Step-by-Step Guide: Connecting ESP32 to Firebase

📋 Prerequisites

- ✅ ESP32 connected to Wi-Fi (complete previous guide)
- 🔥 Firebase Project with Realtime Database or Firestore
- 💻 Arduino IDE with ESP32 support installed

📚 Install Required Libraries

1. Open Arduino IDE → **Sketch** → **Include Library** → **Manage Libraries**
2. Search for and install:
   - 📦 **Firebase ESP Client** by mobizt
   - 📦 **ArduinoJson** by Benoit Blanchon (dependency)

🔥 Firebase Project Setup

1. 🌐 Go to Firebase Console
2. 🆕 **Create a New Project** or select existing one
3. 🗄️ **Enable Realtime Database** or **Firestore**:
   - Realtime Database: Create database in test mode
   - Firestore: Create database in test mode
4. 🔑 **Get Project Credentials**:
   - Project Settings → Service Accounts → Generate new private key → Download JSON file
   - Project Settings → General → Copy **Project ID**
5. 🌐 **Get Database URL** (for Realtime Database):
   - Realtime Database → Copy URL (e.g., https://your-project-id.firebaseio.com)

💻 Code Implementation

Create a new sketch with this code (replace placeholders with your values):

cpp

```
#include <WiFi.h>
#include <FirebaseESP32.h>

// Wi-Fi credentials
#define WIFI_SSID "YOUR_WIFI_SSID"
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

// Firebase project credentials
#define FIREBASE_HOST "your-project-id.firebaseio.com"  // For RTDB
#define FIREBASE_AUTH "YOUR_DATABASE_SECRET" // Or use Service Account key

// For service account authentication (more secure):
// #define FIREBASE_AUTH PROJECT_ID, CLIENT_EMAIL, PRIVATE_KEY

// Firebase objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

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

  // Connect to Wi-Fi
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  // Firebase configuration
  config.host = FIREBASE_HOST;
  config.signer.tokens.legacy_token = FIREBASE_AUTH;

  // Alternatively for service account:
  // config.service_account.data.client_email = "your-service-account-email@project-id.iam.gserviceaccount.com";
  // config.service_account.data.project_id = "your-project-id";
  // config.service_account.data.private_key = "-----BEGIN PRIVATE KEY-----\\nyour_private_key\\n-----END PRIVATE KEY-----\\n";

  Firebase.reconnectNetwork(true);
  fbdo.setBSSLBufferSize(4096, 1024);
  fbdo.setResponseSize(2048);

  // Initialize Firebase
```

```
  Firebase.begin(&config, &auth);
  Firebase.setReadTimeout(fbdo, 1000 * 60);
  Firebase.setwriteSizeLimit(fbdo, "tiny");
}

void loop() {
  // Test write operation
  if (Firebase.setInt(fbdo, "/test/data", 42)) {
    Serial.println("Data written successfully!");
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
    Serial.println("Value: " + String(fbdo.intData()));
  } else {
    Serial.println("Error: " + fbdo.errorReason());
  }

  delay(5000);

  // Test read operation
  if (Firebase.getInt(fbdo, "/test/data")) {
    Serial.println("Read value: " + String(fbdo.intData()));
  } else {
    Serial.println("Read failed: " + fbdo.errorReason());
  }

  delay(5000);
```

}

⚙️ Configuration Details

1. **Authentication Methods**:
   - 🔑 **Legacy Token** (simpler):
     - Get from: Firebase Console → Project Settings → Service Accounts → Database Secrets
   - 🔐 **Service Account** (recommended):
     - Use the downloaded JSON file values for client_email, project_id, and private_key
     - Format private key with \n as shown in the JSON
2. **Database Rules** (temporary for testing):

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

   - Set in: Realtime Database → Rules (replace with secure rules for production)

🚀 Upload and Test

1. ⬆️ **Upload** the code to ESP32
2. 📊 **Open Serial Monitor** (115200 baud)
3. ✅ You should see:
   - Wi-Fi connection confirmation

