



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

ROBOTICS AND ELECTRONICS

IoT Basics with NodeMCU

Connect sensors to Wi-Fi with an ESP8266 board

INSIDE THIS BOOK

- What the Internet of Things means
- Meet the NodeMCU board
- Setting up the Arduino IDE
- Your first sketch: blink
- Connecting to Wi-Fi
- ...and 6 more chapters, practice questions and a glossary

Intermediate

Students of class 9 and above, diploma and BTech beginners

engineergurukul.com

+91 97116 99511

IoT Basics with NodeMCU

Learn what the Internet of Things is and build real Wi-Fi projects with a NodeMCU ESP8266 board. You will set up the Arduino IDE, blink an LED, switch it from your phone through a tiny web server, show DHT11 temperature readings on a web page, and understand the cloud, MQTT and security ideas behind real IoT products.

WHO IT IS FOR	Students of class 9 and above, diploma and BTech beginners
LEVEL	Intermediate
SUBJECT AREA	Robotics and electronics
WHAT IS INSIDE	11 chapters, practice questions with answers, and a glossary

CONTENTS

1	What the Internet of Things means	4
2	Meet the NodeMCU board	5
3	Setting up the Arduino IDE	7
4	Your first sketch: blink	8
5	Connecting to Wi-Fi	10
6	A tiny web server to switch an LED	12
7	Reading a DHT11 and showing it on a web page	15
8	Sending data to the cloud and the MQTT idea	18
9	Security basics for IoT	20
10	Five project ideas	21
11	Troubleshooting	22

Practice questions	24
Answers	25
Glossary	26

Free ebook: you may read, print and share it for learning; it may not be sold or changed. © 2026 Gurukul Technicals Institute. First edition, October 2026. Contact details are on the back cover.

What the Internet of Things means

Look around your home. Some things in it can talk to the internet. A camera can send a picture to a phone in another city. A water pump can tell an app that the tank is full. When ordinary things are joined to the internet so that they can send information or receive commands, we call it the **Internet of Things**, or **IoT** for short.

A phone was always meant to be online. A water tank, a plant pot or a street light was not. IoT adds a tiny computer and a connection to these everyday objects.

The four parts of every IoT device

Almost every IoT project, big or small, has the same four parts.

Part	What it does	Example
Sensor	Measures something in the real world	A temperature sensor in a cold store
Controller	A small computer that reads the sensor and decides what to do	An ESP8266 chip
Connection	Sends data out and receives commands	Wi-Fi to the home router
Action or display	Shows the data or does a job	A web page, an alarm, a motor

A **sensor** turns something like heat, light or distance into an electrical signal. A **controller** is a small computer on a single chip.

EXAMPLE

Mr Iyer runs a dairy near Coimbatore. His milk must stay cold. A sensor inside the chiller measures the temperature every minute. A controller sends the reading over Wi-Fi to a server. If the milk gets too warm, his phone shows an alert, even when he is at the market. That is a complete IoT system: sense, decide, connect, act.

Where IoT is used

- **Homes:** smart bulbs, door bells with cameras, water-level alarms.
- **Farms:** soil-moisture sensors that tell the farmer when to water.
- **Cities:** street lights that switch on at dusk, bins that report when full.
- **Factories:** machines that report heat or shaking before they break down.

Why we start with the NodeMCU

To learn IoT you need a controller with Wi-Fi built in that is cheap and easy to program. The **NodeMCU** board is a popular choice. It carries an **ESP8266** chip, which is a small computer with Wi-Fi inside it. You can program it with the same free Arduino software that many schools already use.

DID YOU KNOW

The ESP8266 chip is made by Espressif, a company based in China. The name NodeMCU first belonged to a project that ran programs written in a language called Lua. Today most students program NodeMCU boards with the Arduino IDE instead, and that is what this book does.

CHAPTER 2

Meet the NodeMCU board

Hold the board with the USB socket facing you. The metal-covered square at the far end is the **ESP-12E module**, and the ESP8266 chip sits under its cover. The zigzag gold line on the module is the Wi-Fi antenna.

The parts you can see

- **USB socket:** gives the board power and lets your computer send programs to it.
- **USB-to-serial chip:** converts USB signals into a form the ESP8266 understands. It is usually a chip called CP2102 or CH340.
- **RST button:** restarts your program from the beginning.
- **FLASH button:** sometimes needed when uploading; usually you can ignore it.
- **Two rows of pins:** these connect to LEDs, sensors and other parts.

