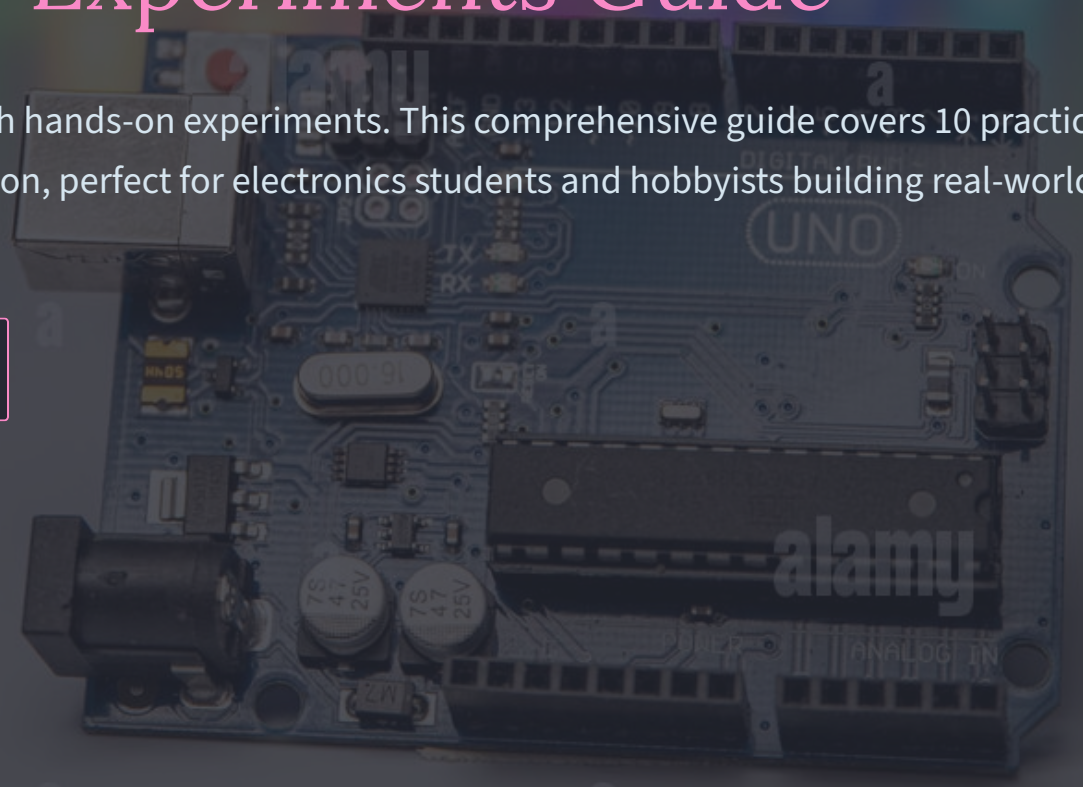
Arduino UNO Lab Experiments Guide

Master essential Arduino programming through hands-on experiments. This comprehensive guide covers 10 practical projects, from basic LED control to advanced motor automation, perfect for electronics students and hobbyists building real-world embedded systems.

Start Learning

Download Resources



Experiment 1: LED ON/OFF Control

This experiment aims to introduce fundamental digital output control using the Arduino UNO, a cornerstone skill for embedded systems development.

🎯 AIM

Write a program to turn ON and OFF an LED using Arduino UNO, demonstrating basic digital output control and understanding of `digitalWrite()` and `delay()` functions to create timed blinking patterns.

🔧 EQUIPMENT REQUIRED

- Arduino UNO R3 board (Official or compatible)
- LED (5mm, 2V-3V forward voltage, any common color like Red, Green, or Yellow)
- Resistor (220Ω, 1/4W, metal film preferred for precision)
- Breadboard (Half-size, 400 tie points)
- Jumper Wires (Male-to-male, assorted lengths)
- USB 2.0 A-B Cable (for Arduino programming and power)

📚 THEORY

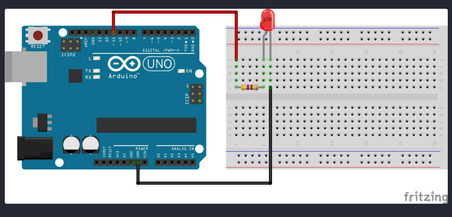
Arduino's digital pins can be configured as outputs to send either a HIGH (typically +5V) or LOW (0V/GND) signal. LEDs are Diodes that emit light when current flows in the correct direction (anode to cathode). To protect the LED from excessive current and prevent damage to the Arduino pin, a current-limiting resistor is always required in series with the LED. The `pinMode()` function is used to declare a pin as an `OUTPUT`. The `digitalWrite()` function sets the voltage level of the specified output pin to `HIGH` or `LOW`. The `delay()` function pauses the program for a specified number of milliseconds, allowing for visible timing intervals in the LED's blinking pattern.

🔌 CIRCUIT CONNECTION TABLE

The following table details the pin-to-pin connections required to build the circuit for this experiment:

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
Arduino UNO R3	Digital Pin 13	220Ω Resistor (one end)	Digital Output	Configured as OUTPUT, provides +5V (HIGH) or 0V (LOW).
Arduino UNO R3	GND Pin	LED Cathode	Ground Reference	0V reference point for the circuit.
LED (5mm)	Anode (Longer Leg)	220Ω Resistor (other end)	Positive Terminal	Requires current flow from Anode to Cathode. Forward Voltage: 2V-3V.
LED (5mm)	Cathode (Shorter Leg)	Arduino GND Pin	Negative Terminal	Identified by a flat edge on the LED casing.
Resistor (220Ω)	Any End	LED Anode	Current Limiting	Protects LED and Arduino pin. No polarity.
Resistor (220Ω)	Other End	Arduino Digital Pin 13	Current Limiting	Ensures current does not exceed LED's maximum rating.

Important Note: Ensure the LED's anode (longer leg) is connected towards the positive voltage (through the resistor to Pin 13) and the cathode (shorter leg) is connected to ground. Reversing the polarity will prevent the LED from lighting up and could potentially damage it or the Arduino.



⚡ CIRCUIT DIAGRAM EXPLANATION

The image illustrates the simple breadboard circuit. The LED's anode (positive terminal, indicated by the longer lead) is connected in series with a 220Ω resistor. The resistor limits the current flowing through the LED and protects it from burning out, as well as safeguarding the Arduino's output pin. The other end of the resistor is connected to Arduino's Digital Pin 13. The LED's cathode (negative terminal, indicated by the shorter lead or a flat edge on the LED casing) is connected directly to one of the Arduino's GND (Ground) pins. This completes the circuit, allowing current to flow from Pin 13, through the resistor and LED, and back to ground when Pin 13 is set to HIGH.

💻 PROGRAM CODE

```
void setup() {
  pinMode(13, OUTPUT); // Configure digital pin 13 as an output pin
}

void loop() {
  digitalWrite(13, HIGH); // Turn the LED ON (set voltage to +5V)
  delay(1000);            // Wait for 1000 milliseconds (1 second)
  digitalWrite(13, LOW);  // Turn the LED OFF (set voltage to 0V)
  delay(1000);            // Wait for another 1000 milliseconds (1 second)
}
```

01

WORKING/OPERATION

Upon uploading the code to the Arduino UNO, the `setup()` function executes once, configuring digital pin 13 as an output. The `loop()` function then runs continuously. In each iteration, it first sets pin 13 to HIGH, supplying 5V and turning the LED ON. After a 1-second delay, it sets pin 13 to LOW, cutting off the voltage and turning the LED OFF. Another 1-second delay follows before the loop repeats, creating a consistent blinking pattern.

