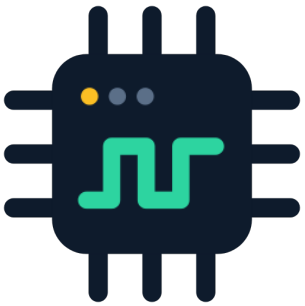


IoT Virtual Lab

Student Manual

Build Arduino circuits in your browser, do assignments and take quizzes

Version 1.6, November 2026

Copyright NIELIT Gorakhpur

Contents

1. Getting started 3
2. Join your class 4
3. Sign in again 4
4. Your home page 5
5. The editor 6
6. Build and run a circuit 6
7. While the simulation runs 7
8. Do an assignment 8
9. Teacher's notes and corrections 9
10. Take a quiz 10
11. Special components 11
12. Boards without an Uno 11
13. All components and starters 13
14. IoT dashboard 14
15. Send data to real ThingSpeak 15
16. Monitor wireless devices 15
17. Sharing and saving 18
18. Your account 19
19. If something goes wrong 19

1. Getting started

IoT Virtual Lab is a virtual electronics lab. You build circuits on screen, program an Arduino Uno and watch it run, with nothing to install and no hardware to buy.

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

You need: a computer with an up-to-date web browser and an internet connection, plus the **class code** from your teacher. A large screen is much easier to use than a phone.

You can try the editor without an account, but you need one to save to your account, compile changed code, and do assignments.

On your first visit a message explains what the site stores in your browser. Press **OK** to close it.

The home page

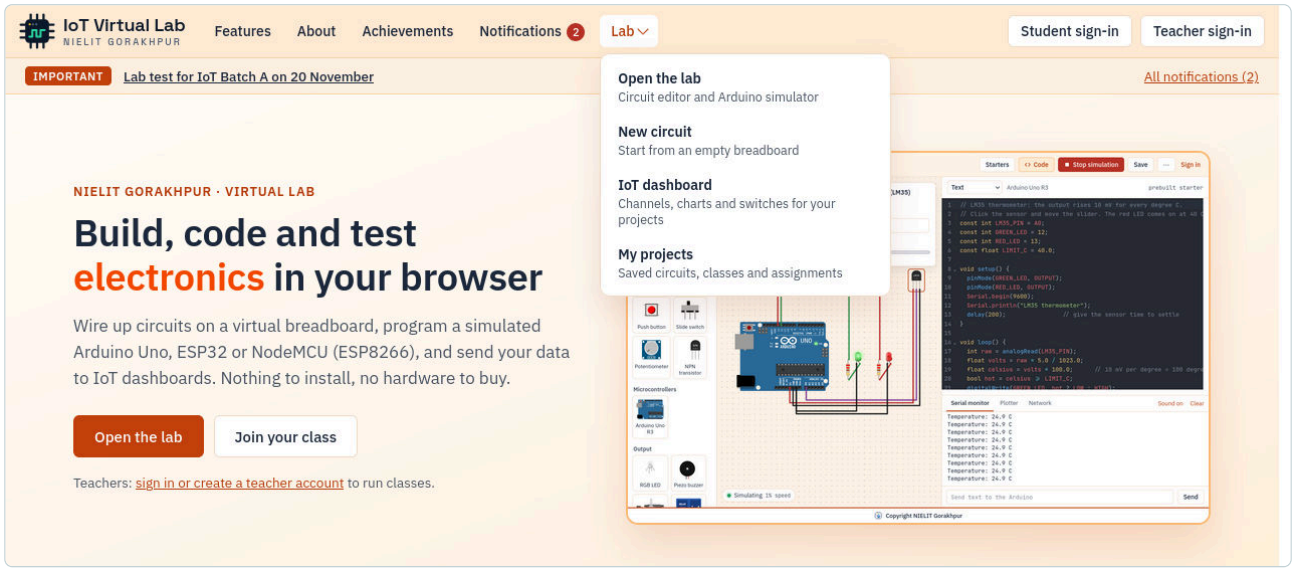


Figure 1. The home page with the Lab menu open. The yellow bar shows the most important notification.

- **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.

2. Join your class

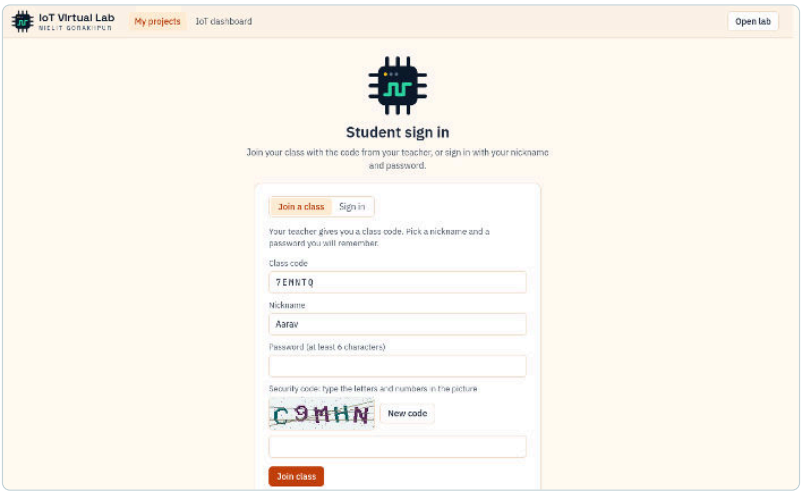


Figure 2. The student sign-in page.

