



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

SCIENCE PROJECTS AND KITS

Mini and Major Project Guide

For diploma and BTech students: choose, plan, build, document and present

INSIDE THIS BOOK

- Choosing the right project
- Literature survey and problem statement
- Block diagram, plan and budget
- Selecting components
- Simulation before hardware
- ...and 6 more chapters, practice questions and a glossary

Intermediate

Diploma, polytechnic and BTech students

engineergurukul.com

+91 97116 99511

Mini and Major Project Guide

A practical, start-to-finish guide to your mini or major project. You will learn to choose a project that fits your branch and skills, survey what others have done, plan with a block diagram and budget, simulate, build and test, write a clear report, and give a confident demo and viva. Thirty project ideas are included.

WHO IT IS FOR	Diploma, polytechnic and BTech students
LEVEL	Intermediate
SUBJECT AREA	Science projects and kits
WHAT IS INSIDE	11 chapters, practice questions with answers, and a glossary

HOW TO USE IT

Read one chapter at a time and try the examples as you go: on a computer, at your workbench or in your notebook. The boxes give tips, safety points and short activities. When you finish, test yourself with the practice questions and check your answers at the end. Words you may not know are explained in the glossary.



Gurukul Technicals Institute

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

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

engineergurukul@yahoo.co.in · engineergurukul.com

Free ebook. You may read, print and share this ebook for learning. It may not be sold or changed.

© 2026 Gurukul Technicals Institute. First edition, October 2026. The information was checked when it was written; software, prices and rules change, so for anything official always check the latest source.

What you will find inside

1	Choosing the right project	4
2	Literature survey and problem statement	5
3	Block diagram, plan and budget	7
4	Selecting components	9
5	Simulation before hardware	11
6	From breadboard to perfboard and PCB	13
7	Code that works and that you can explain	15
8	Testing and results	17
9	Writing the project report	19
10	Presentation, demo and viva	21
11	Thirty project ideas, and building your own	22
	Practice questions	25
	Answers	26
	Glossary	27

Choosing the right project

Your project is the one part of your course where you decide what to build. Choose well and you will learn more in a few months than in a year of lectures. Choose badly and you will spend those months fighting parts you do not understand.

Mini project and major project

	Mini project	Major project
When	Usually in a middle semester	Usually in the final year
Size	One clear function	A complete system with several parts
Team	Often one to three students	Often three to five students
Report	Short report and demo	Full report, presentation and viva

Your university or board sets the exact rules, so read its project guidelines first.

Match the project to your branch

The examiner will ask how your project relates to your subjects. Pick something that uses them.

- **Electronics and communication:** sensors, signal processing, wireless links, embedded systems.
- **Electrical:** motor control, battery systems, solar charging, energy measurement, protection circuits.
- **Computer science and IT:** apps, databases, IoT dashboards, data analysis, machine learning on sensor data.
- **Mechanical:** mechanisms, robotics, test rigs with sensors, automation of a simple process.
- **Instrumentation:** measuring something accurately, calibration, data logging.

Match the project to your skills

Be honest about what you can do now, then add one or two new skills. If you have never written code, do not start with a project that needs a mobile app, a cloud server and machine learning.

EXAMPLE

Rahul, a diploma electrical student in Nagpur, wanted to build a "smart grid". His guide asked what he could finish in ten weeks. He chose a solar battery charge monitor that measures voltage and current and logs them. It used his subjects, he understood every part, and he answered every viva question.

Five checks before you decide

1. Can you explain the idea in two sentences?
2. Can you buy the parts locally or online within your budget?
3. Does it work on low voltage, without touching mains wiring?
4. Is there a clear way to test whether it works?
5. Does your guide (the teacher supervising you) approve it?

SAFETY

Keep student projects on batteries or low-voltage DC adapters, up to 12 V. Represent mains appliances with LEDs, DC fans or 12 V lamps. If a project truly needs mains power, it must be wired and checked by a qualified electrician, not by students.

REMEMBER

The best project is not the most impressive one on paper. It is the one you can build, test and explain completely.

CHAPTER 2

Literature survey and problem statement

Before you build, find out what already exists. This is called a literature survey: reading earlier work on your topic and noting what it did well and what it missed.

