

Internet of Things

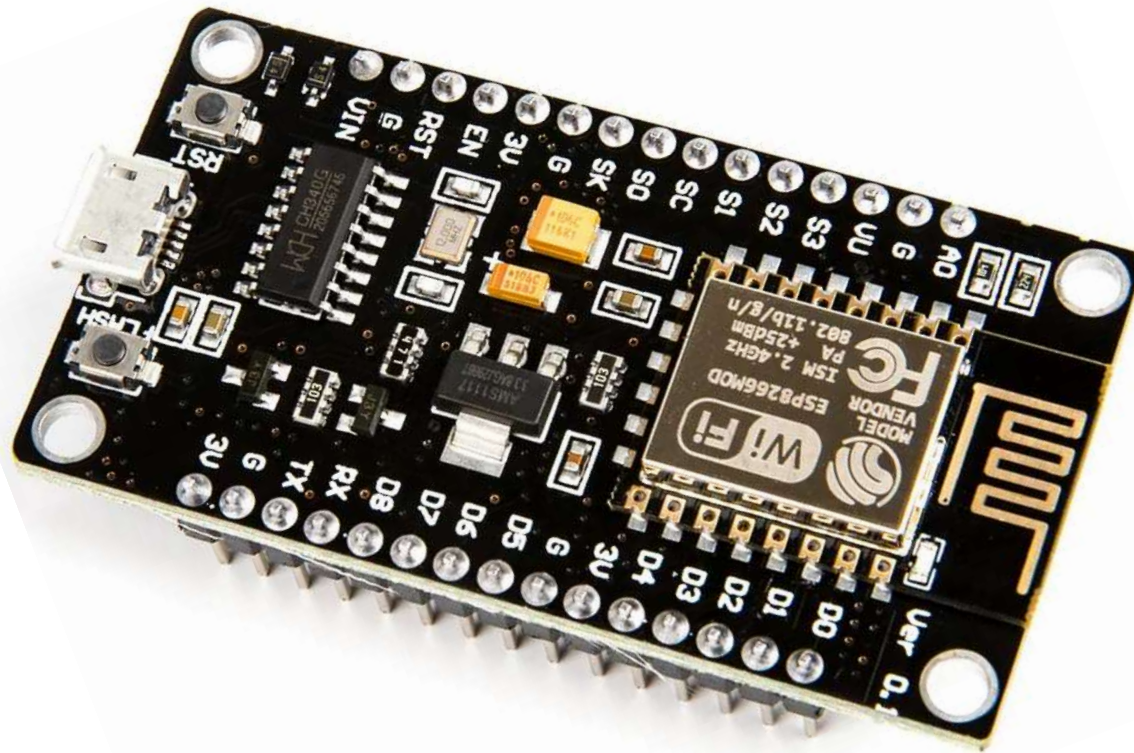
NodeMCU: Connecting to the Internet

IoT Team, BFC AI



NodeMCU ESP8266

- NodeMCU is a low-cost open-source IoT platform based on the **ESP8266** **Wi-Fi** system on a chip.
- NodeMCU **Version 3** runs on the **ESP-12E (ESP8266MOD)** module.



IoT Application: Smart Umbrella

- An umbrella that **provides information about the likelihood of rain** so that users can make a **simple decision** about whether to **take the umbrella with them** as they leave their home.



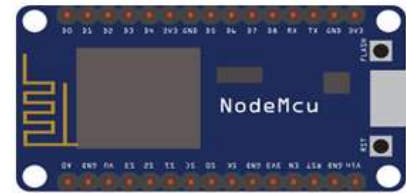
IoT Application: Smart Umbrella

- The umbrella has a handle that would illuminate when snow or rain was in the forecast.



IoT Application: Smart Umbrella

- Using existing **Wi-Fi** technology to pull information about the weather from the Internet.

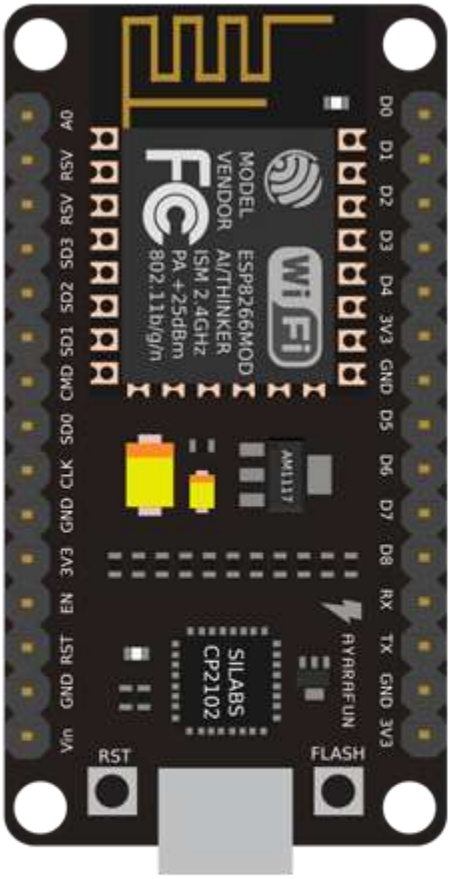


NodeMCU ESP8266: WiFi Modes

- The NodeMCU WiFi can run in **three modes**: **WiFi Station**, **Access Point**, or **both at the same time**.
- To set the NodeMCU WiFi mode, you can use the `WiFi.mode()` function.

WiFi Mode	Description
<code>WiFi.mode(WIFI_STA);</code>	Station Mode (STA) NodeMCU connects to other networks
<code>WiFi.mode(WIFI_AP);</code>	Access Point Mode (AP) NodeMCU creates its own network , and other WiFi stations can connect to it
<code>WiFi.mode(WIFI_AP_STA);</code>	Access Point + Station Mode (AP_STA) NodeMCU WiFi will act as both Access Point and WiFi Station at the same time

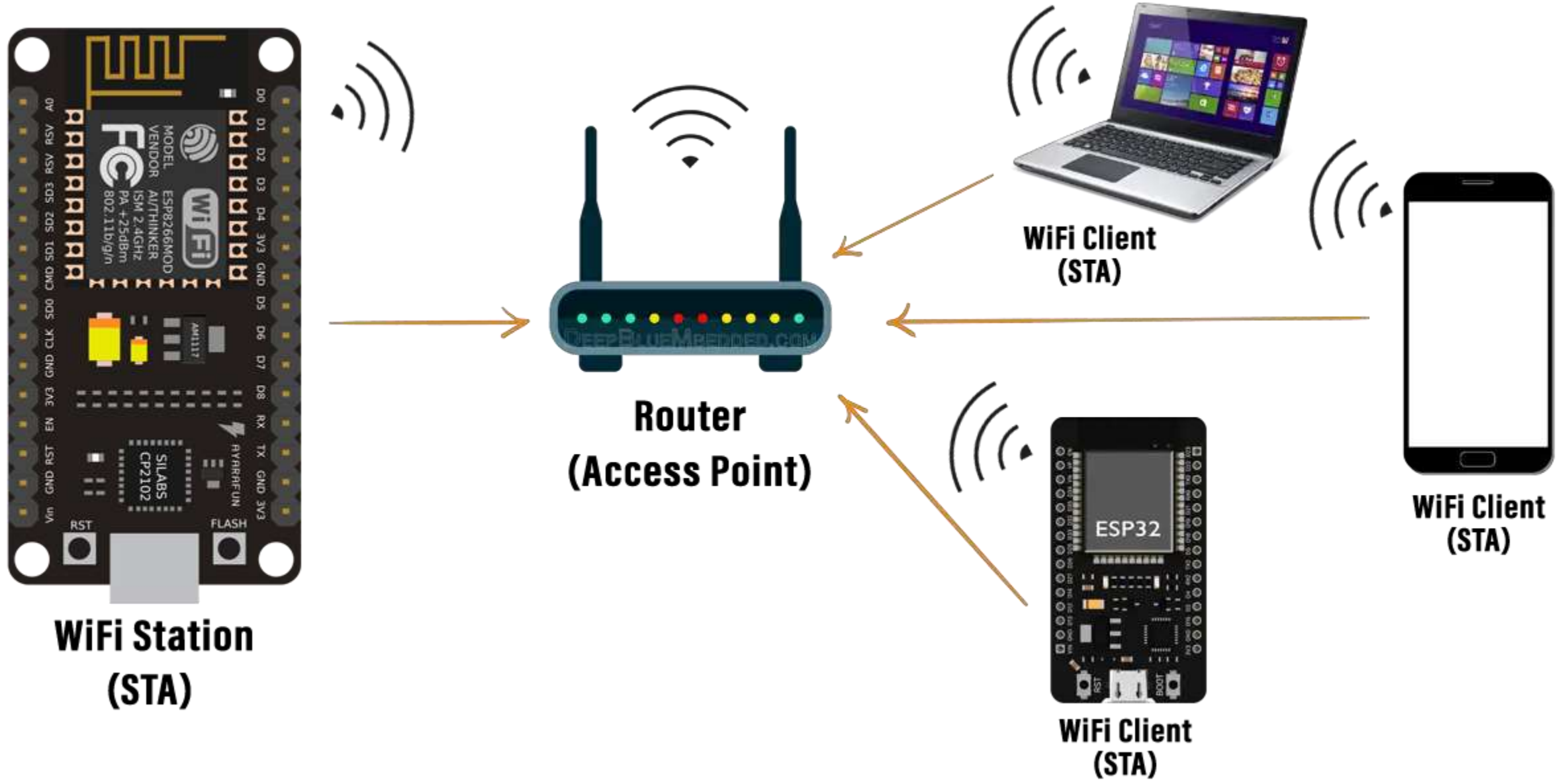
NodeMCU ESP8266: Access Point Mode



**WiFi AP
(Access Point)**



NodeMCU ESP8266: WiFi Station Mode



NodeMCU ESP8266: WiFi Station Mode – Code

```
#include <ESP8266WiFi.h> // Include ESP8266WiFi library for WiFi features

const char* WIFI_SSID = "iotlab"; // Define the WiFi network SSID
const char* WIFI_PASS = "hostiotlab"; // Define the WiFi network password

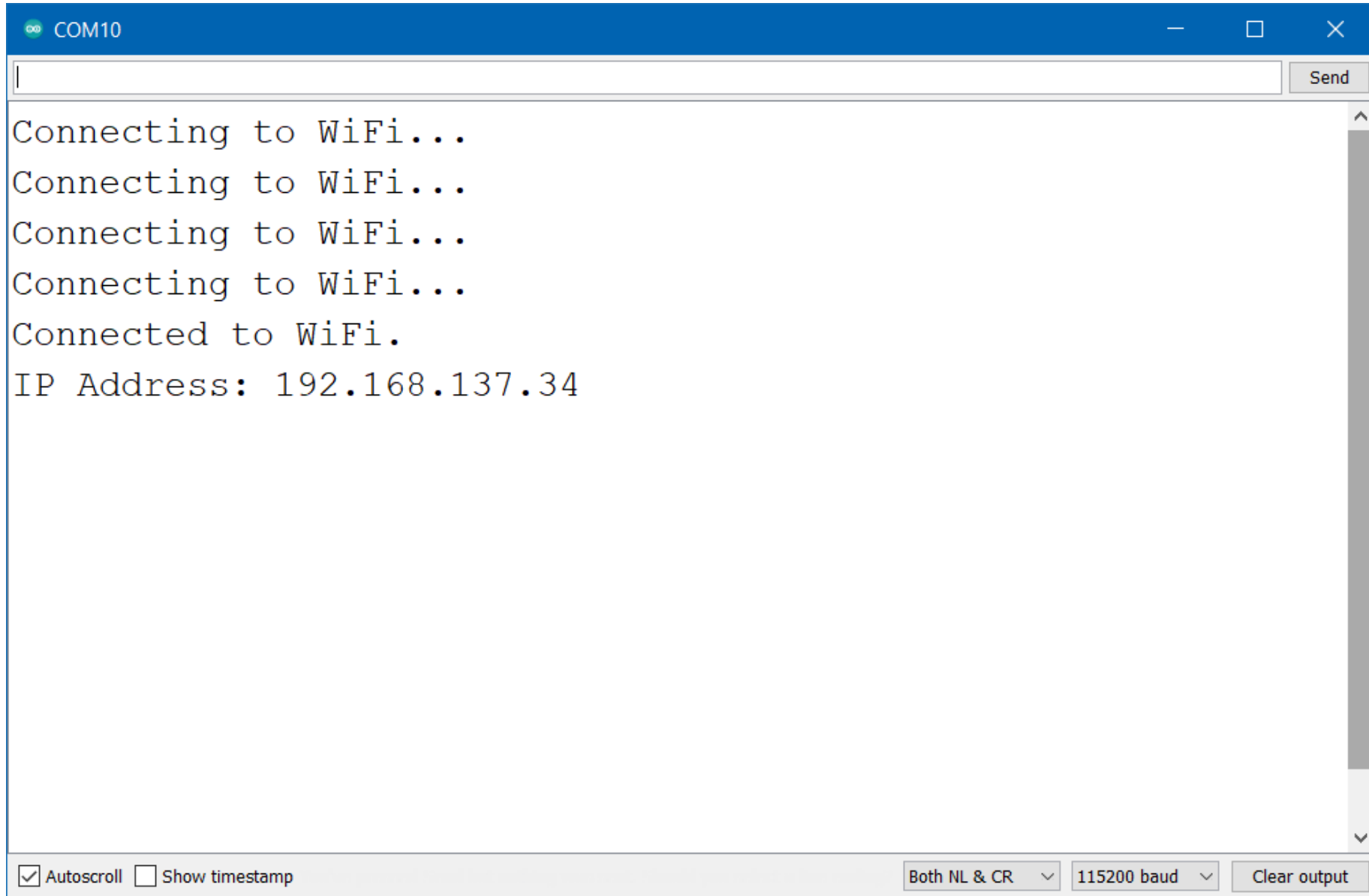
void setup() {
  Serial.begin(115200); // Start serial communication at 115200 baudrate
  WiFi.begin(WIFI_SSID, WIFI_PASS); // Begin WiFi connection using SSID and password

  while(WiFi.status() != WL_CONNECTED){ // Check if the WiFi status is not connected
    delay(1000); // Wait until the WiFi connection is established
    Serial.println("Connecting to WiFi..."); // Print message indicating an attempt to connect to WiFi
  }

  Serial.println("Connected to WiFi."); // Print message when WiFi connection is successful
  Serial.print("IP Address: "); // Print the label for the IP address
  Serial.println(WiFi.localIP()); // Print the assigned IP address
}

void loop() {
}
```

NodeMCU ESP8266: WiFi Station Mode – Output



The image shows a serial terminal window titled "COM10". The window contains the following text output:

```
Connecting to WiFi...  
Connecting to WiFi...  
Connecting to WiFi...  
Connecting to WiFi...  
Connected to WiFi.  
IP Address: 192.168.137.34
```

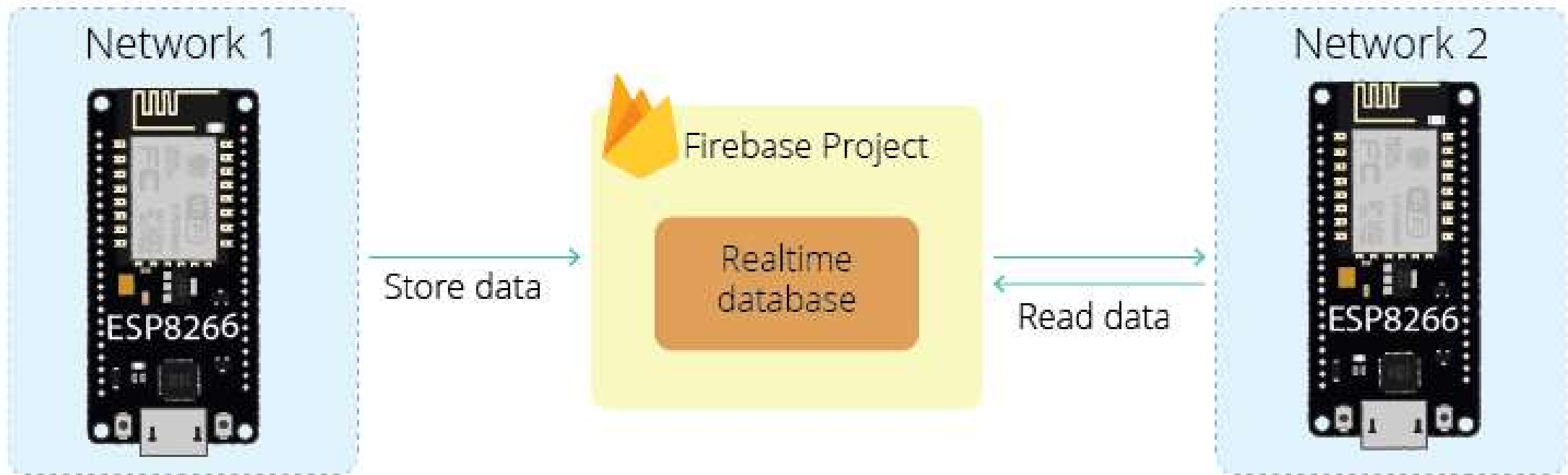
At the bottom of the window, there are several controls: a checked "Autoscroll" checkbox, an unchecked "Show timestamp" checkbox, a dropdown menu set to "Both NL & CR", a dropdown menu set to "115200 baud", and a "Clear output" button.

Firebase

- Firebase is Google's mobile and web application development platform that includes **many services to manage data** from IOS, Android, or web applications.
- This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on.
- The services are **hosted in the cloud** and scale with little to no effort on the part of the developer.
- You can use the **ESP8266** to connect and interact with your Firebase project, and you can create **applications to control the ESP8266 via Firebase from anywhere in the world.**

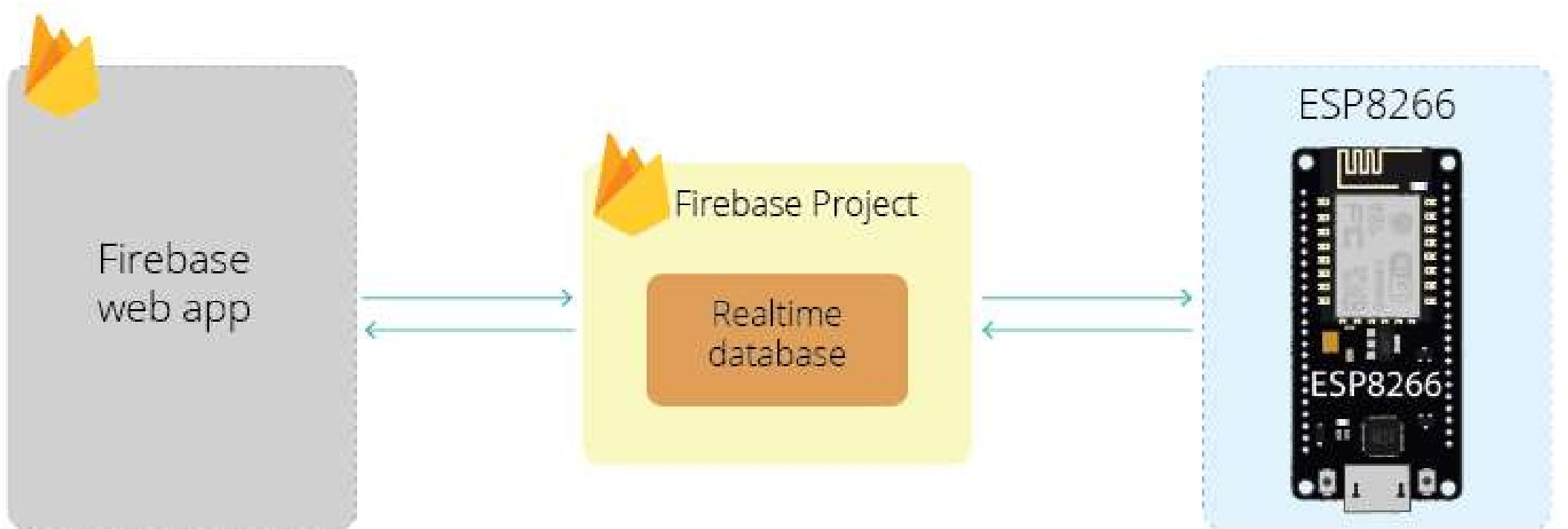
Firestore

- The ESP8266 can interact with the database **from anywhere in the world**.
- You can have **two ESP8266 boards in different networks**, with **one board storing data** and the **other board reading the most recent data**, for example.