1. Open the student sign-in page and stay on the **Join a class** tab.
2. Type the **Class code** your teacher gave you.
3. Choose a **Nickname**. Your teacher sees this name, and nobody else in the class can already be using it.
4. Choose a **Password** of at least 6 characters. Do not use your nickname or a common password such as `123456`.
5. Type the **security code** from the picture and press **Join class**.

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.

> Write down your class code, nickname and password. You need all three to sign in again.

3. Sign in again

1. Open the student sign-in page and choose the **Sign in** tab.
2. Type your class code, nickname and password.
3. Type the security code and press **Sign in**.

- **Forgot your password?** Ask your teacher to reset it, then change it under **My account**.
- **Blocked after wrong passwords?** After 5 wrong passwords an account is blocked for 15 minutes. Wait, or ask your teacher to reset your password.
- **Signed out by itself?** You are signed out after 60 minutes with no activity. Sign in again. Press

Save often so that you do not lose work.

- **On a shared computer,** always press **Sign out** when you finish.

4. Your home page

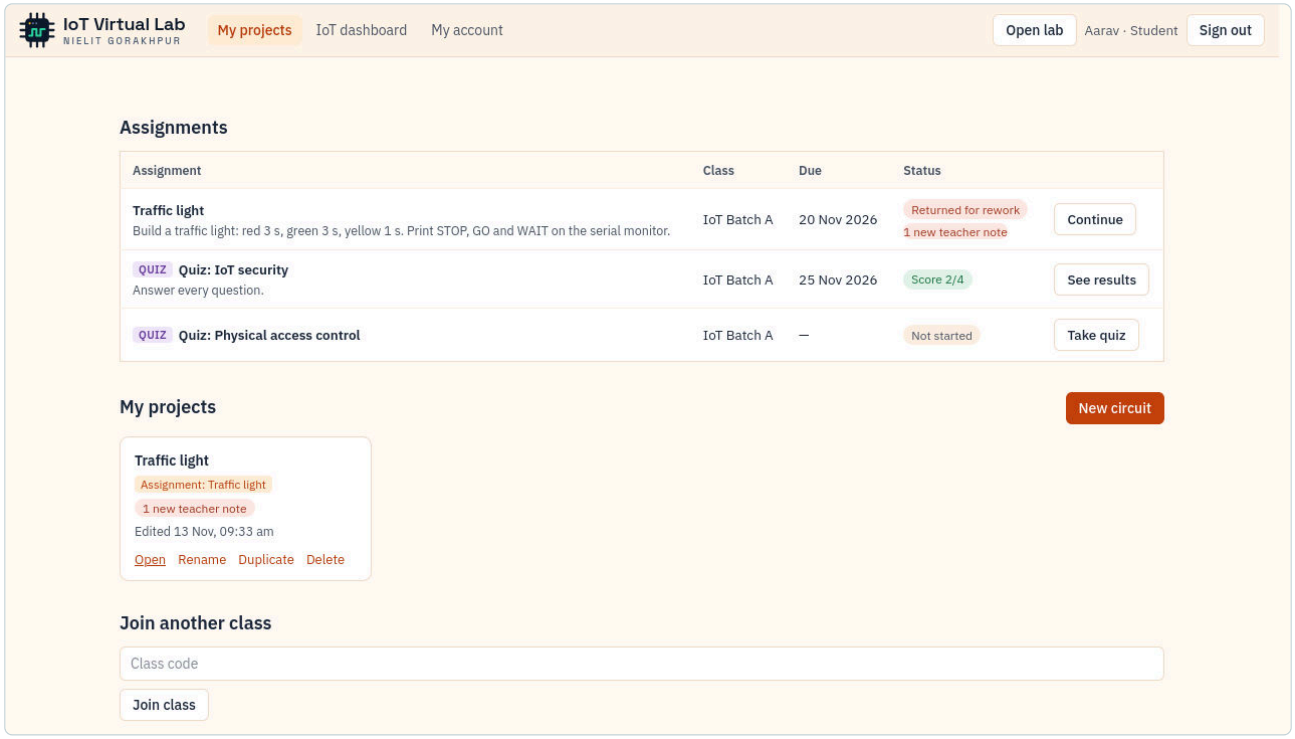


Figure 3. The home page: your assignments, projects and other classes.

- **Assignments** lists work from your teachers, with due dates, your status and any feedback. A red **new teacher note** label means your teacher has checked your work; press it to open the project and read the note.
- **My projects** holds your saved circuits. Use **Open**, **Rename**, **Duplicate** or **Delete**. Deleting cannot be undone.
- **New circuit** starts an empty project.
- **Join another class** adds a class with a new code, using the same account.