02

OBSERVATION/RESULTS

The LED connected to digital pin 13 should visibly blink at regular 1-second intervals. When the LED is supposed to be ON, it glows brightly; when OFF, it is dark. If the LED does not light up or blinks erratically, meticulously check all physical connections, ensuring correct LED polarity and resistor placement, and verify that the code was uploaded successfully without errors. Also, confirm the resistor value is appropriate (220Ω-330Ω are common for 5V circuits).

03

CONCLUSION

This experiment successfully demonstrates the fundamental principles of digital output control on the Arduino UNO. By manipulating a single digital pin, we were able to switch an LED ON and OFF, illustrating the use of `pinMode()`, `digitalWrite()`, and `delay()` functions. This basic control mechanism forms the essential foundation for more complex projects involving switching devices, controlling relays, and generating timed signals in various embedded systems applications.

Experiment 2: LED Brightness Control with Potentiometer

🎯 AIM

Control LED brightness using a potentiometer, demonstrating analog input reading and PWM output for smooth dimming effects.

🔧 EQUIPMENT REQUIRED

- Arduino UNO board (Model: Arduino UNO R3)
- 10kΩ potentiometer (Linear taper, e.g., B10K)
- LED (Standard 5mm Red LED)
- 220Ω resistor (1/4W, +/- 5% tolerance)
- Breadboard (400-tie point or similar)
- Jumper wires (Male-to-male)

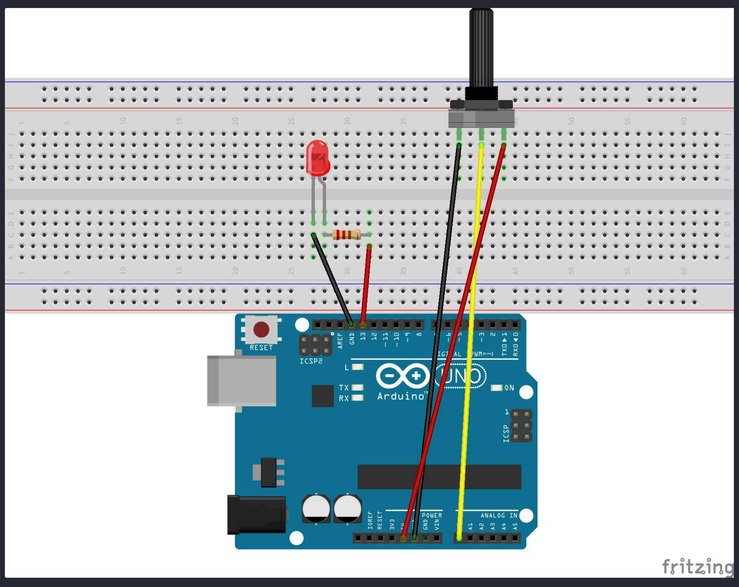
📚 THEORY

Potentiometers provide variable resistance (0-10kΩ), creating voltage divider circuits. As the knob is turned, the resistance changes, which in turn varies the voltage supplied to the Arduino's analog input pin. Arduino's analogRead() function captures this voltage, converting it into a 10-bit digital value ranging from 0 (0V) to 1023 (5V). The map() function then scales this 0-1023 range to the PWM (Pulse Width Modulation) range of 0-255. The analogWrite() function uses PWM to control the LED's brightness by rapidly switching it ON and OFF. A higher PWM value means the LED is ON for a longer duration within each cycle, making it appear brighter, while a lower value results in a dimmer appearance.

🔌 CIRCUIT CONNECTION TABLE

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
Potentiometer	Middle Pin (Wiper)	A0	Reads variable voltage from potentiometer	Analog Input (0-5V)
Potentiometer	Outer Pin 1	5V	Provides 5V supply to one end of the potentiometer	+5V DC Supply
Potentiometer	Outer Pin 2	GND	Connects the other end of the potentiometer to ground	Ground (0V) reference
LED	Anode (Longer Leg)	Digital Pin 9 (via 220Ω Resistor)	Controls LED brightness via PWM output	Positive terminal (+ve), PWM Digital Output (0-5V)
LED	Cathode (Shorter Leg)	GND	Completes the LED circuit to ground	Negative terminal (-ve), Ground (0V) reference

💡 CIRCUIT DIAGRAM EXPLANATION



The circuit diagram illustrates the connections for controlling LED brightness using a potentiometer. The potentiometer acts as a variable voltage divider. Its two outer pins are connected to the 5V and GND pins of the Arduino, establishing a voltage range across the potentiometer. The middle pin (wiper) is connected to Arduino's Analog Pin A0, which reads the variable voltage set by rotating the potentiometer's knob. The LED's anode (longer leg) is connected to Arduino's Digital Pin 9 (a PWM-capable pin) through a 220Ω current-limiting resistor to protect the LED from excessive current. The LED's cathode (shorter leg) is connected to GND. This setup allows the analog input from the potentiometer to directly control the brightness of the LED via PWM output.

💻 PROGRAM CODE

```
void setup() {
  pinMode(9, OUTPUT); // Set digital pin 9 as an output
}

void loop() {
  int val = analogRead(A0); // Read the analog value from potentiometer (0-1023)
  int brightness = map(val, 0, 1023, 0, 255); // Map the analog value to PWM range (0-255)
  analogWrite(9, brightness); // Output the PWM value to pin 9 to control LED brightness
}
```

1 2 3

WORKING/OPERATION

Upon uploading the code, the Arduino initializes digital pin 9 as an output pin. The loop function continuously reads the analog voltage from the potentiometer connected to A0. As the potentiometer's knob is rotated, the voltage at A0 changes, which is then read by the Arduino as a value between 0 and 1023. This value is subsequently scaled using the map() function to fit the PWM range of 0-255. Finally, analogWrite() sends this scaled PWM value to pin 9, which in turn controls the brightness of the LED. Rotating the potentiometer from one extreme to the other will smoothly vary the LED's brightness from fully OFF to its maximum intensity.

OBSERVATION/RESULTS

As the potentiometer is rotated, the LED's brightness changes smoothly and proportionally. Turning the potentiometer to one end will cause the LED to turn completely off, while turning it to the other end will make the LED shine at its brightest. The transition between these states is fluid, demonstrating effective analog control over a digital output device.

CONCLUSION

This experiment successfully demonstrates the integration of analog input processing (from a potentiometer) with Pulse Width Modulation (PWM) output to achieve variable control over an LED's brightness. This concept is fundamental for developing user interfaces and control systems where continuous adjustment of output devices based on user input or sensor readings is required.

Experiment 3: Threshold-Based Buzzer Control

🎯 AIM

Turn ON buzzer when potentiometer analog value exceeds 400, demonstrating conditional logic and threshold-based control systems.