Firebase

- You can have a **web or mobile app** using **Firebase** that will use **ESP8266** to **display sensor readings** or **control outputs from anywhere in the world**.

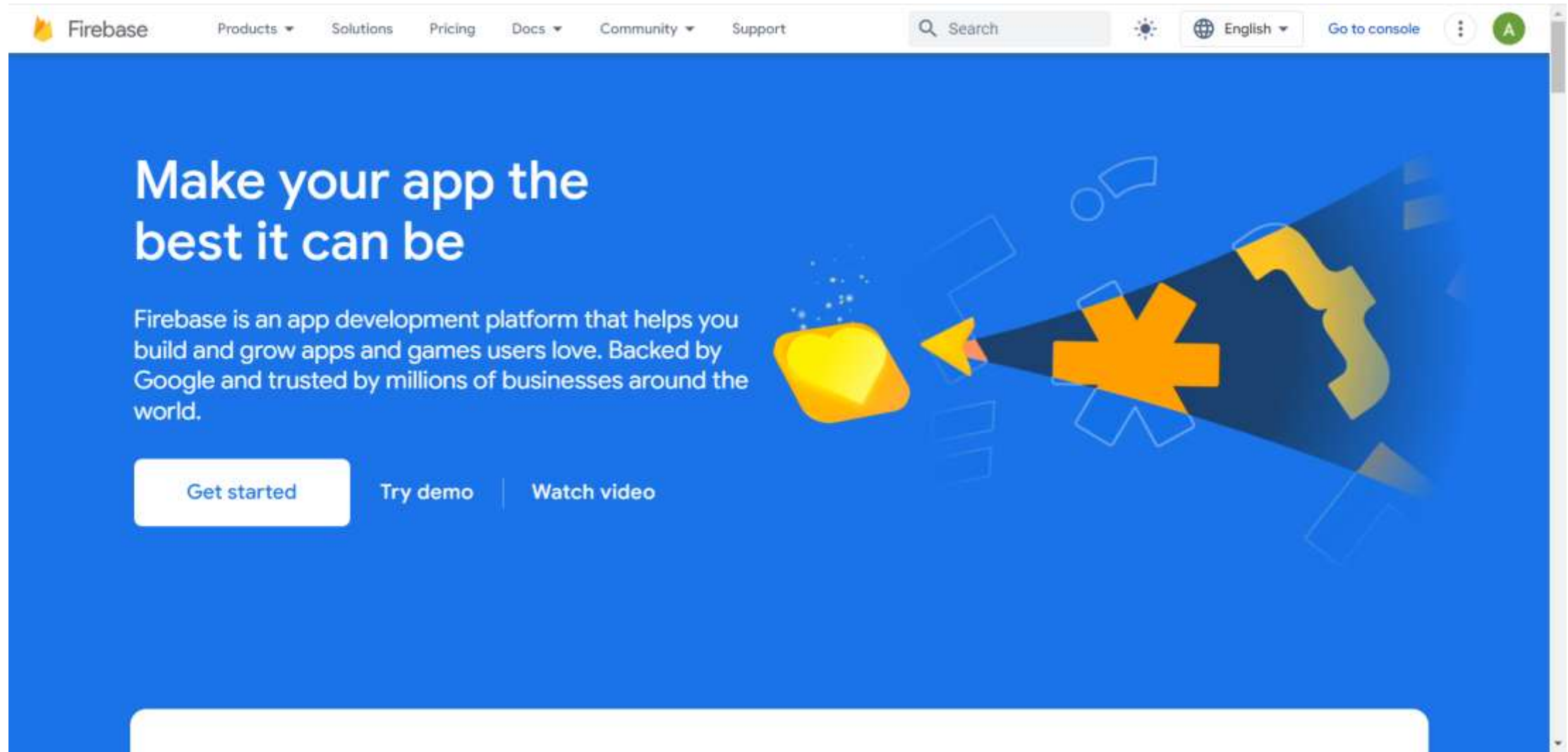


- Display sensor readings
- Control ESP8266 Outputs

- Send sensor readings
- Update GPIOs states

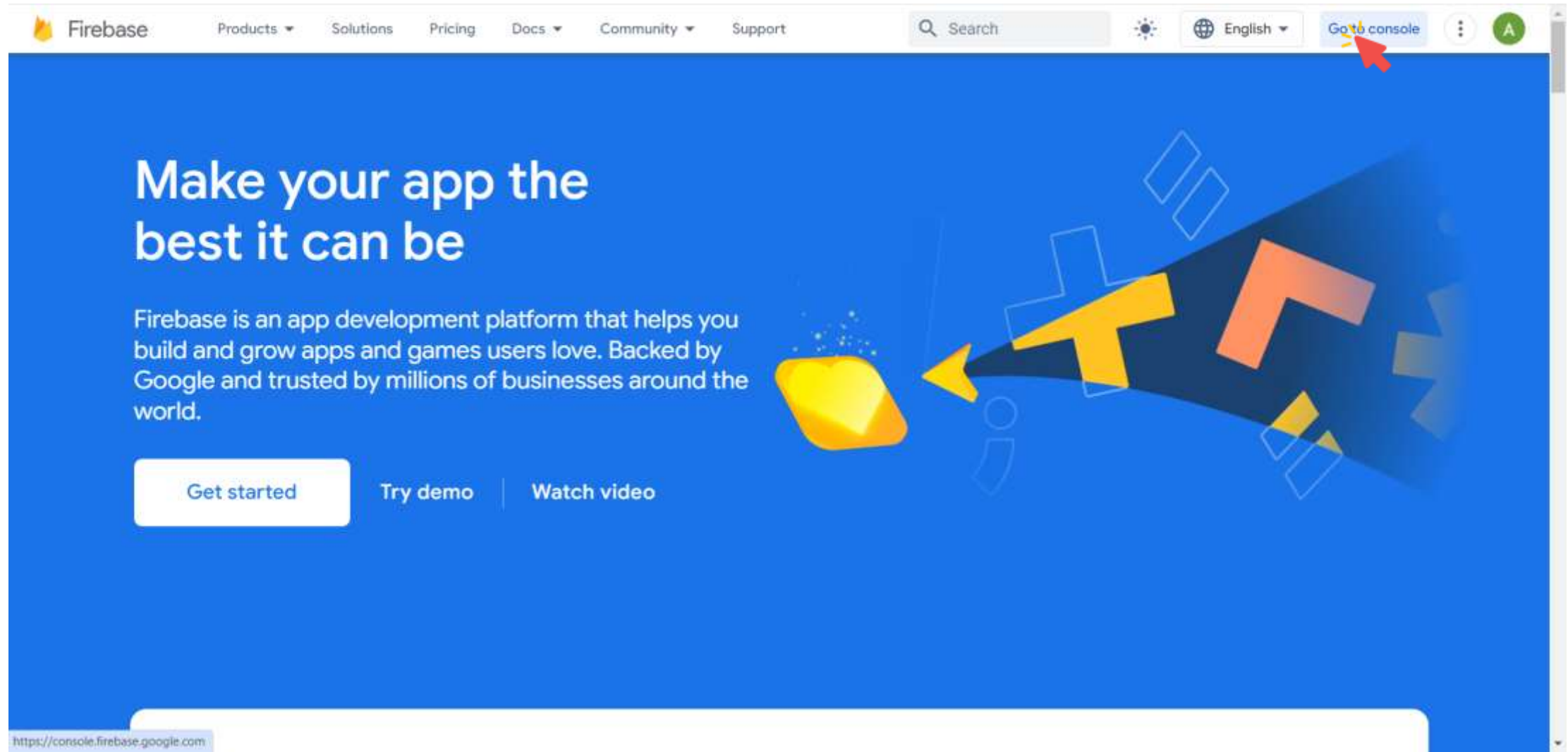
Firestore: Creating a New Project

- Go to <https://firebase.google.com>



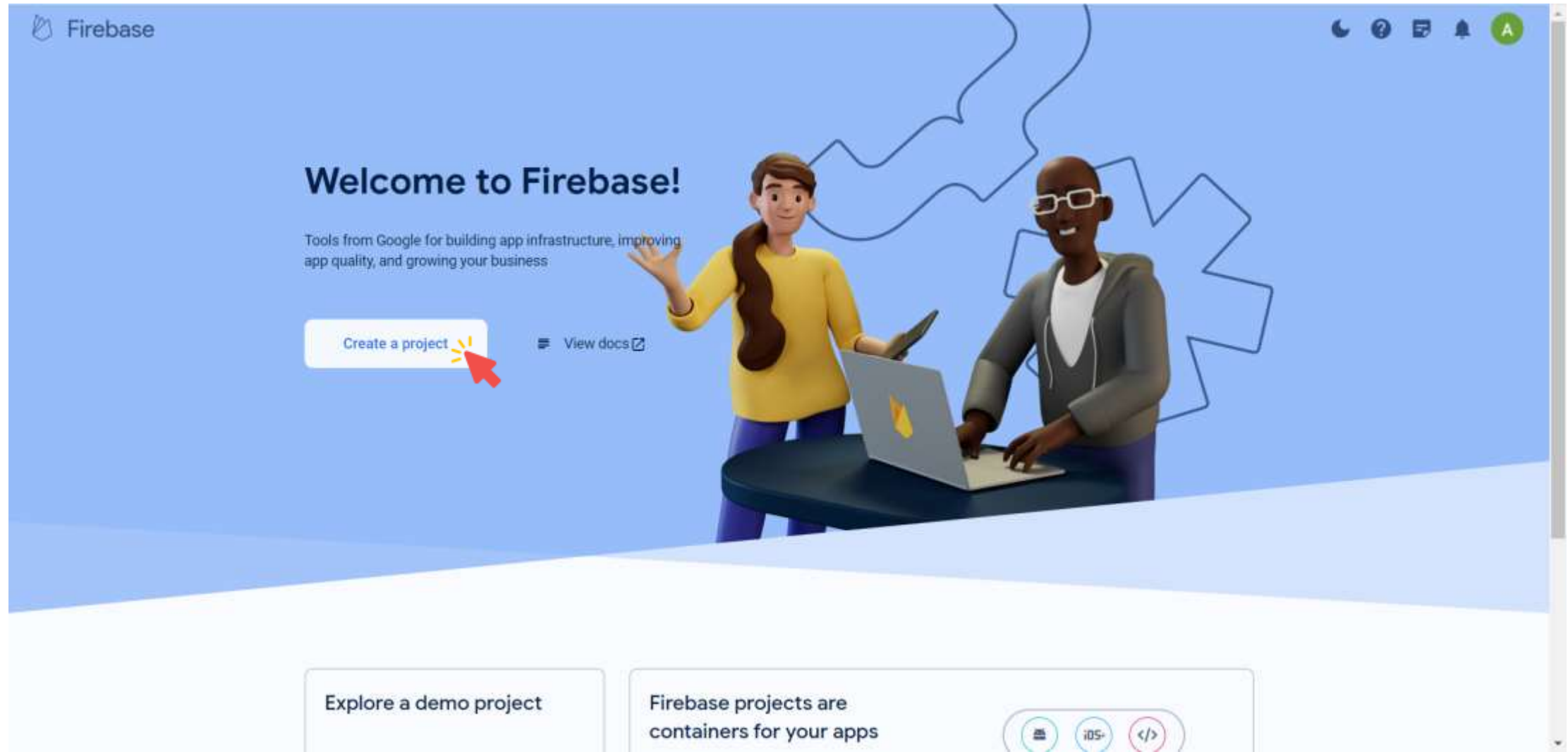
Firebase: Creating a New Project

- Click on **Go to Console**.



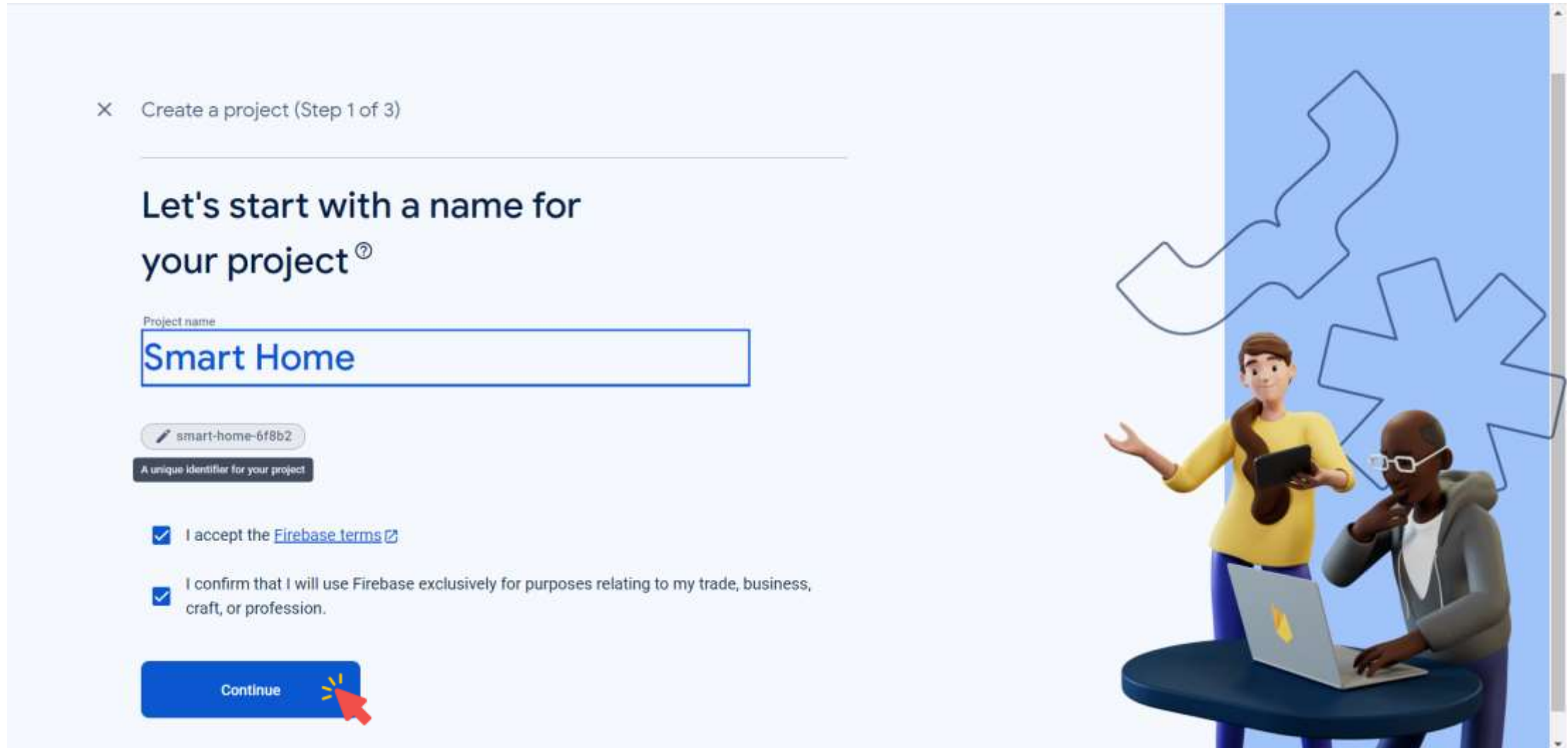
Firestore: Creating a New Project

- Click on **Create a Project**.



Firestore: Creating a New Project

- Enter the **name** of your project, accept terms, and click **Continue**.



✕ Create a project (Step 1 of 3)

Let's start with a name for your project [?]

Project name

Smart Home

smart-home-6f8b2

A unique identifier for your project

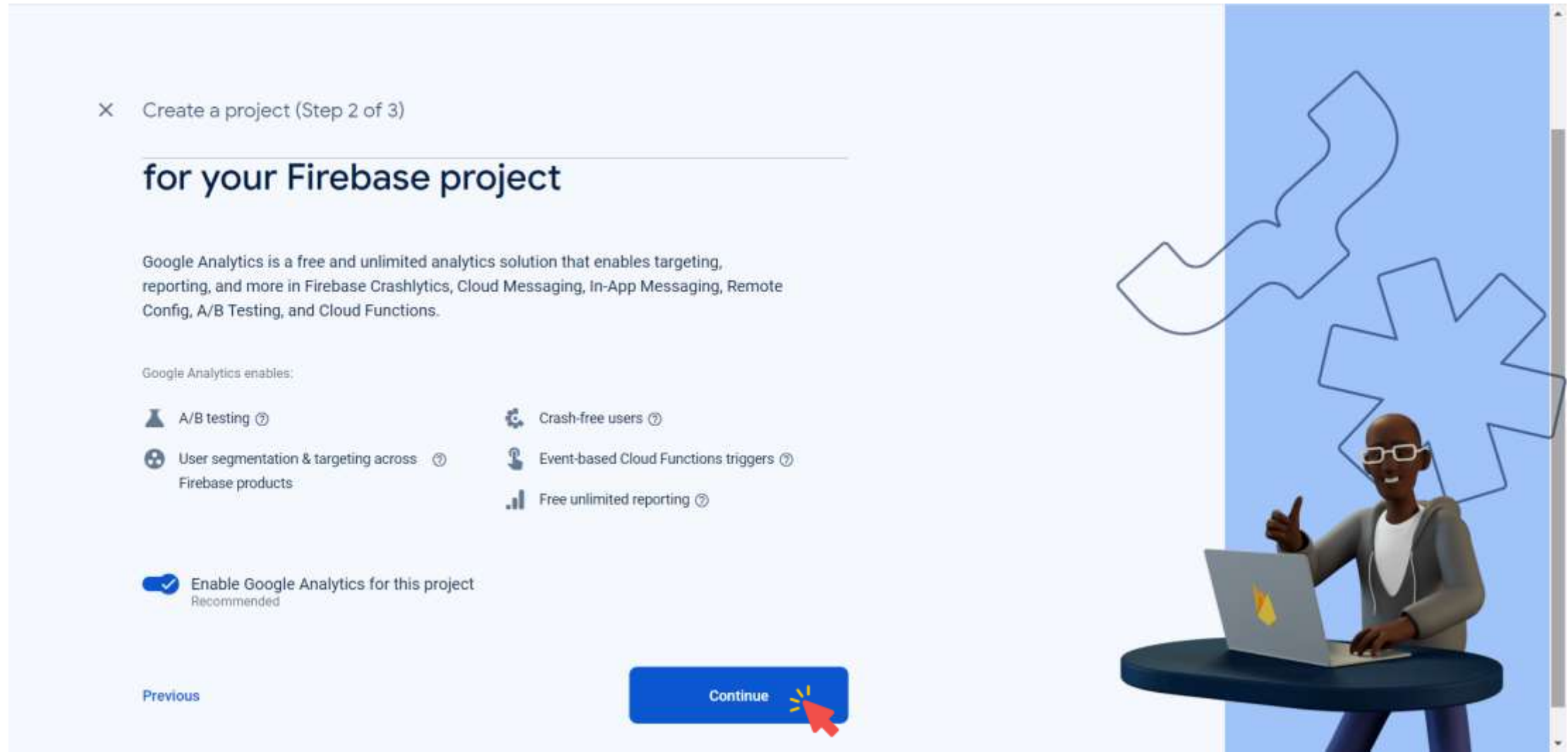
I accept the [Firebase terms](#)

I confirm that I will use Firebase exclusively for purposes relating to my trade, business, craft, or profession.

Continue

Firebase: Creating a New Project

- Click **Continue**.