5. The editor

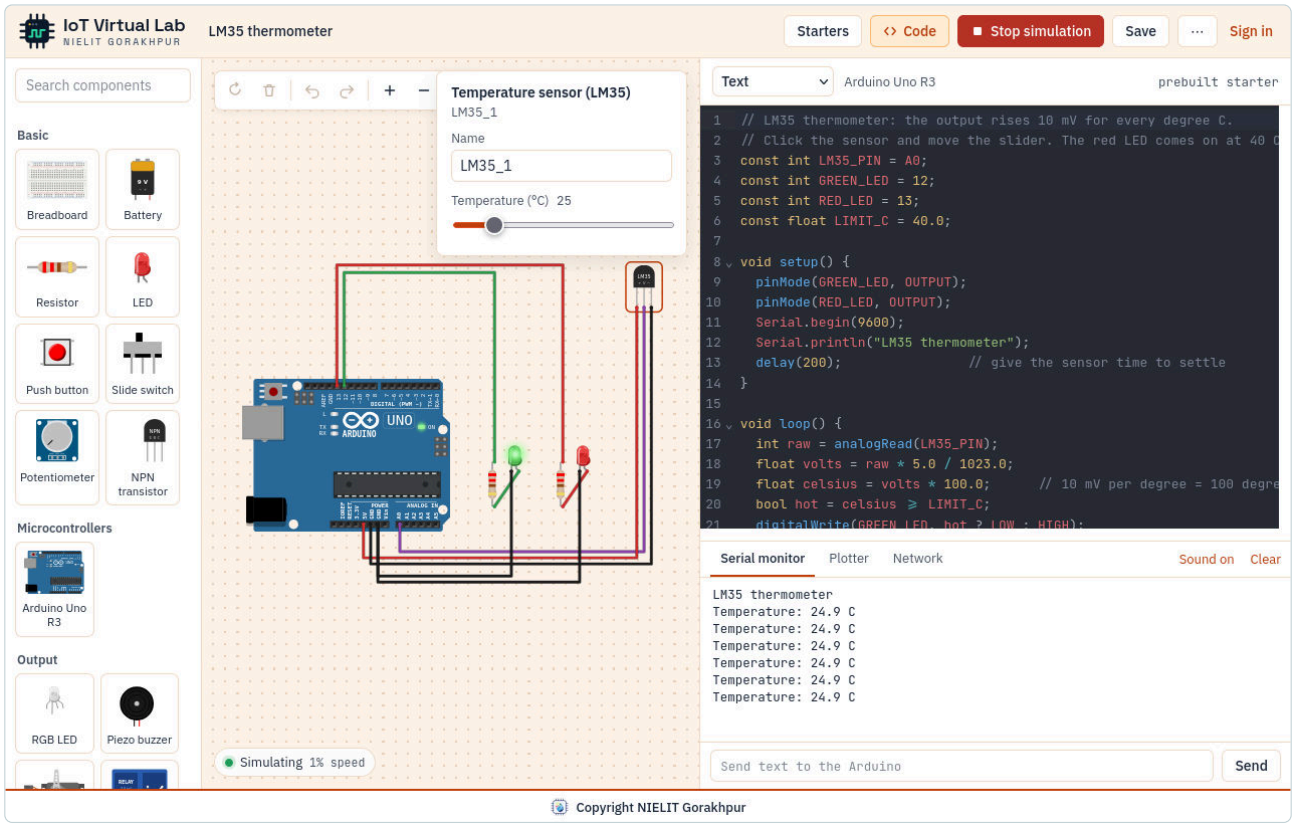


Figure 4. The editor while a simulation runs. Component list (left), canvas (centre), code and serial monitor (right).

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.

6. Build and run a circuit

The quickest way to learn is to press **Starters** and open a ready-made circuit, such as **Blink** or **Traffic light**. Run it, then change it. To build your own:

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.

If your code has a mistake, the compiler errors appear under the code panel. Read the first error, fix that line and try again.

7. 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.

