KARNATAK LAW SOCIETY'S

KLS Gogte Institute of Technology, Belagavi

LABORATORY MANUAL

INTERNET OF THINGS

M. Tech II SEM

(COURSE CODE: MSCSL204)

Department of Computer Science and Engineering

KLS GIT, Belagavi

LAB RULES

Always:

1. **Enter the Lab on time and leave on time.**
2. **Keep the bag outside in the respective racks.**
3. **Utilize the lab hours appropriately.**
4. **If you notice a problem with a piece of equipment (sensors or computer) or the room in general (cooling, heating or lighting) Please report to the lab staff immediately. Do not attempt by yourself.**
5. **Disconnect the devices and other peripherals connected to the computer and shut down the system before leaving the lab.**

Table of Content

DEPARTMENT VISION STATEMENT

To be a center of Excellence for Education, Research and Entrepreneurship in Computer Science and Engineering in creating professionals who are competent to meet emerging challenges to benefit society.

DEPARTMENT MISSION STATEMENT

To impart and strengthen fundamental knowledge of students, enabling them to cultivate professional skills, entrepreneurial and research mindset with right attitude and aptitude.

Karnatak Law Society's
Gogte Institute of Technology, Belagavi
Department of Computer Science & Engineering

Program: M.Tech (Computer Science & Engineering)
Course Title: Internet of Things and Applications Lab
Course Code: MSCSL204

COURSE OUTCOME (COs)	**BLOOM'S LEVEL**
Understand and implement basic I/O operations and sensor interfacing with Arduino UNO and Raspberry Pi.	Un
Apply IoT concepts to design simple real-time applications.	Ap
Analyze sensor data collected from IoT devices for practical applications.	An
PROGRAM OUTCOMES (POs)	**PO NO.**
Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.	**1**
Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.	**2**
Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.	**3**

Introduction to Arduino and Raspberry Pi

Arduino UNO R3:

Arduino is an open-source hardware, software and content platform with a global community.

Specifications:

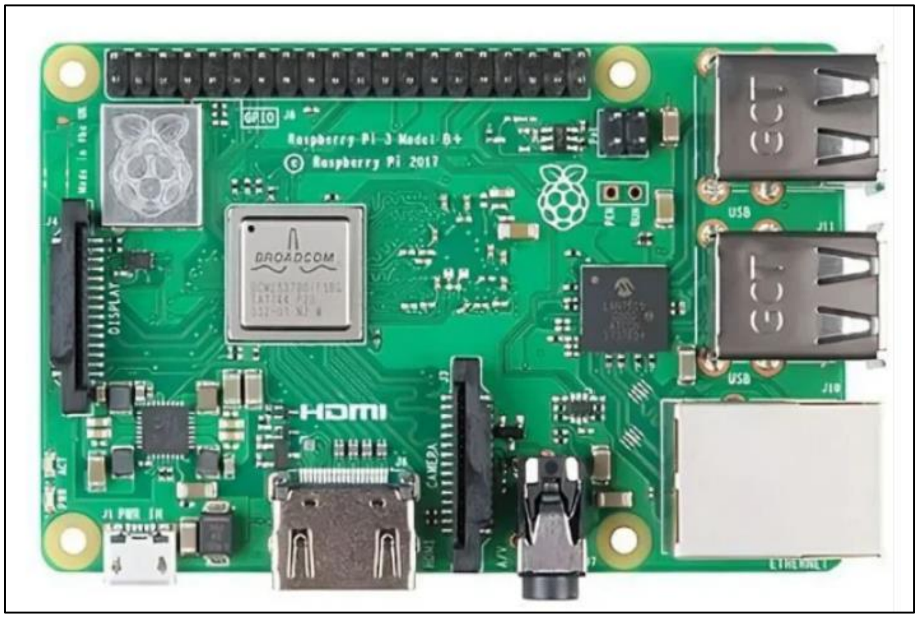
- Micro-controller: ATmega328.
- Operating Voltage: 5V.
- Input Voltage (recommended): 7-12V.
- Input Voltage (limits): 6-20V.
- Digital I/O Pins: 14 (of which 6 provide PWM output).
- Analog Input Pins: 6. • DC Current per I/O Pin: 40 mA.
- DC Current for 3.3V Pin: 50 mA.
- Flash Memory: 32 KB of which 0.5 KB used by boot-loader.
- SRAM: 2 KB (ATmega328).
- EEPROM: 1 KB (ATmega328).
- Clock Speed: 16 MHz

Arduino UNO R3

Raspberry Pi 3 B+:

Specifications:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.
- 100 mbps Base Ethernet.
- 40-pin extended GPIO.
- 4 USB 2.0 ports.
- 4 Pole stereo output and composite video port.
- Full size HDMI.
- CSI camera port for connecting a Raspberry Pi 3 B+ camera.
- DSI display port for connecting a Raspberry Pi 3 B+ touchscreen display.
- Micro SD port for loading your operating system and storing data.
- Upgraded switched Micro USB power source up to 2.5A.



Raspberry Pi 3 B+

Term work 1:

Introduction to Arduino UNO, installing Arduino IDE, basic programming structure, and interfacing an LED.