    - Successful write/read operations
    - Data visible in Firebase Console

❗ Troubleshooting

- 🔐 **Authentication Failed**:
    - Verify project credentials
    - Check private key formatting (use \n for newlines)
- ⏰ **Connection Timeout**:
    - Check Wi-Fi strength
    - Verify Firebase host URL
- 🚫 **Permission Denied**:
    - Check database security rules

🎯 Next Steps

- Implement error handling
- Add data validation
- Set up proper security rules
- Explore other Firebase services (Firestore, Authentication, Storage)

Complete Android Kotlin Code for Firebase Integration

📱 Full Android Kotlin Implementation

build.gradle (Module: app)

gradle

```
plugins {
  id 'com.android.application'
```

```
    id 'org.jetbrains.kotlin.android'
    id 'com.google.gms.google-services'
}

android {
    namespace 'com.example.firebaseapp'
    compileSdk 34

    defaultConfig {
        applicationId "com.example.firebaseapp"
        minSdk 24
        targetSdk 34
        versionCode 1
        versionName "1.0"
    }

    buildFeatures {
        viewBinding true
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules.pro'
        }
```

```
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}

dependencies {
    implementation 'androidx.core:core-ktx:1.12.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.10.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'

    // Firebase
    implementation platform('com.google.firebase:firebase-bom:32.5.0')
    implementation 'com.google.firebase:firebase-analytics'
    implementation 'com.google.firebase:firebase-database-ktx'
    implementation 'com.google.firebase:firebase-auth'

    // Coroutines
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.7.3'
```

```
    // Lifecycle
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.6.2'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.2'

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

AndroidManifest.xml

xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.FirebaseApp"
        tools:targetApi="31">
```

```
    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

MainActivity.kt

kotlin

```
package com.example.firebaseapp


import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.firebaseapp.databinding.ActivityMainBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.google.firebase.database.ktx.database
```

```
import com.google.firebase.ktx.Firebase
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.tasks.await
import java.util.Date

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private lateinit var database: FirebaseDatabase
    private lateinit var auth: FirebaseAuth
    private lateinit var adapter: HistoryItemAdapter

    private var dataListener: ValueEventListener? = null
    private var historyListener: ValueEventListener? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        initializeFirebase()
        setupRecyclerView()
        setupClickListeners()
    }
```

```
private fun initializeFirebase() {
    auth = FirebaseAuth.getInstance()
    database = Firebase.database

    // Try anonymous authentication first
    auth.signInAnonymously().addOnCompleteListener { task ->
        if (task.isSuccessful) {
            updateConnectionStatus("Connected to Firebase", true)
            startListening()
        } else {
            updateConnectionStatus("Authentication failed", false)
            Log.e("FirebaseApp", "Authentication failed", task.exception)
        }
    }
}

private fun setupRecyclerView() {
    adapter = HistoryItemAdapter()
    binding.rvHistory.layoutManager = LinearLayoutManager(this)
    binding.rvHistory.adapter = adapter
}

private fun setupClickListeners() {
    binding.btnUpdateValue.setOnClickListener {
        updateData()
```

```
    }

    binding.btnStartListening.setOnClickListener {
        startListening()
    }

    binding.btnStopListening.setOnClickListener {
        stopListening()
    }
}

private fun startListening() {
    stopListening() // Clean up any existing listeners

    // Listen to the main data path (same as ESP32)
    dataListener = database.getReference("test/data").addValueEventListener(
        object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val value = snapshot.getValue(Int::class.java) ?: 0
                updateCurrentValue(value)
            }

            override fun onCancelled(error: DatabaseError) {
                Log.e("FirebaseApp", "Data listener cancelled", error.toException())
            }
        }
```

```
    )

