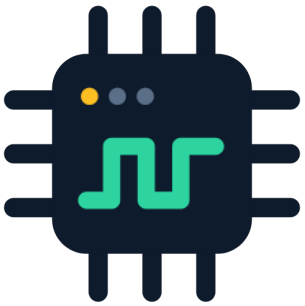


IoT Virtual Lab

Teacher Manual

Run classes, set assignments and quizzes, and review student work

Version 1.6, November 2026
Copyright NIELIT Gorakhpur

Contents

1. Getting started 3
2. Your account 4
3. Classes 5
4. Managing students 6
5. Circuit assignments 6
6. Quizzes 7
7. Reviewing work 8
8. The editor 11
9. While the simulation runs 12
10. Teaching material 12
11. Boards without an Uno 14
12. IoT dashboard 17
13. Real ThingSpeak 17
14. Monitor wireless devices 18
15. Your account 22
16. If something goes wrong 23

1. Getting started

IoT Virtual Lab is a browser-based Arduino and electronics lab. Students need no software or hardware. As a teacher, you create classes, set circuit projects and quizzes, and review the work.

Website (home page)	`https://regn.nielitvte.edu.in/circuit-lab/`
The lab (editor)	`https://regn.nielitvte.edu.in/circuit-lab/lab.html`
Teacher sign-in	`https://regn.nielitvte.edu.in/circuit-lab/teacher.html`

On the first visit, a message explains what the site stores in the browser. Press **OK** to close it. It may cover the bottom of the editor until then, so ask students to close it first.

The usual flow

1. Create a class.
2. Give students its class code. They join by themselves.
3. Build a starting circuit, if you want one, and create an assignment or quiz.
4. Students work and submit.
5. You review, grade and give feedback.

The home page

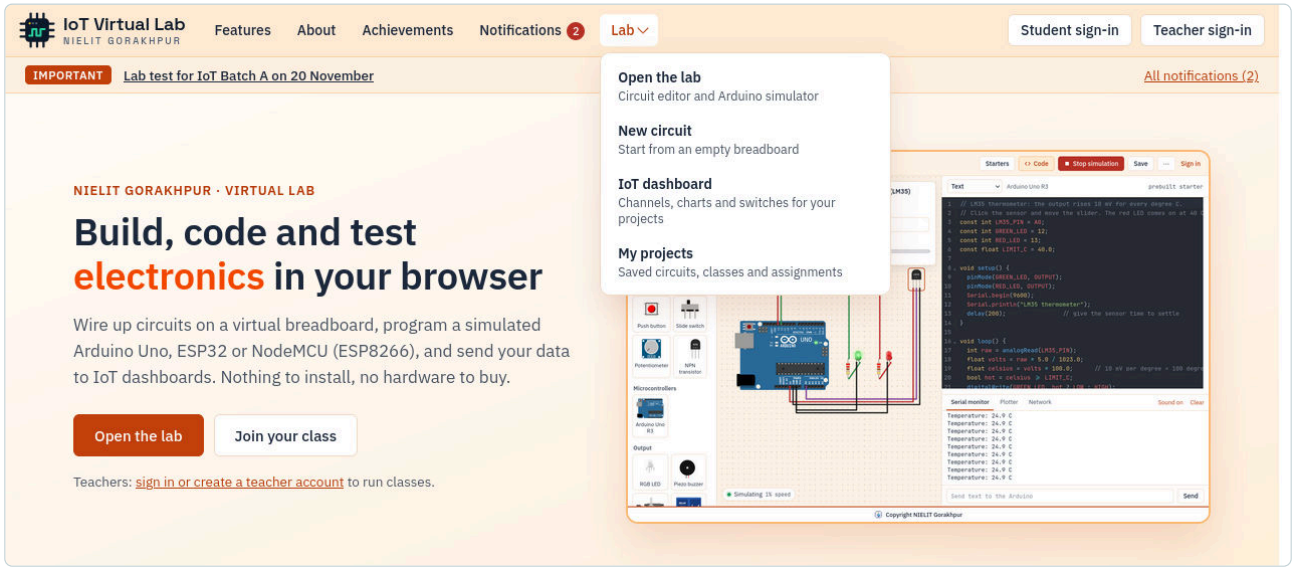


Figure 1. The home page with the Lab menu open. The yellow bar shows the most important notification.

- **Features**, **About** and **Achievements** describe the lab and show how it is being used.
- **Notifications** lists news from the lab, such as test dates. The red number in the menu counts notifications from the last 7 days, and the most important one also appears in the yellow bar under the menu.
- **Lab** opens the editor (**Open the lab**), an empty circuit (**New circuit**), the **IoT dashboard** or **My projects**.
- **Student sign-in** and **Teacher sign-in** are at the top right. Once you are signed in, your name and **My projects** appear there instead.

The notifications, the About text and the achievements are set by your administrator. Ask them to post

test dates or lab news there.

2. Your account

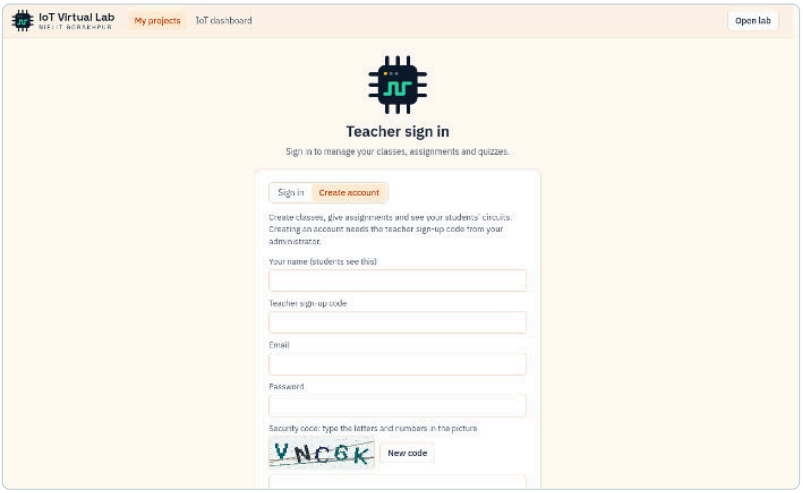


Figure 2. Teacher sign-in, on the Create account tab.

There are two ways to get an account:

- **Your administrator creates it** and gives you a temporary password. Change it under **My account**.
- **You create it yourself** on the **Create account** tab, using the **teacher sign-up code** from your administrator. This tab is hidden if the administrator has turned self sign-up off.

Your password needs at least 10 characters, with letters and at least one number or symbol. It cannot contain your name or email.

The security code

Every sign-in and sign-up form has a **security code**: a small picture of 5 letters and numbers. Type them into the box under the picture. Capitals or small letters both work.

- Press **New code** if you cannot read the picture.
- Each code works once and expires after 10 minutes, so a new picture appears after any failed attempt. Retype the new code, and your password if it was cleared.
- A wrong security code does not count as a wrong password, so it will not lock your account.

> **Forgot your password?** Ask your administrator to reset it. After 5 wrong passwords, an account is blocked for 15 minutes. You are signed out after 60 minutes with no activity.

3. Classes

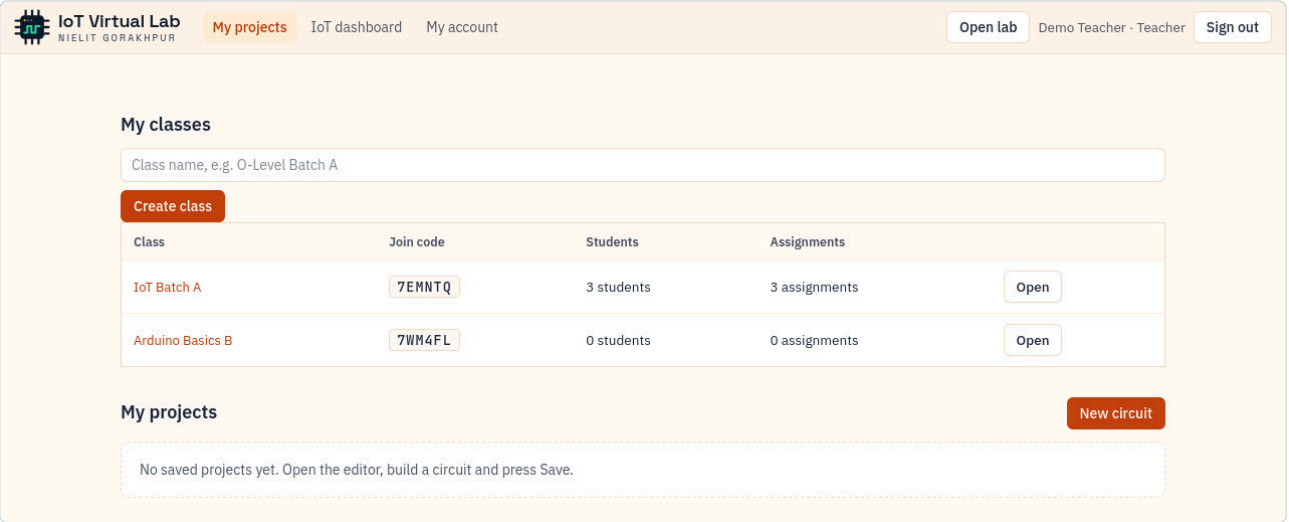


Figure 3. My classes, on your home page.

