

概要

- 20211013 (水) 13:05 ~
- Tello を音声で制御
- Google Home Mini + IFTTT + Firebase Realtime Database + Tello
- Node.js

参考サイト

- [Google Homeに話しかけてドローンを音声操作してみる](#)

Tello 制御のための設定

ディレクトリ作成

```
$ mkdir Tello && cd &
$ mkdir google_tello && cd $_
$ nodenv local 16.10.0
```

UDP パッケージ・インストール

```
$ npm install dgram
```

Tello 制御プログラム（音声なし）

`test.js` を作成し以下を記述

```
"use strict"

const dgram = require("dgram")
const sock = dgram.createSocket("udp4")

// Telloの固定情報
const tello = {
    ip: "192.168.10.1",
    port: 8889
}

// Telloにコマンド送信する関数
const send = async (buf, ms = 0) => {
    console.log(buf)
    const command = new Buffer(buf)
    sock.send(command, 0, command.length, tello.port, tello.ip)
    await wait(ms)
}
const wait = ms => new Promise(res => setTimeout(res, ms))

// Telloに操作したいコマンドを記述
const main = async () => {
    await send("command", 100)
    await send("takeoff", 4000)
    await send("land")
    process.exit()
}
main()
```

実行

Tello を起動し、RaspberryPi と繋げる

```
node test.js
```

Firebase Realtime Database

1. プロジェクトを作成 ボタンをクリック

The screenshot shows the Firebase homepage with a blue header. Below it, a large central area features the text "Firebaseへようこそ" (Welcome to Firebase) and "Googleが提供する、アプリインフラストラクチャの構築、アプリの品質向上、ビジネスの成長のためのツール" (Tools for building app infrastructure, improving app quality, and driving business growth). At the bottom left is a white button labeled "プロジェクトを作成". To its right is another button with a document icon and the text "ドキュメントを表示". On the right side of the screen, there are two people: one sitting at a desk working on a laptop, and another standing and writing on a chalkboard.

デモプロジェクトを見る

プロジェクトを作成

ドキュメントを表示

Firebase プロジェクトがアプリのコンテナとなります
プロジェクト内のアプリは、Realtime Database やアナリティクスなどの機能を共有しています。

詳細

iOS

2. プロジェクト名を入力し 続行 ボタンをクリック

The screenshot shows the first step of creating a project. It has a white header with a close button and the text "プロジェクトの作成 (手順 1/3)". Below this, there is a large text area with the instruction "まず、プロジェクトに名前を付けましょう^①". A text input field is present with the placeholder "プロジェクト名を入力します" and the value "my-awesome-project-id". At the bottom is a grey button labeled "続行". On the right side of the screen, there is an illustration of two people: a woman in a yellow shirt working on a laptop, and a man in a suit holding a smartphone.

×

プロジェクトの作成 (手順 1/3)

