## Blink LED

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

**Explanation**

1. void setup() {

    This is the setup section.

    It runs only once when the Arduino starts.

2. pinMode(13, OUTPUT);

    We are telling Arduino:

    👉 *“Pin 13 will be used to give output.”*

    Output means the pin will send voltage to control something (like an LED).

3. }

    End of setup.

4. void loop() {

    This is the main loop.

    It runs again and again forever.

5. digitalWrite(13, HIGH);

    Turn the LED ON.

    HIGH means Arduino sends 5 volts on pin 13.

6. delay(1000);

    Wait for 1000 milliseconds = 1 second.

    This makes the LED stay ON for 1 second.

7. digitalWrite(13, LOW);

    Turn the LED OFF.

    LOW means Arduino sends 0 volts.

8. delay(1000);

Wait 1 second again.

This keeps the LED OFF for 1 second.

9. }

End of loop.

Arduino goes back to the start of loop() and repeats → LED keeps blinking.

**SUMMARY**

- Pin 13 is set as output.
- LED is turned ON for 1 second.
- LED is turned OFF for 1 second.
- This repeats forever, so the LED blinks.

## Button Controlled LED

```
int button = 2;
int led = 13;

void setup() {
  pinMode(button, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  int state = digitalRead(button);
  digitalWrite(led, state);
}
```

**Explanation**

1. int button = 2;

   We are creating a variable named button.

   We assign pin number 2 to it.

   👉 This means the button is connected to pin 2.

2. int led = 13;

   We create another variable named led.

   We assign pin number 13 to it.

   👉 This means the LED is connected to pin 13.

3. void setup() {

   Setup runs one time when Arduino starts.

4. pinMode(button, INPUT);

   We are telling Arduino:

   👉 *"Pin 2 will be used to read input."*

   Input means we will read the button state (pressed or not).

5. pinMode(led, OUTPUT);

   We are telling Arduino:

   👉 *"Pin 13 will be used to give output."*

   Output means the pin controls the LED.

6. }

   End of setup.

7. void loop() {

Loop runs again and again forever.

8. int state = digitalRead(button);

Arduino checks the button.

- If button is pressed → value becomes HIGH (1)
- If button is not pressed → value becomes LOW (0)

We store this value in a variable state.

9. digitalWrite(led, state);

We write the same value to the LED.

Meaning:

If button = HIGH → LED turns ON

If button = LOW → LED turns OFF

So the LED copies the button behaviour.

10. }

End of loop.

**SUMMARY**

- Pin 2 is used as input to read the button.
- Pin 13 is used as output to control LED.
- When the button is pressed, LED turns ON.
- When the button is released, LED turns OFF.

## LM35 Temperature Sensor

```
int sensor = A0;
float temp;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int val = analogRead(sensor);
  temp = (val * 5.0 * 100) / 1023;
  Serial.println(temp);
  delay(500);
}
```

**Explanation**

1. int sensor = A0;

    We create a variable called sensor and store A0 in it.

    👉 This means the LM35 output pin is connected to Analog pin A0.

2. float temp;

    We create a variable named temp to store the temperature in decimal form (float).

3. void setup() {

    Setup runs once when Arduino starts.

4. Serial.begin(9600);

    Start serial communication at 9600 baud.

    This allows us to print temperature on the Serial Monitor.

5. }

    End of setup.

6. void loop() {

    Loop runs repeatedly forever.

7. int val = analogRead(sensor);

    We read the analog voltage from LM35 sensor.

    analogRead gives values from 0 to 1023.

    - 0 → 0 volts
    - 1023 → 5 volts

    LM35 gives 10mV per °C.

8. temp = (val * 5.0 * 100) / 1023;

This formula converts analog value to temperature.

Breaking it down:

- val * 5.0 / 1023 → converts analog value into voltage
- Multiply by 100 → LM35 gives 10mV = 0.01V per °C, so multiply by 100 to convert to °C

Final result = temperature in °C

9. Serial.println(temp);

Print the temperature value on the Serial Monitor.

10. delay(500);

Wait for 500 ms (0.5 seconds) before reading again.

11. }

End of loop.