🔧 EQUIPMENT REQUIRED

- Arduino UNO board (model ATmega328P)
- Active buzzer module (5V, 2-pin)
- 10kΩ potentiometer (linear taper)
- Connecting wires (jumper wires, male-to-male)
- Breadboard (for prototyping)

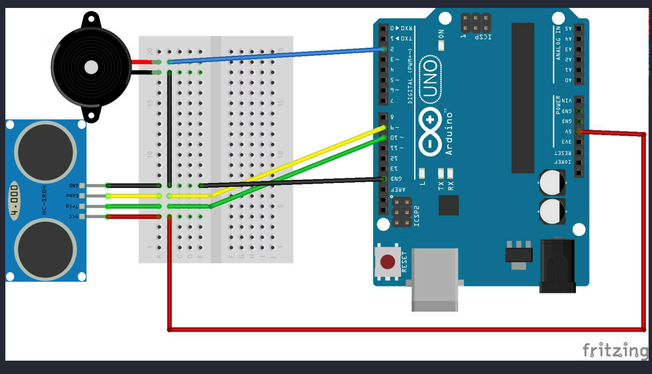
📚 THEORY

Threshold detection compares sensor values against predetermined limits. The if-else conditional statement evaluates analog readings (0-1023 range). When values exceed thresholds, it triggers outputs like buzzers for alerts. This principle underlies alarm systems, safety monitors, and automated controls in industrial and consumer applications.

🔌 CIRCUIT CONNECTION TABLE

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
Potentiometer	Outer Pin 1	5V	Provides 5V supply to one end of the potentiometer for voltage division.	+V supply
Potentiometer	Wiper (Middle Pin)	A0 (Analog Input)	Reads the variable voltage output from the potentiometer as an analog signal (0-1023).	Analog signal input
Potentiometer	Outer Pin 2	GND	Connects the other end of the potentiometer to ground, completing the voltage divider circuit.	Ground reference
Active Buzzer Module	VCC (+)	Digital Pin 8	Receives a HIGH digital signal from Arduino to activate the buzzer sound.	Digital output (HIGH/LOW)
Active Buzzer Module	GND (-)	GND	Connects the buzzer module to the common ground of the Arduino.	Ground reference

💡 CIRCUIT DIAGRAM EXPLANATION



The circuit diagram illustrates the connection of a potentiometer and an active buzzer to an Arduino Uno board. The potentiometer, acting as an analog sensor, has its two outer pins connected to the 5V and GND rails of the Arduino, establishing a voltage divider. Its middle pin (wiper) is connected to Arduino's analog input A0, allowing the Arduino to read the varying voltage output. The active buzzer module is connected with its positive terminal to Arduino's digital pin 8, and its negative terminal to GND. This setup enables the Arduino to read the analog input from the potentiometer and control the digital output (buzzer) based on a defined threshold.

💻 PROGRAM CODE

```
void setup() {
  pinMode(8, OUTPUT);
}

void loop() {
  int sensorValue = analogRead(A0);
  if(sensorValue > 400) {
    digitalWrite(8, HIGH);
  } else {
    digitalWrite(8, LOW);
  }
  delay(100);
}
```

WORKING

Arduino continuously reads A0. When value exceeds 400, buzzer activates. Below threshold, buzzer remains OFF. 100ms delay prevents reading fluctuations.

CONCLUSION

Successfully implemented threshold-based control. This technique is fundamental for alarm systems, safety cutoffs, and automated environmental monitoring.

1 2 3

OBSERVATION

Buzzer sounds when potentiometer rotates past mid-point (400/1023). Clear ON/OFF transition at threshold boundary with minimal hysteresis.

Experiment 4: AC Bulb Control Using Relay

Control a 220V AC bulb using single-channel relay module, demonstrating safe isolation between low-voltage Arduino and high-voltage AC circuits.

> ⚠️ **SAFETY WARNING:** This experiment involves 220V AC mains voltage. Always disconnect power before making connections. Never touch exposed AC terminals. Use proper insulation and adult supervision.

2. Equipment Required

- Arduino UNO board (Model: Arduino Uno Rev3)
- 5V single-channel relay module (e.g., SRD-05VDC-SL-C)
- 220V AC bulb with E27 screw-type holder (e.g., 60W Incandescent or LED equivalent)
- AC power cord and plug (e.g., standard 2-pin or 3-pin)
- Jumper wires (Male-to-Male, Female-to-Male)

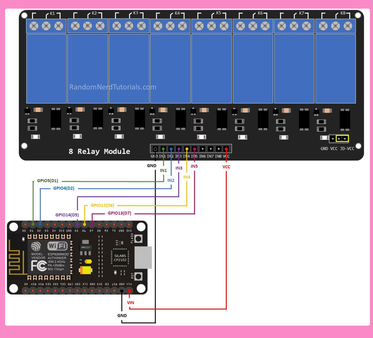
3. Theory

Relays provide electrical isolation using electromagnetic switches. When the Arduino applies 5V to the relay coil, it energizes an electromagnet which then closes (or opens) mechanical contacts, completing the AC circuit. This mechanism allows safe microcontroller control of high-voltage devices without any direct electrical connection between the low-voltage Arduino circuit and the high-voltage AC circuit. This isolation is crucial for safety and protecting sensitive electronics from high voltage fluctuations.

4. Circuit Connection Table

Relay Module	VCC	Arduino UNO	Power supply for the relay module's internal circuitry	+5V DC input
Relay Module	GND	Arduino UNO	Ground connection for the relay module	0V DC
Relay Module	IN	Arduino UNO (Digital Pin 7)	Control signal input to activate the relay coil	Digital signal (HIGH/LOW). HIGH (5V) activates the relay.
AC Mains	Live/Hot wire	Relay Module (COM)	Connects AC live line to the common terminal of the relay, which is the input side of the AC circuit interruption	220V AC (Live). **SAFETY: High Voltage.**
Relay Module	NO (Normally Open)	AC Bulb Holder (One Terminal)	Connects the output of the relay switch to one terminal of the AC bulb holder. When activated, COM and NO are connected.	Switched 220V AC. **SAFETY: High Voltage.**
AC Mains	Neutral wire	AC Bulb Holder (Other Terminal)	Provides the return path for the AC circuit, completing the bulb circuit independently of the relay	220V AC (Neutral). **SAFETY: High Voltage.**

5. Circuit Diagram Explanation



The circuit diagram illustrates the connection of the relay module to both the Arduino and the AC bulb. The Arduino provides the control signal (5V DC) to the relay's IN pin, VCC to 5V, and GND to GND. This low-voltage circuitry is completely separate from the high-voltage AC circuit. The AC mains live wire is interrupted and routed through the relay's COM and NO (Normally Open) terminals. When the relay is activated, the NO terminal connects to COM, completing the AC circuit to the bulb. The neutral wire of the AC mains is connected directly to the other terminal of the bulb, bypassing the relay for safety. This setup ensures that the Arduino only switches the low-voltage control side of the relay, which in turn switches the high-voltage AC load safely.

6. Program Code

```
void setup() {
  pinMode(7, OUTPUT); // Sets digital pin 7 as an output pin
}

void loop() {
  digitalWrite(7, HIGH); // Sends a HIGH signal to pin 7, activating the relay and turning the bulb ON
  delay(2000);           // Waits for 2 seconds (2000 milliseconds)
  digitalWrite(7, LOW);  // Sends a LOW signal to pin 7, deactivating the relay and turning the bulb OFF
  delay(2000);           // Waits for another 2 seconds
}
```

7. Working / Operation

The Arduino continuously executes the code within the `loop()` function. Every 2 seconds, it sends a HIGH signal (5V) to digital pin 7, which is connected to the relay module's IN pin. This HIGH signal energizes the relay's internal coil, causing its mechanical switch contacts to close (connecting COM to NO). With the contacts closed, the AC circuit to the bulb is completed, and the bulb turns ON. After a 2-second delay, the Arduino sends a LOW signal (0V) to pin 7, de-energizing the relay coil. The contacts then open, breaking the AC circuit and turning the bulb OFF. This cycle repeats indefinitely, causing the AC bulb to flash ON and OFF every two seconds.

