A. Arduino – Bulb (LED) Blink

Aim:

- To blink an LED (or bulb) using Arduino at fixed time intervals.

Components Required:

- Arduino Uno
- LED / small bulb
- 220Ω resistor
- Jumper wires
- Breadboard (optional)

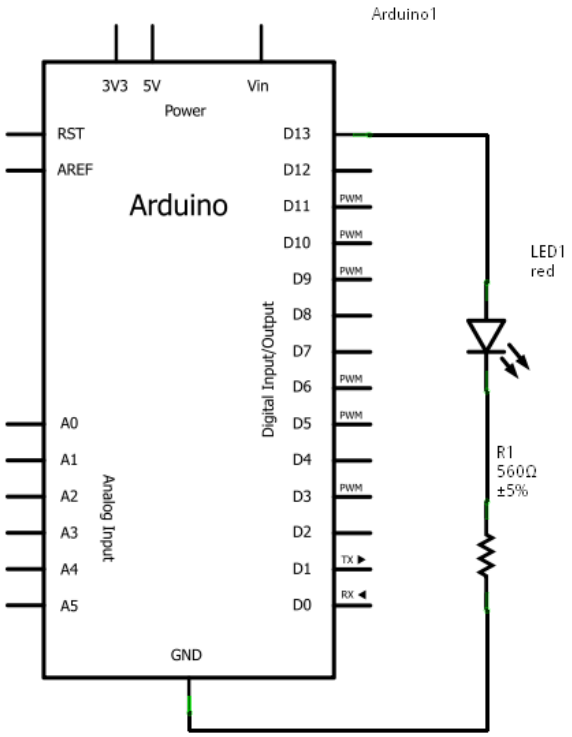
Procedure:

1. Open the Arduino IDE on your computer.
2. Connect the Arduino board to your computer via USB cable.
3. If needed, install Arduino drivers.
4. Launch Arduino IDE.
5. Select your board: **Arduino Uno**.
6. Select the correct **COM/serial port**.
7. Connect the LED:
    - LED anode (+) → Arduino pin 12 through 220Ω resistor
    - LED cathode (–) → GND on Arduino
8. Open a new sketch in Arduino IDE.
9. Write/upload the **LED blink code**:

```
int ledPin = 12;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH); // LED ON
  delay(1000);              // wait 1 second
  digitalWrite(ledPin, LOW);  // LED OFF
  delay(1000);              // wait 1 second
}
```

10. Click **Upload** and wait for the code to compile and upload.
11. Observe the LED blinking at 1-second intervals.

Diagram :



Output / Observation:

- LED blinks ON and OFF repeatedly every 1 second.

Result:

- Successfully blinked an LED using Arduino.

B. Arduino – Temperature Sensor (LM35)

Aim:

- To read temperature from LM35 sensor and display it on Serial Monitor.

Components Required:

- Arduino Uno
- LM35 temperature sensor
- Jumper wires
- Breadboard (optional)

Procedure:

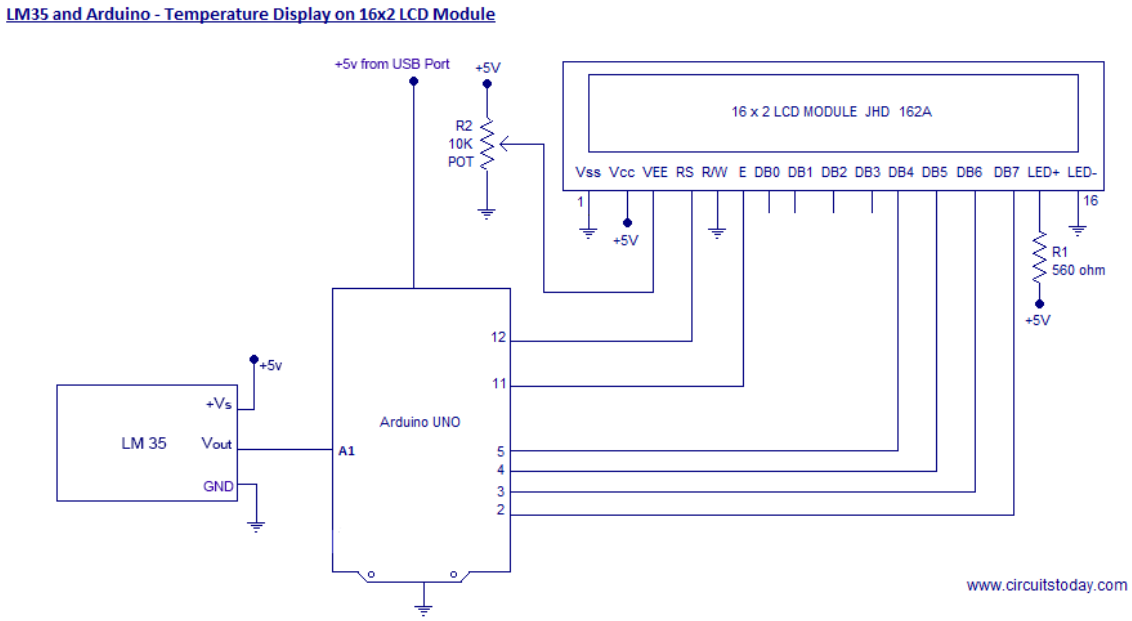
1. Open the Arduino IDE on your computer.
2. Connect the Arduino board to your computer via USB cable.
3. If needed, install Arduino drivers.
4. Launch Arduino IDE.
5. Select your board: **Arduino Uno**.