Boards from different makers look a little different, but the pin labels are the same.

D1 is not pin 1

Read this slowly, because it confuses everyone at first. The pins printed on the board are named **D0**, **D1**, **D2** and so on. Inside the chip, the same pins have different numbers, called **GPIO numbers**. GPIO stands for General Purpose Input Output. It means a pin that your program can use as an input or an output.

So the pin marked D1 on the board is GPIO5 inside the chip, and D2 is GPIO4.

Label on board	GPIO number	Notes for beginners
D0	GPIO16	Also used to wake the board from deep sleep

Label on board	GPIO number	Notes for beginners
D1	GPIO5	Safe for LEDs and sensors
D2	GPIO4	Safe for LEDs and sensors
D3	GPIO0	Boot pin: keep it free when starting up
D4	GPIO2	Boot pin; the blue LED on the module is here
D5	GPIO14	Safe for LEDs and sensors
D6	GPIO12	Safe for LEDs and sensors
D7	GPIO13	Safe for LEDs and sensors
D8	GPIO15	Boot pin: keep it free when starting up
RX and TX	GPIO3 and GPIO1	Used by the USB link; leave them alone
A0	ADC	Reads a changing voltage, such as from a light sensor

A **boot pin** is a pin the chip checks at the moment it starts. If a part pulls a boot pin the wrong way, the board may refuse to start your program. For your first projects, use D1, D2, D5, D6 and D7.

In the Arduino software, when you choose the NodeMCU board, you can write **D1** in your code and the software turns it into GPIO5 for you. If you write just **5**, that also means GPIO5, which is D1. If you write **1**, you get GPIO1, which is the TX pin. That mistake can stop your board from talking to the computer.

TIP

Always write the board label, such as **D1** or **D5**, in your code. It matches what you see on the board, and you will make fewer wiring mistakes.

The power pins

- **3V3:** gives out 3.3 volts from the board's own regulator. Use it to power small sensors.
- **GND:** ground, the negative side. Every part in your circuit must share this ground.
- **Vin:** you may feed 5 volts here when the board is not on USB. Beginners should simply use USB.

The 3.3 volt rule

The ESP8266 works at **3.3 volts**, not 5 volts like an Arduino Uno. Its pins send out 3.3 V for HIGH and expect at most about 3.3 V coming in.

SAFETY

Never connect a 5 V signal straight to a NodeMCU pin. Many sensors made for the Arduino Uno send out 5 V signals. Use a sensor that works at 3.3 V, or ask your teacher about a voltage divider or a level shifter. Too much voltage on a pin can damage the chip for good.

On most NodeMCU boards, the A0 pin reads voltages from 0 to 3.3 V and gives your program a number from 0 to 1023.

REMEMBER

D1 on the board is GPIO5 inside the chip. Write `D1` in your code, power sensors from 3V3, and never feed 5 V into a pin.

CHAPTER 3

Setting up the Arduino IDE

The **Arduino IDE** is a free program for writing code and sending it to a board. IDE stands for Integrated Development Environment, which means one program that lets you write, check and upload your code. It works on Windows, macOS and Linux. Download it from arduino.cc.

At first the Arduino IDE knows only Arduino's own boards. You must add the ESP8266 boards once, by installing the **board package**.

Step 1: Tell the IDE where to find ESP8266 boards

1. Open the Arduino IDE.
2. Open **File**, then **Preferences**.
3. Find the box called **Additional boards manager URLs**.
4. Type this address exactly into the box, then click **OK**.

```
https://arduino.esp8266.com/stable/package_esp8266com_index.json
```

If the box already has an address in it, put a comma after it and then add this one.

Step 2: Install the board package

1. Open the **Boards Manager**. In the Arduino IDE 2 it is an icon on the left side bar. You can also reach it from **Tools**, then **Board**.
2. Type **esp8266** in the search box.
3. Find **esp8266 by ESP8266 Community** and click **Install**.
4. Wait while it downloads. It is large, so it can take several minutes.

Step 3: Choose your board and port

1. Plug the NodeMCU into your computer with a USB cable.
2. Open **Tools**, then **Board**, then **esp8266**, and pick **NodeMCU 1.0 (ESP-12E Module)**.
3. Open **Tools**, then **Port**, and pick the new port. On Windows it looks like COM3 or COM5. On Linux it looks like `/dev/ttyUSB0`.