✕ Create a project (Step 2 of 3)

for your Firebase project

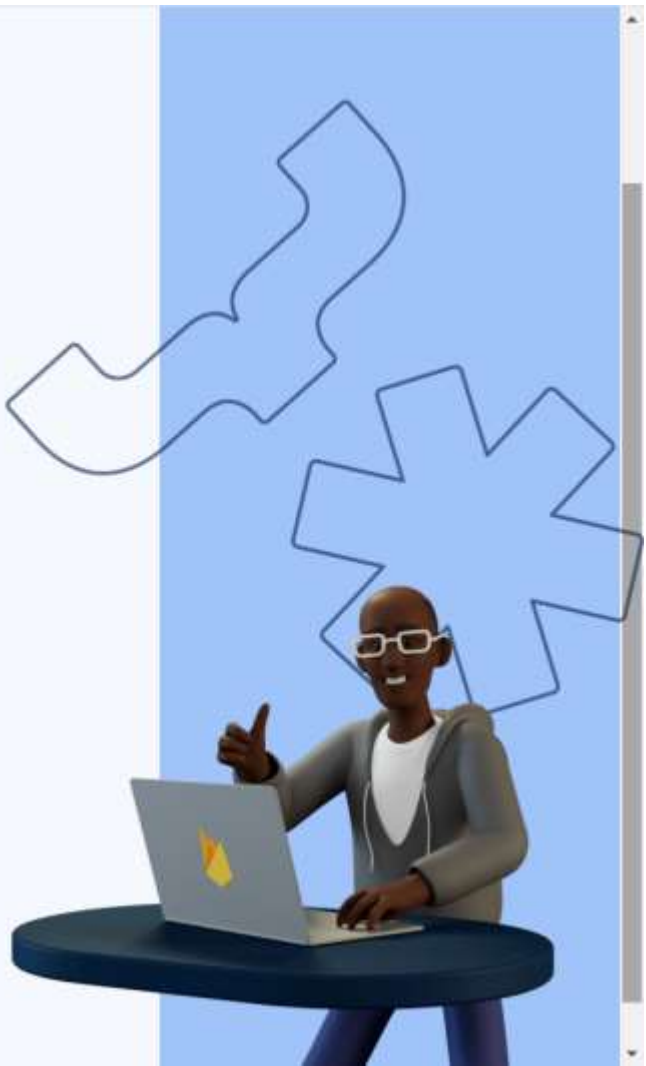
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

-  A/B testing [?](#)
-  Crash-free users [?](#)
-  User segmentation & targeting across Firebase products [?](#)
-  Event-based Cloud Functions triggers [?](#)
-  Free unlimited reporting [?](#)

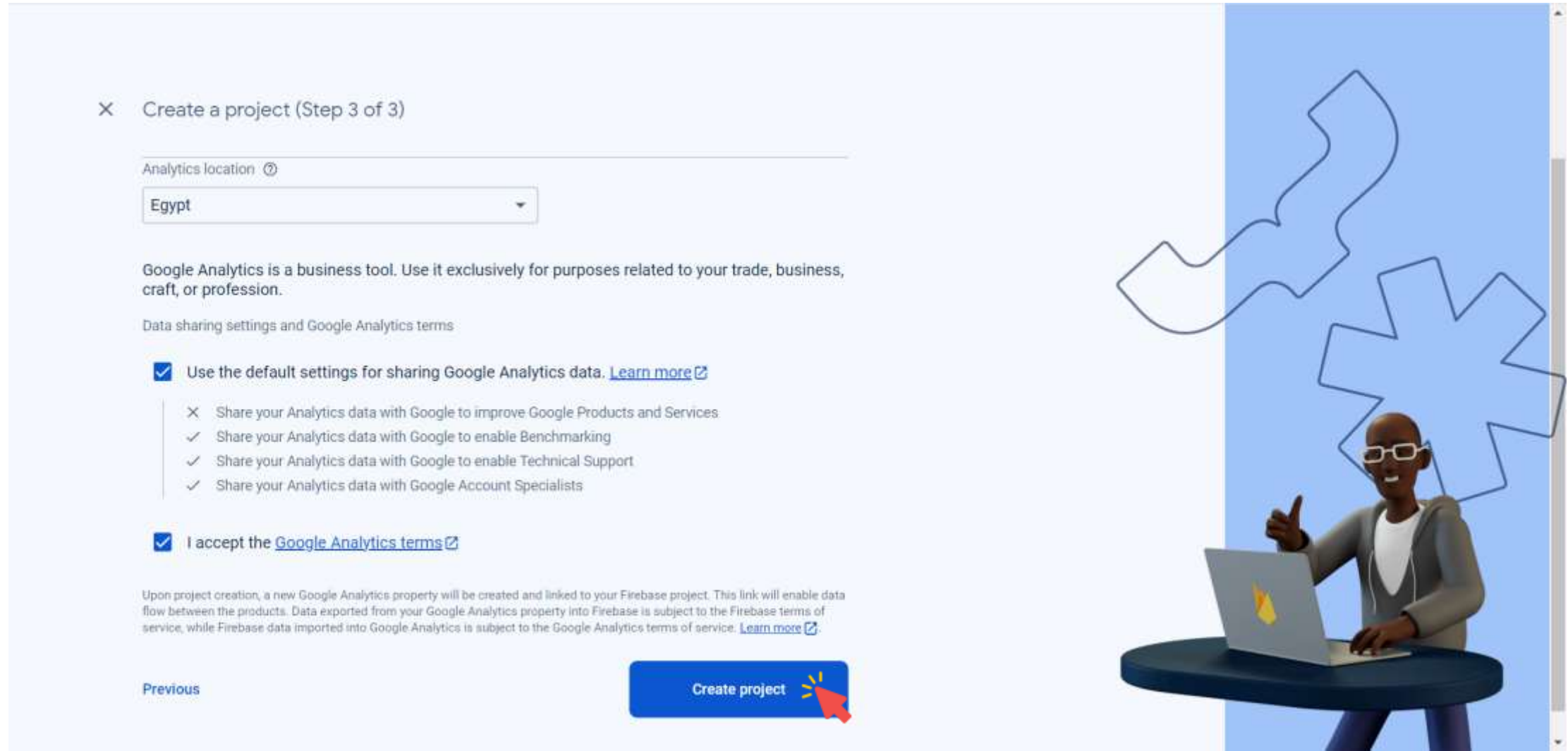
Enable Google Analytics for this project
Recommended

[Previous](#) [Continue](#)



Firestore: Creating a New Project

- Choose the **location**, accept terms, and click **Create Project**.



✕ Create a project (Step 3 of 3)

Analytics location ⓘ

Egypt ▾

Google Analytics is a business tool. Use it exclusively for purposes related to your trade, business, craft, or profession.


Data sharing settings and Google Analytics terms

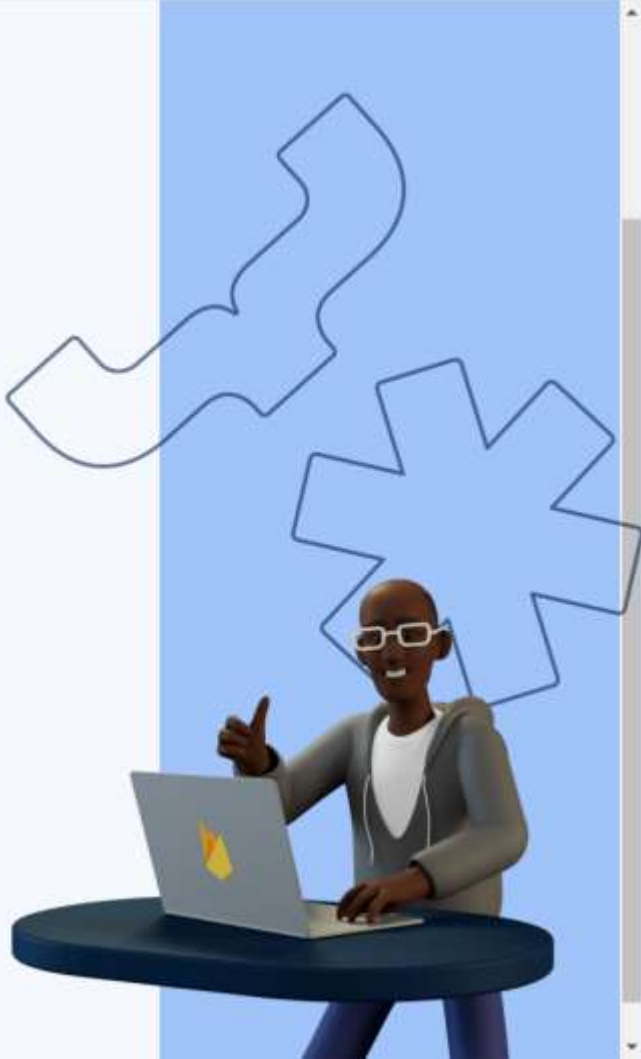
- Use the default settings for sharing Google Analytics data. [Learn more](#) ⓘ
- Share your Analytics data with Google to improve Google Products and Services
- Share your Analytics data with Google to enable Benchmarking
- Share your Analytics data with Google to enable Technical Support
- Share your Analytics data with Google Account Specialists

I accept the [Google Analytics terms](#) ⓘ

Upon project creation, a new Google Analytics property will be created and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#) ⓘ.

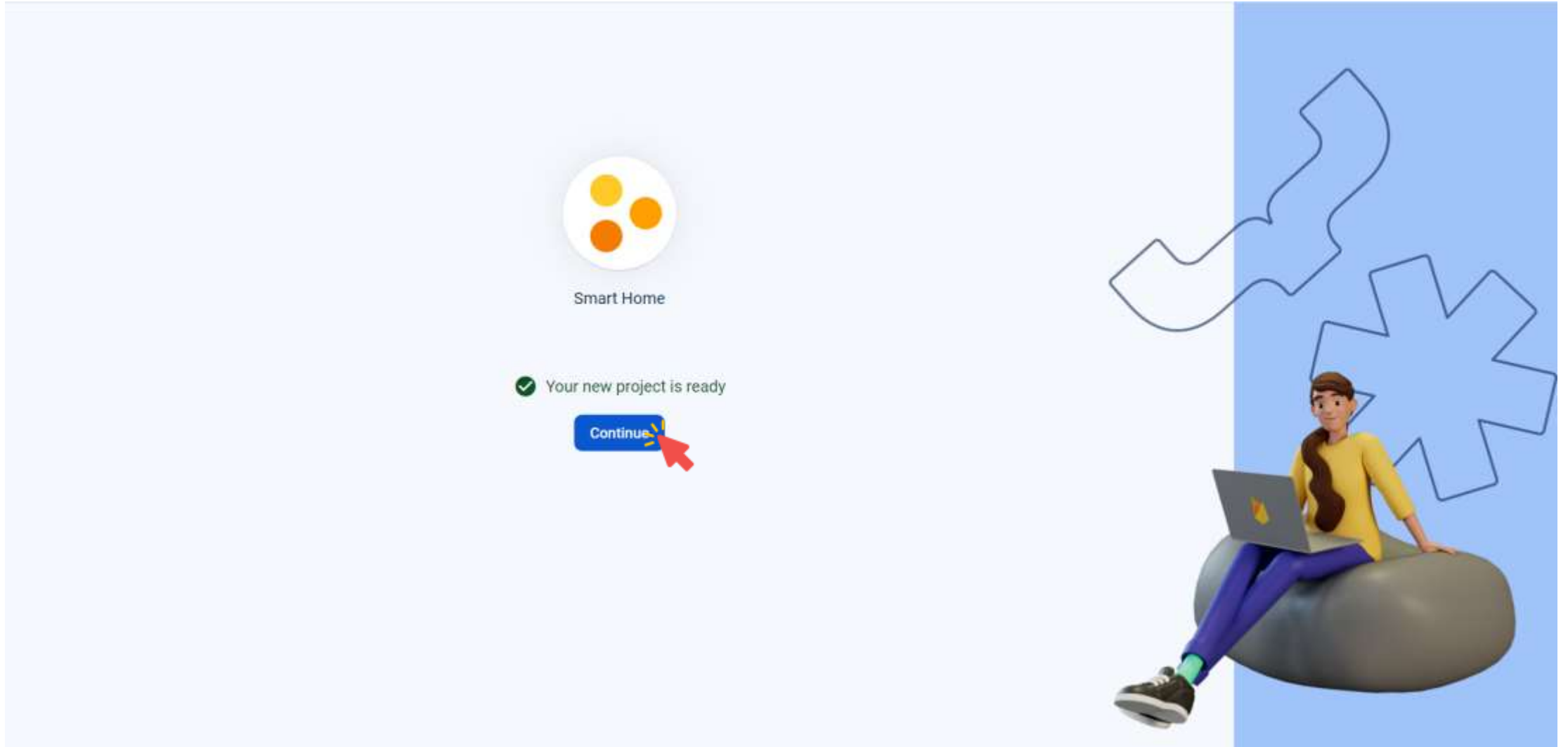
[Previous](#)

Create project 



Firestore: Creating a New Project

- Wait until the project is created, and click **Continue**.



Firebase: Creating a New Project

- You'll be redirected to your project console page.

The screenshot displays the Firebase Project Console interface. On the left is a navigation sidebar with the 'Project Overview' tab selected. The main content area features a header for the 'Smart Home' project, including a 'Spark plan' button and an email subscription prompt. The central focus is a large blue banner with the text 'Get started by adding Firebase to your app' and illustrations of two people. Below this banner are icons for various platforms and development tools. At the bottom, a section titled 'Store and sync app data in milliseconds' is partially visible.

Smart Home

Receive email updates about new Firebase features, research, and events [Sign up](#)

Smart Home [Spark plan](#)

Get started by adding Firebase to your app

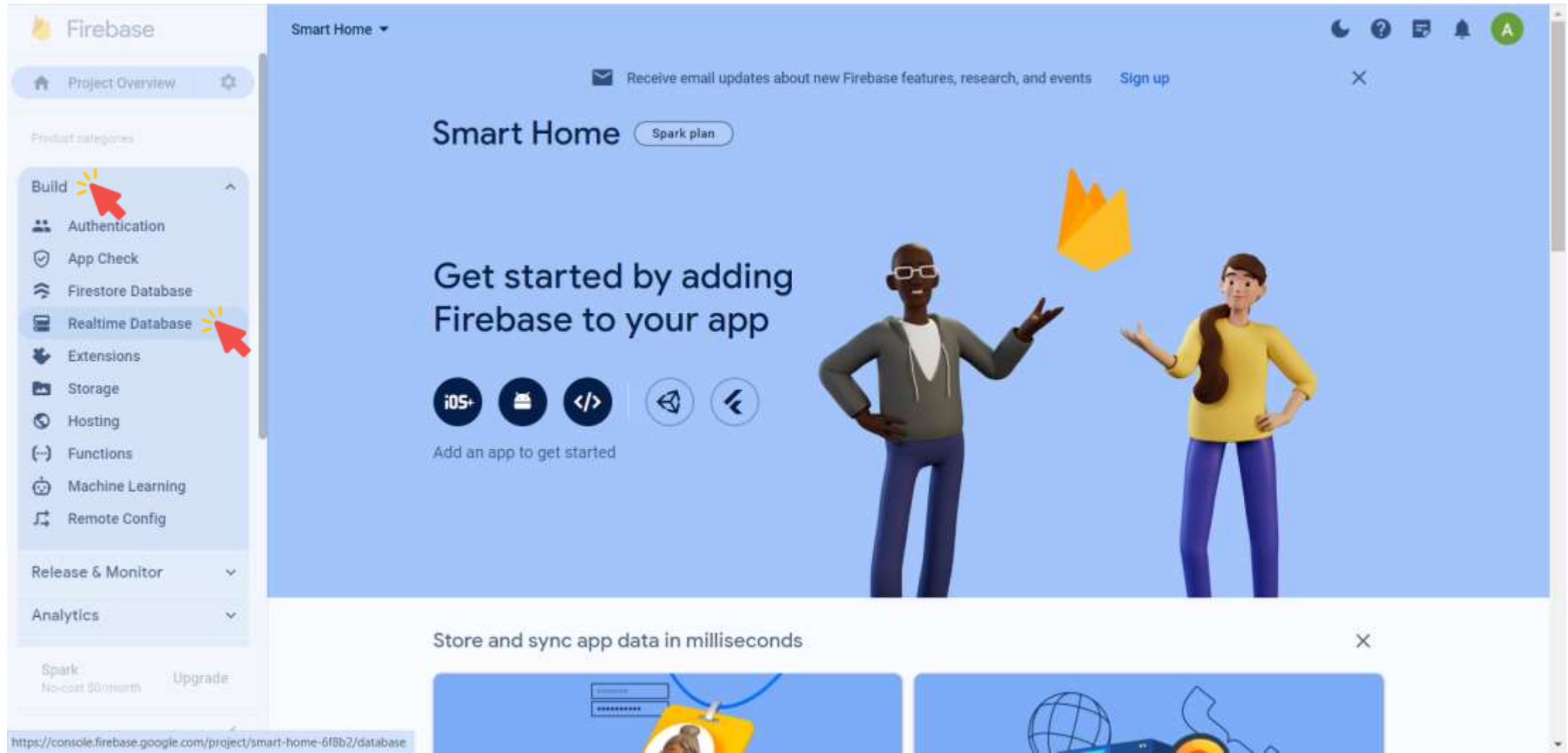
iOS+ Android </> [Icons]

Add an app to get started

Store and sync app data in milliseconds

Firestore: Creating a Realtime Database

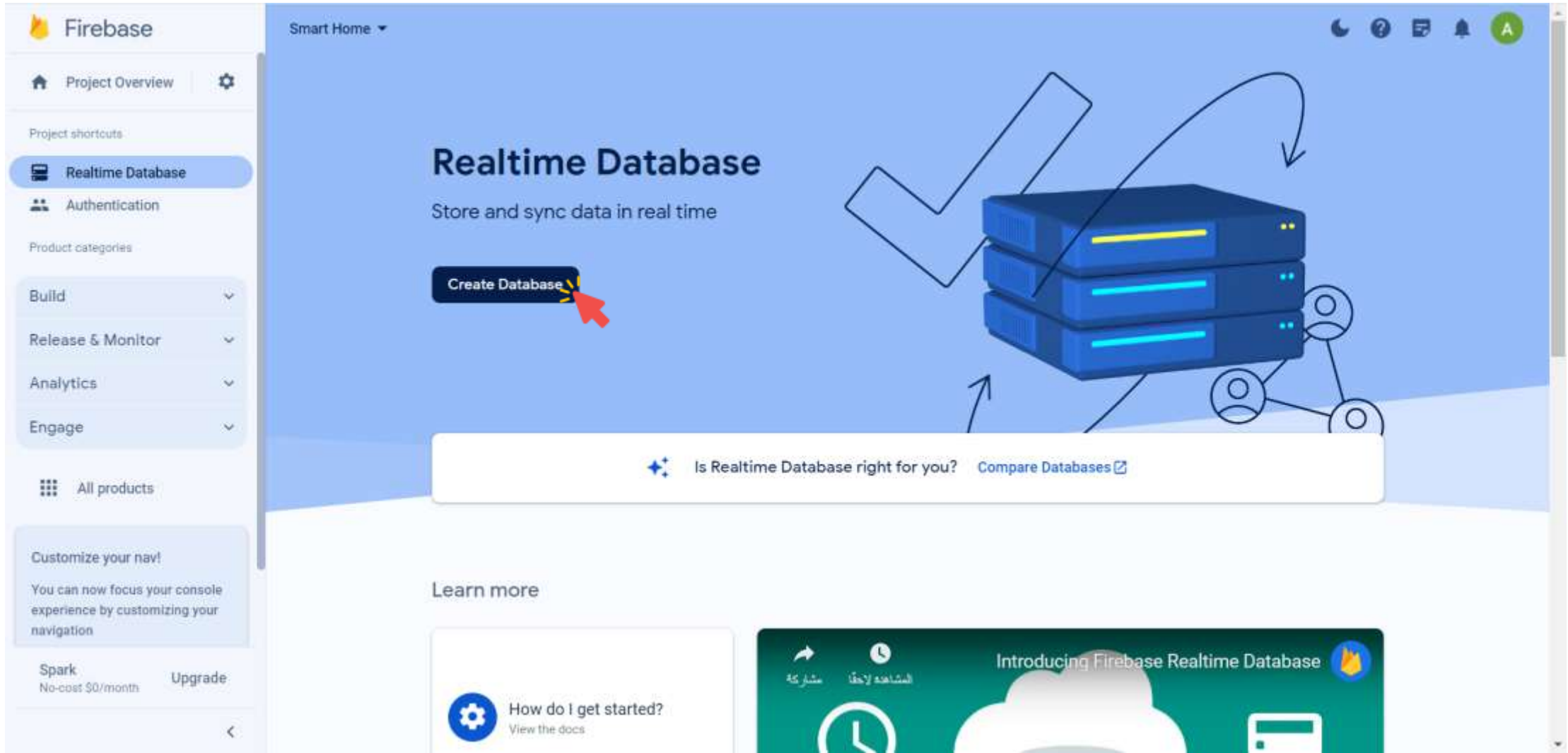
- Expand **Build** tap, and go to **Realtime Database**.



