SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Recognized by AICTE and Accredited by NBA, NAAC and an ISO 9001-2008 Certified Institution)
Bengaluru – 562157

DEPARTMENT OF CSE (IOT & CYBERSECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)

IOT LABORATORY

CHOICE BASED CREDIT SYSTEM

BICL504- V Semester B.E

(*Academic Year 2024-25*)

Compiled and Prepared by:
Mrs. Roopa H L
Assistant Professor

Under the Guidance of:
Dr. Savitha Choudhary
Associate Professor
HOD. Dept. of IOT

Department Vision and Mission

VISION

Building a Proficient Internet of Things Engineers with core values to solve real time problems.

MISSION

M1: To offer state-of-the-art Internet of Things education imparting skills for building Cutting-edge and innovative applications with block chain and cyber security.

M2: To inculcate ethically and socially committed Internet of Things professionals through Value-added courses.

M3: Promoting research blend among students and enabling continuous learning

PROGRAM OUTCOMES

PO's	PO Description
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**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.
PO3	**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.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO's	PSO Description
PSO1	Ability to adapt to a rapidly changing environment by learning and employing new programming skills in Cyber Security and Block Chain Technologies
PSO2	Ability to use diverse knowledge across the domains with inter-personal skills to deliver the industry's needs

IoT Lab		Semester	5
Course Code	**BICL504**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Exam Hours	100
Examination type (SEE)	Practical		

Course objectives:

- Learn the fundamental concept of Internet of Things.
- Learn the connections and working of Arduino board.

Sl.NO	Experiments
1	Develop a program to illustrate the working of LED with a push button.
2	Develop a program to illustrate the working of traffic lights for pedestrians.
3	Develop a program for fading the LED.
4	Develop a program to blink 6 LEDs in ODD and Even Fashion.
5	Develop a program to rotate servo motor both in clockwise and anticlockwise direction.
6	Develop a program to simulate the interfacing of LDR with Arduino and control the intensity of LED usingLDR.
7	Develop a program to simulate the working of potentiometer and LED by varying the intensity of LEDusing potentiometer.
8	Develop a program to simulate the working of LCD and print the room temperature value on LCD.
9	Develop a program for scrolling 5 LEDs back and forth.
10	Develop a program to calculate the distance of an object using ultrasonic sensor.
11	Develop a program to detect the collision using infrared sensor.
12	Develop a program to interface temperature sensor to read the room temperature, humidity and heatindex and print the readings on the serial monitor.

Course outcomes (Course Skill Set):

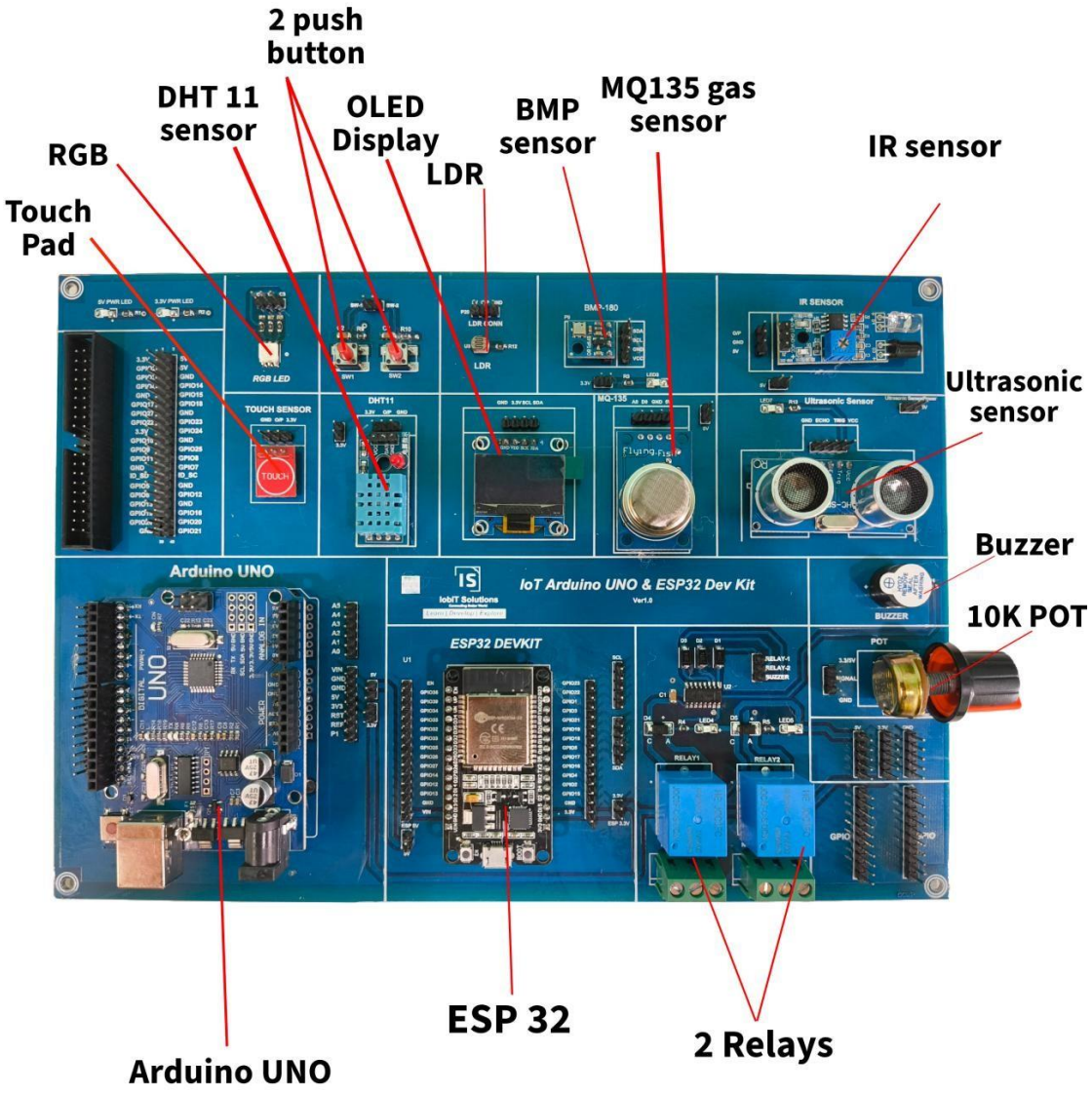
At the end of the course the student will be able to:

- **Design the experiment for a given problem using concepts of IoT.**
- **Develop the solution for the given real world problem using IoT tools and techniques.**
- **Analyze the results and produce substantial written documentation.**

SL NO	INDEX
1	Introduction to Hardware's: • Microcontroller ESP32 • Microcontroller Arduino UNO
2	Sensors
3	Introduction to Software
4	Installing the Microcontroller Board in Arduino IDE
	ESP 32
1	Develop a program to illustrate the working of LED with a pushbutton
2	Develop a program to illustrate the working of traffic lights forpedestrians.
3	Develop a program for fading the LED
4	Develop a program to blink 6 LEDs in ODD and Even Fashion.
5	Develop a program to rotate servo motor both in clockwise andanticlockwise direction.
6	Develop a program to simulate the interfacing of LDR with Arduinoand control the intensity of LED using LDR.
7	Develop a program to simulate the working of potentiometer and LEDby varying the intensity of LED using potentiometer.
8	Develop a program to simulate the working of LCD and print the roomtemperature value on LCD.
9	Develop a program for scrolling 5 LEDs back and forth.
10	Develop a program to calculate the distance of an object usingultrasonic sensor.

11	Develop a program to detect the collision using infrared sensor.
12	Develop a program to interface temperature sensor to read the room Temperature, humidity and heat index and print the readings on theserial monitor.

NOTE: For Arduino Uno programming refer to ESP 32 codes and only the pin number i.e GPIO number to be changed based on your program.



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

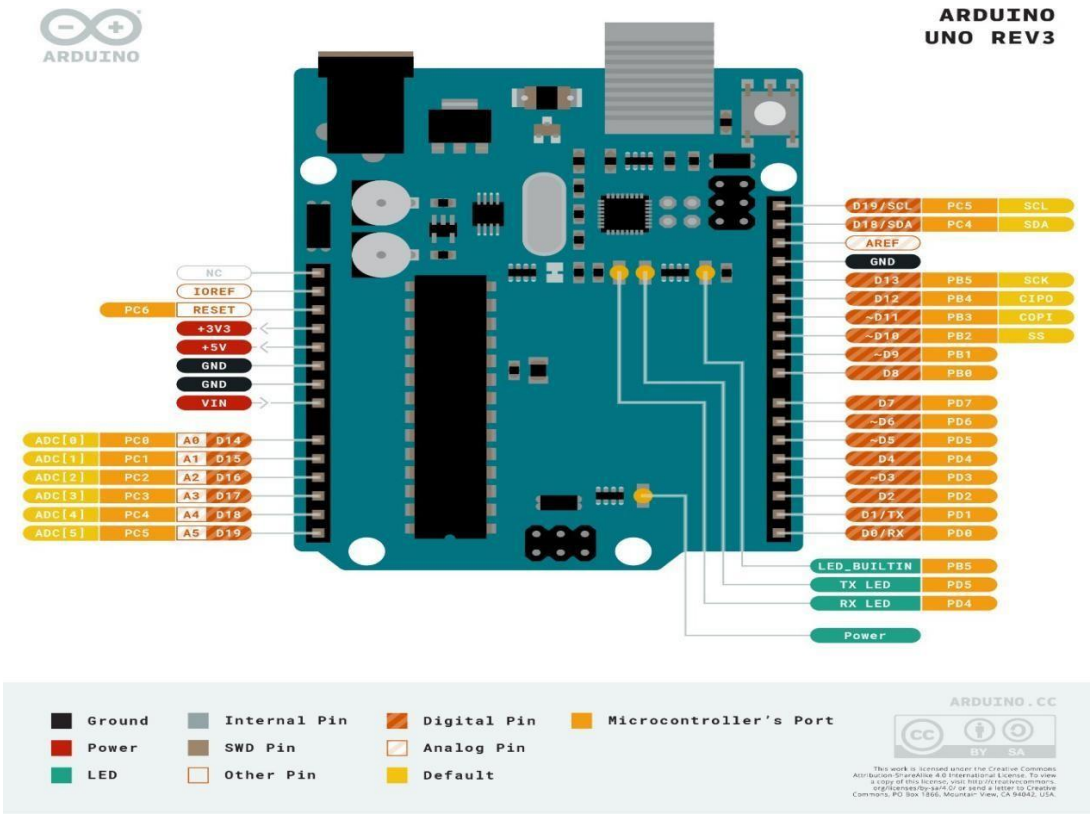
Introduction of Arduino UNO:

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

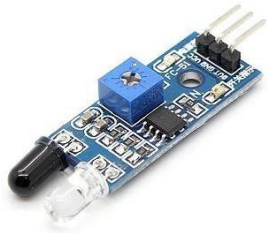
input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC 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 11 :** 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).

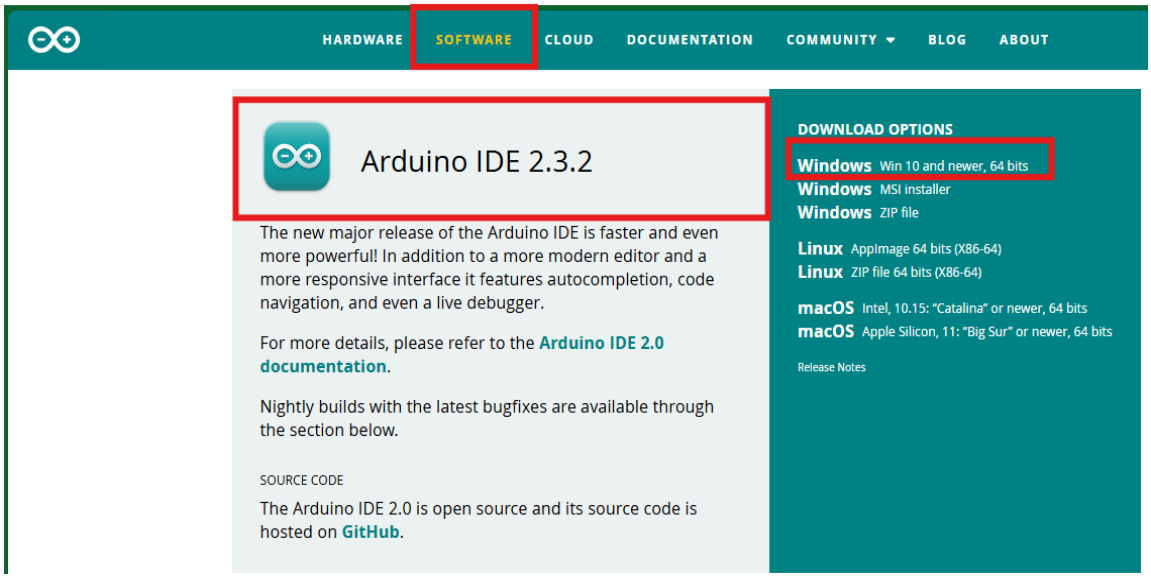
Introduction to Software

Installing Arduino IDE:

The first thing you need is the Arduino IDE. If your computer doesn't have Arduino IDE installed, then visit the official Arduino download page and download the installation file for your preferred operating system

Note: Download the current latest Arduino IDE software from below link.

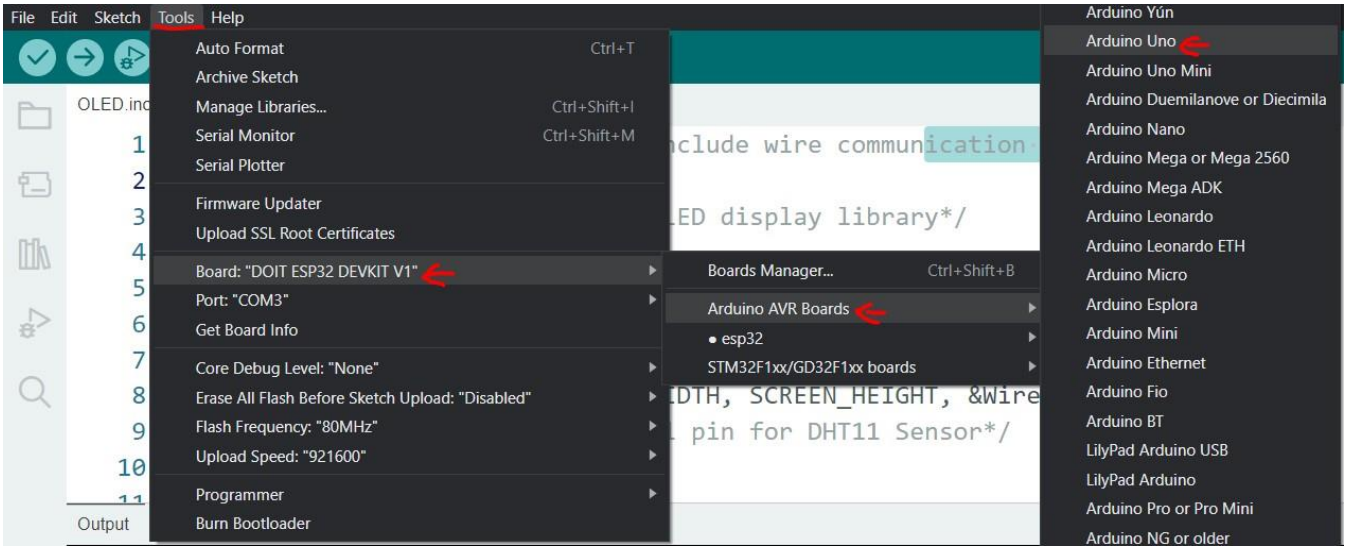
Link:https://www.arduino.cc/en/software



After the installing Arduino IDE

If you are using Arduino UNO then follow the steps that are given below;

- Go to Tools---->Board---->Arduino AVR Board --- >Arduino UNO.
- Select Arduino UNO.
- Select COM port ,then upload the code.



Installing the Microcontroller Board in Arduino IDE

ESP 32

There's an add-on for the Arduino IDE that allows you to program the ESP32 using the Arduino IDE and its programming language. We'll show you how to install the ESP32 board in Arduino IDE.