A **port** is the address of the USB connection to your board. If no new port appears, read the next box.

TIP

Some USB cables only charge a phone and carry no data. If your computer does not notice the board, try another cable first. If it still fails, the computer may need a driver for the board's USB chip. Look at the small chip near the USB socket. Search for the driver by its name, CP2102 or CH340, on the chip maker's official website.

EXAMPLE

Kabir in Lucknow spent an hour wondering why no port appeared. He was using the short cable from a power bank. With a phone data cable, COM4 appeared at once.

TRY THIS

Unplug the board and open the **Port** list. Plug it back in and open the list again. The port that appears is your board.

CHAPTER 4

Your first sketch: blink

A program for an Arduino-style board is called a **sketch**. Every sketch has two main parts.

- `setup()` runs once, when the board starts or restarts.
- `loop()` runs again and again, forever, after `setup` finishes.

Blink the built-in LED

Most NodeMCU boards have a small blue LED on the metal module. It is joined to GPIO2, which is the D4 pin. This LED is **active LOW**: it lights up when the pin is LOW and goes dark when the pin is HIGH. That is the opposite of what most people expect.

CPP

```
// Blink the small blue LED on the NodeMCU's metal module.
// That LED is on GPIO2 (labelled D4) and lights when the pin is LOW.
const int LED_PIN = D4;

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

void loop() {
  digitalWrite(LED_PIN, LOW); // LED on
  delay(500);                // wait half a second
  digitalWrite(LED_PIN, HIGH); // LED off
  delay(500);
}
```

Click the **Upload** button, the arrow at the top. The IDE first **compiles** your sketch, which means it turns your code into instructions the chip can run. Then it sends them to the board. When the upload is done, the blue LED starts to blink.

What each line does

Line	Meaning
<code>const int LED_PIN = D4;</code>	Gives the pin a name, so the code is easy to read
<code>pinMode(LED_PIN, OUTPUT);</code>	Sets the pin to send out a signal
<code>digitalWrite(LED_PIN, LOW);</code>	Puts the pin at 0 V, so this LED lights
<code>delay(500);</code>	Waits 500 milliseconds, that is half a second

Talking to the computer: the Serial Monitor

The board can send text back to your computer through the USB cable, which helps you find problems.

CPP

```
void setup() {
  Serial.begin(115200);
  Serial.println();
  Serial.println("Hello from the NodeMCU!");
}

void loop() {
  Serial.print("The board has been on for ");
  Serial.print(millis() / 1000);
  Serial.println(" seconds.");
  delay(1000);
}
```

Upload it, then open **Tools**, then **Serial Monitor**. At the bottom right of the Serial Monitor, set the speed to **115200 baud**. **Baud** is the speed of the serial link. The number in your code and the number in the Serial Monitor must match, or you will see strange symbols.

DID YOU KNOW

When the ESP8266 starts, it prints a few lines of its own at a different speed. So a few strange characters at the very top of the Serial Monitor are normal. Your own messages after that should be clear.

TRY THIS

Change both `delay(500)` lines in the blink sketch to `delay(100)`. Upload again. Then try 100 for on and 900 for off. Can you make the LED flash twice quickly, then rest for one second?

CHAPTER 5

Connecting to Wi-Fi

Now for the part that makes the ESP8266 special: with a few lines of code, it joins the same Wi-Fi network as your phone.

REMEMBER

The ESP8266 works only with **2.4**

GHz Wi-Fi. Many new routers also send out a 5 GHz network, and the board cannot see that one. If your router shows two network names, use the one for 2.4 GHz. A mobile hotspot also works if it is set to the 2.4 GHz band.

