orange.
- □ I can test on unseen examples and explain at least one error.
- □ I can recognise a pretrained vision model and test it responsibly.
- □ I can show a fair test before deciding whether AI helped the SMARTSPACE.

The most important thing I learned...

_____

Assessment: CG-1 □ CG-2 □ CG-3 □ CG-4 □ • Critical thinking • Creativity • Collaboration • Communication

Teacher verification: _____ **Date:** _____ **Team:** _____

MISSION COMPLETE Keep the same SMARTSPACE ready for Activity 10. Students will show the sensors, dashboard, controls, automation and their AI learning evidence.

ACTIVITY 9 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY

SMARTSPACE • AFTER ACTIVITY 9

OUR SAME WORKING MODEL + AI TRAINING + SUPERVISED GESTURES



AI ON THE COMPUTER

1. **Trained model**
   Orange decision tree
2. **Tested AI**
   Unseen data + errors
3. **Gesture demo**
   Camera -> ESP32 light

OUR ACHIEVEMENT

We trained and tested a classifier, then demonstrated a separate pretrained gesture model.

AI runs on a classroom computer. It does not automatically run on the ESP32.

ACTIVITY 10 • GRAND CHALLENGE

WE BUILT SMARTSPACE!

Same model • Final integration • Test • Improve • Present



Your big challenge

Your team has constructed ONE SMARTSPACE across nine missions. Today you will prove that its electronic parts, software and safety practices work together. You will NOT assemble a different project.

SUCCESS MEANS A safe, working demonstration with real evidence, documented limitations and a useful improvement—not a flawless performance.

SESSION 1 • 45 MIN	SESSION 2 • 45 MIN	SESSION 3 • 45 MIN
Inspect + verify parts • 15 Test LOCAL + REMOTE • 20 Record problems • 10	Test sensors + AUTO • 20 Retest improvements • 15 Prepare evidence • 10	Showcase / peer review • 25 Five thinking questions • 10 Clean up + log • 10

THINK Which SMARTSPACE feature would you show first to prove your model is really working?

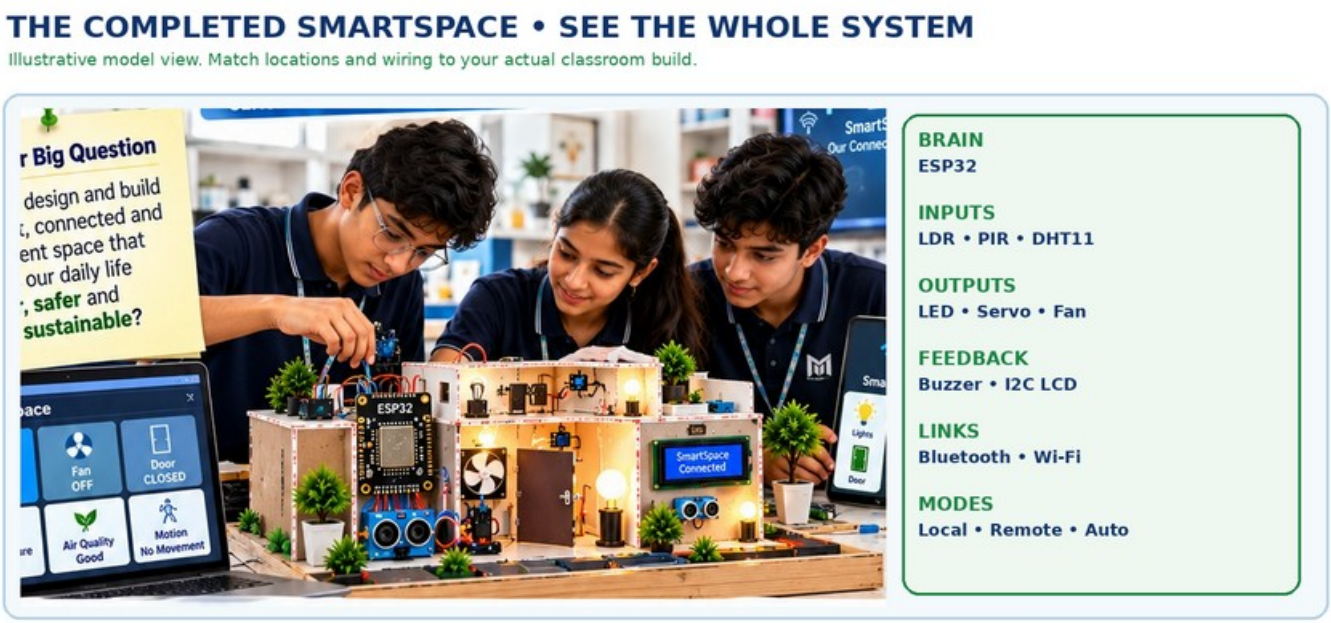
_____

⚠ **SAFETY ● RED CHECK** Teacher approves every power connection before the final demonstration. No exposed fan blades, loose wires, hot parts or wet surfaces.

ACTIVITY 10 • VISUAL ORIENTATION

Explore Our Finished Model

Look • Point • Name • Explain



Find these on YOUR real project

Locate and point	What should it do?	Checked?
ESP32 + ROOM LIGHT 1	Program controls a model light	☐
LDR + DHT11 + PIR	Report light, temperature/humidity, motion	☐
Servo door + model fan + buzzer	Move or alert with approved power	☐
I2C display	Show mode, light/fan and environment	☐
Phone + laptop	Bluetooth / private local Wi-Fi access	☐

THINK What is the difference between showing a model and explaining how its sensors lead to actions?

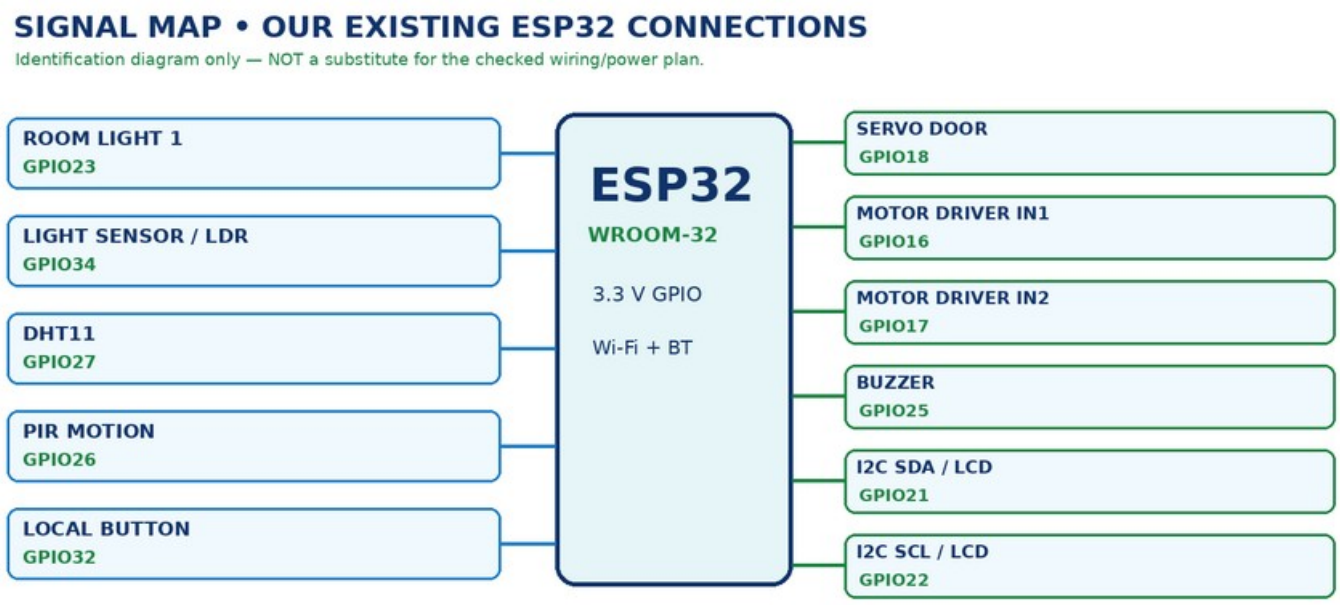
_____

PICTURE HINT Your exact model may differ from the illustration. Use the actual teacher-checked connections and module labels—not positions inferred from a photo.

ACTIVITY 10 • STEP 1

Before Power: The Final Inspection

Switch OFF → Trace → Check → Ask



Trace each signal label in your existing wiring

Do not make new connections today. Find the actual pin labels and compare with the previous activity wiring guide and the integrated program.

System	Pins to confirm	Tick
LED + sensors	LED GPIO23 • LDR GPIO34 • DHT11 GPIO27 • PIR GPIO26	☐
Door, fan and alert	Servo GPIO18 • driver GPIO16/17 • buzzer GPIO25	☐
Manual control + display	Button GPIO32 • I2C SDA GPIO21 / SCL GPIO22	☐

THINK What should you do if the label printed on your real module differs from a workbook illustration?

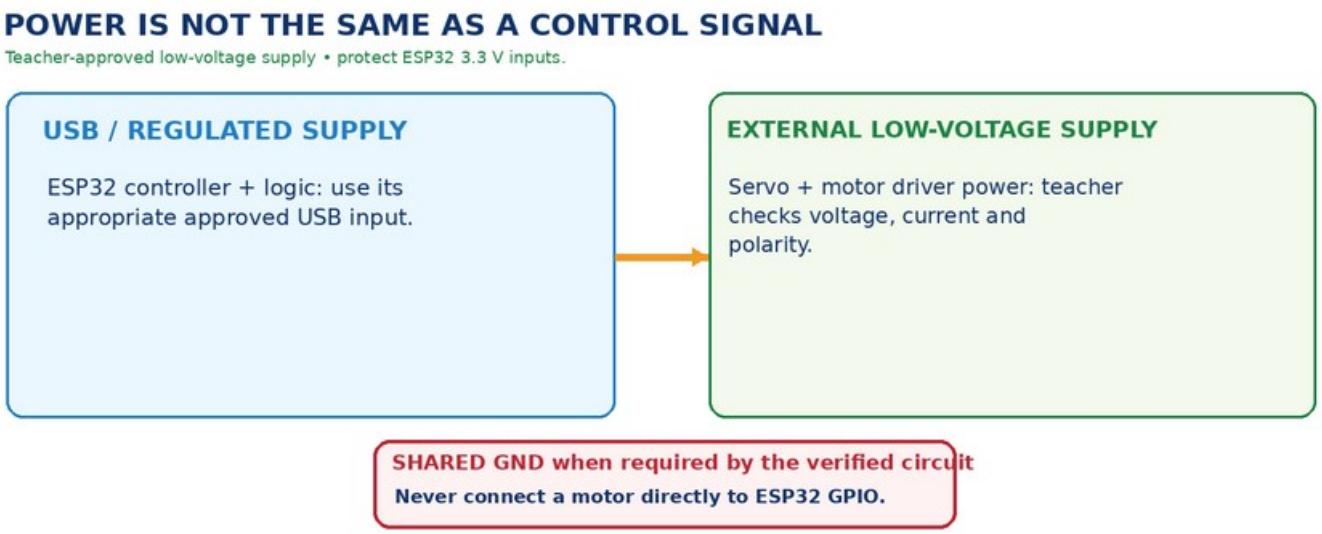
_____

⚠ **SAFETY • RED CHECK** Inspect while power is OFF. ESP32 GPIO is 3.3 V logic. The exact sensor, LCD level shifting and motor-driver wiring must already be approved by the teacher.

ACTIVITY 10 • STEP 2

Power Safely — Then Start the System

Observe the teacher check before plugging anything in



Our last pre-flight checklist

- ☐ All wires are firmly seated and labelled; no exposed conductor can short.
- ☐ ESP32 USB power and separately powered moving parts have correct voltage/polarity.
- ☐ A shared GND is used only where required by the verified circuit.
- ☐ Servo door travels freely; model fan has a guard or safe motor indicator.
- ☐ The classroom network is private and the teacher has verified account settings.

1 Teacher powers the system. Observe the ESP32 and LCD. Stop immediately for heat, smell, unusual noise or movement.

2 Watch startup. Our reference sketch starts in LOCAL mode with outputs OFF. Confirm your own tested version behaves as expected.

3 Ask before pressing. No student touches moving parts or rewires while the model is powered.

THINK Why is a software stop button not the same as physically removing power from an unsafe moving part?

