

ESP32와 Firebase 실시간 데이터베이스 연동



▣ 강의 개요

이 강의에서는 ESP32 보드를 사용하여 Google Firebase의 실시간 데이터베이스(Realtime Database)와 연동하는 방법을 학습합니다. Firebase는 Google의 모바일 애플리케이션 개발 플랫폼으로, iOS, Android, 웹 애플리케이션에서 데이터를 관리하는 다양한 서비스를 제공합니다.

학습 목표

- Firebase 프로젝트 생성 및 설정
- ESP32에서 Firebase 실시간 데이터베이스로 데이터 전송
- Firebase 실시간 데이터베이스에서 데이터 읽기
- Firebase 인증 시스템 이해 및 구현

ⓐ Firebase란?



Firebase는 "앱을 구축, 개선, 성장시키기 위한 도구 모음"으로, 개발자들이 일반적으로 직접 구축해야 하지만 실제로는 앱 경험 자체에 집중하고 싶어하는 서비스들의 상당 부분을 다룹니다. 여기에는 분석, 인증, 데이터베이스, 구성, 파일 저장, 푸시 메시징 등이 포함됩니다. 이러한 서비스들은 클라우드에서 호스팅되며, 개발자의 노력을 거의 들이지 않고도 확장됩니다.

Firebase의 주요 장점

- **완전 관리형 서비스:** 서버 관리 불필요
- **실시간 동기화:** 실시간으로 데이터 업데이트
- **확장성:** 자동 스케일링
- **보안:** 강력한 보안 규칙
- **크로스 플랫폼:** 다양한 플랫폼 지원

▣ 강의 목차

1. Firebase 프로젝트 설정

1.1 새 프로젝트 생성

- [Firebase Console](#)에서 프로젝트 생성
- 프로젝트 이름 설정 (예: ESP32-IoT-Project)
- Google Analytics 설정 (선택사항)

1.2 인증 방법 설정

Firebase의 인증 시스템은 사용자 신원을 확인하고 관리합니다. ESP32와 같은 IoT 디바이스의 경우, 이메일/패스워드 인증을 주로 사용합니다.

인증 설정 단계:

1. 왼쪽 사이드바에서 **Build > Authentication** 클릭
2. **Get started** 버튼 클릭
3. **Email/Password** 인증 방법 선택 및 활성화
4. **Users** 탭에서 **Add user** 클릭
5. 이메일과 패스워드로 새 사용자 생성

The screenshot shows the Firebase Authentication console for the project 'ESP-Project'. The left sidebar has a 'Build' dropdown and several services listed: App Check, App Hosting, **Authentication**, Data Connect, Extensions, Firestore Database, Functions, and Hosting. The 'Authentication' service is selected and highlighted with a red box. The main content area features a large 'Authentication' heading and a sub-heading: 'Authenticate and manage users from a variety of providers without server-side code'. Below this are two buttons: 'Get started' (highlighted with a red box) and 'Ask Gemini'. A stylized orange lightning bolt graphic is on the right.

Email/Password

Enable

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#)

Email link (passwordless sign-in)

Enable

Cancel

Save

Authentication

The following Authentication features will stop working when Firebase Dynamic Links shuts down on August 25, 2025: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Identifier	Providers	Created	Signed In	User UID
No users for this project yet				