Where to look

- **Research papers:** search IEEE Xplore and Google Scholar using your topic's key words. Your college library may give access.
- **Datasheets:** the official document for each component, showing its pins, voltages, limits and timing.
- **Application notes:** guides from chip makers showing how to use their parts.
- **Textbooks and earlier project reports** in your college library.

How to read a paper quickly

1. Read the title and abstract (the short summary at the top).
2. Look at the block diagram and the results figures.
3. Read the conclusion.
4. Only then, if it is useful, read the full method.

Keep a survey table

No.	Source	What they did	Result	Gap or limitation
1	Paper on soil-moisture irrigation	Fixed threshold, relay pump	Saved water in trials	Pump chattered near threshold
2	Project report, earlier batch	Timer-based watering	Simple, cheap	Ignored actual soil condition
3	Sensor datasheet	Sensor output vs moisture	Output falls as soil gets wetter	Needs calibration for each soil

The "gap" column is gold. Your project should fill one of those gaps.

TIP

Write each source in your reference list the day you read it. Hunting for a paper again at report time wastes hours.

The problem statement

A problem statement says, in three or four sentences, what problem you are solving, for whom, and how you will know you succeeded.

EXAMPLE

"Small farmers and home gardeners in dry areas often over-water or under-water plants because watering is done by guesswork or by timer. Existing low-cost systems switch the pump on and off rapidly near a single threshold. This project will build a battery-powered watering controller that uses two moisture levels to avoid rapid switching, and logs readings so water use can be measured. Success means the soil stays within the set range for seven days with no pump chatter."

Objectives

Turn the problem statement into three to five objectives that start with a verb: design, build, test, compare, measure. Each objective should be checkable at the end.

TRY THIS

Write your problem statement and show it to someone outside engineering. If they cannot tell you back what you are building and why, rewrite it.

CHAPTER 3

Block diagram, plan and budget

The block diagram

A block diagram shows the main parts of the system as boxes, joined by arrows that show the flow of power, signals or data. It is the first drawing in your report and the one the examiner looks at longest.

Most embedded projects fit this pattern:

1. **Power supply:** battery or adapter, regulator.
2. **Inputs:** sensors, switches, keypad, RFID reader.
3. **Controller:** Arduino, ESP8266, ESP32 or another microcontroller.
4. **Outputs:** motors, pumps, buzzers, displays, LEDs.
5. **Communication (optional):** Wi-Fi, Bluetooth, GSM, a USB link to a computer.

EXAMPLE

For the watering project the blocks are: 12 V battery → regulator and controller board; soil moisture sensor → controller; controller → MOSFET driver → 12 V pump; controller → USB serial log to a laptop.

The flowchart

A flowchart shows the steps your program follows, in order, with diamond-shaped boxes for decisions. Draw it before you write code. For the watering project, the flowchart reads like this:

1. Start: set up the pins and switch the pump off.
2. Have two seconds passed since the last reading? If not, keep checking.
3. Read the soil sensor.
4. Is the pump off and the reading above the dry level? If yes, switch the pump on.
5. Is the pump on and the reading below the wet level? If yes, switch the pump off.
6. Send the time, reading and pump state to the computer.
7. Go back to step 2.

If you cannot draw the flowchart, you are not ready to write the code.

The plan

Break the work into tasks and give each a week. A simple timetable is enough.

Week	Task
1–2	Literature survey, problem statement, block diagram
3	Component selection and ordering
4	Simulation of the circuit and code
5–6	Breadboard prototype, block by block
7	Full system test, fixes
8	Perfboard or PCB build
9	Final testing and results
10	Report, presentation, viva practice

Add a week of spare time. Parts arrive late and modules arrive dead more often than you expect.

Team roles

In a team, give each member a lead area, such as hardware, software or documentation. But every member must understand the whole project, because the examiner can ask anyone anything.

The budget

Item	Qty	Example price (!)
Arduino Uno compatible board	1	450
Soil moisture sensor	2	160
MOSFET driver module	1	60
Small 12 V pump and tubing	1	250
12 V battery pack and holder	1	400
Breadboard, jumper wires	1 set	200
Perfboard, connectors, solder	1 set	150
Total		1,670

These are example prices only. Prices change, so check current prices before you plan. Buy one spare of any cheap sensor you depend on.