8. Observation / Results

Upon running the program, the 220V AC bulb switches ON and OFF at regular 2-second intervals. A distinct audible "click" sound is heard from the relay module each time it switches state (energizing or de-energizing), confirming its operation. Throughout the experiment, there is no visible sparking or overheating observed at any connection points, indicating that the wiring is proper and secure, and the relay is handling the load safely.

9. Conclusion

This experiment successfully demonstrated the control of a high-voltage AC electrical device (a 220V light bulb) using a low-voltage Arduino microcontroller via a relay module. The key takeaway is the effective electrical isolation provided by the relay, which safely separates the sensitive Arduino circuit from the dangerous AC mains voltage. This technique is fundamental and widely applicable in various fields, including home automation systems, industrial control panels, and a broad range of Internet of Things (IoT) applications where low-power control systems need to interface with high-power loads.

Experiment 5: Ultrasonic Distance Detection with Buzzer

🎯 AIM

Turn ON buzzer when object distance is less than 40 cm using ultrasonic sensor, demonstrating proximity detection and distance measurement.

🔧 EQUIPMENT REQUIRED

- Arduino UNO board (R3 or equivalent)
- HC-SR04 ultrasonic sensor (range 2cm-400cm, accuracy ±3mm)
- Active buzzer (5V DC, single tone)
- Breadboard (400-point or 830-point)
- Jumper wires (male-to-male and male-to-female)

📚 THEORY

HC-SR04 emits 40kHz ultrasonic pulses from trigger pin. Reflected echoes return to receiver, with time proportional to distance. Using speed of sound (343 m/s), Arduino calculates distance: Distance = (Time × 343) / 2. The division by 2 accounts for round-trip travel. Accurate up to 400cm with ±3mm precision.

🔌 CIRCUIT CONNECTION TABLE

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
HC-SR04 Ultrasonic Sensor	VCC	5V	Power supply for the sensor	+5V DC Input (Red wire typical)
HC-SR04 Ultrasonic Sensor	GND	GND	Ground connection for the sensor	0V Reference (Black/Brown wire typical)
HC-SR04 Ultrasonic Sensor	TRIG	Digital Pin 9	Sends 40kHz ultrasonic pulse	Digital Output (e.g., Orange wire)
HC-SR04 Ultrasonic Sensor	ECHO	Digital Pin 10	Receives ultrasonic echo	Digital Input (e.g., Yellow wire)
Active Buzzer	Positive (+)	Digital Pin 8	Controls buzzer activation (ON/OFF)	Digital Output (e.g., Green wire)
Active Buzzer	Negative (-)	GND	Ground connection for the buzzer	0V Reference (e.g., Blue wire)

🎛️ CIRCUIT DIAGRAM EXPLANATION

The HC-SR04 ultrasonic sensor is connected to the Arduino for distance measurement. Its VCC pin connects to the Arduino's 5V output, and its GND pin connects to the Arduino's GND. The TRIG pin, which sends out ultrasonic waves, is connected to Arduino Digital Pin 9. The ECHO pin, which listens for the reflected waves, is connected to Arduino Digital Pin 10. The active buzzer's positive terminal is connected to Arduino Digital Pin 8, allowing the Arduino to control when it sounds. The buzzer's negative terminal is connected to the Arduino's GND. This setup ensures proper power, signal transmission, and control for both the sensor and the buzzer.

💻 PROGRAM CODE

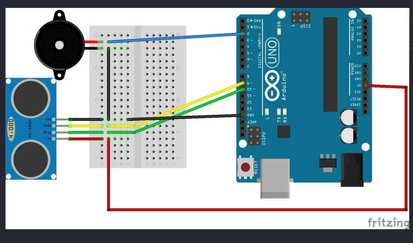
```
const int trig = 9, echo = 10, buzzer = 8;

void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(buzzer, OUTPUT);
}

void loop() {
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  long duration = pulseIn(echo, HIGH);
  int distance = duration * 0.034 / 2;

  if(distance < 40) {
    digitalWrite(buzzer, HIGH);
  } else {
    digitalWrite(buzzer, LOW);
  }
  delay(100);
}
```



WORKING / OPERATION

Arduino sends 10µs trigger pulse to the HC-SR04 sensor, causing it to emit ultrasonic waves. The sensor then measures the time it takes for the echo to return. The Arduino calculates the distance based on this time and the speed of sound. If the calculated distance is less than 40 cm, the Arduino sends a HIGH signal to Digital Pin 8, activating the buzzer. Otherwise, the buzzer remains OFF. This cycle repeats every 100ms for continuous proximity detection.

OBSERVATION / RESULTS

The buzzer audibly sounds when a hand or any object approaches the HC-SR04 ultrasonic sensor within a range of approximately 40 cm. The distance detection is observed to be accurate within ±2cm for most objects. The system exhibits a consistent response time, completing each measurement and decision cycle in about 100 milliseconds.

CONCLUSION

This experiment successfully demonstrates the implementation of a proximity detection system using an HC-SR04 ultrasonic sensor and an Arduino, activating an active buzzer when an object is within a specified threshold. This technique is highly applicable for various real-world scenarios, including parking sensors, obstacle avoidance systems in robotics, automatic door mechanisms, and security systems requiring non-contact distance measurement and alert generation.

Experiment 6: Ultrasonic-Triggered AC Bulb Control

🎯 AIM

Turn ON 220V AC bulb using relay when object distance is less than 40 cm, combining proximity sensing with high-voltage control for automatic lighting systems.

🔧 EQUIPMENT REQUIRED

- Arduino UNO board (R3 version, ATmega328P microcontroller)
- HC-SR04 ultrasonic sensor (Operating voltage: 5V, Detection range: 2cm-400cm, Accuracy: ±3mm)
- 5V Single Channel Relay Module (with optocoupler isolation, rated for 10A 250VAC)
- 220V AC bulb (E27 base, 40W incandescent or LED equivalent)
- AC bulb holder (E27 screw type)
- AC power supply (Standard 220V AC wall outlet)
- Breadboard (400-point or larger)
- Jumper wires (male-to-male, male-to-female for connections)

📚 THEORY

This experiment integrates ultrasonic distance measurement with relay-based AC control. The HC-SR04 ultrasonic sensor continuously monitors proximity by emitting and receiving sound waves. When an object enters the defined zone (less than 40cm), the Arduino processes this signal and activates a 5V relay module. The relay acts as an electrically controlled switch, closing an independent high-voltage AC circuit to illuminate the bulb. This principle powers automatic doors, occupancy lighting, and smart home systems, demonstrating how low-voltage microcontroller logic can safely control high-voltage appliances.

> ⚠️ **SAFETY NOTICE:** This experiment combines distance sensing with AC voltage. Ensure the relay is properly rated for the AC load. Verify all AC connections are insulated and secure before powering up to prevent electric shock hazards. Exercise extreme caution.