6. Select the correct **COM/serial port**.
7. Connect the LM35 sensor:
    - VCC → 5V on Arduino
    - GND → GND on Arduino
    - Output → A0 (analog pin) on Arduino
8. Open a new sketch in Arduino IDE.
9. Write/upload the **temperature reading code**:

```
int tempPin = A0;
float temperature;

void setup() {
 Serial.begin(9600);
}
void loop() {
 int tempReading = analogRead(tempPin);
 temperature = (tempReading * 5.0 * 100.0) / 1024.0; // Convert to Celsius
 Serial.print("Temperature: ");
 Serial.print(temperature);
 Serial.println(" C");
 delay(1000);
}
```

10. Open **Serial Monitor** (baud rate 9600).
11. Observe the temperature readings updating every second.

Diagram :



Output / Observation:

- Serial Monitor displays temperature in Celsius.
- Values update approximately every 1 second.

Result:

- Successfully read and displayed temperature from LM35 using Arduino.

C Raspberry Pi – LED Blink via GPIO

Aim:

- To blink an LED using Raspberry Pi GPIO pins with Python code.

Components Required:

- Raspberry Pi (any model with GPIO pins)
- LED
- 220Ω resistor
- Jumper wires
- Breadboard (optional)

Procedure:

1. Boot the Raspberry Pi and open a terminal window.
2. Connect the LED anode (+) to GPIO17 through a 220Ω resistor.
3. Connect the LED cathode (–) to GND on Raspberry Pi.
4. Open a Python IDE or terminal text editor on Raspberry Pi.
5. Write the following Python code:

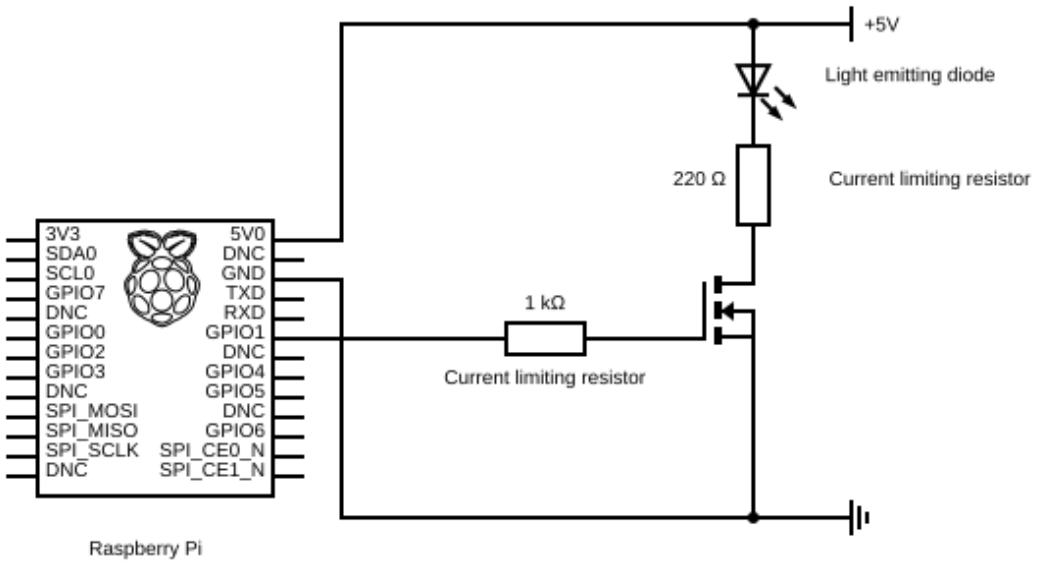
```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

try:
  while True:
    GPIO.output(17, GPIO.HIGH)  # LED ON
    time.sleep(1)
    GPIO.output(17, GPIO.LOW)   # LED OFF
    time.sleep(1)
except KeyboardInterrupt:
  GPIO.cleanup()
```