REMEMBER

A plan you revise every week is far more useful than a perfect plan you never look at again.

CHAPTER 4

Selecting components

Choosing parts is engineering, not shopping. For every component, write down why you chose it.

Questions to ask about each part

- **Voltage:** what supply does it need, and what voltage do its signal pins use?
- **Current:** how much current does it draw, especially at start-up?
- **Interface:** does it give an analog voltage, a digital on/off, or use a bus such as I2C, SPI or UART (serial)?
- **Accuracy and range:** can it measure what you need, as precisely as you need?
- **Availability:** can you buy a replacement quickly if it fails?
- **Support:** is there a clear datasheet, and a well-known library for your controller?

Logic levels

The Arduino Uno works with 5 V signals. The ESP8266 and ESP32 work with 3.3 V signals. Feeding a 5 V signal into a 3.3 V pin can damage it. Use a level shifter or a voltage divider (two resistors that scale a voltage down) where the two meet.

EXAMPLE

An HC-SR04 ultrasonic sensor's echo pin gives 5 V pulses. Pooja in Indore connected it to an ESP8266 through a 1 k Ω resistor, with a 2 k Ω resistor from the ESP8266 pin to ground. The pin sees $5 \times 2 \div 3$, which is about 3.3 V.

The power budget

Add up the current of every part and choose a supply that can give more, with margin.

Part	Typical current
Controller board	Tens of milliamps
Each LED	About 10–20 mA
Small servo while moving	Can be a few hundred mA
GSM module while transmitting	Short bursts that can reach about 2 A
Small DC pump or motor	Hundreds of mA, more at start-up

Motors and servos should get their own supply, with its ground joined to the controller's ground. Powering them from the controller's 5 V pin causes resets and strange faults.

TIP

Read the "absolute maximum ratings" table in every datasheet. Those are limits you must never reach, not values to run at.

Common choices

Need	Common part	Note
Temperature	DS18B20, LM35	DS18B20 is digital and needs only one data wire, plus a pull-up resistor
Temperature and humidity	DHT22	Slow to read; read every two seconds or more
Distance	HC-SR04 ultrasonic	Struggles with soft or angled surfaces
Current	INA219 module	Measures voltage and current over I2C
Switching a DC load	Logic-level MOSFET module	Switches fully on from a controller pin; silent and fast; DC loads only
RFID	RC522 module	Runs on 3.3 V, uses SPI
SMS	GSM module	Many cheap modules work only on 2G networks, so check that your SIM's network offers 2G in your area, or use a 4G module

DID YOU KNOW

Many cheap relay modules switch ON when their input pin is LOW, not HIGH. They are called "active-low". Test your module before writing the rest of your code.

CHAPTER 5

Simulation before hardware

Simulation means testing your circuit and code on a computer before you build it. It catches wiring mistakes and logic errors without burning a single part.

Commonly used tools

- **Tinkercad Circuits:** free and browser-based. You can wire an Arduino Uno with sensors, LEDs, motors and a multimeter, write code, and run it. It is excellent for checking logic and teaching.
- **Proteus:** a widely used commercial package in colleges for drawing schematics, simulating circuits including microcontrollers, and designing PCBs. Your college lab may have licences.
- Other tools you may meet are LTspice for analog circuits and KiCad for schematics and PCB design.

What to simulate

1. The logic of your code with simple inputs, such as a potentiometer in place of a sensor.
2. Voltage dividers, transistor switches and LED resistors, using a simulated multimeter.
3. Edge cases: what happens when the sensor reads zero, or the maximum?

What simulation cannot tell you

- Whether your real sensor behaves like the ideal one.
- Electrical noise, loose wires and poor solder joints.
- Whether your battery can supply the current your motors need.

REMEMBER

Simulation proves your idea and your code logic. Only real hardware proves your project.