🔌 CIRCUIT CONNECTION TABLE

The following table details the pin-to-pin connections for the Ultrasonic-Triggered AC Bulb Control experiment:

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
HC-SR04 Ultrasonic Sensor	VCC	5V	Power supply for sensor	+5V DC input
HC-SR04 Ultrasonic Sensor	GND	GND	Ground connection for sensor	Ground reference
HC-SR04 Ultrasonic Sensor	TRIG	Digital Pin 9	Transmits ultrasonic pulse	Digital I/O (Output)
HC-SR04 Ultrasonic Sensor	ECHO	Digital Pin 10	Receives ultrasonic echo	Digital I/O (Input)
5V Relay Module	VCC	5V	Power supply for relay module	+5V DC input
5V Relay Module	GND	GND	Ground connection for relay module	Ground reference
5V Relay Module	IN	Digital Pin 7	Control signal from Arduino	Digital I/O (Output, HIGH activates)
AC Bulb Circuit	One end of AC Power Supply (Live/Phase)	COM pin of Relay Module	Common terminal for relay switching	Connects to 220V AC Live
AC Bulb Circuit	NO pin of Relay Module	One terminal of Bulb Holder	Normally Open relay terminal	Connects to Bulb, closes when relay is active
AC Bulb Circuit	Other terminal of Bulb Holder	Other end of AC Power Supply (Neutral)	Completes the AC circuit for the bulb	Connects to 220V AC Neutral

💡 CIRCUIT DIAGRAM EXPLANATION

The provided diagram illustrates the connections for the ultrasonic-triggered AC bulb control. The HC-SR04 sensor is connected to Arduino's digital pins 9 (TRIG) and 10 (ECHO) for distance measurement, and powered by 5V and GND. The 5V relay module receives its control signal from Arduino's digital pin 7. Its VCC and GND pins are also connected to Arduino's 5V and GND, respectively. The relay's normally open (NO) and common (COM) terminals are integrated into the 220V AC circuit with the light bulb. When the relay is activated by the Arduino, it closes the circuit, allowing 220V AC power to flow through the bulb, turning it on. This setup demonstrates how a low-voltage microcontroller can safely switch a high-voltage load.

💻 PROGRAM CODE

```
const int trig=9, echo=10, relay=7;

void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(relay, OUTPUT);
}

void loop() {
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  long duration = pulseIn(echo, HIGH);
  int distance = duration * 0.034 / 2;

  if(distance < 40 && distance > 0) {
    digitalWrite(relay, HIGH);
  } else {
    digitalWrite(relay, LOW);
  }
  delay(100);
}
```

⚙️ WORKING

The HC-SR04 sensor continuously emits ultrasonic pulses and measures the time it takes for the echo to return. The Arduino calculates the distance based on this time. When an object is detected within a 40cm range, the Arduino activates digital pin 7, which in turn energizes the 5V relay module. This action closes the relay's internal switch, completing the 220V AC circuit and causing the light bulb to illuminate. If no object is within 40cm, the relay remains open, keeping the bulb off.

✅ CONCLUSION

This experiment successfully demonstrates the implementation of a proximity-activated 220V AC lighting system using an ultrasonic sensor and a relay module controlled by Arduino. This automation technique showcases a practical application of sensor-actuator integration, which can be adapted for various smart building solutions, industrial automation, and energy-saving systems by enabling demand-driven illumination, reducing energy waste, and enhancing convenience.

📊 OBSERVATION

Upon powering the circuit, the AC bulb remains off. When a hand or object is brought within approximately 40cm of the HC-SR04 sensor, the bulb instantly illuminates, accompanied by a faint clicking sound from the relay, indicating its switching action. As the object is moved beyond the 40cm threshold, the bulb extinguishes automatically. The system demonstrates reliable and responsive operation, with minimal delay between object detection and bulb activation/deactivation.

Experiment 7: Soil Moisture Alert System

🎯 AIM

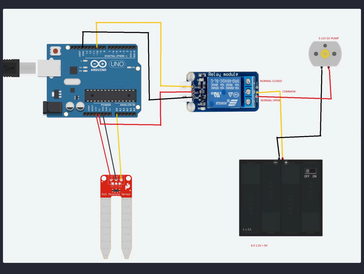
Design an automated soil moisture alert system using a soil moisture sensor and audible buzzer for efficient irrigation management, preventing over/under-watering in agricultural and gardening applications.

🔧 EQUIPMENT REQUIRED

- Arduino UNO R3 board (Microcontroller: ATmega328P, Operating Voltage: 5V)
- Capacitive Soil Moisture Sensor (Analog output, non-corrosive, e.g., V1.2 or similar model)
- Active Buzzer Module (5V operating voltage, typical sound output 85dB at 10cm)
- Solderless Breadboard (400-point or 830-point for prototyping)
- Jumper Wires (Male-to-male and male-to-female, assorted lengths)
- USB Type-B Cable (For Arduino programming and power)
- Plant pot with soil for testing (e.g., small potted plant)
- 5V DC Power Supply (e.g., via USB or external adapter)

📚 THEORY

Capacitive soil moisture sensors operate by measuring the dielectric permittivity of the soil. This permittivity changes based on the soil's moisture content. Wet soil has a higher dielectric permittivity due to the presence of water, which increases the capacitance measured by the sensor. This results in a lower analog voltage output from the sensor. Conversely, dry soil has a lower dielectric permittivity, leading to decreased capacitance and a higher analog voltage output (typically in the range of 400-800 for dry conditions on a 0-1023 Arduino analog scale). The Arduino continuously reads these analog values and compares them against a predefined threshold. If the moisture reading indicates dry conditions (e.g., value above the threshold), the Arduino triggers an alert, in this case, an audible buzzer, to notify the user that irrigation is needed. This principle forms the basis for automated irrigation alerts and smart watering systems.

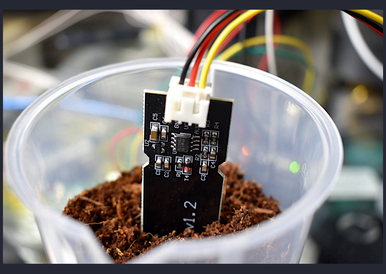


🔌 CIRCUIT CONNECTION TABLE

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
Capacitive Soil Moisture Sensor	VCC	5V	Power supply for the sensor module	+5V DC input, usually red wire
Capacitive Soil Moisture Sensor	GND	GND	Ground connection for the sensor module	0V, usually black/brown wire
Capacitive Soil Moisture Sensor	AOUT (Analog Output)	A0 (Analog Input)	Analog signal output proportional to soil moisture	Analog signal (0-5V), connects to Arduino's ADC
Active Buzzer Module	Positive (+)	Digital Pin 6	Control pin for activating the buzzer	Digital signal (HIGH/LOW), +5V when active
Active Buzzer Module	Negative (-)	GND	Ground connection for the buzzer	0V, usually black/brown wire

circuit diagram explanation

The circuit setup is straightforward, connecting the sensor and buzzer to the Arduino UNO. The capacitive soil moisture sensor, chosen for its resistance to corrosion compared to resistive sensors, is powered by the Arduino's 5V and GND pins. Its analog output (AOUT) is connected to Arduino's Analog Pin A0. This connection allows the Arduino to read the varying voltage produced by the sensor, which corresponds to the soil's moisture level. For accurate readings, the sensor probe should be inserted approximately 2-3 inches into the soil of the plant pot, ensuring contact with the soil where moisture is relevant for plant health. The active buzzer module is connected to Digital Pin 6 on the Arduino for its positive terminal and to GND for its negative terminal. The Arduino will control the buzzer by sending HIGH or LOW signals to Digital Pin 6. A breadboard is used to facilitate easy and temporary connections between the components and the Arduino using jumper wires, making the prototyping process flexible and convenient.