_____

⚠ **STOP RULE** If the model behaves unsafely, call the teacher to disconnect suitable external power. Do not try random code changes while equipment is moving.

Prepare the Demonstration Program

📎 **CODE POINTER | See Annexure A9 for the integrated final ESP32 programme; use the same saved classroom model. Do not create a separate sketch.**

Use a known, saved sketch — not a brand-new program

Important: keep our previous working features

Use the Activity 8/9 integrated ESP32 sketch or your teacher-validated classroom version. Activity 10 adds NO new ESP32 pins. The separate Activity 9 webcam gesture bridge runs on the TEACHER computer; it is not firmware uploaded to ESP32.

Check before upload	What the team should confirm
Board and USB	Original ESP32-WROOM-32 • correct board / port
Components / libraries	BluetoothSerial, WiFi, WebServer, DHT, Wire, LiquidCrystal_I2C, ESP32Servo
GPIO / I2C	Pin map matches actual wiring; LCD address matches module
Sensor calibration	Real LDR direction and tested thresholds; servo angle is safe
Digital safety	Only teacher handles Wi-Fi passwords; browser is LOCAL, not public

1 Save a version. Store a clean copy named SMARTSPACE_FINAL_CLASSROOM with the teacher; keep the previous working version.

2 Verify before uploading. Check for compiler messages. Do not change libraries or board settings just to make an error disappear.

3 Upload safely. Teacher confirms stationary model, approved power and correct pin wiring; upload and run a low-risk test first.

THINK Why must a final working program be saved separately from experimental code changes?

_____

⚠️ **INTERNET SAFETY** Never print or publish network passwords, dashboard credentials, identifiable students or private IP URLs in the workbook. Local HTTP authentication does not encrypt traffic.

ACTIVITY 10 • STEP 4

Run Our Six Test Stations

Test • Observe • Tick • Fix



Build a real evidence trail

1 Startup check. Confirm the LCD displays status and no motor or servo moves unexpectedly.

2 LOCAL test. Press the push button. Record whether the existing ROOM LIGHT 1 changes state.

3 REMOTE test. Ask teacher to select REMOTE. Test one Bluetooth command and one browser command on approved devices.

4 Sensor check. Read light, temperature/humidity and motion values. Use realistic light changes and gentle walking; no heated objects.

5 AUTO check. Switch to AUTO and test safe dark/bright, temperature and motion events as allowed by the teacher.

6 Show final status. Read the LCD and browser/dashboard to explain what the system currently knows and does.

TEST RULE If a test fails, circle the step and return to the last successful checkpoint. Never fake an observation.

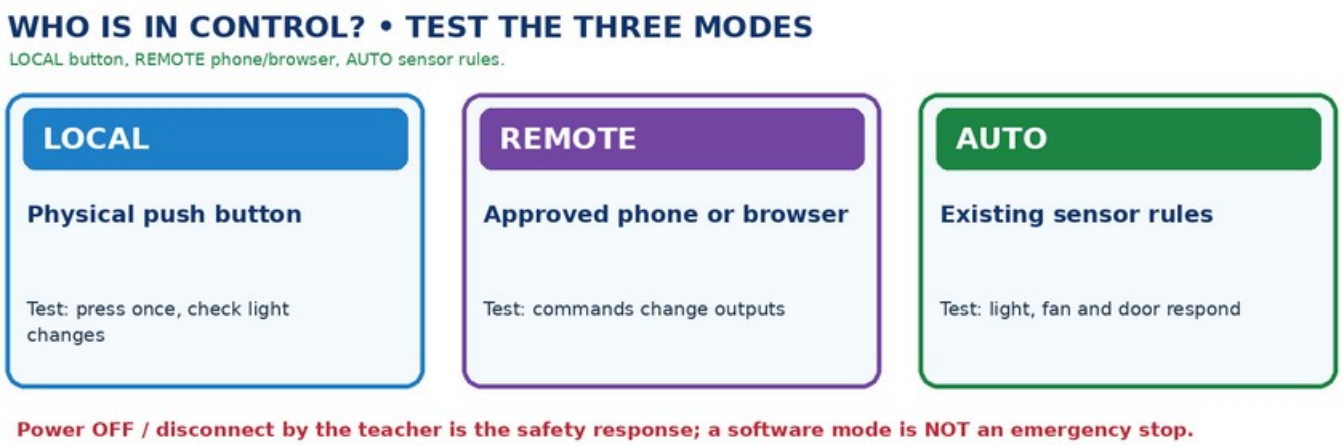
THINK Why should we test only one feature at a time before giving a full SMARTSPACE demonstration?

_____

ACTIVITY 10 • STEP 5

Observe the Three Modes

Who decides? Human • Remote command • Sensor rule



Test	Expected observation	Actual ✓ / ✗
LOCAL button	Physical button changes ROOM LIGHT 1	☐ ✓ ☐ ✗
REMOTE via Bluetooth	Approved phone command changes allowed output	☐ ✓ ☐ ✗
REMOTE via browser	Private classroom page shows status / accepts allowed command	☐ ✓ ☐ ✗
AUTO: bright/shade	Verified LDR rules control light with hysteresis	☐ ✓ ☐ ✗
AUTO: motion	PIR event causes a brief safe door/alert response	☐ ✓ ☐ ✗
Mode transition	Output controls obey selected mode; no unexpected movement	☐ ✓ ☐ ✗

Record one unexpected result

Mode: _____ Command / trigger: _____ What happened: _____

THINK What should happen when someone sends a REMOTE output command while the system is in AUTO?

_____

⚠ **SAFETY • RED CHECK** Never use software modes as an emergency stop. Teacher-supervised physical power disconnect remains the safety action.

ACTIVITY 10 • STEP 6

Run the Full SMARTSPACE Test

Put our parts together — collect evidence from the SAME build

Final system checklist (one row per capability)

Feature	What we tested	Result
ROOM LIGHT 1	LOCAL button, REMOTE and AUTO	☐ Pass ☐ Retry
LIGHT SENSOR	Shaded vs bright produces different actual readings	☐ Pass ☐ Retry
ENVIRONMENT	DHT11 reads plausible temperature/humidity	☐ Pass ☐ Retry
MOTION + DOOR	PIR event / servo action tested safely	☐ Pass ☐ Retry
MODEL FAN	Motor driver responds to allowed commands/rules	☐ Pass ☐ Retry
BUZZER	Short alert sounds and stops	☐ Pass ☐ Retry
LCD	Status text is readable and matches tested state	☐ Pass ☐ Retry
BLUETOOTH	Known device pairs / permitted commands work	☐ Pass ☐ Retry
WI-FI DASHBOARD	Local status and REMOTE controls work	☐ Pass ☐ Retry
MODE CONTROL	LOCAL, REMOTE and AUTO do not fight	☐ Pass ☐ Retry

Engineer's observation

The feature we trust most is _____ because our evidence was _____.

The feature that still needs work is _____ because _____.

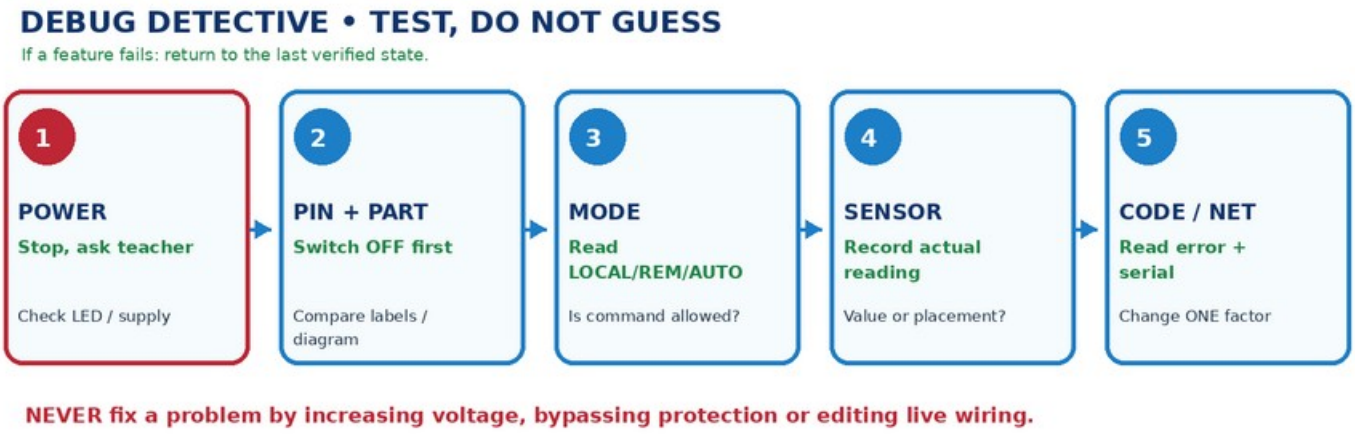
THINK If all features work individually but fail when combined, what should your team check first?

_____

ACTIVITY 10 • STEP 7

Debug Detective — Fix One Fault

A reliable engineer finds evidence before changing anything



Try this real-world fault card

FAULT SCENARIO Your browser shows the light is ON, but the LED remains OFF. Do not assume the website or the LED is broken. Check mode, observed output state, power, LED direction and the known working circuit.

1 Write the symptom. Describe what you observed without guessing the cause.

2 Check the mode. Are you in REMOTE when sending an output command? Did LOCAL button or AUTO take priority?

3 Look for evidence. Check safe serial/status data and the unchanged connection map; ask the teacher before touching wiring.

4 Change one thing. Only after the teacher identifies an approved adjustment, retest the same symptom.

Symptom observed	Possible cause to check	One safe retest
_____	_____	_____
_____	_____	_____

THINK Why is testing the LED with a known working program more useful than swapping several wires randomly?

_____

⚠ **SAFETY • RED CHECK** Power OFF before any connection change. Never bypass a logic-level converter or motor driver to “solve” a fault.

Can Our SMARTSPACE Help Save Resources?

Show evidence — do not exaggerate

Use your Activity 9 AI training, gesture-demonstration and fair-test evidence

Your final demonstration should include a recorded comparison of ROOM LIGHT 1 ON-time under two tested control strategies. Explain the conditions you kept the same and whether the improved rule still provides useful light.

Evidence	What our team recorded
LOCAL trial: light minutes ON	_____ minutes
AUTO trial: light minutes ON	_____ minutes
After selected rule adjustment	_____ minutes
Did the model provide light when needed?	☐ Yes ☐ No ☐ Needs retest
What was NOT measured?	Actual watt-hours / costs without a suitable meter

How to make a responsible claim

GOOD CLAIM: “In our 8-minute classroom test, the light was ON ___ fewer minutes.” NOT SUPPORTED: “Our classroom used 50% less electricity,” unless suitable energy measurements justify it.

THINK Could lower LED ON-time ever make SMARTSPACE less useful? Explain using comfort, safety or task needs.

_____

GREEN SKILLS Use only the materials already purchased; repair and reuse components where practical. Share evidence, limitations and responsible design choices.

ACTIVITY 10 • STEP 9

Present Your Completed SMARTSPACE

Tell the story of the one model your team built

FINAL SHOWCASE • STUDENTS DEMONSTRATE THEIR OWN MODEL

Each learner has a role, and every claim needs a test or observation.



BUILD LEAD
Explain wiring, model and power checks

CODE LEAD
Demonstrate modes, code and dashboard

EVIDENCE LEAD
Read logs, test results and improvements

Our three-minute live demonstration

1 **Introduce the need.** "We built SMARTSPACE to explore connected, useful and responsible classroom technology."

2 **Show the brain.** Point out ESP32, three inputs, outputs, low-voltage power and the local network.

3 **Demonstrate evidence.** Operate LOCAL light, one REMOTE command and one safe AUTO sensor response.

4 **Explain a limitation.** Describe one failed trial, network outage, sensor ambiguity or energy claim that needs more evidence.

5 **Show your improvement.** Explain the one chosen change, why you made it and how your test supported the result.

PRO PRESENTATION Show actual project performance. Use the illustration as a guide, never as proof that your exact build has been tested.

THINK Which single observation would convince visitors that SMARTSPACE responds to its surroundings?