A quick worked check

EXAMPLE

Sneha in Coimbatore needed a resistor for a red LED on a 5 V pin. She took the LED's forward voltage as about 2 V and chose a current of 15 mA. $R = (5 - 2) \div 0.015 = 200 \, \Omega$. The nearest common value above that is 220 Ω , which gives $(5 - 2) \div 220 =$ about 13.6 mA. Her simulated multimeter showed nearly the same current, so she knew her wiring and her sum were both right.

Record it

Take screenshots of your simulation with the inputs and outputs clearly visible. They go into the methodology chapter of your report and show the examiner that you worked step by step.

TRY THIS

In Tinkercad, build your circuit with a potentiometer standing in for the sensor. Turn it slowly from end to end and watch whether your output switches exactly where your code says it should.

TIP

Name your simulation files with a date and a short note, such as "watering_v3_hysteresis". When something breaks, you can go back to the last version that worked.

CHAPTER 6

From breadboard to perfboard and PCB

Stage 1: breadboard, block by block

A breadboard lets you push components and wires in without soldering. Build one block at a time, and test it before adding the next.

1. Power: check the regulator output with a multimeter before connecting anything else.
2. One sensor: print its readings to the serial monitor.
3. One output: switch it on and off from a simple test program.
4. Join the blocks, then load the full program.

TIP

Use colours consistently: red for positive supply, black for ground, other colours for signals. Photograph the breadboard from above once it works, so you can rebuild it if wires get pulled out.

Stage 2: perfboard or stripboard

A perfboard has a grid of holes with copper pads. You solder parts in and join them with wire. A stripboard has long copper strips that you cut where needed. Both make a permanent circuit that will survive a trip to college.

- Plan the layout on squared paper first.
- Use headers (small sockets) for modules and chips, so they can be replaced.
- Use screw terminals for battery and motor wires.
- Check every joint with the multimeter's continuity mode.

Stage 3: a custom PCB (optional)

A PCB (printed circuit board) has copper tracks designed on a computer. You draw the schematic, lay out the board in software such as KiCad, Proteus or EasyEDA, and send the manufacturing files, called Gerber files, to a PCB maker. It looks professional and is reliable, but it takes time to arrive, so order it early.

SAFETY

A soldering iron tip is hot enough to burn skin badly. Always rest it in its stand, work in a ventilated room, wear safety glasses, and wash your hands after handling solder. Younger students must solder only with an adult present. Unplug the iron when you leave the bench, even for a minute.

Good assembly habits

- Keep wires short and neat. Tie loose bunches with cable ties.
- Fix the board inside a box with standoffs (small spacers), not loose on the table.
- Label connectors: "12V IN", "PUMP", "SENSOR".
- Add a fuse or a polyfuse on the battery line, and a switch.
- Fit a flyback diode, such as a 1N4007, across every motor, pump, relay coil and solenoid, with its stripe towards the positive side. It absorbs the voltage spike when the load switches off, which would otherwise damage the transistor or MOSFET.

DID YOU KNOW

Most faults in student projects are not in the code. They are loose jumper wires, a missing common ground, a weak battery, or a cold solder joint (a dull, lumpy joint that does not make good contact).

Code that works and that you can explain

Your code will be read by your guide and examiner. Write it so that you, and they, can understand it months later.

Good practices

1. **Name things clearly.** `PUMP_PIN`, not `p`. `DRY_LEVEL`, not `600` buried in the middle of the code.
2. **Put settings at the top** as named constants, so changing a pin or a threshold means changing one line.
3. **Comment why, not what.** "Two levels stop the pump flickering" helps. "Set pumpOn to true" does not.
4. ****Avoid long `delay()` calls**** in the main loop. While `delay()` runs, the controller cannot read buttons or sensors. Use `millis()` to check whether enough time has passed instead.
5. **Print what is happening** to the serial monitor while testing.
6. **Save versions.** Use Git, or at least save dated copies, so you can return to a version that worked.