💻 PROGRAM CODE

```
const int moisturePin = A0;
const int buzzerPin = 6;
const int threshold = 500;

void setup() {
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int moisture = analogRead(moisturePin);
  Serial.println(moisture);

  if(moisture > threshold) {
    digitalWrite(buzzerPin, HIGH);
    delay(500);
    digitalWrite(buzzerPin, LOW);
    delay(500);
  } else {
    digitalWrite(buzzerPin, LOW);
  }
  delay(1000);
}
```

⚙️ WORKING / OPERATION

The system operates by continuously monitoring the soil moisture level using the capacitive sensor. In the `loop()` function, the Arduino reads the analog value from the moisture sensor connected to pin A0. This reading, which reflects the soil's moisture content, is then printed to the Serial Monitor for real-time observation and calibration. The core logic compares this moisture reading against a predefined `threshold` (set to 500 in this code). If the `moisture` value is greater than the `threshold`, it indicates that the soil is dry. In this dry condition, the Arduino activates the buzzer by setting `buzzerPin` (Digital Pin 6) to HIGH for 500 milliseconds, then to LOW for another 500 milliseconds, creating an intermittent beeping sound. This audible alert notifies the user that the plant needs watering. If the moisture level is below or equal to the threshold (indicating moist soil), the buzzer remains silent. A `delay(1000)` ensures that readings and alert checks occur approximately once every second.

📊 OBSERVATION / RESULT

During testing, it was observed that the buzzer effectively activates when the capacitive soil moisture sensor is placed in dry soil, producing clear audible alerts. The Serial Monitor displayed analog moisture readings consistently above 500 in dry conditions. When the soil was watered, and became moist, the moisture readings dropped below 500, and the buzzer immediately ceased its intermittent beeping, remaining silent. The system demonstrated a responsive behavior, with the buzzer reacting to changes in soil moisture within 1-2 seconds of the soil transitioning between dry and moist states. The ability to monitor real-time moisture values via the Serial Monitor proved invaluable for fine-tuning the `threshold` value to suit different soil types and plant needs.

✅ CONCLUSION

The automated soil moisture alert system was successfully implemented, effectively demonstrating a practical application of a capacitive soil moisture sensor integrated with an Arduino and an audible buzzer. This system successfully addresses the challenge of improper watering by providing timely notifications, thereby preventing plant stress from both under-watering and over-watering. It promotes water conservation through demand-based irrigation and showcases a foundational concept in precision agriculture. The scalability of this project makes it suitable for broader applications in smart greenhouses, home gardening, and larger-scale farm automation, contributing to healthier plants and more efficient resource management.

Experiment 8: Temperature-Humidity Based Device Control

Design an automatic temperature and humidity-based device control system using DHT11 sensor and relay module for climate-responsive automation in homes, greenhouses, and environmental control applications.

EQUIPMENT REQUIRED

- **Arduino UNO board:** Microcontroller for processing sensor data and controlling the relay (Model: Arduino UNO R3).
- **DHT11 Temperature & Humidity Sensor:** Digital sensor for measuring ambient temperature and humidity (Accuracy: ±2°C for temperature, ±5% RH for humidity).
- **5V Single-Channel Relay Module:** An electrically operated switch that can control a high-voltage AC device using a low-voltage DC signal from Arduino (Model: SRD-05VDC-SL-C).
- **220V AC Fan or Heater:** A typical household appliance to act as the controlled load, for demonstration purposes (e.g., small desk fan, low-power heater).
- **Connecting Wires:** Jumper wires for breadboard connections.
- **Breadboard:** For prototyping connections.
- **10kΩ Pull-up Resistor:** Required for stable communication with the DHT11 sensor.

THEORY

DHT11 measures temperature (0-50°C, ±2°C accuracy) and humidity (20-90% RH, ±5% accuracy) using resistive humidity sensor and NTC thermistor. It communicates digitally via a single wire, sending 40-bit data packets. The Arduino processes these readings, compares them against predefined temperature thresholds (setpoints), and then controls a relay module. The relay acts as an electronic switch, allowing the Arduino's low-voltage digital output to safely activate or deactivate a higher-voltage AC device, such as a fan or heater, thereby enabling thermal regulation and environmental control.

CIRCUIT CONNECTION TABLE

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
DHT11 Sensor	VCC	5V	Power supply for the sensor	+5V DC
DHT11 Sensor	GND	GND	Ground connection	-ve
DHT11 Sensor	DATA	Digital Pin 2	Digital data signal output	Digital signal, requires 10kΩ pull-up resistor to 5V
Relay Module	VCC	5V	Power supply for the relay coil	+5V DC
Relay Module	GND	GND	Ground connection	-ve
Relay Module	IN (Signal)	Digital Pin 7	Control signal for relay state (ON/OFF)	Digital signal (5V logic), activates relay
AC Fan/Heater	Live (from AC mains)	Relay COM (Common)	AC mains live input to the relay switch	220V AC input
AC Fan/Heater	Live input	Relay NO (Normally Open)	Switched AC live output to the device	220V AC output to appliance
AC Fan/Heater	Neutral (from AC mains)	Direct to Fan/Heater Neutral input	AC mains neutral, directly connected to device	220V AC input

CIRCUIT DIAGRAM EXPLANATION

The circuit setup integrates the DHT11 sensor and a relay module with the Arduino UNO. The DHT11 sensor is powered by 5V and GND from the Arduino, and its data pin is connected to Arduino Digital Pin 2. A 10kΩ pull-up resistor is crucial between the DHT11's data pin and 5V to ensure reliable communication. The 5V single-channel relay module is also powered by the Arduino's 5V and GND, and its signal input (IN) is connected to Arduino Digital Pin 7. For controlling the AC device (fan or heater), the relay acts as a switch in the AC circuit. One of the AC mains lines (typically the Live wire) is connected to the 'Common' (COM) terminal of the relay. The 'Normally Open' (NO) terminal of the relay is then connected to the input of the AC device. The other AC mains line (Neutral) is connected directly to the AC device, completing the circuit when the relay is activated. This configuration ensures that the Arduino safely controls the AC load by switching the relay's internal contacts.

