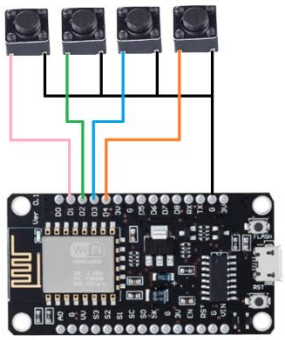
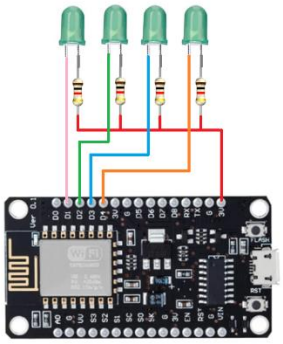
ESP8266 One-way Communication with ESPNOW



SENDER



RECEIVER

ESP-NOW is a protocol developed by Espressif for their ESP8266 and ESP32 series of microcontrollers. It allows for low-power, peer-to-peer communication between ESP devices without the need for a Wi-Fi network or internet connection. **Can communicate using both one-way and two-way communication**. Here's a breakdown of how it works and its key features:

Key Features

Low Power Consumption: ESP-NOW is designed to be energy-efficient, making it suitable for battery-operated devices.

Peer-to-Peer Communication: Devices can communicate directly with each other. This eliminates the need for a central hub or router.

Low Latency: The protocol is optimized for low-latency communication, which is beneficial for real-time applications.

Secure Communication: ESP-NOW supports encryption, ensuring that data transmitted between devices remains secure.

Broadcast and Unicast: ESP-NOW supports both broadcast (sending data to all devices within range) and unicast (sending data to a specific device).

Must match the receiver structure: The size of the variables in the sender and receiver must be the same size.

How It Works

Device Pairing: Before communication begins, devices need to be paired. Each device has a unique MAC address, which is used to identify it on the network. Pairing involves configuring devices to recognize each other's MAC addresses.

Data Transmission: Once paired, devices can exchange messages. The protocol supports sending small packets of data (up to 250 bytes per packet) quickly and efficiently.

Message Handling: Devices can handle messages in real-time. ESP-NOW allows for sending and receiving of messages through callbacks, making it easy to integrate with other code.

Network Topology: ESP-NOW operates in a simple star topology where one device can communicate with multiple devices, but it doesn’t support complex network topologies.

Typical Use Cases

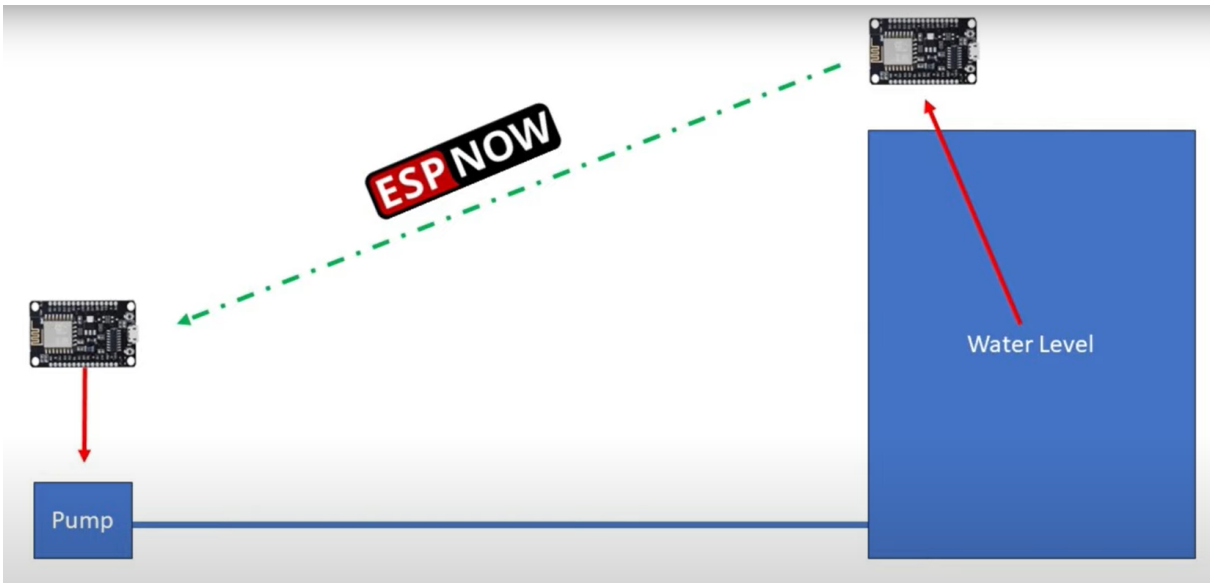
Sensor Networks: Connecting multiple sensors to a central device or among themselves without needing a central server or router.