**SUMMARY**

- LM35 is connected to analog pin A0.
- Arduino reads analog value using analogRead().
- Formula converts analog value to °C.
- Temperature is printed on Serial Monitor.
- Updated every 0.5 seconds.

## Ultrasonic Distance Measurement

```
int trig = 8;
int echo = 7;
long duration;
float distance;

void setup() {
  Serial.begin(9600);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
}

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

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

  Serial.println(distance);
  delay(500);
}
```

**Explanation**

1. int trig = 8;

   We store pin 8 in the variable trig.

   👉 Trigger pin of HC-SR04 is connected to Arduino pin 8.

2. int echo = 7;

   We store pin 7 in echo.

   👉 Echo pin of HC-SR04 is connected to Arduino pin 7.

3. long duration;

   This variable will store the time taken (in microseconds) for the sound wave to return.

4. float distance;

   This will store the final distance in centimeters.

5. void setup() {

Runs only once.

6. Serial.begin(9600);

Starts serial communication to print distance on Serial Monitor.

7. pinMode(trig, OUTPUT);

Trigger pin must send a signal → OUTPUT.

8. pinMode(echo, INPUT);

Echo pin receives the reflected signal → INPUT.

9. }

End of setup.

Now the LOOP (runs repeatedly)

10. digitalWrite(trig, LOW);

Make sure trigger pin starts at LOW.

11. delayMicroseconds(2);

Wait 2 microseconds (very small delay).

12. digitalWrite(trig, HIGH);

Send a short HIGH pulse → this tells the sensor to send ultrasound waves.

13. delayMicroseconds(10);

Keep trigger HIGH for 10 microseconds (required by HC-SR04).

14. digitalWrite(trig, LOW);

End the trigger pulse.

15. duration = pulseIn(echo, HIGH);

This waits until the echo pin becomes HIGH and measures how long it stays HIGH.

👉 This time = time taken by sound to go → hit object → come back

The time is stored in microseconds.

16. distance = duration * 0.034 / 2;

Formula to convert time → distance.

- Sound speed = 0.034 cm per microsecond
- Divide by 2 because

  ✔ sound goes forward and

  ✔ comes back

  So we only want one-way distance.

Final result: Distance in centimeters (cm).

17. Serial.println(distance);

Print the distance value in the Serial Monitor.

18. delay(500);

Wait 0.5 seconds before taking next reading.

19. }

End of loop.

## SUMMARY

- Trigger pin sends a 10 microsecond pulse.
- Echo pin measures the time for the reflected sound wave.
- Distance = (time × speed of sound) / 2.
- Distance is printed on Serial Monitor.

## Turn LED ON

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
}
```

**Explanation**

1. void setup() {

    This is the setup section.

    It runs one time when the Arduino starts.

2. pinMode(13, OUTPUT);

    We tell Arduino:

    👉 *"Pin 13 will be used as OUTPUT."*

    Output means the pin will send voltage to control something (like an LED).

    So the LED is connected to pin 13.

3. }

    End of setup.

4. void loop() {

    This loop runs continuously again and again.

5. digitalWrite(13, HIGH);

    Turn ON the LED.

    HIGH sends 5 volts to pin 13.

    So the LED stays ON all the time.

6. }

    End of loop.

**SUMMARY**

- Pin 13 is set as output.
- LED connected to pin 13 is turned ON using digitalWrite(13, HIGH).
- LED remains ON because loop keeps repeating the same command.

## Turn LED OFF

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, LOW);
}
```

**Explanation**

1. void setup() {

    Setup section.

    It runs only once when Arduino starts.

2. pinMode(13, OUTPUT);

    We set pin 13 as OUTPUT.

    This means pin 13 will control something externally (like an LED).

3. }

    End of setup.

4. void loop() {

    Loop runs again and again.

5. digitalWrite(13, LOW);

    This turns OFF the LED.

    LOW means 0 volts is sent from pin 13.

    Since the loop keeps repeating the same instruction, the LED stays OFF permanently.

6. }

    End of loop.

**SUMMARY**

- Pin 13 is set as output.
- digitalWrite(13, LOW) sends 0V to the LED.
- LED remains OFF continuously.