_____

ACTIVITY 10 • ASSESS AND IMPROVE

My SMARTSPACE Project Review

Self-check • Teamwork • 4CG • Evidence

What we did	Evidence or example	Our rating
Designed and built one continuous model	_____	☐ Ready ☐ Improving
Used materials and tools safely	_____	☐ Ready ☐ Improving
Read sensor data and reasoned about rules	_____	☐ Ready ☐ Improving
Tested and debugged with observations	_____	☐ Ready ☐ Improving
Communicated decisions with teammates	_____	☐ Ready ☐ Improving
Discussed responsible networks and energy	_____	☐ Ready ☐ Improving

4CG learning evidence

CG-1 ☐ CG-2 ☐ CG-3 ☐ CG-4 ☐ | THINK • CREATE • COLLABORATE • COMMUNICATE

Our team reflections

One feature we are proud of: _____

One safe improvement for a future version: _____

One task I contributed: _____

TEACHER OBSERVATION Judge the process, evidence, safe practice and explanation—not just whether every device happened to work during the show.

THINK Which part of building SMARTSPACE required the most teamwork? What did you do together?

_____

ACTIVITY 10 • THINK AND REPORT

Five Final Thinking Questions

Explain your final project — do not just name components

1. How does information travel from an LDR or PIR sensor through the ESP32 to an output?

_____

_____

2. What is the difference between LOCAL, REMOTE and AUTO control? Give an example from your own model.

_____

_____

3. If Wi-Fi fails during your demonstration, which features can still work and what must you check?

_____

_____

4. What was one fault your team investigated? Describe the evidence and the safe fix or next test.

_____

_____

5. How would you improve SMARTSPACE for a real school while protecting privacy, safety and energy use?

_____

_____

My Engineer's Log — Last entry

The system we built	Evidence we collected	What we would improve
_____	_____	_____
_____	_____	_____

Student signature: _____ **Teacher verification:** _____ **Date:** _____

ACTIVITY 10 • OUR FINAL BUILD

Mission Complete — One SMARTSPACE!

THINK • BUILD • CONNECT • INNOVATE

What our team accomplished

We did not make ten separate gadgets. We progressively created one mini SMARTSPACE: first light, then Bluetooth and Wi-Fi, then sensors, outputs, automation, local and remote controls, evidence-based improvement, and a final working demonstration.

OUR FINAL CLAIM We can explain how the connected system works, show our real test results and describe what we would improve. Our model is a supervised classroom prototype—not an unrestricted smart building.

TEAM NAME: _____ **SCHOOL:** _____ **GRADE: 8**

TEACHER INITIALS: _____ **COMPLETION DATE:** _____

INNOVATE NEXT: What one helpful, safe feature would you propose for a future SMARTSPACE?

_____

⚠ **FINAL SAFETY** Switch OFF and disconnect power, return components and tools, store the model safely, and remove classroom Wi-Fi credentials from any files you plan to share.

ACTIVITY 10 FINISHED • OUR MODEL AT THE END OF THIS ACTIVITY

THE COMPLETED SMARTSPACE • SEE THE WHOLE SYSTEM

Illustrative model view. Match locations and wiring to your actual classroom build.



CODE ANNEXURE

One SMARTSPACE ● Integrated ESP32 programmes ● Teacher-supervised uploads

Student instructions come first. When an activity says “CODE POINTER”, turn to the corresponding numbered annexure below. Each Arduino listing is a separate milestone version of the SAME cumulative SMARTSPACE model; do not paste several full sketches into one file. The current tested classroom version should be saved under a new name before any upgrade.

Before uploading: the teacher checks pin labels, power supply, libraries, connected board/port, network settings, and the physical safety of outputs. These listings are teaching references and require actual-kit verification before classroom deployment.

CODE REF	USE THIS AT THE ACTIVITY STEP
A1	Activity 1 — ESP32 and blinking light
A2	Activity 2 — Bluetooth light
A3	Activity 3 — Wi-Fi status page
A4	Activity 4 — Sensor readings
A5	Activity 5 — Door, fan, buzzer and LCD
A6	Activity 6 — Automatic control rules
A7	Activity 7 — IoT dashboard
A8	Activity 8 — LOCAL, REMOTE and AUTO control
A9	Activities 9–10 — AI-compatible final ESP32 reference
G	Activity 9 — teacher-run gesture bridge (Python)

ANNEXURE A1 | Activity 1 — ESP32 and blinking light

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
const int SMART_LIGHT = 23;

void setup() {
  pinMode(SMART_LIGHT, OUTPUT);
}

void loop() {
  digitalWrite(SMART_LIGHT, HIGH);  // ON
  delay(1000);                      // 1 second
  digitalWrite(SMART_LIGHT, LOW);   // OFF
  delay(1000);                      // 1 second
}
```

ANNEXURE A2 | Activity 2 — Bluetooth light

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
const int SMART_LIGHT = 23;

void setup() {
  pinMode(SMART_LIGHT, OUTPUT);
  digitalWrite(SMART_LIGHT, LOW);
  Serial.begin(115200);
  SerialBT.begin("MMX_SMARTSPACE");
}

void loop() {
  if (SerialBT.available()) {
    char cmd = SerialBT.read();
    if (cmd == '1') {
      digitalWrite(SMART_LIGHT, HIGH);
      SerialBT.println("Light ON");
    } else if (cmd == '0') {
      digitalWrite(SMART_LIGHT, LOW);
      SerialBT.println("Light OFF");
    }
  }
}
```

ANNEXURE A3 | Activity 3 — Wi-Fi status page

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"

BluetoothSerial BT;
WebServer web(80);
const int ROOM_LIGHT = 23;
const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";

void statusPage() {
  String state = digitalRead(ROOM_LIGHT) ? "ON" : "OFF";
  web.send(200, "text/plain",
    String("SMARTSPACE ONLINE\nROOM LIGHT 1: ") + state);
}

void setup() {
  pinMode(ROOM_LIGHT, OUTPUT);
  digitalWrite(ROOM_LIGHT, LOW);
  Serial.begin(115200);
  BT.begin("MMX_SMARTSPACE");
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_NAME, WIFI_PASS);
  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED &&
         millis() - start < 15000) delay(300);
  if (WiFi.status() == WL_CONNECTED) {
    web.on("/", statusPage);
    web.begin();
    Serial.print("OPEN: http://");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("No Wi-Fi; Bluetooth still works");
  }
}

void loop() {
  if (BT.available()) {
    char cmd = BT.read();
    if (cmd == '1') {
      digitalWrite(ROOM_LIGHT, HIGH);
      BT.println("Light ON");
    } else if (cmd == '0') {
      digitalWrite(ROOM_LIGHT, LOW);
      BT.println("Light OFF");
    }
  }
  if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE A4 | Activity 4 — Sensor readings

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
// SMARTSPACE Activity 4 - one model, added senses
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"
#include <DHT.h>

BluetoothSerial BT;
WebServer web(80);
const int ROOM_LIGHT = 23;  // same LED as Activities 1-3
const int LDR_AO = 34;      // light-sensor analog output
const int PIR_OUT = 26;     // verified 3.3V digital output
const int DHT_PIN = 27;     // temperature / humidity
DHT dht(DHT_PIN, DHT11);

const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";
int lightValue = 0;
bool motionNow = false;
float temperatureC = NAN, humidityPct = NAN;
unsigned long lastRead = 0;

void readSensors() {
  lightValue = analogRead(LDR_AO);
  motionNow = (digitalRead(PIR_OUT) == HIGH);
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (!isnan(t)) temperatureC = t;
  if (!isnan(h)) humidityPct = h;
  Serial.print("Light ADC: "); Serial.print(lightValue);
  Serial.print("  Motion: "); Serial.print(motionNow);
  Serial.print("  Temp C: "); Serial.print(temperatureC);
  Serial.print("  Humidity %: "); Serial.println(humidityPct);
}

void statusPage() {
  String page = "SMARTSPACE - SENSOR REPORT\n";
  page += "ROOM LIGHT 1: ";
  page += (digitalRead(ROOM_LIGHT) ? "ON" : "OFF");
  page += "\nLight ADC (0-4095): " + String(lightValue);
  page += "\nMotion: " + String(motionNow ? "YES" : "NO");
  page += "\nTemperature C: ";
  if (isnan(temperatureC)) page += "warming up";
  else page += String(temperatureC, 1);
  page += "\nHumidity %: ";
  if (isnan(humidityPct)) page += "warming up";
  else page += String(humidityPct, 1);
  page += "\nRefresh this page for a new reading.";
  web.send(200, "text/plain", page);
}

void setup() {
  pinMode(ROOM_LIGHT, OUTPUT);
  digitalWrite(ROOM_LIGHT, LOW);
  pinMode(PIR_OUT, INPUT);
  analogReadResolution(12);
  Serial.begin(115200);
  dht.begin();
  BT.begin("MMX_SMARTSPACE");
  web.on("/", statusPage);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_NAME, WIFI_PASS);
  unsigned long started = millis();
  while (WiFi.status() != WL_CONNECTED &&
         millis() - started < 15000) delay(300);
  if (WiFi.status() == WL_CONNECTED) {
    web.begin();
    Serial.print("Open locally: http://");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("Wi-Fi unavailable; Bluetooth still works");
  }
}

void loop() {
```

```
 if (BT.available()) {
   char command = BT.read();
   if (command == '1') {
     digitalWrite(ROOM_LIGHT, HIGH);
     BT.println("Light ON");
   } else if (command == '0') {
     digitalWrite(ROOM_LIGHT, LOW);
     BT.println("Light OFF");
   }
 }
 if (millis() - lastRead >= 2500) {
   lastRead = millis();
   readSensors();
 }
 if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE A5 | Activity 5 — Door, fan, buzzer and LCD

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
// SMARTSPACE Activity 5 - ONE integrated model
// Board: original ESP32-WROOM-32 (Classic Bluetooth)
// Libraries: DHT sensor library, ESP32Servo, LiquidCrystal_I2C
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>

BluetoothSerial BT;
WebServer web(80);
DHT dht(27, DHT11);
LiquidCrystal_I2C lcd(0x27, 16, 2); // Teacher checks address
Servo door;

const int LED_PIN = 23;   // Activity 1; keep connected
const int LDR_PIN = 34;   // Activity 4; analog ADC1
const int PIR_PIN = 26;   // Activity 4; 3.3V-safe signal
const int SERVO_PIN = 18;
const int FAN_A = 16;     // L9110S A-IA
const int FAN_B = 17;     // L9110S A-IB
const int BUZZER_PIN = 25;
const int CLOSED_ANGLE = 15; // calibrate gently
const int OPEN_ANGLE = 75;   // never force hinge
const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";

bool lightOn = false, doorOpen = false, fanOn = false;
bool motionNow = false, buzzerOn = false;
int lightValue = 0;
float tempC = NAN, humidPct = NAN;
unsigned long lastSensor = 0, lastScreen = 0;
unsigned long buzzerStop = 0;

void setLight(bool on) {
  lightOn = on;
  digitalWrite(LED_PIN, on ? HIGH : LOW);
}
void setDoor(bool open) {
  doorOpen = open;
  door.write(open ? OPEN_ANGLE : CLOSED_ANGLE);
}
void setFan(bool on) {
  fanOn = on;
  digitalWrite(FAN_A, on ? HIGH : LOW);
  digitalWrite(FAN_B, LOW);
}
void startBeep() {
  digitalWrite(BUZZER_PIN, HIGH);
  buzzerOn = true;
  buzzerStop = millis() + 150;
}
void readSensors() {
  lightValue = analogRead(LDR_PIN);
  motionNow = digitalRead(PIR_PIN) == HIGH;
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (!isnan(t)) tempC = t;
  if (!isnan(h)) humidPct = h;
  Serial.printf("Light:%d PIR:%d T:%.1f H:%.1f\n",
                lightValue, motionNow, tempC, humidPct);
}

void statusPage() {
  String s = "SMARTSPACE: ACTIVITY 5\n";
  s += String("Light: ") + (lightOn ? "ON" : "OFF");
  s += String("\nDoor: ") + (doorOpen ? "OPEN" : "SHUT");
  s += String("\nFan: ") + (fanOn ? "ON" : "OFF");
  s += "\nLDR ADC: " + String(lightValue);
  s += String("\nMotion: ") + (motionNow ? "YES" : "NO");
  s += "\nTemperature C: " + String(tempC, 1);
  s += "\nHumidity %: " + String(humidPct, 1);
```

