IoT Final Year Project Ideas PDF

A good IoT final year project connects real sensors to a microcontroller such as an ESP32 or Raspberry Pi, sends data to a cloud or local server over MQTT or HTTP, shows it on a dashboard or app, and takes an action such as an alert or switching a relay. Smart irrigation, energy monitoring, health monitoring and smart parking are reliable choices.

Key takeaways

- A complete IoT project has four layers: sensors, device, network and application.
- ESP32 boards give Wi-Fi and Bluetooth at low cost; Raspberry Pi suits camera and edge AI work.
- MQTT is the standard light protocol; ThingSpeak, Blynk, Node-RED or Firebase speed up dashboards.
- Show a live demo with real readings and a fallback video in case Wi-Fi fails.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech students in CSE, IT, ECE and EEE who want an Internet of Things project with real hardware. Each plan lists the boards and sensors, the software stack, the modules a team of 3 or 4 can split (hardware, firmware, backend, dashboard), and the points examiners usually ask about. Component names are common modules sold by electronics shops and online stores in India.

How to choose your project

- **Buy parts early.** Order sensors in week one; delivery and faulty parts cost time.
- **Keep power simple.** USB or a 5V adapter first; add batteries only if needed.
- **Choose a cloud you can use free** (ThingSpeak, Firebase free tier, or a local Node-RED server).
- **Add one action**, not only readings: alert, relay, buzzer or SMS.

Standard level projects

1. Smart Irrigation System

- **Problem:** Water plants only when the soil is dry.
- **Hardware:** ESP32, capacitive soil moisture sensor, DHT22, relay module, small DC pump.
- **Tech stack:** Arduino IDE or PlatformIO, MQTT, Node-RED or Blynk.
- **Main modules:** Sensor reading, threshold logic, pump control, dashboard, manual override.

- **Steps:** Calibrate dry and wet readings, set thresholds, publish readings over MQTT, switch the pump through the relay.
- **Show in the viva:** Calibration values and the water saved in your test week.
- **Extensions:** Use a weather forecast API to skip watering before rain.

2. Home Energy Monitoring

- **Problem:** Track appliance power use and cost.
- **Hardware:** ESP32, PZEM-004T energy meter module (with CT coil), relay module.
- **Tech stack:** Arduino IDE, MQTT, InfluxDB, Grafana.
- **Main modules:** Meter reading, data logging, time-series database, cost calculator, alerts.
- **Steps:** Read voltage, current and energy, send every few seconds, store in InfluxDB, build Grafana panels.
- **Show in the viva:** Safe mains wiring (enclosed, guided by a faculty member) and daily usage charts.
- **Extensions:** Detect which appliance is on from its power pattern.

3. Patient Health Monitoring

- **Problem:** Send heart rate and SpO2 readings to a caregiver dashboard.
- **Hardware:** ESP32, MAX30102 pulse oximeter sensor, DS18B20 temperature sensor, OLED display.
- **Tech stack:** Arduino IDE, Firebase Realtime Database, a web or Android dashboard.
- **Main modules:** Sensor drivers, filtering, cloud upload, alert rules, caregiver view.
- **Steps:** Read and smooth the signal, upload to Firebase, alert when values cross limits.
- **Show in the viva:** Why readings are indicative only, not medical grade.
- **Extensions:** Add fall detection with an MPU6050.

4. Smart Parking System

- **Problem:** Show free parking slots on a web page.
- **Hardware:** ESP32 or NodeMCU ESP8266, HC-SR04 ultrasonic or IR sensors per slot, servo for gate.
- **Tech stack:** Arduino IDE, MQTT, Node.js, a simple web map.
- **Main modules:** Slot detection, gate control, backend, live availability page.
- **Steps:** Detect a car per slot, publish status, update the page, open the gate when a slot is free.
- **Show in the viva:** How you avoid false readings from people walking past.
- **Extensions:** Add booking with a time limit.

Advanced level projects

5. Air Quality Monitoring Network

- **Problem:** Measure air quality at several points on campus.
- **Hardware:** Several ESP32 nodes, PMS5003 or SDS011 particulate sensor, MQ-135 gas sensor, DHT22.
- **Tech stack:** MQTT broker (Mosquitto), Python backend, PostgreSQL, Grafana or a web map.
- **Main modules:** Node firmware, broker, data store, map dashboard, alert rules.
- **Steps:** Build three nodes, place them across campus, compare readings by time and place.

- **Show in the viva:** How low-cost sensor readings differ from official stations.
- **Extensions:** Add LoRa for nodes without Wi-Fi.

6. RFID and Face-based Smart Attendance

- **Problem:** Mark attendance with an RFID card and confirm with a photo.
- **Hardware:** Raspberry Pi 4, RC522 RFID reader, Pi camera module.
- **Tech stack:** Python, SQLite or MySQL, Flask dashboard.
- **Main modules:** Card reading, photo capture, attendance database, teacher dashboard.
- **Steps:** Read card ID, capture photo, store record, show daily reports.
- **Show in the viva:** Why the photo step stops proxy attendance.
- **Extensions:** Add face matching against enrolled photos.

