Two way communication in esp32 wifi module servo motor controlling using potentio meter

---

Introduction

In this project, we utilize the ESP32, a powerful microcontroller with built-in WiFi and Bluetooth capabilities, to establish communication between two ESP32 modules over WiFi. A servo motor is controlled based on the analog input from a potentiometer, simulating real-time control over a remote device.

The ESP32 server reads the potentiometer value, maps it to the servo motor's angular position, and sends the potentiometer data over WiFi to the client ESP32. The client can be used to display the values or control additional hardware components based on the received data.

Components Required

- 2 x ESP32 Modules (One as Server, One as Client)
- 1 x Servo Motor
- 1 x Potentiometer (10k Ohms)
- Jumper Wires
- Breadboard (Optional)
- Power Supply (USB or External Power for Servo Motor)

Libraries:

ESP-32 Library:

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

Servo Control:

```
#include <ESP32Servo.h>
```

Server code:

```
#include <WiFi.h>
#include <ESP32Servo.h>

const char* ssid = "EdgeHW";
const char* password = "hw_edgeprod";
WiFiServer server(80);
Servo myservo;

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  server.begin();
  myservo.attach(2);

  Serial.println("Server started");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
```

```
    while (client.connected()) {
      if (client.available()) {
        int potValue = client.parseInt();
        Serial.print("Potentiometer Value: ");
        Serial.println(potValue);
        myservo.write(map(potValue, 0, 4095, 0, 180));
      }
    }
```

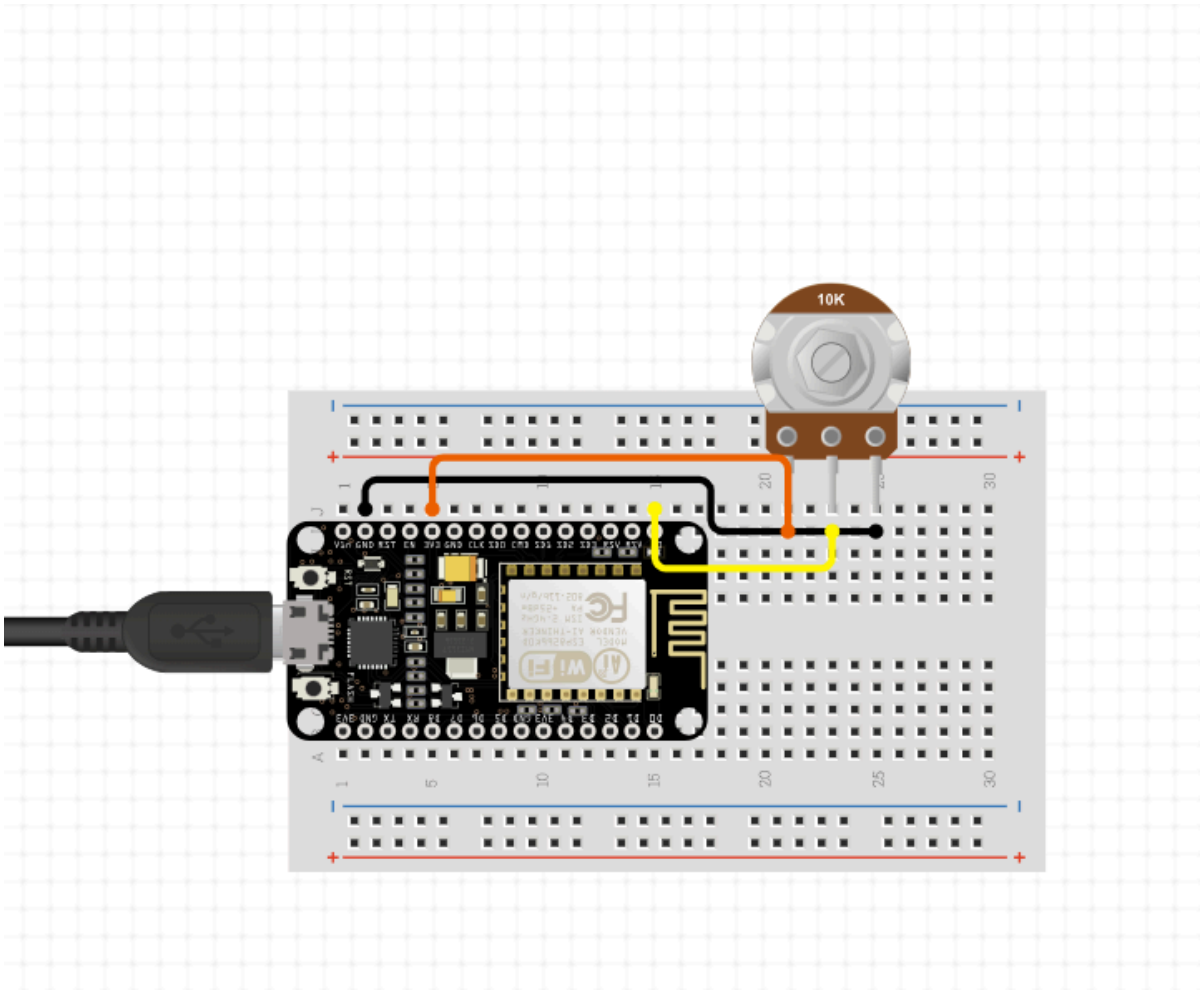
Client Side:

```
#include <WiFi.h>

const char* ssid = "EdgeHW";
const char* password = "hw_edgeprod";
const char* serverIP = "192.168.2.198";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
  }
}

    void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("WiFi connected!");
    WiFiClient client;
    if (client.connect(serverIP, 80)) {
      int potValue = analogRead(34);
      client.print(potValue);
      delay(50);
      client.stop();
    }
  } else {
    Serial.println("WiFi not connected!");
  }
```

```
  delay(500);
}
```