```
  s += "\nRefresh for the latest status.";
  web.send(200, "text/plain", s);
}
void updateDisplay() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(String("D:") + (doorOpen ? "OPEN" : "SHUT") +
          " F:" + (fanOn ? "ON" : "OFF"));
  lcd.setCursor(0, 1);
  if (isnan(tempC) || isnan(humidPct)) {
    lcd.print("Sensors warming");
  } else {
    lcd.print("T:" + String(tempC, 0) + "C H:" +
            String(humidPct, 0) + "%");
  }
}
void readPhone() {
  if (!BT.available()) return;
  char c = BT.read(); // one-character commands
  switch (c) {
    case '1': setLight(true);  BT.println("Light ON"); break;
    case '0': setLight(false); BT.println("Light OFF"); break;
    case 'D': setDoor(true);  BT.println("Door OPEN"); break;
    case 'd': setDoor(false); BT.println("Door SHUT"); break;
    case 'F': setFan(true);   BT.println("Fan ON"); break;
    case 'f': setFan(false);  BT.println("Fan OFF"); break;
    case 'B': startBeep();    BT.println("Alert TEST"); break;
    default: break; // ignore newline/unknown commands
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);  setLight(false);
  pinMode(PIR_PIN, INPUT);
  pinMode(FAN_A, OUTPUT);   pinMode(FAN_B, OUTPUT);
  setFan(false);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);
  analogReadResolution(12);
  dht.begin();
  door.setPeriodHertz(50);
  door.attach(SERVO_PIN);
  setDoor(false);
  Wire.begin(21, 22); // SDA, SCL through level shifter
  lcd.init(); lcd.backlight();
  lcd.setCursor(0, 0); lcd.print("SMARTSPACE READY");
  BT.begin("MMX_SMARTSPACE");
  web.on("/", statusPage);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_NAME, WIFI_PASS);
  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED &&
        millis() - start < 15000) delay(250);
  if (WiFi.status() == WL_CONNECTED) {
    web.begin();
    Serial.print("Local status: http://");
    Serial.println(WiFi.localIP());
  } else Serial.println("Wi-Fi offline; Bluetooth still works");
}

void loop() {
  readPhone();
  if (buzzerOn && (long)(millis() - buzzerStop) >= 0) {
    digitalWrite(BUZZER_PIN, LOW); buzzerOn = false;
  }
  if (millis() - lastSensor >= 2500) {
    lastSensor = millis(); readSensors();
  }
  if (millis() - lastScreen >= 2500) {
    lastScreen = millis(); updateDisplay();
  }
  if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE A6 | Activity 6 — Automatic control rules

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
// MINDMETRIX LABS | SMARTSPACE Activity 6
// Same ESP32-WROOM-32 project from Activities 1-5.
// New: MANUAL/AUTO modes + tested IF/THEN automation.
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>

BluetoothSerial BT;
WebServer web(80);
DHT dht(27, DHT11);
LiquidCrystal_I2C lcd(0x27, 16, 2); // Verify address!
Servo door;

const int LED_PIN = 23;
const int LDR_PIN = 34; // ADC1 works with Wi-Fi
const int PIR_PIN = 26;
const int SERVO_PIN = 18;
const int FAN_A = 16;
const int FAN_B = 17;
const int BUZZER_PIN = 25;
const int CLOSED_ANGLE = 15; // Adjust gently
const int OPEN_ANGLE = 75;

// STUDENTS MODIFY THESE AFTER MEASURING:
const bool DARK_WHEN_LOW = true; // Change if your LDR is opposite
const int LDR_LOW_LIMIT = 1600;  // Example, NOT universal
const int LDR_HIGH_LIMIT = 2000;
const float TEMP_FAN_ON_C = 29.0;
const float TEMP_FAN_OFF_C = 27.0;
const unsigned long DOOR_OPEN_MS = 4000;
const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";

bool autoMode = false; // Start safely in MANUAL mode
bool lightOn = false, doorOpen = false, fanOn = false;
bool motionNow = false, previousMotion = false;
bool buzzerOn = false, doorTimerActive = false;
int lightValue = 0;
float tempC = NAN, humidPct = NAN;
unsigned long lastSensor = 0, lastScreen = 0;
unsigned long buzzerStop = 0, doorCloseAt = 0;

void setLight(bool on) {
  lightOn = on;
  digitalWrite(LED_PIN, on ? HIGH : LOW);
}
void setDoor(bool open) {
  doorOpen = open;
  door.write(open ? OPEN_ANGLE : CLOSED_ANGLE);
}
void setFan(bool on) {
  fanOn = on;
  digitalWrite(FAN_A, on ? HIGH : LOW);
  digitalWrite(FAN_B, LOW);
}
void stopBeep() {
  digitalWrite(BUZZER_PIN, LOW);
  buzzerOn = false;
}
void startBeep() {
  digitalWrite(BUZZER_PIN, HIGH);
  buzzerOn = true;
  buzzerStop = millis() + 150;
}

void selectMode(bool enableAuto) {
  autoMode = enableAuto;
  doorTimerActive = false;
  previousMotion = motionNow; // Do not trigger on old motion
  setDoor(false); // Leave door closed while switching
```

```
  setFan(false);
  setLight(false);
  stopBeep();
  BT.println(autoMode ? "AUTO mode" : "MANUAL mode");
}
void readSensors() {
  lightValue = analogRead(LDR_PIN);
  motionNow = digitalRead(PIR_PIN) == HIGH;
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (!isnan(t)) tempC = t;
  if (!isnan(h)) humidPct = h;
  Serial.printf("LDR:%d PIR:%d T:%.1f H:%.1f\n",
                lightValue, motionNow, tempC, humidPct);
}

void runAutomaticRules() {
  // RULE 1: light sensor controls existing ROOM LIGHT 1.
  bool darkEnough = DARK_WHEN_LOW
      ? (lightValue <= LDR_LOW_LIMIT)
      : (lightValue >= LDR_HIGH_LIMIT);
  bool brightEnough = DARK_WHEN_LOW
      ? (lightValue >= LDR_HIGH_LIMIT)
      : (lightValue <= LDR_LOW_LIMIT);
  if (darkEnough) setLight(true);
  else if (brightEnough) setLight(false);
  // Between limits, keep the previous state.

  // RULE 2: fan uses two temperature thresholds.
  if (!isnan(tempC)) {
    if (tempC >= TEMP_FAN_ON_C) setFan(true);
    else if (tempC <= TEMP_FAN_OFF_C) setFan(false);
  }

  // RULE 3: one movement event = short door + beep.
  if (motionNow && !previousMotion) {
    setDoor(true);
    startBeep();
    doorCloseAt = millis() + DOOR_OPEN_MS;
    doorTimerActive = true;
  }
  previousMotion = motionNow;
}
void statusPage() {
  String s = "SMARTSPACE ACTIVITY 6\n";
  s += String("MODE: ") + (autoMode ? "AUTO" : "MANUAL");
  s += String("\nLight: ") + (lightOn ? "ON" : "OFF");
  s += String("\nDoor: ") + (doorOpen ? "OPEN" : "SHUT");
  s += String("\nFan: ") + (fanOn ? "ON" : "OFF");
  s += "\nLDR ADC: " + String(lightValue);
  s += String("\nMotion: ") + (motionNow ? "YES" : "NO");
  s += "\nTemp C: " + String(tempC, 1);
  s += "\nHumidity %: " + String(humidPct, 1);
  s += "\nRefresh to see a new reading.";
  web.send(200, "text/plain", s);
}
void updateDisplay() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(autoMode ? "AUTO " : "MAN  ");
  lcd.print(lightOn ? "L1 " : "L0 ");

  lcd.print(fanOn ? "F1" : "F0");
  lcd.setCursor(0, 1);
  if (isnan(tempC) || isnan(humidPct)) {
    lcd.print("Sensor warming");
  } else {
    lcd.print("T:" + String(tempC, 0) + "C H:" +
              String(humidPct, 0) + "%");
  }
}
void readPhone() {
  if (!BT.available()) return;
  char c = BT.read();
  if (c == 'A') { selectMode(true); return; }
  if (c == 'M') { selectMode(false); return; }
  if (c == 'S') {
```

```
  BT.println(autoMode ? "MODE AUTO" : "MODE MANUAL");
  return;
 }
 if (autoMode) {
  BT.println("AUTO active; send M for manual controls");
  return;
 }
 switch (c) { // Existing Activity 5 commands
  case '1': setLight(true);  break;
  case '0': setLight(false); break;
  case 'D': setDoor(true);   break;
  case 'd': setDoor(false);  break;
  case 'F': setFan(true);    break;
  case 'f': setFan(false);   break;
  case 'B': startBeep();     break;
  default: return;
 }
 BT.println("MANUAL command received");
}
void setup() {
 Serial.begin(115200);
 pinMode(LED_PIN, OUTPUT); setLight(false);
 pinMode(PIR_PIN, INPUT);
 pinMode(FAN_A, OUTPUT); pinMode(FAN_B, OUTPUT);
 setFan(false);
 pinMode(BUZZER_PIN, OUTPUT); stopBeep();
 analogReadResolution(12);
 dht.begin();
 door.setPeriodHertz(50);
 door.attach(SERVO_PIN);

 setDoor(false);
 Wire.begin(21, 22); // I2C via verified level shifter
 lcd.init(); lcd.backlight();
 lcd.print("SMARTSPACE READY");
 BT.begin("MMX_SMARTSPACE");
 web.on("/", statusPage);
 WiFi.mode(WIFI_STA);
 WiFi.begin(WIFI_NAME, WIFI_PASS);
 unsigned long started = millis();
 while (WiFi.status() != WL_CONNECTED &&
        millis() - started < 15000) delay(250);
 if (WiFi.status() == WL_CONNECTED) {
  web.begin();
  Serial.print("Local status: http://");
  Serial.println(WiFi.localIP());
 } else Serial.println("Wi-Fi offline; Bluetooth still works");
}
void loop() {
 readPhone();
 if (buzzerOn && (long)(millis() - buzzerStop) >= 0)
  stopBeep();
 if (doorTimerActive &&
     (long)(millis() - doorCloseAt) >= 0) {
  setDoor(false);
  doorTimerActive = false;
 }
 if (millis() - lastSensor >= 2500) {
  lastSensor = millis();
  readSensors();
  if (autoMode) runAutomaticRules();
 }
 if (millis() - lastScreen >= 2500) {
  lastScreen = millis(); updateDisplay();
 }
 if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE A7 | Activity 7 — IoT dashboard

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
// MINDMETRIX LABS | SMARTSPACE Activity 7
// Same ESP32-WROOM-32 project from Activities 1-5.
// New: local authenticated browser dashboard. Existing sensors and rules stay.
// CLASSROOM DEMO ONLY: HTTP Basic authentication is not encrypted.
// Use an isolated teacher-approved 2.4 GHz network. Never port-forward.
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>

BluetoothSerial BT;
WebServer web(80);
DHT dht(27, DHT11);
LiquidCrystal_I2C lcd(0x27, 16, 2); // Verify address!
Servo door;

const int LED_PIN = 23;
const int LDR_PIN = 34; // ADC1 works with Wi-Fi
const int PIR_PIN = 26;
const int SERVO_PIN = 18;
const int FAN_A = 16;
const int FAN_B = 17;
const int BUZZER_PIN = 25;
const int CLOSED_ANGLE = 15; // Adjust gently
const int OPEN_ANGLE = 75;

// STUDENTS MODIFY THESE AFTER MEASURING:
const bool DARK_WHEN_LOW = true; // Change if your LDR is opposite
const int LDR_LOW_LIMIT = 1600;  // Example, NOT universal
const int LDR_HIGH_LIMIT = 2000;
const float TEMP_FAN_ON_C = 29.0;
const float TEMP_FAN_OFF_C = 27.0;
const unsigned long DOOR_OPEN_MS = 4000;
const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";
// Teacher sets these privately before uploading; NEVER print them.
// Basic authentication limits casual access on an isolated lab network;
// it does NOT encrypt data. Not suitable for public / production use.
const char* DASH_USER = "teacher";
const char* DASH_PASS = "CHANGE_ME_IN_CLASS";

bool autoMode = false; // Start safely in MANUAL mode
bool lightOn = false, doorOpen = false, fanOn = false;
bool motionNow = false, previousMotion = false;
bool buzzerOn = false, doorTimerActive = false;
int lightValue = 0;
float tempC = NAN, humidPct = NAN;
unsigned long lastSensor = 0, lastScreen = 0;
unsigned long buzzerStop = 0, doorCloseAt = 0;

void setLight(bool on) {
  lightOn = on;
  digitalWrite(LED_PIN, on ? HIGH : LOW);
}
void setDoor(bool open) {
  doorOpen = open;
  door.write(open ? OPEN_ANGLE : CLOSED_ANGLE);
}
void setFan(bool on) {
  fanOn = on;
  digitalWrite(FAN_A, on ? HIGH : LOW);
  digitalWrite(FAN_B, LOW);
}
void stopBeep() {
  digitalWrite(BUZZER_PIN, LOW);
  buzzerOn = false;
}
void startBeep() {
  digitalWrite(BUZZER_PIN, HIGH);
  buzzerOn = true;
  buzzerStop = millis() + 150;
```