8. Do an assignment

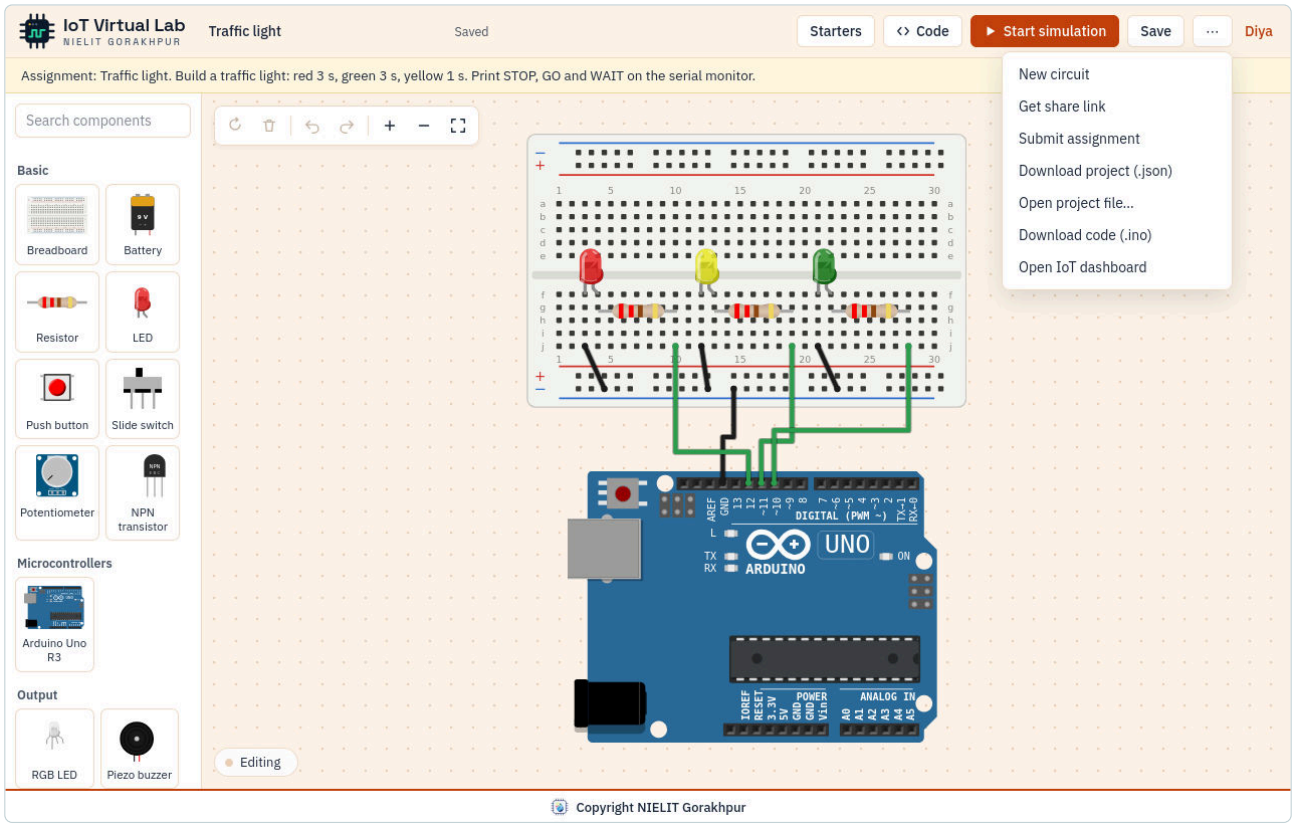


Figure 5. Inside an assignment. The instructions are at the top, and Submit assignment is in the ⋯ menu.

1. On your home page, press **Start** next to the assignment. You get your own copy, which may already contain a starting circuit from your teacher.
2. Build the circuit and write the code. The instructions stay at the top of the screen.
3. Press **Save**.
4. Choose **⋯** → **Submit assignment**. Your teacher sees the version you have just submitted.
5. To improve it, press **Continue**, keep working and submit again.

Status	Meaning
Not started	You have not opened it yet.
In progress	You have started, but not submitted.
Submitted	Your teacher can see your work.
Graded	Your teacher has given a grade. Feedback appears underneath.
Returned for rework	Your teacher wants you to fix something. Read the note, correct your work and submit again.

9. Teacher's notes and corrections

Your teacher can open and run your saved projects, both assignments and your own lab work, and send you a note. A note can include a **corrected version** of your circuit and code.

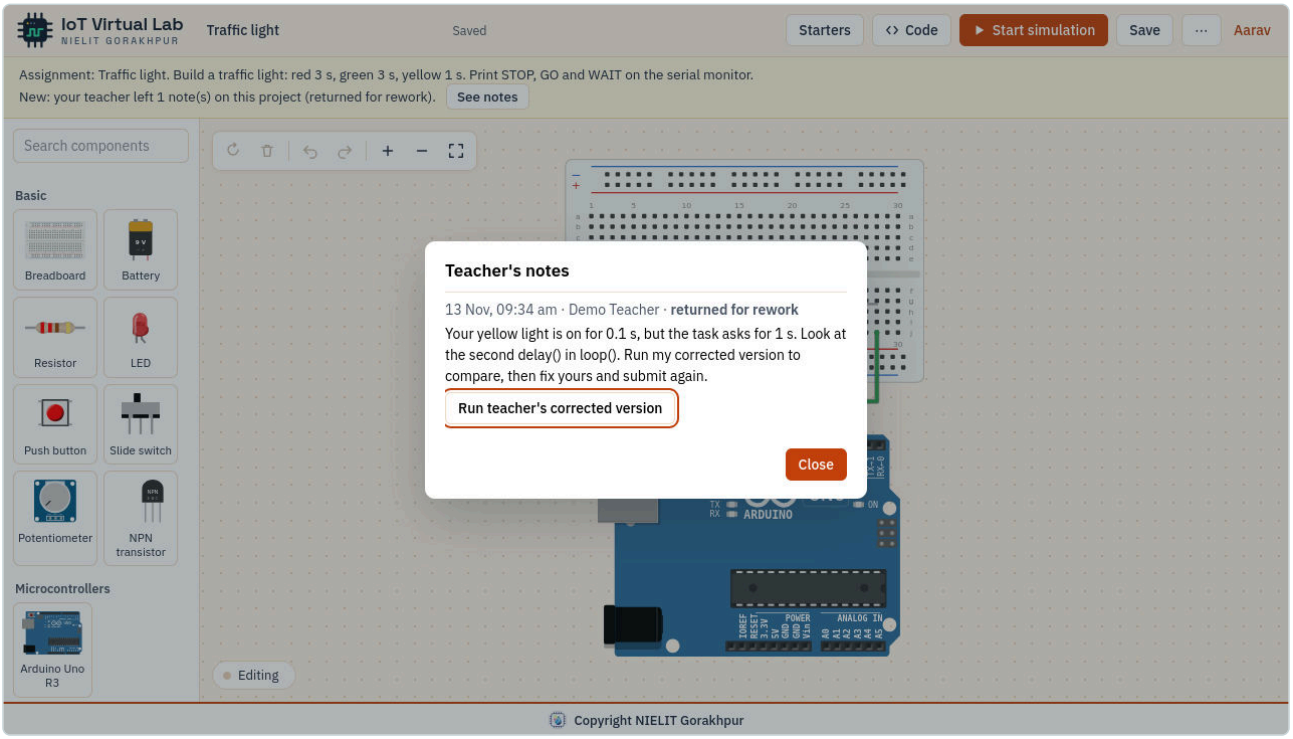


Figure 6. Teacher's notes, opened with See notes. The newest note is at the top.

1. Open the project. A message at the top says your teacher left a note. Press **See notes**.
2. Read the note. If it has a corrected version, press **Run teacher's corrected version**.
3. Press **Start simulation** to run your teacher's version and compare it with yours.
4. Then either press **Back to my version** and fix it yourself, or press **Use this version in my project**.
5. For an assignment, choose ⋯ → **Submit assignment** again when you are done.