    // Listen to history updates
    historyListener = database.getReference("test/history").addValueEventListener(
        object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val historyItems = mutableListOf<HistoryItem>()
                snapshot.children.forEach { child ->
                    val value = child.child("value").getValue(Int::class.java) ?: 0
                    val timestamp = child.child("timestamp").getValue(Long::class.java) ?: 0
                    val source = child.child("source").getValue(String::class.java) ?: "Unknown"
                    historyItems.add(HistoryItem(value, timestamp, source))
                }
                adapter.updateItems(historyItems)
            }

            override fun onCancelled(error: DatabaseError) {
                Log.e("FirebaseApp", "History listener cancelled", error.toException())
            }
        }
    )

    updateConnectionStatus("Listening for updates...", true)
}

private fun stopListening() {
```

```
    dataListener?.let { database.getReference("test/data").removeEventListener(it) }
    historyListener?.let { database.getReference("test/history").removeEventListener(it) }
    dataListener = null
    historyListener = null
    updateConnectionStatus("Stopped listening", false)
}

private fun updateData() {
    val newValueStr = binding.etNewValue.text.toString()
    if (newValueStr.isBlank()) {
        Toast.makeText(this, "Please enter a value", Toast.LENGTH_SHORT).show()
        return
    }

    val newValue = newValueStr.toIntOrNull()
    if (newValue == null) {
        Toast.makeText(this, "Please enter a valid number", Toast.LENGTH_SHORT).show()
        return
    }

    CoroutineScope(Dispatchers.IO).launch {
        try {
            // Update main data value
            database.getReference("test/data").setValue(newValue).await()

            // Add to history
```

```
            val historyRef = database.getReference("test/history").push()
            val historyData = mapOf(
                "value" to newValue,
                "timestamp" to System.currentTimeMillis(),
                "source" to "Android App"
            )
            historyRef.setValue(historyData).await()

            runOnUiThread {
                Toast.makeText(this@MainActivity, "Value updated successfully",
Toast.LENGTH_SHORT).show()
                binding.etNewValue.text.clear()
            }
        } catch (e: Exception) {
            runOnUiThread {
                Toast.makeText(this@MainActivity, "Update failed: ${e.message}",
Toast.LENGTH_SHORT).show()
                Log.e("FirebaseApp", "Update failed", e)
            }
        }
    }
}

private fun updateCurrentValue(value: Int) {
    binding.tvCurrentValue.text = "Current Value: $value"
    binding.tvLastUpdate.text = "Last Update: ${getCurrentTime()}"
}
```

```
    private fun updateConnectionStatus(message: String, isConnected: Boolean) {
        binding.tvConnectionStatus.text = message
        binding.tvConnectionStatus.setTextColor(
            if (isConnected) getColor(android.R.color.holo_green_dark)
            else getColor(android.R.color.holo_red_dark)
        )
    }

    private fun getCurrentTime(): String {
        return android.text.format.DateFormat.format("dd/MM/yyyy HH:mm:ss",
Date()).toString()
    }

    override fun onDestroy() {
        super.onDestroy()
        stopListening()
    }
}
```

HistoryItemAdapter.kt

kotlin

```
package com.example.firebaseapp

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

data class HistoryItem(
    val value: Int,
    val timestamp: Long,
    val source: String
)

class HistoryItemAdapter : RecyclerView.Adapter<HistoryItemAdapter.ViewHolder>() {

    private val items = mutableListOf<HistoryItem>()

    fun updateItems(newItems: List<HistoryItem>) {
        items.clear()
        items.addAll(newItems.sortedByDescending { it.timestamp })
        notifyDataSetChanged()
    }

    fun addItem(item: HistoryItem) {
        items.add(0, item)
        notifyItemInserted(0)
    }
```

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
    val view = LayoutInflater.from(parent.context)
        .inflate(R.layout.item_history, parent, false)
    return ViewHolder(view)
}

override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val item = items[position]
    holder.bind(item)
}

override fun getItemCount(): Int = items.size

class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    private val tvValue: TextView = itemView.findViewById(R.id.tvValue)
    private val tvTime: TextView = itemView.findViewById(R.id.tvTime)
    private val tvSource: TextView = itemView.findViewById(R.id.tvSource)

    fun bind(item: HistoryItem) {
        tvValue.text = "Value: ${item.value}"
        tvTime.text = formatTime(item.timestamp)
        tvSource.text = "From: ${item.source}"
    }

    private fun formatTime(timestamp: Long): String {
```

```
        val sdf = SimpleDateFormat("dd/MM/yyyy HH:mm:ss", Locale.getDefault())
        return sdf.format(Date(timestamp))
      }
    }
}
```