```
}

void selectMode(bool enableAuto) {
  autoMode = enableAuto;
  doorTimerActive = false;
  previousMotion = motionNow; // Do not trigger on old motion
  setDoor(false); // Leave door closed while switching
  setFan(false);
  setLight(false);
  stopBeep();
  BT.println(autoMode ? "AUTO mode" : "MANUAL mode");
}
void readSensors() {
  lightValue = analogRead(LDR_PIN);
  motionNow = digitalRead(PIR_PIN) == HIGH;
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (!isnan(t)) tempC = t;
  if (!isnan(h)) humidPct = h;
  Serial.printf("LDR:%d PIR:%d T:%.1f H:%.1f\n",
                lightValue, motionNow, tempC, humidPct);
}

void runAutomaticRules() {
  // RULE 1: light sensor controls existing ROOM LIGHT 1.
  bool darkEnough = DARK_WHEN_LOW
      ? (lightValue <= LDR_LOW_LIMIT)
      : (lightValue >= LDR_HIGH_LIMIT);
  bool brightEnough = DARK_WHEN_LOW
      ? (lightValue >= LDR_HIGH_LIMIT)
      : (lightValue <= LDR_LOW_LIMIT);
  if (darkEnough) setLight(true);
  else if (brightEnough) setLight(false);
  // Between limits, keep the previous state.

  // RULE 2: fan uses two temperature thresholds.
  if (!isnan(tempC)) {
    if (tempC >= TEMP_FAN_ON_C) setFan(true);
    else if (tempC <= TEMP_FAN_OFF_C) setFan(false);
  }

  // RULE 3: one movement event = short door + beep.
  if (motionNow && !previousMotion) {
    setDoor(true);
    startBeep();
    doorCloseAt = millis() + DOOR_OPEN_MS;
    doorTimerActive = true;
  }
  previousMotion = motionNow;
}
// A complete local dashboard: one existing SMARTSPACE, no new wires.
const char DASH_HTML[] PROGMEM = R"HTML(
<!doctype html><html lang="en"><head><meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>SMARTSPACE | Local Lab</title>
<style>
body{font-family:Arial,sans-serif;background:#f3f7fa;color:#103065;
 max-width:680px;margin:20px auto;padding:0 14px}
h1{margin-bottom:4px}small{color:#567}
.card{background:white;padding:16px;border-radius:12px;
 border:1px solid #cadae5;margin:12px 0}
.metric{display:inline-block;margin:8px 12px 8px 0}
button{padding:10px 15px;margin:5px;border:0;border-radius:8px;
 background:#145b94;color:white;font-weight:bold}
button.mode{background:#287a44}
#msg{color:#287a44;font-weight:bold}
</style></head><body>
<h1>SMARTSPACE Control Centre</h1>
<small>MindMetrix Labs | LOCAL classroom network only</small>
<div class="card"><h2>Live status</h2><div id="mode">Connecting...</div>
<div id="data">Waiting for readings...</div>
<small>Refreshes about every 3 seconds</small></div>
<div class="card"><h2>Choose a mode</h2>
<form action="/do" method="post">
<button class="mode" name="c" value="M">MANUAL</button>
<button class="mode" name="c" value="A">AUTO</button>
</form></div>
```

```
<div class="card"><h2>Manual controls</h2>
<p>These commands work only in MANUAL mode.</p>
<form action="/do" method="post">
<button name="c" value="1">Light ON</button>
<button name="c" value="0">Light OFF</button><br>
<button name="c" value="D">Door OPEN</button>
<button name="c" value="d">Door SHUT</button><br>
<button name="c" value="F">Fan ON</button>
<button name="c" value="f">Fan OFF</button>
<button name="c" value="B">Short beep</button>
</form></div>
<p id="msg">Teacher-supervised prototype: no real access control.</p>
<script>
async function refresh(){try{
 const r=await fetch('/status',{cache:'no-store'});
 if(!r.ok)throw Error('status'); const s=await r.json();
 document.getElementById('mode').textContent='MODE: '+s.mode;
 document.getElementById('data').textContent=
 'Light '+s.light+' | Door '+s.door+' | Fan '+s.fan+
 ' | LDR '+s.ldr+' | Motion '+s.motion+
 ' | Temp '+(s.temp===null?'--':s.temp+' C')+
 ' | Humidity '+(s.humidity===null?'--':s.humidity+' %');
 }catch(e){document.getElementById('mode').textContent='Check network';}}
refresh();setInterval(refresh,3000);
</script></body></html>
)HTML";

bool authorized() {
  if (web.authenticate(DASH_USER, DASH_PASS)) return true;
  web.requestAuthentication();
  return false;
}
void dashboardPage() {
  if (!authorized()) return;
  web.sendHeader("Cache-Control", "no-store");
  web.send_P(200, "text/html", DASH_HTML);
}
void dashboardStatus() {
  if (!authorized()) return;
  String j = "{\"mode\":\"" + String(autoMode ? "AUTO" : "MANUAL") + "\"";
  j += ",\"light\":\"" + String(lightOn ? "ON" : "OFF") + "\"";
  j += ",\"door\":\"" + String(doorOpen ? "OPEN" : "SHUT") + "\"";
  j += ",\"fan\":\"" + String(fanOn ? "ON" : "OFF") + "\"";
  j += ",\"ldr\":" + String(lightValue);
  j += ",\"motion\":\"" + String(motionNow ? "YES" : "NO") + "\"";
  j += ",\"temp\":";
  if (isnan(tempC)) j += "null"; else j += String(tempC, 1);
  j += ",\"humidity\":";
  if (isnan(humidPct)) j += "null"; else j += String(humidPct, 1);
  j += "}";
  web.sendHeader("Cache-Control", "no-store");
  web.send(200, "application/json", j);
}
void dashboardAction() {
  if (!authorized()) return;
  String cmd = web.arg("c");
  if (cmd.length() != 1) {
    web.send(400, "text/plain", "One command only"); return;
  }
  char c = cmd.charAt(0);
  if (c == 'A') selectMode(true);
  else if (c == 'M') selectMode(false);
  else if (!autoMode) {
    switch (c) {
      case '1': setLight(true);  break;
      case '0': setLight(false); break;
      case 'D': setDoor(true);   break;
      case 'd': setDoor(false);  break;
      case 'F': setFan(true);    break;
      case 'f': setFan(false);   break;
      case 'B': startBeep();     break;
      default: web.send(400, "text/plain", "Unknown command"); return;
    }
  }
  // AUTO deliberately blocks direct manual control requests.
  web.sendHeader("Location", "/");
  web.send(303, "text/plain", "Return to dashboard");
```

```
}
void updateDisplay() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(autoMode ? "AUTO " : "MAN  ");
  lcd.print(lightOn ? "L1 " : "L0 ");
  lcd.print(fanOn ? "F1" : "F0");
  lcd.setCursor(0, 1);
  if (isnan(tempC) || isnan(humidPct)) {
    lcd.print("Sensor warming");
  } else {
    lcd.print("T:" + String(tempC, 0) + "C H:" +
              String(humidPct, 0) + "%");
  }
}
void readPhone() {
  if (!BT.available()) return;
  char c = BT.read();
  if (c == 'A') { selectMode(true); return; }
  if (c == 'M') { selectMode(false); return; }
  if (c == 'S') {
    BT.println(autoMode ? "MODE AUTO" : "MODE MANUAL");
    return;
  }
  if (autoMode) {
    BT.println("AUTO active; send M for manual controls");
    return;
  }
  switch (c) { // Existing Activity 5 commands
    case '1': setLight(true);  break;
    case '0': setLight(false); break;
    case 'D': setDoor(true);   break;
    case 'd': setDoor(false);  break;
    case 'F': setFan(true);    break;
    case 'f': setFan(false);   break;
    case 'B': startBeep();     break;
    default: return;
  }
  BT.println("MANUAL command received");
}
void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT); setLight(false);
  pinMode(PIR_PIN, INPUT);
  pinMode(FAN_A, OUTPUT); pinMode(FAN_B, OUTPUT);
  setFan(false);
  pinMode(BUZZER_PIN, OUTPUT); stopBeep();
  analogReadResolution(12);
  dht.begin();
  door.setPeriodHertz(50);
  door.attach(SERVO_PIN);
  setDoor(false);
  Wire.begin(21, 22); // I2C via verified level shifter
  lcd.init(); lcd.backlight();
  lcd.print("SMARTSPACE READY");
  BT.begin("MMX_SMARTSPACE");
  web.on("/", HTTP_GET, dashboardPage);
  web.on("/status", HTTP_GET, dashboardStatus);
  web.on("/do", HTTP_POST, dashboardAction);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_NAME, WIFI_PASS);
  unsigned long started = millis();
  while (WiFi.status() != WL_CONNECTED &&
         millis() - started < 15000) delay(250);
  if (WiFi.status() == WL_CONNECTED) {
    web.begin();
    Serial.print("Local dashboard: http://");
    Serial.println(WiFi.localIP());
  } else Serial.println("Wi-Fi offline; Bluetooth still works");
}
void loop() {
  readPhone();
  if (buzzerOn && (long)(millis() - buzzerStop) >= 0)
    stopBeep();
  if (doorTimerActive &&
      (long)(millis() - doorCloseAt) >= 0) {
    setDoor(false);
```

```
    doorTimerActive = false;
  }
  if (millis() - lastSensor >= 2500) {
    lastSensor = millis();
    readSensors();
    if (autoMode) runAutomaticRules();
  }
  if (millis() - lastScreen >= 2500) {
    lastScreen = millis(); updateDisplay();
  }
  if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE A8 | Activity 8 — LOCAL, REMOTE and AUTO control

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
// MINDMETRIX LABS | SMARTSPACE Activity 8 — three clear control modes
// Same ESP32-WROOM-32 project from Activities 1-5.
// New: one LOCAL push button on GPIO32; REMOTE browser/Bluetooth; AUTO sensors.
// All Activity 7 features stay on the SAME ESP32. This is a classroom prototype.
// CLASSROOM DEMO ONLY: HTTP Basic authentication is not encrypted.
// Use an isolated teacher-approved 2.4 GHz network. Never port-forward.
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>

BluetoothSerial BT;
WebServer web(80);
DHT dht(27, DHT11);
LiquidCrystal_I2C lcd(0x27, 16, 2); // Verify address!
Servo door;

const int LED_PIN = 23;
const int LDR_PIN = 34; // ADC1 works with Wi-Fi
const int PIR_PIN = 26;
const int SERVO_PIN = 18;
const int FAN_A = 16;
const int FAN_B = 17;
const int BUZZER_PIN = 25;
const int LOCAL_BUTTON_PIN = 32; // Reused button between GPIO32 and GND; INPUT_PULLUP
const int CLOSED_ANGLE = 15; // Adjust gently
const int OPEN_ANGLE = 75;

// STUDENTS MODIFY THESE AFTER MEASURING:
const bool DARK_WHEN_LOW = true; // Change if your LDR is opposite
const int LDR_LOW_LIMIT = 1600;  // Example, NOT universal
const int LDR_HIGH_LIMIT = 2000;
const float TEMP_FAN_ON_C = 29.0;
const float TEMP_FAN_OFF_C = 27.0;
const unsigned long DOOR_OPEN_MS = 4000;
const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";
// Teacher sets these privately before uploading; NEVER print them.
// Basic authentication limits casual access on an isolated lab network;
// it does NOT encrypt data. Not suitable for public / production use.
const char* DASH_USER = "teacher";
const char* DASH_PASS = "CHANGE_ME_IN_CLASS";