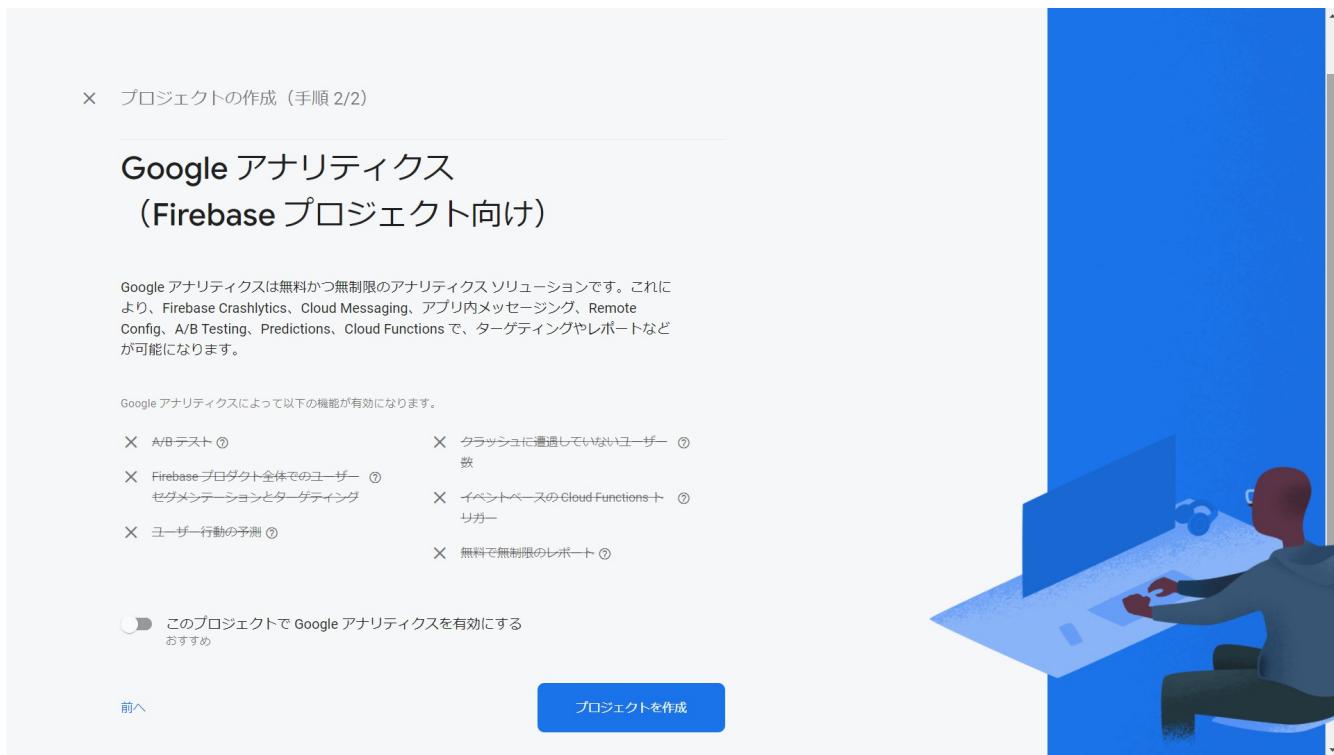
まず、
プロジェクトに名前を付けましょう^①

プロジェクト名を入力します

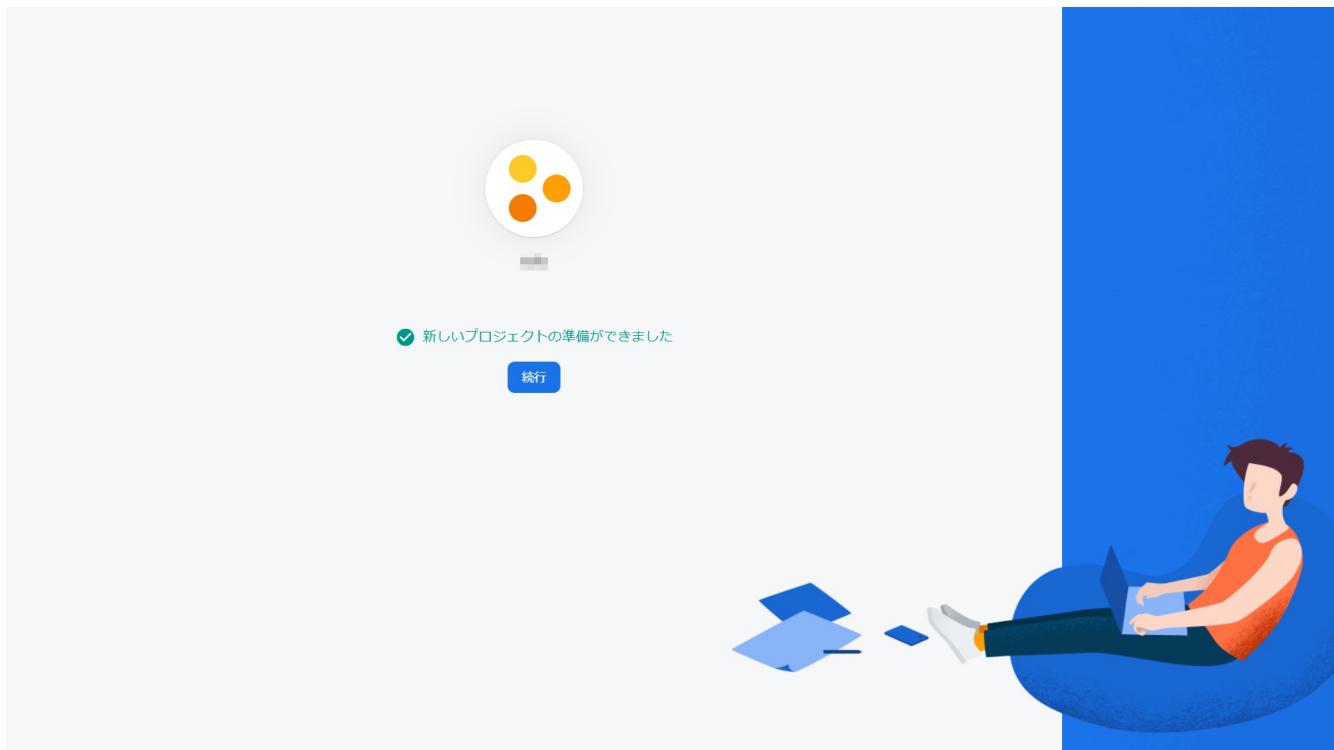