activity_main.xml

xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <!-- Connection Status -->
        <TextView
            android:id="@+id/tvConnectionStatus"
            android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:text="Disconnected"
    android:textColor="@color/red"
    android:textSize="16sp"
    android:textStyle="bold"
    android:layout_marginBottom="16dp"/>

<!-- ESP32 Data Section -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="ESP32 Data"
    android:textSize="18sp"
    android:textStyle="bold"
    android:layout_marginBottom="8dp"/>

<androidx.cardview.widget.CardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    app:cardElevation="4dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
```

```
        android:padding="16dp">

        <TextView
            android:id="@+id/tvCurrentValue"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Current Value: --"
            android:textSize="16sp"/>

        <TextView
            android:id="@+id/tvLastUpdate"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Last Update: --"
            android:textSize="14sp"
            android:layout_marginTop="8dp"/>

    </LinearLayout>
</androidx.cardview.widget.CardView>

<!-- Update Data Section -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Update Data"
    android:textSize="18sp"
```

```
    android:textStyle="bold"
    android:layout_marginBottom="8dp"/>

<EditText
    android:id="@+id/etNewValue"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Enter new value"
    android:inputType="number"
    android:layout_marginBottom="8dp"/>

<Button
    android:id="@+id/btnUpdateValue"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Update Value"
    android:layout_marginBottom="16dp"/>

<!-- History Section -->
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Update History"
    android:textSize="18sp"
    android:textStyle="bold"
    android:layout_marginBottom="8dp"/>
```

```
    <androidx.recyclerview.widget.RecyclerView
      android:id="@+id/rvHistory"
      android:layout_width="match_parent"
      android:layout_height="300dp"
      android:layout_marginBottom="16dp"/>

    <!-- Control Buttons -->
    <Button
      android:id="@+id/btnStartListening"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Start Listening"
      android:layout_marginBottom="8dp"/>

    <Button
      android:id="@+id/btnStopListening"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Stop Listening"
      android:layout_marginBottom="16dp"/>

  </LinearLayout>
</ScrollView>
```

item_history.xml

xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginBottom="8dp"
  app:cardElevation="2dp">

  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="12dp">

    <TextView
      android:id="@+id/tvValue"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textSize="16sp"
      android:textStyle="bold"/>

    <TextView
      android:id="@+id/tvTime"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
```

```
        android:textSize="14sp"
        android:layout_marginTop="4dp"/>

    <TextView
        android:id="@+id/tvSource"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="12sp"
        android:textColor="@android:color/darker_gray"
        android:layout_marginTop="2dp"/>

  </LinearLayout>
</androidx.cardview.widget.CardView>
```

Firebase Database Rules

```
json
{
  "rules": {
    "test": {
      ".read": "auth != null",
      ".write": "auth != null",
      "data": {
        ".validate": "newData.isNumber()"
      },
      "history": {
        "$pushId": {
          "value": {
```

```
        ".validate": "newData.isNumber()"
      },
      "timestamp": {
        ".validate": "newData.isNumber()"
      },
      "source": {
        ".validate": "newData.isString()"
      }
    }
  }
 }
}
}
```

📱 **Features Included:**

- ✅ Real-time data synchronization with ESP32
- ✅ Update values from Android app
- ✅ View update history with timestamps
- ✅ Connection status monitoring
- ✅ Error handling and validation
- ✅ Clean Material Design UI

🔧 **Setup Instructions:**

1. Add your google-services.json file to the app module
2. Update Firebase database rules as shown above
3. Make sure your ESP32 is using the same database paths (/test/data and /test/history)

4. Run the app on your Android device/emulator

The app will automatically connect to Firebase and start listening for updates from your ESP32!