## Fade LED (PWM on Pin 9)

```
void setup() {
  pinMode(9, OUTPUT);
}

void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(9, i);
    delay(10);
  }
  for (int i = 255; i >= 0; i--) {
    analogWrite(9, i);
    delay(10);
  }
}
```

**Explanation**

1. void setup() {

    Setup runs once when Arduino starts.

2. pinMode(9, OUTPUT);

    We set pin 9 as OUTPUT.

    👉 Pin 9 supports PWM, so we can control LED brightness.

3. }

    End of setup.

4. void loop() {

    Loop runs forever.

5. for (int i = 0; i <= 255; i++) {

    A for loop that starts from 0 and goes to 255.

    This value (i) represents brightness level.

    - 0 = LED completely OFF
    - 255 = LED fully ON

    It will increase brightness step by step.

6. analogWrite(9, i);

    This sends a PWM signal to pin 9.

    👉 PWM = fast ON/OFF switching

    👉 Controls LED brightness

✔ Higher value = brighter

✔ Lower value = dimmer

7. delay(10);

Wait 10 milliseconds.

This controls the speed of fading.

8. }

End of first for loop.

At this point LED has gradually brightened from OFF → ON.

9. for (int i = 255; i >= 0; i--) {

Second for loop.

Starts at brightness 255 and goes down to 0.

👉 This will make the LED fade down.

10. analogWrite(9, i);

Again, set LED brightness based on ‘i’.

11. delay(10);

Small delay for smooth fading.

12. }

End of second loop.

LED has now faded from ON → OFF.

13. }

End of main loop.

Loop will repeat and LED will continue fading up and down continuously.

**SUMMARY**

- Pin 9 is used because it supports PWM.
- analogWrite() sends values from 0–255 to control brightness.
- First loop increases brightness (fade-in).
- Second loop decreases brightness (fade-out).
- LED smoothly fades up and down.

## Button Press Counter

```
int button = 2;
int count = 0;

void setup() {
  Serial.begin(9600);
  pinMode(button, INPUT);
}

void loop() {
  if (digitalRead(button) == HIGH) {
    count++;
    Serial.println(count);
    delay(300);
  }
}
```

**Explanation**

1. int button = 2;

   We store pin number 2 in the variable button.

   👉 The button is connected to digital pin 2.

2. int count = 0;

   This variable will keep track of

   how many times the button is pressed.

   Start value = 0.

3. void setup() {

   Runs only once when Arduino starts.

4. Serial.begin(9600);

   Start Serial Monitor at 9600 baud.

   We will print the value of count on Serial Monitor.

5. pinMode(button, INPUT);

   Set pin 2 as INPUT.

   👉 This allows Arduino to read button state (pressed / not pressed).

6. }

   End of setup.

   Loop (runs repeatedly)

7. if (digitalRead(button) == HIGH) {

Arduino reads the button.

- HIGH = button pressed
- LOW = button not pressed

So this line checks:

👉 *Is the button pressed?*

8. count++;

Increase the count by 1.

Each press adds one to the counter.

Example:

0 → 1 → 2 → 3 → …

9. Serial.println(count);

Print the updated count on Serial Monitor.

10. delay(300);

Wait 300 milliseconds.

WHY?

To avoid multiple counts due to button bouncing

(when you press once, the button signal shakes very fast → this delay prevents double counting)

11. }

End of the if-condition.

12. }

End of loop.

**SUMMARY**

- Pin 2 reads button input.
- A variable count is used to store number of presses.
- When the button is pressed, count increases.
- Updated count is printed on Serial Monitor.
- A delay is added to avoid multiple counts from one press (debouncing).

