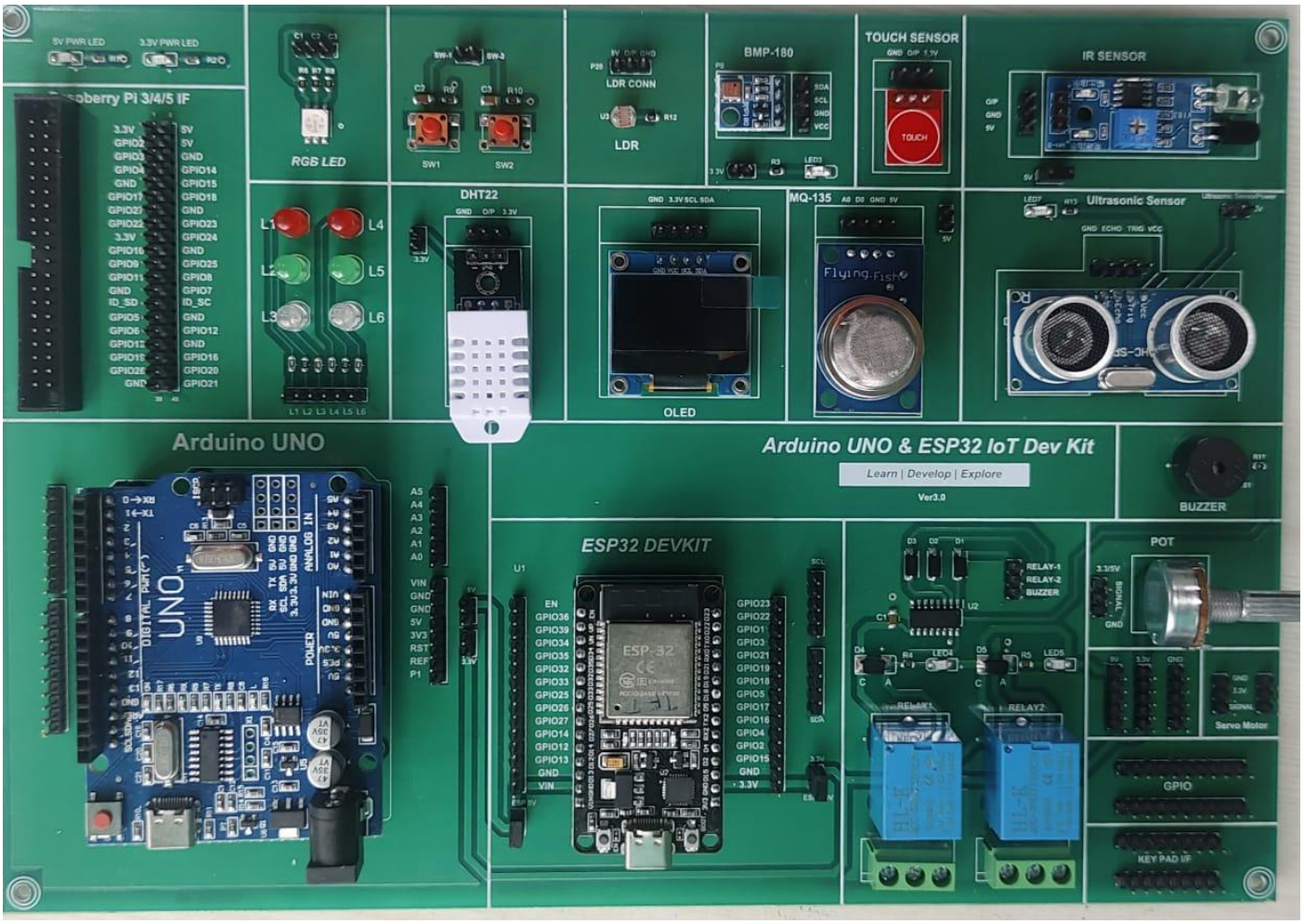
IoT Trainer Kit Manual



Microcontroller: ESP32

Introduction of ESP32:

The ESP32 is a dual-core system with two Harvard Architecture Xtensa LX6 CPUs. All embedded memory, external memory and peripherals are located on the data bus and/or theinstruction bus of these CPUs.