A worked example

This Arduino sketch controls the watering project. It reads the sensor every two seconds without blocking, uses two levels (called hysteresis) so the pump does not switch on and off rapidly, and prints one comma-separated line per reading.

```
// Automatic plant watering
// Soil moisture sensor on A0, pump driver (MOSFET module) on pin 7.

const int SOIL_PIN = A0;      // analog output of the soil sensor
const int PUMP_PIN = 7;       // signal pin of the pump driver
const int DRY_LEVEL = 600;    // above this the soil is dry (calibrate your
                               // sensor)
const int WET_LEVEL = 450;    // below this the soil is wet enough
const unsigned long READ_EVERY_MS = 2000; // read the sensor every 2
seconds

unsigned long lastReadMs = 0;
bool pumpOn = false;

void setup() {
    pinMode(PUMP_PIN, OUTPUT);
    digitalWrite(PUMP_PIN, LOW);          // pump off at start
    Serial.begin(9600);
    Serial.println("time_ms,reading,pump");
}

void loop() {
    unsigned long now = millis();
    if (now - lastReadMs >= READ_EVERY_MS) {
        lastReadMs = now;
        int reading = analogRead(SOIL_PIN);

        // Two levels (hysteresis) stop the pump flickering on and off
        if (!pumpOn && reading > DRY_LEVEL) {
            pumpOn = true;
        } else if (pumpOn && reading < WET_LEVEL) {
            pumpOn = false;
        }
        digitalWrite(PUMP_PIN, pumpOn ? HIGH : LOW);

        // One line of comma-separated values per reading, for the results
chapter
        Serial.print(now);
        Serial.print(',');
        Serial.print(reading);
        Serial.print(',');
        Serial.println(pumpOn ? 1 : 0);
    }
}
```

With many common soil sensor modules, the reading rises as the soil gets drier. Check yours: note the reading in dry soil and in freshly watered soil, then set `DRY_LEVEL` and `WET_LEVEL` between them.

TIP

Compile (verify) your sketch after every few lines you add. Fixing one error at a time is far easier than fixing twenty at once.

REMEMBER

Be ready to explain every line, including any library you used. "I copied it from a website" is not an answer an examiner accepts.

CHAPTER 8

Testing and results

Testing proves that your project meets its objectives. Results are the evidence, shown as numbers, tables and graphs.

Three levels of testing

- 1. Unit testing:** test each block alone. Does the sensor give sensible readings? Does the pump run when the pin goes high?
- 2. Integration testing:** join blocks and test them together.
- 3. System testing:** run the whole project in real conditions for a long time.

Write test cases

A test case states an input, the expected result and the actual result.

No.	Test	Expected	Actual	Pass?
1	Sensor in dry soil	Reading above 600	712	Yes
2	Sensor in wet soil	Reading below 450	388	Yes
3	Soil dries past 600	Pump turns on	Pump turned on	Yes
4	Reading near 600 for 5 minutes	No rapid switching	No switching	Yes

No.	Test	Expected	Actual	Pass?
5	Sensor wire pulled out	Safe state, pump off	Pump stayed on	No

A failed test is useful. Case 5 shows a real weakness, and fixing it improves the design. Report it honestly.

Measuring accuracy

When your project measures something, compare it with a trusted reference. This short Python program works out the error for an ultrasonic distance meter, using a measuring tape as the reference.

PYTHON

```
# Accuracy test of an ultrasonic distance meter.
# Each pair is (true distance in cm, distance shown by the project in cm).
readings = [
    (10, 10.4),
    (20, 20.3),
    (30, 29.5),
    (50, 50.9),
    (100, 101.8),
    (150, 152.6),
]

print(f"{'True':>6} {'Shown':>7} {'Error':>7} {'% error':>8}")
total_pct = 0.0
for true_cm, shown_cm in readings:
    error = shown_cm - true_cm
    pct = abs(error) / true_cm * 100
    total_pct += pct
    print(f"{true_cm:>6} {shown_cm:>7.1f} {error:>+7.1f} {pct:>8.2f}")

print(f"Mean percentage error: {total_pct / len(readings):.2f}%")
```

It prints each error and ends with "Mean percentage error: 2.08%". The 10 cm reading has the largest percentage error, which is worth discussing in your report.

TRY THIS

Replace the six pairs with your own readings and run the program with python3. If one reading has a much bigger error than the rest, measure it again before you trust it.