The screenshot displays the Firebase console interface for a project named "Smart Home". On the left, a navigation sidebar is visible with the "Build" tab expanded, and the "Realtime Database" option highlighted by a red mouse cursor. The main content area features a "Smart Home" header with a "Spark plan" button, a "Get started by adding Firebase to your app" section with icons for iOS, Android, and web, and a "Store and sync app data in milliseconds" section with a close button. The URL at the bottom is <https://console.firebase.google.com/project/smart-home-6f8b2/database>.

Firestore: Creating a Realtime Database

- Click on **Create Database**.

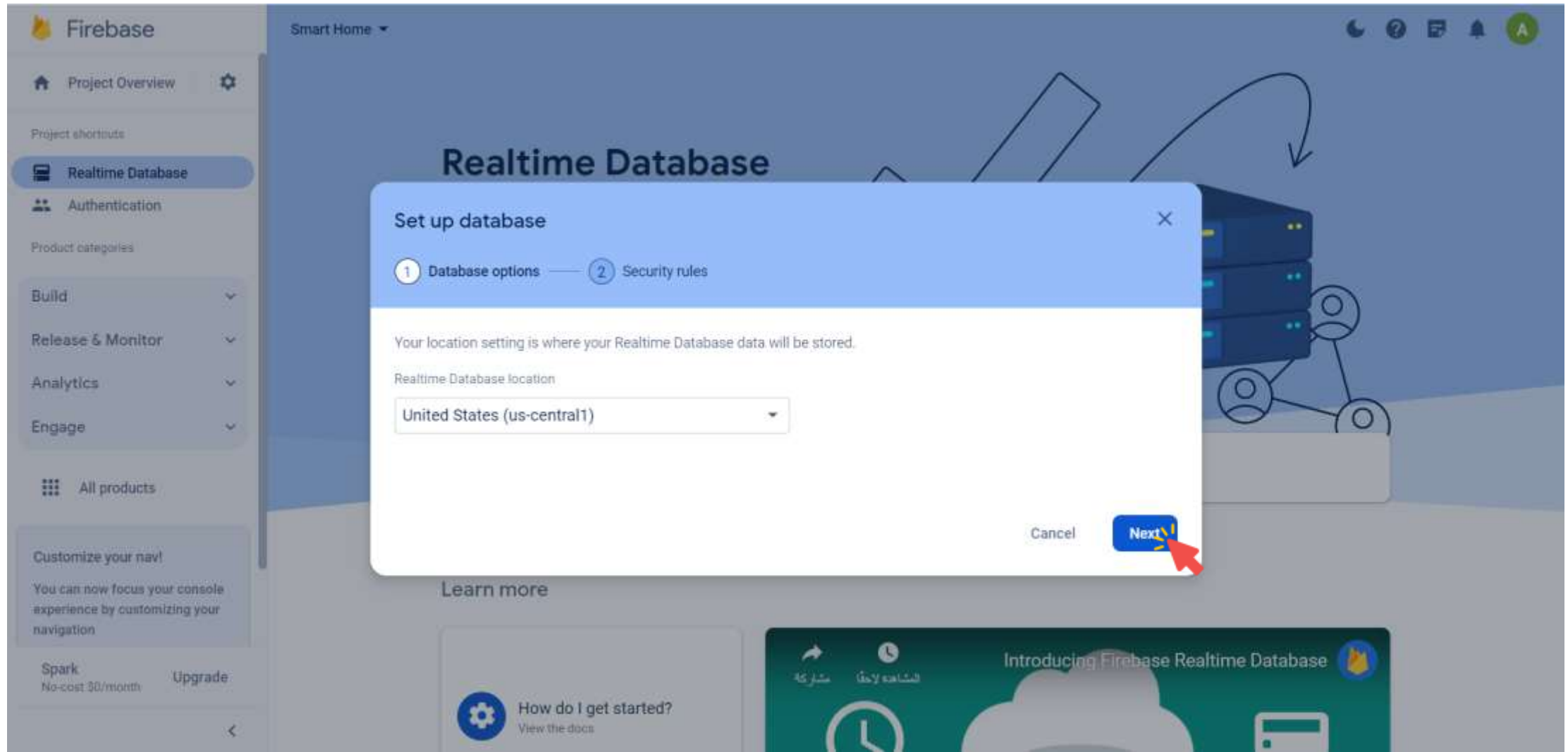


The screenshot displays the Firebase console interface. On the left, a navigation sidebar includes the 'Project Overview' and 'Project shortcuts' sections. Under 'Project shortcuts', the 'Realtime Database' option is highlighted with a blue bar. Below this, there are sections for 'Product categories' (Build, Release & Monitor, Analytics, Engage) and 'All products'. At the bottom of the sidebar, there is a 'Customize your nav!' section and a 'Spark' upgrade offer.

The main content area is titled 'Smart Home' and features a large blue banner for 'Realtime Database'. The banner text reads 'Store and sync data in real time' and includes a prominent 'Create Database' button. A red arrow points to this button. To the right of the text is an illustration of server racks with a checkmark and a network diagram. Below the banner, there is a white box with the text 'Is Realtime Database right for you?' and a link to 'Compare Databases'. At the bottom, a 'Learn more' section contains a card for 'How do I get started?' and a video thumbnail for 'Introducing Firebase Realtime Database'.

Firestore: Creating a Realtime Database

- Select your **database location**, and click **Next**.



Firestore: Creating a Realtime Database

- For testing purposes, select **Start in test mode**, and click **Enable**.

The screenshot shows the 'Set up database' dialog in the Firebase console. The dialog is titled 'Set up database' and has two steps: 'Database options' and 'Security rules'. The 'Security rules' step is active. The dialog contains the following text:

Once you have defined your data structure you will have to write rules to secure your data. [Learn more](#)

Start in locked mode
Your data is private by default. Client read/write access will only be granted as specified by your security rules.

Start in test mode
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

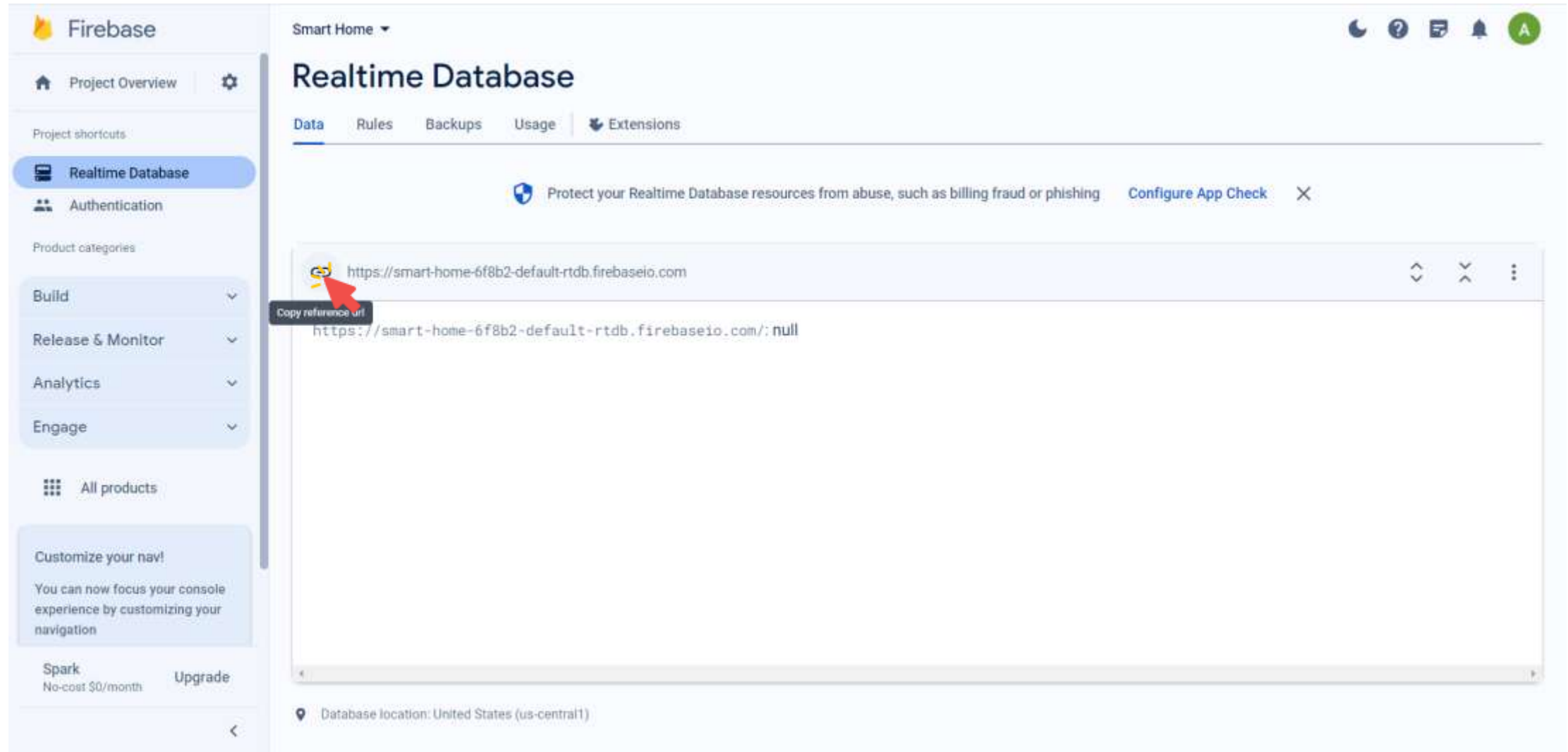
```
{
  "rules": {
    ".read": "now < 1705615200000", // 2024-1-19
    ".write": "now < 1705615200000", // 2024-1-19
  }
}
```

! The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Cancel **Enable**

Firestore: Creating a Realtime Database

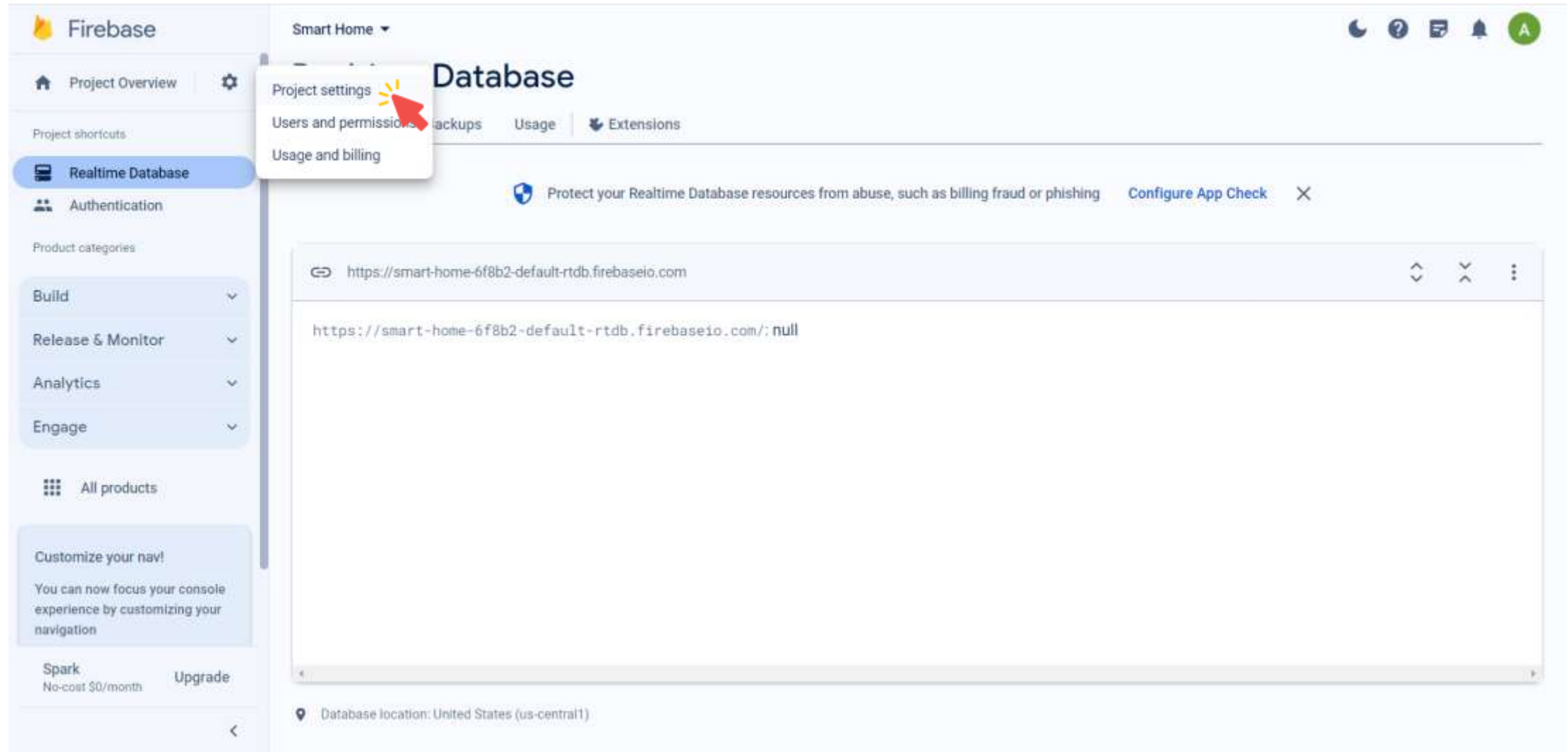
- Your database is now created. You need to copy the **database URL**.



The screenshot shows the Firebase console interface for a project named "Smart Home". The "Realtime Database" section is active, displaying the "Data" tab. A notification banner at the top reads "Protect your Realtime Database resources from abuse, such as billing fraud or phishing" with a "Configure App Check" link. Below this, a browser window is open, showing the URL `https://smart-home-6f8b2-default-rtdb.firebaseio.com` in the address bar. A red arrow points to the URL, and a tooltip says "Copy reference url". The main content area of the browser window displays the database reference URL: `https://smart-home-6f8b2-default-rtdb.firebaseio.com/:null`. At the bottom of the console, the database location is noted as "United States (us-central1)".

Firestore: Getting Database Secret

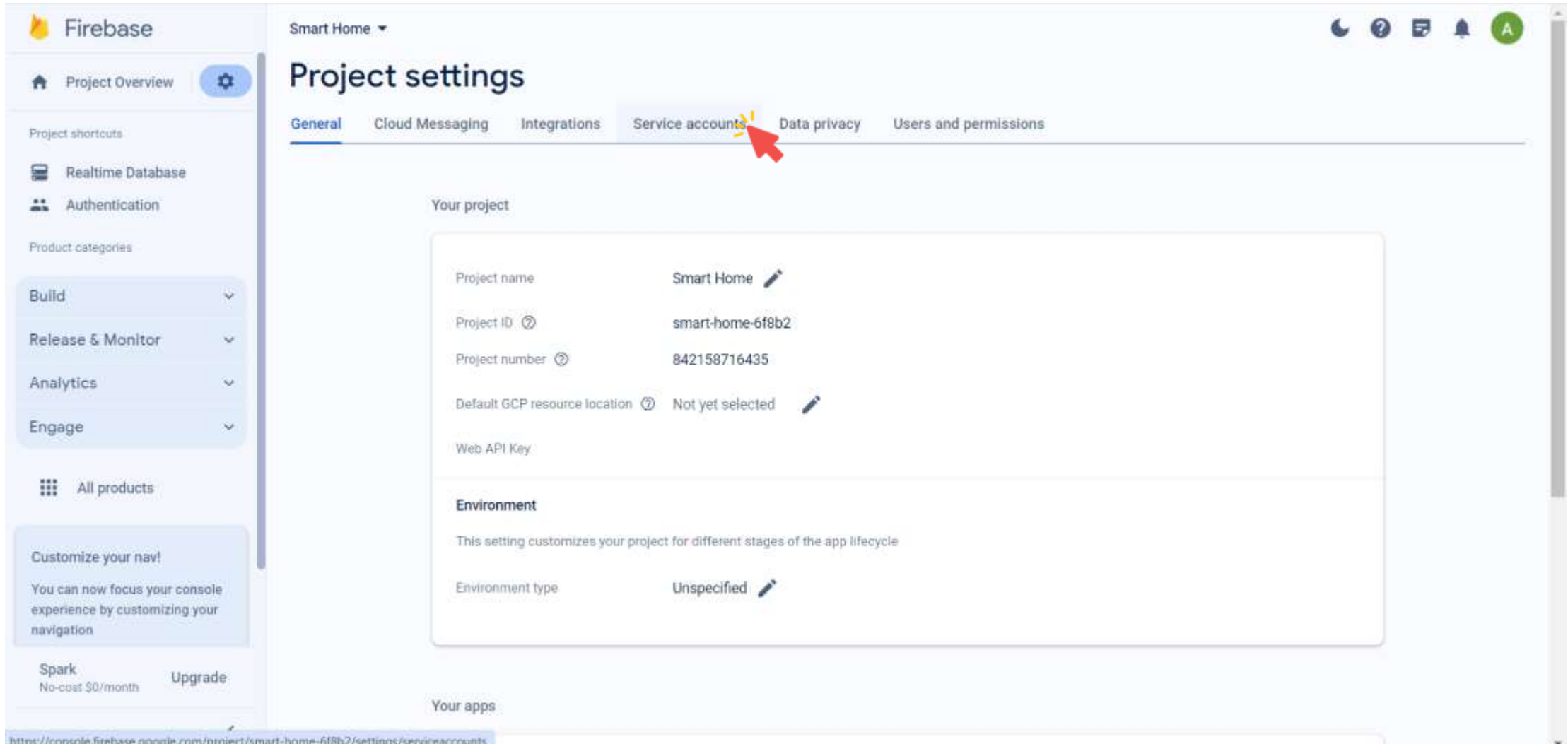
- Goto **Project settings**.



The screenshot displays the Firebase console interface. On the left, the navigation sidebar is visible, with 'Project settings' highlighted by a red arrow. The main content area shows the 'Database' settings for a Realtime Database. The URL bar displays `https://smart-home-6f8b2-default-rtdb.firebaseio.com`. Below the URL bar, the database URL is shown as `https://smart-home-6f8b2-default-rtdb.firebaseio.com/:null`. At the bottom, the database location is indicated as 'United States (us-central1)'. A notification banner at the top right of the main content area reads 'Protect your Realtime Database resources from abuse, such as billing fraud or phishing' with a 'Configure App Check' link.

Firestore: Getting Database Secret

- Click on **Service accounts**.



The screenshot shows the Firebase Project settings interface for a project named 'Smart Home'. The 'Service accounts' tab is selected and highlighted with a red arrow. The page displays project details and environment settings.