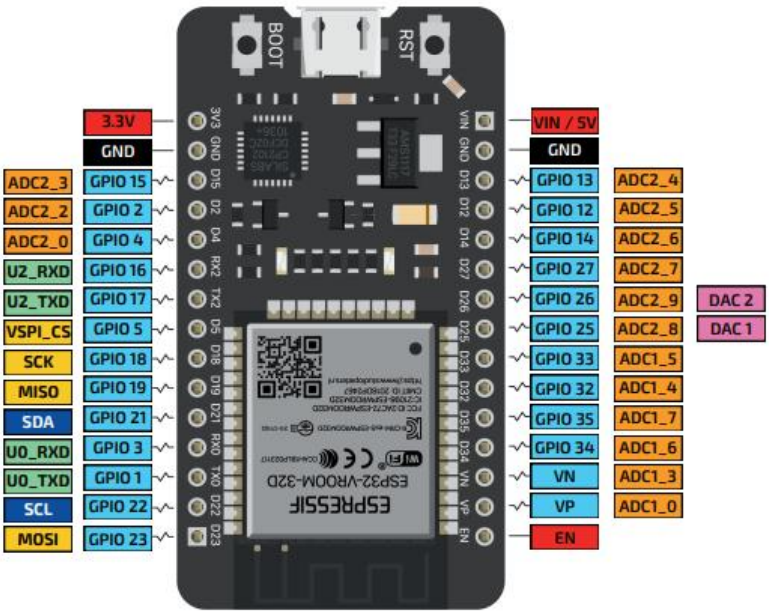
ESP32 can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces. The two CPUs are named “PRO_CPU” and “APP_CPU” (for “protocol” and “application”), however, for most purposes the two CPUs are interchangeable.

GPIO Pins:

The ESP32 peripherals include:

- 18 Analog-to-Digital Converter (ADC) channels
- 3 SPI interfaces
- 3 UART interfaces
- 2 I2C interfaces
- 16 PWM output channels
- 2 Digital-to-Analog Converters (DAC)
- 2 I2S interfaces
- 10 Capacitive sensing GPIOs

The ADC (analog to digital converter) and DAC (digital to analog converter) features are assigned to specific static pins. However, you can decide which pins are UART, I2C, SPI, PWM, etc – you just need to assign them in the code. This is possible due to the ESP32 chip’s multiplexing feature.