1. On **My projects**, type a class name and press **Create class**.
2. Press **Open**. The class code is next to the class name.
3. Give students the code. They use **Join a class** on the student sign-in page, or **Join another class** if they already have an account.

Use **Rename** to change the name. **Delete class** removes the class, its assignments and all submissions. Student accounts and their projects stay.

> Teachers cannot add students by hand. Each student joins with the class code and chooses their own nickname and password.

4. Managing students

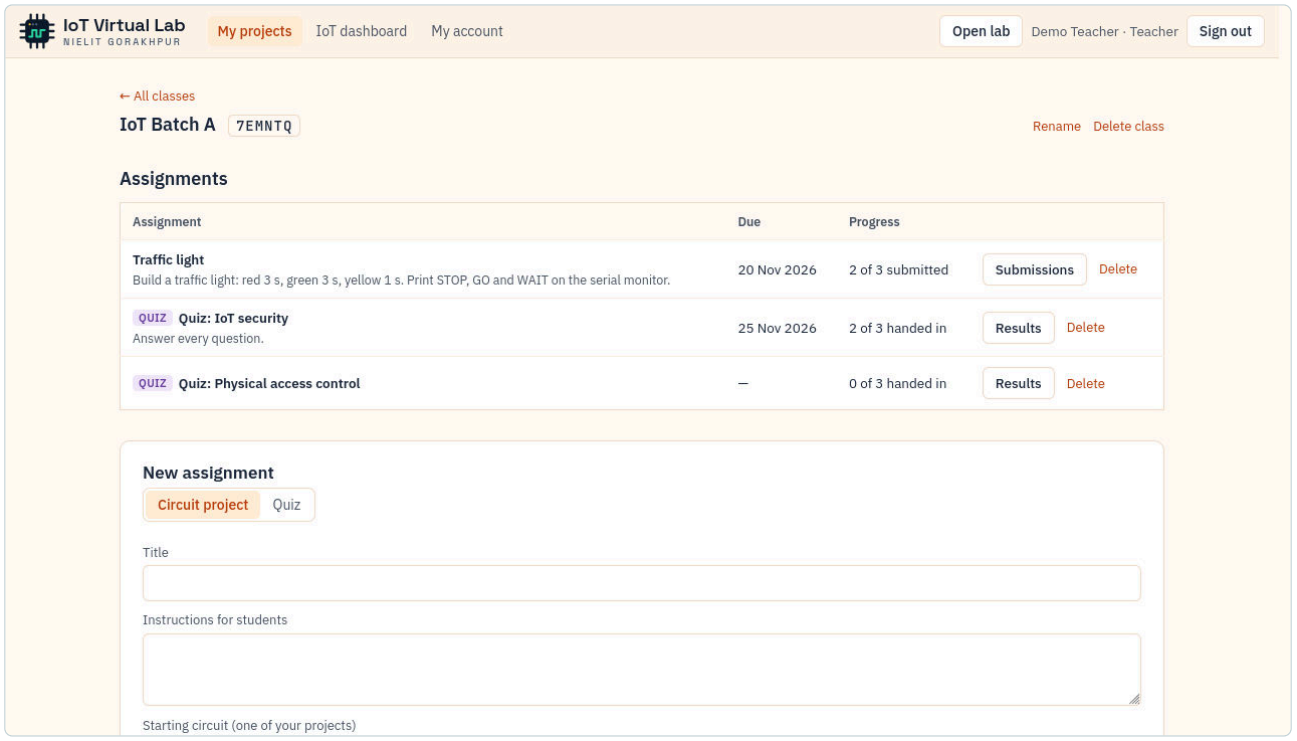


Figure 4. A class: its code, assignments, the new assignment form and students.

- **Lab activity** (above the student list) shows every saved project of every student in the class. See **Reviewing work**.
- **Reset password:** set a new password for a student who has forgotten theirs or is blocked, then tell them. They can change it under **My account**.
- **Remove from class:** takes the student out of this class. Their projects are kept.

Only your administrator can rename, disable or delete a student account.

5. Circuit assignments

1. Build the starting circuit in the editor and **Save** it as one of your projects, or leave this out for an empty circuit.
2. Open the class. Under **New assignment**, keep **Circuit project** selected.
3. Enter the **Title** and **Instructions for students**.
4. Choose the **Starting circuit** and, if you want, a **Due date**.
5. Press **Create assignment**.

Each student gets their own copy when they press **Start**, and your original stays unchanged.

Using a lab template

1. In the editor, press **Starters** and open one, such as **Keypad door lock**.
2. Choose **⋯ → Give as assignment…**.
3. Choose the class. The title is filled in, and many starters also suggest instructions you can edit.

4. Add a due date if you want, then press **Create assignment**.

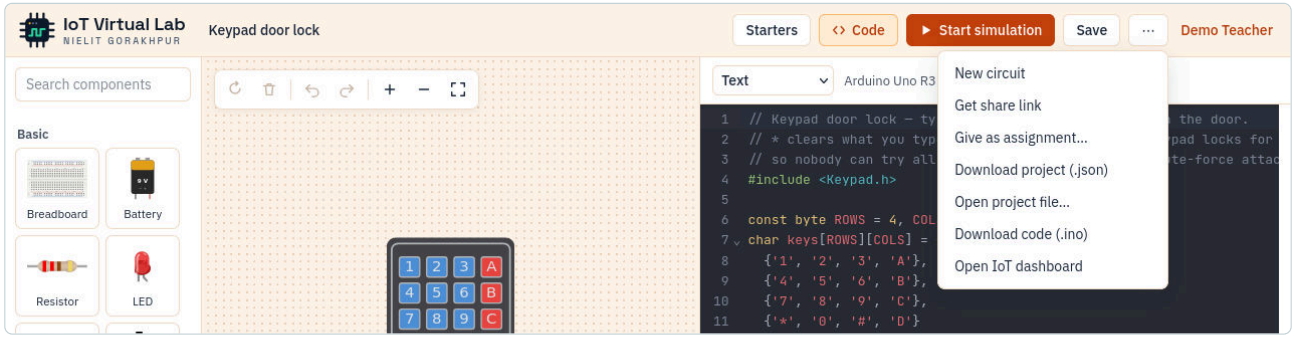


Figure 5. Giving a starter circuit as an assignment, from the ⋯ menu.

6. Quizzes

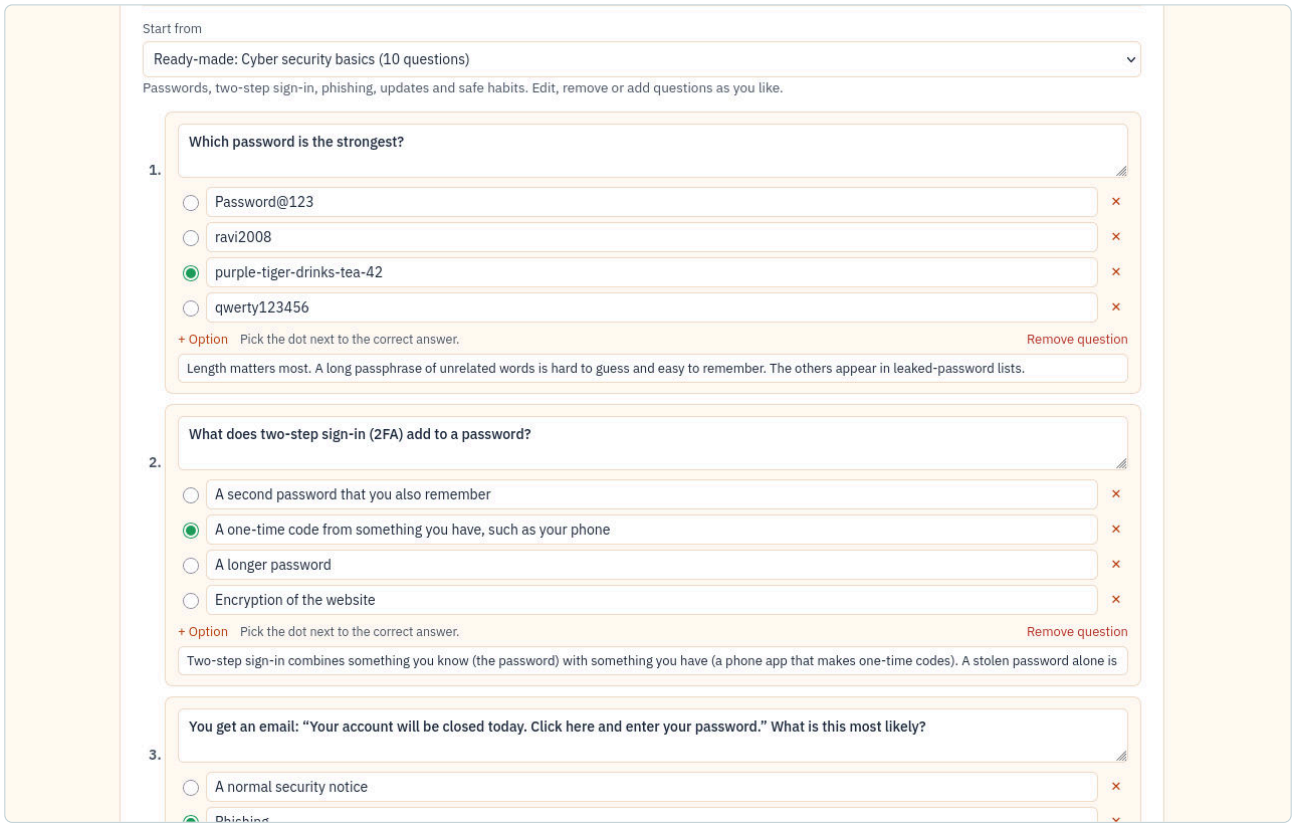


Figure 6. The quiz builder. The green dot marks the correct answer.