Project settings

General | Cloud Messaging | Integrations | **Service accounts** | Data privacy | Users and permissions

Your project

Project name	Smart Home
Project ID	smart-home-6f8b2
Project number	842158716435
Default GCP resource location	Not yet selected
Web API Key	

Environment

This setting customizes your project for different stages of the app lifecycle

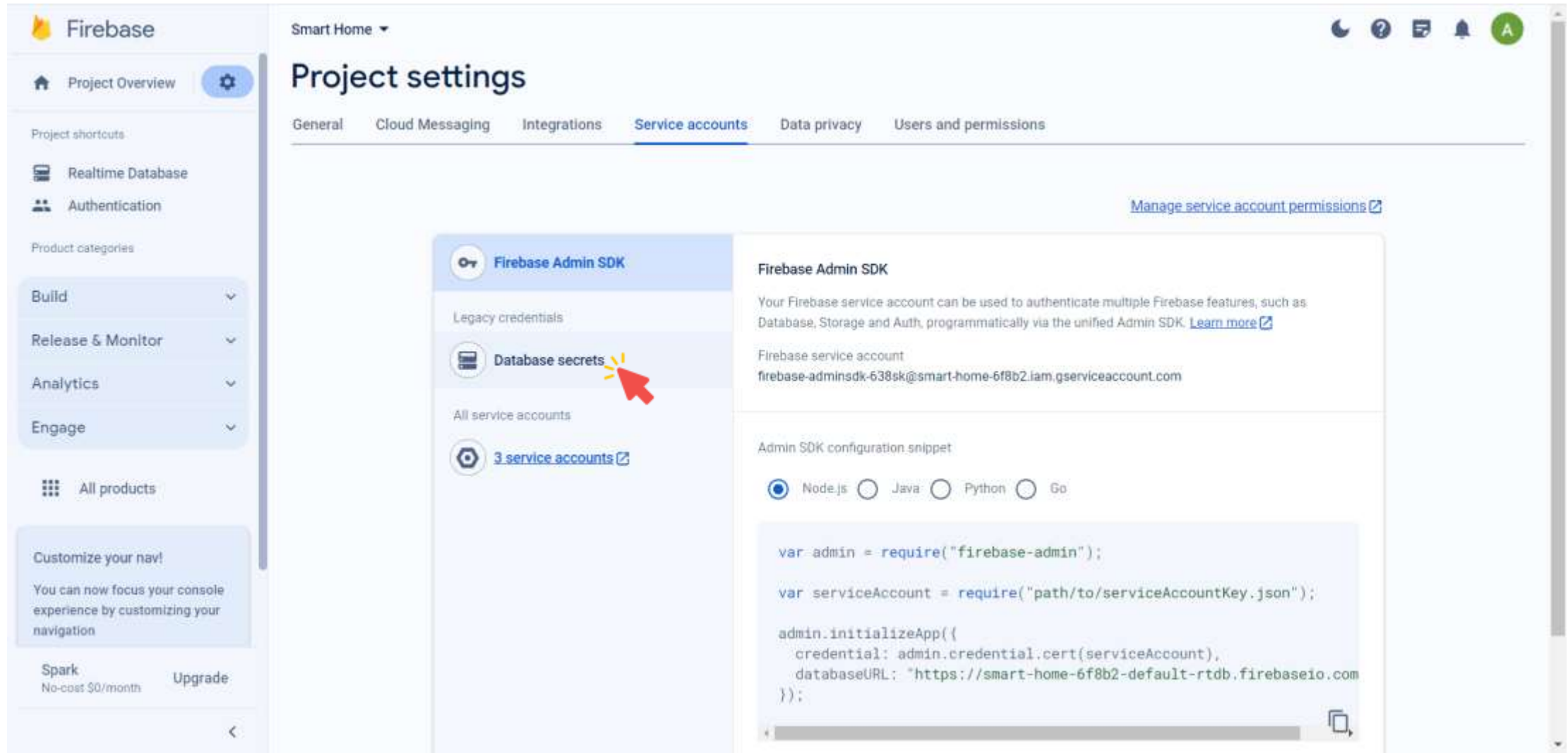
Environment type	Unspecified
------------------	-------------

Your apps

<https://console.firebase.google.com/project/smart-home-6f8b2/settings/serviceaccounts>

Firestore: Getting Database Secret

- Click on **Database secrets**.



The screenshot shows the Firebase Project settings interface. The left sidebar contains navigation options like 'Project Overview', 'Realtime Database', and 'Authentication'. The main content area is titled 'Project settings' and has tabs for 'General', 'Cloud Messaging', 'Integrations', 'Service accounts', 'Data privacy', and 'Users and permissions'. The 'Service accounts' tab is active. In the left-hand list of this tab, 'Database secrets' is highlighted with a red arrow. The right-hand pane shows details for the 'Firebase Admin SDK' service account, including its email address and a configuration snippet for Node.js.

Smart Home ▾

Project settings

General Cloud Messaging Integrations **Service accounts** Data privacy Users and permissions

[Manage service account permissions](#)

Database secrets

Legacy credentials

All service accounts

[3 service accounts](#)

Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. [Learn more](#)

Firestore service account
firebase-adminsdk-638sk@smart-home-6f8b2.iam.gserviceaccount.com

Admin SDK configuration snippet

Node.js Java Python Go

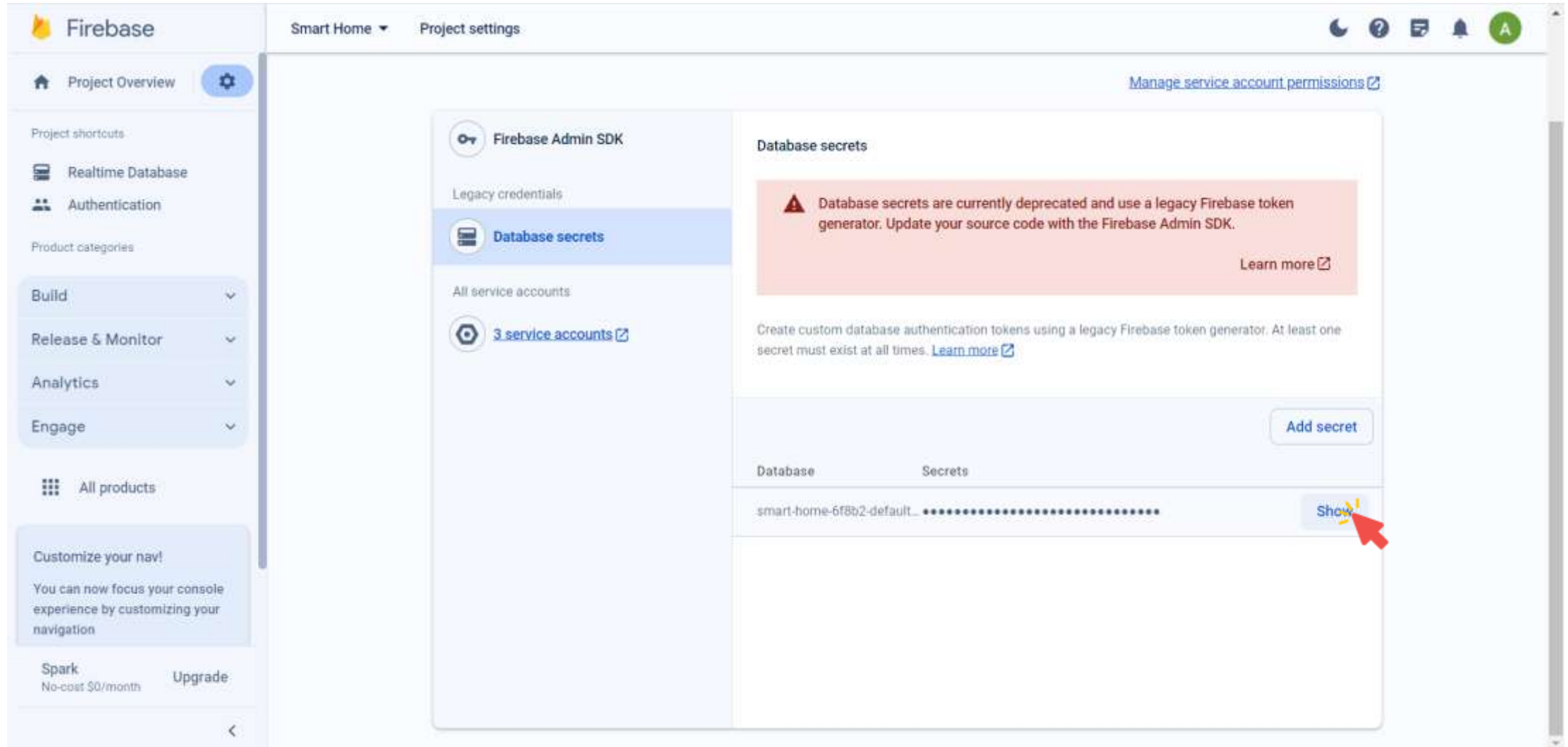
```
var admin = require("firebase-admin");

var serviceAccount = require("path/to/serviceAccountKey.json");

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: "https://smart-home-6f8b2-default-rtdb.firebaseio.com"
});
```

Firestore: Getting Database Secret

- Click on **Show**.

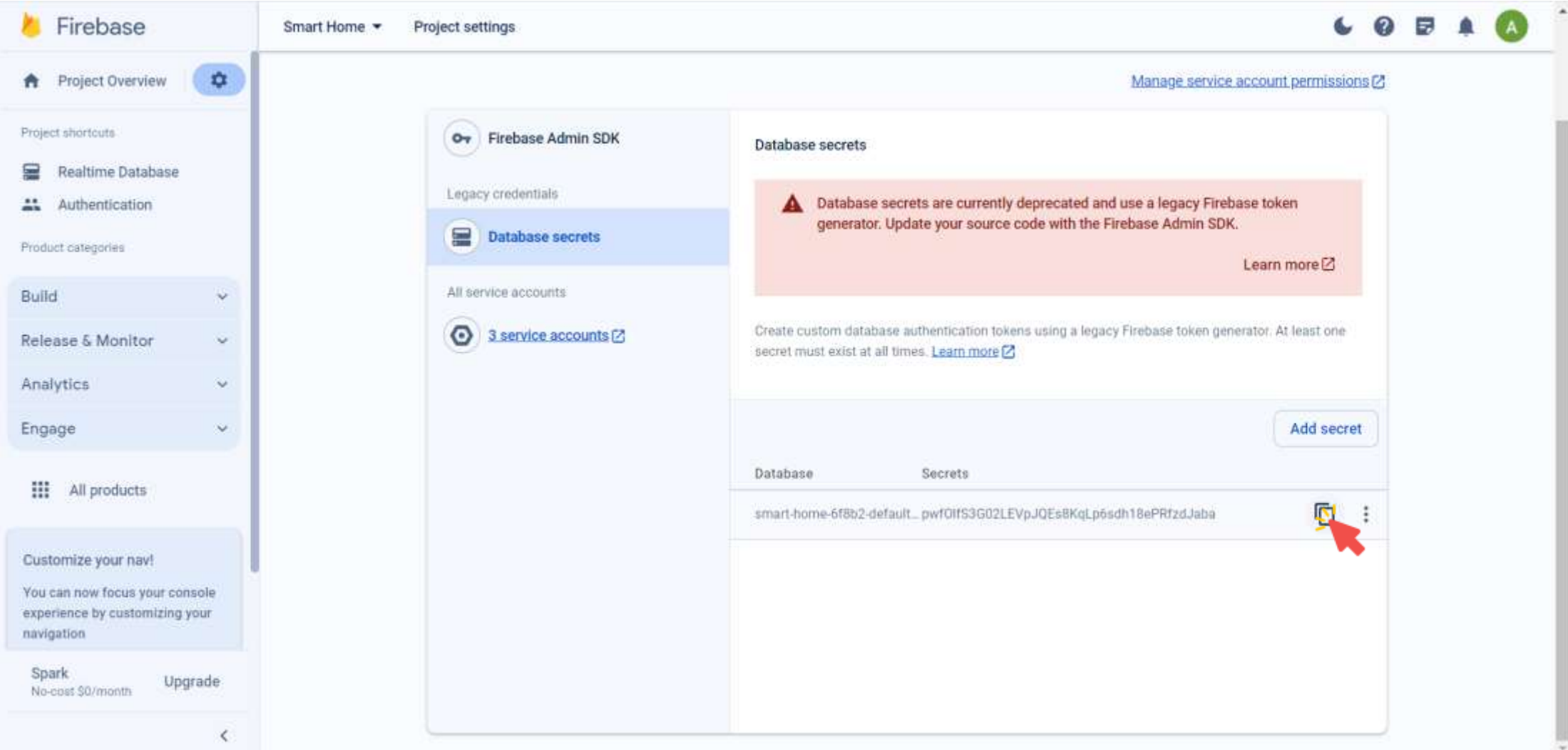


The screenshot shows the Firebase console interface for a project named "Smart Home". The left sidebar contains navigation options like "Project Overview", "Realtime Database", and "Authentication". The main content area is titled "Database secrets" and includes a warning message: "Database secrets are currently deprecated and use a legacy Firebase token generator. Update your source code with the Firebase Admin SDK." Below this, there is a table with columns "Database" and "Secrets". The first row shows the database name "smart-home-6f8b2-default_" and a masked secret value. A "Show" button is located to the right of the masked secret, and a red arrow points to it.

Database	Secrets
smart-home-6f8b2-default_ Show

Firestore: Getting Database Secret

- Copy the **Secret**.



Firebase: Structure Your Database

- All Firebase Realtime Database **data is stored as JSON objects**.
- You can think of the database as a **cloud-hosted JSON tree**.
- Unlike a SQL database, there are **no tables or records**.
- When you **add data to the JSON tree**, it becomes a **node in the existing JSON structure** with an associated key.

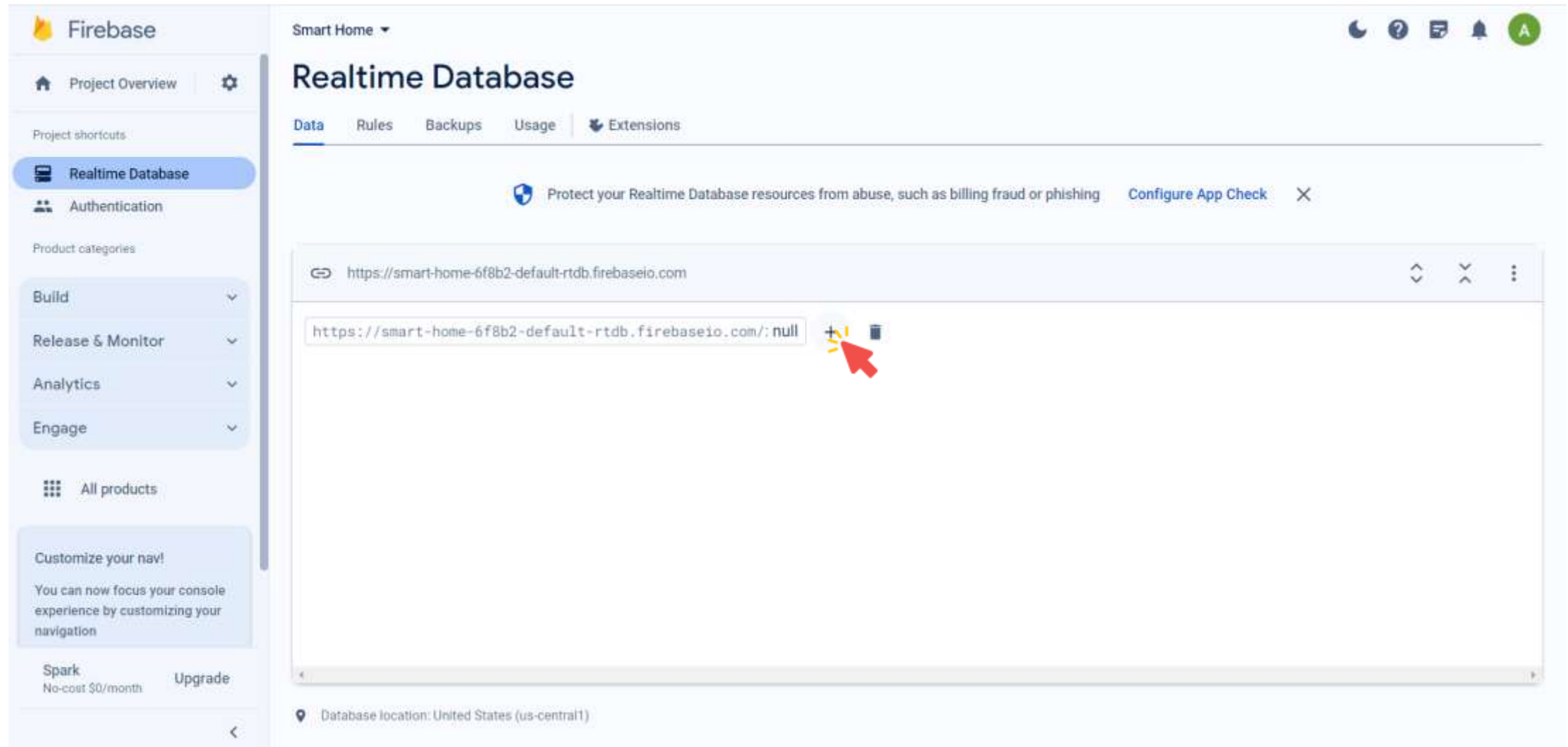
```
{
  "smartHome": {
    "kitchen": {
      "light": "off",
      "temperature": 20
    },
    "bedroom": {
      "light": "on",
      "temperature": 18
    }
  }
}
```


Firestore: Structure Your Database

```
{
  "smartHome": {
    "devices": {
      "light": {
        "status": "on",
        "brightness": 75
      },
      "thermostat": {
        "temperature": 22,
        "mode": "auto"
      },
      "door": {
        "status": "closed"
      }
    },
    "security": {
      "alarm": {
        "status": "armed"
      },
      "camera": {
        "status": "active"
      }
    }
  }
}
```

Firestore: Adding Nodes to Your Database

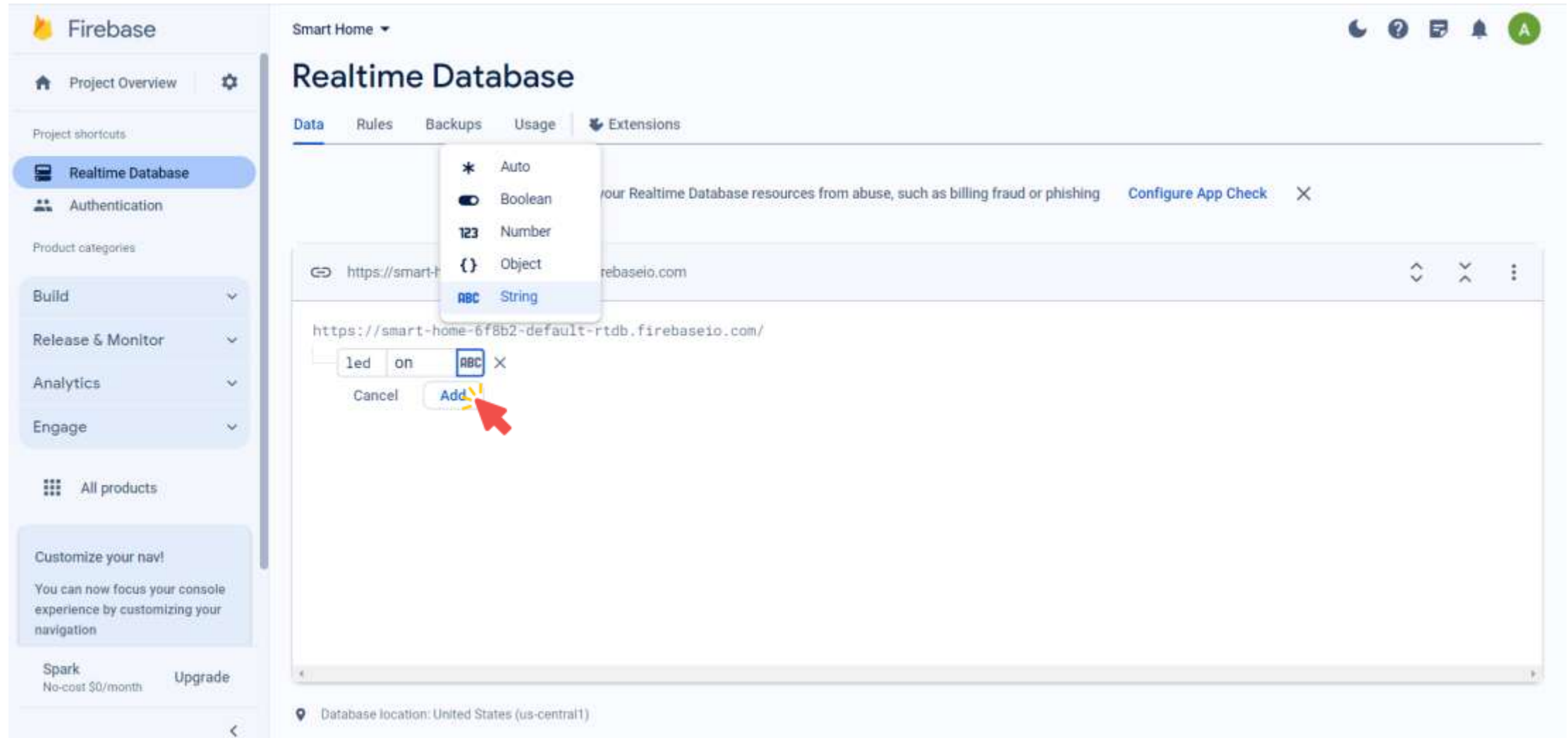
- Add a new [node](#).



