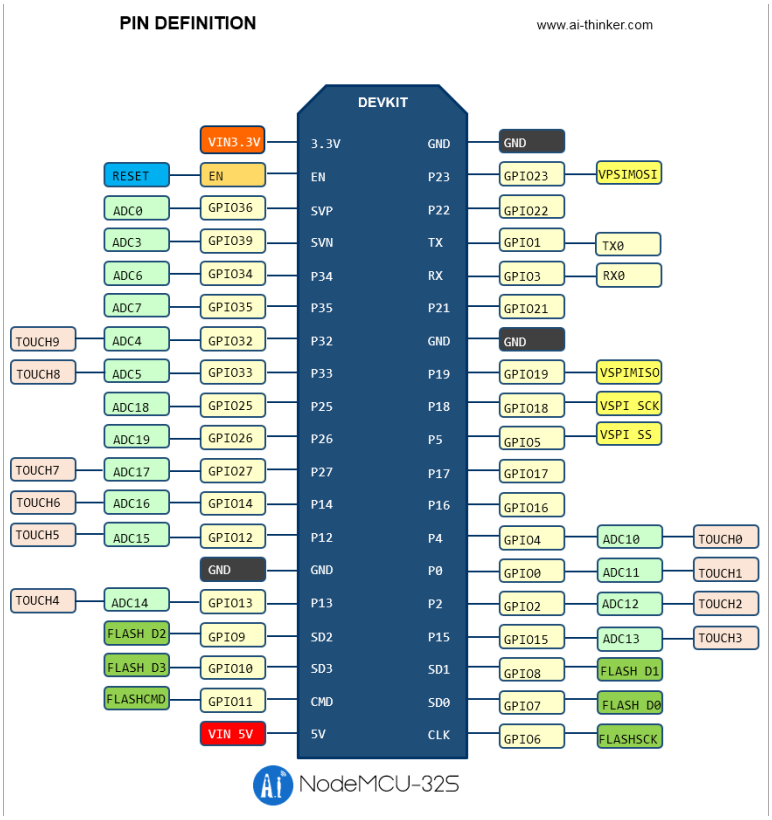
ESP32

구매링크



1. ESP32 CORE ARDUINO IDE 설치 방법

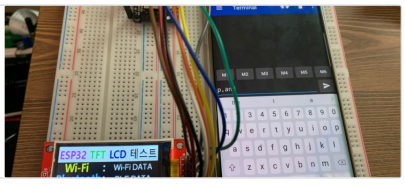
https://www.smart-prototyping.com/blog/How

2. 시리얼 통신을 앱을 사용한 ESP32 블루투스 통신

ESP32 블루투스 통신 예제 테스트

며칠 전부터 ESP32를 가지고 놀면서 TFT LCD 모니터를 연결하고 데이터를 출력할 준비를 했었다. ...

https://m.blog.naver.com/kwy1052aa/221773706276



3. 블루투스와 와이파이를 동시에 사용할 수 없다고..

4. 블루투스 통신 ESP32 c++코드, Flutter 코드

▼ esp32 코드

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLEServer* server;
BLECharacteristic* txCharacteristic;
BLECharacteristic* rxCharacteristic;

// BLEServerCallbacks 클래스 상속
class MyServerCallbacks : public BLEServerCallbacks {
  // 클라이언트가 연결되면 호출되는 콜백 함수
  void onConnect(BLEServer* pServer) {
    Serial.println("Client connected");
  }

  // 클라이언트가 연결 해제되면 호출되는 콜백 함수
  void onDisconnect(BLEServer* pServer) {
    Serial.println("Client disconnected");
  }
};

// BLECharacteristicCallbacks 클래스 상속
class MyCallbacks : public BLECharacteristicCallbacks {
  // 속성이 쓰여지면 호출되는 콜백 함수
  void onWrite(BLECharacteristic* pCharacteristic) {
    // 속성의 값을 가져옴
    std::string rxValue = pCharacteristic->getValue();

    // 값이 존재하면 로그 출력
    if (rxValue.length() > 0) {
      Serial.println("Received data: " + rxValue);
    }
  }
};

void setup() {
  Serial.begin(115200);
  delay(1000);

  // BLE 디바이스 초기화
  BLEDevice::init("ESP32 BLE");

  // BLE 서버 생성
  server = BLEDevice::createServer();

  // BLE 서버 콜백 함수 설정
  server->setCallbacks(new MyServerCallbacks());

  // BLE 서비스 생성
  BLEService* service = server->createService("6E400001-B5A3-F393-E0A9-E50E24DCCA9E");

  // BLE 특성 생성
  // NOTIFY 속성을 가진 txCharacteristic 생성
  txCharacteristic = service->createCharacteristic("6E400002-B5A3-F393-E0A9-E50E24DCCA9E", BLECharacteristic::PROPERTY_NOTIFY);
  // 클라이언트에게 값을 전송하기 위한 디스크립터 추가
  txCharacteristic->addDescriptor(new BLE2902());

  // WRITE 속성을 가진 rxCharacteristic 생성
  rxCharacteristic = service->createCharacteristic("6E400003-B5A3-F393-E0A9-E50E24DCCA9E", BLECharacteristic::PROPERTY_WRITE);
  // rxCharacteristic 콜백 함수 설정
  rxCharacteristic->setCallbacks(new MyCallbacks());

  // 서비스 시작
  service->start();
  // 광고 시작
  server->getAdvertising()->start();
}

void loop() {
  delay(1000);

  // 전송할 데이터 생성
  std::string txValue = "Hello from ESP32!";
  // txCharacteristic에 데이터 설정
  txCharacteristic->setValue((uint8_t*)txValue.c_str(), txValue.length());
  // 클라이언트에게 데이터 전송
  txCharacteristic->notify();
}
```

▼ flutter 코드

