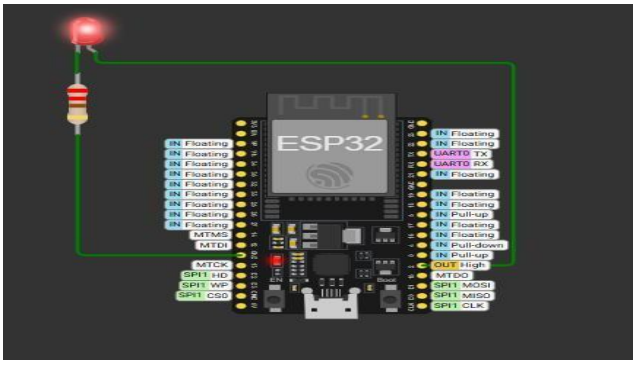
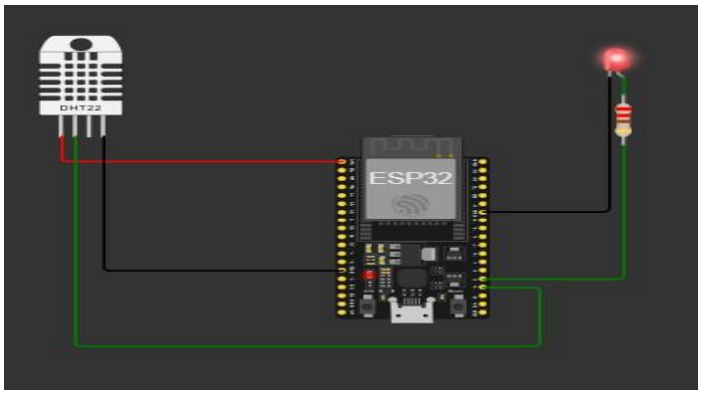
	ITER, SIKSHA 'O' ANUSANDHAN (Deemed to be University)		**Assignment**
Branch	**Computer Science and Engineering,**	Programme	**B.Tech**
Course Name	**Practical Robotics Projects with Arduino**	Semester	**7th**
Course Code	**CSE-4571**	Academic Year	**2025/Odd**
Assignment-1	**Topic- WiFi-Communication**		

Learning Level (LL)	**L1**: Remembering	**L3:** Applying	**L5**: Evaluating
	L2: Understanding	**L4**: Analyzing	**L6**: Creating

To interface an ESP01 Wi-Fi module with an Arduino Uno to establish wireless communication for transmitting real-time sensor data to a remote server or monitoring system. This involves utilizing serial communication between Arduino and ESP01, configuring Wi-Fi connectivity using AT commands, and implementing TCP/IP protocols to enable accurate and reliable wireless data transmission for IoT-based applications.

Q's	Questions	COs	LL
1	What is the primary purpose of the ESP8266 ESP-01 module when interfaced with an Arduino UNO?	**CO1**	**L1 &L2**
2	Why does the ESP-01 module operate at 3.3V instead of 5V, and what could happen if connected directly to 5V?	**CO1**	**L1 &L2**
3	List all 8 pins of the ESP-01 module and briefly describe the function of each.	**CO1**	**L1 &L2**
4	What is the default baud rate of the ESP-01 module, and why is it important to match it with the Arduino serial communication settings?	**CO1**	**L1 &L2**
5	Differentiate between Station (STA) and Access Point (AP) Wi-Fi modes, as well as both (STA+AP) Wi-Fi modes, supported by the ESP-01.	**CO2**	**L2 & L3**
6	Explain the function of the AT commands AT, AT+RST, and AT+CWJAP.	**CO2**	**L2 & L3**
7	Explain the role of the ESP01 Wi-Fi module in an IoT-based Arduino system.	**CO2**	**L2 & L3**
8	Describe how serial communication occurs between the Arduino UNO and the ESP01 module.	**CO2**	**L2 & L3**