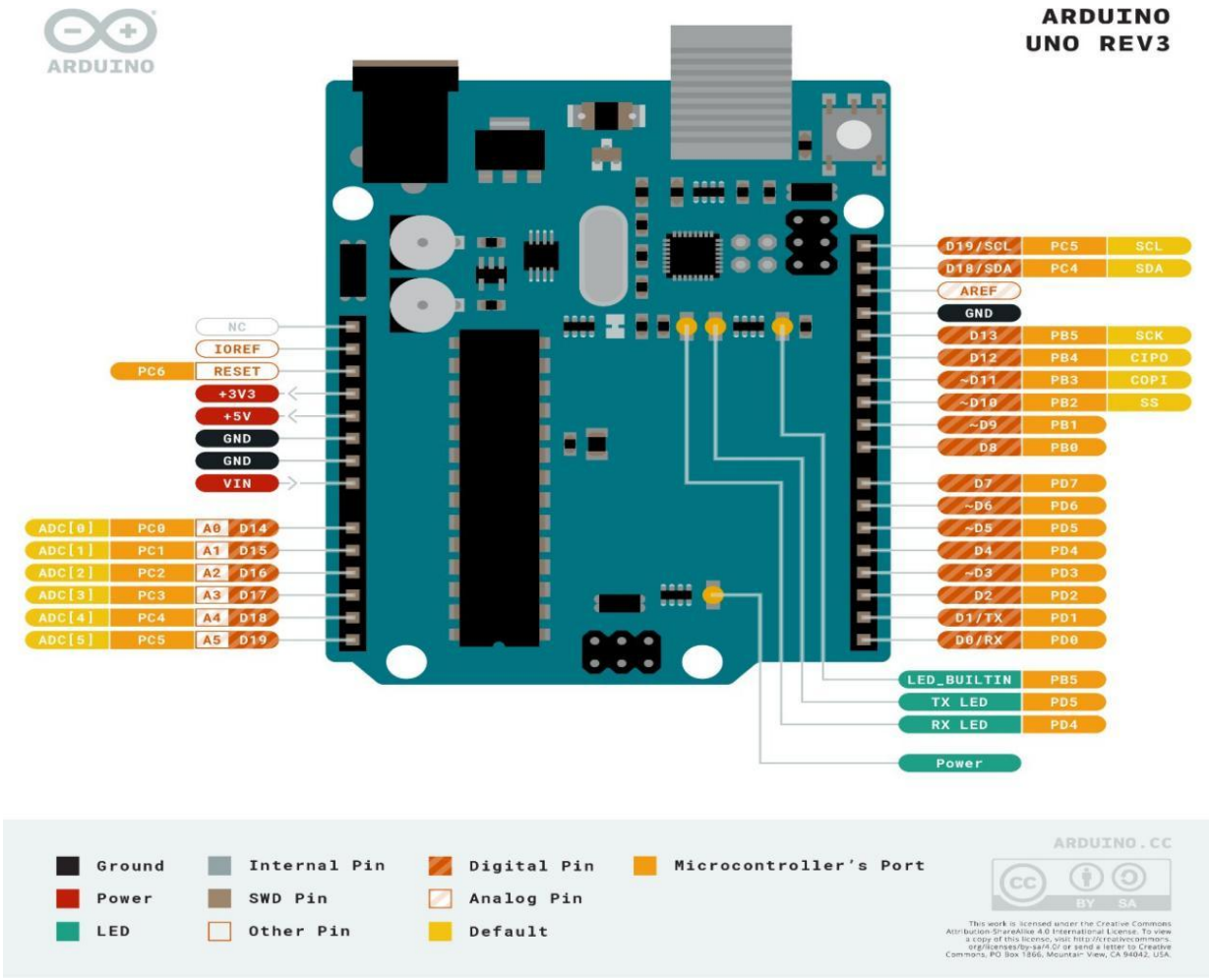
Microcontroller: Arduino UNO

Introduction of Arduino UNO:

The Arduino UNO is the best board to get started with electronics and coding. If this is your first experience tinkering with the platform, the UNO is the most robust board you can start playing with. The UNO is the most used and documented board of the whole Arduino family.

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to getstarted. You can tinker with your UNO withoutworrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

GPIO Pin Outs:



Sensors

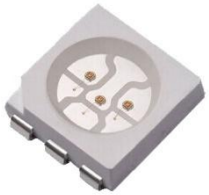
Sensors	Description
	**DHT 22 :** The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.
	**BMP180 Barometric Pressure and Altitude Sensor:** The BMP180 Breakout has been designed to be used in indoor/outdoor navigation, weather forecasting, home automation, and even personal health and wellness monitoring. And to measure barometric pressure, and temperature readings all without taking up too much space.
	**TOUCH PAD:** A touch sensor is a type of sensor that captures and records physical touch or proximity. Provides direct mode i.e. toggle mode by pad option.
	**Ultrasonic Sensor** An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.
	**LDR Sensor:** The LDR is a special type of resistor that works on the photoconductivity principle means that resistance changes according to the intensity of light. Its resistance decreases with an increase in the intensity of light.
	**IR Sensor:** IR sensor is a device that uses infrared technology to detect objects or changes in the environment. IR sensors can detect a wide range of physical properties such as temperature, motion, and proximity.

MQ135 Gas Sensor:

The MQ 135 sensor can be implemented to detect smoke, benzene, vapors, and other hazardous gases. It can detect various harmful gases.

Active Buzzer:

Buzzer meaning electronic component that generates sound through the transmission of electrical signals. Its primary function is to provide an audible alert or notification and typically operates within a voltage range of 5V to 12V.



RGB LED:

SMD RGB LED technology allows the colour of light to be changed by using three diodes: red, blue and green.

Push Button:

A push button switch is a mechanical device used to control an electrical circuit in which the operator manually presses a button to actuate an internal switching mechanism.

10K POT:

This adjustable or Variable resistor are PCB mountable and has 3 terminals. The voltage between the terminal varies as the preset is rotated. The Variable resistors are used for variation voltage as per the need in a circuit.

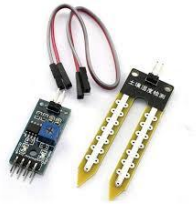
5V Realy:

The primary purpose of a relay is to protect the electrical system from too high of a voltage or current, allowing the safe operation of any equipment it connects to.

OLED :

An organic light-emitting diode (OLED), also known as organic electroluminescent (organic EL) diode is a type of light-emitting diode (LED).

OLEDs enable emissive displays - which means that each pixel is controlled individually and emits its own light (unlike LCDs in which the light comes from a backlighting unit).

	**Soil Mositure:** A **Soil Moisture Sensor** is used to measure the water content in soil. It's often used in gardening, agriculture, and environmental monitoring projects.
	**4x4 Matrix Keypad:** A **4x4 matrix keypad** is a 16-button input device arranged in 4 rows and 4 columns. Each button connects a row to a column when pressed. It's commonly used in embedded systems for entering numbers, passwords, or menu selections.
	**PH Sensor:** A **pH sensor** measures how acidic or basic a solution is, usually in the range of **0 (acidic) to 14 (basic)**. Pure water has a neutral pH of ~7.