A sketch that joins Wi-Fi

CPP

```
#include <ESP8266WiFi.h>

const char* WIFI_NAME = "your-wifi-name";
const char* WIFI_PASSWORD = "your-wifi-password";

void setup() {
  Serial.begin(115200);
  delay(100);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected. My IP address is ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

Replace the two words in quotes with your real network name and password. Both are case-sensitive, which means capital and small letters must match exactly. In the next chapter you will move them into a private file.

How it works

- `#include <ESP8266WiFi.h>` brings in the Wi-Fi code that came with the board package.
- `WiFi.mode(WIFI_STA)` makes the board a **station**, which means a device that joins a network, just like your phone does.
- The `while` loop waits, printing a dot every half second, until the status becomes `WL_CONNECTED`.
- `WiFi.localIP()` gives the board's **IP address**. This is the number that other devices on the same network use to reach it, for example 192.168.1.7.

When the board joins, your router gives it an IP address, like a hostel warden giving a new student a room number. In the next chapter, your phone will visit that number in a web browser.

EXAMPLE

Sana's router in Bhopal gave her board the address 192.168.0.105. Her phone was 192.168.0.102. The first three parts match, which shows that both are on the same home network.

TIP

If the dots go on forever, check three things in order: the 2.4 GHz network, the spelling of the name and password, and the distance from the router. Move the board closer and try again.

CHAPTER 6

A tiny web server to switch an LED

A **web server** is a program that sends web pages when a browser asks for them. Your ESP8266 can run a very small one. Any phone or laptop on the same Wi-Fi can then open its page. You will make a page with two links, **Turn ON** and **Turn OFF**, that switch a real LED.

Wiring the LED

You need one LED, one 220 Ω resistor, a breadboard and two jumper wires. Ω is the symbol for ohm, the unit of resistance.

1. Push the LED into the breadboard. The longer leg is the positive side, called the **anode**.
2. Connect a jumper wire from **D1** to the long leg of the LED.
3. Connect the 220 Ω resistor from the short leg to the board's **GND**.

This LED is **active HIGH**: it lights when D1 is HIGH. That is the normal way, unlike the blue LED on D4.

EXAMPLE

Why 220 Ω ? A red LED uses about 2 V, so about 1.3 V is left for the resistor from a 3.3 V pin. Current = voltage $\div$ resistance = $1.3 \div 220$, which is about 0.006 A, or 6 milliamps. That is bright enough to see and small enough to keep the pin safe.

Keep your password in its own file

Before writing the server, make a separate file for your Wi-Fi details. In the Arduino IDE 2, use the small menu at the right end of the tab bar and choose **New Tab**. Name it `secrets.h` and type:

CPP

```
// secrets.h: keep this file private. Never share or upload it.
#define WIFI_NAME "your-wifi-name"
#define WIFI_PASSWORD "your-wifi-password"
```

You will learn why this matters in the chapter on security.

The server sketch

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "secrets.h"

const int LED_PIN = D1; // GPIO5, LED through a 220 ohm resistor
ESP8266WebServer server(80);
bool ledOn = false;

String page() {
    String html = "<!DOCTYPE html><html><head>";
    html += "<meta name='viewport' content='width=device-width, initial-scale=1'>";
    html += "<title>LED switch</title></head><body>";
    html += "<h1>LED is ";
    html += ledOn ? "ON" : "OFF";
    html += "</h1><p><a href='/on'>Turn ON</a></p>";
    html += "<p><a href='/off'>Turn OFF</a></p></body></html>";
    return html;
}

void handleRoot() {
    server.send(200, "text/html", page());
}

void handleOn() {
    ledOn = true;
    digitalWrite(LED_PIN, HIGH);
    server.send(200, "text/html", page());
}

void handleOff() {
    ledOn = false;
    digitalWrite(LED_PIN, LOW);
    server.send(200, "text/html", page());
}

void setup() {
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println();
    Serial.print("Open this address in a browser: http://");
    Serial.println(WiFi.localIP());
    server.on("/", handleRoot);
    server.on("/on", handleOn);
    server.on("/off", handleOff);
    server.begin();
}
```

```
void loop() {  
  server.handleClient();  
}
```

Try it

1. Upload the sketch and open the Serial Monitor at 115200 baud.
2. Note the IP address it prints, for example 192.168.1.7.
3. Make sure your phone is on the **same Wi-Fi**, not on mobile data.
4. Type the IP address into the phone's browser and open it.
5. Tap **Turn ON**. The LED lights. Tap **Turn OFF**. It goes dark.

How it works

- `ESP8266WebServer server(80);` creates a server on **port 80**, the usual door for web pages.
- `server.on("/on", handleOn);` says: when a browser asks for the path `/on`, run `handleOn`.
- `server.send(200, "text/html", ...)` sends the page back. 200 means "OK, here it is".
- `server.handleClient();` keeps checking whether a browser is asking for something. Without it, the server never answers.

TRY THIS

Add a third link called **Blink** at the path `/blink`. In its handler, flash the LED three times, then send the page. Remember to add a matching `server.on` line in `setup()`.

