perplexity

Wireless Control of ESP32 P10 LED Display: ESP-NOW vs BLE Implementation Guide

Based on your requirements for controlling a P10 LED display from another ESP32 board over approximately 100 meters, I'll provide a comprehensive guide comparing both ESP-NOW and BLE protocols, with a recommendation for your specific use case.

Protocol Comparison for Your Requirements

ESP-NOW Protocol

ESP-NOW is the recommended choice for your application due to its superior range capabilities[1]. ESP-NOW can achieve stable communication up to 220 meters in open field conditions, with some implementations reaching up to 500 meters in long-range mode[2]. The protocol supports payload sizes up to 250 bytes, which is more than sufficient for sending text strings and control commands[3].

BLE (Bluetooth Low Energy)

While BLE has a theoretical range of up to 100 meters in open areas, it typically achieves only 10 meters indoors and has higher power consumption compared to ESP-NOW[4]. For your 100-meter range requirement, ESP-NOW provides better reliability and performance.

Step-by-Step Implementation Guide

Step 1: Project Architecture

You'll need two ESP32 boards:

- **Sender ESP32**: Acts as the controller to send text and commands
- **Receiver ESP32**: Connected to your P10 display, receives and displays the text

Step 2: Hardware Setup

Receiver ESP32 (Display Controller):

- Keep your existing P10 LED display connections
- Use the DMD32 library as you're already doing[5]
- Ensure proper power supply for both ESP32 and P10 display

Sender ESP32 (Remote Controller):

- Can include buttons, potentiometer, or other input devices for control

- Optional: Add a small display for status feedback

Step 3: Code Implementation - ESP-NOW Receiver

Here's the modified code for your display controller ESP32:

```
#include <DMD32.h>
#include <fonts/Arial_black_16.h>
#include <esp_now.h>
#include <WiFi.h>

#define DISPLAYS_ACROSS 1
#define DISPLAYS_DOWN 1
DMD dmd(DISPLAYS_ACROSS, DISPLAYS_DOWN);

hw_timer_t* timer = NULL;

// Structure to receive data
typedef struct struct_message {
    char text[^100];        // Text to display
    int scrollSpeed;       // Scroll speed control
    bool startStop;        // Start/stop scrolling
} struct_message;

struct_message receivedData;
String currentText = "HELLO";
int currentScrollSpeed = 50;
bool isScrolling = true;
bool newDataReceived = false;

void IRAM_ATTR triggerScan() {
    dmd.scanDisplayBySPI();
}

// Callback when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
    memcpy(&receivedData, incomingData, sizeof(receivedData));

    // Update display parameters
    currentText = String(receivedData.text);
    currentScrollSpeed = receivedData.scrollSpeed;
    isScrolling = receivedData.startStop;
    newDataReceived = true;

    Serial.println("Data received:");
    Serial.println("Text: " + currentText);
    Serial.println("Speed: " + String(currentScrollSpeed));
    Serial.println("Scrolling: " + String(isScrolling));
}

void setup() {
    Serial.begin(115200);
    delay(500);

    // Initialize DMD display
```

```
    uint8_t cpuClock = ESP.getCpuFreqMHz();
    timer = timerBegin(0, cpuClock, true);
    timerAttachInterrupt(timer, &triggerScan, true);
    timerAlarmWrite(timer, 300, true);
    timerAlarmEnable(timer);

    dmd.clearScreen(true);

    // Initialize ESP-NOW
    WiFi.mode(WIFI_STA);
    if (esp_now_init() != ESP_OK) {
        Serial.println("Error initializing ESP-NOW");
        return;
    }

    // Register callback function
    esp_now_register_recv_cb(OnDataRecv);

    Serial.println("ESP-NOW Receiver Ready");
    Serial.print("MAC Address: ");
    Serial.println(WiFi.macAddress());
}

void loop() {
    if (isScrolling && currentText.length() > 0) {
        dmd.selectFont(Arial_Black_16);

        char charArray[currentText.length() + 1];
        currentText.toCharArray(charArray, currentText.length() + 1);

        if (newDataReceived) {
            dmd.clearScreen(true);
            dmd.drawMarquee(charArray, currentText.length(), (32 * DISPLAYS_ACROSS) - 1,
            newDataReceived = false;
        }

        long startTime = millis();
        long timer_1 = startTime;
        boolean ret = false;

        while (!ret && isScrolling) {
            if ((timer_1 + 30) < millis()) {
                ret = dmd.stepMarquee(-1, 0);
                delay(currentScrollSpeed);
                timer_1 = millis();
            }

            // Check for new data during scrolling
            if (newDataReceived) {
                break;
            }
        }
    } else {
        delay(100); // Small delay when not scrolling
    }
}
```