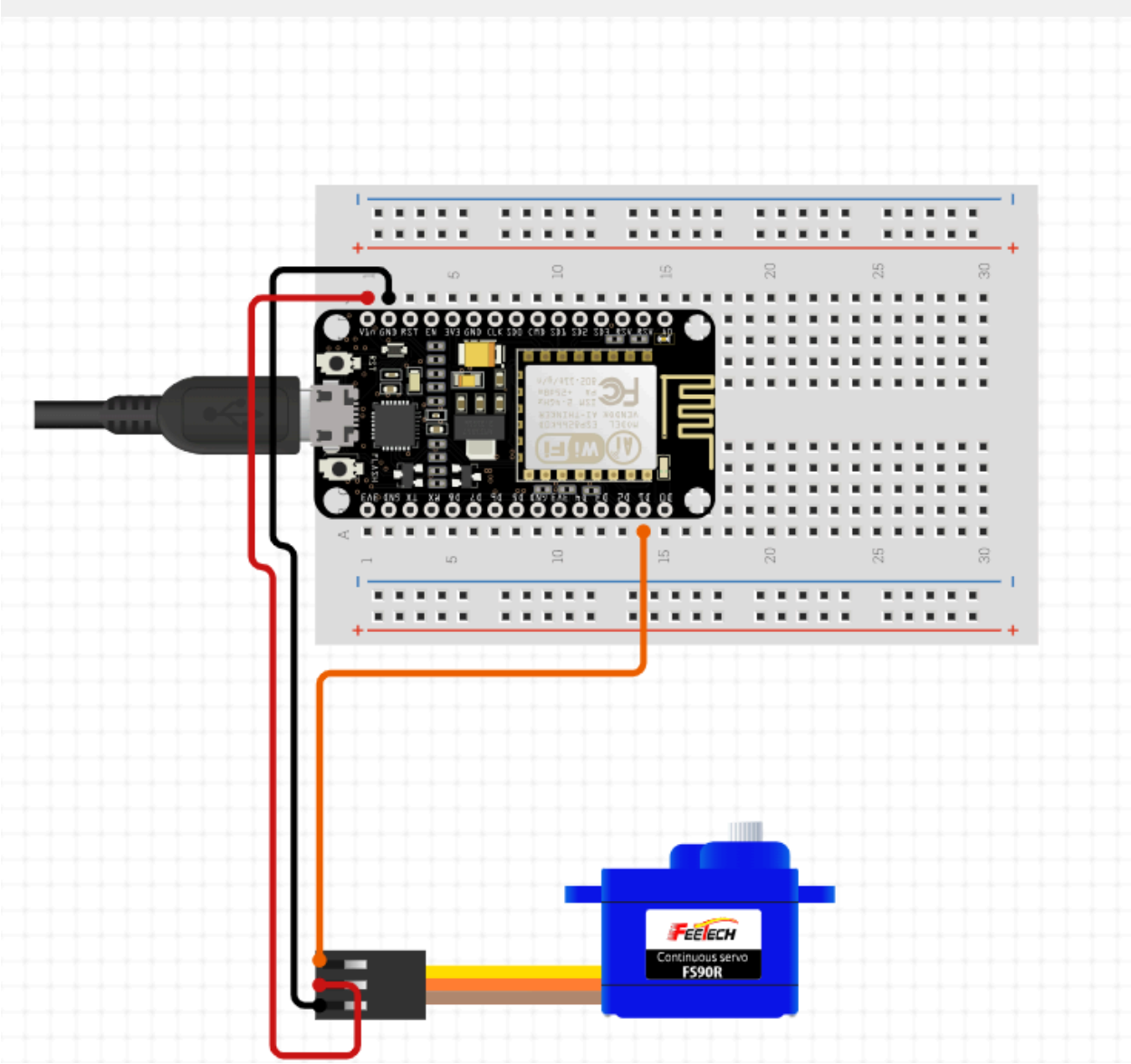
1. ESP32 Server Side Connections:

- **Potentiometer:**
    - Middle pin (wiper) connected to GPIO34 (Analog input pin of ESP32).
    - One end connected to 3.3V (Vcc), and the other end to GND.
-



Servo Motor:

- Signal pin connected to GPIO13 of the ESP32.
- Power (Vcc) connected to an external 5V power supply (if needed for higher torque).
- Ground connected to ESP32 GND



Working Principle

1. **ESP32 Server**:

    - The ESP32 server reads the analog value from the potentiometer using `analogRead()`.
    - This value is mapped to the servo motor's angular position using `map()`, which converts the 0-4095 potentiometer range into a 0-180 degree range for the servo.
    - The potentiometer value is also transmitted over WiFi to the client ESP32 via the `WiFiServer` class.
2. **ESP32 Client**:
    - The client ESP32 connects to the server's WiFi network, established by the ESP32 server.
    - Once connected, the client receives the potentiometer values sent by the server and can either display the values on the serial monitor or perform additional tasks like logging data or triggering other actuators.

Code Explanation

Server ESP32 Code

1. **WiFi Setup**:
    - The ESP32 is set up as an Access Point (AP) with a defined SSID and password.
    - The `WiFiServer` object is used to initialize a server listening on port 80.
2. **Potentiometer and Servo Control**:
    - The analog value from the potentiometer is read from GPIO34.
    - The potentiometer value (0-4095) is mapped to the servo angle (0-180 degrees) and used to control the servo via GPIO13.
3. **Sending Data to the Client**:
    - The potentiometer value is sent over WiFi to any connected clients via the `WiFiClient` object.

Client ESP32 Code

1. **WiFi Client Setup**:
    - The client ESP32 connects to the server's WiFi using `WiFi.begin()` with the same SSID and password as the server.
    - It attempts to connect to the server's IP address on port 80.
2. **Receiving Potentiometer Data**:
    - The client reads incoming data from the server, which represents the potentiometer value.
    - It then processes or displays the data, which could be used to trigger other components.

Conclusion

The ESP32 is a versatile and powerful microcontroller with built-in WiFi and Bluetooth capabilities, making it an ideal choice for IoT, automation, and embedded systems projects. Its dual-core processor, support for multiple communication protocols, and ability to handle analog and digital inputs/outputs provide endless possibilities for development. Whether you're building a smart home system or working on robotics, the ESP32 can serve as a central component in in this project.

Two way communication in esp32 bluetooth module servo motor controlling using potentio meter

Components Required

- 2 x ESP32 Modules (One as Server, One as Client)
- 1 x Servo Motor
- 1 x Potentiometer (10k Ohms)
- Jumper Wires
- Breadboard (Optional)
- Power Supply (USB or External Power for Servo Motor)

Libraries:

BluetoothSerial:

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;
```

Bluetooth mac address code:

```
#include "esp_bt.h"

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

  esp_bt_controller_init();
  esp_bt_controller_enable(ESP_BT_MODE_BTDM);
  esp_bt_dev_t *local_bt_device = esp_bt_dev_get_address();
  Serial.print("ESP32 Bluetooth MAC Address: ");
  for (int i = 0; i < ESP_BD_ADDR_LEN; i++) {
    Serial.printf("%02X", local_bt_device->bd_addr[i]);
    if (i < ESP_BD_ADDR_LEN - 1) {
      Serial.print(":");
    }
  }
  Serial.println();
}

void loop() {
}
```

Master code:

```
#include "BluetoothSerial.h"
#include <Servo.h>

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;
Servo myServo;

#ifdef USE_NAME
  String slaveName = "ESP32-BT-Slave";
#else
  String MACadd = "94:E6:86:3D:75:2E";
  uint8_t address[6]  = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
#endif

String myName = "ESP32-BT-Master";
int servoPin = 13;
int potPin = 34;

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

  SerialBT.begin(myName, true);
  Serial.printf("The device \"%s\" started in master mode, make sure slave BT device is on!\n", myName.c_str());

  myServo.attach(servoPin);