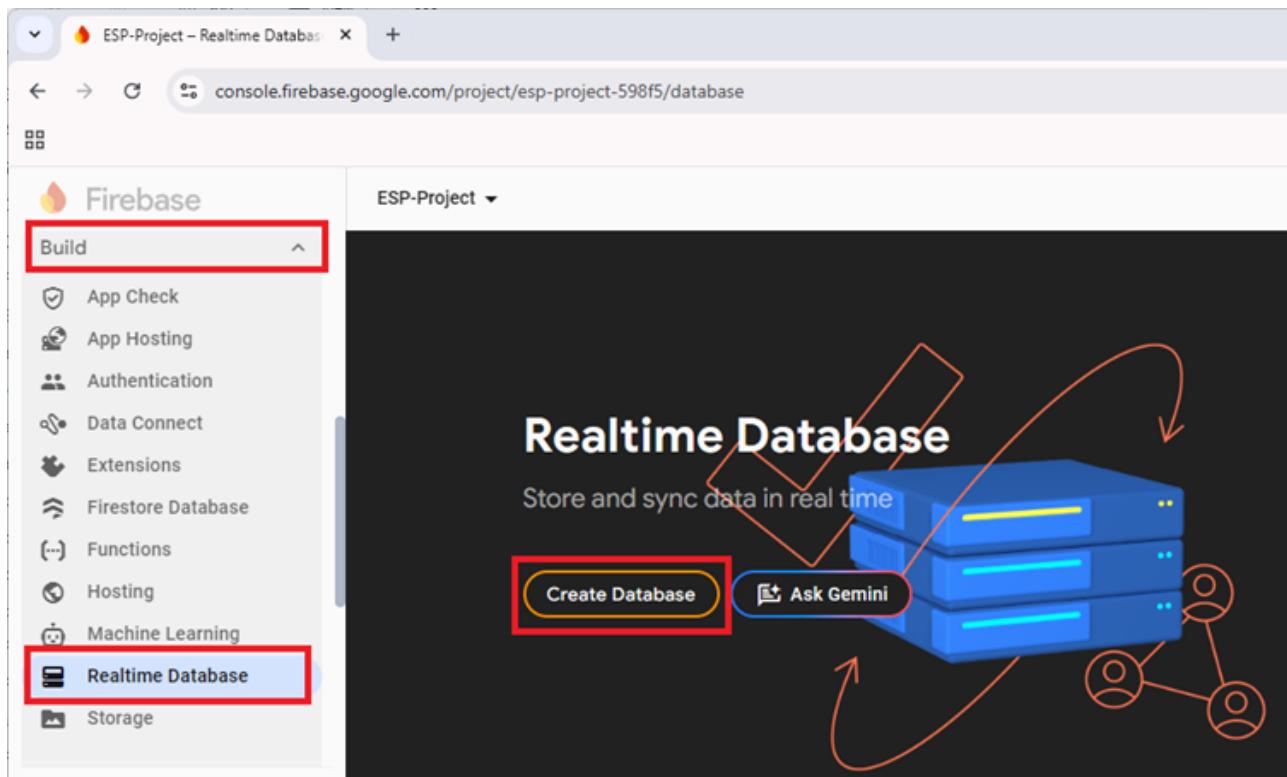
Identifier	Providers	Created	Signed in	User UID
[Redacted]	[Redacted]	19 Apr ...		fUTuDJ66U9RIJ...