9.	Why is a voltage divider circuit or logic level converter required when connecting Arduino TX (5V) to ESP-01 RX (3.3V)?	**CO2**	**L2 & L3**
10	What is the purpose of using the `AT+CWJAP="SSID","PASSWORD"` command during ESP01 configuration.	**CO2**	**L2 & L3**
11.	Using the ESP8266 ESP-01 Wi-Fi module and Arduino UNO, demonstrate how to configure the module in Station (STA) mode to connect to an existing Wi-Fi network.  (a) Draw the circuit diagram showing the connection between Arduino UNO and ESP-01 module for Station mode configuration. (b) Perform a Simulation-based Configuration to connect ESP-01 to a Wi-Fi network and record your observations. (c) Perform Hardware-based Configuration to connect ESP-01 to a Wi-Fi network and capture the serial monitor output.	**CO3**	**L3 & L4**
12	Using the ESP8266 ESP-01 Wi-Fi module and Arduino Uno, demonstrate how to wirelessly transmit sensor data from Arduino to a remote server or monitor through both simulation and hardware implementation.  (a) Draw the circuit diagram showing the connection between Arduino UNO, ESP-01 module, and a sensor for wireless data transmission. (b) Write a short Arduino code snippet to send temperature or humidity data from a DHT11/LM35 sensor through ESP-01 to a remote server. (c) Perform Simulation-based implementation to wirelessly transmit sensor data and record your observations. (d) Perform Hardware-based implementation to demonstrate actual data transmission and capture the serial monitor output or server response. (e) Conclude your observations by discussing how reliable data transmission can be achieved using ESP-01 in IoT applications.	**CO4**	**L4 & L5**
13	Using the ESP8266 ESP-01 Wi-Fi module and Arduino UNO, demonstrate how to implement TCP/IP protocols for client-server communication through hardware-based development (a) How can you configure the ESP-01 module as a TCP client to connect with a remote server? (b) Write the AT commands sequence for establishing a TCP connection and	**CO5**	**L4 & L5**

	transmitting data. (c) Draw the circuit diagram showing the setup for client-server communication between Arduino UNO and ESP-01. (d) Perform Hardware-based development to implement TCP/IP communication and record the serial monitor output.		
14	Explain how ESP-01 Wi-Fi communication contributes to the development of smart IoT systems with an example	**CO6**	**L5 & L6**

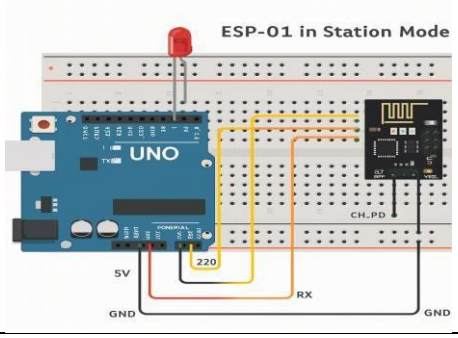
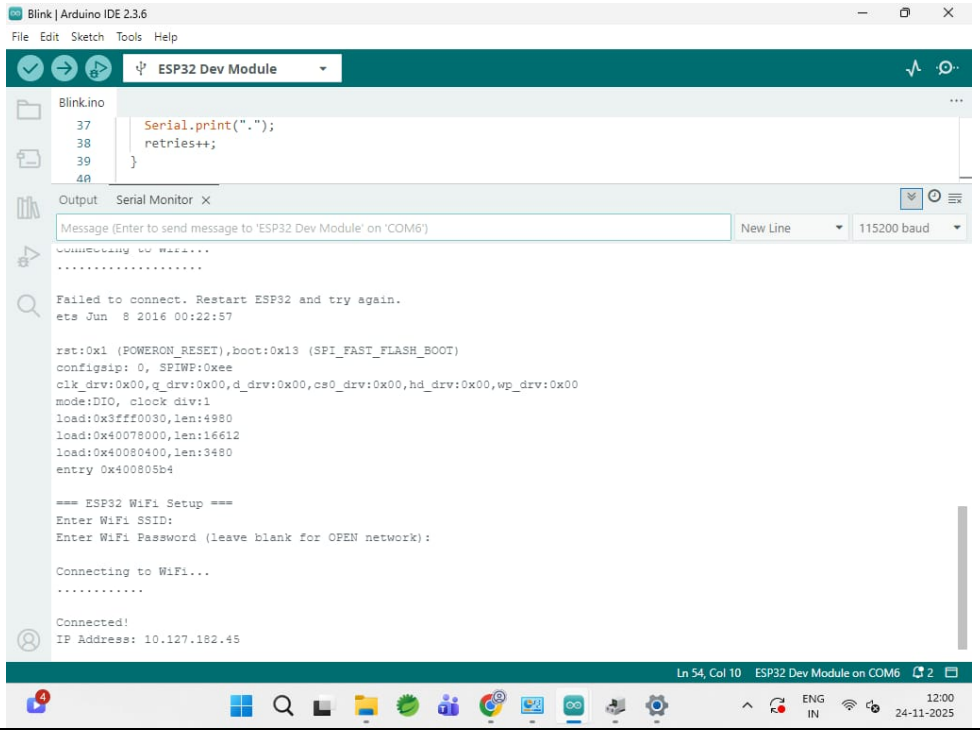
Assignment 1	**Topic: WiFi-Communication**	**Date of Assignment1: 08.11.2025**	**Date of Submission: --- 24.11.2025**