What to measure

- Accuracy against a reference.
- Response time: how long from a change to the system reacting.
- Repeatability: the same input, many times.
- Battery life or power use.
- Range, for wireless projects.

TIP

Run the system for at least a few days before the demo. Faults that appear after hours, such as a battery sagging or a sensor drifting, rarely show up in a five-minute test.

CHAPTER 9

Writing the project report

Your report is the permanent record of your project. Follow your university's format exactly: margins, font, line spacing and chapter names. The structure below is the one most universities use.

Chapter	What it contains
Abstract	One paragraph: problem, method, main result. Written last.
Introduction	Background, problem statement, objectives, scope
Literature review	What others have done, and the gap you address
Methodology	Block diagram, circuit diagram, flowchart, component choices, simulation
Implementation	How you built it: hardware assembly, code structure, photos
Results and discussion	Test cases, tables, graphs, and what they mean
Conclusion	What you achieved against each objective
Future scope	Sensible next steps
References	Every source you used, in one consistent style

Front pages usually include a title page, certificate, declaration, acknowledgement, contents, list of figures and list of tables. Appendices can hold full code and datasheet extracts.

The abstract

The abstract is the first thing every reader sees, and often the only part they read. In about 150 to 250 words, unless your university says otherwise, it states the problem, what you built, how you tested it and the main result, with a number.

EXAMPLE

"Watering plants by guesswork or by timer can waste water and harm plants. This project presents a battery-powered watering controller based on an Arduino Uno and a soil moisture sensor. Two switching levels prevent the pump from switching rapidly near a single threshold. Readings are logged to a computer for analysis. In a seven-day trial, the soil stayed within the set range and the pump switched 14 times, with no rapid switching. Future work includes a solar supply and a phone alert when the tank is low."

Writing tips

- Write in clear, simple sentences, in the past tense for what you did.
- Number every figure and table, give it a caption, and refer to it in the text: "as shown in Figure 3.2".
- Explain your own circuit and code. Do not paste pages of general theory.
- Show real results, including failures and how you fixed them.

EXAMPLE

A reference in IEEE style, which many engineering colleges use, follows the pattern: [1] A. Author, "Title of paper," Name of Journal, vol. 5, no. 2, pp. 10–15, year. Number references in the order you first cite them.

REMEMBER

Many colleges check reports with plagiarism-detection software. Write in your own words and cite every source. Copied text can lead to the report being rejected.

Presentation, demo and viva

The presentation

Keep the slides simple. Ten to fifteen slides is usually enough for a ten-minute talk.

1. Title, team and guide.
2. Problem statement and objectives.
3. Literature survey summary and the gap.
4. Block diagram.
5. Circuit diagram and flowchart.
6. Hardware photos.
7. Results: one table and one graph.
8. Conclusion and future scope.

Use large text, few words per slide, and clear diagrams. Speak to the examiners, not to the screen.

The demo

TIP

Prepare a demo kit: fresh batteries, spare jumper wires, a spare sensor, a USB cable, a charged laptop with the code ready to upload, and a short video of the project working in case something fails on the day.

- Test the demo in the exam room, if allowed, before your turn.
- Show the result clearly: start with dry soil, show the pump switching on, and show the log on screen.
- If something fails, stay calm. Explain what should happen, check the obvious causes, and show your video.

Common viva questions

Question	What to cover
Why did you choose this project?	The problem and the gap you found

Question	What to cover
Explain your block diagram.	Each block, and the signal flow
Why this sensor and not another?	Range, accuracy, interface, cost, availability
What is the role of this resistor or transistor?	Its exact job in your circuit
How does your code decide when to switch?	Thresholds, hysteresis, timing
What are the limitations?	Honest, specific weaknesses
How would you improve it?	Practical next steps

REMEMBER

Every member should be able to explain every block. Examiners often ask the quietest student about the part they did not build.

SAFETY

Carry the project on battery power only during the demo, with all wiring covered and the switch off until it is your turn.

CHAPTER 11

Thirty project ideas, and building your own

Each idea below is a starting point. Adapt it, narrow it and find your own gap. Keep every one on batteries or low-voltage DC.