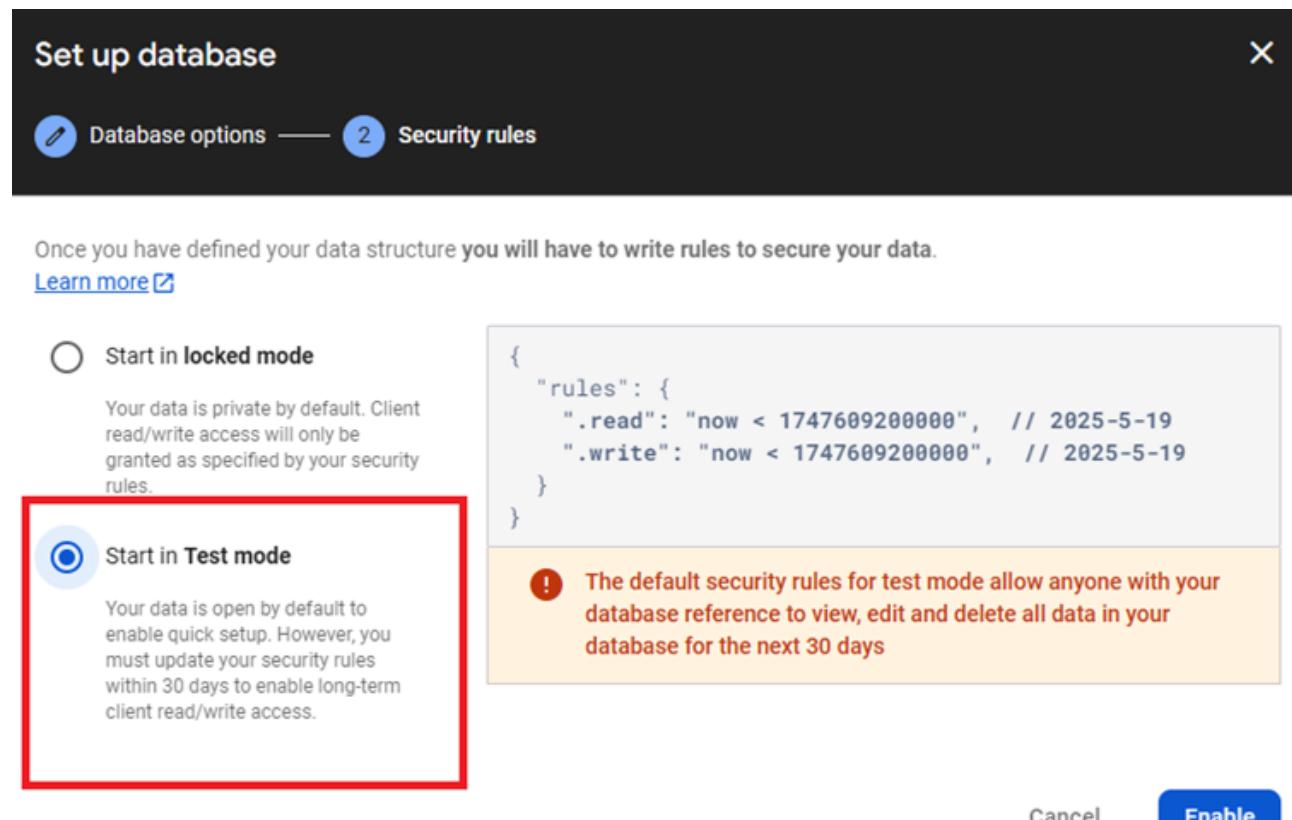
Rows per page: 50 | 1 - 1 of 1 | < >

1.3 실시간 데이터베이스 생성

1. Build > Realtime Database 선택
2. Create Database 클릭
3. 데이터베이스 위치 선택 (가장 가까운 지역)
4. Start in test mode 선택 (보안 규칙은 나중에 설정)

5. 데이터베이스 URL 복사 및 저장





ESP-Project ▾

Realtime Database

Need help with Realtime Database? Ask Gemini

Data Rules Backups Usage Extensions

Protect your Realtime Database resources from abuse, such as billing fraud or phishing [Configure App Check](#) X

https://esp-project-598f5-default-rtbd.firebaseio.west1.firebaseio.app

https://esp-project-598f5-default-rtbd.firebaseio.west1.firebaseio.app/:null

1.4 API 키 획득

1. 왼쪽 사이드바에서 **Project Settings** 클릭
 2. **General** 탭에서 **Web API Key** 복사 및 저장
2. ESP32 개발 환경 설정

2.1 필요한 라이브러리

- **FirebaseClient**: Firebase와의 통신을 위한 라이브러리
- **WiFi**: 네트워크 연결
- **WiFiClientSecure**: 보안 연결