6. Save the file (e.g., led_blink.py).
7. Run the Python script using:

python3 led_blink.py

8. Observe the LED blinking ON and OFF at 1-second intervals.

Diagram :



Output / Observation:

- LED blinks ON and OFF every 1 second.

Result:

- Successfully controlled an LED using Raspberry Pi GPIO pins.

D.Node MCU

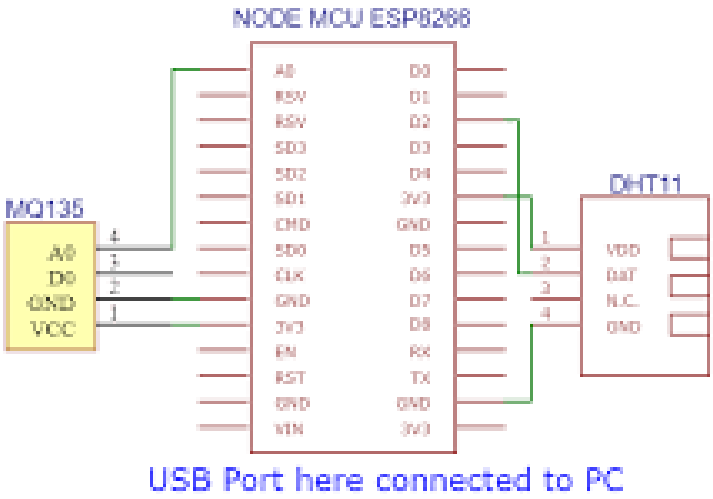
Aim:

To create a local web server using NodeMCU and display sensor readings on the webpage.

Components Required:

- NodeMCU ESP8266
- Temperature sensor (LM35 / DHT11)
- Jumper wires
- Breadboard

Diagram:



Procedure:

1. Connect NodeMCU to the PC.
2. Open Arduino IDE and set board to **NodeMCU 1.0 (ESP-12E)**.
3. Install required libraries (DHT sensor library if using DHT11).
4. Connect the sensor:
    - Sensor VCC → 3.3V
    - GND → GND
    - Output → D5 (GPIO14)
5. Write/upload a web server program to show sensor values.
6. Open Serial Monitor and check IP address.
7. Type IP address in a mobile/PC browser.
8. Read the temperature/humidity values displayed on webpage.

Sample Code (for DHT11):

```
#include <ESP8266WiFi.h>
#include <DHT.h>

#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "YourWiFi";
const char* password = "YourPassword";
```

```
WiFiServer server(80);

void setup() {
 WiFi.begin(ssid, password);
 dht.begin();
 while (WiFi.status() != WL_CONNECTED) delay(500);
 server.begin();
}

void loop() {
 WiFiClient client = server.available();
 if (!client) return;

 float t = dht.readTemperature();
 float h = dht.readHumidity();

 client.println("<html><body>");
 client.println("<h2>Sensor Values</h2>");
 client.print("Temperature: "); client.print(t); client.println(" C<br>");
 client.print("Humidity: "); client.print(h); client.println(" %<br>");
 client.println("</body></html>");
}
```

Output / Observation:

- When the IP address is opened, the webpage displays:
    - Temperature
    - Humidity
    - Values update each refresh

Result:

Successfully created a local Wi-Fi web server using NodeMCU and displayed real-time sensor values.

5. Design and Development of Basic Sensor-Based Detection & Monitoring Systems

AIM

To study, design, and develop different sensor-based systems—Gas Detection, Moisture Detection, Intrusion Detection, and Heartbeat Monitoring—using Arduino/ESP-based microcontrollers.