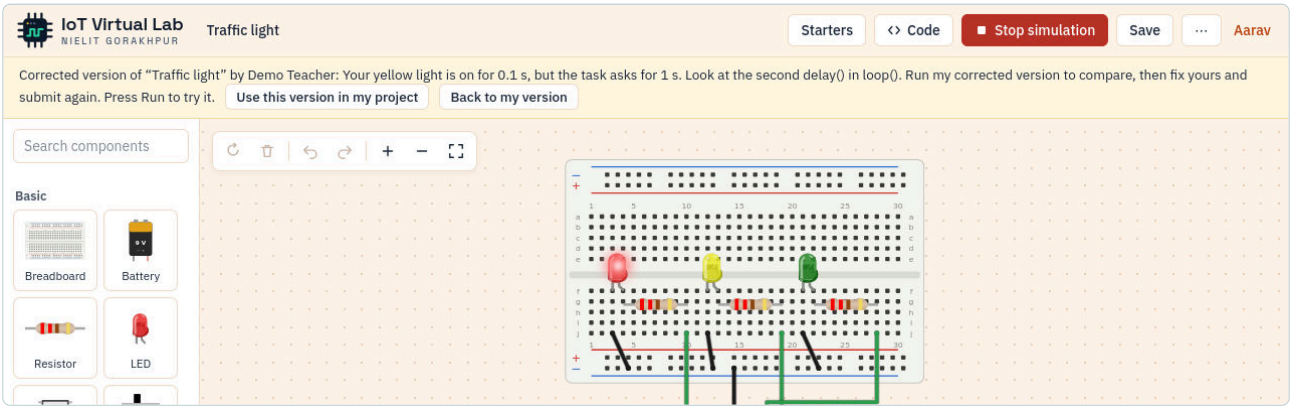


Figure 7. A corrected version from the teacher, with the two choices at the top.

> **Use this version in my project** replaces your project with your teacher's version. Your own version is not lost: it is kept as a separate project called **My version before correction**. Trying it yourself first is usually the better way to learn.

10. Take a quiz

1. On your home page, press **Take quiz**.
2. Answer the questions, and check your answers.
3. Press **Hand in quiz**. It is marked straight away.
4. Use **See results** to view your score at any time.

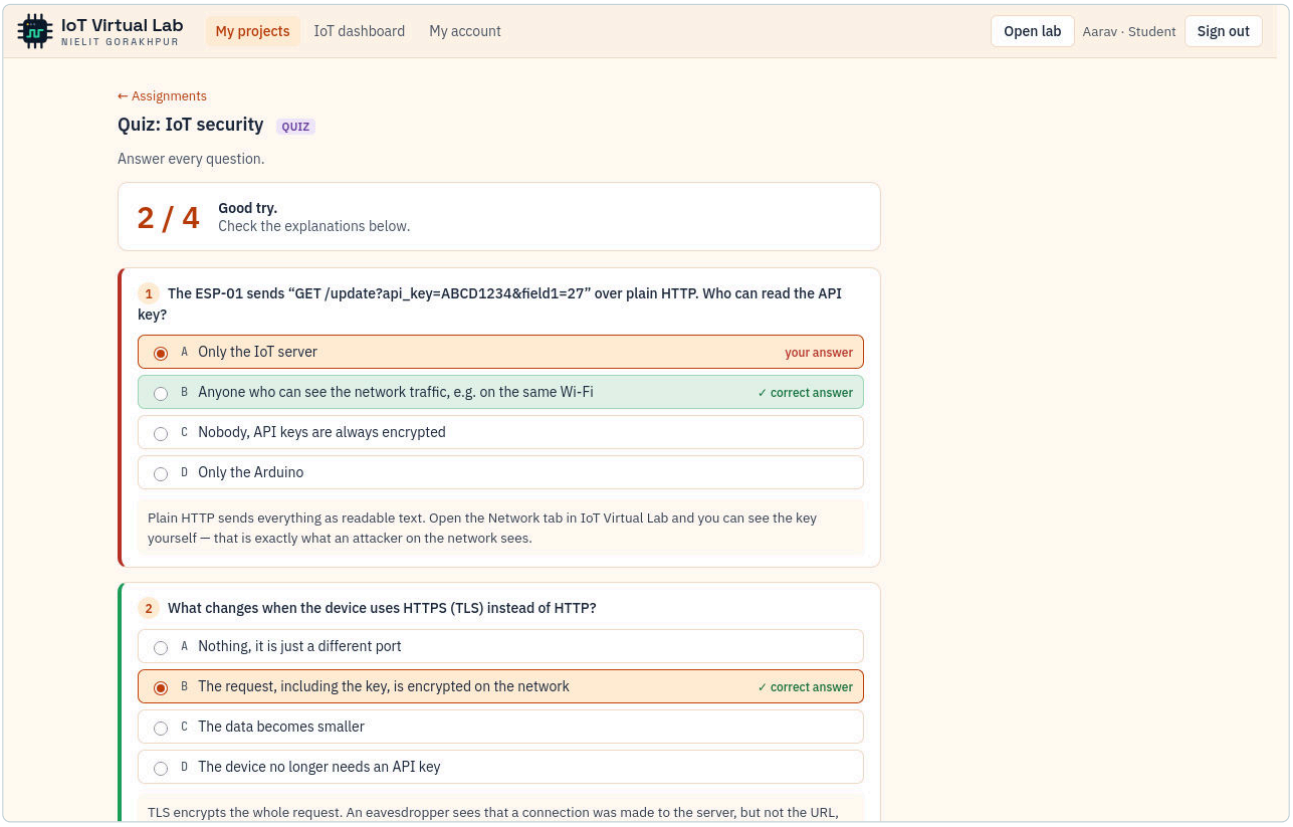


Figure 8. Quiz results, when your teacher has chosen to show the answers.