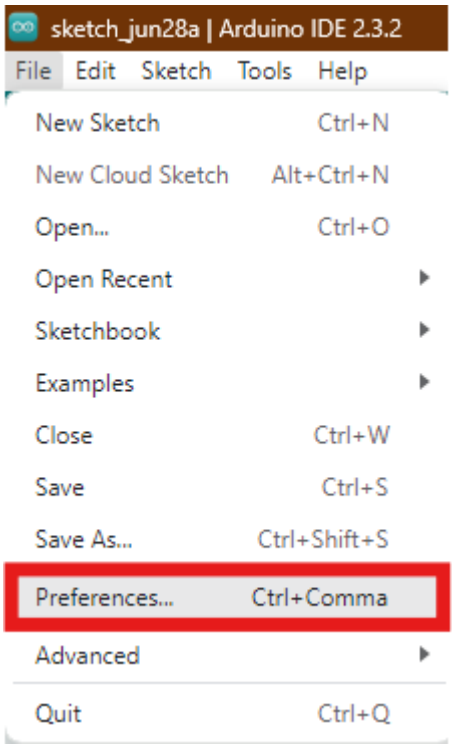
Prerequisites: Arduino IDE Installed

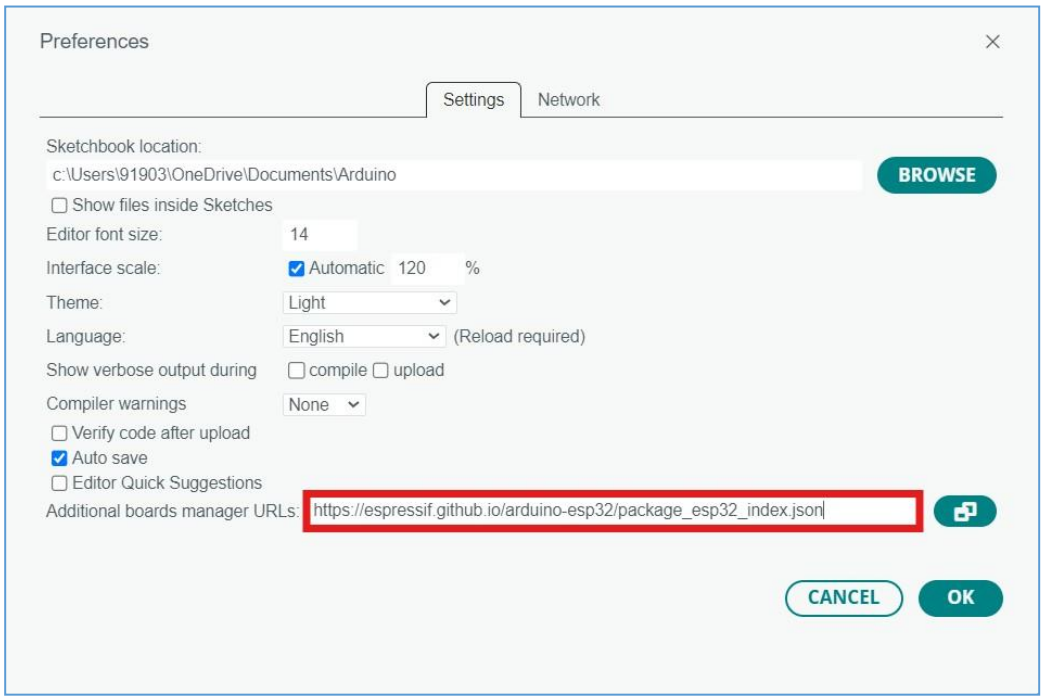
Before starting this installation procedure, make sure you have the latest version of the Arduino IDE installed in your computer. If you don't, uninstall it and install it again. Otherwise, it may not work.

Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

i. In your Arduino IDE, go to File> Preferences

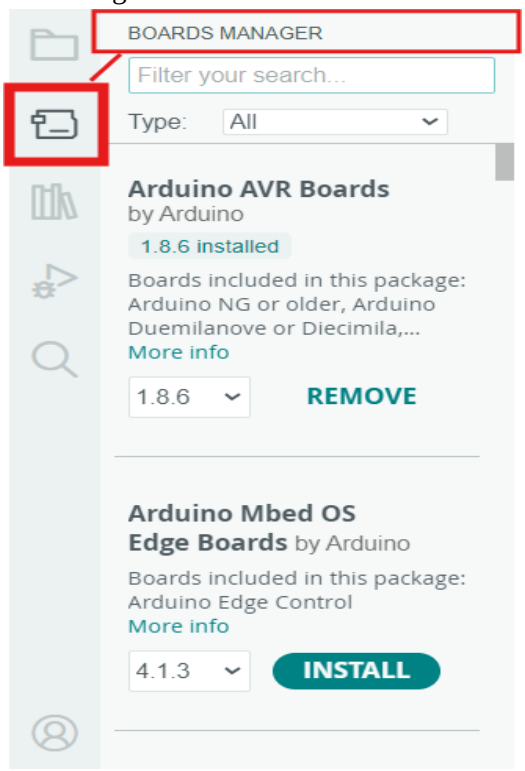




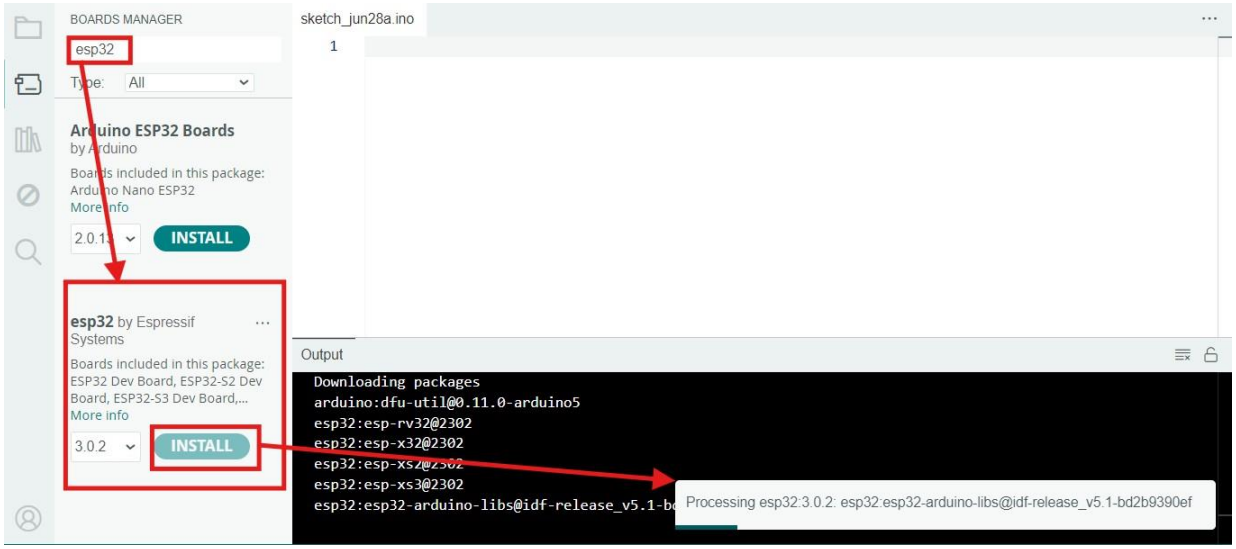
ii. Enter the following into the "Additional Board Manager URLs" field: https://espressif.github.io/arduino-esp32/package_esp32_index.json

Then, click the "OK" button:

iii. Open the Boards Manager.

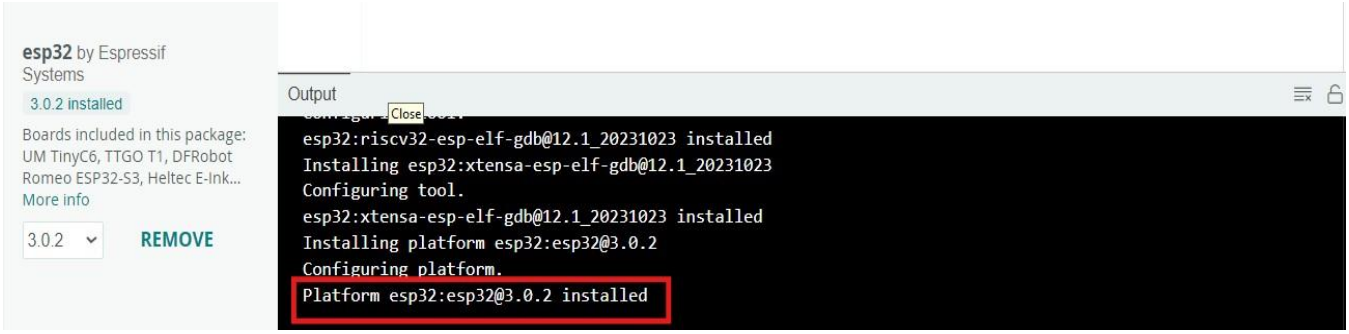


iv. Search for ESP32 and press install button for the "ESP32 by Espressif Systems":



v. That's it. It should be installed after a few seconds

.