1. Under **New assignment**, choose **Quiz**.
2. In **Start from**, choose a ready-made bank or **Write my own questions**. There are three banks: Cyber security basics, IoT security and Physical access control.
3. Edit, remove or add questions. Pick the dot next to the correct answer. An explanation is optional.
4. Choose whether to **Show correct answers and explanations** after hand-in, and whether **Students may try again until the due date**.
5. Enter a title and due date, then press **Create assignment**.

> If retries are allowed, students who can see the answers can use them to score full marks on the next try. For assessed work, consider turning one of these options off.

7. Reviewing work

You can open, run and correct your students' work: assignment submissions and anything else they have saved (their lab activity). The student's project only changes if the student chooses to use your corrected version.

Find the work

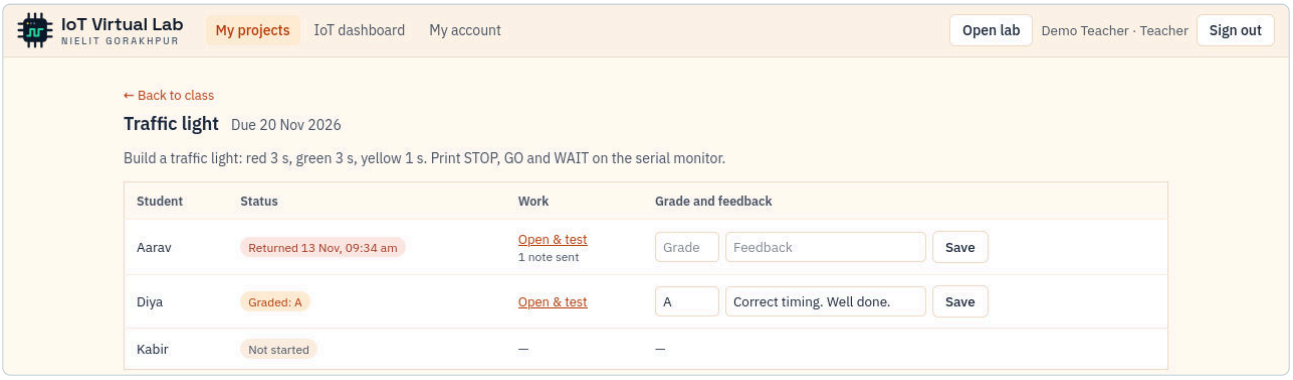


Figure 7. Submissions for a circuit assignment. Aarav was returned for rework; Diya is graded.

- **Assignments:** in the class, press **Submissions** next to the assignment, then **Open & test** next to a student. The **Work** column shows how many notes you have sent.
- **Lab activity:** in the class, press **Lab activity**. It lists every saved project of every student, newest first. Type in the filter box to find a student or project, then press **Open & test**.

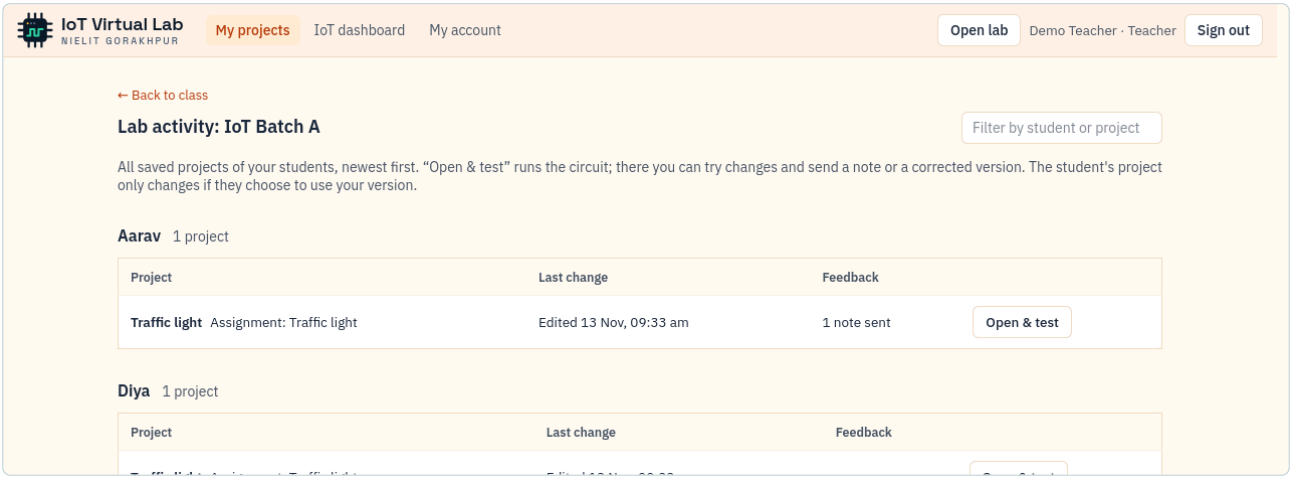


Figure 8. Lab activity: all saved projects of the students in a class.

Run and test it

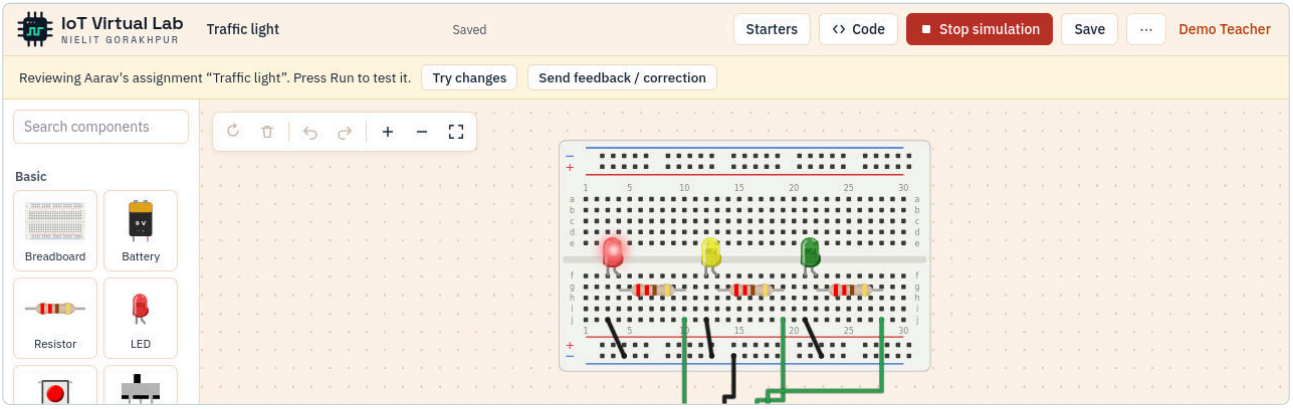


Figure 9. Reviewing a student's assignment. Run it, try changes, or send feedback.

1. Press **Start simulation**. The student's code is compiled and runs exactly as it did for them. Watch the circuit and the serial monitor (**Code**).
2. To find or show a fix, press **Try changes**. You can now edit the circuit and code and run them again. Nothing is saved to the student's project or to your own projects.
3. Press **Send feedback / correction**.