Your teacher decides whether you see the correct answers and explanations, and whether you may try again before the due date. A new attempt replaces the old one.

11. 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.

Reading an IR remote

```
#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.

12. 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)**.

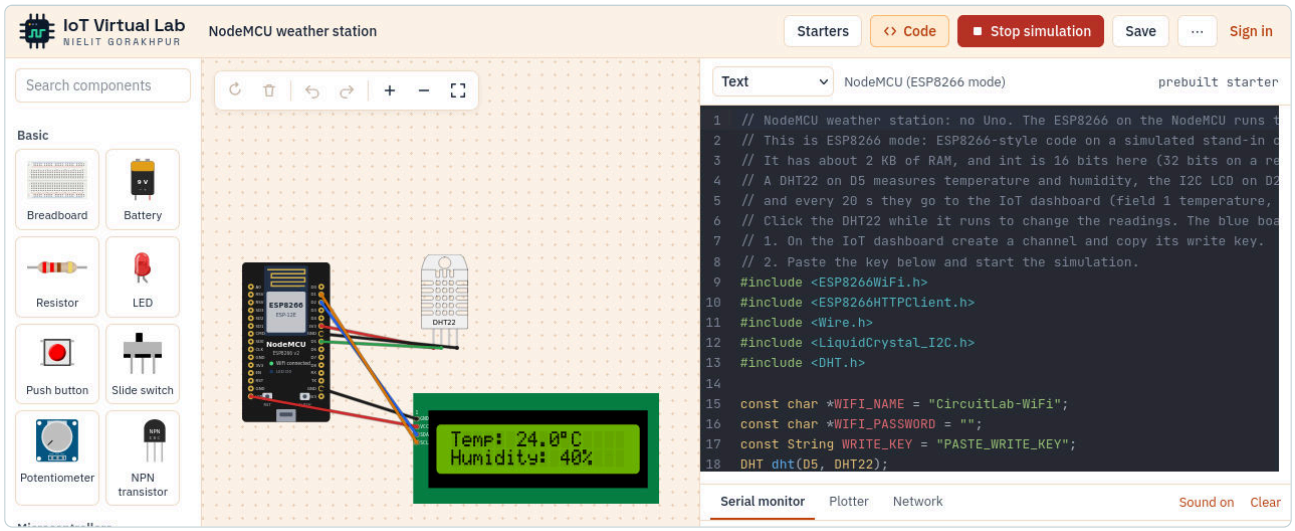


Figure 9. The NodeMCU weather station starter: the NodeMCU runs the sketch itself, with no Uno.

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.

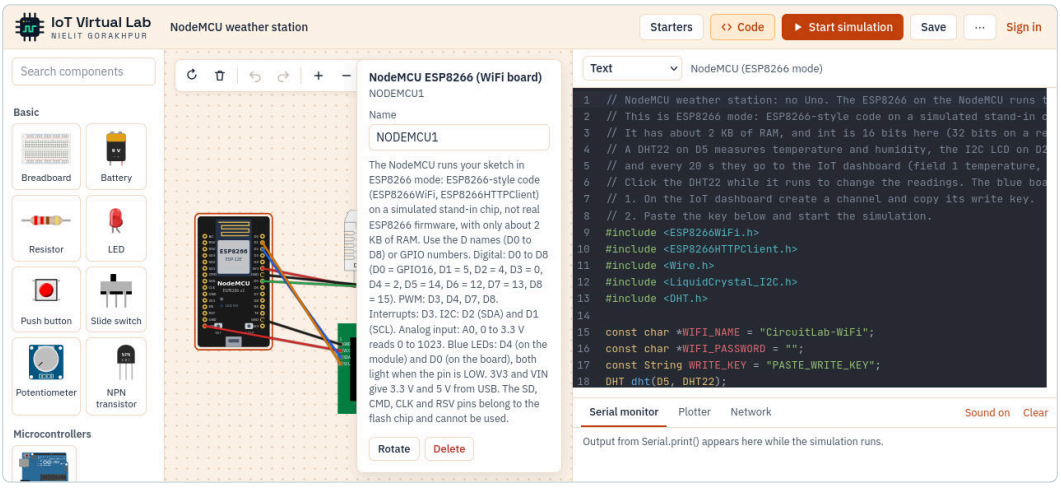


Figure 10. Click a board to see the pins that work, in its part panel.

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.

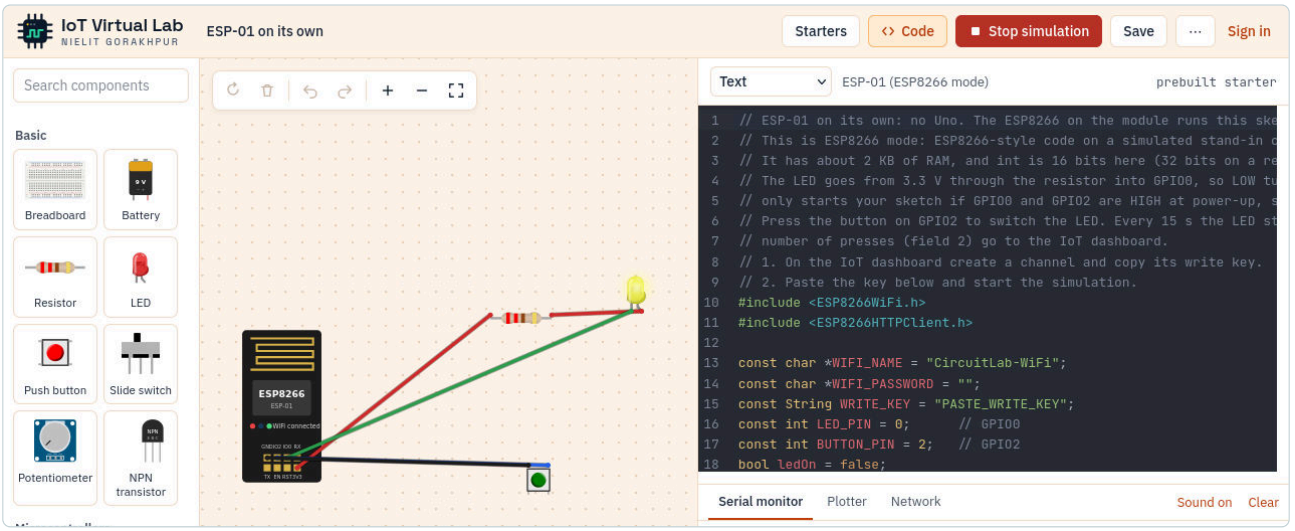


Figure 11. ESP-01 on its own: the button on GPIO2 has switched the LED on GPIO0.

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.

13. All components and starters

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

Starter circuits

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

14. IoT dashboard

- 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.

> **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.

15. Send data to real ThingSpeak

Your simulated WiFi module can also send readings to a real channel on ThingSpeak, an online IoT service.

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.