Arduino

- 1. Line-following robot:** two infrared sensors steer a two-wheeled robot along a black line.
- 2. Obstacle-avoiding robot:** an ultrasonic sensor detects objects and the robot turns away.
- 3. Automatic plant watering:** a soil sensor switches a small DC pump, with hysteresis.
- 4. Digital distance meter:** an ultrasonic sensor shows distance on an LCD.
- 5. Smart dustbin:** a servo opens the lid when a hand comes near.

- 6. Traffic light with pedestrian button:** timed lights that respond to a crossing request.

IoT (ESP8266 or ESP32)

- 1. Weather station:** temperature, humidity and pressure uploaded to an online dashboard.
- 2. Water tank level monitor:** tank level shown on a phone, with a low-level alert.
- 3. DC energy monitor:** current and voltage of a 12 V load logged online.
- 4. Smart parking indicator:** sensors show which parking slots are free on a web page.
- 5. Air quality monitor:** a gas sensor reports air quality trends on a local web page.
- 6. Phone-controlled low-voltage loads:** LEDs and a DC fan switched from a phone over Wi-Fi.

Solar

- 1. Single-axis solar tracker:** two light sensors and a servo keep a small panel facing the Sun; compare its output with a fixed panel.
- 2. Solar charge monitor:** panel and battery voltage and current measured and logged.
- 3. Solar mobile charger:** a small panel, battery and step-down module give a steady 5 V USB output.
- 4. Automatic solar street light:** a light sensor switches an LED lamp on at dusk from a battery.

GSM

- 1. Intrusion alert:** a door sensor sends an SMS when the door opens.
- 2. Accident alert:** a tilt or impact sensor sends the GPS location by SMS.
- 3. SMS pump control:** a DC pump model switched on and off by text message.

RFID

- 1. RFID attendance system:** card taps logged with time to an SD card or a spreadsheet.
- 2. RFID door lock:** an authorised card opens a 12 V solenoid lock.
- 3. RFID toll gate model:** a card is checked and a servo barrier lifts.

Embedded

1. **Digital thermometer with alarm:** a temperature sensor and a buzzer at a set limit.
2. **Keypad door lock:** a password entered on a keypad unlocks a servo latch.
3. **Pulse rate display:** a pulse sensor shows beats per minute, for learning only, not medical use.
4. **Bicycle speedometer:** a Hall sensor, which detects a magnet, counts turns of a wheel magnet to give speed and distance.

Electrical (low voltage)

1. **DC motor speed control:** PWM (pulse width modulation) through a MOSFET controls the speed; an infrared sensor measures it.
2. **12 V battery monitor:** a voltage divider and display show the charge state, with a low-battery warning.
3. **Wireless power transfer demo:** two coils light an LED across a small gap.
4. **Overload protection for DC loads:** a current sensor cuts the load if the current goes too high.

DID YOU KNOW

Examiners have seen these titles many times. What makes yours stand out is a clear gap, careful testing and honest results, not a new title.

Build it yourself, and understand it

Buying a ready-made project, or paying someone to build it, is dishonest, and it rarely survives the viva. Ethical project work means:

- You build the hardware and write or fully understand the code yourself.
- You credit every library, circuit and idea you borrowed, in your references.
- You report your real results, including failures.
- You protect people's data. An RFID attendance log holds personal information, so keep it private and delete test data when the project ends.

REMEMBER

Help from your guide, seniors, forums and datasheets is normal and welcome. The line is crossed when the work is no longer yours.

Practice questions

1. Give two differences between a mini project and a major project.
2. List the five checks to make before choosing a project.
3. What is a literature survey?
4. Why is the "gap or limitation" column in a survey table important?
5. What should a problem statement contain?
6. Name the five common blocks in an embedded project's block diagram.
7. Why should you add spare time to your project plan?
8. Why must a 5 V signal not go directly into an ESP8266 pin?
9. A 1 k Ω and 2 k Ω divider is fed 5 V, with the output taken across the 2 k Ω resistor. What is the output voltage?
10. Why should motors not be powered from the controller board's 5 V pin?
11. What does "active-low" mean for a relay module?
12. Name two tools commonly used for circuit simulation.
13. Give two things that simulation cannot tell you.
14. Why should you build and test a breadboard circuit block by block?
15. What is a Gerber file used for?
16. Why should `delay()` be avoided in the main loop?
17. What is hysteresis, and why does the watering code use it?
18. What is the difference between unit testing and system testing?
19. A meter shows 51.0 cm for a true distance of 50.0 cm. What is the percentage error?
20. Which report chapter is written last?
21. What should a demo kit include?
22. How should you answer a viva question you cannot answer?