Note:

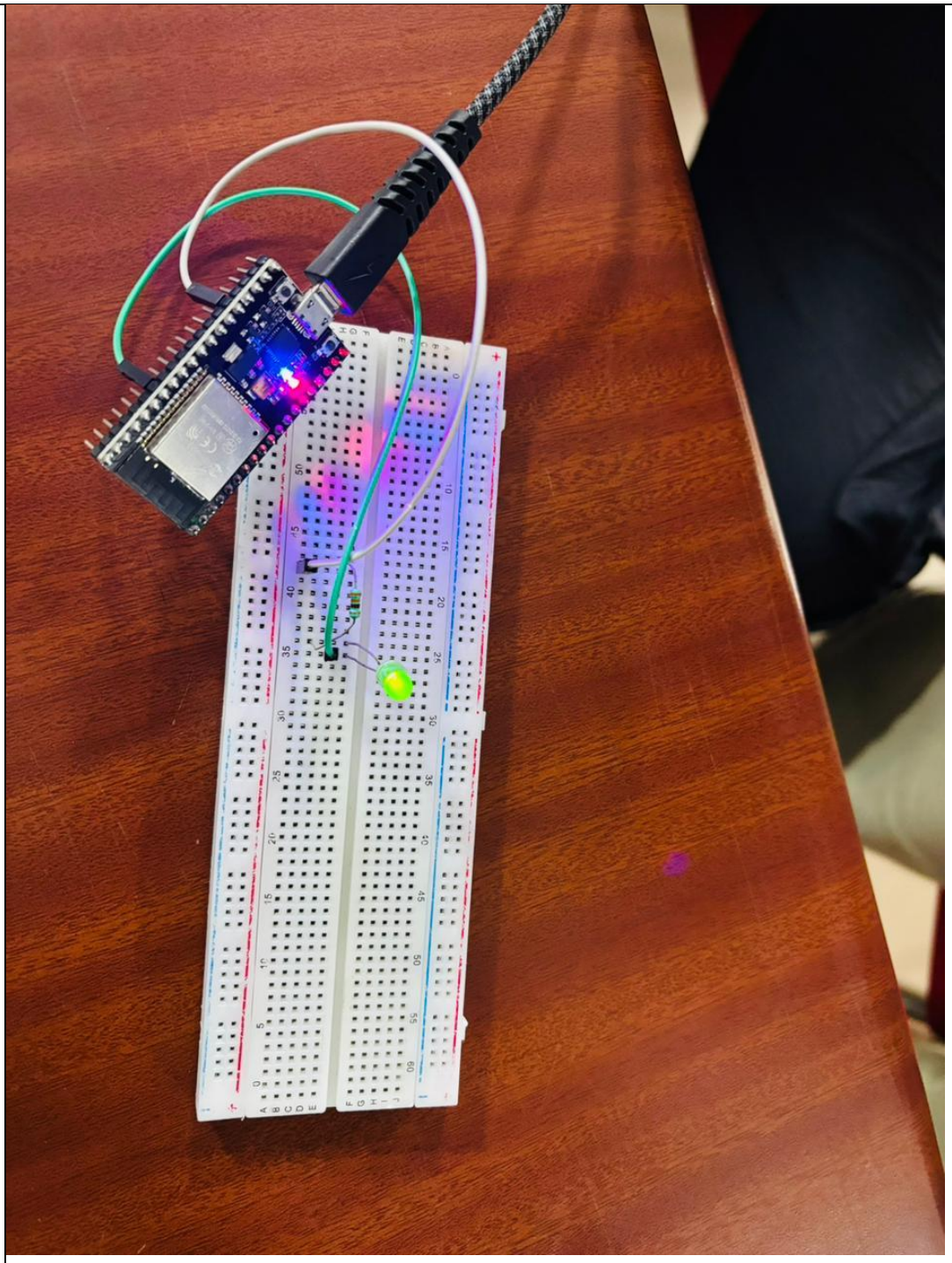
1. Assignment carries a weightage of **20 marks out of 100**
2. Course outcome CO1 to CO2 was covered.