Send feedback or a correction

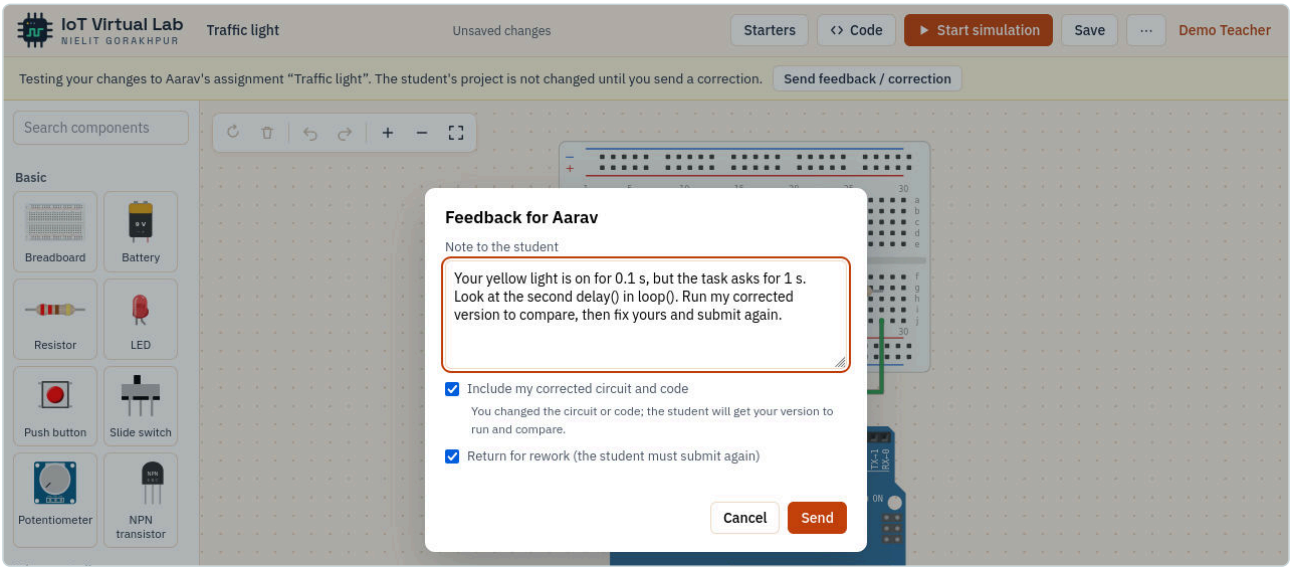


Figure 10. Feedback for a student, with a corrected version and Return for rework.

- Write a **note**: what works, what is wrong and what to try.
- Tick **Include my corrected circuit and code** to send your changed version. The student can run it, compare it with theirs and, if they want, use it.
- For an assignment, tick **Return for rework** if the student must correct it. Their status becomes **Returned for rework** until they submit again.
- Press **Send**. The student sees a **new teacher note** label on their home page and a message in the project. **Earlier notes** in the same window lists what you sent before and whether the student has seen it.

> Tip: a short note that points to the problem (for example, "look at the second delay()") teaches more than a full corrected version. Send the corrected version when the student is stuck.

Grade

- On the Submissions page, type a **Grade** (for example A or 8/10) and **Feedback**, then press **Save**. The student sees both on their home page.
- A student can submit again after you grade or return the work. Check the submitted time if you are waiting for a correction.

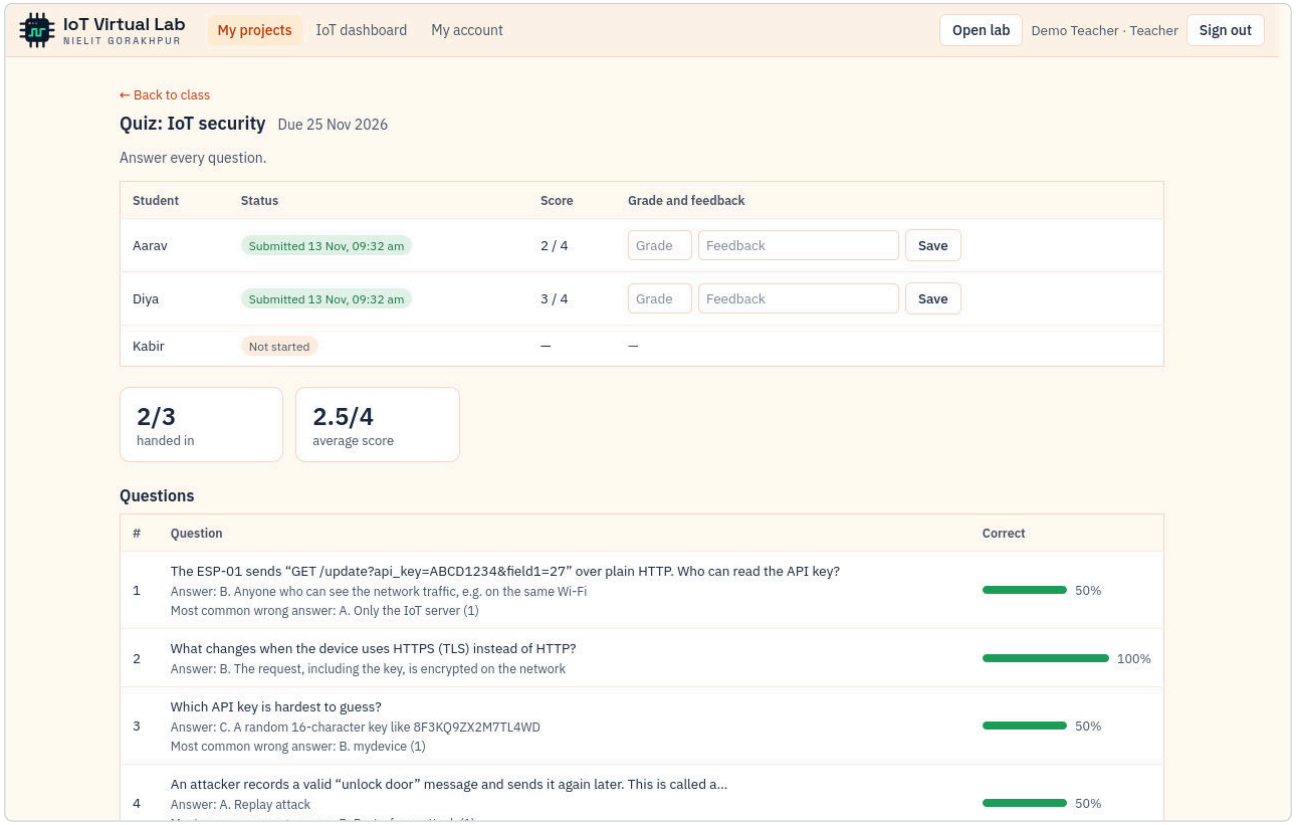


Figure 11. Quiz results: marks are automatic, with statistics for each question.

For a quiz, press **Results**. Marks are automatic. The **Questions** table shows the percentage correct for each question and the most common wrong answer, which helps you decide what to teach again. You can still add a grade and feedback.

Delete next to an assignment removes it and its submissions. Students keep their projects.

8. The editor

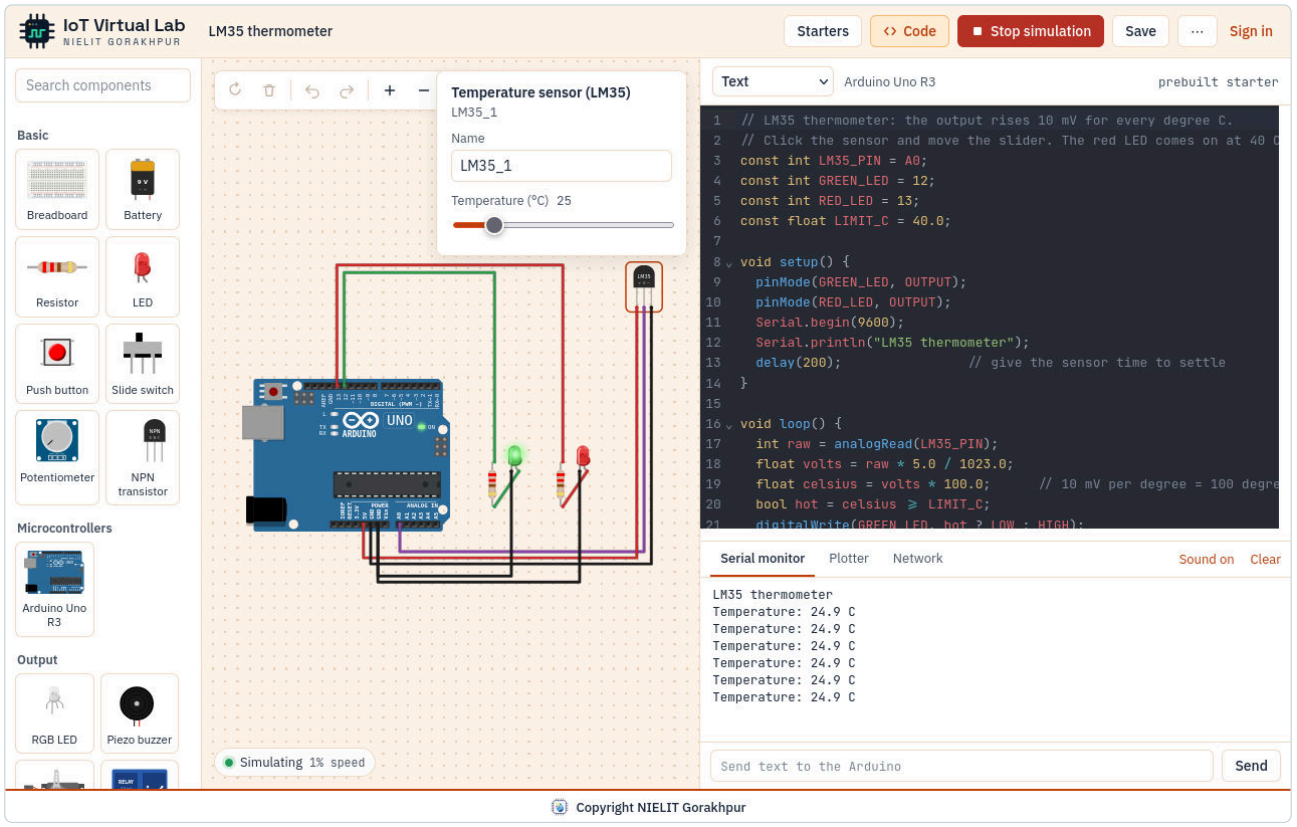


Figure 12. The editor while a simulation runs. Component list (left), canvas (centre), code and serial monitor (right).