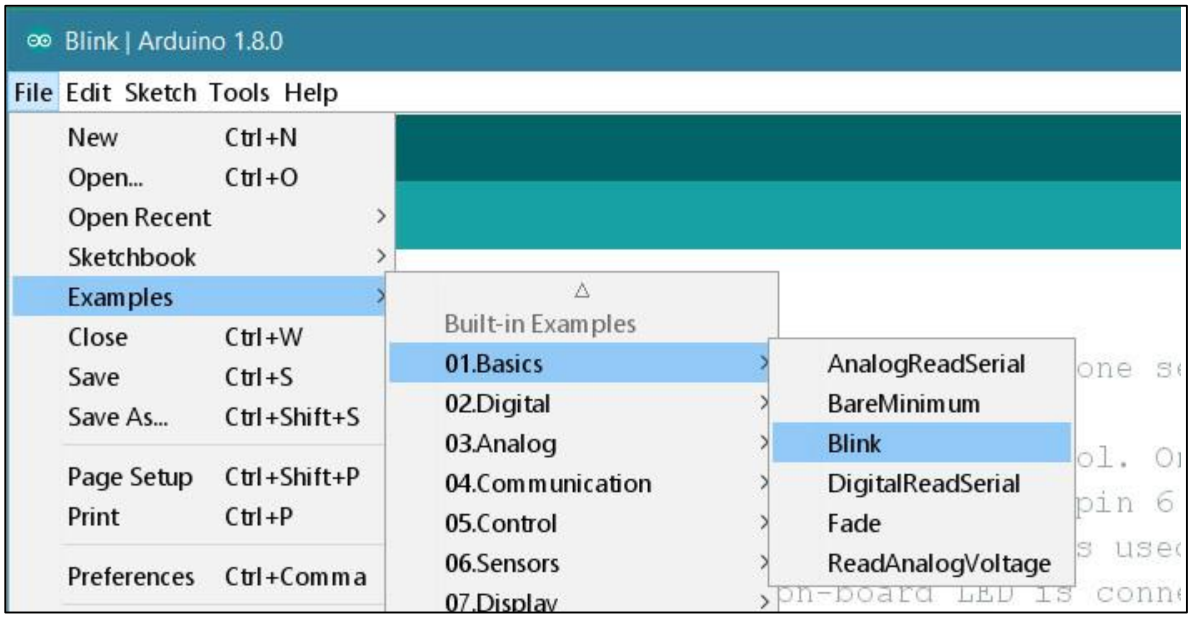
Getting Started with Arduino IDE.

Course Learning Objectives of the Experiment:

1. To understand the Arduino Environment.
2. To blink the built-in LED of Arduino UNO.

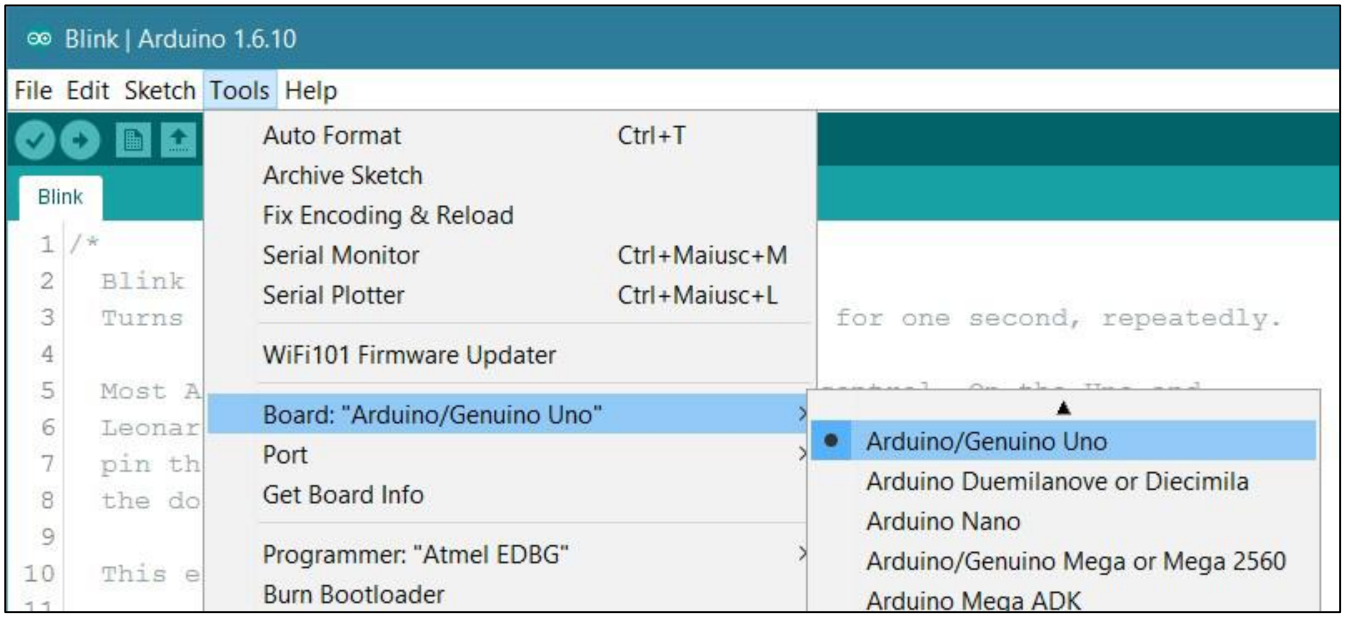
Arduino software IDE on windows:

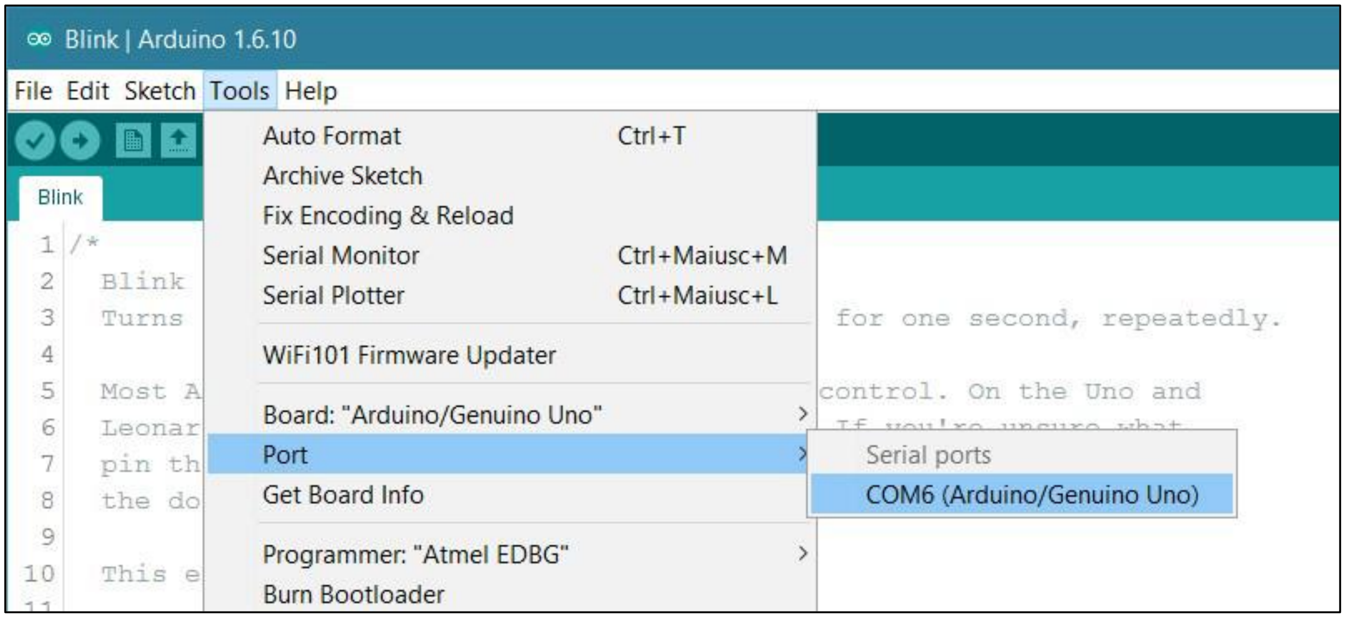
- For installation details, refer to the manual.
- Getting Started with Sketch
    - Open the LED blink example sketch: File > Examples >01.Basics > Blink.



- Select the Board Type and Port
- You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board.

- Select the serial device of the board from the Tools → Serial Port menu.
- This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).
- To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port.





- Upload the Program
- Now, simply click the "Upload" button in the environment. Wait for few seconds you should see the RX and TX LEDs on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.
- A few seconds after the upload finishes, you should see the pin 13 LED on the board starts blinking (in orange).

Course Learning Outcomes:

1. Students will be able to describe the components and functionalities of the Arduino IDE, including the code editor, serial monitor, and the process of uploading sketches to the Arduino board.
2. Students will be able to write, compile, and upload a basic Arduino sketch to blink the built-in LED on the Arduino UNO board, demonstrating their ability to apply programming concepts in a practical context.

Term work 2: Interfacing DHT11 temperature and humidity sensor with Arduino UNO, collecting data and visualizing it on a serial monitor.

Course Learning Objectives of the Experiment:

- Interface DHT11 sensor with Arduino UNO.
- Develop a code for real life applications for the sensor interfaced.

Arduino UNO Overview:

- The Arduino UNO is a microcontroller board based on the ATmega328P. It features 14 digital input/output pins, 6 analogue inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- It can read inputs from sensors and control outputs, such as LEDs, motors, and displays.

DHT11 (Temperature and Humidity Sensor):

- The DHT11 sensor measures temperature and humidity. It provides digital output.
- Common application: Weather monitoring systems, where temperature and humidity data are collected and displayed or processed.

Interfacing and Data Acquisition:

Pin Mode Configuration:

Sensors are connected to the Arduino's digital or analog pins depending on their output type. Digital sensors use digitalRead() or digitalWrite() functions, while analog sensors use analogRead() to acquire data.

Reading Sensor Data:

For analog sensors (e.g., LDR, soil moisture sensor), the `analogRead(pin)` function is used to read the sensor value, which is typically between 0 and 1023.

For digital sensors (e.g., DHT11, ultrasonic sensor), specific libraries or protocols (e.g., pulse timing for ultrasonic) are used to read data.

Processing Data:

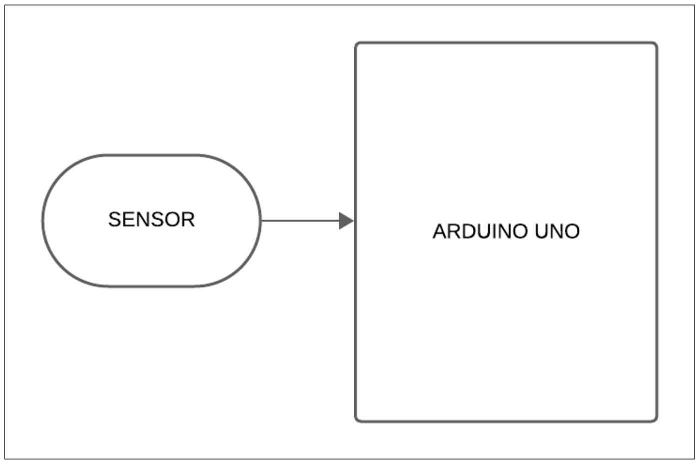
Sensor data is processed in the Arduino sketch to determine the required action. For instance, turning on an LED, sending data to a display, or controlling actuators.