my-awesome-project-id

続行

3. Google アナリティクスを無効にして プロジェクトを作成 ボタンをクリック



4. 新しいプロジェクトの準備ができましたと表示されたら 続行



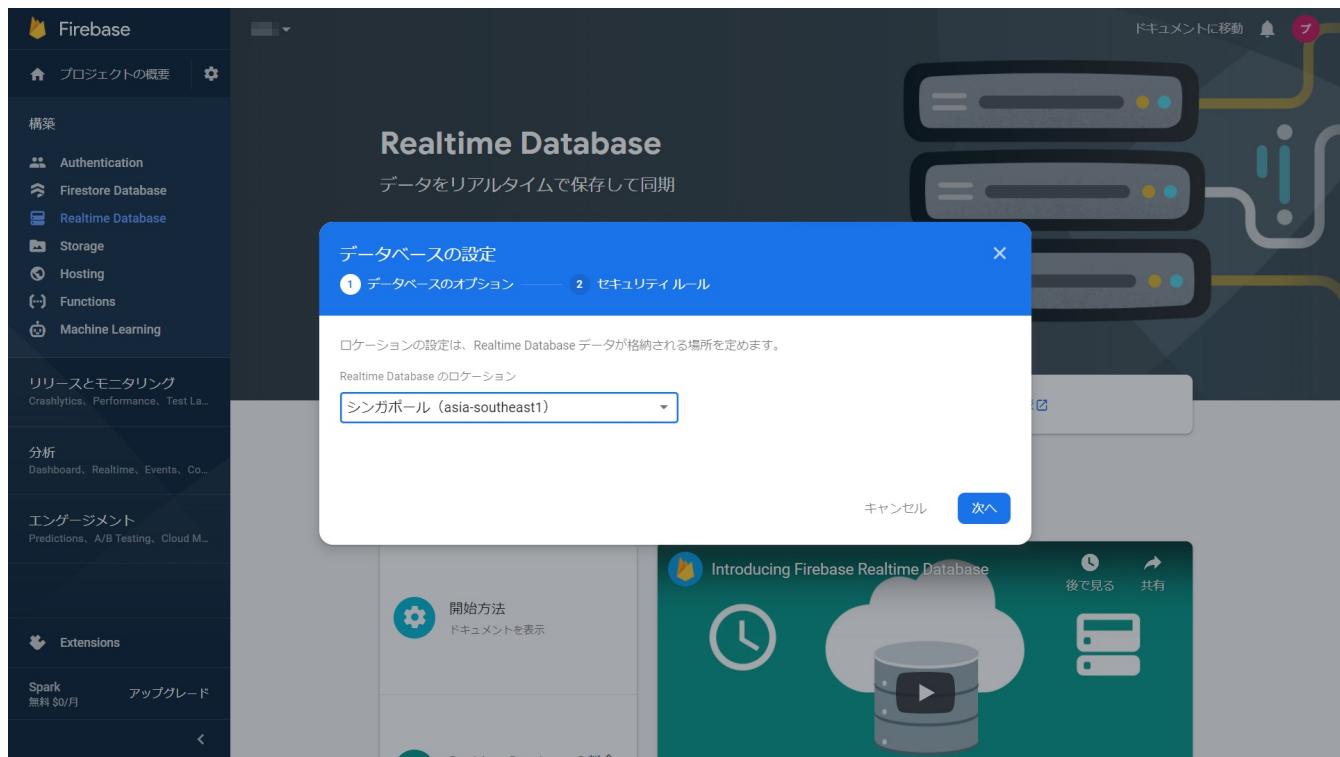
5. サイドバーの **Realtime Database** リンクをクリック