Area	What it does
Top bar	Project title, **Starters**, **Code** (shows or hides the code panel), **Start simulation**, **Save** and the ⋯ menu.
Components (left)	Every part you can use, in groups. Type in **Search components** to find one quickly.
Canvas (centre)	Where you build the circuit. The small toolbar has undo, redo, delete, zoom and fit-to-screen.
Code panel (right)	Your program. Choose **Text**, **Blocks** or **Blocks + Text** from the list at the top.
Serial monitor	Shows what your program prints. The tabs also give a **Plotter** and a **Network** view.

1. Drag a component from the left onto the canvas. Parts plug into breadboard holes and Arduino header pins when you drop them close enough.
2. **To draw a wire:** click a pin, then click the pin you want to connect it to. Click on empty canvas along the way to add bends. Press **Esc** to cancel.
3. **To change a part:** click it. A small panel shows its name and settings, such as resistor value or LED colour. Click a wire to change its colour.
4. Write your program in the code panel, or use the one that came with a starter.
5. Press **Start simulation**. Your code is compiled and the circuit comes alive. Press **Stop simulation** to make changes.

6. Press **Save** (or Ctrl+S) often.

Key	Action
R	Rotate the selected part
Delete or **Backspace**	Delete the selected part or wire
Esc	Cancel a wire, clear the selection
Ctrl+Z / **Ctrl+Y**	Undo / redo
Ctrl+S	Save

> **Compiling needs sign-in.** Without an account you can run the starter circuits as long as you do not change their code. After you change the code, you must sign in to compile it. Each person can run one compile at a time and up to 20 a minute.

9. While the simulation runs

- **Interact with parts.** Press buttons and keypad keys directly. For a sensor, click it and move the slider or tick the box in its panel, then watch the circuit respond.
- **Warnings** appear on a part if something is wrong, for example no power or no resistor in series with an LED.
- **Serial monitor.** Shows anything printed with `Serial.print()`. Type in the box below it to send text to the Arduino.
- **Plotter.** Draws the numbers your program prints, line by line, as a graph.
- **Network.** Monitors the wireless parts in your circuit and shows their traffic as someone listening on the same network would see it. You can save it for Wireshark. See **Monitor wireless devices**.
- **Speed.** The label at the bottom left shows how close to real time the simulation runs. It can drop on slower computers.

10. Teaching material

Starter circuits

Each starter is a working circuit with its code, and most run without sign-in if the code is unchanged. Good for demonstrations.

Group	Starter circuits
Basics	Blink, Button and LED, Buzzer tune, Fade (PWM), LED and battery, Measure PWM, Night light, Potentiometer, Serial echo, Servo sweep, Traffic light
IoT & sensors	Weather station, Parking sensor, Motion alarm, OLED meter, NeoPixel rainbow, Motor speed, IoT weather station, IoT remote switch, IR obstacle alarm, IR remote control, Air quality monitor, LM35 thermometer, Bluetooth LED control, ESP32 WiFi thermometer, ESP32 on its own, NodeMCU weather station, ESP-01 on its own
Security lab	Keypad door lock, Keypad PIN check, RFID access control, Door tamper alarm, Fingerprint door

Components

Group	Components
Basic	Breadboard, Battery, Resistor, LED, Push button, Slide switch, Potentiometer, NPN transistor
Microcontrollers	Arduino Uno R3
Output	RGB LED, Piezo buzzer, Micro servo, Relay module, DC motor
Input	Light sensor (LDR), Temperature sensor (NTC), DHT22, Ultrasonic distance (HC-SR04), PIR motion sensor, Gas sensor (MQ-2), Soil moisture sensor, IR obstacle sensor (FC-51), IR receiver, IR remote, Air quality sensor (MQ-135), Temperature sensor (LM35)
Displays	7-segment display, LCD 16x2 (I2C), OLED 128x64 (I2C), NeoPixel ring
IoT	WiFi module ESP-01, ESP32 DevKit, NodeMCU ESP8266, Bluetooth module HC-05
Security	Keypad 4×4, Keypad 4×3, RFID reader RC522, Fingerprint sensor, Door contact (reed switch), Solenoid door lock
Instruments	Multimeter (voltage), Oscilloscope

Special components

Part	What to know
ESP32, NodeMCU, ESP-01	Without an Arduino Uno in the circuit, they run your sketch themselves (see **Boards without an Uno**). With a Uno, the ESP32 and ESP-01 work as WiFi modules: the Uno sends AT commands over `RX2` / `TX2` (ESP32) or `RX` / `TX` (ESP-01).
HC-05 Bluetooth	Click the module and tick **Phone paired**. Use the **Bluetooth terminal** in its panel as the phone. Set **Mode** to AT commands (38400 baud) to practise configuring it.
IR remote and receiver	The remote needs no wires, but there must be an IR receiver on the canvas. Click its buttons while the simulation runs. On real hardware, the button codes will probably differ, so read them from the serial monitor first.
MQ-135 and LM35	Move the slider to set the air quality or temperature. A real MQ-135 must warm up and be calibrated before its ppm reading can be trusted.
Real 3.3 V pins	The simulator does not damage parts, but on real hardware the HC-05 `RXD`, ESP32 `RX2` and ESP-01 `RX` pins need a voltage divider or level shifter when driven by a 5 V Uno.

Libraries

These libraries are ready to use with `#include`: Servo, LiquidCrystal I2C, Adafruit SSD1306 and GFX, DHT sensor library, Adafruit NeoPixel, Keypad, MFRC522, Adafruit Fingerprint, IRremote. SoftwareSerial and Wire come with the Arduino core.

IR remote code for students

```
#include <IRremote.hpp>

void setup() {
  Serial.begin(9600);
  IrReceiver.begin(2);          // receiver DAT pin
}

void loop() {
  if (IrReceiver.decode()) {
    Serial.println(IrReceiver.decodedIRData.command);
    IrReceiver.resume();
  }
}
```

> **Old IR tutorials will not work.** Sketches using `IRrecv irrecv(2)` and `irrecv.decode(&results)` compile, but print a "no longer supported" notice and decode nothing. Change them to the style above. The circuit itself is fine.

11. Boards without an Uno

Besides the Arduino Uno, three WiFi boards can run your sketch themselves: the **ESP32 DevKit**, the **NodeMCU** and the **ESP-01** (the last two use the ESP8266 chip). Put one of them in a circuit without a Uno and press **Start simulation**. The code panel shows which board runs the sketch, for example **NodeMCU (ESP8266 mode)**.

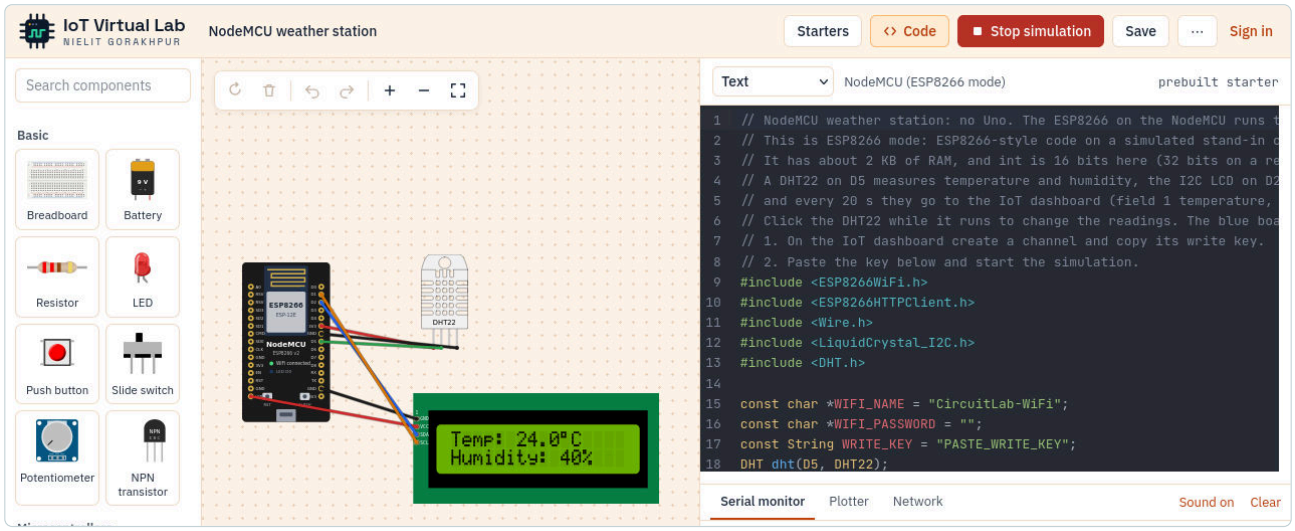


Figure 13. The NodeMCU weather station starter: the NodeMCU runs the sketch itself, with no Uno.