Real-Life Application Implementation:

Weather Monitoring with DHT11:

The DHT11 sensor provides temperature and humidity data, which can be displayed on an LCD or sent to a cloud service for monitoring.

Interfacing Block Diagram:



b. DHT11 for Weather Monitoring

//Connect RM2- RM 19

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 4
void setup(){
  Serial.begin(9600);
}
void loop()
{
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(2000);
}
```

Connection Details:RM2 to RM19

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- Interface DHT11 sensor with Arduino UNO R3.
- Develop code for real life applications with the sensor interfaced.

Term work 3: Interfacing an LDR (Light Dependent Resistor) with Arduino UNO to measure light intensity.

Course Learning Objectives of the Experiment:

- Interface LDR sensor with Arduino UNO.
- Develop a code for real life applications for the sensor interfaced.

Brief Theory:

Interfacing sensors with an Arduino UNO allows for the creation of various practical applications by gathering and processing environmental data. Each sensor serves a specific purpose and helps in monitoring or controlling real-world parameters. This experiment provides hands-on experience with different types of sensors, demonstrating their integration and use in real-life scenarios.

Arduino UNO Overview:

- The Arduino UNO is a microcontroller board based on the ATmega328P. It features 14 digital input/output pins, 6 analogue inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- It can read inputs from sensors and control outputs, such as LEDs, motors, and displays.

Sensors Overview:

LDR (Light Dependent Resistor):

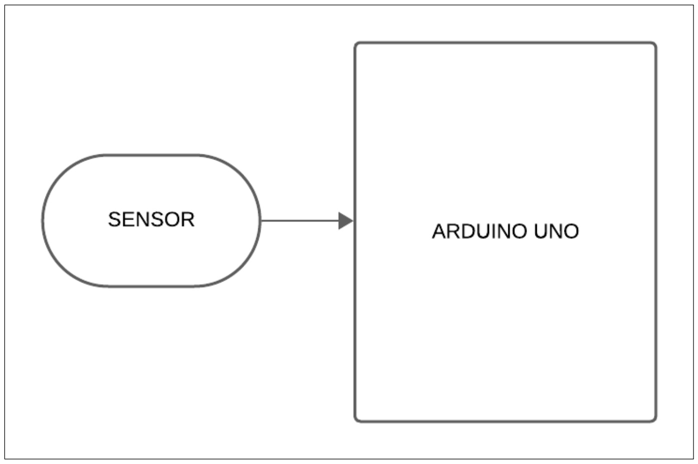
- An LDR is a variable resistor whose resistance decreases with increasing light intensity. It is used to measure light levels.
- Common application: Street light automation, where the LDR senses ambient light and turns street lights on or off accordingly.

Real-Life Application Implementation:

Street Light Automation with LDR:

The LDR's analogue value is read and compared to a threshold to turn on/off a street light (LED).

Interfacing Block Diagram:



Code:

LDR for Street Light

//Connect RM3 to RM20

```
int light_pin = 5;     //Arduino board pin #19 / D5
void setup() {
  pinMode(light_pin, INPUT);
  Serial.begin(9600);
}
void loop() {
  int light_data = digitalRead(light_pin);
  if(light_data)
    Serial.println("Light Not Detected!");
  else
    Serial.println("Light Detected!");
      delay(1000);
}
```

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- Interface LDR sensor with Arduino UNO R3.
- Develop code for real life applications with the sensor interfaced.

Term work 4: Interfacing an ultrasonic sensor with Arduino UNO for distance measurement.

Course Learning Objectives of the Experiment:

- Interface ultrasonic sensor with Arduino UNO.
- Develop a code for real life applications for the sensor interfaced.

Brief Theory:

Interfacing sensors with an Arduino UNO allows for the creation of various practical applications by gathering and processing environmental data. Each sensor serves a specific purpose and helps in monitoring or controlling real-world parameters. This experiment provides hands-on experience with different types of sensors, demonstrating their integration and use in real-life scenarios.

Arduino UNO Overview:

- The Arduino UNO is a microcontroller board based on the ATmega328P. It features 14 digital input/output pins, 6 analogue inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- It can read inputs from sensors and control outputs, such as LEDs, motors, and displays.

Sensors Overview:

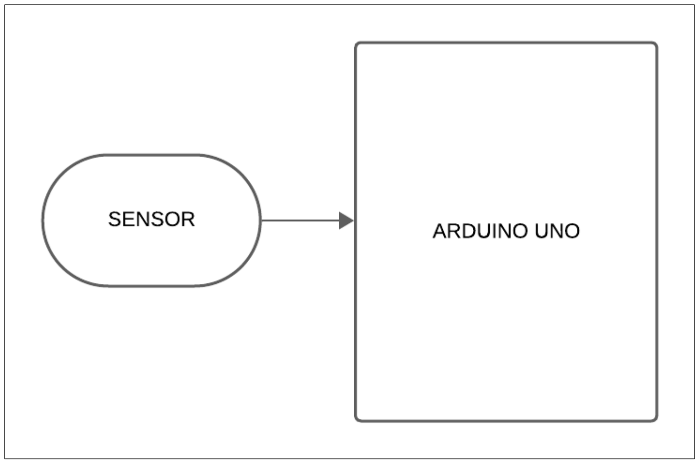
Ultrasonic Sensor (e.g., HC-SR04):

- Ultrasonic sensors measure distance by sending and receiving sound waves. The time taken for the echo to return is used to calculate distance.
- Common application: Obstacle detection in robotics or automotive applications.

Obstacle Detection with Ultrasonic Sensor:

The ultrasonic sensor measures distance, and if an obstacle is detected within a certain range, an action (e.g., stopping a robot) is taken.

Interfacing Block Diagram:



Code:

Ultrasonic Sensor for Obstacle Detection