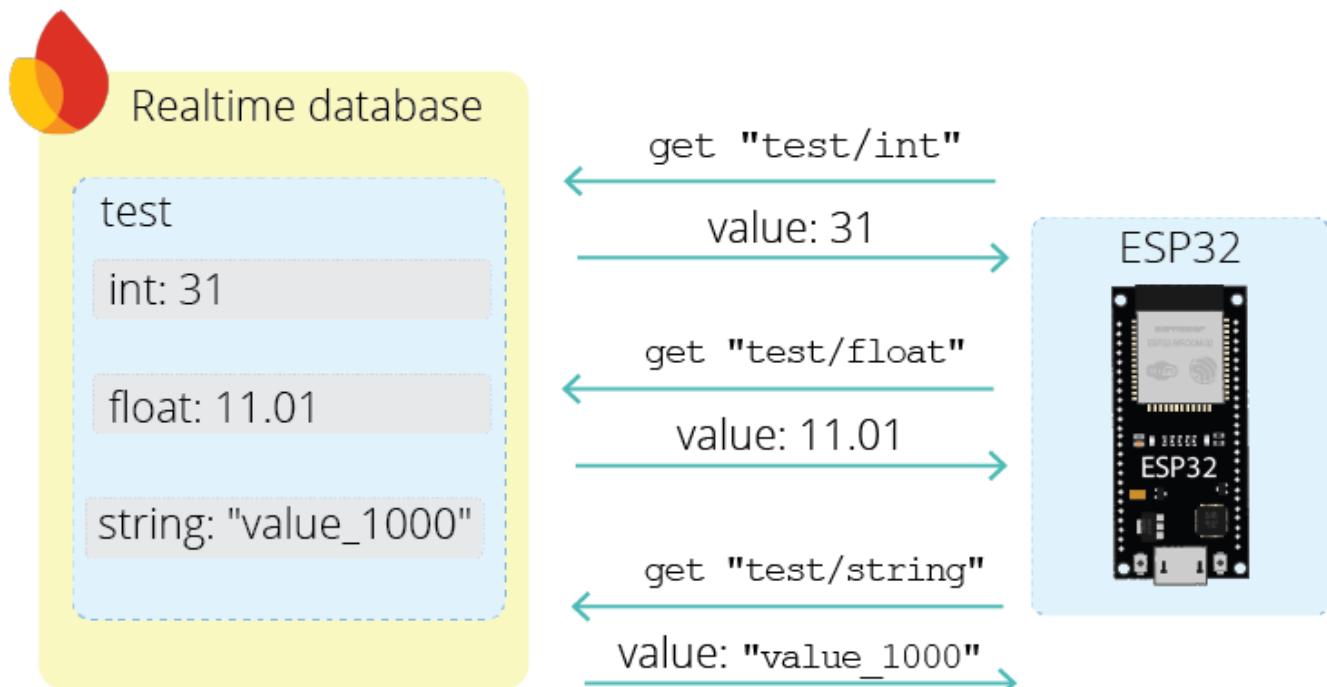
2.2 Arduino IDE 라이브러리 설치

스케치 > 라이브러리 포함 > 라이브러리 관리
 "FirebaseClient" 검색 후 Mobitz 제작 라이브러리 설치

2.3 VS Code + PlatformIO 라이브러리 설치

PIO Home > Libraries 탭 > "FirebaseClient" 검색
 Firebase Client Library by Mobitz 선택 후 프로젝트에 추가

3. ESP32에서 Firebase로 데이터 전송



ESP32는 다음과 같은 방식으로 Firebase에 데이터를 전송합니다:

1. **WiFi 연결**: ESP32가 인터넷에 연결
2. **Firebase 인증**: 이메일/패스워드로 사용자 인증
3. **데이터 전송**: 센서 데이터를 JSON 형태로 전송
4. **실시간 업데이트**: 데이터베이스가 실시간으로 업데이트

3.1 기본 코드 구조

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <FirebaseClient.h>

// 네트워크 및 Firebase 자격 증명
#define WIFI_SSID "YOUR_WIFI_SSID"
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"
#define WEB_API_KEY "YOUR_FIREBASE_API_KEY"
#define DATABASE_URL "YOUR_DATABASE_URL"
#define USER_EMAIL "YOUR_USER_EMAIL"
#define USER_PASS "YOUR_USER_PASSWORD"

// Firebase 구성 요소
UserAuth user_auth(WEB_API_KEY, USER_EMAIL, USER_PASS);
FirebaseApp app;
WiFiClientSecure ssl_client;
AsyncClient aClient(ssl_client);
RealtimeDatabase Database;
```

3.2 데이터 전송 함수

```
void sendDataToFirebase() {
    // 센서 데이터 준비
    int sensorValue = analogRead(A0);
    float temperature = 23.5;
    String deviceStatus = "online";

    // 데이터베이스에 전송
    Database.set<int>(aClient, "/sensor/value", sensorValue, processData,
    "Send_Sensor");
    Database.set<float>(aClient, "/sensor/temperature", temperature, processData,
    "Send_Temp");
    Database.set<String>(aClient, "/device/status", deviceStatus, processData,
    "Send_Status");
}
```

4. Firebase에서 데이터 읽기

The screenshot shows the Arduino Serial Monitor interface. The top bar includes tabs for 'Output' and 'Serial Monitor' (which is selected), and a message input field with placeholder text 'Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on ...)'. Below the message field are dropdown menus for 'New Line' and '115200 baud'. The main window displays a log of events from the ESP32. The log includes:

```

...
Connected with IP: 192.168.1.74

Event task: 🔒 authTask, msg: authenticating, code: 7
Debug task: 🔒 authTask, msg: Connecting to server...
Event task: 🔒 authTask, msg: auth request sent, code: 8
Event task: 🔒 authTask, msg: auth response received, code: 9
Debug task: 🔒 authTask, msg: Terminating the server connection...
Event task: 🔒 authTask, msg: ready, code: 10
Requested data from /test/int, /test/float, and /test/string
Debug task: RTDB.GetInt, msg: Connecting to server...
task: RTDB.GetInt, payload: 109
Stored intValue: 109
task: RTDB.GetFloat, payload: 60.01
Stored floatValue: 60.01

```

At the bottom of the monitor window, status information is shown: 'Ln 79, Col 1 DOIT ESP32 DEVKIT V1 on COM3' and a connection icon.

Firebase에서 데이터를 읽는 방법은 두 가지가 있습니다:

4.1 동기적 읽기

데이터를 요청하고 응답을 기다립니다.

```
void readDataSync() {
    String path = "/sensor/temperature";
    Database.get(aClient, path, processData, false, "Read_Temperature");
}
```

4.2 비동기적 읽기

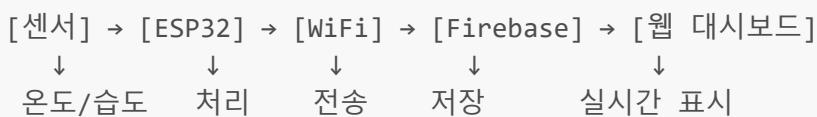
콜백 함수를 통해 데이터가 도착하면 처리합니다.