// Exactly one source of control is selected at a time.
enum ControlMode { MODE_LOCAL, MODE_REMOTE, MODE_AUTO };
ControlMode controlMode = MODE_LOCAL; // Begin with local, outputs OFF
bool lastRawButton = HIGH;
bool stableButton = HIGH;
unsigned long buttonEdgeTime = 0;
const unsigned long DEBOUNCE_MS = 50;
const char* modeName() {
  if (controlMode == MODE_LOCAL) return "LOCAL";
  if (controlMode == MODE_REMOTE) return "REMOTE";
  return "AUTO";
}
bool lightOn = false, doorOpen = false, fanOn = false;
bool motionNow = false, previousMotion = false;
bool buzzerOn = false, doorTimerActive = false;
int lightValue = 0;
float tempC = NAN, humidPct = NAN;
unsigned long lastSensor = 0, lastScreen = 0;
unsigned long buzzerStop = 0, doorCloseAt = 0;

void setLight(bool on) {
  lightOn = on;
  digitalWrite(LED_PIN, on ? HIGH : LOW);
}
void setDoor(bool open) {
  doorOpen = open;
  door.write(open ? OPEN_ANGLE : CLOSED_ANGLE);
}
```

```
void setFan(bool on) {
  fanOn = on;
  digitalWrite(FAN_A, on ? HIGH : LOW);
  digitalWrite(FAN_B, LOW);
}
void stopBeep() {
  digitalWrite(BUZZER_PIN, LOW);
  buzzerOn = false;
}
void startBeep() {
  digitalWrite(BUZZER_PIN, HIGH);
  buzzerOn = true;
  buzzerStop = millis() + 150;
}

void selectMode(ControlMode requested) {
  // Switching modes starts from a known, calm output state.
  controlMode = requested;
  doorTimerActive = false;
  previousMotion = motionNow;
  setDoor(false);
  setFan(false);
  setLight(false);
  stopBeep();
  BT.print("MODE "); BT.println(modeName());
}
// Run this on every loop; the button works even without Wi-Fi.
void handleLocalButton() {
  bool raw = digitalRead(LOCAL_BUTTON_PIN);
  if (raw != lastRawButton) buttonEdgeTime = millis();
  lastRawButton = raw;
  if (millis() - buttonEdgeTime >= DEBOUNCE_MS && raw != stableButton) {
    stableButton = raw;
    if (stableButton == LOW) { // One confirmed button press
      if (controlMode != MODE_LOCAL) {
        selectMode(MODE_LOCAL); // Local override of REMOTE/AUTO
      }
      setLight(!lightOn);
      Serial.println("Local button: ROOM LIGHT 1 toggled");
    }
  }
}
void readSensors() {
  lightValue = analogRead(LDR_PIN);
  motionNow = digitalRead(PIR_PIN) == HIGH;
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (!isnan(t)) tempC = t;
  if (!isnan(h)) humidPct = h;
  Serial.printf("LDR:%d PIR:%d T:%.1f H:%.1f\n",
                lightValue, motionNow, tempC, humidPct);
}

void runAutomaticRules() {
  // RULE 1: light sensor controls existing ROOM LIGHT 1.
  bool darkEnough = DARK_WHEN_LOW
      ? (lightValue <= LDR_LOW_LIMIT)
      : (lightValue >= LDR_HIGH_LIMIT);
  bool brightEnough = DARK_WHEN_LOW
      ? (lightValue >= LDR_HIGH_LIMIT)
      : (lightValue <= LDR_LOW_LIMIT);
  if (darkEnough) setLight(true);
  else if (brightEnough) setLight(false);
  // Between limits, keep the previous state.

  // RULE 2: fan uses two temperature thresholds.
  if (!isnan(tempC)) {
    if (tempC >= TEMP_FAN_ON_C) setFan(true);
    else if (tempC <= TEMP_FAN_OFF_C) setFan(false);
  }

  // RULE 3: one movement event = short door + beep.
  if (motionNow && !previousMotion) {
    setDoor(true);
    startBeep();
    doorCloseAt = millis() + DOOR_OPEN_MS;
    doorTimerActive = true;
```

```
 }
  previousMotion = motionNow;
}
// A complete local dashboard: one existing SMARTSPACE, no new wires.
const char DASH_HTML[] PROGMEM = R"HTML(
<!doctype html><html lang="en"><head><meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>SMARTSPACE | Local Lab</title>
<style>
body{font-family:Arial,sans-serif;background:#f3f7fa;color:#103065;
 max-width:680px;margin:20px auto;padding:0 14px}
h1{margin-bottom:4px}small{color:#567}
.card{background:white;padding:16px;border-radius:12px;
 border:1px solid #cadae5;margin:12px 0}
.metric{display:inline-block;margin:8px 12px 8px 0}
button{padding:10px 15px;margin:5px;border:0;border-radius:8px;
 background:#145b94;color:white;font-weight:bold}
button.mode{background:#287a44}
#msg{color:#287a44;font-weight:bold}
</style></head><body>
<h1>SMARTSPACE Three Control Modes</h1>
<small>MindMetrix Labs | LOCAL classroom network only</small>
<div class="card"><h2>Live status</h2><div id="mode">Connecting...</div>
<div id="data">Waiting for readings...</div>
<small>Refreshes about every 3 seconds</small></div>
<div class="card"><h2>Choose a mode</h2>
<form action="/do" method="post">
<button class="mode" name="c" value="L">LOCAL</button>
<button class="mode" name="c" value="R">REMOTE</button>
<button class="mode" name="c" value="A">AUTO</button>
</form></div>
<div class="card"><h2>Remote controls</h2>
<p>These output buttons work only in REMOTE mode.</p>
<p>LOCAL uses the new physical button. AUTO uses sensor rules.</p>
<form action="/do" method="post">
<button name="c" value="1">Light ON</button>
<button name="c" value="0">Light OFF</button><br>
<button name="c" value="D">Door OPEN</button>
<button name="c" value="d">Door SHUT</button><br>
<button name="c" value="F">Fan ON</button>
<button name="c" value="f">Fan OFF</button>
<button name="c" value="B">Short beep</button>
</form></div>
<p id="msg">Teacher-supervised prototype: no real access control.</p>
<script>
async function refresh(){try{
 const r=await fetch('/status',{cache:'no-store'});
 if(!r.ok)throw Error('status'); const s=await r.json();
 document.getElementById('mode').textContent='MODE: '+s.mode;
 document.getElementById('data').textContent=
 'Light '+s.light+' | Door '+s.door+' | Fan '+s.fan+
 ' | LDR '+s.ldr+' | Motion '+s.motion+
 ' | Temp '+(s.temp===null?'--':s.temp+' C')+
 ' | Humidity '+(s.humidity===null?'--':s.humidity+' %');
 }catch(e){document.getElementById('mode').textContent='Check network';}}
refresh();setInterval(refresh,3000);
</script></body></html>
)HTML";

bool authorized() {
  if (web.authenticate(DASH_USER, DASH_PASS)) return true;
  web.requestAuthentication();
  return false;
}
void dashboardPage() {
  if (!authorized()) return;
  web.sendHeader("Cache-Control", "no-store");
  web.send_P(200, "text/html", DASH_HTML);
}
void dashboardStatus() {
  if (!authorized()) return;
  String j = "{\"mode\":\"" + String(modeName()) + "\"";
  j += ",\"light\":\"" + String(lightOn ? "ON" : "OFF") + "\"";
  j += ",\"door\":\"" + String(doorOpen ? "OPEN" : "SHUT") + "\"";
  j += ",\"fan\":\"" + String(fanOn ? "ON" : "OFF") + "\"";
  j += ",\"ldr\":" + String(lightValue);
  j += ",\"motion\":\"" + String(motionNow ? "YES" : "NO") + "\"";
```

```
  j += ",\"temp\":";
  if (isnan(tempC)) j += "null"; else j += String(tempC, 1);
  j += ",\"humidity\":";
  if (isnan(humidPct)) j += "null"; else j += String(humidPct, 1);
  j += "}";
  web.sendHeader("Cache-Control", "no-store");
  web.send(200, "application/json", j);
}
void dashboardAction() {
  if (!authorized()) return;
  String cmd = web.arg("c");
  if (cmd.length() != 1) {
    web.send(400, "text/plain", "One command only"); return;
  }
  char c = cmd.charAt(0);
  if (c == 'A') selectMode(MODE_AUTO);
  else if (c == 'R') selectMode(MODE_REMOTE);
  else if (c == 'L') selectMode(MODE_LOCAL);
  else if (controlMode == MODE_REMOTE) {
    switch (c) {
      case '1': setLight(true);  break;
      case '0': setLight(false); break;
      case 'D': setDoor(true);   break;
      case 'd': setDoor(false);  break;
      case 'F': setFan(true);    break;
      case 'f': setFan(false);   break;
      case 'B': startBeep();     break;
      default: web.send(400, "text/plain", "Unknown command"); return;
    }
  } else {
    web.send(409, "text/plain", "Change to REMOTE mode first");
    return;
  }
  // LOCAL/AUTO deliberately block direct remote output requests.
  web.sendHeader("Location", "/");
  web.send(303, "text/plain", "Return to dashboard");
}
void updateDisplay() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(controlMode == MODE_AUTO ? "AUTO " :
            (controlMode == MODE_LOCAL ? "LOC  " : "REM  "));
  lcd.print(lightOn ? "L1 " : "L0 ");
  lcd.print(fanOn ? "F1" : "F0");
  lcd.setCursor(0, 1);
  if (isnan(tempC) || isnan(humidPct)) {
    lcd.print("Sensor warming");
  } else {
    lcd.print("T:" + String(tempC, 0) + "C H:" +
              String(humidPct, 0) + "%");
  }
}
void readPhone() {
  if (!BT.available()) return;
  char c = BT.read();
  if (c == 'A') { selectMode(MODE_AUTO); return; }
  if (c == 'R') { selectMode(MODE_REMOTE); return; }
  if (c == 'L') { selectMode(MODE_LOCAL); return; }
  if (c == 'S') {
    BT.print("MODE "); BT.println(modeName());
    return;
  }
  if (controlMode != MODE_REMOTE) {
    BT.println("Choose R for REMOTE output commands");
    return;
  }
  switch (c) { // Existing Activity 5 output commands
    case '1': setLight(true);  break;
    case '0': setLight(false); break;
    case 'D': setDoor(true);   break;
    case 'd': setDoor(false);  break;
    case 'F': setFan(true);    break;
    case 'f': setFan(false);   break;
    case 'B': startBeep();     break;
    default: return;
  }
  BT.println("REMOTE command received");
```

```
}
void setup() {
  Serial.begin(115200);
  pinMode(LOCAL_BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT); setLight(false);
  pinMode(PIR_PIN, INPUT);
  pinMode(FAN_A, OUTPUT); pinMode(FAN_B, OUTPUT);
  setFan(false);
  pinMode(BUZZER_PIN, OUTPUT); stopBeep();
  analogReadResolution(12);
  dht.begin();
  door.setPeriodHertz(50);
  door.attach(SERVO_PIN);
  setDoor(false);
  Wire.begin(21, 22); // I2C via verified level shifter
  lcd.init(); lcd.backlight();
  lcd.print("SMARTSPACE READY");
  BT.begin("MMX_SMARTSPACE");
  web.on("/", HTTP_GET, dashboardPage);
  web.on("/status", HTTP_GET, dashboardStatus);
  web.on("/do", HTTP_POST, dashboardAction);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_NAME, WIFI_PASS);
  unsigned long started = millis();
  while (WiFi.status() != WL_CONNECTED &&
         millis() - started < 15000) delay(250);
  if (WiFi.status() == WL_CONNECTED) {
    web.begin();
    Serial.print("Local dashboard: http://");
    Serial.println(WiFi.localIP());
  } else Serial.println("Wi-Fi offline; Bluetooth still works");
}
void loop() {
  handleLocalButton(); // Physical local control, regardless of network
  readPhone();
  if (buzzerOn && (long)(millis() - buzzerStop) >= 0)
    stopBeep();
  if (doorTimerActive &&
      (long)(millis() - doorCloseAt) >= 0) {
    setDoor(false);
    doorTimerActive = false;
  }
  if (millis() - lastSensor >= 2500) {
    lastSensor = millis();
    readSensors();
    if (controlMode == MODE_AUTO) runAutomaticRules();
  }
  if (millis() - lastScreen >= 2500) {
    lastScreen = millis(); updateDisplay();
  }
  if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE A9 | Activities 9–10 — AI-compatible final ESP32 reference