The screenshot shows the Firebase console interface. On the left, a sidebar menu includes 'Authentication', 'Firestore Database', 'Realtime Database' (which is highlighted in red), 'Storage', 'Hosting', 'Functions', and 'Machine Learning'. Below these are sections for 'リリースとモニタリング' (Crashlytics, Performance, Test Lab...), '分析' (Dashboard, Realtime, Events, Co...), and 'エンゲージメント' (Predictions, A/B Testing, Cloud M...). At the bottom of the sidebar are 'Extensions' and 'Spark' (無料 \$0/月) with an 'アップグレード' button. The main content area features a large blue banner with the text 'アプリに Firebase を追加して利用を開始しましょう' (Add Firebase to your app and start using it) and illustrations of two people interacting with a screen. Below the banner, a call-to-action button says 'データを瞬時に保存して同期' (Save data instantly and sync) with icons for iOS, Android, and Cloud Functions. A modal window titled 'Realtime Database' is open, showing a diagram of three database nodes connected by lines, and a message asking if it's suitable for the purpose.

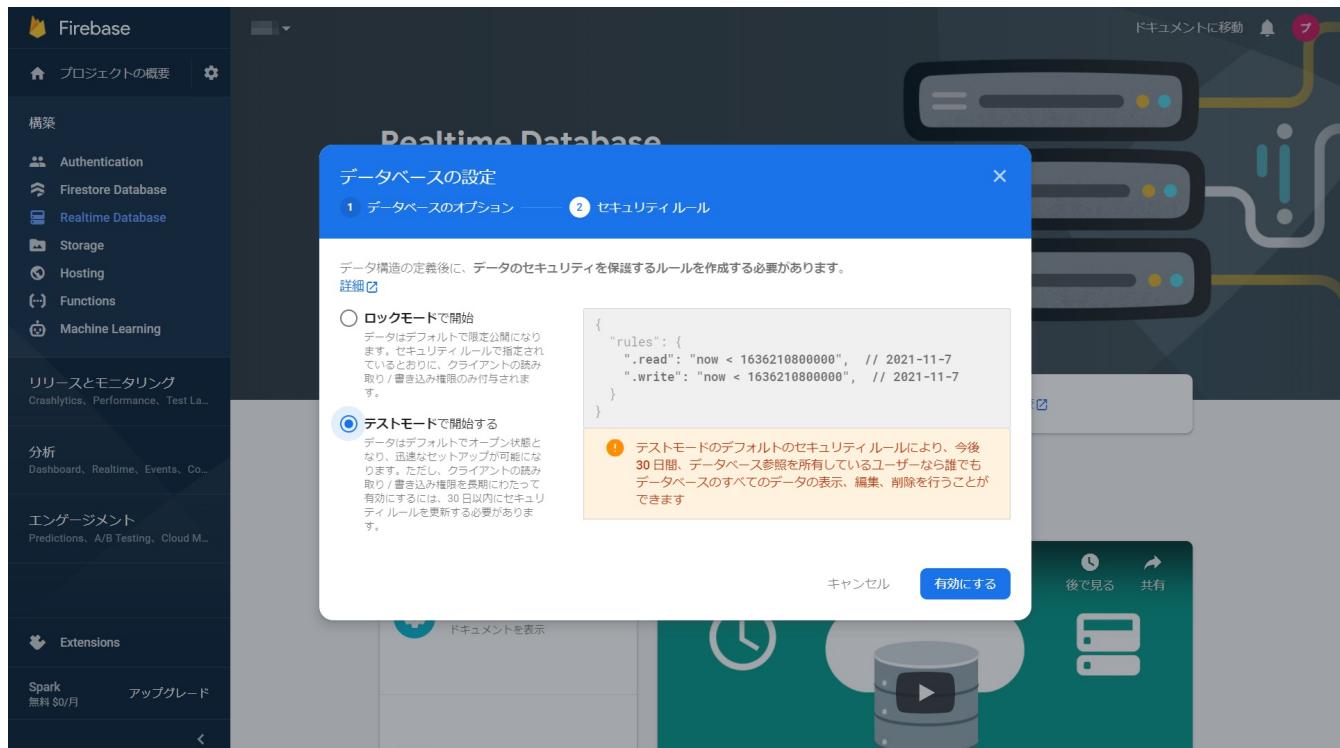
6. 'データベースを作成'ボタンをクリック

The screenshot shows the 'Realtime Database' creation page. The sidebar remains the same as the previous screenshot. The main area has a dark background with a central 'Realtime Database' heading and a sub-headline 'データをリアルタイムで保存して同期' (Save data instantly and sync). A prominent button labeled 'データベースを作成' (Create database) is centered. To the right, there's a diagram of three database nodes. Below the button, a message asks if the service is suitable for the purpose, with a 'データベースを比較' (Compare databases) link. At the bottom, there's a '詳細' (Details) section with a '開始方法' (Getting started) link and a video thumbnail titled 'Introducing Firebase Realtime Database'.