Board	Write code like	Pins you can use
ESP32 DevKit ESP32 mode	`#include <WiFi.h>` and `<HTTPClient.h>`. `analogRead` gives 0 to 4095. PWM with `ledcWrite`.	GPIO 2, 4, 5, 13, 14, 15, 18, 19, 21, 22, 23, 25, 26, 27. Analog: 34, 35, VP (36), VN (39). I2C: 21 and 22. Not 12, 16, 17, 32, 33.
NodeMCU ESP8266 mode	`#include <ESP8266WiFi.h>` and `<ESP8266HTTPClient.h>`. Pin names D0 to D8. `analogRead(A0)` gives 0 to 1023 for 0 to 3.3 V.	D0 to D8. PWM: D3, D4, D7, D8. Interrupt: D3. I2C: D2 (SDA), D1 (SCL). Analog: A0. The blue LEDs on D4 and D0 light when the pin is LOW.
ESP-01 ESP8266 mode	Same as the NodeMCU, with GPIO numbers.	GPIO0 and GPIO2 (both PWM, interrupt on GPIO0), TX and RX for Serial. The blue LED on GPIO2 lights when it is LOW.

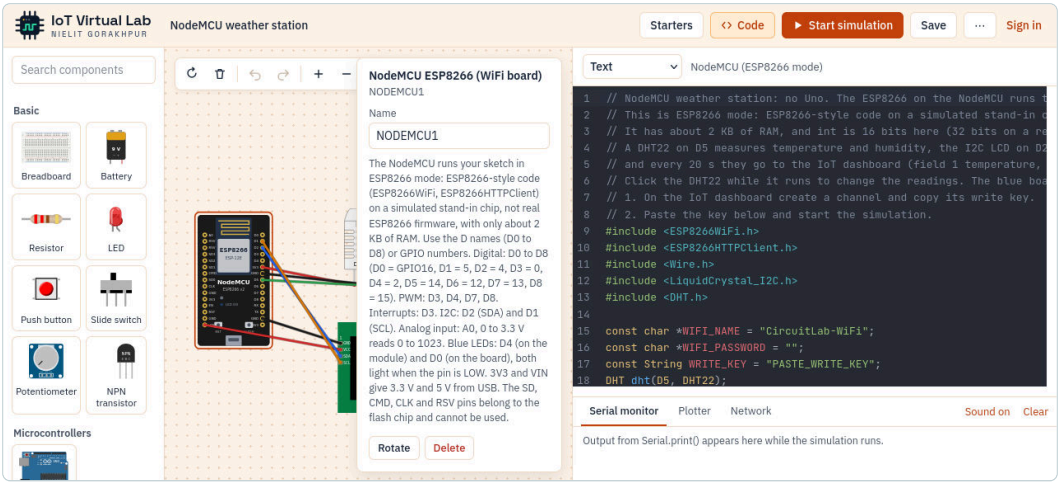


Figure 14. Click a board to see the pins that work, in its part panel.

Which part runs the sketch

- With an Arduino Uno in the circuit, the Uno always runs the sketch. An ESP32 or ESP-01 then works as the Uno's WiFi module, driven by AT commands, as in the **ESP32 WiFi thermometer** and **IoT weather station** starters.
- Without a Uno, the board runs the sketch in this order: ESP32, then NodeMCU, then ESP-01. Any other board shows a warning, except an ESP-01, which can still be the WiFi module of a NodeMCU.

Starters

- **ESP32 on its own**: blink an LED, read a knob and send the value over WiFi.
- **NodeMCU weather station**: a DHT22 and an I2C LCD, with the readings sent to the IoT dashboard.
- **ESP-01 on its own**: a button switches an LED, and the state goes to the IoT dashboard.

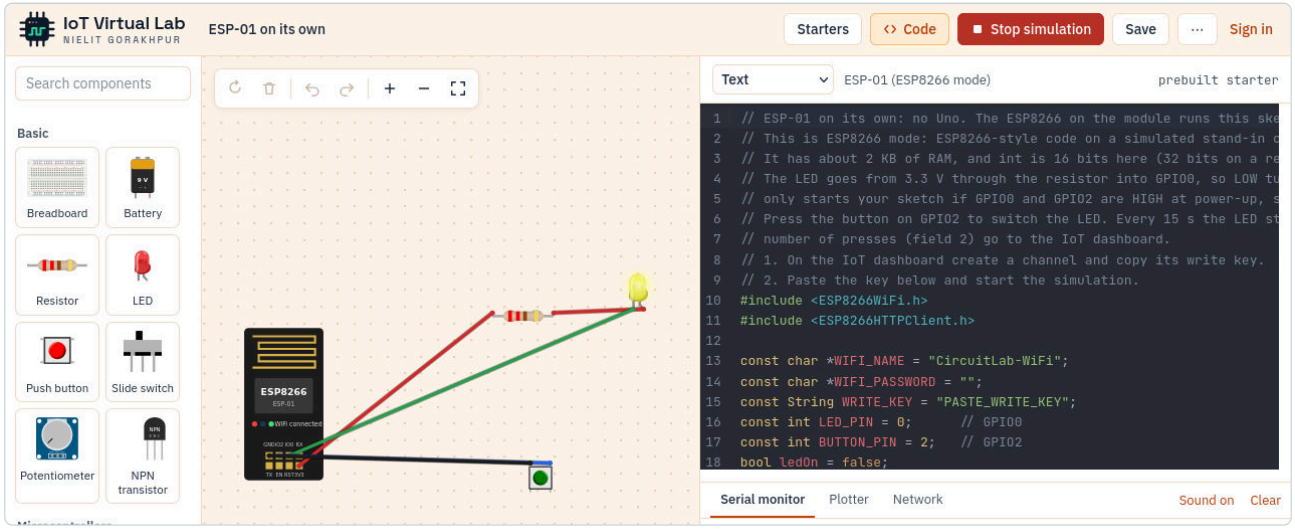


Figure 15. ESP-01 on its own: the button on GPIO2 has switched the LED on GPIO0.

How this differs from a real board

- These modes run ESP-style code on the simulated Arduino chip, with stand-in libraries. They are for learning, and are not real ESP32 or ESP8266 firmware.
- There is only about 2 KB of memory, and `int` is 16 bits here (32 bits on a real ESP). Use `long` for large numbers.
- Available: GPIO, `analogRead`, PWM, `Serial` (with `printf`), `Wire` (I2C), WiFi, the HTTP client, `WiFiClient`, and the common sensor and display libraries.
- Not available: web servers (`ESP8266WebServer`, `WebServer`), Bluetooth on the ESP32, FreeRTOS tasks, OTA updates, SPIFFS/LittleFS and mDNS. If a sketch uses one, the compiler says so.

> Real ESP boards use 3.3 V logic. Nothing is damaged in the simulator, but on real hardware a 5 V signal into an ESP pin can destroy it. A real ESP-01 also only starts your sketch if GPIO0 and GPIO2 are HIGH when it powers up, so do not pull them LOW.

Lesson ideas

- Run the same task (blink, read a sensor, send to the IoT dashboard) on a Uno, an ESP32 and a NodeMCU, and compare the code: pin numbers, `analogRead` range, and the WiFi libraries.
- Ask students why `int x = 100000;` prints a wrong value here but not on a real ESP, and what the safe type is.
- Use the ESP-01 starter to discuss boot pins: why must GPIO0 and GPIO2 be HIGH at power-up on real hardware?