List of Experiments:

S.No	Experiment Description
1	Develop a program to blink 5 LEDs back and forth.
2	Develop a program to interface a relay with Arduino board.
3	Develop a program to deploy an intrusion detection system using Ultrasonic and sound sensor.
4	Develop a program to control a DC motor with Arduino board.
5	Develop a program to deploy smart street light system using LDR sensor.
6	Develop a program to classify dry and wet waste with the moisture sensor.
7	Develop a program to read the PH value of a various substances like milk, lime and water.
8	Develop a program to detect the gas leakage in the surrounding environment.
9	Develop a program to demonstrate weather station readings using Arduino.
10	Develop a program to setup a UART protocol and pass a string through the protocol.
11	Develop a program to a water level detection system using Ultrasonic sensor.
12	Develop a program to simulate interfacing with the keypad module to record the keystrokes.

✅ 1. Install Arduino IDE

1. Open your browser and go to:
   👉 https://www.arduino.cc/en/software
2. Download the Arduino IDE for your operating system (Windows, Mac, or Linux).
3. Run the downloaded file and follow the on-screen installation instructions.
4. After installation, open the Arduino IDE.

✅ 2. Install ESP32 Board Manager

1. Open the Arduino IDE.
2. Go to File → Preferences.
3. In the "Additional Board Manager URLs" field, paste this URL:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

4. Click OK.

✅ 3. Add ESP32 Board in Board Manager

1. Go to Tools → Board → Boards Manager.
2. In the search box, type esp32.
3. Select "esp32 by Espressif Systems", and click Install.
4. Wait for installation to complete.

💻 Selecting Your Board for Projects

🔹 For Arduino Uno Projects

1. Go to Tools → Board → Arduino AVR Boards → Arduino Uno.

🔹 For ESP32 Projects (IoT Experiments)

1. Go to Tools → Board → ESP32 Arduino → DOIT ESP32 DEVKIT V1.

✅ 4. Select the Right COM Port

1. Connect your board to your PC using a USB cable.
2. Go to Tools → Port, and select the port labeled (Arduino Uno) or (ESP32).
   - Example: COM3 (Arduino Uno) or COM5 (DOIT ESP32 DEVKIT V1)

Experiment 1: Develop a program to blink 5 LEDs back and forth.

Pin Connections:

LED	ESP32 Pin	Arduino UNO Pin
L1	GPIO 2	D5
L2	GPIO 4	D6
L3	GPIO 5	D7
L4	GPIO 18	D8
L5	GPIO 19	D9

Code:

```
// Define GPIO pins for the LEDs
const int ledPins[] = {2, 4, 5, 18, 19}; // Adjust pins if needed
const int numLeds = sizeof(ledPins) / sizeof(ledPins[0]);


void setup() {
  // Set all LED pins as OUTPUT
  for (int i = 0; i < numLeds; i++) {
    pinMode(ledPins[i], OUTPUT);
    digitalWrite(ledPins[i], LOW); // Turn off all LEDs at start
  }
}
void loop() {
  // Blink LEDs from left to right
  for (int i = 0; i < numLeds; i++) {
    digitalWrite(ledPins[i], HIGH);
    delay(1000);
    digitalWrite(ledPins[i], LOW);
  }
  // Blink LEDs from right to left
  for (int i = numLeds - 2; i > 0; i--) {
    digitalWrite(ledPins[i], HIGH);
    delay(1000);
    digitalWrite(ledPins[i], LOW);
  }
}
```

Experiment 2: Develop a program to interface a relay with Arduino board.

Pin Connections:

Relay	ESP32 Pin	Arduino UNO Pin
Relay 1/2	GPIO 15	D7

Code:

```
// Define the relay control pin
const int relayPin = 7;  // You can use any digital pin


void setup() {
  pinMode(relayPin, OUTPUT);      // Set the relay pin as output
  digitalWrite(relayPin, HIGH);   // Turn relay off initially (assuming active LOW)
}


void loop() {
  digitalWrite(relayPin, LOW);    // Turn ON relay (connects the load)
  delay(1000);                    // Wait for 2 seconds


  digitalWrite(relayPin, HIGH);   // Turn OFF relay (disconnects the load)
  delay(1000);                    // Wait for 2 seconds
}
```

Experiment 3: Develop a program to deploy an intrusion detection system using Ultrasonic and sound sensor.