Home Automation: Communicating between smart devices in a home automation system without relying on Wi-Fi.

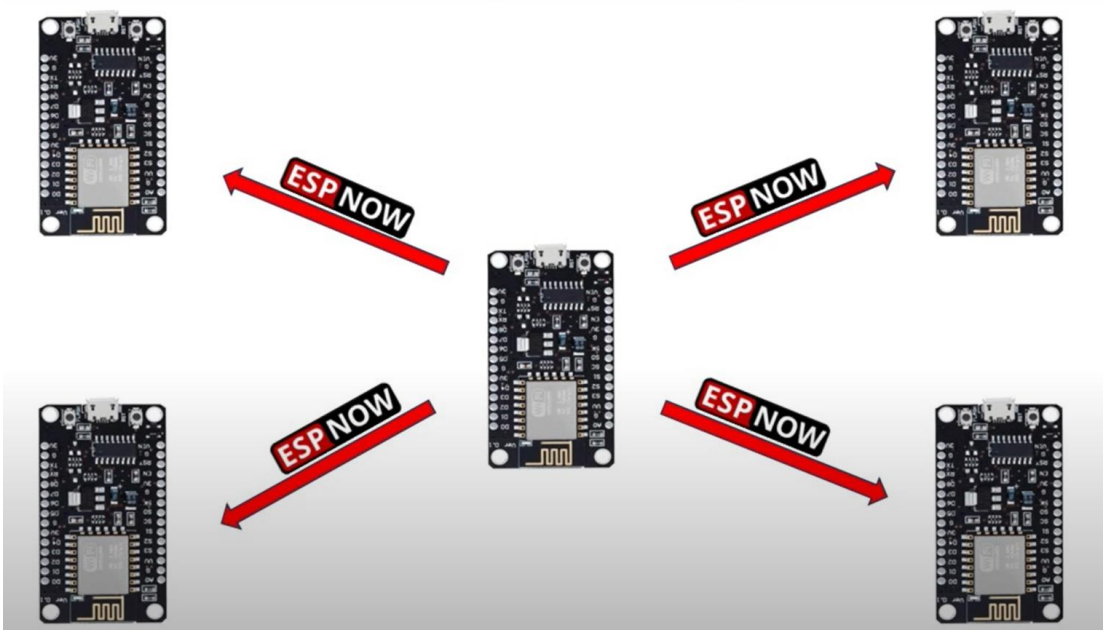
Remote Controls: Creating remote controls for devices, where commands can be sent directly from one device to another.