## Print “Hello”

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("Hello");
  delay(1000);
}
```

**Explanation**

1. void setup() {

    The setup function.

    Runs only one time when Arduino starts.

2. Serial.begin(9600);

    This starts the Serial Communication.

    - 9600 → baud rate (speed of communication)
    - This allows Arduino to send messages to the Serial Monitor on your laptop.

    Without this line, you will NOT see anything on the Serial Monitor.

3. }

    End of setup.

4. void loop() {

    The loop function runs again and again forever.

5. Serial.println("Hello");

    This sends the text “Hello” to the Serial Monitor.

    println means:

    print the word

    go to the next line

    So every time this line runs, a new “Hello” appears.

6. delay(1000);

    Wait for 1000 milliseconds = 1 second.

    This creates a 1-second gap between each “Hello”.

7. }

    End of loop.

    Arduino repeats the loop forever → keeps printing “Hello” every second.

**SUMMARY**

- Serial communication starts at 9600 baud.
- Arduino prints “Hello” on Serial Monitor.
- A delay of 1 second is used between each print.
- Loop repeats, so “Hello” is printed continuously.

## Button-Activated Buzzer

```
int button = 2;
int buzzer = 8;

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

void loop() {
  int state = digitalRead(button);
  digitalWrite(buzzer, state);
}
```

**Explanation**

1. int button = 2;

   We make a variable button and assign value 2.

   👉 The button is connected to digital pin 2.

2. int buzzer = 8;

   We make another variable buzzer and assign value 8.

   👉 The buzzer is connected to digital pin 8.

3. void setup() {

   This runs once when Arduino starts.

4. pinMode(button, INPUT);

   Pin 2 is set as INPUT.

   This means Arduino will read the button (pressed or not pressed).

5. pinMode(buzzer, OUTPUT);

   Pin 8 is set as OUTPUT.

   This means Arduino will control the buzzer (turn ON or OFF).

6. }

   End of setup.

   Loop Section (Repeats Forever)

7. int state = digitalRead(button);

   Arduino checks the button’s state.

   - If button is pressed → value = HIGH (1)
   - If not pressed → value = LOW (0)

We store this value in the variable state.

8. digitalWrite(buzzer, state);

We directly send the button’s value to the buzzer.

If button is pressed → buzzer turns ON

If button is not pressed → buzzer stays OFF

So the buzzer copies the button behavior.

9. }

End of loop.

**SUMMARY**

- Pin 2 is input for button.
- Pin 8 is output for buzzer.
- When button is pressed (HIGH), buzzer turns ON.
- When button is not pressed, buzzer stays OFF.

## Two LEDs ON

```
void setup() {
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
}
```

**Explanation**

1. void setup() {

   Setup section — runs one time when Arduino starts.

2. pinMode(12, OUTPUT);

   Pin 12 is set as OUTPUT.

   👉 This means an LED is connected to pin 12.

3. pinMode(13, OUTPUT);

   Pin 13 is also set as OUTPUT.

   👉 Another LED is connected to pin 13.

4. }

   End of setup.

5. void loop() {

   Loop section — runs again and again forever.

6. digitalWrite(12, HIGH);

   Turn ON the LED connected to pin 12.

   HIGH → Arduino sends 5 volts to that pin.

7. digitalWrite(13, HIGH);

   Turn ON the LED connected to pin 13.

   Again, HIGH → LED gets power and stays ON.

8. }

   End of loop.

   Since the loop only turns both LEDs ON, they stay ON forever.

**SUMMARY**

- Pins 12 and 13 are set as output.
- Both pins are given HIGH output.
- Both LEDs connected to those pins turn ON and stay ON.

### Serial Print Numbers

```
int i = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(i);
  i++;
  delay(1000);
}
```

**Explanation**

1. int i = 0;

   We create a variable i and set its starting value to 0.

   👉 This will be our counter.

2. void setup() {

   Setup section — runs only once.

3. Serial.begin(9600);

   Start Serial communication at 9600 baud.

   This is required to display output on the Serial Monitor.

4. }

   End of setup.

   Loop Section (Runs Continuously)

5. Serial.println(i);

   Print the current value of i on the Serial Monitor.

   println prints the value and moves to a new line.

   Example output:

   ```
   0
   1
   2
   3
   …
   ```

6. i++;

   Increase i by 1.

- After first loop: i becomes 1
- Next loop: i becomes 2
- And so on…

👉 It creates an increasing number sequence.

7. delay(1000);

Wait for 1000 milliseconds = 1 second.

This makes the numbers print one per second.

8. }

End of loop.

The loop repeats → numbers keep increasing forever.

**SUMMARY**

- A counter variable i starts from 0.
- Serial communication begins at 9600 baud.
- Arduino prints the value of i every second.
- i++ increments the value after each print.