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
// MINDMETRIX LABS | SMARTSPACE Activity 9 — 4-hour Open Source AI investigation
// The trained Orange model and pretrained MediaPipe recognizer run ONLY on computer.
// This firmware retains the same Activity 8 LOCAL / REMOTE / AUTO system.
// Teacher-operated gesture bridge posts only c=1/c=0 to existing REMOTE dashboard.
// SAME Activity 8 circuits, modes, Bluetooth and browser dashboard.
// Record actual LDR readings, then update threshold constants with teacher.
// No AI service is connected to the ESP32: AI only suggests ideas in class.
// Use only teacher-reviewed, non-identifying data. This code does not measure energy.
// Same ESP32-WROOM-32 project from Activities 1-5.
// New: one LOCAL push button on GPIO32; REMOTE browser/Bluetooth; AUTO sensors.
// All Activity 7 features stay on the SAME ESP32. This is a classroom prototype.
// CLASSROOM DEMO ONLY: HTTP Basic authentication is not encrypted.
// Use an isolated teacher-approved 2.4 GHz network. Never port-forward.
#include <WiFi.h>
#include <WebServer.h>
#include "BluetoothSerial.h"
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>

BluetoothSerial BT;
WebServer web(80);
DHT dht(27, DHT11);
LiquidCrystal_I2C lcd(0x27, 16, 2); // Verify address!
Servo door;

const int LED_PIN = 23;
const int LDR_PIN = 34; // ADC1 works with Wi-Fi
const int PIR_PIN = 26;
const int SERVO_PIN = 18;
const int FAN_A = 16;
const int FAN_B = 17;
const int BUZZER_PIN = 25;
const int LOCAL_BUTTON_PIN = 32; // Reused button between GPIO32 and GND; INPUT_PULLUP
const int CLOSED_ANGLE = 15; // Adjust gently
const int OPEN_ANGLE = 75;

// ACTIVITY 9 TUNING TASK: record real bright/dark readings in the workbook.
// With the teacher, adjust values below; keep valid spacing and direction.
// These are EXAMPLE thresholds, NOT readings from your SMARTSPACE:

const bool DARK_WHEN_LOW = true; // Change if your LDR is opposite
const int LDR_LOW_LIMIT = 1600;  // Example, NOT universal
const int LDR_HIGH_LIMIT = 2000;
const float TEMP_FAN_ON_C = 29.0;
const float TEMP_FAN_OFF_C = 27.0;
const unsigned long DOOR_OPEN_MS = 4000;
const char* WIFI_NAME = "CLASS_WIFI_2G";
const char* WIFI_PASS = "TEACHER_ONLY";
// Teacher sets these privately before uploading; NEVER print them.
// Basic authentication limits casual access on an isolated lab network;
// it does NOT encrypt data. Not suitable for public / production use.
const char* DASH_USER = "teacher";
const char* DASH_PASS = "CHANGE_ME_IN_CLASS";

// Exactly one source of control is selected at a time.
enum ControlMode { MODE_LOCAL, MODE_REMOTE, MODE_AUTO };
ControlMode controlMode = MODE_LOCAL; // Begin with local, outputs OFF
bool lastRawButton = HIGH;
bool stableButton = HIGH;
unsigned long buttonEdgeTime = 0;
const unsigned long DEBOUNCE_MS = 50;
const char* modeName() {
  if (controlMode == MODE_LOCAL) return "LOCAL";
  if (controlMode == MODE_REMOTE) return "REMOTE";
  return "AUTO";
}
bool lightOn = false, doorOpen = false, fanOn = false;
bool motionNow = false, previousMotion = false;
bool buzzerOn = false, doorTimerActive = false;
int lightValue = 0;
float tempC = NAN, humidPct = NAN;
unsigned long lastSensor = 0, lastScreen = 0;
```

```
unsigned long buzzerStop = 0, doorCloseAt = 0;

void setLight(bool on) {
  lightOn = on;
  digitalWrite(LED_PIN, on ? HIGH : LOW);
}
void setDoor(bool open) {
  doorOpen = open;
  door.write(open ? OPEN_ANGLE : CLOSED_ANGLE);
}
void setFan(bool on) {
  fanOn = on;
  digitalWrite(FAN_A, on ? HIGH : LOW);
  digitalWrite(FAN_B, LOW);
}
void stopBeep() {
  digitalWrite(BUZZER_PIN, LOW);
  buzzerOn = false;
}
void startBeep() {
  digitalWrite(BUZZER_PIN, HIGH);
  buzzerOn = true;
  buzzerStop = millis() + 150;
}

void selectMode(ControlMode requested) {
  // Switching modes starts from a known, calm output state.
  controlMode = requested;
  doorTimerActive = false;
  previousMotion = motionNow;
  setDoor(false);
  setFan(false);
  setLight(false);
  stopBeep();
  BT.print("MODE "); BT.println(modeName());
}
// Run this on every loop; the button works even without Wi-Fi.
void handleLocalButton() {
  bool raw = digitalRead(LOCAL_BUTTON_PIN);
  if (raw != lastRawButton) buttonEdgeTime = millis();
  lastRawButton = raw;
  if (millis() - buttonEdgeTime >= DEBOUNCE_MS && raw != stableButton) {
    stableButton = raw;
    if (stableButton == LOW) { // One confirmed button press
      if (controlMode != MODE_LOCAL) {
        selectMode(MODE_LOCAL); // Local override of REMOTE/AUTO
      }
      setLight(!lightOn);
      Serial.println("Local button: ROOM LIGHT 1 toggled");
    }
  }
}
void readSensors() {
  lightValue = analogRead(LDR_PIN);
  motionNow = digitalRead(PIR_PIN) == HIGH;
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (!isnan(t)) tempC = t;
  if (!isnan(h)) humidPct = h;
  Serial.printf("LDR:%d PIR:%d T:%.1f H:%.1f\n",
                lightValue, motionNow, tempC, humidPct);
}

void runAutomaticRules() {
  // RULE 1: light sensor controls existing ROOM LIGHT 1.
  bool darkEnough = DARK_WHEN_LOW
      ? (lightValue <= LDR_LOW_LIMIT)
      : (lightValue >= LDR_HIGH_LIMIT);
  bool brightEnough = DARK_WHEN_LOW
      ? (lightValue >= LDR_HIGH_LIMIT)
      : (lightValue <= LDR_LOW_LIMIT);
  if (darkEnough) setLight(true);
  else if (brightEnough) setLight(false);
  // Between limits, keep the previous state.

  // RULE 2: fan uses two temperature thresholds.
  if (!isnan(tempC)) {
```

```
  if (tempC >= TEMP_FAN_ON_C) setFan(true);
  else if (tempC <= TEMP_FAN_OFF_C) setFan(false);
 }

 // RULE 3: one movement event = short door + beep.
 if (motionNow && !previousMotion) {
  setDoor(true);
  startBeep();
  doorCloseAt = millis() + DOOR_OPEN_MS;
  doorTimerActive = true;
 }
 previousMotion = motionNow;
}
// A complete local dashboard: one existing SMARTSPACE, no new wires.
const char DASH_HTML[] PROGMEM = R"HTML(
<!doctype html><html lang="en"><head><meta charset="utf-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>SMARTSPACE | Local Lab</title>
<style>
body{font-family:Arial,sans-serif;background:#f3f7fa;color:#103065;
 max-width:680px;margin:20px auto;padding:0 14px}
h1{margin-bottom:4px}small{color:#567}
.card{background:white;padding:16px;border-radius:12px;
 border:1px solid #cadae5;margin:12px 0}
.metric{display:inline-block;margin:8px 12px 8px 0}
button{padding:10px 15px;margin:5px;border:0;border-radius:8px;
 background:#145b94;color:white;font-weight:bold}
button.mode{background:#287a44}
#msg{color:#287a44;font-weight:bold}
</style></head><body>
<h1>SMARTSPACE Three Control Modes</h1>
<small>MindMetrix Labs | LOCAL classroom network only</small>
<div class="card"><h2>Live status</h2><div id="mode">Connecting...</div>
<div id="data">Waiting for readings...</div>
<small>Refreshes about every 3 seconds</small></div>
<div class="card"><h2>Choose a mode</h2>
<form action="/do" method="post">
<button class="mode" name="c" value="L">LOCAL</button>
<button class="mode" name="c" value="R">REMOTE</button>
<button class="mode" name="c" value="A">AUTO</button>
</form></div>
<div class="card"><h2>Remote controls</h2>
<p>These output buttons work only in REMOTE mode.</p>
<p>LOCAL uses the new physical button. AUTO uses sensor rules.</p>
<form action="/do" method="post">
<button name="c" value="1">Light ON</button>
<button name="c" value="0">Light OFF</button><br>
<button name="c" value="D">Door OPEN</button>
<button name="c" value="d">Door SHUT</button><br>
<button name="c" value="F">Fan ON</button>
<button name="c" value="f">Fan OFF</button>
<button name="c" value="B">Short beep</button>
</form></div>
<p id="msg">Teacher-supervised prototype: no real access control.</p>
<script>
async function refresh(){try{
 const r=await fetch('/status',{cache:'no-store'});
 if(!r.ok)throw Error('status'); const s=await r.json();
 document.getElementById('mode').textContent='MODE: '+s.mode;
 document.getElementById('data').textContent=
 'Light '+s.light+' | Door '+s.door+' | Fan '+s.fan+
 ' | LDR '+s.ldr+' | Motion '+s.motion+
 ' | Temp '+(s.temp===null?'--':s.temp+' C')+
 ' | Humidity '+(s.humidity===null?'--':s.humidity+' %');
 }catch(e){document.getElementById('mode').textContent='Check network';}}
refresh();setInterval(refresh,3000);
</script></body></html>
)HTML";

bool authorized() {
 if (web.authenticate(DASH_USER, DASH_PASS)) return true;
 web.requestAuthentication();
 return false;
}
void dashboardPage() {
 if (!authorized()) return;
 web.sendHeader("Cache-Control", "no-store");
```

```
  web.send_P(200, "text/html", DASH_HTML);
}
void dashboardStatus() {
  if (!authorized()) return;
  String j = "{\"mode\":\"" + String(modeName()) + "\"";
  j += ",\"light\":\"" + String(lightOn ? "ON" : "OFF") + "\"";
  j += ",\"door\":\"" + String(doorOpen ? "OPEN" : "SHUT") + "\"";
  j += ",\"fan\":\"" + String(fanOn ? "ON" : "OFF") + "\"";
  j += ",\"ldr\":" + String(lightValue);
  j += ",\"motion\":\"" + String(motionNow ? "YES" : "NO") + "\"";
  j += ",\"temp\":";
  if (isnan(tempC)) j += "null"; else j += String(tempC, 1);
  j += ",\"humidity\":";
  if (isnan(humidPct)) j += "null"; else j += String(humidPct, 1);
  j += "}";
  web.sendHeader("Cache-Control", "no-store");
  web.send(200, "application/json", j);
}
void dashboardAction() {
  if (!authorized()) return;
  String cmd = web.arg("c");
  if (cmd.length() != 1) {
    web.send(400, "text/plain", "One command only"); return;
  }
  char c = cmd.charAt(0);
  if (c == 'A') selectMode(MODE_AUTO);
  else if (c == 'R') selectMode(MODE_REMOTE);
  else if (c == 'L') selectMode(MODE_LOCAL);
  else if (controlMode == MODE_REMOTE) {
    switch (c) {
      case '1': setLight(true);  break;
      case '0': setLight(false); break;
      case 'D': setDoor(true);   break;
      case 'd': setDoor(false);  break;
      case 'F': setFan(true);    break;
      case 'f': setFan(false);   break;
      case 'B': startBeep();     break;
      default: web.send(400, "text/plain", "Unknown command"); return;
    }
  } else {
    web.send(409, "text/plain", "Change to REMOTE mode first");
    return;
  }
  // LOCAL/AUTO deliberately block direct remote output requests.
  web.sendHeader("Location", "/");
  web.send(303, "text/plain", "Return to dashboard");
}
void updateDisplay() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(controlMode == MODE_AUTO ? "AUTO " :
          (controlMode == MODE_LOCAL ? "LOC  " : "REM  "));
  lcd.print(lightOn ? "L1 " : "L0 ");
  lcd.print(fanOn ? "F1" : "F0");
  lcd.setCursor(0, 1);
  if (isnan(tempC) || isnan(humidPct)) {
    lcd.print("Sensor warming");
  } else {
    lcd.print("T:" + String(tempC, 0) + "C H:" +
            String(humidPct, 0) + "%");
  }
}
void readPhone() {
  if (!BT.available()) return;
  char c = BT.read();
  if (c == 'A') { selectMode(MODE_AUTO); return; }
  if (c == 'R') { selectMode(MODE_REMOTE); return; }
  if (c == 'L') { selectMode(MODE_LOCAL); return; }
  if (c == 'S') {
    BT.print("MODE "); BT.println(modeName());
    return;
  }
  if (controlMode != MODE_REMOTE) {
    BT.println("Choose R for REMOTE output commands");
    return;
  }
  switch (c) { // Existing Activity 5 output commands
```