  #ifndef USE_NAME
    SerialBT.setPin(pin);
    Serial.println("Using PIN");
  #endif

  #ifdef USE_NAME
    connected = SerialBT.connect(slaveName);
    Serial.printf("Connecting to slave BT device named \"%s\"\n", slaveName.c_str());
  #else
    connected = SerialBT.connect(address);
    Serial.print("Connecting to slave BT device with MAC "); Serial.println(MACadd);
  #endif

  if (connected) {
    Serial.println("Connected Successfully!");
```

```
  } else {
    while (!SerialBT.connected(10000)) {
      Serial.println("Failed to connect. Make sure remote device is available and in range, then restart app.");
    }
  }

  if (SerialBT.disconnect()) {
    Serial.println("Disconnected Successfully!");
  }

  SerialBT.connect();
  if (connected) {
    Serial.println("Reconnected Successfully!");
  } else {
    while (!SerialBT.connected(10000)) {
      Serial.println("Failed to reconnect. Make sure remote device is available and in range, then restart app.");
    }
  }
}

void loop() {
  static unsigned long lastSerialPrintTime = 0;
  unsigned long currentMillis = millis();

  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }

  if (SerialBT.available()) {
    char receivedChar = SerialBT.read();

    if (receivedChar >= '0' && receivedChar <= '9') {
      int servoPosition = map(receivedChar - '0', 0, 9, 0, 180);
      myServo.write(servoPosition);

      if (currentMillis - lastSerialPrintTime >= 1000) {
        Serial.println(servoPosition);
        lastSerialPrintTime = currentMillis;
      }
    }
  }

  int potValue = analogRead(potPin);
  int mappedValue = map(potValue, 0, 4095, 0, 180);

  if (currentMillis - lastSerialPrintTime >= 1000) {
    Serial.printf("Potentiometer value: %d\n", mappedValue);
    lastSerialPrintTime = currentMillis;
  }

  SerialBT.printf("Potentiometer value: %d\n", mappedValue);

  delayMicroseconds(100);
}
```

Slave code:

```
#include "BluetoothSerial.h"
#include <Servo.h>

String device_name = "ESP32-BT-Slave";

// Check if Bluetooth configurations are enabled
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run make menuconfig to enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;
Servo myServo;

void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name);
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());

  #ifdef USE_PIN
    SerialBT.setPin(pin);
    Serial.println("Using PIN");
  #endif
  myServo.attach(13);
}

void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }

  if (SerialBT.available()) {
    char command = SerialBT.read();
    Serial.write(command);

    if (command == '0') {
      myServo.write(0);
      Serial.println("Servo Position: 0 degrees");
    } else if (command == '1') {
      myServo.write(90);
      Serial.println("Servo Position: 90 degrees");
    } else if (command == '2')
 myServo.write(180);
      Serial.println("Servo Position: 180 degrees");
    }
  }

  delayMicroseconds(100);
```

Overview:

This project enables two-way Bluetooth communication between two ESP32 modules to control a servo motor based on the input from a potentiometer.

- Master ESP32:
    - Reads the potentiometer values.
    - Maps the values to servo positions (0° to 180°).
    - Sends the mapped values to the Slave via Bluetooth.
- Slave ESP32:
    - Receives the potentiometer data from the Master.
    - Moves the servo motor to the corresponding angle based on the received values.

In summary, the Master reads the potentiometer, sends the data to the Slave, and the Slave adjusts the servo accordingly.

Two way communication in esp32 bluetooth module stepper motor controlling using potentio meter

Components:

- 2 ESP32 Modules (Master & Slave)
- Stepper Motor (on Slave)
- TB6600 Motor Driver (on Slave)
- 10k Potentiometer (on Master)
- Jumper wires, breadboard, power supply

Libraries :

BluetoothSerial:

#include "BluetoothSerial.h"

AccelStepper:

#include <AccelStepper.h>

Master code:

```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

String slaveName = "ESP32-BT-Slave";
String MACadd = "94:E6:86:3D:75:2E";
uint8_t address[6] = {0x94, 0xE6, 0x86, 0x3D, 0x75, 0x2E};

String myName = "ESP32-BT-Master";
const int potPin = 34;

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

  SerialBT.begin(myName, true);
  Serial.printf("The device \"%s\" started in master mode. Make sure the slave BT device is on!\n", myName.c_str());

  bool connected = false;

  #ifdef USE_NAME
    connected = SerialBT.connect(slaveName);
    Serial.printf("Connecting to slave BT device named \"%s\"\n", slaveName.c_str());
  #else
    connected = SerialBT.connect(address);
    Serial.printf("Connecting to slave BT device with MAC %s\n", MACadd.c_str());
```