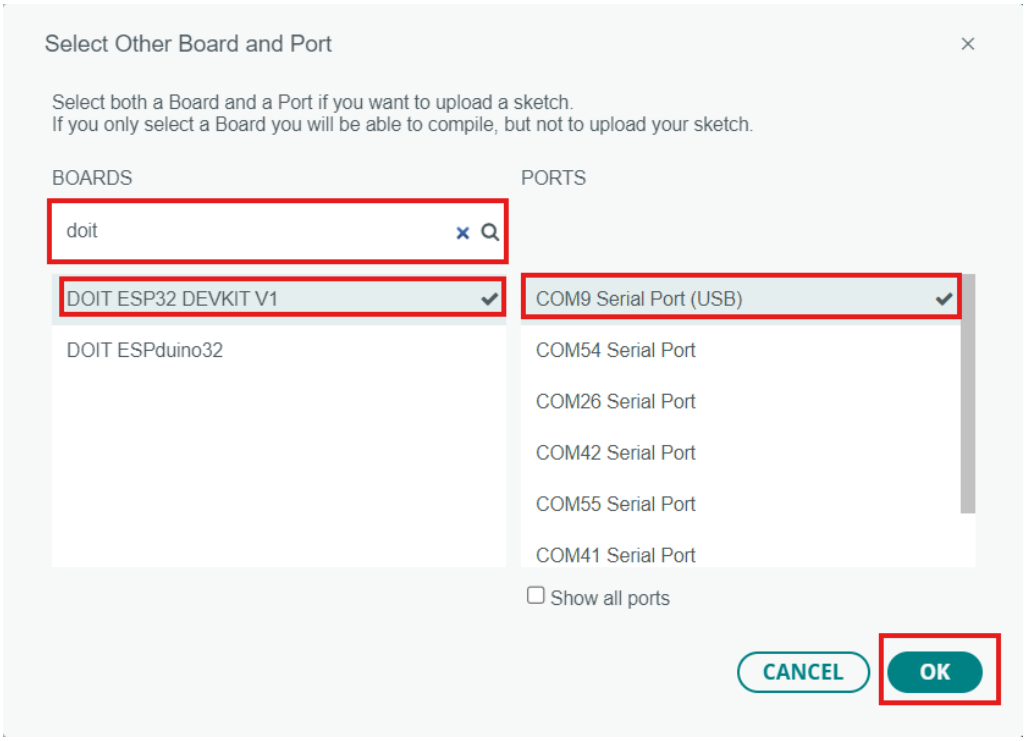


Testing the Installation

Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps:

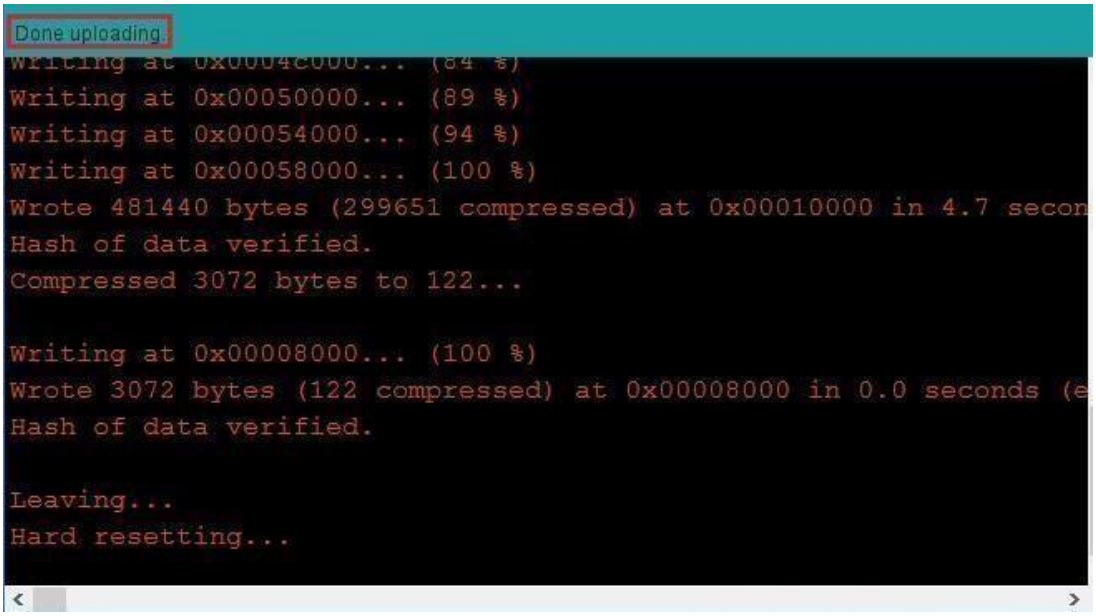
1. Click on "Select other board and port..." (as show in below image) to select the board and port.

2. Select the Port and board (if you don't see the COM Port in your Arduino IDE, you need to install theCP210x USB to UART Bridge VCP Drivers) ,then click on OK .