```
//RM4 and RM21 connected
const int trigPin = 6;   //trig Arduino pin 21/D7   or Board pi 37
const int echoPin = 7;   //Echo Arduino pin 20/D6   or Board pi 35
// defines variables
long duration;
int distance;
void setup() {
   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
   Serial.begin(9600); // Starts the serial communication
}
void loop() {
   // Clears the trigPin
   digitalWrite(trigPin, LOW);
   delayMicroseconds(2);
   // Sets the trigPin on HIGH state for 10 micro seconds
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
   // Reads the echoPin, returns the sound wave travel time in microseconds
   duration = pulseIn(echoPin, HIGH);
   // Calculating the distance
   distance= duration*0.034/2;
   // Prints the distance on the Serial Monitor
   Serial.print("Distance: ");
   Serial.println(distance);
   delay(2000);
}
```

Connection Details:

ULTRASONIC: Connect RM4 and RM21

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- Interface ultrasonic sensor with Arduino UNO R3.
- Develop code for real life applications with the sensor interfaced.

Term work 5: Interfacing a gas sensor with Arduino UNO for air quality monitoring.

Course Learning Objectives of the Experiment:

- Interface gas sensor with Arduino UNO.
- Develop a code for real life applications for the sensor interfaced.

Brief Theory: Interfacing sensors with an Arduino UNO allows for the creation of various practical applications by gathering and processing environmental data. Each sensor serves a specific purpose and helps in monitoring or controlling real-world parameters. This experiment provides hands-on experience with different types of sensors, demonstrating their integration and use in real-life scenarios.

Arduino UNO Overview:

- The Arduino UNO is a microcontroller board based on the ATmega328P. It features 14 digital input/output pins, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- It can read inputs from sensors and control outputs, such as LEDs, motors, and displays.

Sensors Overview:

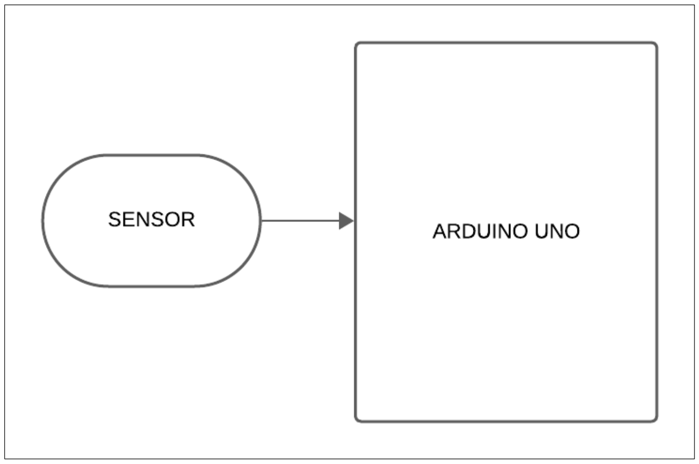
Gas Sensor (e.g., MQ-2):

- Gas sensors detect the presence of gases such as methane, propane, and smoke. They output an analog signal proportional to the gas concentration.
- Common application: Home automation systems for detecting gas leaks and ensuring safety.

Real-Life Application Implementation:

Home Automation with Gas Sensor: The gas sensor's analog value is monitored, and an alarm (buzzer) is triggered if gas concentration exceeds a safe limit.

Interfacing Block Diagram:



Code:

Gas Sensor for Home Automation

```
// RM5 - RM22 connected
int gas_pin = 0;     // Arduino pin #8/A0  or Board pin P11
void setup() {
  pinMode(gas_pin, INPUT);
  Serial.begin(9600);
}
void loop() {
  int gas_value = analogRead(gas_pin);
  Serial.print("Gas Value : ");
  Serial.println(gas_value);
  // Checks if it has reached the threshold value
    delay(1000);
}
```

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- Interface Gas sensor with Arduino UNO R3.
- Develop code for real life applications with the sensor interfaced.

Term work 6: Interfacing a relay with Arduino UNO.

Objectives of the Experiment:

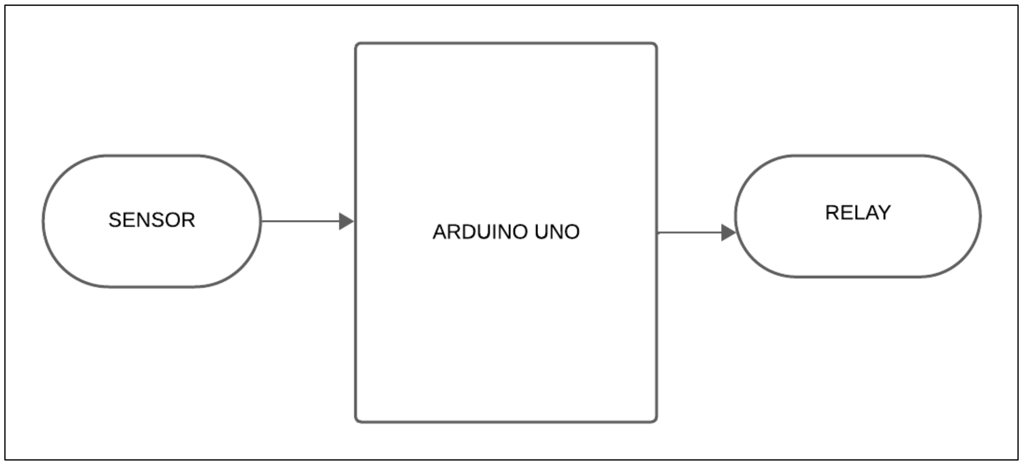
1. To demonstrate the interfacing of Sensor & Relay to Arduino UNO R3.
2. To develop the code to turn ON/OFF the Relay based on Sensor Readings.

Brief Theory:

Relay:

- A relay basically isolates or change the state of an electric circuit from one state to another.
- Relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays.
- Relay are used to control a circuit by a separate low-power signal.

Interfacing block diagram:



Code:

LDR Sensor to control the Relay operation.