```
  #endif

  if (connected) {
    Serial.println("Connected Successfully!");
  } else {
    while (!SerialBT.connected(10000)) {
      Serial.println("Failed to connect. Make sure the remote device is available and in range, then restart.");
    }
  }
}

void loop() {
  static unsigned long lastSerialPrintTime = 0;
  unsigned long currentMillis = millis();

  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }

  int potValue = analogRead(potPin);
  int mappedValue = map(potValue, 0, 4095, 0, 360);

  if (currentMillis - lastSerialPrintTime >= 1000) {
    Serial.printf("Potentiometer value: %d\n", mappedValue);
    lastSerialPrintTime = currentMillis;
  }

  SerialBT.printf("%d\n", mappedValue);
  delay(100);}
```

Slave code:

```
#include <AccelStepper.h>
#include "BluetoothSerial.h"

String device_name = "ESP32-BT-Slave";

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run make menuconfig to enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

// Motor driver pins
const int dirPin = 13;
const int stepPin = 14;
const int potPin = 34;

const int threshold = 10;
int lastPosition = 0;


AccelStepper stepper(AccelStepper::DRIVER, stepPin, dirPin);

void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name);
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());


  stepper.setMaxSpeed(1000);
  stepper.setAcceleration(500);
}

void loop() {
  if (SerialBT.available()) {
    String receivedData = SerialBT.readStringUntil('\n');
    Serial.printf("Received: %s\n", receivedData.c_str());

    int potValue = receivedData.toInt();
    int targetAngle = map(potValue, 0, 360, 0, 360);
    int targetPosition = (long)targetAngle * 200L / 360L;

    if (abs(targetPosition - lastPosition) > threshold) {
      stepper.moveTo(targetPosition);
      lastPosition = targetPosition;
    }
```

```
 }

  stepper.run();
 delay(10);}
```

Connections:

Master esp32 connections:

Middle pin (Signal) → GPIO 34 (Analog Input)
One outer pin → 3.3V (VCC)
Other outer pin → GND

Slave esp32 :

TB6600 Driver to ESP32:

- PUL+ (Step pin) → GPIO 14 (Step Signal)
- DIR+ (Direction pin) → GPIO 13 (Direction Signal)
- PUL- (GND) → GND of ESP32
- DIR- (GND) → GND of ESP32
- ENA+ (Optional Enable pin) → GPIO (optional, can leave disconnected for testing)
- ENA- (GND) → GND of ESP32

TB6600 Power Connections:

- **VCC (from Power Supply)** → 9-40V DC input (depending on your stepper motor voltage)
- **GND** → Ground of Power Supply

Stepper Motor:

- Connect the motor's A+, A-, B+, B- to the corresponding outputs on the TB6600 driver.

Circuit connections:

OVERVIEW:This overview clearly separates the master ESP32 responsible for reading the potentiometer and transmitting data via Bluetooth from the slave ESP32, which controls the stepper motor using the received data. The master reads the analog signal from the potentiometer and transmits it via Bluetooth, while the slave receives that signal and moves the stepper motor accordingly.

Two way communication in esp32 bluetooth module stepper motor controlling using encoder

Components

- 2 ESP32 Modules (Master & Slave)
- Stepper Motor (on Slave)
- TB6600 Motor Driver (on Slave)
- Rotary Encoder
- Jumper wires, breadboard, power supply

Libraries :
BluetoothSerial:
#include "BluetoothSerial.h"
AccelStepper:
#include <AccelStepper.h>

Master code:

```
 #include "BluetoothSerial.h"

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

String slaveName = "ESP32-BT-Slave";
String MACadd = "94:E6:86:3D:75:2E";
uint8_t address[6] = {0x94, 0xE6, 0x86, 0x3D, 0x75, 0x2E};

String myName = "ESP32-BT-Master";
const int encoderPinA = 34;
const int encoderPinB = 35;

int encoderPos = 0;
int lastEncoded = 0;

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

  SerialBT.begin(myName, true);
  Serial.printf("The device \"%s\" started in master mode. Make sure the slave BT device is on!\n", myName.c_str());

  bool connected = false;