The screenshot displays the Firebase Realtime Database console for a project named "Smart Home". The left sidebar shows the navigation menu with "Realtime Database" selected. The main content area shows the "Data" tab with a list of nodes. A red arrow points to a yellow plus icon next to the text "https://smart-home-6f8b2-default-rtdb.firebaseio.com/: null", indicating the process of adding a new node. The database location is noted as "United States (us-central1)".

Firestore: Adding Nodes to Your Database

- Enter node **key**, **value**, choose its **data type**, and click **Add**.

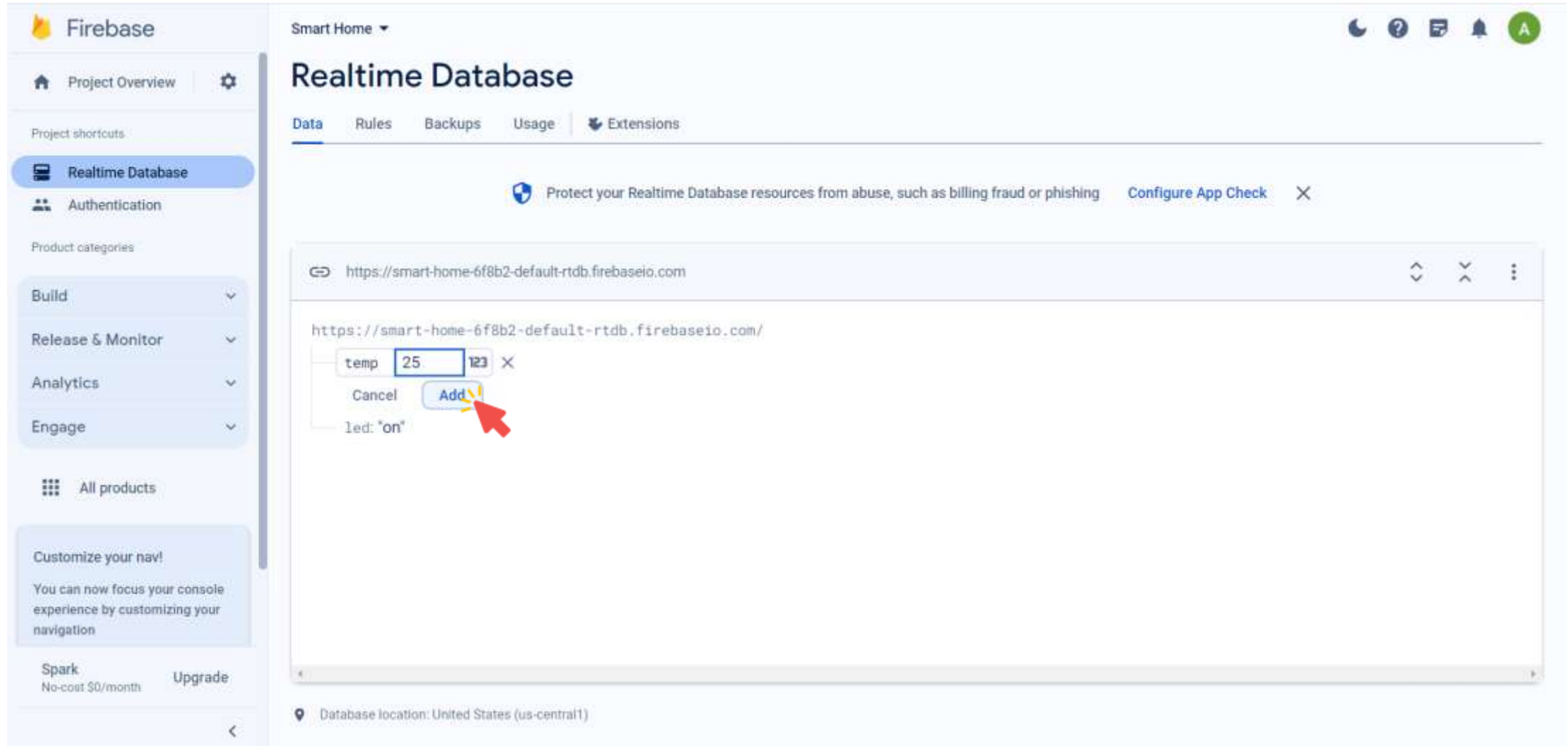


The screenshot shows the Firebase Realtime Database console for a project named "Smart Home". The "Data" tab is active, and a new node is being added. The URL bar shows the database path: `https://smart-home-6f8b2-default-rtdb.firebaseio.com/`. The "Add" dialog box is open, showing the key "led" and the value "on". The data type is set to "String" (indicated by "ABC" in a blue box). The "Add" button is highlighted with a red arrow, indicating the next step in the process.

Database location: United States (us-central1)

Firestore: Adding Nodes to Your Database

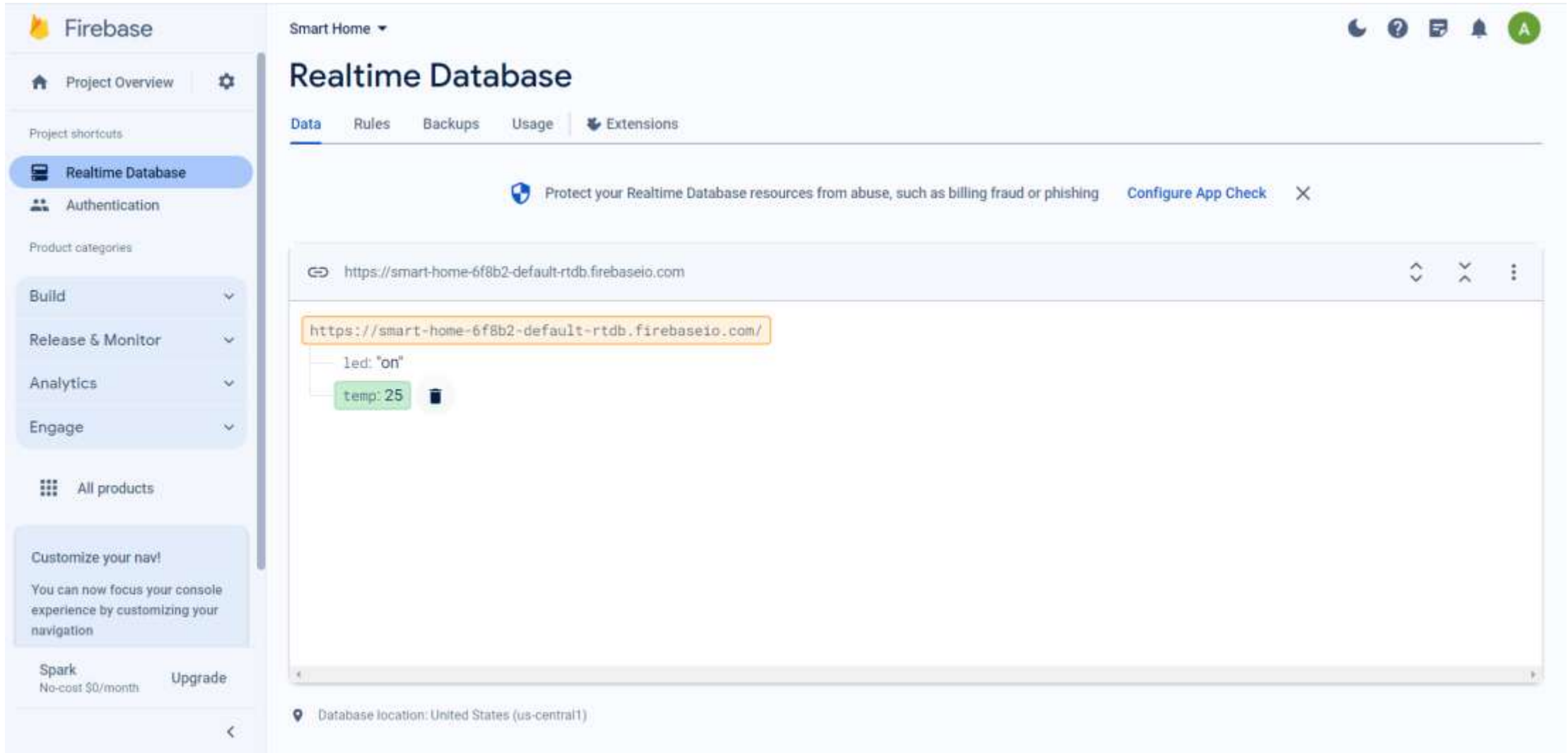
- Add a second node.



The screenshot shows the Firebase Realtime Database console for a project named "Smart Home". The left sidebar contains navigation options: Project Overview, Realtime Database (selected), Authentication, Build, Release & Monitor, Analytics, Engage, and All products. The main content area displays the "Realtime Database" interface with tabs for Data, Rules, Backups, Usage, and Extensions. A warning banner at the top right states: "Protect your Realtime Database resources from abuse, such as billing fraud or phishing" with a "Configure App Check" link. The browser address bar shows the URL: `https://smart-home-6f8b2-default-rtdb.firebaseio.com`. The main content area shows a data node being added to the path `https://smart-home-6f8b2-default-rtdb.firebaseio.com/`. The node is named "temp" and has a value of "25". A red arrow points to the "Add" button. Below the node, the text "led: 'on'" is visible. The bottom of the console shows the database location: "United States (us-central1)".

Firestore: Adding Nodes to Your Database

- Now, we have two nodes.



The screenshot displays the Firebase Realtime Database console for a project named "Smart Home". The left sidebar shows the navigation menu with "Realtime Database" selected. The main content area shows the "Data" tab with a tree view of the database structure. The root node is "https://smart-home-6f8b2-default-rtdb.firebaseio.com/". Underneath, there are two nodes: "led: on" and "temp: 25". The "temp: 25" node is highlighted with a green background. At the bottom, it indicates the database location is "United States (us-central1)".

Smart Home

Realtime Database

Data Rules Backups Usage Extensions

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

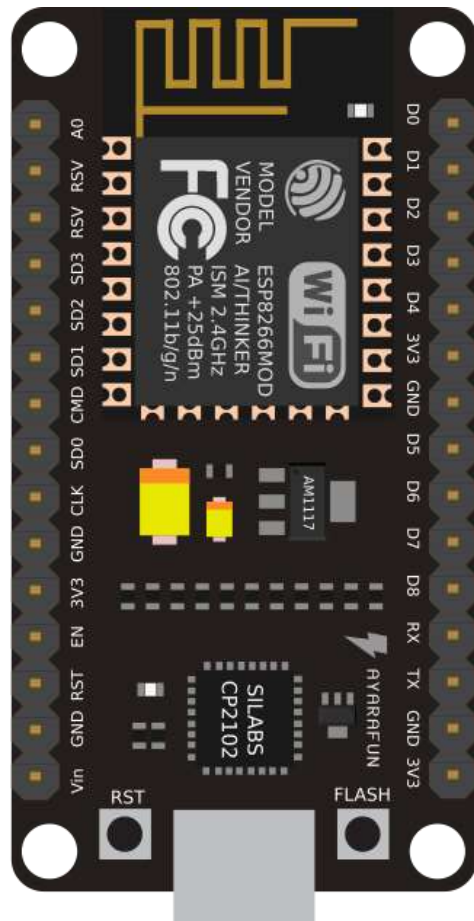
https://smart-home-6f8b2-default-rtdb.firebaseio.com/

```
https://smart-home-6f8b2-default-rtdb.firebaseio.com/  
├── led: "on"  
└── temp: 25
```

Database location: United States (us-central1)

FirestoreESP8266 Library

- The `FirestoreESP8266` library provides **Firestore Realtime database** and **Firestore Cloud Messaging** functions and supports ESP8266 MCU.



FirebaseESP8266 Library: Installation

- Download the **ZIP file** from the following link

<https://www.arduino-libraries.info/libraries/firebase-esp8266-client>

Firebase ESP8266 Client

Google Firebase Realtime Database Arduino Client Library for Espressif ESP8266

Author	Mobizt
Website	https://github.com/mobizt/Firebase-ESP8266
Category	Communication
License	MIT
Library Type	Contributed
Architectures	esp8266, esp32, sam, samd, stm32, STM32F1, STM32F4, teensy, avr, megaavr, mbed_nano, mbed_rp2040, rp2040

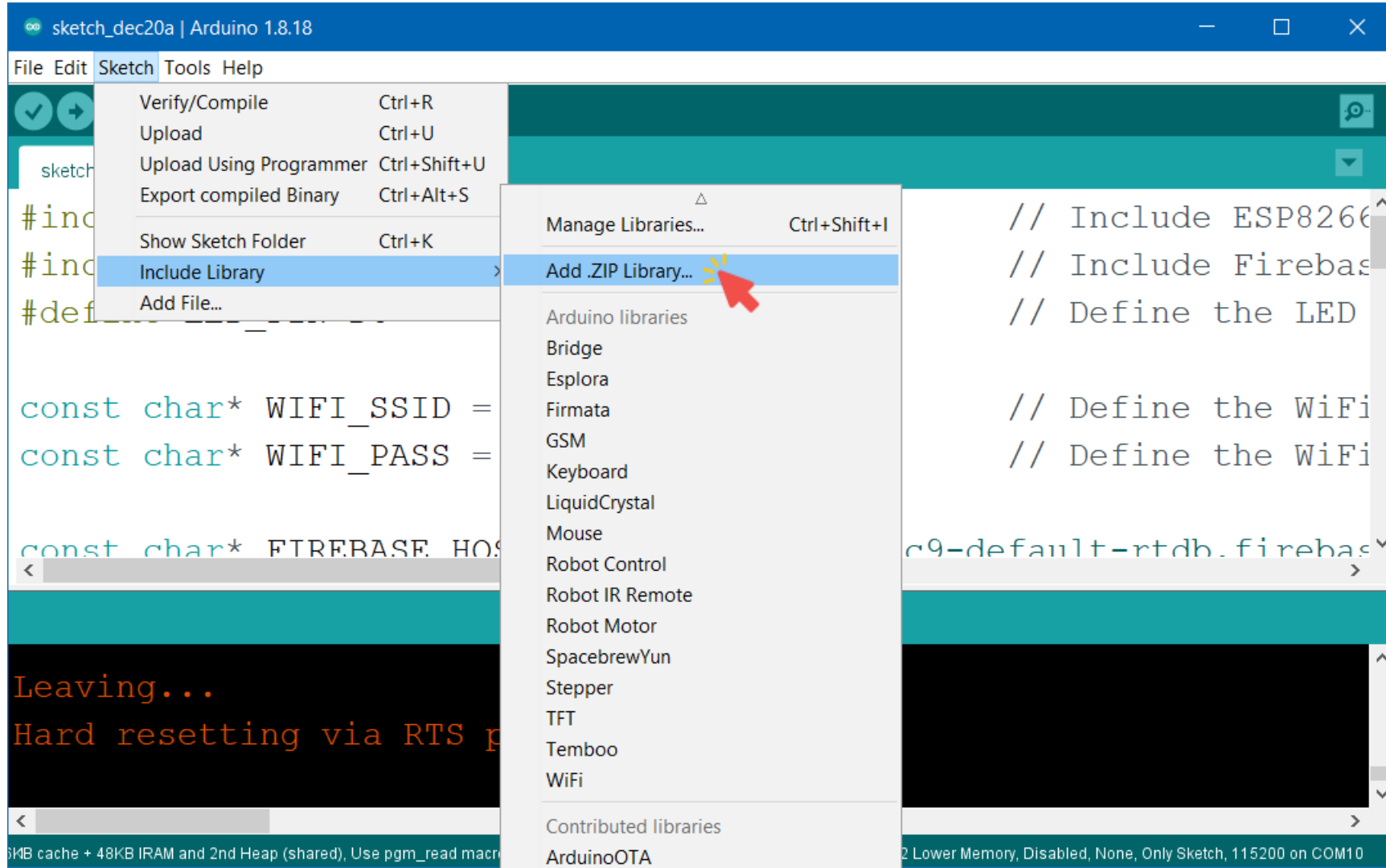
The secure, fast and reliable Firebase Realtime database library to read, store, update, delete, listen, backup, and restore data. You can also read and modify the database security rules with this library.