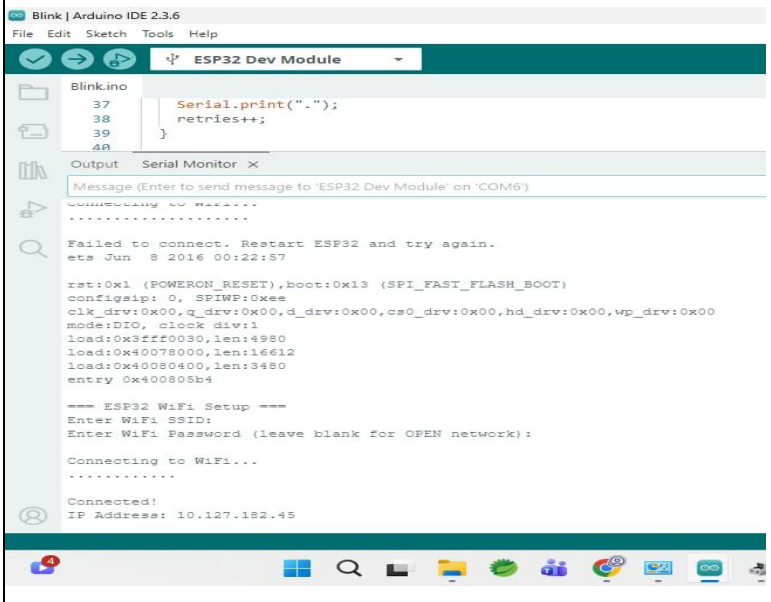
Course Outcomes	CO1	Understand the fundamentals of Arduino hardware and software.
	CO2	Interface various sensors and actuators with Arduino.
	CO3	Apply programming logic to control robotic systems.
	CO4	Design and build basic to intermediate level robotics projects.
	CO5	Demonstrate problem-solving and debugging skills in robotics.
	CO6	Collaborate in teams to plan, execute, and present robotics projects

A's	Answers
1	The primary purpose of the ESP8266 ESP-01 module when interfaced with an Arduino UNO is to provide Wi-Fi connectivity, allowing the Arduino to communicate wirelessly with servers, IoT platforms, or monitoring systems.
2	The ESP-01 module operates at 3.3V because its internal circuitry is designed for low-voltage operation. If it is connected directly to 5V, the module can overheat or become permanently damaged due to over-voltage stress.
3	**VCC** → Power (3.3V) **GND** → Ground **TX** → Transmit data (UART) **RX** → Receive data (UART) **CH_PD / EN** → Chip enable (must be HIGH to run) **RST** → Reset (active LOW) **GPIO0** → General-purpose I/O (used for programming mode) **GPIO2** → General-purpose I/O (used for normal boot mode)
4	The default baud rate of the ESP-01 module is 115200 bps. It is important to match this baud rate with the Arduino's serial communication settings; otherwise, the two devices will not be able to exchange data correctly.

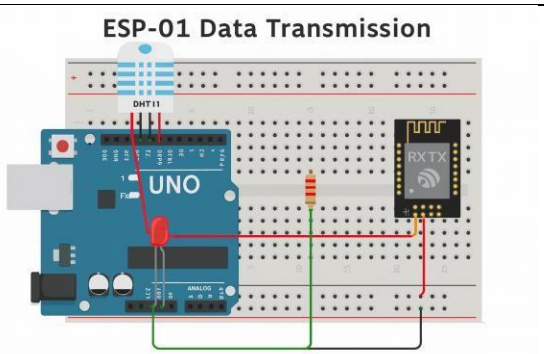
5	In Station (STA) mode, the ESP-01 connects to an existing Wi-Fi router as a client. In Access Point (AP) mode, the ESP-01 creates its own Wi-Fi hotspot to which other devices can connect. In the combined STA+AP mode, the ESP-01 can simultaneously connect to a router while also acting as a hotspot for other devices.
6	The AT command “AT” is used to test whether the module is responding. The command “AT+RST” resets the module, and the command “AT+CWJAP="SSID", "PASSWORD"” is used to connect the module to a Wi-Fi network by providing the network’s SSID and password.
7	In an IoT-based Arduino system, the ESP-01 module plays the role of a communication bridge between the Arduino and the internet. It enables the Arduino to send real-time sensor data to cloud servers or mobile applications for monitoring and control.
8	Serial communication between the Arduino UNO and the ESP-01 module occurs through UART, where the Arduino’s TX pin is connected to the ESP’s RX pin and the Arduino’s RX pin is connected to the ESP’s TX pin. Data is exchanged in the form of AT commands and responses.
9	A voltage divider circuit or a logic level converter is required when connecting the Arduino TX pin (which outputs 5V) to the ESP-01 RX pin (which only accepts 3.3V). Without this, the ESP-01 could be damaged by the higher voltage.
10	The command AT+CWJAP="SSID", "PASSWORD" is used during ESP-01 configuration to connect the module to a Wi-Fi network by providing the correct network name and password.
11 a	
11 b	