12. IoT dashboard

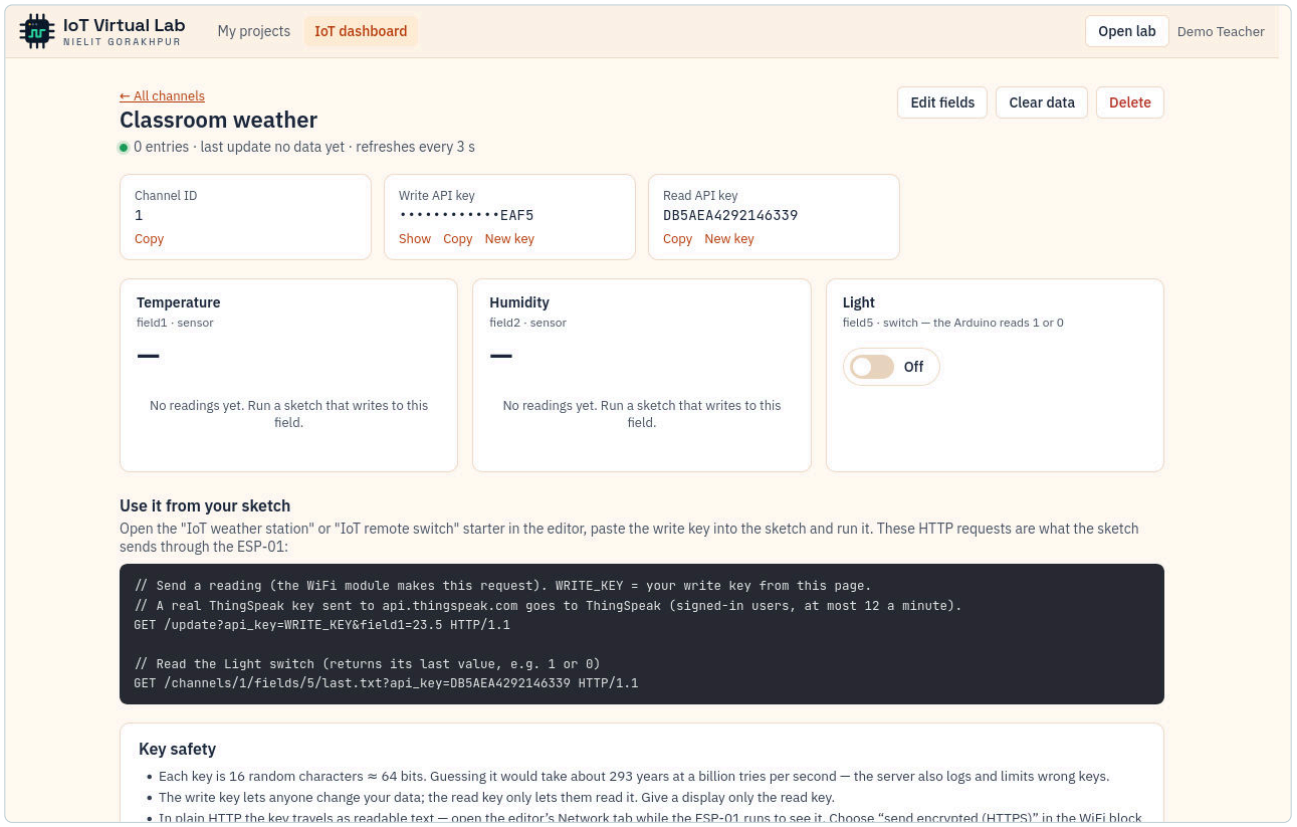


Figure 16. An IoT channel with its keys, sensor fields and a switch.

- Open **IoT dashboard** from the top of the page, or ⋯ → **Open IoT dashboard** in the editor.
- Press **New channel**, name it and set up to 8 fields. A **Sensor** field is written by the Arduino and shown as a chart. A **Switch** or **Slider** is set on the dashboard and read by the Arduino.
- Each channel has a **Write API key** and a **Read API key**. Paste the write key into your sketch. The **IoT weather station**, **IoT remote switch** and **ESP32 WiFi thermometer** starters show how.
- The page refreshes every 3 seconds while your simulation runs.

This suits lessons on data logging and on security. Run the **IoT weather station** starter with the **Network** tab open, and students can see the key travel as readable text. Then compare it with HTTPS. **Monitor wireless devices** has a full lesson. Each account can have up to 20 channels.

> **Protect your keys.** Anyone with a write key can change your data. If a key appears in a screenshot, a shared project or a chat, press **New key**. Plain HTTP sends the key as readable text, as the Network tab shows. In the blocks WiFi block, choose "send encrypted (HTTPS)" to hide it.

13. Real ThingSpeak

Signed-in students and teachers can send simulated readings to a real ThingSpeak channel. It suits a final project where students want a public chart. For most lessons, the IoT Virtual Lab dashboard is simpler, because no outside account is needed.

1. Create a free account and a channel at `thingspeak.mathworks.com`, and copy the channel's **Write API Key**.

2. Sign in to IoT Virtual Lab and open the **ESP32 WiFi thermometer** starter.
3. Paste the key into the line `const String WRITE_KEY = "...";`.
4. Press **Start simulation**. The serial monitor shows the reply from ThingSpeak: an entry number if the reading was stored, or `0` if it was rejected.

- **You must be signed in,** and the sketch must connect to `api.thingspeak.com`.
- **Keys decide where data goes.** A real ThingSpeak key goes to ThingSpeak. A key from the IoT Virtual Lab dashboard stays in the lab, even though the sketch names `api.thingspeak.com`.
- **Do not send too fast.** IoT Virtual Lab allows 12 requests a minute for each person. ThingSpeak's free plan may reply `0` to updates sent more often than about every 15 seconds, so keep a delay of 15 seconds or more.
- **Only ThingSpeak works.** The simulator cannot reach other websites or APIs, and ThingSpeak talkback and other features are not available.

> **Your key passes through the IoT Virtual Lab server** on its way to ThingSpeak. The relay does not store it, but a key in your sketch is saved with your project and is included if you share it. Remove the key before sharing, and create a new one in ThingSpeak if it has appeared in a screenshot or chat.

Each student needs their own ThingSpeak account or channel key. If your administrator has turned this off, ThingSpeak keys are rejected and the sketch shows `0`.

14. Monitor wireless devices

The **Network** tab shows each wireless part in your circuit while it runs: whether it is online, how much data it has used, and the traffic it sends. You can also save that traffic and study it in Wireshark.

1. Add an ESP-01, ESP32, NodeMCU or HC-05 and wire it to the Arduino, or open the **IoT weather station** starter.
2. Press **Start simulation**.
3. Open the **Network** tab under the code panel. If you cannot see it, press **Code**.

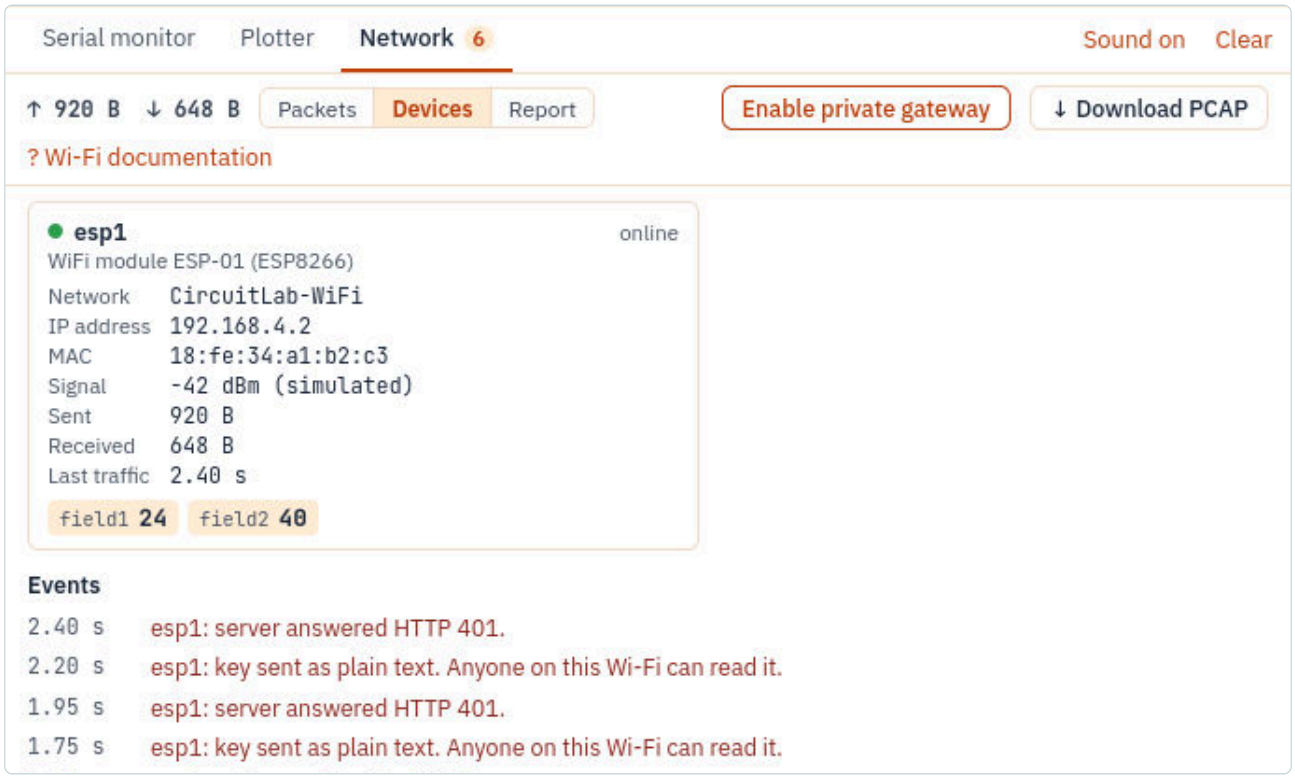


Figure 17. The Devices view: one card for each wireless part, with events below.

Control	What it does
↑ and ↓ counters	Bytes sent and received by your Wi-Fi modules in this run.
Packets	Lists the traffic. Click a line to see its contents. Plain HTTP is readable and any key is highlighted. You can also replay or change a request, as an attacker would.
Devices	One card for each ESP-01, ESP32, NodeMCU or HC-05, with its status, data used and latest `fieldN` values. Events below list joins, keys sent as plain text and server errors. The signal strength is simulated.
Report	A summary of the run: devices, protocols, conversations and findings. **Download report** saves it as a text file. Key values are hidden.
Enable private gateway	Keeps your devices inside the lab. See below.
Download PCAP	Saves the traffic as a `.pcap` file for Wireshark.
Wi-Fi documentation	A short help page inside the lab.