One-way Communication

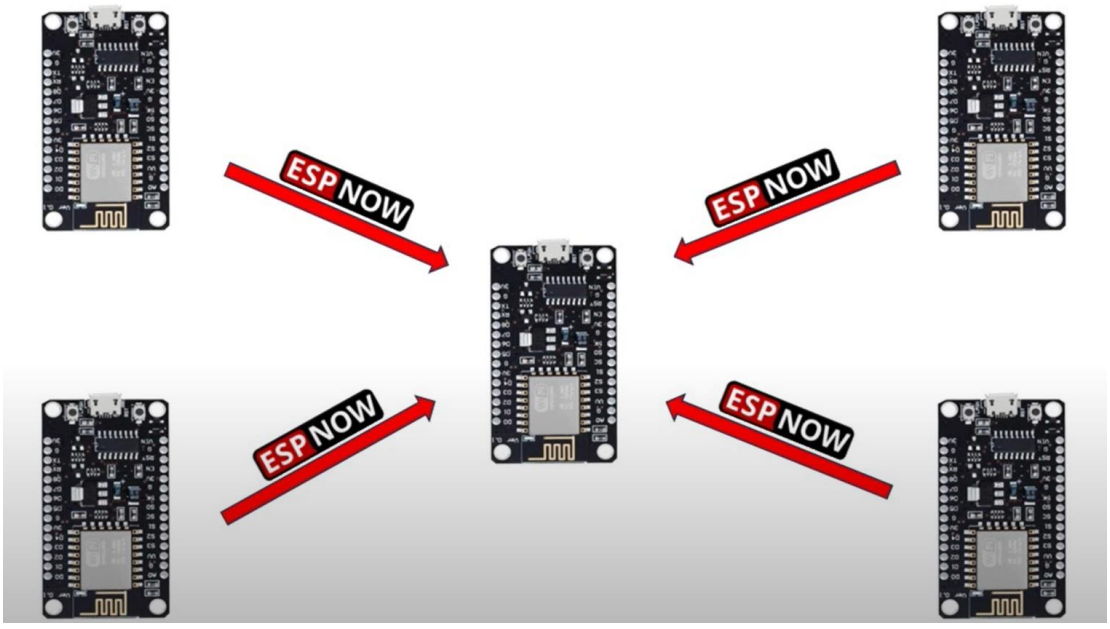
One Node sender to One Node Receiver



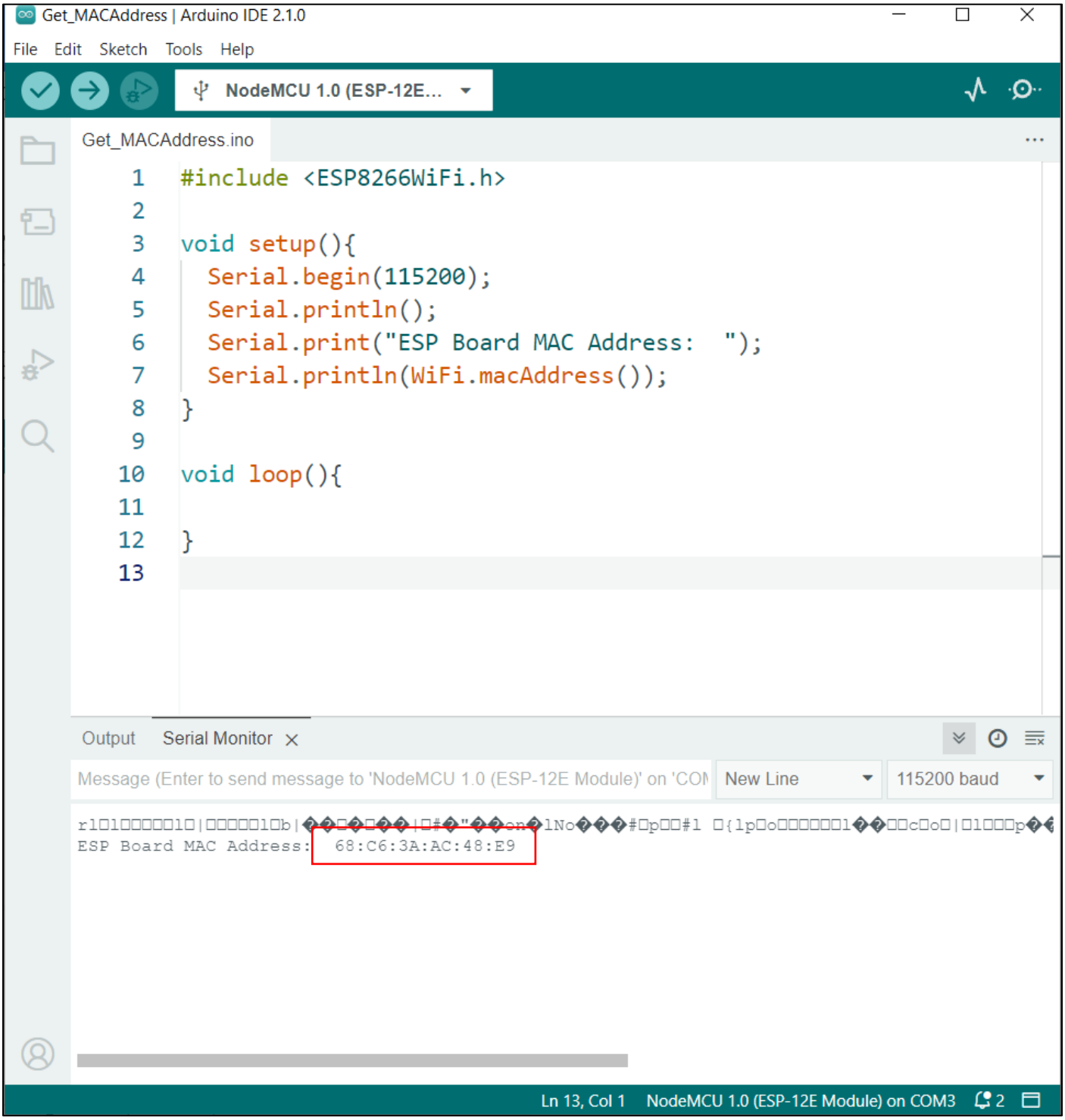
One “Master” Node Sending to Multiple Node “Slaves”



One “Slave” Node Receiving data from Multiple Node “Masters”



1. Check NodeMCU “Slave” Receiver MAC Address



Modify MAC Address 68:C6:3A:AC:48:E9 as {0x68, 0xC6, 0x3A, 0xAC, 0x48, 0xE9} for insert to Sender.