```
import 'package:flutter/material.dart';
import 'package:flutter_blue/flutter_blue.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  final String _serviceUuid = '6E400001-B
```

5. 아두이노와 안드로이드 Bluetooth 통신하기

[안드로이드] 아두이노와 안드로이드 Bluetooth 통신하기

안드로이드 기기와 아두이노 보드간 블루투스 통신 코드를 소개하겠습니다. 아두이노는 블루투스 통신을 위해 HC-06을 사용하였습니다. 먼저 bluetooth 통신을 위해 connect 버튼을 누르면 버튼이벤트를 시작으로 블루투스 연결을 시도하였습니다. 코드를 차근차근 따라해보면 이해가 쉬울 것입니다. 구현한 소스는 안드로이드 기기에서 데이

https://ddangeun.tistory.com/59

6. 블루투스 자동 연결

- 페어링된 ESP32와 안드로이드 기기는 기기 간 연결 정보를 기억
- 한 번 기기가 페어링 되면, 일반적으로는 다시 페어링 과정을 거치지 않아도 사동으로 블루투스 연결이 됨
- 그러나 블루투스 연결이 끊어진 경우에는 다시 페어링 해야함

7. 안드로이드(flutter)와 블루투스 통신

▼ ESP32 코드

▼ c++

```
#include <BluetoothSerial.h>

BluetoothSerial SerialBT; // BluetoothSerial 객체 생성

void setup() {
  Serial.begin(115200); // 시리얼 통신 객체 생성 및 초기화
  SerialBT.begin("ESP32 Bluetooth Device"); // 블루투스 모듈 초기화

  // Wait for connection
  while (!SerialBT.connected()) { // 블루투스 연결 대기
    delay(100);
  }

  Serial.println("Bluetooth connected"); // 블루투스 연결 완료 메시지 출력
}

void loop() {
  if (SerialBT.available()) { // 블루투스로 데이터 수신 시
    // Read incoming data from Bluetooth
    String data = SerialBT.readString(); // 수신 데이터를 문자열로 읽어옴

    // Do something with the data
    Serial.println(data); // 수신된 데이터 시리얼 모니터에 출력
  }
}
```

▼ 설명

- 블루투스 통신을 위해 SerialBT 객체를 생성하고, begin() 함수를 사용하여 블루투스 연결을 초기화합니다. loop() 함수에서는 시리얼 모니터로부터 데이터를 수신하면 해당 데이터를 ESP32에서 연결된 블루투스 기기로 전송하고, ESP32에서 수신한 데이터가 있으면 해당 데이터를 시리얼 모니터에 출력합니다.

1. `#include <BluetoothSerial.h>` : BluetoothSerial 라이브러리를 include합니다.
2. `BluetoothSerial SerialBT;` : BluetoothSerial 객체인 SerialBT를 선언합니다.
3. `void setup()` : 초기화 함수로, 보통 프로그램이 시작될 때 한 번 실행됩니다.
4. `Serial.begin(115200);` : 시리얼 통신을 위한 설정을 합니다.
5. `SerialBT.begin("ESP32 Bluetooth Device");` : 블루투스 통신을 위한 설정을 합니다. "ESP32 Bluetooth Device"는 ESP32의 블루투스 이름으로 설정됩니다.
6. `void loop()` : 메인 루프로, 프로그램이 실행되는 동안 반복적으로 실행됩니다.
7. `if (SerialBT.available())` : ESP32에서 수신한 데이터가 있는 경우입니다.
8. `Serial.write(SerialBT.read());` : ESP32에서 수신한 데이터를 시리얼 모니터에 출력합니다.
9. `if (Serial.available())` : 시리얼 모니터에서 수신한 데이터가 있는 경우입니다.
10. `SerialBT.write(Serial.read());` : 시리얼 모니터에서 수신한 데이터를 ESP32로 전송합니다.

▼ Flutter 코드

▼ Dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Bluetooth Example',
      home: HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  BluetoothConnection _connection;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Bluetooth Example'),
      ),
      body: Center(
        child: ElevatedButton(
          child: Text('Connect to ESP32'),
          onPressed: _connectToESP32,
        ),
      ),
    );
  }

  void _connectToESP32() async {
    try {
      // Get the list of paired Bluetooth devices
      List<BluetoothDevice> devices = await FlutterBluetoothSerial.instance.getBondedDevices();

      // Find the ESP32 device by name
      String deviceName = "ESP32 Bluetooth Device";
      BluetoothDevice device = devices.firstWhere((d) => d.name == deviceName);

      // Establish a Bluetooth connection with the device
      BluetoothConnection connection = await BluetoothConnection.toAddress(device.address);

      setState(() {
        _connection = connection;
```

```
      });

      print('Bluetooth connection established');
    } catch (e) {
      print('Error connecting to Bluetooth device: $e');
    }
  }

  void _sendData() {
    if (_connection != null && _connection.isConnected) {
      // Send data to ESP32 via Bluetooth
      String data = "Hello from Flutter!";
      _connection.output.add(data.codeUnits);
      _connection.output.allSent.then((_) {
        print('Data sent: $data');
      });
    } else {
      print('Bluetooth not connected');
    }
  }

  @override
  void dispose() {
    if (_connection != null && _connection.isConnected) {
      _connection.finish();
    }
    super.dispose();
  }
}
```

▼ 설명

위 코드에서 `MY_UUID`
는 ESP32와 페어링할 때 사용되는 UUID입니다. ESP32와 페어링할 때도 동일한 UUID를 사용해야 합니다.

8. vscode를 사용한 개발환경 세팅

https://www.youtube.com/watch?v=kYdtKh7zydI