SMART HOME APPLIANCES USING ARDUINO AND RELAY

For Energy-Efficient Control

Project Presentation • 20 Slides • Expanded Notes and Example Code



Prepared by: Aastha Pravin Kshirsagar • Vaibhavi Mahesh Nanajkar • Sayali Ganesh Ladhane • Samiksha Siddheshwar Kute

JSPM’s Bhivrabai Sawant Polytechnic, Wagholi, Pune, Maharashtra, India

02 Introduction

Smart home automation connects household devices with controllers, sensors and communication modules.

- Home automation is growing with the Internet of Things (IoT), microcontrollers and affordable sensors.
- Arduino Uno can read inputs and control electrical loads through a relay driver.
- This project demonstrates a low-cost way to switch lights, fans and other suitable appliances.
- The system can be expanded with Wi-Fi, mobile apps and energy monitoring.

Key point: Project focus: convenient control, improved energy use and safer operation when correctly designed.

03 Project Objectives

The goal is to build a simple, understandable and expandable home-control prototype.

- Switch appliances automatically or manually using input buttons and sensors.
- Use an Arduino Uno to process input signals and operate a relay module.
- Demonstrate control of a low-voltage lamp or fan in a supervised prototype.
- Create a design that can later support Wi-Fi, schedules and usage monitoring.
- Apply correct electrical isolation and safe wiring practices.

Key point: Expected outcome: a working demonstration of input → controller → relay → appliance.

04 Smart Home Overview

A smart home uses connected devices to monitor and control home functions.

- Remote access: operate supported devices through a phone or network interface.
- Automation: trigger actions from buttons, schedules or sensor readings.
- Energy efficiency: switch off unused loads and avoid unnecessary operation.
- Comfort: simplify everyday tasks and provide repeatable control.
- Security: add motion, door or window sensors where appropriate.

Key point: Smart does not automatically mean energy-saving; benefits depend on settings and user behaviour.

05

System Block Diagram

The system is arranged as a sequence of input, processing, switching and output blocks.

- Inputs: push buttons and optional temperature or motion sensors.
- Controller: Arduino Uno reads the inputs and applies programmed logic.
- Driver: relay module provides electrically controlled switching contacts.
- Outputs: light, fan or another compatible load.
- Power: regulated 5 V for the controller/relay logic; load power is kept separate.

Key point: Signal flow: Input / Sensor → Arduino Uno → Relay Module → Appliance.

06

Arduino Uno

Arduino Uno is the central controller for reading inputs and making switching decisions.

- Common classic Uno board uses an ATmega328P microcontroller.
- Operating logic voltage is 5 V; board input supply is typically 7–12 V via the recommended input.
- It provides 14 digital I/O pins, with 6 commonly usable as PWM outputs, and 6 analog inputs.
- The classic ATmega328P version has 32 KB flash memory (some space used by the bootloader).
- Programs are written and uploaded using the Arduino IDE over USB.

Key point: Pin count and specifications can vary by board revision; check the exact board documentation.

07 Relay Module

A relay is an electrically operated switch that lets a low-voltage control signal switch a separate circuit.

- A 4-channel module can control four independent loads when wired and rated correctly.
- Many modules accept 5 V logic/power; some boards use active-low inputs.
- Relay contacts are commonly labelled COM, NO and NC.
- Opto-isolated modules may improve separation when correctly wired and powered.
- Contact voltage/current ratings and load type must be checked before use.

Key point: Arduino pins must not power an appliance directly. Use a suitable driver/module and rated contacts.

08 ESP8266 Wi-Fi Module (Optional)

An ESP8266 can add network connectivity to a suitable version of the project.

- Supports 2.4 GHz Wi-Fi and is widely used for IoT prototypes.
- Its GPIO logic is 3.3 V; do not apply 5 V directly to its GPIO pins.
- Can exchange commands with a controller or run compatible firmware itself.
- A phone dashboard or local web interface can provide remote switching.
- Network access should be protected with strong credentials and secure configuration.

Key point: Wi-Fi is optional; the basic button-controlled project can work without an internet connection.

09

Hardware Components

The project uses a small set of widely available electronics components.

- Arduino Uno and USB cable for programming.
- Relay module with enough channels for the planned outputs.
- Push buttons and optional temperature/motion sensor.
- LED lamp or low-voltage DC fan for a safe demonstration.
- Breadboard, jumper wires and a suitable regulated supply.
- Optional ESP8266/ESP32 module for connectivity.

Key point: Use low-voltage loads for classroom testing. Mains appliances require qualified supervision and enclosure.

10 Circuit Design

The control circuit connects input devices to Arduino pins and relay inputs to output pins.

- Example: connect a push button between D2 and GND and configure D2 as INPUT_PULLUP.
- Connect relay IN1 to D8, relay VCC to its specified supply and GND to the logic ground as required by its design.
- Use a relay module compatible with the controller voltage and current.
- Connect the load only to correctly rated relay contacts and follow the module's terminal markings.
- Keep high-voltage wiring physically separated from low-voltage control wiring.

Key point: Do not assemble or test exposed mains wiring. Use a low-voltage lamp/fan prototype for demonstrations.

11 Working Principle

The Arduino continuously reads the input, decides the desired state and changes the relay output.

- 1. User presses a button or a sensor crosses a threshold.
- 2. Arduino reads the digital/analog input.
- 3. The program checks the condition and updates the output pin.
- 4. The relay switches its contact state (ON/OFF).
- 5. The appliance responds, provided its supply and wiring are correct.
- 6. The loop repeats so the state can be updated.

Key point: Add debounce logic for buttons and suitable thresholds/hysteresis for noisy sensor readings.