Study the traffic in Wireshark

1. Let the simulation run until the Packets view shows some traffic.
2. Press **Download PCAP**. You can do this while it runs or after you stop it.
3. Open the file in Wireshark, a free program for your own computer (`wireshark.org`).
4. Type a display filter from the table below at the top of the Wireshark window.

Display filter	What it shows
`http.request`	Requests sent as plain text.
`http.request.uri contains "api_key"`	Keys that anyone listening could read.
`tls.handshake.extensions_server_name`	Servers contacted over HTTPS. The name is visible even though the data is encrypted.
`ip.addr == 192.168.4.66`	Traffic from the attacker buttons in the Packets view.
`tcp.stream eq 0`	One conversation. The Report view lists the number of each one.

> **A PCAP file can contain your keys.** Plain HTTP is saved exactly as it was sent, including any API key. Do not share a file that holds a real ThingSpeak key. If you already have, create a new key in ThingSpeak. Use HTTPS and the key stays out of the file.

- **Save before you restart.** Pressing **Start simulation** clears the capture and the report. The file is made in your browser and is not uploaded.
- **This traffic is generated by the simulator.** No real radio is involved. Wireshark shows Ethernet frames, not 802.11 Wi-Fi frames. Addresses starting `203.0.113.` stand in for the real servers.
- **HTTPS cannot be decrypted.** The encrypted data in the file is random bytes of the correct length, so there is nothing to recover, and the handshake has no certificate.
- **Bluetooth is not in the file.** An HC-05 is counted in the Devices view only.

The analysis report

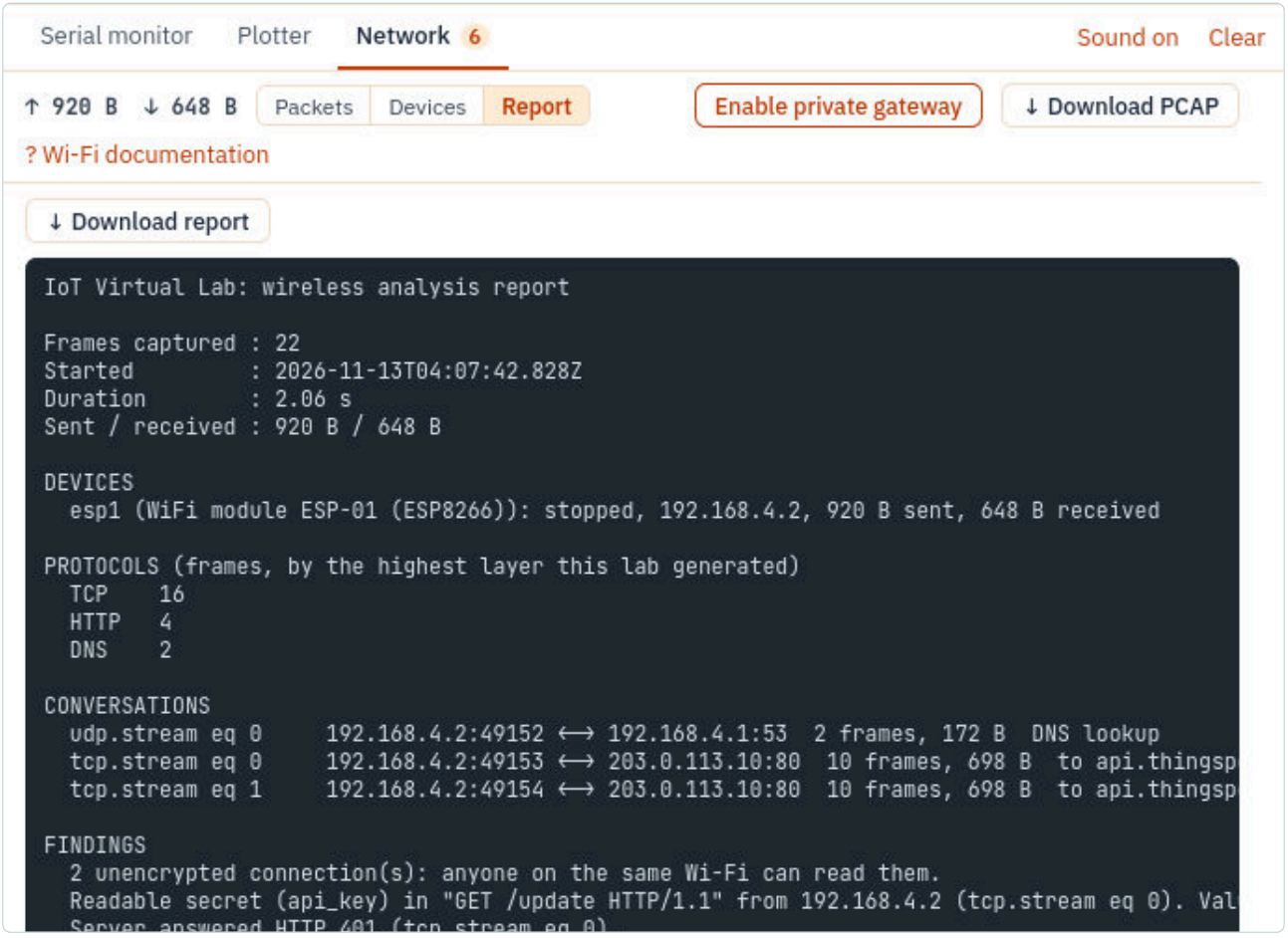


Figure 18. The Report view, ready to download as a text file.

Private gateway

While **Enable private gateway** is on, your devices can reach only the IoT Virtual Lab server. Requests to `api.thingspeak.com` stay inside the lab, so a real ThingSpeak key does not work, but a key from your IoT Virtual Lab dashboard does. Press it again to turn it off. It also turns off when you reload the page. It does not connect the simulator to your own network or to real devices.

Lesson: why HTTPS matters

Students need to be signed in, because changing the code means compiling it. Use a IoT Virtual Lab dashboard key, not a real ThingSpeak key.

1. Students open the **IoT weather station** starter, paste their dashboard write key and press **Start simulation**.
2. In the **Packets** view they find the highlighted key, then press **Replay (as attacker)** and see the same reading stored again on their dashboard.
3. They press **Download PCAP** and find the key in Wireshark with `http.request.uri contains "api_key"`.
4. They stop the simulation and choose "send encrypted (HTTPS)" in the WiFi block, then run it again.
5. They download a second PCAP, repeat the filter and compare. The key has gone, but the server name still shows under `tls.handshake.extensions_server_name`.

> **Download before each new run. Start simulation** clears the capture, so a student who forgets loses the first file and has nothing to compare.

Running it with a class

- **Wireshark is not part of the lab.** It must be installed on each computer. If it cannot be, use the **Packets** and **Report** views, which need nothing installed.
- **You cannot see a student's capture from your account.** PCAP files and reports stay in the student's browser until downloaded, and are not saved with projects or submissions. To collect them, use your own process, such as email or a shared folder.
- **Prefer the report for handing in work.** It hides key values. A PCAP from a plain HTTP run holds the student's key, so ask for dashboard keys only, and have students press **New key** afterwards.
- **Private gateway is each student's own switch.** You cannot turn it on for a class, and students can turn it off. To stop ThingSpeak for the whole site, ask your administrator to use the setting in the Admin Desk.

15. Your account

Under **My account** you can change your password, see recent sign-ins and turn on **two-step sign-in**. It is strongly recommended, because your account can see all your students' work.

1. Install an authenticator app on your phone, such as Google Authenticator.
2. Press **Set up two-step sign-in** and scan the QR code.
3. Type the 6-digit code to confirm.
4. **Save the recovery codes.** They are shown only once, and each works one time if you lose your phone.

After that, you are asked for a 6-digit code after your password. If you lose both your phone and your recovery codes, ask your administrator to reset two-step sign-in.

16. If something goes wrong

Problem	What to do
A student forgot their password	Open the class and use **Reset password**.
A student is blocked	Wait 15 minutes, or use **Reset password**. If a whole lab is blocked, ask your administrator. A shared address is blocked after 50 failures.
Students cannot join	Check the class code. If they see "Joining classes is turned off", ask your administrator.
A student joined with the wrong nickname	Your administrator can rename them under **Students**.
"Create a class on the My projects page first"	**Give as assignment** needs at least one class.
A class must move to another teacher	Your administrator can change the teacher under **Classes**.
Student sketch prints "no longer supported" (IR)	It uses an old IRremote style. See the code in **Teaching material**.
An ESP32 or NodeMCU sketch does not run on the board	Remove the Arduino Uno from the circuit: with a Uno, the Uno runs the sketch. If the error names a missing file such as `ESP8266WebServer.h`, that feature is not available in the simulator (see **Boards without an Uno**).
ThingSpeak reply is always 0	Check the key and the delay between readings. Check too that the student is signed in, and ask your administrator whether the relay is on.
Download PCAP is grey	Nothing has been sent yet. The Wi-Fi module must join the network and open a connection first.
A wireless part is missing from the Devices view	Check its signal wires to the Arduino, then start the simulation again.
ThingSpeak stopped working for one student	Check that the private gateway is off in their **Network** tab.
Page looks wrong after an update	Press Ctrl+F5 to reload it.