Pin Connections:

Component		ESP32 Pin	Arduino UNO Pin
Ultrasonic	TRIG	GPIO 13	D9
	ECHO	GPIO 12	D10
Sound Sensor	OUTPUT Pin	GPIO 14	D8
Buzzer	Buzzer pin	GPIO 15	D7

Code:

```
// Pin definitions
const int trigPin = 9;
const int echoPin = 10;
const int buzzerPin = 7;

long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  // --- Ultrasonic sensor distance measurement ---
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // Convert to cm

   // --- Intrusion Detection Condition ---
  if (distance < 50 ) { // Threshold: 50 cm or sound detected
   digitalWrite(buzzerPin, HIGH); // Alarm ON
   Serial.println("Intrusion Detected!");
  } else {
   digitalWrite(buzzerPin, LOW);  // Alarm OFF
  }

  // Print distance for debugging
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(500);
}
```

Experiment 4: Develop a program to control a DC motor with Arduino board.

Pin Connections:

Relay	ESP32 Pin	Arduino UNO Pin

Relay	ESP32 Pin	Arduino UNO Pin
Relay 1/2	GPIO 15	D7
Push Button 1/2	GPIO 4	D2

Motor to Relay:

- One terminal of the motor connects to **+ of power supply.**
- The other motor terminal connects to **COM (Common)** of the relay
- **NO (Normally Open)** of the relay goes to **GND** of power supply

Code :

```
const int relayPin = 7;       // Relay control pin
const int buttonPin = 2;      // Push button pin


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


void loop() {
  int buttonState = digitalRead(buttonPin);
Serial.println(buttonState);
  if (buttonState == 0) {
    digitalWrite(relayPin,HIGH);
  }
  else {
    digitalWrite(relayPin, LOW);
  }
}
```

Experiment 5: Develop a program to deploy smart street light system using LDR sensor.

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LED	GPIO 25	D5
LDR	GPIO 34	A0

Code:

```
const int ldrPin = A0;      // LDR connected to analog pin A0
const int lightPin = 5;     // LED or relay control pin
int lightThreshold = 500;   // Adjust threshold based on environment

void setup() {
  pinMode(ldrPin, INPUT);
  pinMode(lightPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int ldrValue = analogRead(ldrPin);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);

  if (ldrValue < lightThreshold) {
    digitalWrite(lightPin, HIGH); // Turn ON light (dark environment)
  } else {
    digitalWrite(lightPin, LOW);  // Turn OFF light (bright environment)
  }

  delay(500);
}
```

Experiment 6: Develop a program to classify dry and wet waste with the moisture sensor.

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
Moisture Sensor A0	GPIO 32	A0

Code for Arduino UNO:

```
#define MOISTURE_SENSOR_PIN 34  // Use A0 for Arduino Uno

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("Moisture-Based Waste Classification Started...");
}

void loop() {
  int moistureValue = analogRead(MOISTURE_SENSOR_PIN);
  Serial.print("Moisture Sensor Value: ");
  Serial.println(moistureValue);

  if (moistureValue > 2000) {
    Serial.println("Waste Type: DRY");
  } else {
    Serial.println("Waste Type: WET");
  }

  Serial.println("----------------------");
  delay(2000); // Delay for readability
}
```

Experiment 7: Develop a program to read the PH value of a various substances like milk, lime and water.

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
P0	GPIO 34	A3
VCC	5V	5V
GND	GND	GND

Code:

```
#define PH_SENSOR_PIN 36  // GPIO36 (VP on ESP32)


void setup() {
  Serial.begin(115200);
  analogReadResolution(12); // 12-bit ADC (0–4095)
}


void loop() {
  int adc_value = analogRead(PH_SENSOR_PIN);  // Read raw ADCs
  float voltage = (adc_value * 5.0) / 4095.0;  // Convert ADC to voltage (0–3.3V)


  // Assume pH sensor gives 0V = pH 0 and 3.0V = pH 14 (standard range)
  float pH_value = (voltage / 3.0) * 14.0;  // Linear mapping to pH


  Serial.print("ADC: ");
  Serial.print(adc_value);
  Serial.print(" | Voltage: ");
  Serial.print(voltage, 3);
  Serial.print(" V | Estimated pH: ");
  Serial.println(pH_value, 2);


  delay(2000);  // Read every 2 seconds
}
```

Experiment 8: Develop a program to detect the gas leakage in the surrounding environment

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
MQ135 A0	GPIO 15	A0
Buzzer	GPIO 4	D13