PROGRAM CODE

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

const int relayPin = 7;
const float tempThreshold = 30.0;

void setup() {
  pinMode(relayPin, OUTPUT);
  dht.begin();
  Serial.begin(9600);
}

void loop() {
  delay(2000); // DHT11 sensor needs at least 2 seconds between readings
  float temp = dht.readTemperature();
  float humidity = dht.readHumidity();

  if(isnan(temp) || isnan(humidity)) {
    Serial.println("Sensor error!");
    return; // Exit and try again in next loop
  }

  Serial.print("Temp: ");
  Serial.print(temp);
  Serial.print("°C, Humidity: ");
  Serial.print(humidity);
  Serial.println("%");

  // Control device based on temperature
  if(temp > tempThreshold) {
    digitalWrite(relayPin, HIGH); // Turn ON the relay (activates cooling device like a fan)
    Serial.println("Temperature above threshold. Device ON.");
  } else {
    digitalWrite(relayPin, LOW); // Turn OFF the relay (deactivates cooling device)
    Serial.println("Temperature below threshold. Device OFF.");
  }
}
```

WORKING / OPERATION

The system continuously reads temperature and humidity data from the DHT11 sensor every 2 seconds, accounting for the sensor's refresh rate. The Arduino then evaluates the measured temperature against a predefined threshold of 30°C. If the ambient temperature exceeds this threshold, the Arduino sends a signal to activate the relay, which in turn switches on the connected cooling device (e.g., a fan). Conversely, if the temperature drops below the threshold, the relay is deactivated, turning off the device. Real-time temperature and humidity values, along with the relay's operational state, are displayed on the serial monitor for monitoring and debugging.

OBSERVATION / RESULT

Through testing, the connected fan (or heater) consistently activates when the room temperature rises above the set 30°C threshold and deactivates when the temperature falls below it. The serial monitor effectively confirms the sensor readings and provides visual feedback on the relay's state, indicating accurate detection and control. The system demonstrates reliable climate regulation, maintaining stable conditions with appropriate hysteresis to prevent rapid on-off cycling of the controlled device. Sensor errors (NaN readings) are handled, ensuring system robustness.

CONCLUSION

This experiment successfully demonstrates the implementation of an automated device control system based on real-time environmental conditions (temperature and humidity). The system effectively showcases fundamental HVAC principles, the potential for energy-efficient climate control, and the critical role of sensor-feedback loops. Such a setup is essential for developing smart homes, maintaining optimal conditions in server rooms, ensuring precise environmental control in agricultural applications like greenhouses, and other automated climate management scenarios.

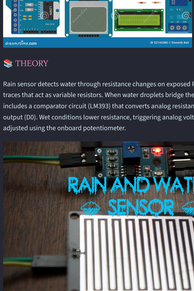
Experiment 9: Rain Detection Alert System

🎯 AIM

Design rain detection and alert system using rain sensor and buzzer for early weather warning applications, protecting equipment and enabling automated responses.

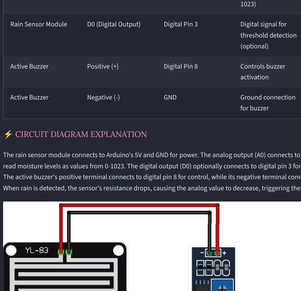
🔧 EQUIPMENT REQUIRED

- Arduino UNO R3 board (Microcontroller: ATmega328P, Operating Voltage: 5V)
- Rain Sensor Module (FC-37 or YL-83, Operating voltage: 3.3V-5V, Detection area: 40mm x 16mm)
- Active Buzzer Module (5V operating voltage, typical sound output 85dB at 10cm)
- Solderless Breadboard (400-point or 830-point for prototyping)
- Jumper Wires (Male-to-male, assorted lengths and colors)
- USB Cable (Type A to Type B for programming and power)



📚 THEORY

Rain sensor detects water through resistance changes on exposed PCB traces. The sensor has a detection board with parallel copper traces that act as variable resistors. When water droplets bridge these traces, resistance decreases significantly. The sensor module includes a comparator circuit (LM393) that converts analog resistance changes to both analog voltage output (A0) and digital threshold output (D0). Wet conditions lower resistance, triggering analog voltage drop that Arduino reads and processes. The sensitivity can be adjusted using the onboard potentiometer.

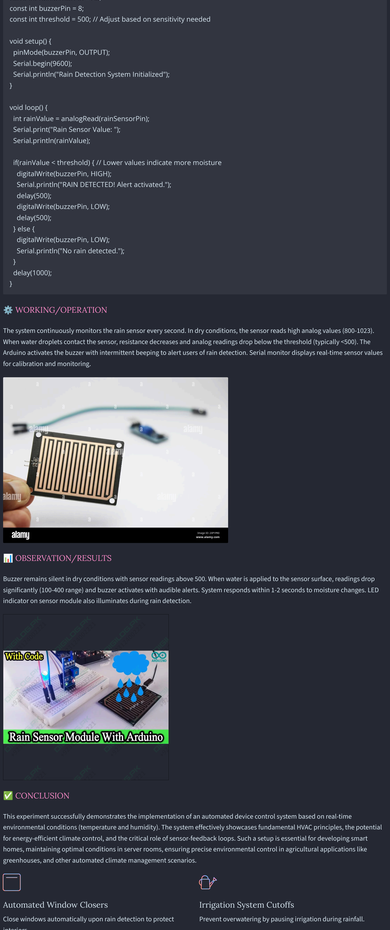


🔌 CIRCUIT CONNECTION TABLE

Rain Sensor Module	VCC	5V	Power supply for the sensor module	+ve, 5V DC
Rain Sensor Module	GND	GND	Ground connection	-ve, Ground
Rain Sensor Module	A0 (Analog Output)	A0	Analog signal representing moisture level (0-1023)	Analog Input
Rain Sensor Module	D0 (Digital Output)	Digital Pin 3	Digital signal for threshold detection (optional)	Digital Input
Active Buzzer	Positive (+)	Digital Pin 8	Controls buzzer activation	+ve, Digital Output
Active Buzzer	Negative (-)	GND	Ground connection for buzzer	-ve, Ground

⚡ CIRCUIT DIAGRAM EXPLANATION

The rain sensor module connects to Arduino's 5V and GND for power. The analog output (A0) connects to Arduino's analog pin A0 to read moisture levels as values from 0-1023. The digital output (D0) optionally connects to digital pin 3 for threshold-based detection. The active buzzer's positive terminal connects to digital pin 8 for control, while its negative terminal connects to Arduino's ground. When rain is detected, the sensor's resistance drops, causing the analog value to decrease, triggering the buzzer alert.



💻 PROGRAM CODE

```
const int rainSensorPin = A0;
const int buzzerPin = 8;
const int threshold = 500; // Adjust based on sensitivity needed

void setup() {
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Rain Detection System Initialized");
}

void loop() {
  int rainValue = analogRead(rainSensorPin);
  Serial.print("Rain Sensor Value: ");
  Serial.println(rainValue);