  #ifdef USE_NAME
    connected = SerialBT.connect(slaveName);
```

```
    Serial.printf("Connecting to slave BT device named \"%s\"\n", slaveName.c_str());
  #else
    connected = SerialBT.connect(address);
    Serial.printf("Connecting to slave BT device with MAC %s\n", MACadd.c_str());
  #endif

  if (connected) {
    Serial.println("Connected Successfully!");
  } else {
    while (!SerialBT.connected(10000)) {
      Serial.println("Failed to connect. Make sure the remote device is available and in range, then restart.");
    }
  }

  pinMode(encoderPinA, INPUT);
  pinMode(encoderPinB, INPUT);
  attachInterrupt(digitalPinToInterrupt(encoderPinA), updateEncoder, CHANGE);
  attachInterrupt(digitalPinToInterrupt(encoderPinB), updateEncoder, CHANGE);
}

void loop() {
  static unsigned long lastSerialPrintTime = 0;
  unsigned long currentMillis = millis();

  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }

  int mappedValue = map(encoderPos, 0, 1000, 0, 500);

  if (currentMillis - lastSerialPrintTime >= 1000) {
    Serial.printf("Encoder value: %d\n", mappedValue);
    //Serial.printf("   pos value: %d\n", encoderPos);
    lastSerialPrintTime = currentMillis;
  }

  SerialBT.printf("%d\n", mappedValue);
  delay(100);
}

void updateEncoder() {
  int MSB = digitalRead(encoderPinA);
  int LSB = digitalRead(encoderPinB);
  int encoded = (MSB << 1) | LSB;
  int sum = (lastEncoded << 2) | encoded;

  if (sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) encoderPos++;
  if (sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) encoderPos--;

  lastEncoded = encoded;
}
```

Slave code:

```
#include <AccelStepper.h>
#include "BluetoothSerial.h"

String device_name = "ESP32-BT-Slave";

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run make menuconfig to enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

// Motor driver pins
const int dirPin = 13;
const int stepPin = 14;
const int enablePin = 12;
const int threshold = 1;
int lastPosition = 0;
AccelStepper stepper(AccelStepper::DRIVER, stepPin, dirPin);

void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name);
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());

  pinMode(enablePin, OUTPUT);
  digitalWrite(enablePin, LOW);

  stepper.setMaxSpeed(1000);
  stepper.setAcceleration(500);
}

void loop() {
  if (SerialBT.available()) {
    String receivedData = SerialBT.readStringUntil('\n');
    Serial.printf("Received: %s\n", receivedData.c_str());

    int targetAngle = receivedData.toInt();
    int targetPosition = map(targetAngle, 0, 100, 0, 100);

      if (abs(targetPosition - lastPosition) > threshold) {
      stepper.moveTo(targetPosition);
      lastPosition = targetPosition;

//      if (abs(targetAngle - lastPosition) > threshold) {
//      stepper.moveTo(targetAngle);
//      lastPosition = targetAngle;
    }
  }
```

```
  stepper.run();
  delay(10);
}
```

Connections:

-

Master ESP32	Connection	Component
GPIO 34	CLK Pin of Encoder (A Pin)	RotaryEncodr
GPIO 35	DT Pin of Encoder (B Pin)	RotaryEncodr
3.3V	VCC Pin of Encoder	RotaryEncodr
GND	GND Pin of Encoder	RotaryEncodr

-

Slave ESP32	Connection	Component
GPIO 13	DIR+ on TB6600	TB6600Driver
GPIO 14	PUL+ (STEP+) on TB6600	TB6600Driver
GND	DIR-, PUL-, ENA- on TB6600	TB6600Driver
External Power	VCC and GND on TB6600	Power Supply

OVERVIEW:

This project uses two ESP32 modules to control a stepper motor via Bluetooth:

- **Master ESP32**: Reads a rotary encoder's position and sends the data via Bluetooth.
- **Slave ESP32**: Receives the encoder values and controls a stepper motor using a TB6600 driver, adjusting motor rotation based on the encoder's position.

Two way communication in esp32 bluetooth module two stepper motors controlling using two encoders

Components:

ESP32 Development Boards (x2)
Rotary Encoders (x2)
TB6600 Stepper Motor Drivers (x2)
Stepper Motors (NEMA 17 or similar) (x2)
Power Supply for TB6600 Drivers (12V or 24V)
Jumper Wires and Breadboard
USB Cables for programming the ESP32 modules

Master code:

```
 #include "BluetoothSerial.h"

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

String slaveName = "ESP32-BT-Slave";
String MACadd = "94:E6:86:3D:75:2E";
uint8_t address[6] = {0x94, 0xE6, 0x86, 0x3D, 0x75, 0x2E};

String myName = "ESP32-BT-Master";
const int encoderPinA1 = 34;
const int encoderPinB1 = 35;
const int encoderPinA2 = 32;
const int encoderPinB2 = 33;

int encoderPos1 = 0;
int encoderPos2 = 0;
int lastEncoded1 = 0;
int lastEncoded2 = 0;