11 c




Blink | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
Blink.ino
37 Serial.print(".");
38 retries++;
39 }
Output Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM6')
Failed to connect. Restart ESP32 and try again.
ets Jun 8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4980
load:0x40078000,len:16612
load:0x40080400,len:3480
entry 0x400805b4
=== ESP32 WiFi Setup ===
Enter WiFi SSID:
Enter WiFi Password (leave blank for OPEN network):
Connecting to WiFi...
Connected!
IP Address: 10.127.182.45

12 a

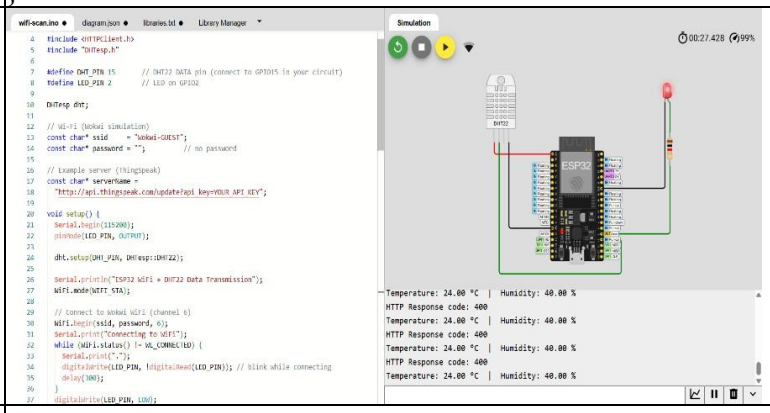
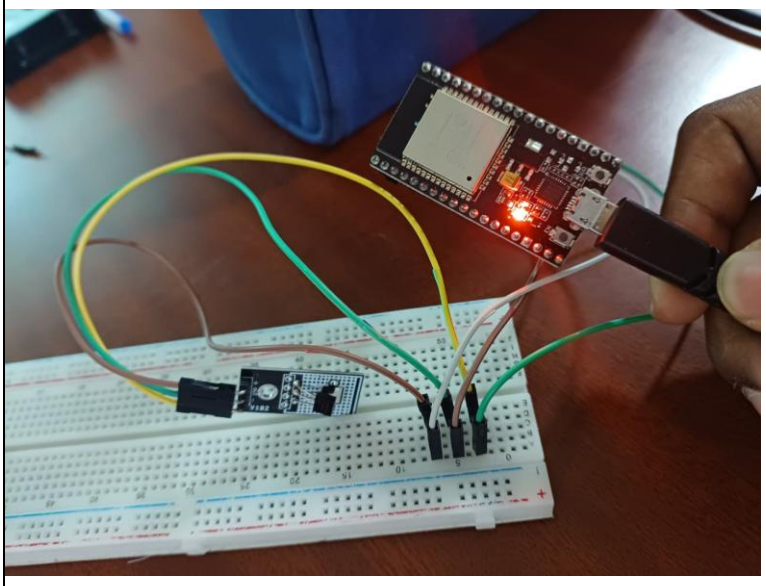