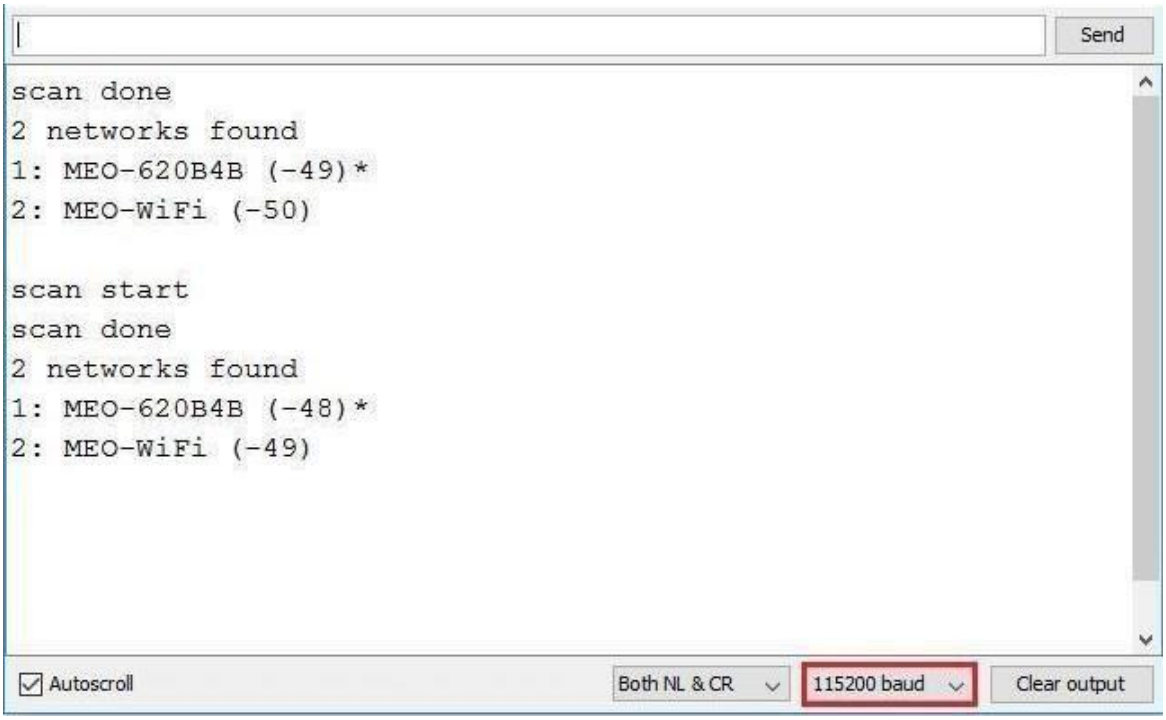
3. Open the following example under File > Examples > WiFi (ESP32) > WiFiScan
4. A new sketch opens in your Arduino IDE
5. Press the Upload button in the Arduino IDE. Wait a few seconds while the code compiles anduploads to your board.

6. If everything went as expected, you should see a "Done uploading." message.



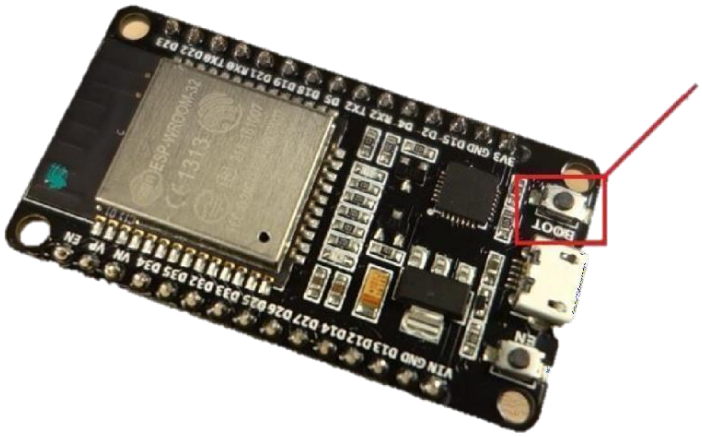
7. Open the Arduino IDE Serial Monitor at a baud rate of 115200

8. Press the ESP32 on-board Enable button and you should see the networks available near your ESP32



Troubleshooting

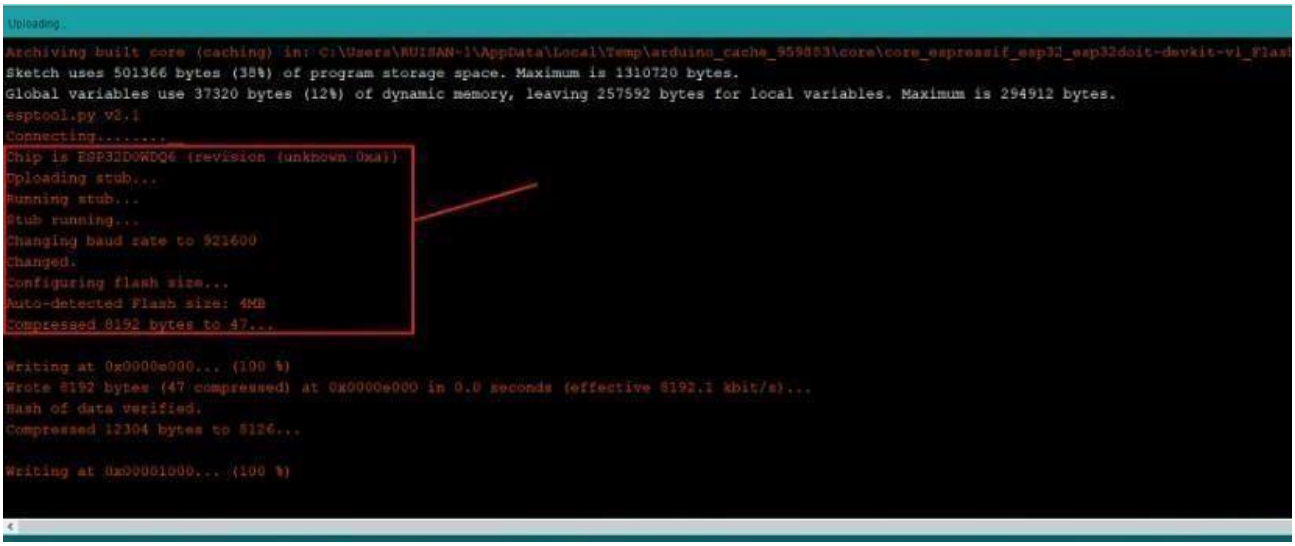
If you try to upload a new sketch to your ESP32 and you get this error message “A fatal error occurred: Failed to connect to ESP32: Timed out… Connecting…“. It means that your ESP32 isnot in flashing/uploading mode.