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

  SerialBT.begin(myName, true);
  Serial.printf("The device \"%s\" started in master mode. Make sure the slave BT device is on!\n", myName.c_str());

  bool connected = false;

  #ifdef USE_NAME
    connected = SerialBT.connect(slaveName);
    Serial.printf("Connecting to slave BT device named \"%s\"\n", slaveName.c_str());
```

```
  #else
    connected = SerialBT.connect(address);
    Serial.printf("Connecting to slave BT device with MAC %s\n", MACadd.c_str());
  #endif

  if (connected) {
    Serial.println("Connected Successfully!");
  } else {
    while (!SerialBT.connected(10000)) {
      Serial.println("Failed to connect. Make sure the remote device is available and in range, then restart.");
    }
  }

  pinMode(encoderPinA1, INPUT);
  pinMode(encoderPinB1, INPUT);
  pinMode(encoderPinA2, INPUT);
  pinMode(encoderPinB2, INPUT);

  attachInterrupt(digitalPinToInterrupt(encoderPinA1), updateEncoder1, CHANGE);
  attachInterrupt(digitalPinToInterrupt(encoderPinB1), updateEncoder1, CHANGE);
  attachInterrupt(digitalPinToInterrupt(encoderPinA2), updateEncoder2, CHANGE);
  attachInterrupt(digitalPinToInterrupt(encoderPinB2), updateEncoder2, CHANGE);
}

void loop() {
  static unsigned long lastSerialPrintTime = 0;
  unsigned long currentMillis = millis();

  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }

  int mappedValue1 = map(encoderPos1, 0, 1000, 0, 500);
  int mappedValue2 = map(encoderPos2, 0, 1000, 0, 500);

  if (currentMillis - lastSerialPrintTime >= 1000) {
    Serial.printf("Encoder1 value: %d, Encoder2 value: %d\n", mappedValue1, mappedValue2);
    lastSerialPrintTime = currentMillis;
  }

  SerialBT.printf("%d,%d\n", mappedValue1, mappedValue2);
  delay(100);
}

void updateEncoder1() {
  int MSB = digitalRead(encoderPinA1);
  int LSB = digitalRead(encoderPinB1);
  int encoded = (MSB << 1) | LSB;
  int sum = (lastEncoded1 << 2) | encoded;

  if (sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) encoderPos1++;
  if (sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) encoderPos1--;

  lastEncoded1 = encoded;
```

```
}

void updateEncoder2() {
  int MSB = digitalRead(encoderPinA2);
  int LSB = digitalRead(encoderPinB2);
  int encoded = (MSB << 1) | LSB;
  int sum = (lastEncoded2 << 2) | encoded;

  if (sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) encoderPos2++;
  if (sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) encoderPos2--;

  lastEncoded2 = encoded;
}
```

Slave code:

```
#include <AccelStepper.h>
#include "BluetoothSerial.h"

String device_name = "ESP32-BT-Slave";

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run make menuconfig to enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

const int dirPin1 = 13;
const int stepPin1 = 14;
const int enablePin1 = 12;

const int dirPin2 = 26;
const int stepPin2 = 27;
const int enablePin2 = 25;

int lastPosition1 = 0;
int lastPosition2 = 0;

AccelStepper stepper1(AccelStepper::DRIVER, stepPin1, dirPin1);
AccelStepper stepper2(AccelStepper::DRIVER, stepPin2, dirPin2);

void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name);
  Serial.printf("The device with name \"%s\" is started.\nNow you can pair it with Bluetooth!\n", device_name.c_str());

  pinMode(enablePin1, OUTPUT);
  digitalWrite(enablePin1, LOW);
  pinMode(enablePin2, OUTPUT);
  digitalWrite(enablePin2, LOW);

  stepper1.setMaxSpeed(1000);
```

```
  stepper1.setAcceleration(500);
  stepper2.setMaxSpeed(1000);
  stepper2.setAcceleration(500);
}

void loop() {
  if (SerialBT.available()) {
    String receivedData = SerialBT.readStringUntil('\n');
    Serial.printf("Received: %s\n", receivedData.c_str());

    int commaIndex = receivedData.indexOf(',');
    int encoderValue1 = receivedData.substring(0, commaIndex).toInt();
    int encoderValue2 = receivedData.substring(commaIndex + 1).toInt();

    int targetPosition1 = encoderValue1;
    int targetPosition2 = encoderValue2;

    if (targetPosition1 != lastPosition1) {
      stepper1.moveTo(targetPosition1);
      lastPosition1 = targetPosition1;
    }

    if (targetPosition2 != lastPosition2) {
      stepper2.moveTo(targetPosition2);
      lastPosition2 = targetPosition2;
    }
  }