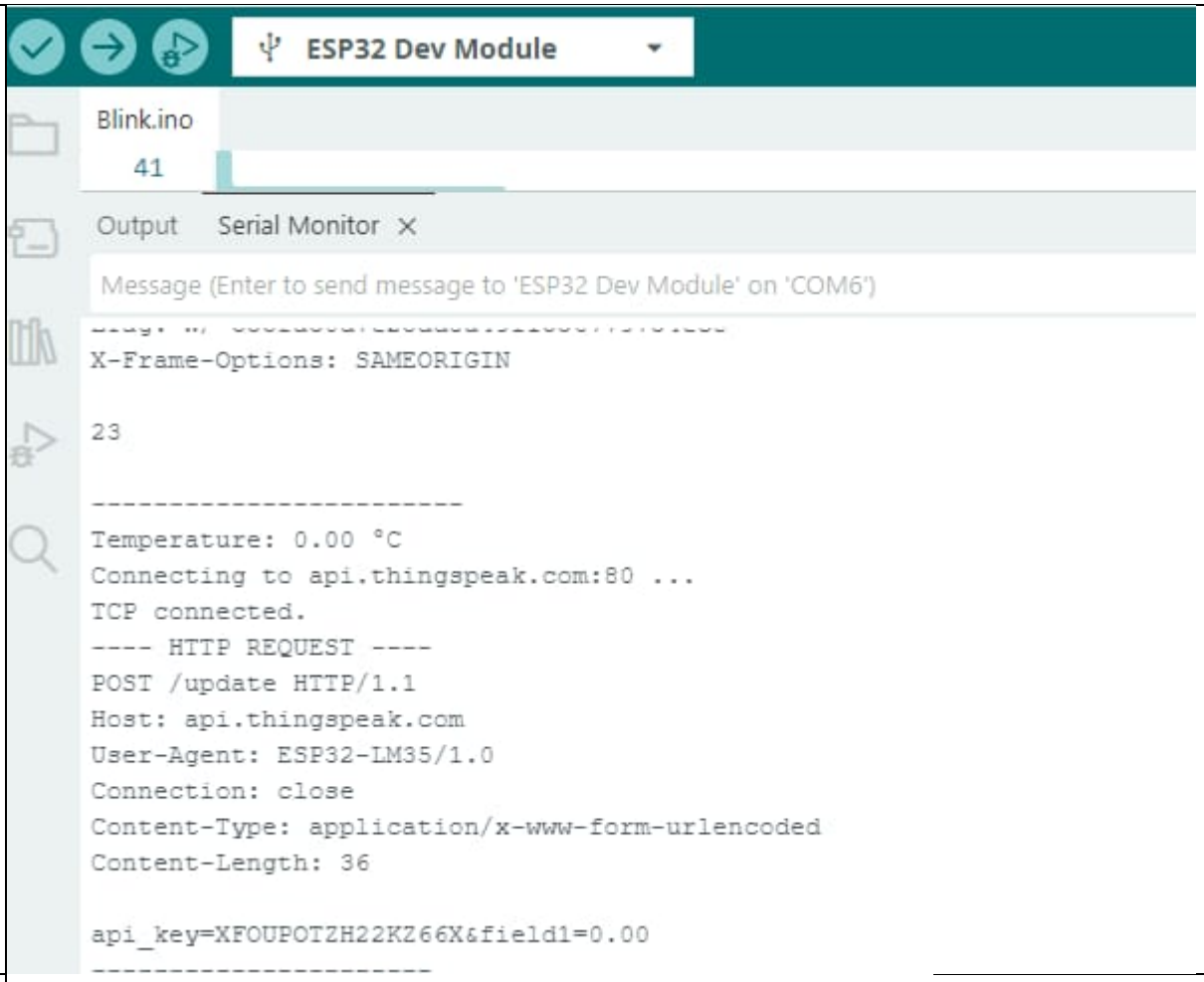
12 b

```
#include <WiFi.h>

#define LM35_PIN 34     // LM35 connected to ADC pin 34
#define LEDPIN 2        // LED connected to GPIO2

const char* ssid = "TP-Link_2240";
const char* password = "12345678";

const char* host = "api.thingspeak.com";
const String writeAPIKey = "XFOUPOTZH22KZ66X";

void setup() {
  Serial.begin(115200);
  delay(100);

  pinMode(LEDPIN, OUTPUT);

  Serial.println("\n== ThingSpeak LM35 Diagnostic Sketch ==");

  WiFi.mode(WIFI_STA);
  Serial.printf("Connecting to WiFi SSID '%s' ...\n", ssid);
  WiFi.begin(ssid, password);

  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - start < 15000) {
    Serial.print(".");
    delay(500);
  }

  Serial.println();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.print("Connected. IP: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("WiFi connect failed.");
  }

  analogReadResolution(12); // ADC range 0–4095
}

float readLM35() {
  int adcValue = analogRead(LM35_PIN);

  // ESP32 ADC range: 0–4095 for 0–3.3V
  float voltage = adcValue * (3.3 / 4095.0);

  // LM35 outputs 10mV per °C
  float temperatureC = voltage / 0.01;   // = voltage * 100

  return temperatureC;
}
```

```
void sendToThingSpeak(float temperature) {
 if (WiFi.status() != WL_CONNECTED) {
  Serial.println("WiFi not connected - skipping send.");
  return;
 }

 WiFiClient client;
 Serial.printf("Connecting to %s:80 ...\n", host);

 if (!client.connect(host, 80)) {
  Serial.println("Connection to ThingSpeak failed!");
  Serial.printf("Client connect failed, state=%d\n", client.connected());
  return;
 }
 Serial.println("TCP connected.");

 String postData = "api_key=" + writeAPIKey +
                   "&field1=" + String(temperature, 2);

 // Build request
 String request = "";
 request += "POST /update HTTP/1.1\r\n";
 request += "Host: " + String(host) + "\r\n";
 request += "User-Agent: ESP32-LM35/1.0\r\n";
 request += "Connection: close\r\n";
 request += "Content-Type: application/x-www-form-urlencoded\r\n";
 request += "Content-Length: " + String(postData.length()) + "\r\n";
 request += "\r\n";
 request += postData;

 Serial.println("---- HTTP REQUEST ----");
 Serial.println(request);
 Serial.println("----------------------");

 client.print(request);

 unsigned long timeout = millis() + 5000;
 String response = "";
 while (client.connected() && millis() < timeout) {
  while (client.available()) {
   String line = client.readStringUntil('\n');
   response += line + "\n";
   timeout = millis() + 5000; // extend timeout on activity
  }
  delay(10);
 }

 if (response.length() == 0) {
  Serial.println("No response received from ThingSpeak (timeout).");
 } else {
  Serial.println("---- HTTP RESPONSE ----");
  Serial.println(response);
  Serial.println("-----------------------");
 }

 client.stop();
}

void loop() {
 float temperature = readLM35();

 Serial.printf("Temperature: %.2f °C\n", temperature);
```