CHAPTER 7

Reading a DHT11 and showing it on a web page

The **DHT11** is a cheap sensor that measures **temperature** and **humidity**. Humidity means how much water vapour is in the air. It is high on a muggy monsoon day in Mumbai and much lower on a dry winter day in Delhi.

The DHT11's readings can be a degree or two off, which is fine for learning. It is also slow, so read it only once every second or two.

Wiring the DHT11 module

Most students buy the DHT11 fixed on a small board, called a module, with three pins. The module already has the extra resistor the sensor needs. Read the labels printed on your module, because the pin order is not the same on every one.

Module pin	Connect to
+ or VCC	3V3 on the NodeMCU
OUT or DATA	D5 (GPIO14)
– or GND	GND on the NodeMCU

If you have the bare sensor with four legs and no module, it needs a 10 k Ω resistor between the data leg and VCC. Ask your teacher to check the wiring before you power it.

SAFETY

Check the + and – wires twice before you plug in the USB cable. A sensor connected backwards can get hot and be damaged. If anything feels hot or smells of burning, unplug the cable at once.

Install the library

A **library** is ready-made code that someone has written so you do not have to. Open **Sketch**, then **Include Library**, then **Manage Libraries**. Search for **DHT sensor library** by Adafruit and click **Install**. When the IDE asks to install the libraries it depends on, choose to install them all.

The weather page sketch

Keep the same `secrets.h` tab in this sketch too.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#include "secrets.h"

#define DHT_PIN D5      // GPIO14
#define DHT_TYPE DHT11

DHT dht(DHT_PIN, DHT_TYPE);
ESP8266WebServer server(80);

float temperature = NAN;
float humidity = NAN;
unsigned long lastRead = 0;

void readSensor() {
    float t = dht.readTemperature(); // degrees Celsius
    float h = dht.readHumidity();    // percent
    if (isnan(t) || isnan(h)) {
        Serial.println("Could not read the DHT11. Check the wires.");
        return;
    }
    temperature = t;
    humidity = h;
}

void handleRoot() {
    String html = "<!DOCTYPE html><html><head>";
    html += "<meta name='viewport' content='width=device-width, initial-scale=1'>";
    html += "<meta http-equiv='refresh' content='10'>";
    html += "<title>Room weather</title></head><body><h1>Room weather</h1>";
    if (isnan(temperature)) {
        html += "<p>No reading yet.</p>";
    } else {
        html += "<p>Temperature: " + String(temperature, 1) + " °C</p>";
        html += "<p>Humidity: " + String(humidity, 0) + " %</p>";
    }
    html += "</body></html>";
    server.send(200, "text/html", html);
}

void setup() {
    Serial.begin(115200);
    dht.begin();
    WiFi.mode(WIFI_STA);
    WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println();
    Serial.println(WiFi.localIP());
    server.on("/", handleRoot);
}
```