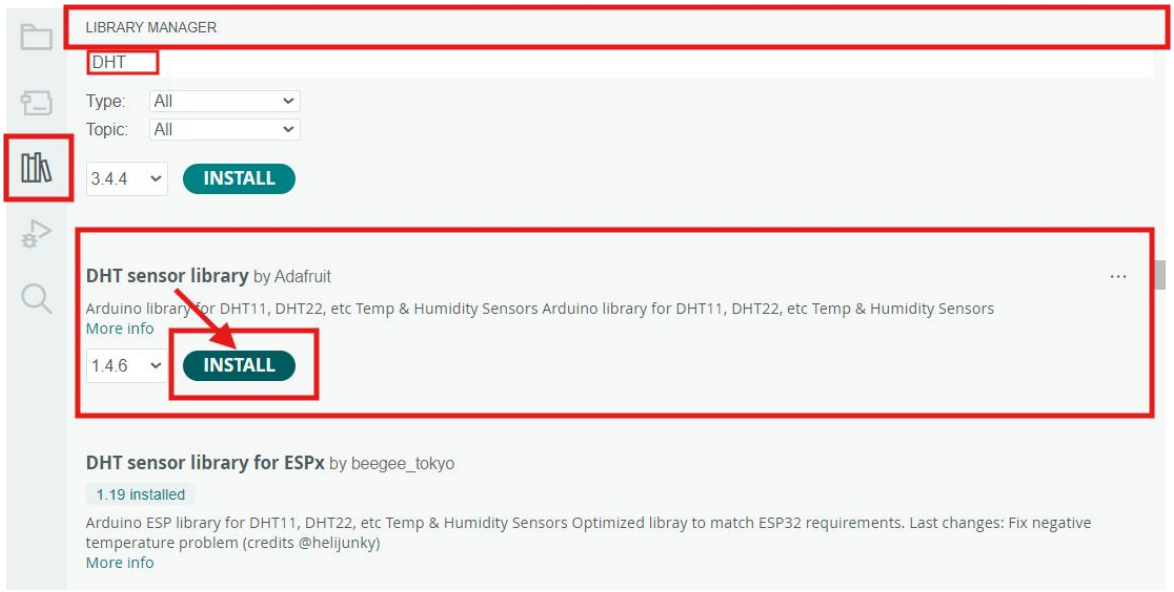
Code:

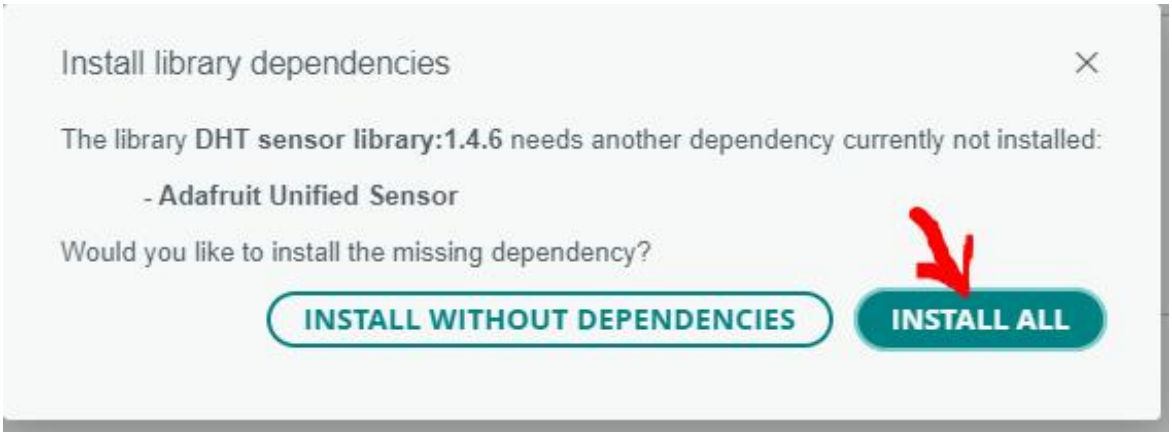
```
const int gasSensorPin = A0;  // Use GPIO34 or GPIO36 if using ESP32 (input-only is fine for analog read)
const int buzzerPin = 13;     // Connect buzzer to digital pin

void setup() {
  Serial.begin(115200);
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  int gasValue = analogRead(gasSensorPin);
  Serial.print("Gas Sensor Value: ");
  Serial.println(gasValue);

  if (gasValue > 300) { // Threshold may vary depending on environment
    digitalWrite(buzzerPin, HIGH);
    Serial.println("Gas leakage detected!");
  } else {
    digitalWrite(buzzerPin, LOW);
  }

  delay(1000);
}
```

Experiment 9: Develop a program to demonstrate weather station readings using Arduino