Downloads

Filename	Release Date	File Size
Firebase ESP8266 Client-4.3.19.zip	2023-07-29	2.01 MiB
Firebase ESP8266 Client-4.3.18.zip	2023-07-20	2.01 MiB

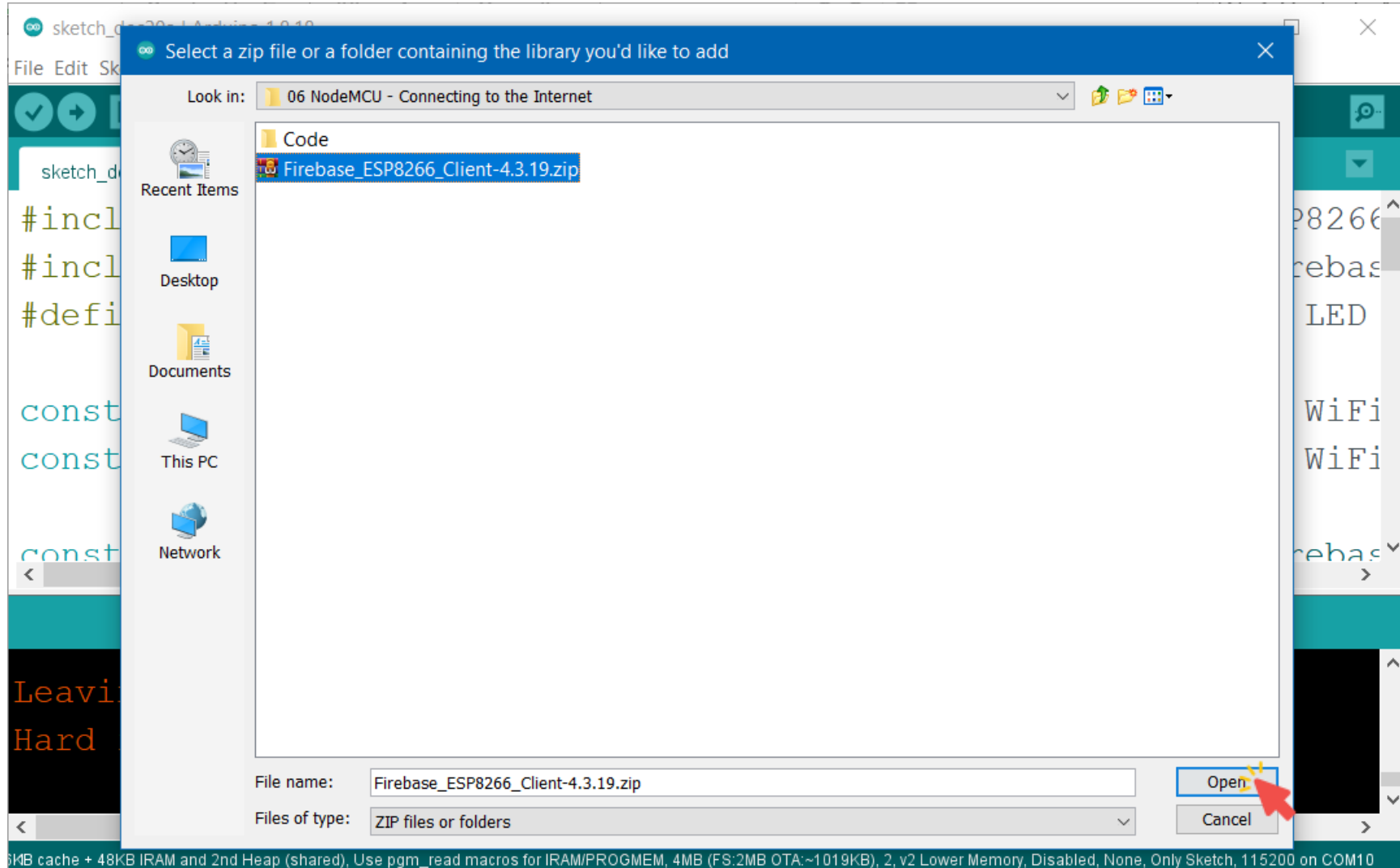
FirebaseESP8266 Library: Installation

- Select **Sketch** → **Include Library** → **Add .ZIP Library...**



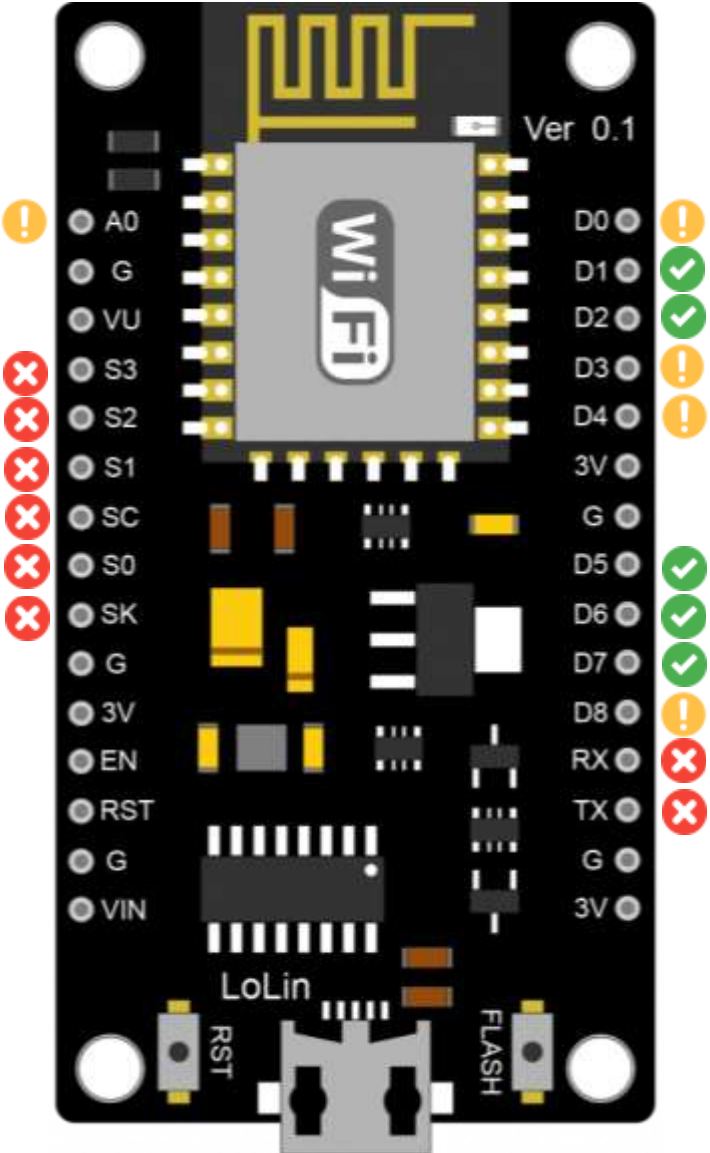
FirebaseESP8266 Library: Installation

- Choose **Firebase_ESP8266_Client-4.3.19.zip** that previously downloaded.

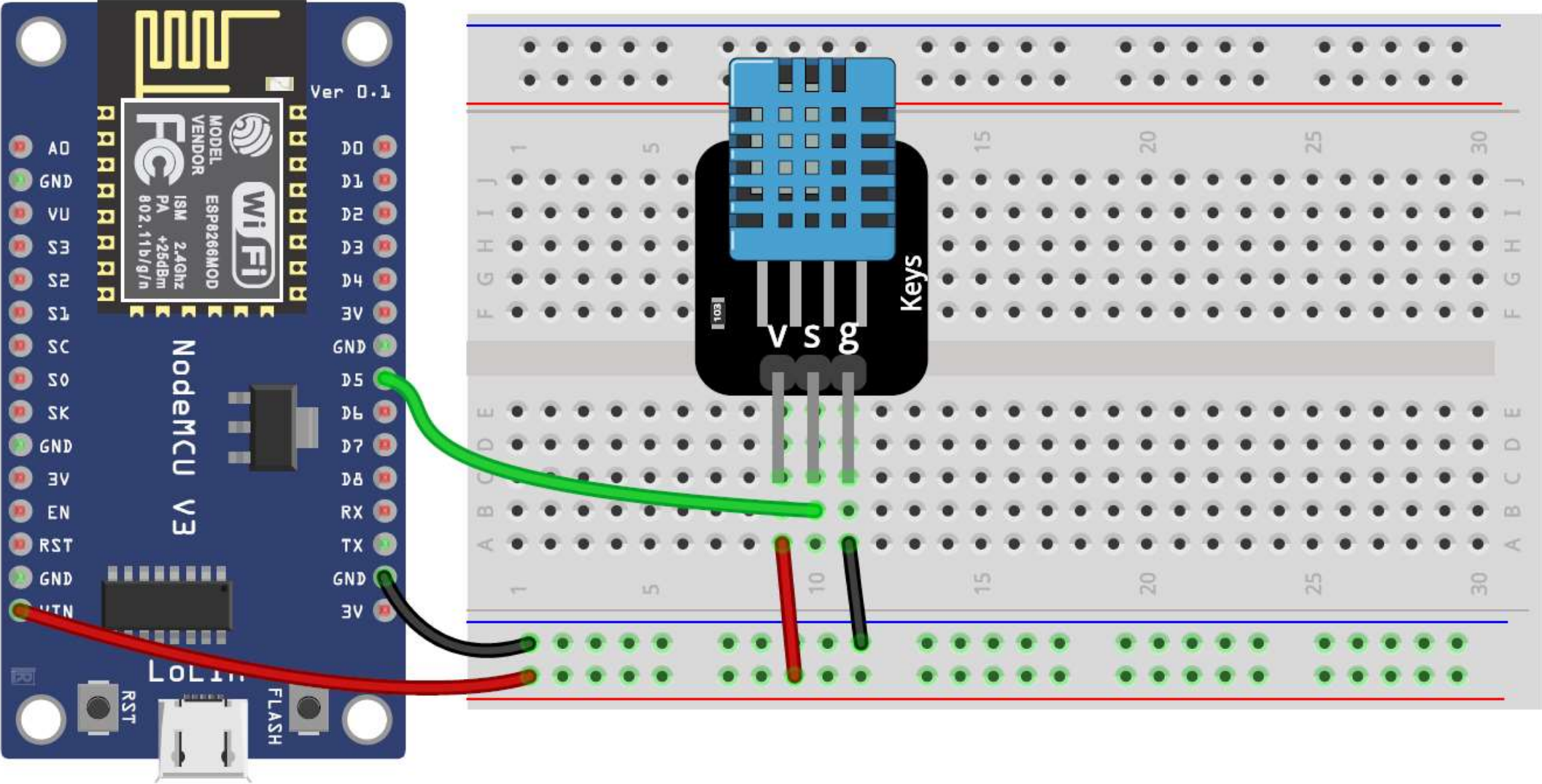


Sending Data to Firebase: NodeMCU ESP8266 Pinout

PIN	GPIO	Why Not Safe?
D0	GPIO16	HIGH at boot Used to wake up from deep sleep
D1	GPIO5	-
D2	GPIO4	-
D3	GPIO0	Connected to FLASH button Boot fails if pulled LOW
D4	GPIO2	HIGH at boot Boot fails if pulled LOW
D5	GPIO14	-
D6	GPIO12	-
D7	GPIO13	-
D8	GPIO15	Required for boot Boot fails if pulled HIGH