23. Give two rules of ethical project work.

ANSWER KEY

Answers

1. Any two: a mini project is smaller, usually done in a middle semester, often by fewer students, with a shorter report; a major project is a complete system, usually in the final year, with a full report, presentation and viva.
2. Can you explain it in two sentences, can you buy the parts within budget, does it run on low voltage, is there a clear way to test it, and does your guide approve it.
3. A study of earlier work on your topic, noting what each source did, what it found and what it missed.
4. It shows what earlier work did not solve, and your project should address one of those gaps.
5. The problem, who it affects, how you will solve it, and how you will know you succeeded.
6. Power supply, inputs, controller, outputs and communication.
7. Parts arrive late and modules can be faulty, so spare time stops these delays from wrecking the deadline.
8. The ESP8266 works with 3.3 V signals, and a 5 V signal can damage its pin.
9. $5 \times 2 \div (1 + 2) = \text{about } 3.3 \text{ V}$.
10. They draw large and sudden currents, which can make the board reset or behave strangely.
11. The module switches on when its input pin is LOW rather than HIGH.
12. Tinkercad Circuits and Proteus.
13. Any two: how a real sensor behaves, electrical noise, loose wires or bad joints, and whether the battery can supply enough current.
14. So that each fault is found in a small part of the circuit before the next part is added, which makes faults much easier to locate.
15. It holds the manufacturing data a PCB maker needs to make your board.
16. While `delay()` runs, the controller cannot read sensors or buttons, so the system stops responding.
17. Hysteresis means using two switching levels instead of one. It stops the pump switching on and off rapidly when the reading is close to a single threshold.
18. Unit testing checks each block alone. System testing checks the whole project working together in real conditions.
19. Error = 1.0 cm, so percentage error = $1.0 \div 50.0 \times 100 = 2\%$.
20. The abstract.

21. Fresh batteries, spare wires and a spare sensor, a USB cable, a charged laptop with the code ready, and a video of the project working.

22. Say honestly that you are not sure, then explain how you would find out.

23. Any two: build it and understand it yourself, credit all borrowed work, report real results, and protect personal data.

WORD LIST

Glossary

Mini project

a smaller project with one clear function, usually done in a middle semester.

Major project

a complete system built in the final year, with a full report and viva.

Guide

the teacher who supervises and approves your project.

Literature survey

a study of earlier work on your topic.

Datasheet

the official document giving a component's pins, ratings and behaviour.

Problem statement

a short paragraph stating the problem, who it affects and how success will be measured.

Block diagram

a drawing of a system's main parts as boxes joined by arrows.

Microcontroller

a small computer on one chip that reads inputs and controls outputs.

Logic level

the voltage a device uses for its high and low signals, such as 5 V or 3.3 V.

Voltage divider

two resistors that produce a smaller, fixed fraction of a voltage.

Power budget

the total current all parts need, used to choose a power supply.

Active-low

switching on when a signal is LOW rather than HIGH.

Simulation

testing a circuit or code on a computer before building it.

Breadboard

a board for building circuits by pushing parts in, without soldering.

Perfboard

a board with a grid of holes and copper pads for soldered circuits.

PCB

a printed circuit board, with copper tracks designed on a computer.

Gerber file

the manufacturing file a PCB maker uses to make a board.

Hysteresis

using two switching levels so a system does not switch rapidly near one threshold.

PWM

pulse width modulation, switching power on and off quickly to control the average power, for example for motor speed.

Test case

a planned test with an input, an expected result and the actual result.

Percentage error

the size of an error as a percentage of the true value.

Abstract

a one-paragraph summary at the start of a report.

Future scope

sensible ways the project could be extended.

Viva

an oral examination about your project.



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.