```

server.begin();
}

void loop() {
  server.handleClient();
  if (millis() - lastRead >= 2000) {
    lastRead = millis();
    readSensor();
  }
}

```

How it works

- **NAN** means "not a number". We start with it so that the page can say "No reading yet" until the first good reading arrives. `isnan()` checks for it.
- The loop reads the sensor every 2,000 milliseconds, using `millis()` instead of `delay()`. This keeps the web server free to answer the phone at any moment. A long `delay()` would make the page slow to open.
- The line with `refresh` tells the browser to reload the page every 10 seconds, so the numbers update on their own.
- `String(temperature, 1)` turns the number into text with one digit after the decimal point, such as 31.4.

EXAMPLE

Ananya in Kochi left her board on her study table for a day. At 2 pm the page showed 31.0 °C and 78 % humidity. After an evening shower it showed 27.0 °C and 88 %. She wrote the readings in a table and made a line graph for her school project.

TIP

If the Serial Monitor keeps printing "Could not read the DHT11", check that the data wire is on D5, that the code says `DHT11` and not `DHT22`, and that the module is powered from 3V3 and GND.

CHAPTER 8

Sending data to the cloud and the MQTT idea

Your weather page works only while your phone is on the same Wi-Fi as the board. To see your data from anywhere, the board must send it to a computer on the internet. We call this sending data **to the cloud**. The **cloud** simply means computers in big data centres that you use over the internet. You never see them, but they are always on.

How a cloud IoT system works

1. The board reads the sensor, for example every minute.
2. It sends the reading over the internet to a cloud service.
3. The cloud service stores every reading with the time.
4. A dashboard in a browser or a phone app shows graphs, from anywhere in the world.
5. The service can also send an alert, such as a message when the temperature crosses a limit.

Many companies offer IoT cloud services with ready-made dashboards. Many have a free plan with limits on how much data you can send. Plans change, so read the current terms on the service's own website before you choose. Most services give you a secret **key** to put in your sketch. Keep it private, just like your Wi-Fi password.

EXAMPLE

Riya grows tomatoes on her terrace in Jaipur. Her board sends the soil moisture reading to a cloud dashboard every 10 minutes. At school, her phone shows that the soil is getting dry. She calls home and asks her grandmother to water the plants.

Two ways to send data

Way 1: a web request. The board visits a special web address on the service, carrying the reading with it, a bit like filling in a form. It is easy, but each request carries a lot of extra information, which is heavy for a tiny device.

Way 2: MQTT. MQTT is a light messaging system built for small devices on slow or unreliable networks. It has three roles.

Role	What it does	Example
Publisher	Sends a message on a topic	The board in the bedroom
Broker	Receives every message and passes it on	A server on the internet
Subscriber	Asks for messages on topics it cares about	Your phone app

A **topic** is a name for a stream of messages, written like a folder path, for example `home/bedroom/temperature`. The board publishes "28.5" to that topic. The broker passes it to every subscriber of that topic. The board and the phone never talk directly. They both talk only to the broker.

Think of a school notice board. A teacher pins a notice under "Class 9 Sports", and every student who checks that section sees it. The notice board is the broker.

DID YOU KNOW

MQTT works both ways. Your phone can publish "ON" to a topic such as `home/bed room/light`, and a board that subscribes to that topic can switch its LED.

CHAPTER 9

Security basics for IoT

Every device on a network is a door into that network. If the door is weak, a stranger may get in. Real IoT products have been hacked by the thousands because simple rules were ignored. Learn them now, while your projects are small.

Do not hard-code passwords in shared code

Hard-coding means writing a secret, such as a password, directly inside your program. If you then share that program with a friend, post it in a group chat or upload it to a code-sharing website, everyone who sees it can read your Wi-Fi password.

That is why this book keeps the Wi-Fi name and password in a separate `secrets.h` file.

1. Share only the main sketch, never `secrets.h`.
2. When you share a project, give a sample file with fake values, such as "your-wifi-name".
3. If you ever share a real password by mistake, change it on your router at once.

EXAMPLE

Arjun from Patna posted his code in a public group to ask for help. Within minutes, a classmate messaged him his home Wi-Fi password, read straight from the code. Arjun's father changed the router password that evening, and Arjun moved his secrets to their own file.

Change default passwords

Many devices, such as routers and cameras, come with a **default password** set at the factory, like "admin" or "1234". Attackers know these lists. Always change them during setup, and use a long password that is hard to guess.

DID YOU KNOW

In 2016, harmful software called Mirai took over huge numbers of cameras and video recorders, mostly by trying common factory passwords. It then used them together to flood websites with traffic until they slowed down or stopped.

More good habits

- **Your tiny web server has no password.** Anyone on the same Wi-Fi can open its page and switch your LED. That is fine for learning, but never use it to control something important.