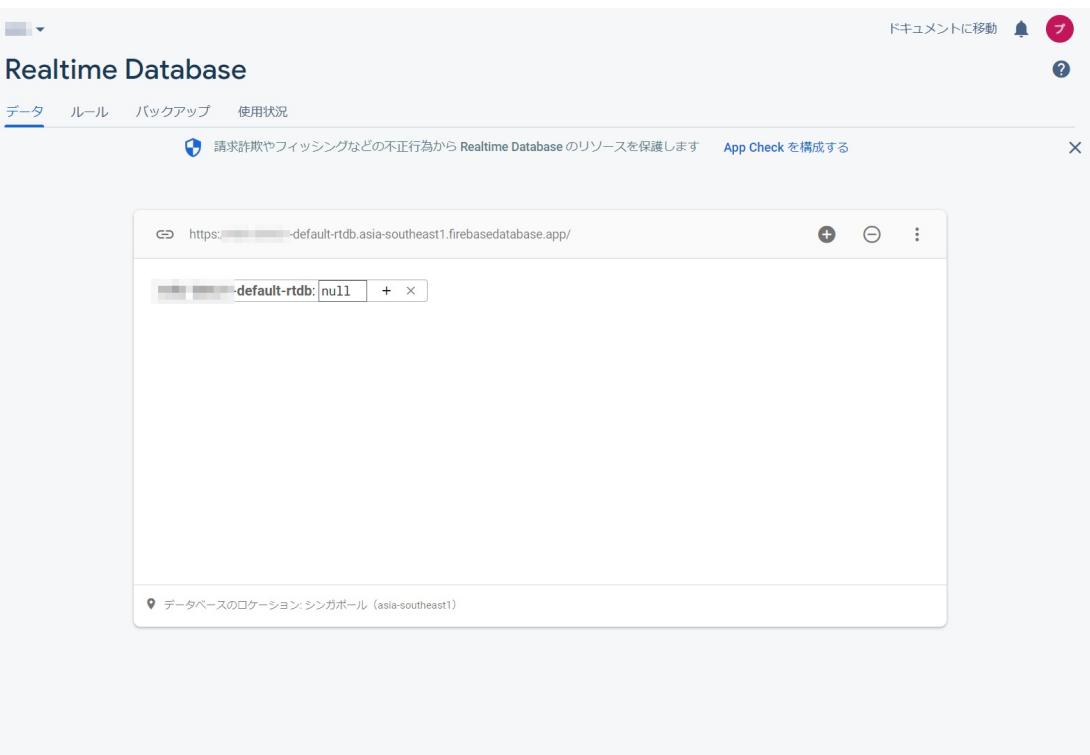
7. データベースのロケーション（今回はシンガポールにしてみました）を選び 次へ ボタンをクリック