```
void readDataAsync() {
    Database.get(aClient, "/sensor/value", processData, false,
    "Read_Sensor_Async");
}

void processData(AsyncResult& aResult) {
    if (aResult.available()) {
        Serial.printf("받은 데이터: %s\n", aResult.c_str());
    }
}
```

5. 실시간 모니터링 시스템 구축

5.1 시스템 아키텍처



5.2 데이터 구조 설계

Firebase 실시간 데이터베이스는 JSON 트리 구조를 사용합니다:

```

{
  "devices": {
    "esp32_001": {
      "sensors": {
        "temperature": 23.5,
        "humidity": 65.2,
        "timestamp": 1640995200
      },
      "status": {
        "online": true,
        "last_seen": 1640995200,
        "battery_level": 87
      }
    }
  },
  "settings": {
    "update_interval": 10,
    "alert_threshold": 30
  }
}
  
```

6. 고급 기능

6.1 실시간 리스너 설정

데이터베이스 변경사항을 실시간으로 감지합니다.

```

void setupRealtimeListener() {
  Database.stream(aClient, "/commands", streamCallback, "Listen_Commands");
}

void streamCallback(AsyncResult<result> {
  if (result.isEvent()) {
    String command = result.c_str();
    executeCommand(command);
  }
}
  
```

6.2 배치 작업

여러 데이터를 한 번에 업데이트합니다.

```
void batchUpdate() {
    String json = "{";
    json += "\"sensor/temperature\": 24.1,";
    json += "\"sensor/humidity\": 68.5,";
    json += "\"device/timestamp\": " + String(millis());
    json += "}";

    Database.updateNode(aClient, "/", json, processData, "Batch_Update");
}
```

7. 보안 규칙 설정

테스트 모드를 벗어나 실제 운영을 위한 보안 규칙을 설정합니다.

```
{
  "rules": {
    "devices": {
      "$deviceId": {
        ".read": "auth != null && auth.uid == $deviceId",
        ".write": "auth != null && auth.uid == $deviceId"
      },
      "public": {
        ".read": true,
        ".write": false
      }
    }
  }
}
```

8. 트러블슈팅

8.1 일반적인 문제들

연결 실패

- WiFi 자격 증명 확인
- Firebase API 키 검증
- 네트워크 방화벽 설정

인증 오류

- 사용자 이메일/패스워드 확인
- Firebase 인증 설정 점검

데이터 전송 실패

- 데이터베이스 URL 확인
- 보안 규칙 검토
- JSON 형식 검증

8.2 디버깅 팁

```
void enableDebugMode() {
    Serial.begin(115200);
    Firebase.printf("디버그 모드 활성화\n");

    // 상세 로그 출력
    Database.setDebugLevel(1);
}
```

🔗 실습 프로젝트

프로젝트 1: IoT 온도 모니터링 시스템

- DHT22 센서로 온도/습도 측정
- Firebase에 실시간 데이터 전송
- 웹 대시보드에서 그래프로 시각화

프로젝트 2: 스마트 조명 제어

- Firebase를 통한 원격 LED 제어
- 스케줄링 및 자동화 기능
- 모바일 앱 연동

프로젝트 3: 환경 모니터링 스테이션

- 다중 센서 데이터 수집
- 알림 시스템 구현
- 데이터 분석 및 리포트

📘 추가 학습 자료

- [Firebase 공식 문서](#)
- [ESP32 Arduino 라이브러리](#)
- [FirebaseClient 라이브러리 GitHub](#)

🤝 기여하기

이 강의 자료에 기여하고 싶으시다면:

- 이 저장소를 포크하세요
- 새로운 브랜치를 만드세요 (`git checkout -b feature/AmazingFeature`)
- 변경사항을 커밋하세요 (`git commit -m 'Add some AmazingFeature'`)

4. 브랜치에 푸시하세요 (`git push origin feature/AmazingFeature`)

5. Pull Request를 생성하세요

📄 라이선스

이 프로젝트는 MIT 라이선스 하에 있습니다. 자세한 내용은 [LICENSE](#) 파일을 참조하세요.

📞 문의사항

강의 관련 문의사항이 있으시면 언제든 연락주세요:

- 이메일: your.email@example.com
- GitHub Issues: [이슈 생성](#)

Happy Coding! 🎉

이 강의는 Random Nerd Tutorials의 ESP32 Firebase 투토리얼을 기반으로 제작되었습니다.