7. Vehicle Tracking and Accident Alert

- **Problem:** Track a vehicle and send an SMS alert after a crash.
- **Hardware:** ESP32, NEO-6M GPS module, SIM800L GSM module, MPU6050 accelerometer.
- **Tech stack:** Arduino IDE, Firebase, Google Maps on a web page.
- **Main modules:** GPS reading, crash detection, SMS alert, live map.
- **Steps:** Read location, detect sudden impact from acceleration, send SMS with a map link.
- **Show in the viva:** Crash threshold choice and false alarms.
- **Extensions:** Add a cancel button before the alert is sent.

8. Industrial Machine Condition Monitoring

- **Problem:** Spot abnormal vibration in a motor before it fails.
- **Hardware:** ESP32, ADXL345 or MPU6050 accelerometer, ACS712 current sensor, small motor.
- **Tech stack:** MQTT, Python, scikit-learn, Grafana.
- **Main modules:** Vibration sampling, FFT features, anomaly model, alerts.
- **Steps:** Record normal running, extract features, train an anomaly detector, test with an unbalanced load.
- **Show in the viva:** The frequency chart for normal and faulty runs.
- **Extensions:** Run the model on the ESP32 with TinyML.

Research-oriented projects

9. LoRa-based Farm Sensor Network

- **Problem:** Send soil and weather data from fields without Wi-Fi.
- **Hardware:** Arduino or ESP32 with SX1278 LoRa modules, soil and weather sensors, one gateway node.
- **Tech stack:** LoRa libraries, MQTT, a cloud dashboard.
- **Main modules:** Field nodes, gateway, range test, battery life test, dashboard.
- **Steps:** Build two field nodes and a gateway, measure range and packet loss, log battery use.
- **Show in the viva:** Range and packet-loss results.
- **Extensions:** Add solar charging.

10. Edge AI Intruder Detection

- **Problem:** Detect people at night and alert without sending video to the cloud.
- **Hardware:** Raspberry Pi 4 or ESP32-CAM, PIR sensor, camera.
- **Tech stack:** Python, TensorFlow Lite or OpenCV DNN, Telegram bot API.
- **Main modules:** Motion trigger, on-device person detection, snapshot alert, local log.
- **Steps:** Wake on PIR, run a small detector, send a snapshot only when a person is found.
- **Show in the viva:** Privacy gain from on-device processing.
- **Extensions:** Compare power use with cloud processing.

11. IoT Security Assessment Lab

- **Problem:** Show common IoT weaknesses and fix them.
- **Hardware:** ESP32 devices, a local MQTT broker, a laptop.
- **Tech stack:** Mosquitto, Wireshark, TLS certificates.
- **Main modules:** Insecure baseline, sniffing demo, TLS and authentication fix, access control lists.
- **Steps:** Run plain MQTT, capture traffic, enable TLS and passwords, show traffic is protected.
- **Show in the viva:** Before and after packet captures.
- **Extensions:** Add over-the-air update signing.

12. Smart Water Tank and Leak Monitor

- **Problem:** Track tank level and detect leaks in a building.
- **Hardware:** ESP32, waterproof ultrasonic sensor (JSN-SR04T), YF-S201 flow sensor, solenoid valve.
- **Tech stack:** MQTT, Node-RED, a mobile dashboard.
- **Main modules:** Level sensing, flow logging, leak rule (flow at night), valve control.
- **Steps:** Log level and flow, flag unusual night flow, close the valve automatically.
- **Show in the viva:** Your leak rule and a test you ran.
- **Extensions:** Add usage forecasting.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the circuit diagram (Fritzing or a neat hand drawing), a block diagram of the four layers, and the bill of materials.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** sample readings, response time, range or battery tests, and photos of the working hardware.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- Leaving hardware wiring to the last week and finding a faulty sensor.
- Working with mains voltage without supervision.
- Hard-coding Wi-Fi passwords and API keys in public code.
- No offline fallback when college Wi-Fi blocks the device.

Frequently asked questions

Which IoT project is best for a final year?

Projects that sense, send and act: smart irrigation, energy monitoring, health monitoring and smart parking. They show the full chain from sensor to dashboard to action.

ESP32 or Raspberry Pi for an IoT project?

ESP32 for low-cost sensor nodes with Wi-Fi. Raspberry Pi when you need a camera, a database on the device, or on-device AI.

Which cloud platform should I use?

ThingSpeak, Blynk, Firebase or a local Node-RED server are easy for students. AWS IoT Core suits teams that want to learn cloud services.

Can CSE students do IoT projects?

Yes. Pair simple wiring with strong software: a backend, database, dashboard and data analysis.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- Arduino documentation
- Espressif ESP32 documentation
- Raspberry Pi documentation
- MQTT specification and resources
- ThingSpeak documentation
- Node-RED

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550