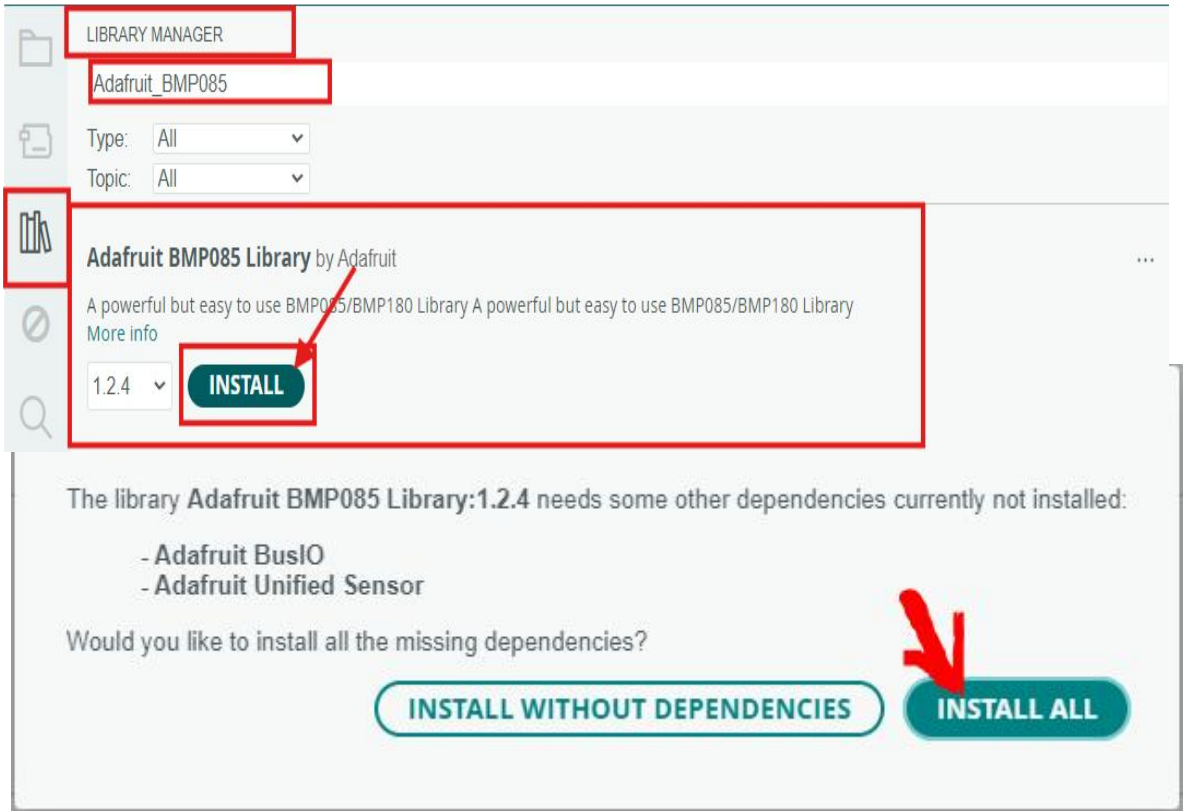
Libraries :

- **Install the DHT library by Adafruit**





- **Install Adafruit_BMP085 Library by Adafruit**



Pin Connections:

Component		ESP32 Pin	Arduino UNO Pin
DHT22 O/P		GPIO 4	D2
BMP180	SDA	GPIO 21	A4
	SCL	GPIO 22	A5

Code:

```
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>  // For BMP180
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP085 bmp;

void setup() {
  Serial.begin(115200);
  dht.begin();
  if (!bmp.begin()) {
    Serial.println("BMP180 not detected.");
    while (1);
  }
}
void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  float pressure = bmp.readPressure() / 100.0;
  Serial.print("Temp: "); Serial.print(temp); Serial.print(" °C, ");
```

```
  Serial.print("Humidity: "); Serial.print(hum); Serial.print(" %, ");
  Serial.print("Pressure: "); Serial.print(pressure); Serial.println(" hPa");
  delay(2000);
}
```

Experiment 10: Develop a program to setup a UART protocol and pass a string through the protocol.