	// LED indicator
	if (temperature > 20.0) digitalWrite(LEDPIN, HIGH);
	else digitalWrite(LEDPIN, LOW);
	
	// Send temperature to ThingSpeak
	sendToThingSpeak(temperature);
	
	delay(20000); // 20 seconds (ThingSpeak limit)
	}

12 c	
12 d	



12 e

ESP-01 can reliably send sensor data in IoT systems if Wi-Fi is stable, sensor data is validated, and proper protocols like HTTP or MQTT are used.
Using Station Mode, reconnection logic, and a stable 3.3V power supply, the data transmission becomes accurate and continuous.
Therefore, ESP-01 is suitable for real-time IoT applications such as monitoring, automation, and cloud-based systems.

13 a

jupyter Untitled Last Checkpoint: 16 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

```
[*]: import socket

HOST = "0.0.0.0"      # Listen on all interfaces
PORT = 5000

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen(1)
    print(f"Server listening on {HOST}:{PORT}")
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print("Received from ESP32:", data.decode().strip())
            reply = b"ACK: " + data
            conn.sendall(reply)
```

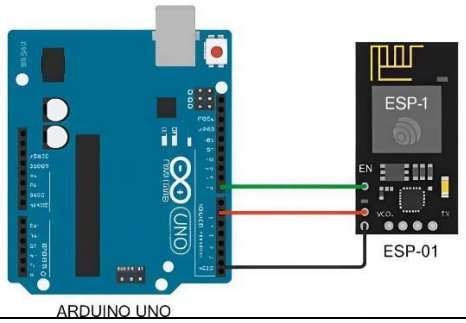
```
Server listening on 0.0.0.0:5000
Connected by ('192.168.137.147', 60845)
Received from ESP32: Hello from ESP32! Time(ms): 32743
Received from ESP32:
Received from ESP32: Hello from ESP32! Time(ms): 34744
Received from ESP32:
Received from ESP32: Hello from ESP32! Time(ms): 36745
Received from ESP32:
Received from ESP32: Hello from ESP32! Time(ms): 38746
```