- **Do not open your board to the whole internet.** Some router settings make a home device reachable from outside. Leave them alone.
- **Keep software updated.** Updates for the board package, libraries and router often fix security holes.
- **Use a guest network if your router has one.** It keeps test gadgets away from family laptops and phones.

SAFETY

Never use an IoT board, relay or switch to control a fan, light or any appliance that plugs into the wall. Mains electricity can kill. Keep all your projects on USB power or batteries of 12 V or less.

CHAPTER 10

Five project ideas

Each project joins the building blocks you know in a new way. Ask a teacher to check your wiring before you power each one.

1. Room weather station

Use the DHT11 sketch from this book. Add the highest and lowest temperature of the day to the web page. To do this, keep two more variables and update them whenever a new reading is higher or lower.

2. Plant thirst alarm

A **soil moisture sensor** gives a voltage that changes with how wet the soil is. Choose a capacitive sensor that works at 3.3 V, and connect its output to A0. Read it with `analogRead(A0)`. Water the soil and note the number. Let it dry and note the number again. Use those two numbers to decide when the page should say "Please water me".

3. Door open alert

A **reed switch** closes when a magnet comes near. Fix the switch to a door frame and the magnet to the door. Wire one side of the switch to D2 and the other to GND, and use `pinMode(D2, INPUT_PULLUP)`. With this setting, the pin reads LOW while the switch is closed and HIGH when the door opens and the magnet moves away. Show "Door closed" or "Door open" on a web page.

4. Water tank level indicator

An ultrasonic distance sensor fixed above a tank measures how far away the water surface is. A full tank gives a short distance, and an empty tank gives a long one. Many common ultrasonic sensors run on 5 V and send back a 5 V signal. Remember the 3.3 volt rule: choose a model that works at 3.3 V, or ask your teacher to add a voltage divider on the signal wire.

5. Study timer with a light

Make a web page with buttons for 25, 45 and 60 minutes. When you tap one, an LED glows steadily while you study and blinks when the time is up. This uses `millis()`, just like the DHT11 sketch.

EXAMPLE

Meera and Vikram from Pune built the water tank indicator for their school exhibition. They showed the tank level on a phone page with three words: Low, Half and Full. They powered the board from a USB power bank, so no mains wiring was needed anywhere.

TRY THIS

Before coding any project, write five answers: what it senses, which pin, which voltage the sensor needs, what it shows or does, and what could go wrong. Show them to your teacher.

CHAPTER 11

Troubleshooting

Every maker gets stuck. Good makers do not guess. They check one thing at a time, simplest first.

Problems with the computer and upload

What you see	Likely cause	What to do
No port appears	Charge-only cable, or no driver	Try a data cable; install the CP2102 or CH340 driver
"Board not found" or upload fails	Wrong board or port chosen	Check Tools , Board and Port again
Upload still fails	The board did not enter upload mode	Hold FLASH, start the upload, let go when it begins
Error says a file is missing	Library not installed	Install the library named in the error

Problems after upload

What you see	Likely cause	What to do
Strange symbols in Serial Monitor	Baud speed does not match	Set it to 115200, the same as <code>Serial.begin</code>
Dots forever, no Wi-Fi	5 GHz network, wrong password, too far	Use 2.4 GHz, check spelling, move closer
Board will not start your program	Something on D3, D4 or D8	Unplug parts from those pins and restart
Board keeps restarting	Weak power, or a short circuit	Try another USB port or cable; check wiring
DHT11 reads fail	Loose wire, wrong pin, wrong type	Check D5, <code>DHT11</code> in code, 3V3 and GND
Phone cannot open the page	Phone on mobile data, or wrong address	Join the same Wi-Fi; read the IP again
Page worked yesterday but not today	The router gave a new IP address	Check the Serial Monitor for the new address

Read the error message

When the code does not compile, the IDE shows red text at the bottom. Scroll to the **first** error. It usually gives a line number and a short reason, such as a missing semicolon. Fix that one error and compile again. One mistake often causes many error lines.

EXAMPLE

Harpreet in Ludhiana got thirty lines of red errors. The first one said 'ledon' was not declared. She had typed ledon instead of led0n. One capital letter fixed all thirty lines.

Divide the problem

If a big project fails, test the parts one by one: blink first, then Wi-Fi, then the sensor. When one step fails, you know where to look.

REMEMBER

Change one thing at a time, test it, and write down what you tried. A small notebook of problems and fixes will save you hours on your next project.

Practice questions

1. What does IoT stand for, and what does it mean in one sentence?
2. Name the four parts that almost every IoT device has.
3. Which chip is on the NodeMCU board, and what makes it useful for IoT?
4. The pin marked D1 on a NodeMCU is which GPIO number inside the chip?
5. Why should you avoid connecting parts to D3, D4 and D8 in your first projects?
6. What voltage does the ESP8266 work at, and what could happen if you feed 5 V into one of its pins?
7. Why do you have to add an address in the Arduino IDE preferences before using a NodeMCU?
8. Your computer does not show any new port when you plug in the board. Give two possible reasons.
9. What is the difference between `setup()` and `loop()` in a sketch?