Pin Connections:

Component	ESP32 Pin	Arduino UNO Pin
LED	GPIO 2	D13

Code:

```
#define LED_PIN 2  // GPIO2 for ESP32, use 13 for Arduino Uno
String inputString = "";   // To store incoming serial data
bool stringComplete = false;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("UART Command Processor Initialized.");
  inputString.reserve(200);  // Optional: reserve memory for speed
}

void loop() {
  // When a complete command is received
  if (stringComplete) {
    processCommand(inputString);
    inputString = "";
    stringComplete = false;
  }
}

// Serial Event triggered by new data
void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;

    // Check for end of line
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
```

```
}

void processCommand(String command) {
  command.trim();  // Remove newline or extra whitespace

  if (command.equalsIgnoreCase("LED ON")) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("LED is turned ON");
  }
  else if (command.equalsIgnoreCase("LED OFF")) {
    digitalWrite(LED_PIN, LOW);
    Serial.println("LED is turned OFF");
  }
  else if (command.equalsIgnoreCase("STATUS")) {
    int state = digitalRead(LED_PIN);
    if (state == HIGH) {
      Serial.println("LED STATUS: ON");
    } else {
      Serial.println("LED STATUS: OFF");
    }
  }
  else {
    Serial.print("Unknown Command: ");
    Serial.println(command);
  }
}
```

Experiment 11: Develop a program to a water level detection system using Ultrasonic sensor.

Pin Connections:

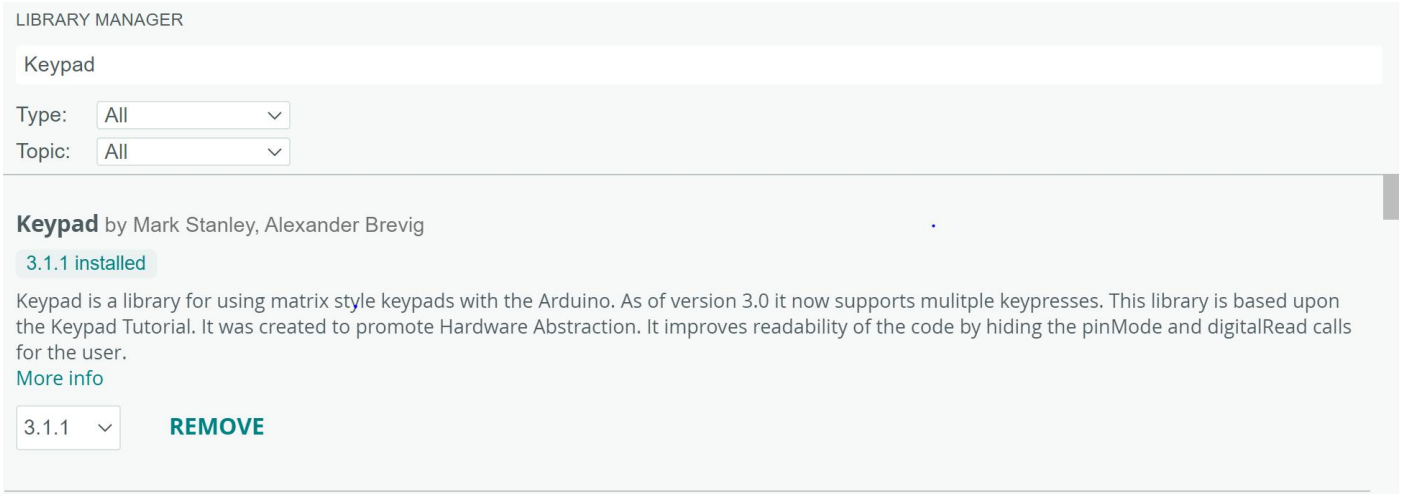
Component		ESP32 Pin	Arduino UNO Pin
Ultrasonic	TRIG	GPIO 5	D9
	ECHO	GPIO 18	D10

Code:

```
const int trigPin = 5;
const int echoPin = 18;


void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
long getDistance() {
  digitalWrite(trigPin, LOW); delayMicroseconds(2);
  digitalWrite(trigPin, HIGH); delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.034 / 2;
  return distance;
}
void loop() {
  long distance = getDistance();
  Serial.print("Water Level Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(1000);
}
```

Experiment 12: Develop a program to simulate interfacing with the keypad module to record the keystrokes.

Libraries :



Pin Connections:

Component		ESP32 Pin	Arduino UNO Pin
Keypad	R1	GPIO 14	D2
	R2	GPIO 27	D3
	R3	GPIO 26	D4
	R4	GPIO 25	D5
	C1	GPIO 33	D6
	C2	GPIO 32	D7
	C3	GPIO 19	D8
	C4	GPIO 18	D9

Code:

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {14, 27, 26, 25}; // Modify as per your wiring
byte colPins[COLS] = {33, 32, 19, 18}; // Avoid GPIOs 34–39

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(115200);
  Serial.println("Keypad Ready");
}
```

```
void loop() {
  char key = keypad.getKey();
  if (key) {
    Serial.print("Key Pressed: ");
    Serial.println(key);
  }
}
```