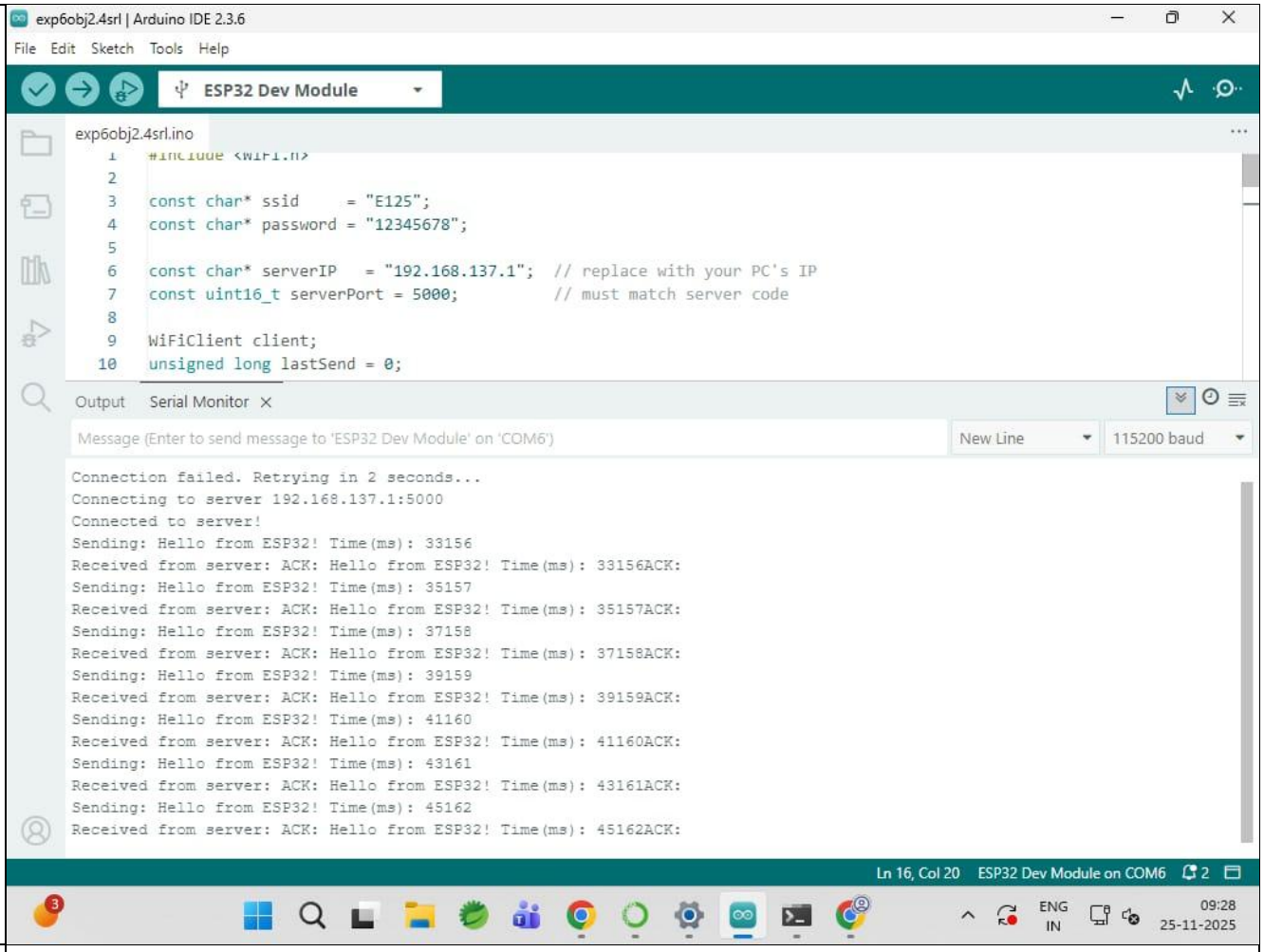
13 b

AT // Test module
AT+RST // (optional) Reset module AT+CWMODE=1 // Station mode AT+CWJAP="SSID","PASSWORD" // Join Wi-Fi AP
AT+CIPMUX=0 // Single connection (TCP client)
AT+CIPSTART="TCP","192.168.1.100",8080 // Open TCP connection

// Tell ESP-01 how many bytes we will send (5 characters: HELLO) AT+CIPSEND=5
HELLO // Actual data sent over TCP AT+CIPCLOSE

13 c



13 d	
14	ESP-01 Wi-Fi communication contributes to the development of smart IoT systems by enabling devices to connect to the internet and share data in real time. For example, in a smart home system, the ESP-01 can send temperature and humidity readings from sensors to a cloud server, where the data can be monitored through a mobile application. This makes IoT systems more efficient, scalable, and user-friendly.