Step 4: Code Implementation - ESP-NOW Sender

Create this code for your sender ESP32:

```
#include <esp_now.h>
#include <WiFi.h>

// Replace with your receiver ESP32 MAC address
uint8_t receiverAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}; // Update this!

// Structure to send data (must match receiver)
typedef struct struct_message {
    char text[^100];
    int scrollSpeed;
    bool startStop;
} struct_message;

struct_message dataToSend;
esp_now_peer_info_t peerInfo;

// Callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
    Serial.print("Last Packet Send Status: ");
    Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail")
}

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

    WiFi.mode(WIFI_STA);

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

    esp_now_register_send_cb(OnDataSent);

    // Register peer
    memcpy(peerInfo.peer_addr, receiverAddress, 6);
    peerInfo.channel = 0;
    peerInfo.encrypt = false;

    if (esp_now_add_peer(&peerInfo) != ESP_OK) {
        Serial.println("Failed to add peer");
        return;
    }

    Serial.println("ESP-NOW Sender Ready");
    Serial.println("Commands:");
    Serial.println("1. Type text to send");
    Serial.println("2. 'SPEED:XX' to set scroll speed (e.g., SPEED:30)");
    Serial.println("3. 'START' to start scrolling");
    Serial.println("4. 'STOP' to stop scrolling");
}
```

```
void loop() {
    if (Serial.available()) {
        String input = Serial.readString();
        input.trim();

        if (input.startsWith("SPEED:")) {
            int speed = input.substring(6).toInt();
            if (speed > 0 && speed <= 200) {
                dataToSend.scrollSpeed = speed;
                strcpy(dataToSend.text, ""); // Empty text, just speed change
                dataToSend.startStop = true;

                esp_err_t result = esp_now_send(receiverAddress, (uint8_t *) &dataToSend,
                Serial.println("Speed set to: " + String(speed));
            }
        }
        else if (input == "START") {
            dataToSend.startStop = true;
            strcpy(dataToSend.text, "");
            dataToSend.scrollSpeed = 50; // Default speed

            esp_err_t result = esp_now_send(receiverAddress, (uint8_t *) &dataToSend, siz
            Serial.println("Scrolling started");
        }
        else if (input == "STOP") {
            dataToSend.startStop = false;
            strcpy(dataToSend.text, "");
            dataToSend.scrollSpeed = 50;

            esp_err_t result = esp_now_send(receiverAddress, (uint8_t *) &dataToSend, siz
            Serial.println("Scrolling stopped");
        }
        else {
            // Send text message
            strcpy(dataToSend.text, input.c_str());
            dataToSend.scrollSpeed = 50; // Default speed
            dataToSend.startStop = true;

            esp_err_t result = esp_now_send(receiverAddress, (uint8_t *) &dataToSend, siz
            Serial.println("Text sent: " + input);
        }
    }

    delay(100);
}
```

Step 5: Configuration Steps

1. **Get MAC Address**: Upload the receiver code first and note the MAC address printed in the Serial Monitor[6]
2. **Update Sender Code**: Replace the `receiverAddress` array in the sender code with the actual MAC address of your receiver ESP32[7]

3. **Test Range**: Start with both boards close together, then gradually increase distance to test your 100-meter range requirement[1]

Step 6: Advanced Features

Optional Enhancements:

- Add multiple text presets that can be selected remotely[8]
- Implement different scrolling effects (left, right, up, down)
- Add brightness control for the P10 display
- Include status feedback from receiver to sender[9]

Step 7: Troubleshooting Tips

Range Optimization:

- Ensure both ESP32 antennas are properly oriented[2]
- Test in open areas first before indoor deployment
- Consider ESP-NOW Long Range (LR) mode for extended distances up to 1 kilometer[10]

Connection Issues:

- Verify MAC addresses are correctly configured
- Check that both boards use the same ESP-NOW channel
- Monitor Serial output for delivery confirmation messages[11]

Why ESP-NOW Over BLE

For your specific requirements, ESP-NOW offers several advantages:

- **Superior Range**: 220+ meters vs BLE's 100 meters maximum[1] [4]
- **Lower Power Consumption**: More efficient than BLE for periodic data transmission[3]
- **Simpler Implementation**: No pairing required, direct peer-to-peer communication[3]
- **Better Reliability**: More stable connection over distance compared to BLE[2]

This implementation provides you with wireless control over your P10 LED display, allowing you to change text, control scrolling speed, and start/stop the display remotely within your 100-meter range requirement.

⁂

1. https://randomnerdtutorials.com/esp-now-two-way-communication-esp32/
2. https://www.youtube.com/watch?v=oz0a7Ur7nko
3. https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/
4. https://deepbluembedded.com/esp32-bluetooth-classic-with-arduino-complete-guide/
5. https://forum.arduino.cc/t/led-p10-32×16-with-esp32/1229586

6. https://randomnerdtutorials.com/esp-now-auto-pairing-esp32-esp8266/
7. https://randomnerdtutorials.com/esp-now-one-to-many-esp32-esp8266/
8. https://www.luisllamas.es/en/esp-now-send-string-esp32/
9. https://randomnerdtutorials.com/esp-now-many-to-one-esp32/
10. https://developer.espressif.com/blog/esp-now-for-outdoor-applications/
11. https://www.niraltek.com/blog/exploring-esp-now-communication-with-esp32/