  stepper1.run();
  stepper2.run();

  delay(10);
}
```

OVERVIEW:

In this project, two ESP32 modules communicate via Bluetooth to control two stepper motors using input from two encoders:

- Master ESP32: Reads the position of two encoders connected to its GPIO pins. It sends the encoder values to the slave ESP32 via Bluetooth.
- Slave ESP32: Receives the encoder values from the master ESP32. Based on these values, it controls two stepper motors using TB6600 drivers, adjusting their rotation to match the encoder input.

This setup enables real-time, wireless control of the stepper motors based on the encoder positions, with seamless two-way Bluetooth communication between the ESP32 modules.

Two way communication in esp32 bluetooth module joystick position

Components:

ESP32 Development Boards (x2)
Joystick Module
Jumper Wires
Breadboard
Power Supply for the ESP32

Master code:

```
#include "BluetoothSerial.h"

#define USE_NAME
const char *pin = "1234";

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

#ifdef USE_NAME
  String slaveName = "ESP32-BT-Slave";
#else
  String MACadd = "94:E6:86:3D:75:2E";
  uint8_t address[6]  = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
#endif

String myName = "ESP32-BT-Master";

const int VRxPin = 34;
const int VRyPin = 35;
const int SWPin = 4;
int Min = 300;
int Max = 3700;

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

  SerialBT.begin(myName, true);
  Serial.printf("The device \"%s\" started in master mode, make sure slave BT device is on!\n", myName.c_str());

  #ifndef USE_NAME
    SerialBT.setPin(pin);
    Serial.println("Using PIN");
  #endif

  #ifdef USE_NAME
    connected = SerialBT.connect(slaveName);
```

```
  Serial.printf("Connecting to slave BT device named \"%s\"\n", slaveName.c_str());
 #else
  connected = SerialBT.connect(address);
  Serial.print("Connecting to slave BT device with MAC "); Serial.println(MACadd);
 #endif

 if (connected) {
  Serial.println("Connected Successfully!");
 } else {
  while (!SerialBT.connected(10000)) {
   Serial.println("Failed to connect. Make sure remote device is available and in range, then restart app.");
  }
 }

 SerialBT.connect();
 if (connected) {
  Serial.println("Reconnected Successfully!");
 } else {
  while (!SerialBT.connected(10000)) {
   Serial.println("Failed to reconnect. Make sure remote device is available and in range, then restart app.");
  }
 }

 pinMode(SWPin, INPUT_PULLUP);
}

void loop() {
 int xValue = analogRead(VRxPin);
 int yValue = analogRead(VRyPin);
 int switchState = digitalRead(SWPin);

 String message;

 if (xValue > Max && yValue > Max) {
  message = "5";
 } else if (xValue > Max && yValue < Min) {
  message = "6";
 } else if (xValue < Min && yValue > Max) {
  message = "7";
 } else if (xValue < Min && yValue < Min) {
  message = "8";
 } else if (xValue > Max) {
  message = "1";
 } else if (xValue < Min) {
  message = "2";
 } else if (yValue > Max) {
  message = "3";
 } else if (yValue < Min) {
  message = "4";
 } else {
  message = "0";
 }

 message += "," + String(switchState == LOW ? 1 : 0);
```

```
  SerialBT.println(message);

  Serial.println("Sent: " + message);

  delay(100);

}
```

Slave code:

```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please enable it in the menuconfig or build the app with Bluetooth support.
#endif

BluetoothSerial SerialBT;

const int ledPin = 2;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32-BT-Slave");
  pinMode(ledPin, OUTPUT);
  Serial.println("Bluetooth device started, ready to pair...");
}

void loop() {
  if (SerialBT.available()) {

    String receivedData = SerialBT.readStringUntil('\n');


    Serial.println("Received: " + receivedData);


    int separatorIndex = receivedData.indexOf(',');
    if (separatorIndex != -1) {
      String switchState = receivedData.substring(separatorIndex + 1);


      switch (switchState.toInt()) {
        case 1:
          digitalWrite(ledPin, HIGH);
          Serial.println("LED ON");
          break;

        case 0:
          digitalWrite(ledPin, LOW);
          Serial.println("LED OFF");
          break;

        default:
```

```
        Serial.println("Unknown switch state received.");
        break;
      }
    }
  }

  delay(100);
}
```

OVERVIEW:

1. Master ESP32 reads the joystick position and sends the values to the slave ESP32.
2. Slave ESP32 receives the data, parses it, and performs actions accordingly.

This setup allows real-time control based on joystick input, facilitating seamless two-way communication between the ESP32 modules over Bluetooth.