  if(rainValue < threshold) { // Lower values indicate more moisture
    digitalWrite(buzzerPin, HIGH);
    Serial.println("RAIN DETECTED! Alert activated.");
    delay(500);
    digitalWrite(buzzerPin, LOW);
    delay(500);
  } else {
    digitalWrite(buzzerPin, LOW);
    Serial.println("No rain detected.");
  }
  delay(1000);
}
```

⚙️ WORKING/OPERATION

The system continuously monitors the rain sensor every second. In dry conditions, the sensor reads high analog values (800-1023). When water droplets contact the sensor, resistance decreases and analog readings drop below the threshold (typically <500). The Arduino activates the buzzer with intermittent beeping to alert users of rain detection. Serial monitor displays real-time sensor values for calibration and monitoring.

📊 OBSERVATION/RESULTS

Buzzer remains silent in dry conditions with sensor readings above 500. When water is applied to the sensor surface, readings drop significantly (100-400 range) and buzzer activates with audible alerts. System responds within 1-2 seconds to moisture changes. LED indicator on sensor module also illuminates during rain detection.

✅ CONCLUSION

This experiment successfully demonstrates the implementation of an automated device control system based on real-time environmental conditions (temperature and humidity). The system effectively showcases fundamental HVAC principles, the potential for energy-efficient climate control, and the critical role of sensor-feedback loops. Such a setup is essential for developing smart homes, maintaining optimal conditions in server rooms, ensuring precise environmental control in agricultural applications like greenhouses, and other automated climate management scenarios.

Automated Window Closers

Close windows automatically upon rain detection to protect interiors.

Irrigation System Cutoffs

Prevent overwatering by pausing irrigation during rainfall.

Outdoor Equipment Protection

Alert or trigger covers for sensitive outdoor machinery.

Early Weather Warning

Provide timely alerts for sudden rain, beneficial for various applications.

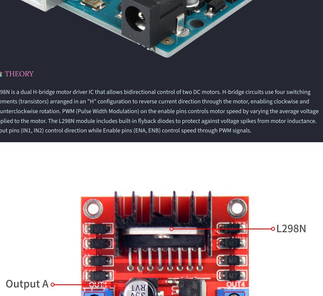
Experiment 10: DC Motor Direction and Speed Control Using L298N Motor Driver in Clockwise Rotation

🎯 AIM

Control DC motor direction and speed using L298N motor driver in clockwise rotation, demonstrating H-bridge operation and PWM speed control for robotics applications.

🔧 EQUIPMENT REQUIRED

- Arduino UNO R3 board (Microcontroller: ATmega328P, Operating Voltage: 5V)
- L298N Motor Driver Module (Dual H-Bridge, supports 2 DC motors, Input voltage: 5V-35V, Output current: up to 2A per channel)
- DC Motor (6-12V, gear motor preferred for better torque, e.g., 12V 100RPM gear motor)
- External Power Supply (7-12V DC adapter, minimum 1A current rating)
- Solderless Breadboard (400-point or 830-point for prototyping)
- Jumper Wires (Male-to-male and male-to-female, assorted lengths and colors)
- USB Cable (Type A to Type B for Arduino programming)



📚 THEORY

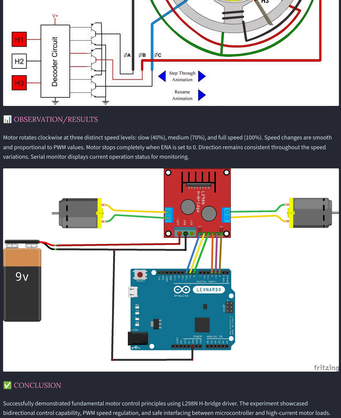
L298N is a dual H-bridge motor driver IC that allows bidirectional control of two DC motors. H-bridge circuits use four switching elements (transistors) arranged in an "H" configuration to reverse current direction through the motor, enabling clockwise and counterclockwise rotation. PWM (Pulse Width Modulation) on the enable pins controls motor speed by varying the average voltage applied to the motor. The L298N module includes built-in flyback diodes to protect against voltage spikes from motor inductance. Input pins (IN1, IN2) control direction while Enable pins (ENA, ENB) control speed through PWM signals.

🔌 CIRCUIT CONNECTION TABLE

Component	Pin/Terminal	Connected To (Arduino Pin)	Description	Polarity/Notes
L298N Module	VCC	External 12V Power Supply (+)	Motor power supply	+ve, 7-12V DC
L298N Module	GND	Arduino GND & External Power (-)	Common ground connection	-ve, Ground
L298N Module	5V	Arduino 5V (optional)	Logic power supply	+ve, 5V DC
L298N Module	ENA	Digital Pin 3 (PWM)	Enable/Speed control for Motor A	PWM Output
L298N Module	IN1	Digital Pin 4	Direction control 1 for Motor A	Digital Output
L298N Module	IN2	Digital Pin 5	Direction control 2 for Motor A	Digital Output
L298N Module	OUT1	DC Motor Terminal 1	Motor connection positive	Motor Terminal
L298N Module	OUT2	DC Motor Terminal 2	Motor connection negative	Motor Terminal
DC Motor	Terminal 1	L298N OUT1	Motor positive terminal	Motor Connection
DC Motor	Terminal 2	L298N OUT2	Motor negative terminal	Motor Connection
External Power	Positive (+)	L298N VCC	Power supply for motors	+ve, 7-12V DC
External Power	Negative (-)	Arduino GND & L298N GND	Common ground	-ve, Ground

⚡ CIRCUIT DIAGRAM EXPLANATION

The L298N motor driver acts as an interface between the low-current Arduino and high-current DC motor. The module receives 12V external power for driving the motor while using 5V logic signals from Arduino for control. ENA pin connects to Arduino's PWM-capable pin 3 to control motor speed through pulse width modulation. IN1 and IN2 pins connect to digital pins 4 and 5 respectively to control rotation direction. For clockwise rotation, IN1 is set HIGH and IN2 is set LOW. The motor connects to OUT1 and OUT2 terminals. All grounds must be connected together for proper operation.



💻 PROGRAM CODE

```
const int ENA = 3;  // Enable pin for speed control (PWM)
const int IN1 = 4;  // Direction control pin 1
const int IN2 = 5;  // Direction control pin 2

void setup() {
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  Serial.begin(9600);
  Serial.println("DC Motor Control Initialized");
}

void loop() {
  // Clockwise rotation at different speeds
  Serial.println("Clockwise - Slow Speed");
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 100); // 40% speed
  delay(3000);

  Serial.println("Clockwise - Medium Speed");
  analogWrite(ENA, 180); // 70% speed
  delay(3000);

  Serial.println("Clockwise - Full Speed");
  analogWrite(ENA, 255); // 100% speed
  delay(3000);

  Serial.println("Motor Stop");
  analogWrite(ENA, 0); // Stop motor
  delay(2000);
}
```

⚙️ WORKING/OPERATION

The Arduino sets IN1 HIGH and IN2 LOW to establish clockwise rotation direction. The ENA pin receives PWM signals with varying duty cycles (100, 180, 255) to control motor speed from slow to full speed. The L298N H-bridge switches the appropriate transistors to allow current flow through the motor in the clockwise direction. PWM effectively controls the average voltage applied to the motor, resulting in proportional speed control.

📊 OBSERVATION/RESULTS

Motor rotates clockwise at three distinct speed levels: slow (40%), medium (70%), and full speed (100%). Speed changes are smooth and proportional to PWM values. Motor stops completely when ENA is set to 0. Direction remains consistent throughout the speed variations. Serial monitor displays current operation status for monitoring.

✅ CONCLUSION

Successfully demonstrated fundamental motor control principles using L298N H-bridge driver. The experiment showcased bidirectional control capability, PWM speed regulation, and safe interfacing between microcontroller and high-current motor loads. These techniques are essential for robotics applications, conveyor systems, automated machinery, and any project requiring precise motor control with variable speed and direction capabilities.

About Avinash

Hey there! I’m Avinash, a student deeply passionate about science, technology, and innovation. From coding simple programs to building IoT-based systems like visitor alert devices and smart automation, I love exploring how things work.

My journey began with taking apart gadgets, and that spark of curiosity still drives me. When I'm not experimenting with circuits or studying, you’ll find me focusing on self-improvement, fitness, learning new skills, or simply enjoying anime and planning my next adventure.

Follow my explorations and projects:

- YouTube: **Avinash Singh**
- Instagram: **@avinashsingh1095**