2. Node “Slave” Receiver Code.

new-ESPNOW-Receiver | Arduino IDE 2.1.0

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E...

new-ESPNOW-Receiver.ino

```
#include <ESP8266WiFi.h>
#include <espnow.h>

#define LED1_PIN  D1
#define LED2_PIN  D2
#define LED3_PIN  D3
#define LED4_PIN  D4

#define LED_OFF   HIGH
#define LED_ON    LOW

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
    bool SW_1;
    bool SW_2;
    bool SW_3;
    bool SW_4;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// Callback function that will be executed when data is received
void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("SW 1: ");
  Serial.println((myData.SW_1 ? "ON" : "OFF"));
  Serial.print("SW 2: ");
  Serial.println((myData.SW_2 ? "ON" : "OFF"));
  Serial.print("SW 3: ");
  Serial.println((myData.SW_3 ? "ON" : "OFF"));
  Serial.print("SW 4: ");
  Serial.println((myData.SW_4 ? "ON" : "OFF"));
  Serial.println();

  // Control LEDs based on received data
  digitalWrite(LED1_PIN, myData.SW_1 ? LED_OFF : LED_ON);  // LED ON when switch is
  digitalWrite(LED2_PIN, myData.SW_2 ? LED_OFF : LED_ON);  // LED ON when switch is
  digitalWrite(LED3_PIN, myData.SW_3 ? LED_OFF : LED_ON);  // LED ON when switch is
  digitalWrite(LED4_PIN, myData.SW_4 ? LED_OFF : LED_ON);  // LED ON when switch is
}

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);

```

```
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
  pinMode(LED3_PIN, OUTPUT);
  pinMode(LED4_PIN, OUTPUT);

  // Initialize LEDs to OFF
  digitalWrite(LED1_PIN, LED_OFF);
  digitalWrite(LED2_PIN, LED_OFF);
  digitalWrite(LED3_PIN, LED_OFF);
  digitalWrite(LED4_PIN, LED_OFF);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register receive callback
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {
  // Nothing to do here, everything is handled in OnDataRecv
}
```

Output Serial Monitor ×

Output Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM3')

```
SW 4: ON

Bytes received: 4
SW 1: OFF
SW 2: OFF
SW 3: OFF
SW 4: ON

Bytes received: 4
SW 1: OFF
SW 2: OFF
SW 3: OFF
SW 4: ON

B
```

3. Node “Master” Sender Code.

new-ESPNOW-Sender | Arduino IDE 2.1.0

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E...

new-ESPNOW-Sender.ino

```
#include <ESP8266WiFi.h>
#include <espnow.h>

#define SW1_PIN   D1
#define SW2_PIN   D2
#define SW3_PIN   D3
#define SW4_PIN   D4

// REPLACE WITH RECEIVER MAC Address
uint8_t broadcastAddress[] = {0x68, 0xC6, 0x3A, 0xAC, 0x48, 0xE9};

// Structure example to send data
typedef struct struct_message {
    bool SW_1;
    bool SW_2;
    bool SW_3;
    bool SW_4;
} struct_message;

// Create a struct_message called myData
struct_message myData;

unsigned long lastTime = 0;
unsigned long timerDelay = 50;  // send readings timer

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }

  //Serial.println(digitalRead(SW4_PIN));
}

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);

  pinMode(SW1_PIN, INPUT_PULLUP);
  pinMode(SW2_PIN, INPUT_PULLUP);
  pinMode(SW3_PIN, INPUT_PULLUP);
  pinMode(SW4_PIN, INPUT_PULLUP);

```

```
48    // Initialize switch states
49    myData.SW_1 = digitalRead(SW1_PIN) == LOW;  // Switch pressed = LOW
50    myData.SW_2 = digitalRead(SW2_PIN) == LOW;
51    myData.SW_3 = digitalRead(SW3_PIN) == LOW;
52    myData.SW_4 = digitalRead(SW4_PIN) == LOW;
53
54    // Set device as a Wi-Fi Station
55    WiFi.mode(WIFI_STA);
56
57    // Init ESP-NOW
58    if (esp_now_init() != 0) {
59      Serial.println("Error initializing ESP-NOW");
60      return;
61    }
62
63    // Register send callback
64    esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
65    esp_now_register_send_cb(OnDataSent);
66
67    // Register peer
68    esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
69  }
70
71  void loop() {
72    if ((millis() - lastTime) > timerDelay) {
73      // Read switch states
74      myData.SW_1 = digitalRead(SW1_PIN) == LOW;  // Switch pressed = LOW
75      myData.SW_2 = digitalRead(SW2_PIN) == LOW;
76      myData.SW_3 = digitalRead(SW3_PIN) == LOW;
77      myData.SW_4 = digitalRead(SW4_PIN) == LOW;
78
79      // Send data
80      esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));
81
82      lastTime = millis();
83    }
84    //delay(1000);
85  }
86
```

Output | Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM3') | New Line | 115200 baud

```
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
Last Packet Send Status: Delivery fail
```

Ln 86, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM3 2