16. 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**.

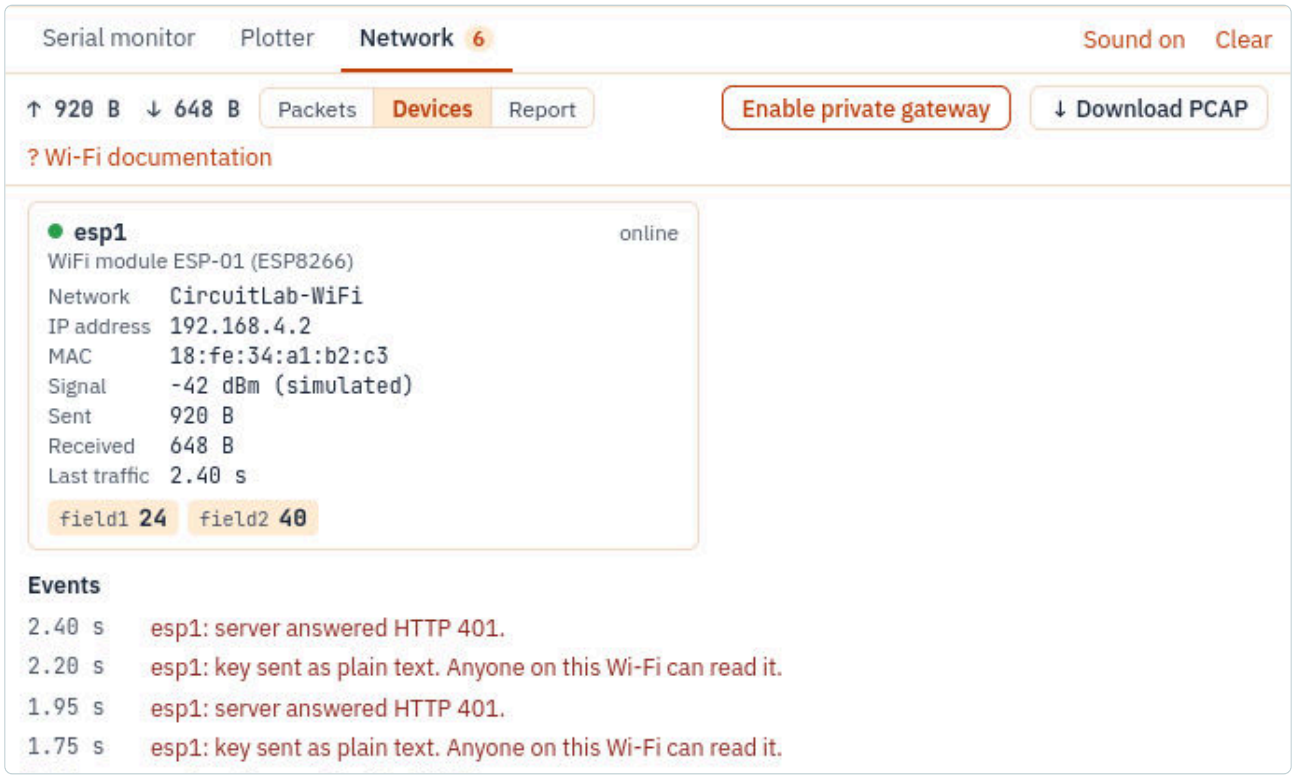


Figure 12. The Devices view: one card for each wireless part, with events below.

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

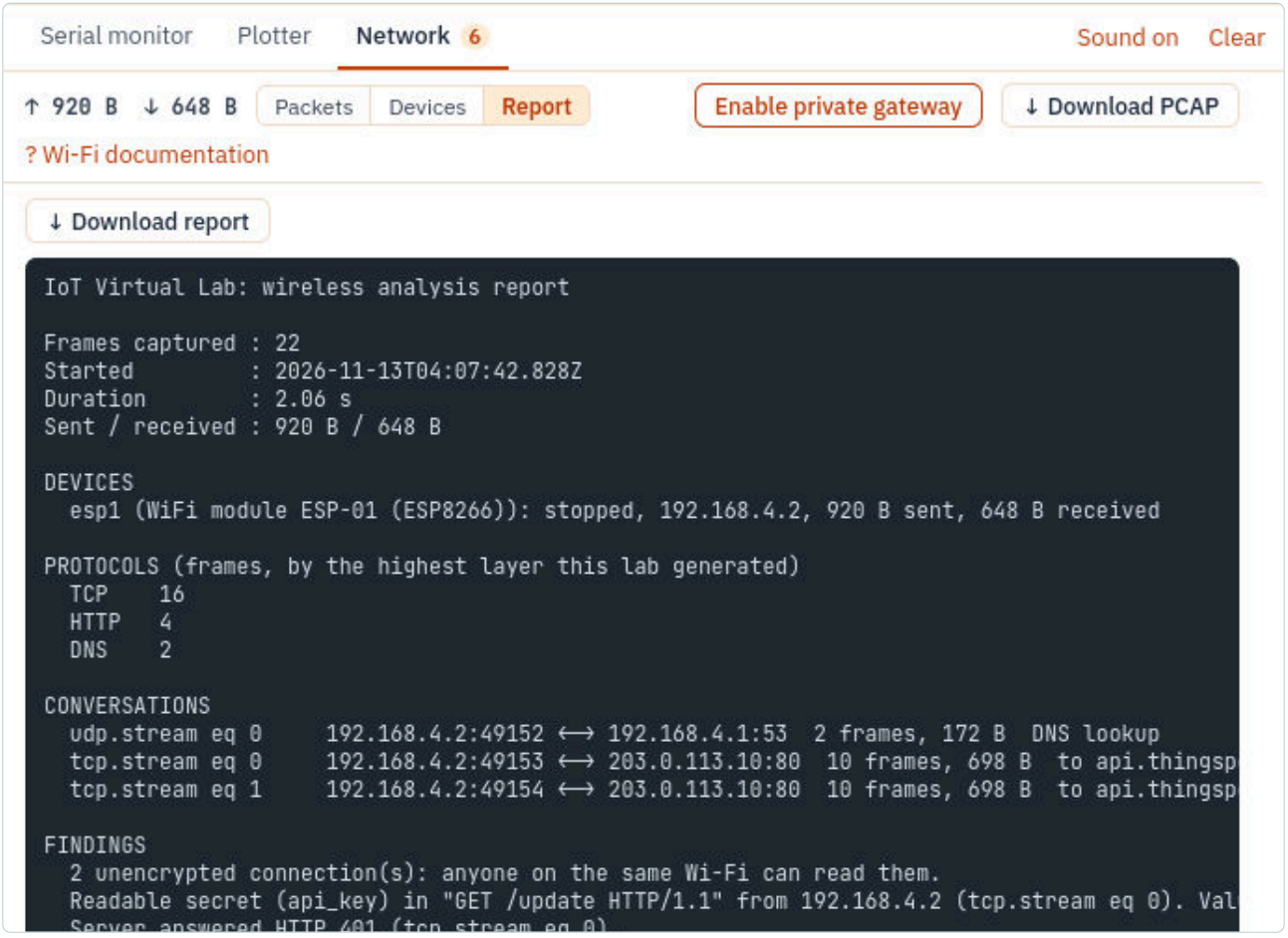


Figure 13. The Report view, ready to download as a text file.

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.

17. Sharing and saving

- **Save** stores the project in your account when you are signed in. If you are not signed in, it is saved only in that browser.
- **⋯ → Get share link** gives a link that lets anyone open a copy of your circuit. They cannot change your project, so do not put a write key in a project you share.
- **⋯ → Download project (.json)** keeps a copy on your computer, and **Open project file…** loads it again.
- **⋯ → Download code (.ino)** saves your program for use in the Arduino software with a real board.

> The teachers of your classes can open and run every project you save to your account, not only assignments, so that they can check your lab work and help you.

18. Your account

My account lets you change your password and see recent sign-ins. A short sentence such as `red-led-blinks-fast` is easy to remember and hard to guess. Check the list for sign-ins that were not you, and tell your teacher if you see any.

19. If something goes wrong

Problem	What to do
"No class has this code"	Check the code with your teacher. Letters and numbers must match.
"Someone in this class already uses this nickname"	Choose another. If it is you, use the **Sign in** tab.
"The security code is not correct" or "has expired"	Type the new code shown, or press **New code**.
"Nickname or password is not correct"	Check the class code and spelling. Ask your teacher to reset the password if needed.
"Too many failed sign-in attempts"	Wait 15 minutes, or ask your teacher to reset your password.
Cannot compile, or told to sign in	Sign in first. Changed code cannot compile without an account.
"Code has errors"	Read the first error under the code panel and fix that line.
ThingSpeak: the reply is 0	The key is wrong, or you sent again too soon. Check the key and keep 15 seconds or more between readings. A IoT Virtual Lab dashboard key must be used on that dashboard, not on ThingSpeak.
"Too many ThingSpeak requests"	Add a delay of 15 seconds or more. The limit is 12 requests a minute.
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	Check that the private gateway is off in the **Network** tab.
A sensor always reads 0, or a part shows a warning	Check its power wires (VCC, GND) and its signal wire. Click the part to read the warning.
Page looks wrong after an update	Press Ctrl+F5 to reload it.
Nothing else helps	Ask your teacher.