10. You see strange symbols in the Serial Monitor. What is the most likely reason?
11. Why can a NodeMCU not join some home networks even when the password is correct?
12. What does `WiFi.localIP()` give you, and why do you need it?
13. In the web server sketch, what happens if you remove `server.handleClient();` from `loop()`?
14. Calculate the LED current with a 220 Ω resistor if 1.3 V is across the resistor.
15. What two things does a DHT11 measure?
16. Why does the DHT11 sketch use `millis()` instead of a long `delay()`?
17. Why can you not open your weather page from your school when the board is at home?
18. Name the three roles in MQTT and say what the broker does.
19. What does hard-coding a password mean, and why is it risky?
20. Why should you change the default password on a router or camera?
21. A project worked yesterday, but today the phone cannot open the page. What should you check first?
22. Why should you never use your IoT board to switch a fan that plugs into the wall?

Answers

1. Internet of Things. It means joining everyday things to the internet so they can send information or receive commands.
2. A sensor, a controller, a connection and an action or display.
3. The ESP8266. It is a small computer with Wi-Fi built in, and it can be programmed with the Arduino IDE.
4. GPIO5.
5. They are boot pins. If a part pulls them the wrong way at start-up, the board may not start your program.
6. It works at 3.3 V. A 5 V signal on a pin can damage the chip for good.
7. The IDE knows only Arduino's own boards at first. The address tells it where to download the ESP8266 board package.
8. The cable may be charge-only, or the computer may need a driver for the CP2102 or CH340 chip.
9. `setup()` runs once when the board starts. `loop()` runs again and again after that.
10. The baud speed in the Serial Monitor does not match the number in `Serial.begin()`.
11. The ESP8266 works only on 2.4 GHz Wi-Fi, so it cannot join a 5 GHz network.
12. It gives the board's IP address on the network. You type this address into a browser to open the board's web page.
13. The server never checks for requests, so the page never opens.
14. $\text{Current} = 1.3 \div 220 \approx 0.006 \text{ A}$, about 6 milliamps.
15. Temperature and humidity.
16. A long `delay()` stops everything, so the web server cannot answer. `millis()` lets the board check the time and keep serving pages.
17. The board's web server works only on the same local Wi-Fi network. To see it from anywhere, the data must go to a cloud service.
18. Publisher, broker and subscriber. The broker receives every message and passes it to every subscriber of that topic.
19. It means writing the password inside the program. Anyone who sees the shared code can read the password.
20. Attackers know the factory passwords, so an unchanged default password lets them in easily.
21. Check the Serial Monitor for the board's IP address, because the router may have given it a new one.
22. Mains electricity can kill. Students must keep projects on USB power or batteries of 12 V or less.

Glossary

Active LOW

an LED or part that switches on when its pin is LOW.

Arduino IDE

free software for writing code and uploading it to a board.

Baud

the speed of a serial link; both ends must use the same number.

Board package

the files that teach the Arduino IDE how to program a new kind of board.

Boot pin

a pin the chip checks at start-up, which can stop the board starting if pulled the wrong way.

Broker

in MQTT, the server that receives messages and passes them to subscribers.

Cloud

computers in data centres that you use over the internet.

DHT11

a low-cost sensor that measures temperature and humidity.

ESP8266

a small, low-cost computer chip with Wi-Fi built in.

GPIO

General Purpose Input Output, a pin a program can use as an input or an output.

Hard-coding

writing a fixed value, such as a password, directly into a program.

IoT

the Internet of Things, everyday objects joined to the internet.

IP address

the number that identifies a device on a network.

Library

ready-made code that adds features to your sketch.

MQTT

a light messaging system in which devices publish and subscribe to topics through a broker.

NodeMCU

a development board that carries an ESP8266 module, a USB socket and pins.

Port

the address of the USB connection between the computer and the board.

Sensor

a part that turns heat, light, distance or similar into an electrical signal.

Serial Monitor

a window in the Arduino IDE that shows text sent by the board.

Sketch

a program written in the Arduino IDE.

Topic

in MQTT, the name of a stream of messages, such as home/kitchen/temperature.

Web server

a program that sends web pages when a browser asks for them.



Gurukul Technicals Institute

Computer education institute in Delhi, with authorised member centres, robotics classes and science projects, working models and DIY kits.

COURSES

Computer & Office Skills · Accounting & Business Software · Programming & Web Development · Design & Digital Skills · Hardware & Networking · Robotics & Coding · Other Programmes

PROJECTS AND KITS

Ready-made school projects · DIY and STEM kits · Robotics and coding · Components and spares · Engineering projects

CONTACT

Head office: A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone: 011 2721 1512 · **Mobile and WhatsApp:** +91 97116 99511

Email: engineergurukul@yahoo.co.in

Website: engineergurukul.com · **Instagram:** [@science_projects_club](https://www.instagram.com/science_projects_club)

More free ebooks for computer courses, robotics and science projects: engineergurukul.com/resources/ebooks/

Free to read, print and share. Not for sale.