Perfect, this is a **very good beginner-friendly requirement** 👍
I'll give you a **clean, working, ROS2-style solution** with **very detailed steps**.

We will build this in a **correct architecture**:

🧠 FINAL ARCHITECTURE (Simple & Correct)

Your Keyboard (Terminal)

↓

ROS2 Python Node

↓ (Serial)

ESP8266 / Arduino

↓

LED (Blink / Fast Blink / Fade)

👉 **You will NOT type in Arduino Serial Monitor**
👉 **You will type commands in ROS2 terminal**

This is the **right way** when using ROS.

🎯 Behavior You Asked For

Input	Action
1	Normal blink
2	Fast blink
3	Fade
q	Stop LED + wait for next command

✓ LED keeps running until **you send another command**
✓ You can send commands **any number of times**

PART 1️⃣ Arduino / ESP8266 Code

📌 What this does

- Listens to **serial input**
- Executes LED mode
- Stops current mode when new command comes

🔌 **Connections**

LED + -> D5 (GPIO14)

LED - -> GND (with 220Ω resistor)

✅ **Arduino Code (UPLOAD THIS FIRST)**

```
int ledPin = D5;

char command = '0';

void setup() {
 Serial.begin(9600);
 pinMode(ledPin, OUTPUT);
 Serial.println("ESP8266 READY");
}

void loop() {
 if (Serial.available()) {
  command = Serial.read();
 }

 if (command == '1') {
  // Normal blink
  digitalWrite(ledPin, HIGH);
  delay(500);
```

```
  digitalWrite(ledPin, LOW);
  delay(500);
 }

 else if (command == '2') {
  // Fast blink
  digitalWrite(ledPin, HIGH);
  delay(100);
  digitalWrite(ledPin, LOW);
  delay(100);
 }

 else if (command == '3') {
  // Fade
  for (int i = 0; i <= 1023 && command == '3'; i += 10) {
    analogWrite(ledPin, i);
    delay(5);
    if (Serial.available()) command = Serial.read();
  }
  for (int i = 1023; i >= 0 && command == '3'; i -= 10) {
    analogWrite(ledPin, i);
    delay(5);
    if (Serial.available()) command = Serial.read();
  }
 }

 else if (command == 'q') {
  digitalWrite(ledPin, LOW);  // Stop LED
```

```
 }
}
```

📌 **Upload this using Arduino IDE**
📌 Select correct **COM port**
📌 Close Arduino Serial Monitor after upload

PART 2️⃣ ROS2 Python Node (User Input)

📌 **What this does**

- Takes **keyboard input**
- Sends it via **serial** to ESP8266
- Runs continuously until Ctrl+C

📂 **ROS2 Package Structure**

```
ros2_ws/
└── src/
    └── led_control/
        ├── led_control/
        │   ├── __init__.py
        │   └── led_serial_node.py
        ├── package.xml
        ├── setup.py
```

✅ **ROS2 Python Code**

📄 led_serial_node.py

```
import rclpy
from rclpy.node import Node
import serial
import time
```

```
class LEDSerialNode(Node):
    def __init__(self):
        super().__init__('led_serial_node')

        # CHANGE THIS PORT
        self.ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
        time.sleep(2)

        self.get_logger().info("LED Controller Ready")
        self.get_logger().info("Enter:")
        self.get_logger().info("1 - Normal Blink")
        self.get_logger().info("2 - Fast Blink")
        self.get_logger().info("3 - Fade")
        self.get_logger().info("q - Stop")

        self.run()

    def run(self):
        while rclpy.ok():
            cmd = input("\nEnter command (1/2/3/q): ")

            if cmd in ['1', '2', '3', 'q']:
                self.ser.write(cmd.encode())
                self.get_logger().info(f"Sent command: {cmd}")
            else:
                self.get_logger().warn("Invalid input")
```

```
def main():
    rclpy.init()
    node = LEDSerialNode()
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

🧠 **IMPORTANT: Serial Port**

Find your ESP port:

```
ls /dev/ttyUSB*
```

OR

```
ls /dev/ttyACM*
```

Update this line:

```
self.ser = serial.Serial('/dev/ttyUSB0', 9600)
```

PART 3️⃣ Build & Run ROS2

```
cd ~/ros2_ws
colcon build
source install/setup.bash
```

Run node:

```
ros2 run led_control led_serial_node
```

🧪 **HOW YOU USE IT (STEP-BY-STEP)**

1️⃣ Upload Arduino code
2️⃣ Close Arduino IDE

3️⃣ Run ROS2 node
4️⃣ Type commands in **terminal**:

1 → normal blink

2 → fast blink

3 → fade

q → stop LED

👉 LED keeps running
👉 New command interrupts previous mode
👉 Run forever until **Ctrl + C**

🔥 **WHY THIS IS THE CORRECT WAY**

❌ Arduino Serial Monitor → NOT ROS
✅ ROS node → controls hardware
✅ Easily expandable to:

- Topics
- Joystick
- Keyboard WASD
- GUI
- Gazebo later

NEXT UP (Tell me when ready)

- Convert this to **ROS2 Topic-based**
- Control LED using **keyboard node**
- Control **motor / robot**
- Replace LED with **PWM motor**
- Add **joystick + WASD**

You're learning ROS the **right way** 👏