```
int light_pin = 5;     //Arduino board pin #19 / D5
int relay=8;
void setup() {
  pinMode(light_pin, INPUT);
  pinMode(relay, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int light_data = digitalRead(light_pin);
  if(light_data)
    Serial.println("Light Not Detected!");
    digitalWrite(relay, HIGH);
  else
    Serial.println("Light Detected!");
    digitalWrite(relay, LOW);
    delay(1000);
}
```

Connection details:

- **Connect RM3 to RM20**
- Connect RM17 - RM9

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- **Interface sensor and relay with Arduino UNO R3.**
- **Develop code for real life applications with the sensor interfaced and relay.**

Term work 7: Interfacing a buzzer with Arduino UNO.

Objectives of the Experiment:

3. To demonstrate the interfacing of Sensor & Buzzer to Arduino UNO R3.
4. To develop the code to turn ON/OFF the Buzzer based on Sensor Readings.

Brief Theory:

Relay:

- A buzzer is an audio signalling device that produces sound when electrical voltage is applied. It is commonly used in embedded systems and electronic circuits to provide audible alerts, alarms, or notifications.

Types of Buzzers

- **Active Buzzer**
  - Contains an internal oscillator.
  - Generates a sound when DC voltage is applied.
  - Easy to interface; just turn ON or OFF using digital signal.
- **Passive Buzzer**
  - Requires an external square wave signal (like from PWM).
  - Allows tone variation (e.g., different musical notes or alerts).
  - Offers more flexibility in sound generation.

Working Principle

- A buzzer operates on the principle of **electromechanical** or **piezoelectric** conversion.
- When voltage is applied, it causes a diaphragm or piezoelectric crystal to vibrate.
- These vibrations produce sound waves in the audible range (typically 2 kHz to 4 kHz).

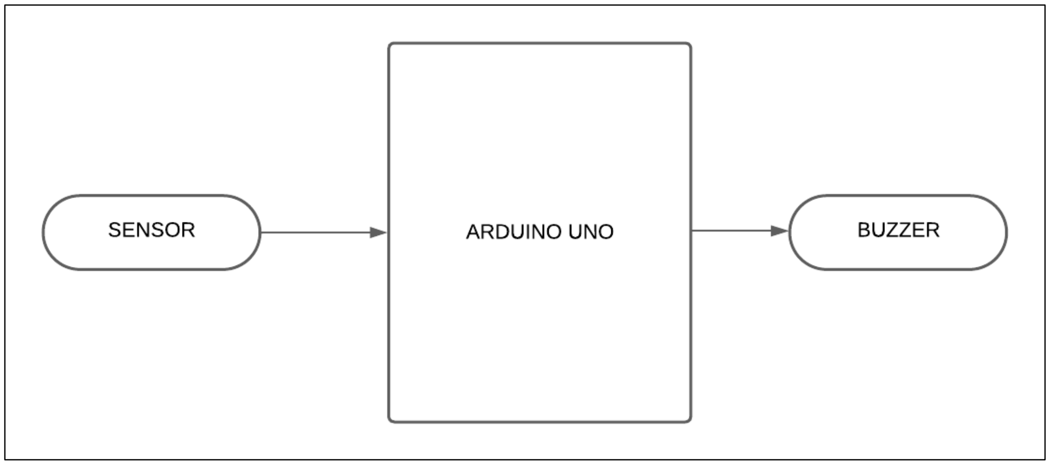
Interfacing with Arduino

- The buzzer is connected to a **digital output pin** and **GND**.
- It is controlled using functions like `digitalWrite ()` for active buzzers or `tone ()` for passive ones.
- Can be triggered based on sensor input or timed events.

Applications

- Alarms and security systems
- Timers and reminders
- Health monitoring alerts (e.g., abnormal heart rate or temperature)
- Notification systems in appliances or industrial control

Interfacing block diagram:



Code:

LDR Sensor to control the Buzzer operation.