Having the right board name and COM por selected, follow these steps:

- Hold-down the “BOOT” button in your ESP32 board.
- Press the “Upload” button in the Arduino IDE to upload your sketch:

- After you see the “Connecting….” message in your Arduino IDE, release the finger from the “BOOT” button:



That’s it. Your ESP32 should have the new sketch running. Press the “ENABLE” button to restart the ESP32 and run the new uploaded sketc

Microcontroller: - ESP32

1. Develop a program to illustrate the working of LED with a push button.

PIN CONFIGURATIONS		
		ESP 32
Button	-	GPIO 4
LED		GPIO13

CODE

```
const int buttonPin = 4 // the number of the pushbutton pin

const int ledPin = 13       // the number of the LED pin
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
   // initialize the LED pin as an output:
   pinMode(ledPin,OUTPUT);
   // initialize the pushbutton pin as an input:
   pinMode(buttonPin, INPUT);
 }
 void loop() {
   // read the state of the pushbutton value:
   buttonState = digitalRead(buttonPin);

   // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
   if (buttonState == HIGH) {
     // turn LED on:
      digitalWrite(ledPin, LOW);
   }
   else {
     //          turn          LED          off:
     digitalWrite(ledPin,HIGH);
   }
}
```

2. Develop a program to illustrate the working of traffic lights for pedestrians.

PIN CONFIGURATIONS	
LED	**ESP32**
R1	GPIO25
Y1	GPIO26
G1	GPIO27
R2	GPIO14
G2	GPIO12

CODE

```
// Pin definitions
const int carRed = 25;
const int carYellow = 26;
const int carGreen = 27;
const int pedRed = 14;
const int pedGreen = 12;

void setup() {
  // Set LED pins as output
  pinMode(carRed, OUTPUT);
  pinMode(carYellow, OUTPUT);
  pinMode(carGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);

  // Initialize traffic light (green for cars, red for pedestrians)
  digitalWrite(carGreen, HIGH);
  digitalWrite(pedRed, HIGH);
}

void loop() {
  // Step 1: Green light for cars, red for pedestrians
  digitalWrite(carGreen, HIGH);
  digitalWrite(carYellow, LOW);
  digitalWrite(carRed, LOW);
```

```
digitalWrite(pedGreen, LOW);
  digitalWrite(pedRed, HIGH);
  delay(5000);  // Cars move for 5 seconds

  // Step 2: Yellow light for cars, red for pedestrians
  digitalWrite(carGreen, LOW);
  digitalWrite(carYellow, HIGH);
  delay(2000);  // Yellow light for 2 seconds

  // Step 3: Red light for cars, green for pedestrians
  digitalWrite(carYellow, LOW);
  digitalWrite(carRed, HIGH);
  digitalWrite(pedRed, LOW);
  digitalWrite(pedGreen, HIGH);
  delay(5000);  // Pedestrians cross for 5 seconds

  // Step 4: Flash yellow light for cars before green light
  digitalWrite(pedGreen, LOW);
  digitalWrite(pedRed, HIGH);
  for (int i = 0; i < 3; i++) {
    digitalWrite(carYellow, HIGH);
    delay(500);
    digitalWrite(carYellow, LOW);
    delay(500);
  }
}
```

3. Develop a program for fading the LED.

PIN CONFIGURATIONS	
LED	**ESP32**
RGB	GPIO5

CODE

```
#define ledPin 5    // Pin for the LED (can be changed based on your board)
int brightness = 0;      // Initial brightness level
int fadeAmount = 5;      // Amount by which the brightness will
//increase/decrease

void setup() {
  // Set the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Set the brightness of the LED using PWM
  analogWrite(ledPin, brightness);

  // Change the brightness for the next loop
  brightness = brightness + fadeAmount;

  // Reverse the direction of the fading if brightness reaches max or min
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }

  // Delay for a smoother fading effect
  delay(30);  // Adjust this for faster/slower fading
}
```

4. Develop a program to blink 6 LEDs in ODD and Even Fashion.

PIN CONFIGURATIONS	
LED	**ESP32**
R1	GPIO12
G1	GPIO13
Y2	GPIO14
Y1	GPIO25
R2	GPIO26
G2	GPIO27

CODE

```
// Pin definitions for the odd and even LEDs
#define led1 12   // Odd
#define led3 13   // Odd
#define led5 14   // Odd
#define led2 25   // Even
#define led4 26   // Even
#define led6 27   // Even

void setup() {
  // Set all the LED pins as outputs
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(led6, OUTPUT);
}

void loop() {
  // Turn on all odd-numbered LEDs (1, 3, 5)
  digitalWrite(led1, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led5, HIGH);

  // Turn off all even-numbered LEDs (2, 4, 6)
```

```
  digitalWrite(led2, LOW);
  digitalWrite(led4, LOW);
  digitalWrite(led6, LOW);

  delay(1000);  // Wait for 1 second

  // Turn off all odd-numbered LEDs (1, 3, 5)
  digitalWrite(led1,  LOW);
  digitalWrite(led3,  LOW);
  digitalWrite(led5, LOW);

  // Turn on all even-numbered LEDs (2, 4, 6)
  digitalWrite(led2,  HIGH);
  digitalWrite(led4,  HIGH);
  digitalWrite(led6, HIGH);

  delay(1000);  // Wait for 1 second
}
```

5. Develop a program to rotate servo motor both in clockwise and anticlockwise direction.

PIN CONFIGURATIONS	
Servo Motor	**ESP32**
Red Wire	5V
Brown Wire	GND
Orange Wire	GPIO4

CODE

```
#include <ESP32Servo.h> // Include the ESP32 Servo library

Servo myServo; // Create a Servo object

int pos = 0;    // Variable to store the servo position

void setup() {
  myServo.attach(4);  // Attach the servo to pin 4
}

void loop() {
  // Rotate the servo clockwise (from 0 to 180 degrees)
  for (pos = 0; pos <= 180; pos += 1) {
    myServo.write(pos); // Set the servo to the current position
    delay(15);          // Wait for 15ms for the servo to reach the position
  }

  // Rotate the servo anticlockwise (from 180 back to 0 degrees)
  for (pos = 180; pos >= 0; pos -= 1) {
    myServo.write(pos); // Set the servo to the current position
    delay(15);          // Wait for 15ms for the servo to reach the position
  }
}
```