```
    case '1': setLight(true);  break;
    case '0': setLight(false); break;
    case 'D': setDoor(true);   break;
    case 'd': setDoor(false);  break;
    case 'F': setFan(true);    break;
    case 'f': setFan(false);   break;
    case 'B': startBeep();     break;
    default: return;
   }
   BT.println("REMOTE command received");
}
void setup() {
   Serial.begin(115200);
   pinMode(LOCAL_BUTTON_PIN, INPUT_PULLUP);
   pinMode(LED_PIN, OUTPUT); setLight(false);
   pinMode(PIR_PIN, INPUT);
   pinMode(FAN_A, OUTPUT); pinMode(FAN_B, OUTPUT);
   setFan(false);
   pinMode(BUZZER_PIN, OUTPUT); stopBeep();
   analogReadResolution(12);
   dht.begin();
   door.setPeriodHertz(50);
   door.attach(SERVO_PIN);
   setDoor(false);
   Wire.begin(21, 22); // I2C via verified level shifter
   lcd.init(); lcd.backlight();
   lcd.print("SMARTSPACE READY");
   BT.begin("MMX_SMARTSPACE");
   web.on("/", HTTP_GET, dashboardPage);
   web.on("/status", HTTP_GET, dashboardStatus);
   web.on("/do", HTTP_POST, dashboardAction);
   WiFi.mode(WIFI_STA);
   WiFi.begin(WIFI_NAME, WIFI_PASS);
   unsigned long started = millis();
   while (WiFi.status() != WL_CONNECTED &&
          millis() - started < 15000) delay(250);
   if (WiFi.status() == WL_CONNECTED) {
     web.begin();
     Serial.print("Local dashboard: http://");
     Serial.println(WiFi.localIP());
   } else Serial.println("Wi-Fi offline; Bluetooth still works");
}
void loop() {
   handleLocalButton(); // Physical local control, regardless of network
   readPhone();
   if (buzzerOn && (long)(millis() - buzzerStop) >= 0)
     stopBeep();
   if (doorTimerActive &&
       (long)(millis() - doorCloseAt) >= 0) {
     setDoor(false);
     doorTimerActive = false;
   }
   if (millis() - lastSensor >= 2500) {
     lastSensor = millis();
     readSensors();
     if (controlMode == MODE_AUTO) runAutomaticRules();
   }
   if (millis() - lastScreen >= 2500) {
     lastScreen = millis(); updateDisplay();
   }
   if (WiFi.status() == WL_CONNECTED) web.handleClient();
}
```

ANNEXURE G | Activity 9 — teacher-run gesture bridge (Python)

Teacher check: one full sketch; save separately; verify physical wiring and libraries. Use only the approved network and avoid storing personal data.

```
#!/usr/bin/env python3
"""Teacher-operated MediaPipe -> existing ESP32 SMARTSPACE browser bridge.

Uses an OPEN-SOURCE library and an externally provided compatible .task model.
On a teacher-only isolated classroom LAN, accepts webcam frames locally; stores none.
Requires explicit REMOTE mode selected on the ESP32 dashboard. Never selects it.
Only LIGHT commands: Open_Palm->ON, Closed_Fist->OFF.

IMPORTANT: ESP32's existing Basic Auth is UNENCRYPTED over HTTP. Never deploy on
public Wi-Fi or the Internet, and do not reuse real passwords. This is a lab demo.
"""
import argparse
import getpass
import sys
import time
from urllib.request import HTTPBasicAuthHandler, HTTPPasswordMgrWithDefaultRealm, build_opener, Request
from urllib.parse import urlencode
import json


class GestureFilter:
    """Requires one stable pose for hold_seconds and imposes a cooldown."""
    def __init__(self, hold_seconds=1.2, cooldown_seconds=3.0):
        self.hold_seconds=hold_seconds
        self.cooldown_seconds=cooldown_seconds
        self.candidate=None
        self.since=0.0
        self.last_sent=float("-inf")
        self.last_command=None

    def update(self, label, score, now):
        if label not in ('Open_Palm','Closed_Fist') or score < 0.75:
            self.candidate=None
            self.last_command=None  # reset after a hand is removed or confidence falls
            return None
        if label != self.candidate:
            self.candidate=label
            self.since=now
            return None
        if now-self.since < self.hold_seconds or now-self.last_sent < self.cooldown_seconds:
            return None
        command='1' if label=='Open_Palm' else '0'
        # Do not resend continuously while a hand remains in one pose.
        if command == self.last_command:
            return None
        self.last_sent=now
        self.last_command=command
        return command


def main():
    p=argparse.ArgumentParser(description='Local, supervised SMARTSPACE gesture bridge')
    p.add_argument('--model', required=True, help='Local licensed MediaPipe gesture_recognizer.task path')
    p.add_argument('--host', help='Local ESP32 IPv4 address, e.g. 192.168.x.x')
    p.add_argument('--camera',type=int,default=0)
    p.add_argument('--dry-run',action='store_true',help='Recognise only; NEVER send to ESP32')
    args=p.parse_args()
    if not args.dry_run and not args.host:
        p.error('--host required unless --dry-run')
    # Restrict to local/private IPv4; no public network operation.
    if args.host:
        import ipaddress
        try:
            addr=ipaddress.ip_address(args.host)
        except ValueError:
            p.error('--host must be a numeric IPv4 address on isolated school LAN')
        if addr.version!=4 or not addr.is_private or addr.is_loopback or addr.is_link_local:
            p.error('--host must be a private IPv4 address on isolated school LAN')
    try:
        import cv2
        import mediapipe as mp
    except ImportError:
        sys.exit('Teacher: first install compatible open-source packages: pip install mediapipe opencv-python')
    from mediapipe.tasks import python
```

```
  from mediapipe.tasks.python import vision
  opener=None
  if not args.dry_run:
      username=input('ESP32 dashboard username: ').strip()
      password=getpass.getpass('ESP32 dashboard password (not saved): ')
      base='http://'+args.host
      auth=HTTPPasswordMgrWithDefaultRealm()
      auth.add_password(None,base+'/',username,password)
      opener=build_opener(HTTPBasicAuthHandler(auth))
  else:
      base=None
  cap=cv2.VideoCapture(args.camera)
  if not cap.isOpened():
      sys.exit('Camera unavailable. Check teacher permission / camera index.')
  options=vision.GestureRecognizerOptions(
      base_options=python.BaseOptions(model_asset_path=args.model),
      running_mode=vision.RunningMode.VIDEO,
      num_hands=1)
  flt=GestureFilter()
  t0=time.monotonic()
  print('Ready. Show OPEN PALM / CLOSED FIST. Press Q to quit. Images are never saved.')
  with vision.GestureRecognizer.create_from_options(options) as rec:
      try:
          while True:
              ok,frame=cap.read()
              if not ok: print('No camera frame'); break
              frame_rgb=cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
              mp_img=mp.Image(image_format=mp.ImageFormat.SRGB,data=frame_rgb)
              ms=int((time.monotonic()-t0)*1000)
              pred=rec.recognize_for_video(mp_img,ms)
              label='No hand'
              score=0.0
              if pred.gestures and pred.gestures[0]:
                  label=pred.gestures[0][0].category_name
                  score=pred.gestures[0][0].score
              cmd=flt.update(label,score,time.monotonic())
              if cmd:
                  if args.dry_run:
                      print('DRY RUN -> light', 'ON' if cmd=='1' else 'OFF')
                  else:
                      try:
                          req=Request(base+'/status',headers={'Cache-Control':'no-store'})
                          status=json.load(opener.open(req,timeout=2.0))
                          if status.get('mode')!='REMOTE':
                              print('Blocked: set REMOTE mode via teacher dashboard; current mode=',status.get('mode'))
                              flt.last_command=None
                          else:
                              data=urlencode({'c':cmd}).encode('ascii')
                              req=Request(base+'/do',data=data,method='POST')
                              opener.open(req,timeout=3.0).close()
                              print('SENT: light', 'ON' if cmd=='1' else 'OFF')
                      except Exception as exc:
                          print('Command not confirmed; inspect LAN / authentication:',type(exc).__name__)
                          flt.last_command=None
              cv2.putText(frame,f'{label} {score:.2f}',(20,35),cv2.FONT_HERSHEY_SIMPLEX,0.8,(20,255,20),2)
              cv2.putText(frame,'Q = QUIT | LOCAL / AUTO reject gestures',(20,70),cv2.FONT_HERSHEY_SIMPLEX,0.52,(30,220,230),2)
              cv2.imshow('SMARTSPACE teacher-supervised gesture demo',frame)
              if cv2.waitKey(1)&0xFF==ord('q'):break
      finally:
          cap.release()
          cv2.destroyAllWindows()

if __name__=='__main__':main()
```

TEACHER ASSESSMENT

GRADE 8 • SMARTSPACE • Robotics, IoT and Applied AI • 30-hour project

Student name: _____ Class/Section: _____ Team: _____

School: _____ Date: _____ Assessor: _____

Evidence matters more than a perfect final model. Assess actual participation, safe technique, process records, testing, reflection, collaboration and improvements throughout the project.

A. Activity-by-Activity Process Assessment

No.	Activity	Evidence / observable competency	Score 1–4
1	Build the Brain	Correct ESP32/LED setup; follows polarity and unpowered check	____
2	Control Without Wires	Explains Bluetooth command and successfully tests both states	____
3	Bring It Online	Connects locally; keeps passwords and private network safe	____
4	Give SMARTSPACE Senses	Reads/calibrates sensor values and records observations	____
5	Make It Respond	Verifies servo, fan, buzzer and LCD safely	____
6	Make SMARTSPACE Automatic	Tests IF/THEN rules and calibration honestly	____
7	IoT Control Centre	Uses dashboard and verifies reading/control agreement	____
8	Three Control Modes	Explains LOCAL/REMOTE/AUTO priority and safe switching	____
9	AI for SMARTSPACE — 4 h	Collects data, trains/tests AI, evaluates gestures and limitations	____
10	Grand Challenge	Demonstrates full model, troubleshoots and presents evidence	____

Rating scale: 1 = Needs guidance • 2 = Developing • 3 = Meets expectation • 4 = Independently explains, tests and improves. Mark N/O if not observed; record why below.

Activity subtotal: _____ / 40 Activities observed: _____ / 10

B. Practical Skills and 4C Competencies

Criterion	What the teacher looks for	1–4
Critical thinking	Prediction, fair testing and evidence-based debugging	____
Creativity	A workable model enhancement or well-reasoned design change	____
Collaboration	Safe role-sharing and contribution to group tasks	____
Communication	Clear diagram, explanation, demonstration and log	____
Responsible AI and digital citizenship	Protects privacy; recognises mistakes and limits of AI	____
Green skills	Interprets device ON-time without claiming unmeasured energy saved	____

Skills subtotal: _____ / 24

C. Mandatory Safety and Participation Check

☐ Disconnects USB / external power before rewiring and seeks teacher checks
☐ Uses verified low-voltage supplies, shared grounds and guarded moving components
☐ Uses approved classroom Wi-Fi and avoids sharing passwords, images and identities
☐ Demonstrates responsible, supervised AI and understands that AI is not an emergency stop
☐ Participates in data collection, testing, reflection and final presentation
Safety status: ☐ Satisfactory ☐ Needs supervised re-assessment ☐ Not yet demonstrated
If a critical safety step is not demonstrated, require supervised reassessment before marking the practical activity complete, irrespective of scores.

D. Teacher Feedback and Next Steps

What the student did well:
_____
_____
Evidence / notebook page / photo of model (without faces):
_____
_____
Improvement or follow-up practice:
_____
_____
Student reflection — One thing I can explain better now:
_____
_____
Combined score (if all criteria observed): _____ / 64 Overall judgement: ☐ Emerging ☐ Developing ☐ Proficient ☐ Advanced
Teacher signature: _____ Student signature: _____
Parent / mentor comment (optional): _____