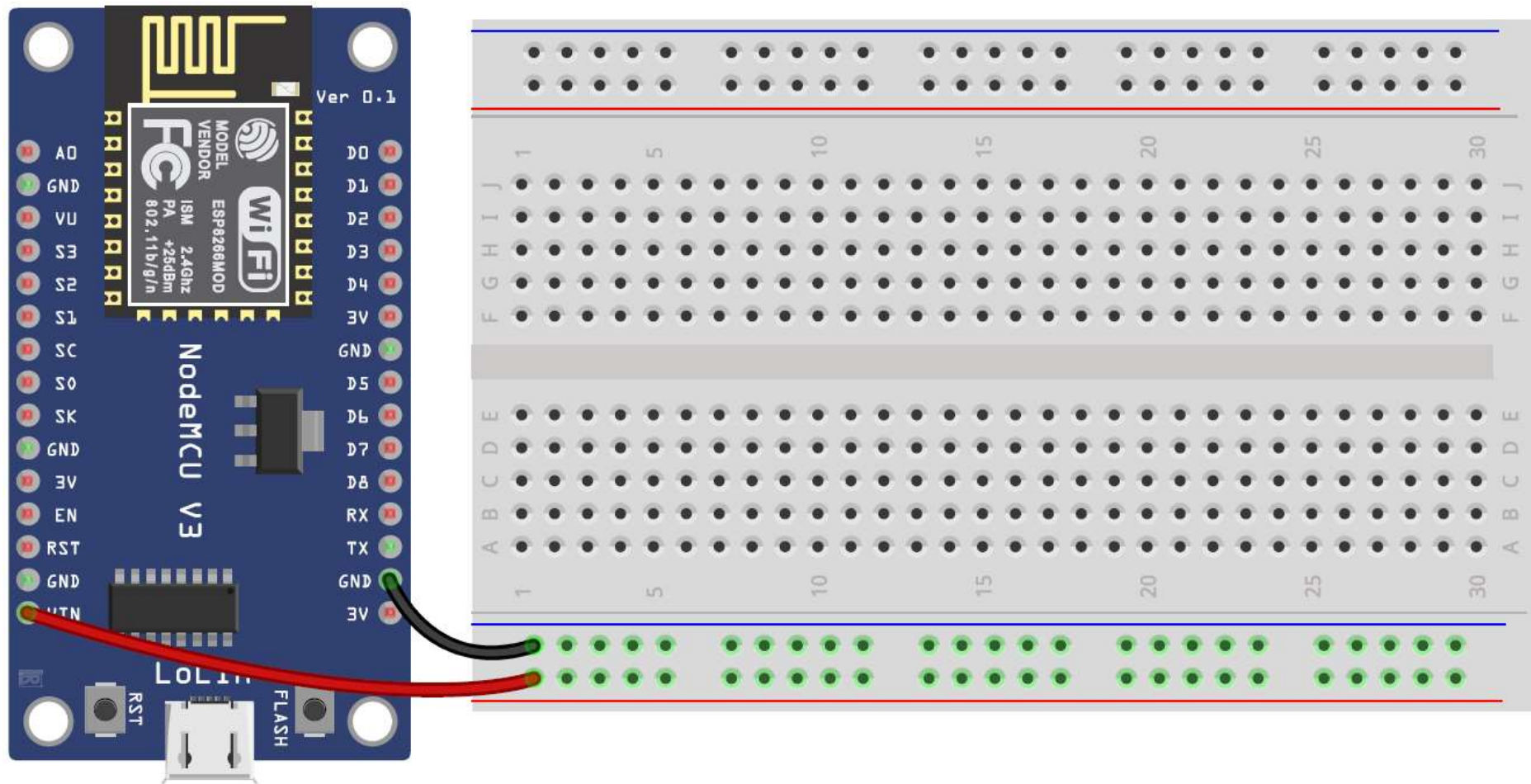


Sending Data to Firebase: DHT11 – Circuit



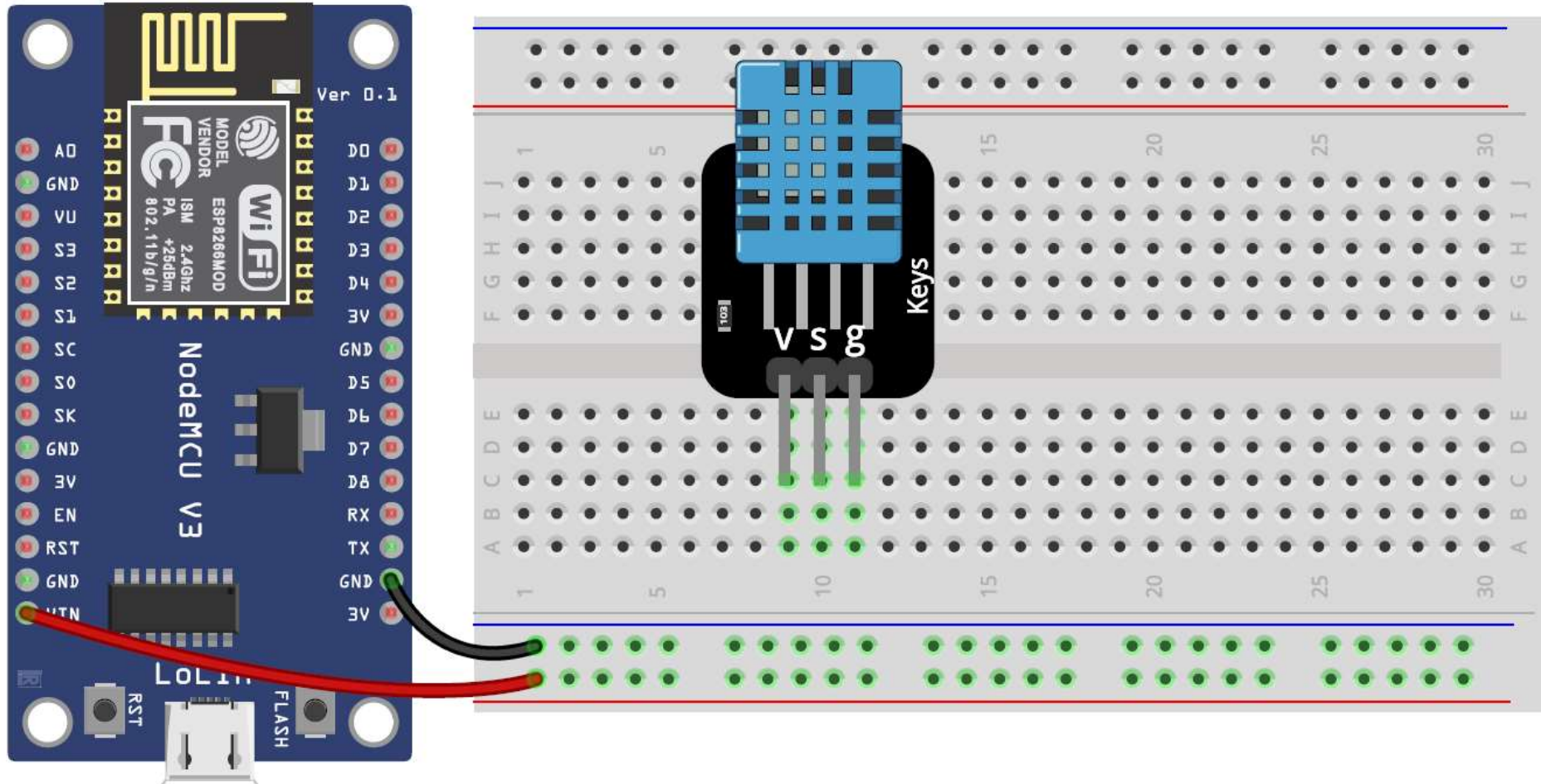
Sending Data to Firebase: DHT11 – Steps

1. Connect breadboard **power (+)** and **ground (-)** rails to NodeMCU **VIN** and **ground (GND)**, respectively.



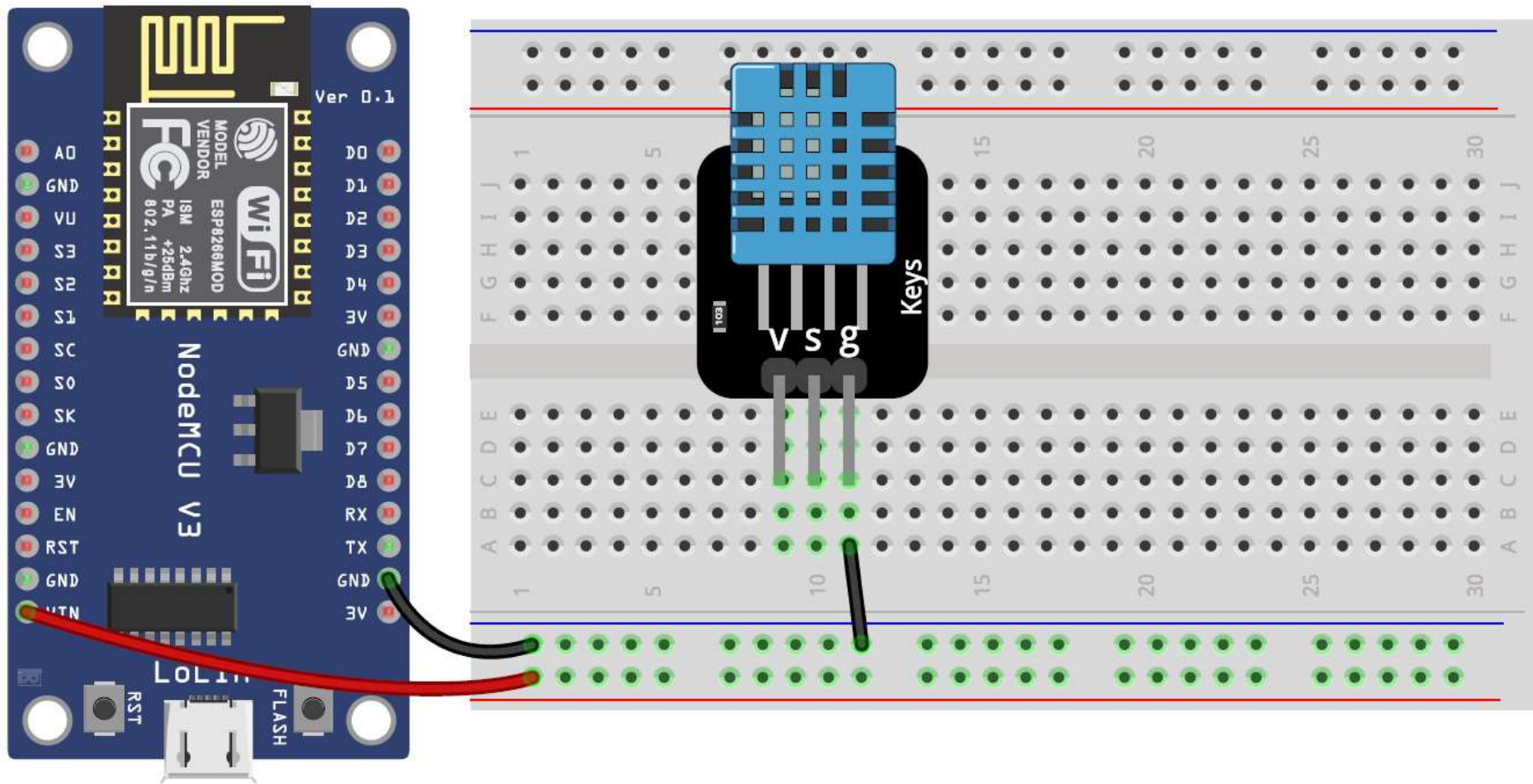
Sending Data to Firebase: DHT11 – Steps

2. Plug the **DHT11 sensor** into the breadboard.



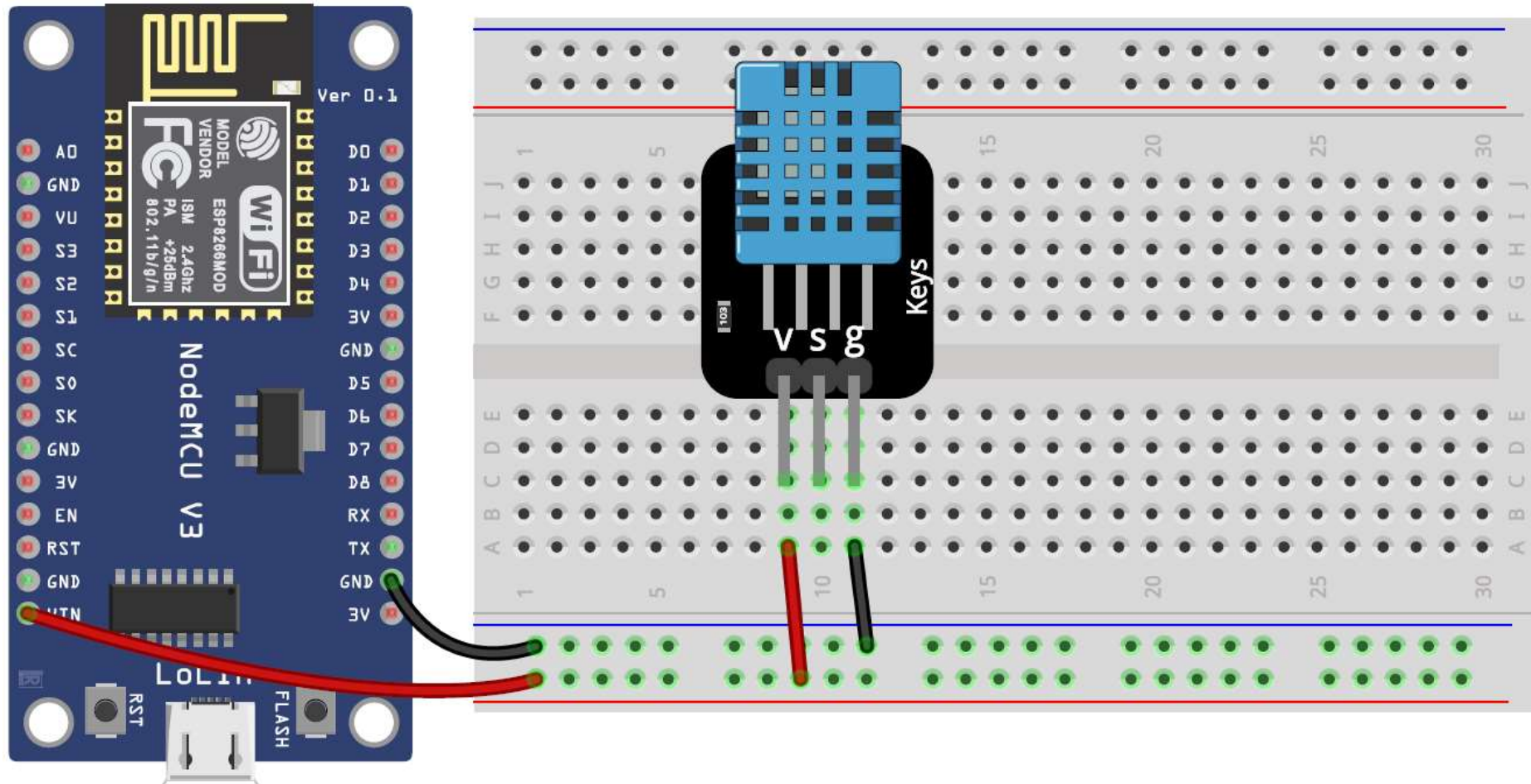
Sending Data to Firebase: DHT11 – Steps

3. The sensor **GND** pin connects to the **ground** on NodeMCU.



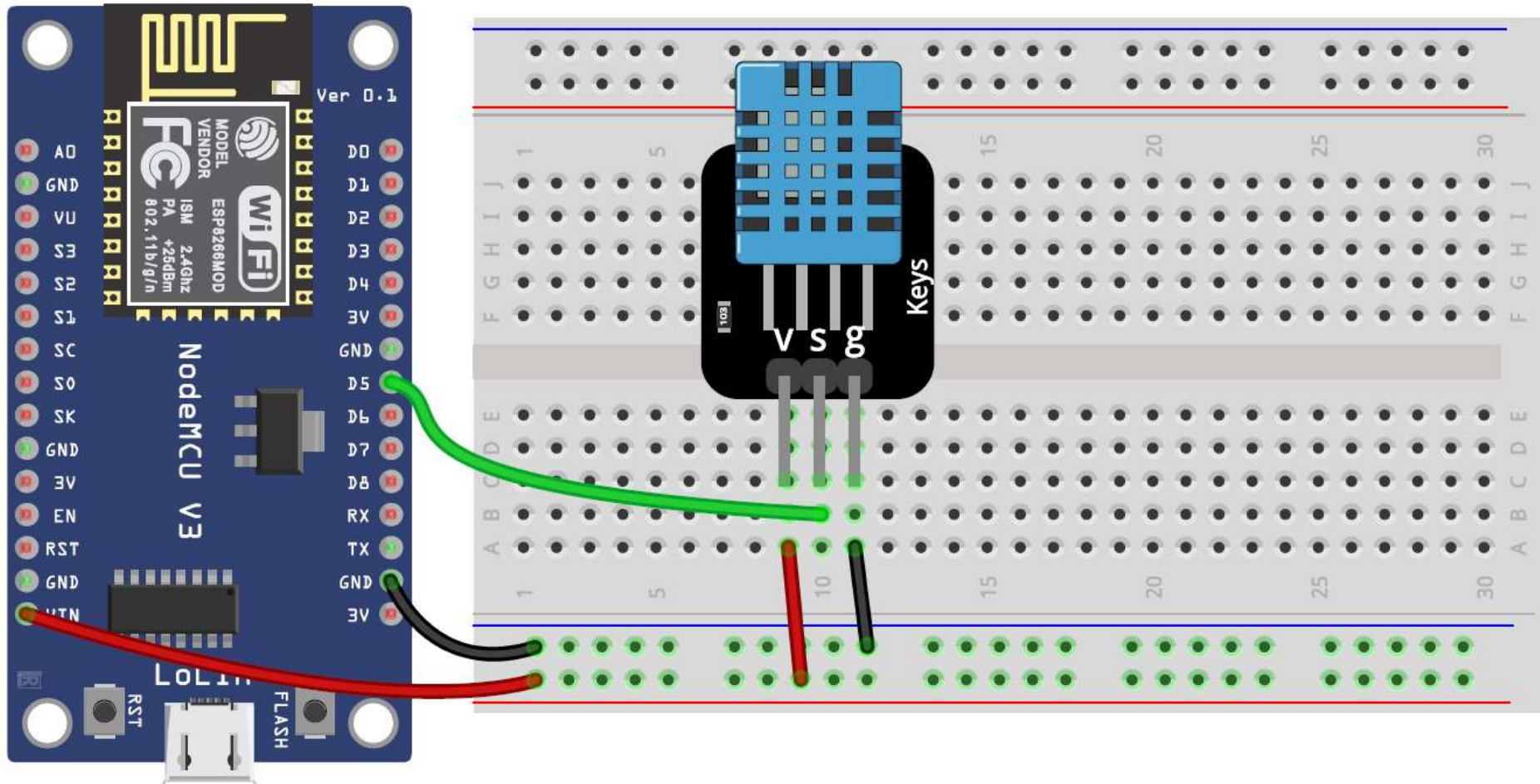
Sending Data to Firebase: DHT11 – Steps

4. The sensor **Power** pin connects to the **VCC** on NodeMCU.



Sending Data to Firebase: DHT11 – Steps

5. Wire up the sensor **Data** pin to the analog pin **D5** on NodeMCU.



Sending Data to Firebase: Code

```
#include <ESP8266WiFi.h> // Include ESP8266WiFi library for WiFi features
#include <FirebaseESP8266.h> // Include FirebaseESP8266 library for Firebase integration
#include "DHT.h" // Include DHT sensor library
#define DHT_PIN D5 // Define the digital pin connected to the DHT sensor
DHT dht(DHT_PIN, DHT11); // Initialize DHT sensor object with pin and sensor type

const char* WIFI_SSID = "iotlab"; // Define the WiFi network SSID
const char* WIFI_PASS = "hostiotlab"; // Define the WiFi network password

// Firebase Realtime Database URL and secret
const char* FIREBASE_HOST = "smart-home-6f8b2-default-rtdb.firebaseio.com";
const char* FIREBASE_AUTH = "pwf01fS3G02LEVpJQEs8KqLp6sdh18ePRfzdJaba";

FirebaseData fbdo; // Define Firebase Data object
```

Sending Data to Firebase: Code

```
void setup() {  
    Serial.begin(115200);           // Start serial communication at 115200 baudrate  
    WiFi.begin(WIFI_SSID, WIFI_PASS); // Begin WiFi connection using SSID and password  
  
    while(WiFi.status() != WL_CONNECTED){ // Check if the WiFi status is not connected  
        delay(1000); // Wait 1 second between WiFi connection checks  
        Serial.println("Connecting to WiFi..."); // Print message indicating an attempt to connect to WiFi  
    }  
  
    Serial.println("Connected to WiFi."); // Print message when WiFi connection is successful  
    Serial.print("IP Address: "); // Print the label for the IP address  
    Serial.println(WiFi.localIP()); // Print the assigned IP address  
  
    dht.begin(); // Start DHT sensor  
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); // Initialize Firebase connection  
    Firebase.reconnectWiFi(true); // Automatic reconnection to WiFi if connection is lost  
}
```


Sending Data to Firebase: Code

```
void loop() {  
    delay(2000);           // Wait 2 seconds between measurements  
    float temp = dht.readTemperature(); // Read temperature in Celsius  
  
    if(Firebase.setFloat(fbdo, "/temp", temp)){ // Set temperature value in the Firebase under the "/temp" path  
        Serial.print("Temperature: "); // Print the label "Temperature: "  
        Serial.print(temp); // Print the temperature value  
        Serial.println("°C "); // Print the unit "°C"  
    }  
    else // If Firebase operation fails,  
        Serial.println(fbdo.errorReason()); // Print the error reason  
}
```

Sending Data to Firebase: Output

Firebase

Project Overview

Project shortcuts

- Realtime Database**
- Authentication

Product categories

- Build
- Release & Monitor
- Analytics
- Engage

All products

Customize your nav!
You can now focus your console experience by customizing your navigation

Spark No-cost \$0/month Upgrade

Smart Home ▾

Realtime Database

Data Rules Backups

<https://smart-home-6f8b2...>

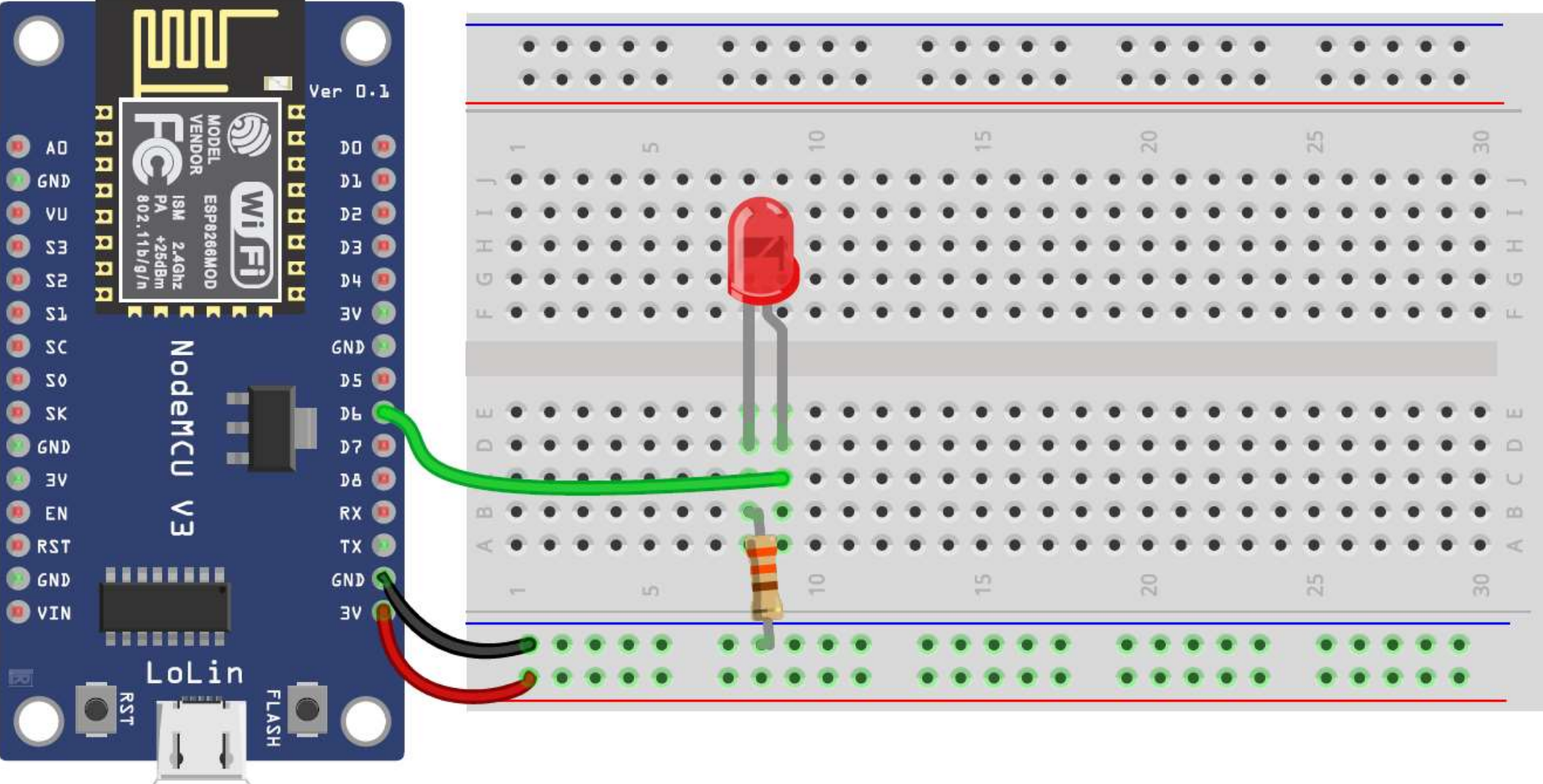
- <https://smart-home-6f8b2...>
 - led: "on"
 - temp: 21.9

```
COM10
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connected to WiFi.
IP Address: 192.168.137.34
Temperature: 22.00°C
Temperature: 22.00°C
Temperature: 21.90°C
```

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

Database location: United States (us-central1)

Reading Data from Firebase: Controlling an LED from Anywhere



Reading Data from Firebase: Code

```
#include <ESP8266WiFi.h> // Include ESP8266WiFi library for WiFi features
#include <FirebaseESP8266.h> // Include FirebaseESP8266 library for Firebase integration
#define LED_PIN D6 // Define the LED pin

const char* WIFI_SSID = "iotlab"; // Define the WiFi network SSID
const char* WIFI_PASS = "hostiotlab"; // Define the WiFi network password

// Firebase Realtime Database URL and secret
const char* FIREBASE_HOST = "smart-home-6f8b2-default-rtdb.firebaseio.com";
const char* FIREBASE_AUTH = "pwf01fS3G02LEVpJQEs8KqLp6sdh18ePRfzdJaba";

FirebaseData fbdo; // Define Firebase Data object
String led_status; // A variable to store LED status
```

Reading Data from Firebase: Code

```
void setup() {  
  Serial.begin(115200);           // Start serial communication at 115200 baudrate  
  WiFi.begin(WIFI_SSID, WIFI_PASS); // Begin WiFi connection using SSID and password  
  pinMode(LED_PIN, OUTPUT);      // Initialize the pin D6 as an output  
  
  while(WiFi.status() != WL_CONNECTED){ // Check if the WiFi status is not connected  
    delay(1000);                  // Wait 1 second between WiFi connection checks  
    Serial.println("Connecting to WiFi..."); // Print message indicating an attempt to connect to WiFi  
  }  
  
  Serial.println("Connected to WiFi."); // Print message when WiFi connection is successful  
  Serial.print("IP Address: ");        // Print the label for the IP address  
  Serial.println(WiFi.localIP());      // Print the assigned IP address  
  
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); // Initialize Firebase connection  
  Firebase.reconnectWiFi(true);           // Automatic reconnection to WiFi if connection is lost  
}
```

Reading Data from Firebase: Code

```
void loop() {  
  if(Firebase.getString(fbdo, "/led")){  
    led_status = fbdo.to<String>;  
  
    if(led_status == "on"){  
      digitalWrite(LED_PIN, HIGH);  
      Serial.println("LED is on.");  
    }  
    else if(led_status == "off"){  
      digitalWrite(LED_PIN, LOW);  
      Serial.println("LED is off.");  
    }  
  }  
  else  
    Serial.println(fbdo.errorReason());  
  
  delay(1000);  
}
```

// Try to get the LED status from the Firebase
// Get the LED status from the Firebase

// If LED status is "on",
// Turn on LED
// Print LED status

// If LED status is "off",
// Turn off LED
// Print LED status

// If Firebase operation fails,
// Print the error reason

// Check Firebase for LED status every 1 second

Reading Data from Firebase: Output

Firestore

Project Overview

Project shortcuts

- Realtime Database
- Authentication

Product categories

- Build
- Release & Monitor
- Analytics
- Engage

All products

Customize your nav!
You can now focus your console experience by customizing your navigation

Spark No-cost \$0/month Upgrade

Smart Home

Realtime Database

Data Rules Backups

```
https://smart-home-6f8b2...
├── led: "on"
└── temp: 21.4
```

COM10

```
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
Connected to WiFi.
IP Address: 192.168.137.34
LED is off.
LED is off.
LED is off.
LED is off.
LED is off.
LED is on.
```

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

Database location: United States (us-central1)

Firebase ESP Client Library

- The **Firebase ESP8266 library** provides Firebase Realtime database and Firebase Cloud Messaging functions and supports only ESP8266 MCU.

[Firebase ESP8266 - GitHub](#)

[Firebase ESP32 - GitHub](#)

- If you use other Arduino devices or want to **use more Firebase services** included Firestore database, Firebase Storage, Google Cloud Storage and Cloud Functions for Firebase, use **Firebase ESP Client library** instead.

[Firebase ESP Client - GitHub](#)

References and Tutorials

- [ESP32 WiFi Tutorial & Library Examples \(Arduino IDE\)](#)
- [ESP8266 NodeMCU Access Point \(AP\) for Web Server](#)
- [ESP8266 NodeMCU: Getting Started with Firebase](#)
- [Firestore Realtime Database Arduino Library for ESP8266](#)
- [Firestore ESP8266 Client](#)
- [Firestore - Structure Your Database](#)
- [Simple Example to Store and Read Data from the Firestore](#)