```
int light_pin = 5;     //Arduino board pin #19 / D5
int buzzer=9;
void setup() {
  pinMode(light_pin, INPUT);
  pinMode(buzzer, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int light_data = digitalRead(light_pin);
  if(light_data)
    Serial.println("Light Not Detected!");
    digitalWrite(buzzer, HIGH);
  else
    Serial.println("Light Detected!");
    digitalWrite(buzzer, LOW);
    delay(1000);
}
```

Connection details:

- **Connect RM3 to RM20**
- Connect RM17 - RM9

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- **Interface sensor and buzzer with Arduino UNO R3.**
- **Develop code for real life applications with the sensor interfaced and buzzer.**

Term work 8: Basic I/O operations with Raspberry Pi/ ESP32: controlling an LED.

Objectives of the Experiment:

- To understand and implement basic GPIO (General Purpose Input/Output) operations using Raspberry Pi/ESP32.
- To interface and control an LED using digital output pins, demonstrating ON/OFF control through Python (Raspberry Pi) or Arduino IDE (ESP32).
- To develop a simple program that control LED behavior.

Brief Theory:

The **ESP32** is a powerful, low-cost, low-power **system-on-chip (SoC)** microcontroller developed by **Espressif Systems**, widely used in IoT and embedded systems projects.

Key Features:

1. **Dual-core CPU**:
   - Tensilica Xtensa LX6, up to 240 MHz.
   - Can run two tasks in parallel.
2. **Wireless Connectivity**:
   - **Wi-Fi** (802.11 b/g/n) – built-in support.
   - **Bluetooth v4.2 (BLE + Classic)** – ideal for mobile communication.
3. **GPIO Pins**:
   - 30–36 GPIO pins (depending on module) for sensors, actuators, displays, etc.
   - Supports **PWM, ADC, DAC, I2C, SPI, UART**, etc.
4. **Analog and Digital IO**:
   - **12-bit ADC** (Analog-to-Digital Converter).
   - **8-bit DAC** (Digital-to-Analog Converter).
5. **Flash Memory & RAM**:
   - Varies by module (commonly 4MB flash, 520KB SRAM).
   - Can store and run complex applications.
6. **Deep Sleep Mode**:
   - Consumes very low power, suitable for battery-operated devices.

Programming & Development:

- Can be programmed using:
  - **Arduino IDE**
  - **MicroPython**
  - **Espressif's ESP-IDF (C/C++)**
- Compatible with many libraries and sensor modules.

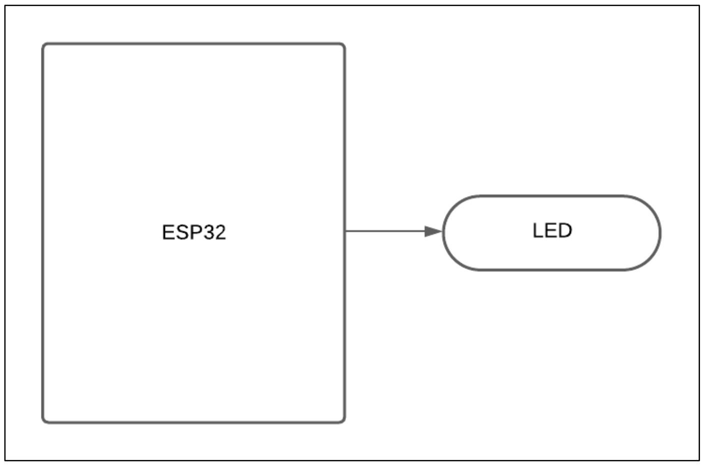
Typical Applications:

- IoT Devices (Smart Home, Agriculture, Wearables)
- Wi-Fi-enabled sensors and controls
- Wireless data logging
- Voice assistants and BLE beacons

Advantages:

- Cost-effective
- Rich in features
- Compact and versatile
- Ideal for both beginners and advanced developers

Interfacing Block Diagram:



Code:

```
int led = 19; // LED pin
void setup() {
pinMode(led, OUTPUT);
}

void loop() {

digitalWrite(led, HIGH); delay(1000);
digitalWrite(led, LOW); delay(1000);
}
```

Connection details:

- **Connect C2 to CN3**

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- **Control an LED using GPIO pins on Raspberry Pi or ESP32.**
- **Understand how to write basic code to perform input/output operations.**

Term work 9: Interfacing DHT11 sensor with Raspberry Pi/ ESP32 and collecting temperature and humidity data.

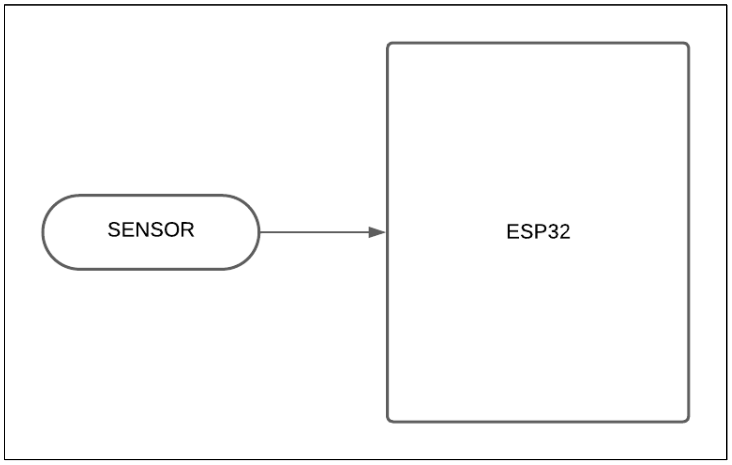
Objectives of the Experiment:

- To interface the DHT11 sensor with Raspberry Pi/ESP32 and accurately read real-time temperature and humidity data using appropriate libraries and GPIO configurations.
- To store or display the collected environmental data on a local interface (LCD, terminal, or web dashboard) for continuous monitoring.
- To analyze and utilize the recorded temperature and humidity data for environment-based decision making or alert generation in smart IoT applications.

Brief Theory:

- Raspberry Pi and ESP32 are widely used microcontrollers in IoT and electronics projects. One of the most fundamental operations is reading data from external devices using GPIO (General Purpose Input/Output) pins.
- The DHT11 sensor, which measures temperature and humidity, is commonly used to test input functionality. By connecting the sensor to a GPIO pin and writing a simple program, we can read environmental data and display it. This demonstrates how digital signals are used to receive information from sensors.
- On the Raspberry Pi, the DHT11 can be interfaced using Python, typically with libraries like Adafruit_DHT. On the ESP32, interfacing is usually done through the Arduino IDE using C/C++ and the DHT.h library.
- Understanding how to read from sensors like the DHT11 is a foundational step toward building smart systems that monitor and respond to real-world conditions—such as activating a fan, sending alerts, or logging data to the cloud.

Interfacing Block Diagram:



Code:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
```

```
#include <DHT.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
#define DHTPIN 32 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(115200);
dht.begin();
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
Serial.println(F("SSD1306 allocation failed"));
for(;;);
}
delay(2000);
display.clearDisplay();
display.setTextColor(WHITE);
}
void loop() {  delay(5000);
float t = dht.readTemperature();
float h = dht.readHumidity();
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
}
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0,0);
display.print("Temperature: ");
display.setTextSize(2);
display.setCursor(0,10);
display.print(t);
display.print(" ");
display.setTextSize(1);
display.cp437(true);
display.write(167);
display.setTextSize(2);
display.print("C");
display.setTextSize(1);
display.setCursor(0, 35);
display.print("Humidity: ");
display.setTextSize(2);
display.setCursor(0, 45);
display.print(h);
display.print(" %");
display.display();
}
```

Connection details:

- **Connect C1 TO CN6**
- **Connect C7 TO CN9**

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- Understand the basic working of the DHT11 temperature and humidity sensor.
- Learn to interface the DHT11 with ESP32/Raspberry Pi using GPIO pins.
- Acquire programming skills to read and display sensor data.
- Understand the role of sensor data in real-time IoT applications.

Term work 10: Interfacing LDR sensor with Raspberry Pi/ESP32 to measure light intensity.

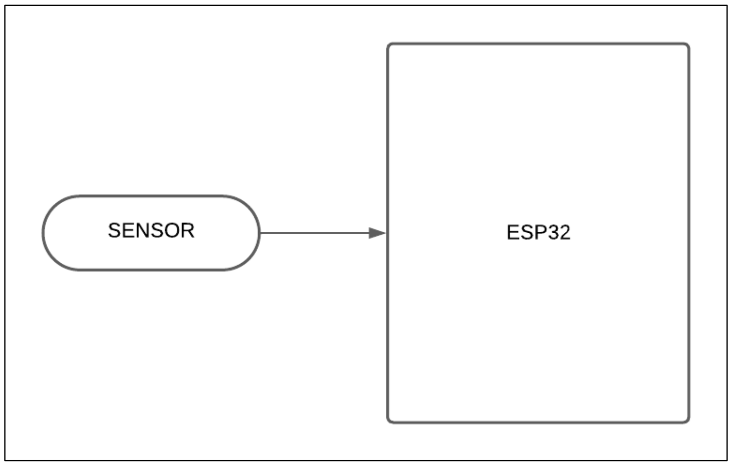
Objectives of the Experiment:

- To understand the working of a digital light sensor and how it measures ambient light intensity (lux).
- To learn how to interface a digital light sensor with Raspberry Pi/ESP32 using I2C communication protocol.
- To write and execute a program that reads lux values from the sensor and displays them.
- To explore the use of digital light sensors in smart lighting, automatic brightness

Brief Theory:

- A **digital light sensor** (e.g., BH1750, TSL2561) measures **ambient light intensity** and provides output in **lux** units. Unlike analog sensors (like LDRs), digital light sensors communicate with microcontrollers using protocols like **I2C or SPI**, offering more accurate and calibrated readings.
- The **ESP32** and **Raspberry Pi** both support I2C communication, making them ideal for interfacing with digital light sensors. These sensors typically have built-in ADCs and return digital lux values directly, simplifying the data acquisition process.
- By connecting the sensor to the I2C pins and using a suitable library in **Arduino IDE (for ESP32)** or **Python (for Raspberry Pi)**, users can read real-time light levels. This setup is commonly used in **smart lighting systems, automatic screen brightness control, and environment monitoring in IoT applications**.

Interfacing Block Diagram:



Code:

```
int light_pin = 32;
void setup() {
pinMode(light_pin, INPUT);
Serial.begin(115200);
}
void loop() {
int light_data = digitalRead(light_pin);
if(light_data==0){
Serial.println("Light Detected!");
}
else{
Serial.println("Light not Detected!");
}
delay(100);
}
```

Connection details:

- **Connect C5 to CN4**

Learning Outcomes of the Experiment:

At the end of the session, students should be able to:

- Understand the working principle and applications of a **digital light sensor** (e.g., BH1750, TSL2561).
- Learn to interface the sensor with **ESP32/Raspberry Pi** using the **I2C communication protocol**.
- Gain hands-on experience in writing code to **read and display light intensity (lux)** values.
- Understand the role of light sensors in **smart and automated IoT systems** such as smart lighting or environmental monitoring.