COMPONENTS REQUIRED (Common)

- Arduino Uno / NodeMCU
- Breadboard
- Jumper wires
- USB Cable
- Power supply

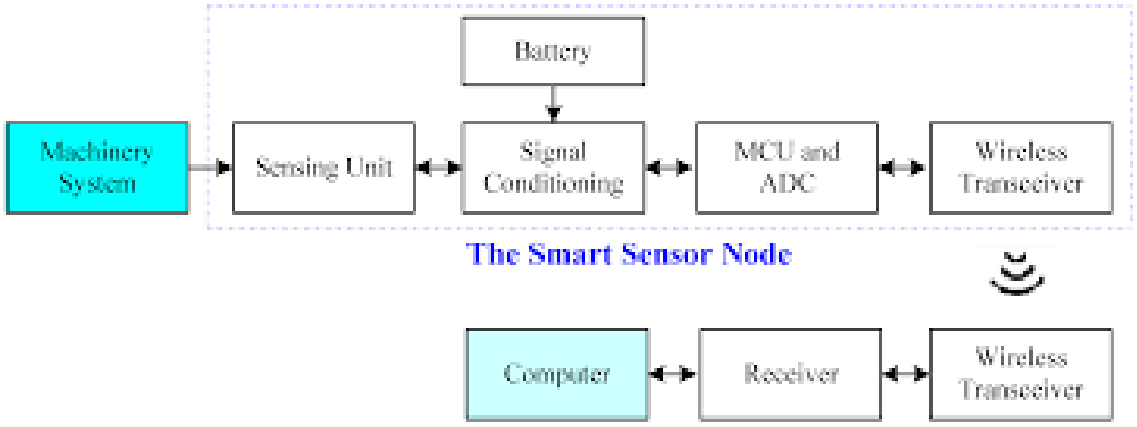
Sensors used:

1. MQ5 Gas Sensor
2. Soil Moisture Sensor
3. PIR Motion Sensor
4. Heartbeat / Pulse Sensor

Outputs:

- LED
- Buzzer
- Serial Monitor / LCD (optional)

Diagram:



THEORY (Combined Explanation for All Sensors)

Sensor-based embedded systems are used to detect physical changes in the environment and convert them into electrical signals for monitoring or alerting purposes.
Each sensor works based on a specific principle:

1. Gas Detection (MQ5 Sensor)

- MQ5 is a gas sensor used to detect LPG, natural gas, and hydrogen.
- Inside the sensor, a small heating element changes resistance when gas concentration increases.

- This resistance change produces an analog voltage indicating gas level.
- If the gas level goes above a threshold, the system triggers an alarm (LED/Buzzer).

---

2. Moisture Detection (Soil Moisture Sensor)

- Works on the **conductivity principle**:
  - Wet soil conducts electricity better → gives LOW analog value
  - Dry soil has poor conductivity → gives HIGH analog value
- Used for irrigation automation.
- If moisture is low, the system can trigger an alert or turn ON a pump.

---

3. Intrusion Detection (PIR Sensor)

- PIR = Passive Infrared Sensor
- Detects the infrared radiation emitted by human bodies.
- When a person moves in front of the sensor, the change in IR radiation triggers a HIGH signal.
- Used in security systems for motion detection.

---

4. Heartbeat Monitoring (Pulse Sensor)

- Works using **photoplethysmography (PPG)**.
- LED shines light onto the skin; blood flow variations reflect different amounts of light.
- The sensor converts this change into pulses.
- The microcontroller reads these pulses and calculates BPM (Beats Per Minute).

BLOCK DIAGRAM (Explain in Viva / Write in Record)

Search on Google:

- "MQ5 Gas Sensor with Arduino Circuit Diagram"
- "Soil Moisture Sensor Circuit Diagram with Arduino"
- "PIR Sensor Arduino Circuit Diagram"
- "Heartbeat Sensor Arduino Circuit Diagram"

PROCEDURE (Common for all 4 Systems)

1. Connect the Arduino/NodeMCU to a laptop using a USB cable.
2. Open the Arduino IDE.
3. Select the correct board and COM port.
4. Make the circuit connections based on the chosen sensor:
   - Sensor VCC → 5V/3.3V
   - Sensor GND → GND

- Sensor Signal → A0/Digital Pins
- LED/Buzzer to digital pin (optional alert).

5. Open the example/typing area and paste the respective program.
6. Verify and upload the code.
7. Open Serial Monitor to observe the output values.
8. Trigger the sensor physically (gas, motion, moisture change, or place finger).
9. Observe behavior of LED/Buzzer/Serial readings.