12 Arduino Program

Example below toggles a low-voltage demonstration output with a push button.

Connections and assumptions

- Button on D2 to GND; use INPUT_PULLUP, so pressed = LOW.
- Relay input on D8; this example assumes an active-HIGH relay input.
- If your relay module is active-LOW, reverse the HIGH/LOW output logic.
- Test with an LED or low-voltage demonstration load before connecting any appliance.

```
const int buttonPin = 2;
const int relayPin  = 8;

bool applianceOn = false;
bool lastButton = HIGH;
unsigned long lastChange = 0;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW); // assumes active-HIGH relay input
  Serial.begin(9600);
}

void loop() {
  bool reading = digitalRead(buttonPin);

  // Simple debounce
  if (reading != lastButton) {
    lastChange = millis();
    lastButton = reading;
  }

  static bool handled = false;
  if (millis() - lastChange > 40) {
    if (reading == LOW && !handled) {
      applianceOn = !applianceOn;
      digitalWrite(relayPin, applianceOn ? HIGH : LOW);
      Serial.println(applianceOn ? "ON" : "OFF");
      handled = true;
    }
    if (reading == HIGH) handled = false;
  }
```

Key point: Code is illustrative; verify the exact relay module's trigger logic and pin wiring.

13 Applications

The same input-controller-output idea can support several simple automation tasks.

- Lighting control: switch a demonstration lamp by button or sensor.
- Fan control: use a suitable sensor and properly rated switching hardware.
- Air-conditioner interface: use manufacturer-approved control methods; do not switch unsuitable loads.
- Home security: trigger a light or alert from a motion sensor.
- Water-level or garden systems: read sensors and operate compatible low-voltage devices.

Key point: Choose one application for a clear demonstration and explain its input, logic and output.

14 Energy-Saving Potential

Automation can reduce waste when it turns off devices that are not needed.

- Use occupancy sensing or timers to avoid lights remaining on in empty rooms.
- Schedule operation around actual use rather than leaving devices continuously enabled.
- Monitor run time to identify unnecessary operation.
- Use efficient appliances and avoid frequent switching where it is unsuitable.
- Estimate savings by comparing energy use before and after changes.

Key point: Energy (kWh) = Power (kW) × operating time (hours). Savings depend on actual usage.

15 Advantages

A small Arduino-based prototype is useful for learning and demonstrating automation.

- Low-cost and widely documented platform for educational projects.
- Simple to program and modify for different rules.
- Can improve convenience through automatic or remote switching.
- Modular design makes sensors and additional outputs easy to add.
- Can support energy-saving routines when designed and used properly.

Key point: The system's reliability and safety depend on component selection, wiring and testing.

16 Limitations

A basic prototype has limits that should be considered before real installation.

- Without Wi-Fi, remote control is unavailable; Wi-Fi may depend on network coverage.
- Basic setup does not measure energy unless a suitable meter/sensor is added.
- Relay contacts wear over time and have limits for current, voltage and load type.
- Network-connected devices can introduce cybersecurity and privacy risks.
- Mains installation requires proper enclosure, insulation, protection and qualified work.

Key point: Never exceed relay ratings or expose live mains terminals during a demonstration.

17 Testing and Evaluation

Test each function in stages before running the complete prototype.

- Inspect wiring and confirm the controller supply voltage.
- Test the button input using Serial Monitor or an LED.
- Test relay operation without a load and confirm its default state at startup.
- Use a low-voltage lamp or fan and check repeated ON/OFF cycles.
- Record response, unexpected switching, heating and stability.
- Repeat tests after adding each sensor or network feature.

Key point: Example acceptance criteria: reliable switching, stable operation and no unexpected reset during repeated trials.

18 Electrical Safety

Electrical safety is essential whenever a relay may control an appliance.

- Use an insulated enclosure, strain relief and properly rated wiring.
- Keep low-voltage electronics separated from hazardous voltage circuits.
- Use appropriate fuse/overcurrent protection and a correctly rated relay.
- Do not touch wiring while powered; disconnect supply before changing connections.
- Use opto-isolation only as designed; it does not make unsafe wiring safe.
- Mains wiring and testing must be performed by a qualified person.

Key point: For student demonstrations, use LEDs and low-voltage DC loads instead of exposed mains appliances.

19 Future Scope

The prototype can be expanded into a more capable smart-home platform.

- Add an ESP8266/ESP32 and a local web or mobile interface.
- Add temperature, motion, light-level or door sensors.
- Introduce schedules, status indicators and manual override.
- Add a properly designed energy-monitoring module.
- Explore voice-assistant integration through supported platforms.
- Use solar power or smart-grid data where appropriate.

Key point: Future features should preserve local manual control and safe fallback behaviour.

20 Conclusion

Arduino and relay modules provide a practical introduction to home automation.

- Inputs and sensors provide information about user actions or the environment.
- Arduino processes the input and controls a relay module.
- Suitable appliances can then be switched automatically or manually.
- The system is modular and can be extended with IoT connectivity.
- Safety, electrical ratings, testing and secure configuration remain essential.

Key point: A smart home is most useful when it is convenient, reliable, energy-aware and safe.

21 References and Q&A;

Suggested references for further study and verification.

- Arduino Uno official documentation: docs.arduino.cc/hardware/uno-rev3
- Arduino Language Reference: docs.arduino.cc/language-reference/
- Espressif ESP8266 documentation: docs.espressif.com/projects/esp8266/
- Relay module datasheet and manufacturer wiring instructions for the exact board used.
- Electrical safety guidance and local wiring regulations; consult a qualified electrician.

Key point: Thank you — Questions?