6. Develop a program to simulate the interfacing of LDR with Arduino and control the intensity of LED using LDR

PIN CONFIGURATIONS	
	ESP 32
LDR	GPIO 15
LED	GPIO 4

CODE

```
#define ldrPin 15     // Pin connected to the LDR
#define ledPin 4      // Pin connected to the LED

int ldrValue = 0;          // Variable to store LDR reading
int ledBrightness = 0;     // Variable to store LED brightness

void setup() {
  pinMode(ledPin, OUTPUT);   // Set the LED pin as output
  Serial.begin(9600);        // Start serial communication for debugging
}

void loop() {
  // Read the value from the LDR (0 to 1023)
  ldrValue = analogRead(ldrPin);

  // Map the LDR value to a range of 0 to 255 for PWM (LED brightness)
  ledBrightness = map(ldrValue, 0, 1023, 0, 255);

  // Set the brightness of the LED using PWM
  analogWrite(ledPin, ledBrightness);

  // Debugging information
  Serial.print("LDR Value: ");
  Serial.print(ldrValue);
  Serial.print(" LED Brightness: ");
  Serial.println(ledBrightness);

  delay(100);  // Short delay for stability
}
```

7. Develop a program to simulate the working of potentiometer and LED by varying the intensity of LED using potentiometer.

PIN CONFIGURATION		
Potentiometer	**ESP32**	**LED**
Vcc	3.3/5 V	
Singal	GPIO13	
GND	GND	
		GPIO4

CODE

```
#define potPin 15     // Pin connected to the PoT
#define ledPin 4      // Pin connected to the LED
int potValue = 0;          // Variable to store PoT reading
int ledBrightness = 0;     // Variable to store LED brightness

void setup() {
  pinMode(ledPin, OUTPUT);   // Set the LED pin as output
  Serial.begin(9600);        // Start serial communication for debugging
}

void loop() {
  // Read the value from the PoT (0 to 1023)
  potValue = analogRead(potPin);
  // Map the PoT value to a range of 0 to 255 for PWM (LED brightness)
  ledBrightness = map(potValue, 0, 1023, 0, 255);
  // Set the brightness of the LED using PWM
  analogWrite(ledPin, ledBrightness);
  // Debugging information
  Serial.print("Pot Value: ");
  Serial.print(potValue);
  Serial.print(" LED Brightness: ");
  Serial.println(ledBrightness);

  delay(100);  // Short delay for stability
}
```

8. Develop a program to simulate the working of LCD and print the room temperature value on LCD.

PIN CONFIGURATION		
OLED	**DHT11**	**ESP32**
SDA		GPIO21
SCL		GPIO22
	O/P	GPIO4

CODE

```
#include <Wire.h>//
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

#define DHTPIN 4 // Pin connected to the DHT11 sensor
#define DHTTYPE DHT11 // Type of DHT sensor
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
-1);

DHT dht(DHTPIN, DHTTYPE);
float t = 0,h = 0;
void setup() {
Serial.begin(9600);
  // Initialize the OLED display
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  }
  dht.begin();
}
void loop() {
  t = dht.readTemperature();
  h = dht.readHumidity();
  display.clearDisplay();
```

```
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  display.print("Temperature = ");
  display.println(t);
  display.print("Humidity = ");
  display.println(h);
  display.display();
  delay(3000);

}
```

9. Develop a program for scrolling 5 LEDs back and forth.

PIN CONFIGURATION	
LED	ESP32
R1	GPIO12
Y1	GPIO13
G1	GPIO14
R2	GPIO25
Y2	GPIO26
G2	GPIO27

CODE

```
// Pin definitions for the 5 LEDs
const int ledPins[] = {12, 13, 14, 25, 26, 27}; // Define LED pins
const int numLeds = 6;                // Total number of LEDs

void setup() {
  // Set all the LED pins as outputs
  for (int i = 0; i < numLeds; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  // Scroll LEDs from left to right (forward)
  for (int i = 0; i < numLeds; i++) {
    digitalWrite(ledPins[i], HIGH);   // Turn on the current LED
    delay(200);                       // Wait for 200ms
    digitalWrite(ledPins[i], LOW);    // Turn off the current LED
  }

  // Scroll LEDs from right to left (backward)
  for (int i = numLeds - 1; i >= 0; i--) {
    digitalWrite(ledPins[i], HIGH);   // Turn on the current LED
    delay(200);                       // Wait for 200ms
    digitalWrite(ledPins[i], LOW);    // Turn off the current LED
  }
}
```

10. Develop a program to calculate the distance of an object using ultrasonic sensor.

PIN CONFIGURATION	
Ultrasonic	**ESP 32**
ECHO	GPIO 21
TRIG	GPIO22

CODE

```
#define TRIG 11
#define ECHO 12

//define sound speed in cm/uS
#define SOUND_SPEED 0.0343
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(9600); // Starts the serial communication
  pinMode(TRIG, OUTPUT); // Sets the trigPin as an Output
  pinMode(ECHO, INPUT); // Sets the echoPin as an Input
}

void loop() {
  // Clears the trigPin
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(TRIG, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(ECHO, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  delay(1000);
}
```

11. Develop a program to detect the collision using infrared sensor.

PIN CONFIGURATION	
IR Sensor	**ESP 32**
O/P Pin	GPIO 15

CODE

```
#define IR 15

void setup() {
  // put your setup code here, to run once:
  pinMode(IR,INPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  int ir = digitalRead(IR);
  // Serial.print(ir);
  // delay(500);
if(ir == 1)
Serial.println("Object Not Detected");
else
Serial.println("Object Detected");
delay(1000);
}
```

12. Develop a program to interface temperature sensor to read the room temperature, humidity and heat index and print the readings on the serial monitor.

PIN CONFIGURATIONS	
DHT11	**ESP 32**
O/P	GPIO 15

CODE

```
#include "DHT.h"
#define DHT11PIN 15

DHT dht(DHT11PIN, DHT11);
void setup()
{
  Serial.begin(9600); /* Start the DHT11 Sensor */
  dht.begin();
}
void loop()
{
  float humi = dht.readHumidity();
  float temp = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" C ");
  Serial.print("Humidity: ");
  Serial.println(humi);
  delay(1000);
}
```