8. 「テストモードで開始する」にチェックを入れ、有効にするボタンをクリック

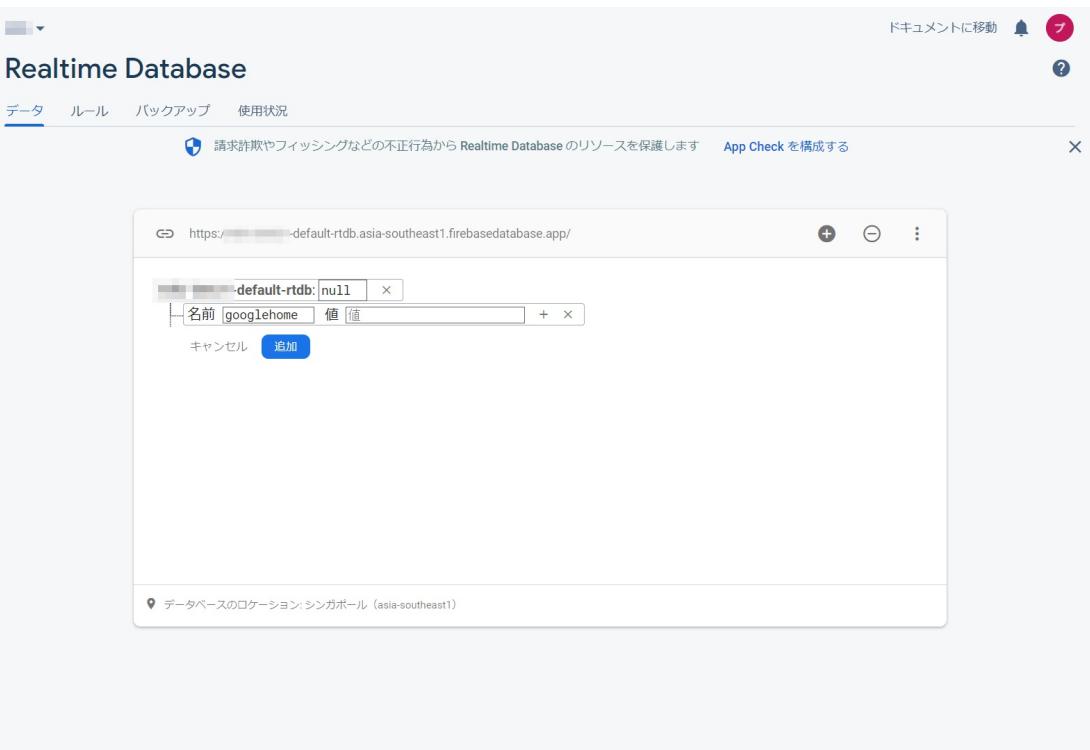


9.  をクリック



The screenshot shows the Firebase Realtime Database console. On the left, there's a sidebar with various services: Authentication, Firestore Database, Realtime Database (which is selected), Storage, Hosting, Functions, and Machine Learning. Below that are sections for Releases & Monitoring, Analytics, and Engage. At the bottom, there's an Extensions section and a Spark plan option. The main area is titled "Realtime Database" and has tabs for Data, Rules, Backup, and Usage. It shows a single child node under the root: "名前: null". The URL in the browser bar is https://[REDACTED]-default-rtdb.firebaseio.com/.json.

10. 名前に  と入力し  をクリック



This screenshot shows the same Firebase Realtime Database interface as the previous one, but with a new child node added. The "名前" node now has a value of "googlehome". The URL in the browser bar remains https://[REDACTED]-default-rtdb.firebaseio.com/.json.

11. 次の段の 名前 に word、 値 に "" を入力し、 追加 ボタンをクリック

The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with various services: Authentication, Firestore Database, Realtime Database (which is selected), Storage, Hosting, Functions, Machine Learning, Releases & Monitoring, Analytics, and Engage. At the bottom, there are sections for Extensions, Spark (free \$0/month), and Upgrade.

The main area is titled "Realtime Database". It shows a tree structure with a root node "-default-rtdb" containing a child node "googlehome". Under "googlehome", a new child node "word" is being added. A modal dialog is open, showing the path "-default-rtdb->googlehome->word" and fields for "名前" (name) set to "word" and "値" (value) set to an empty string (""). A blue "追加" (Add) button is visible at the bottom right of the dialog.

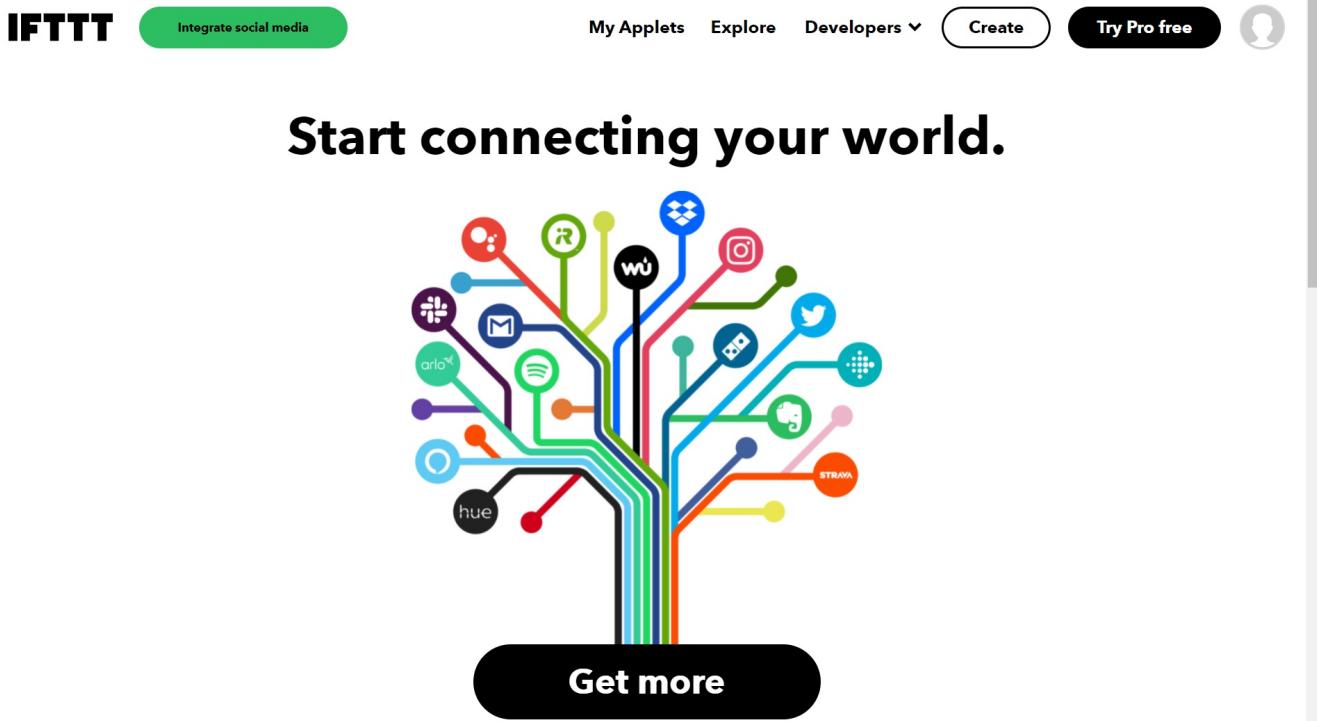
12. 完成

This screenshot shows the same Firebase Realtime Database interface after the addition. The database structure now includes the "word" node under "googlehome". The "Realtime Database" tab is still selected in the sidebar.

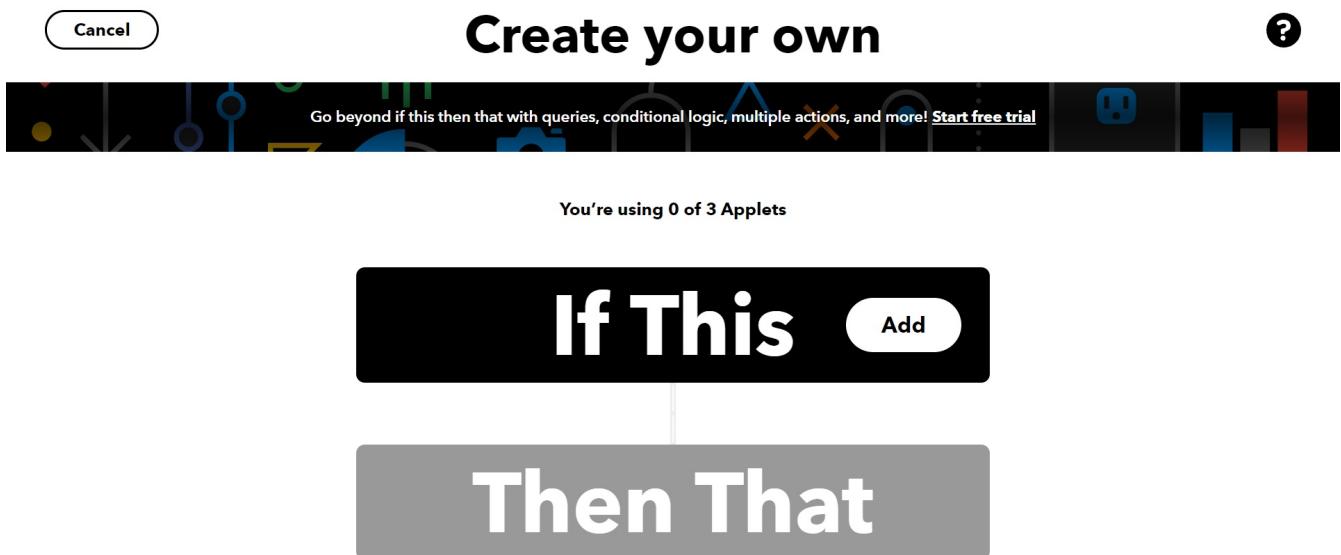
IFTTT

アプレット作成

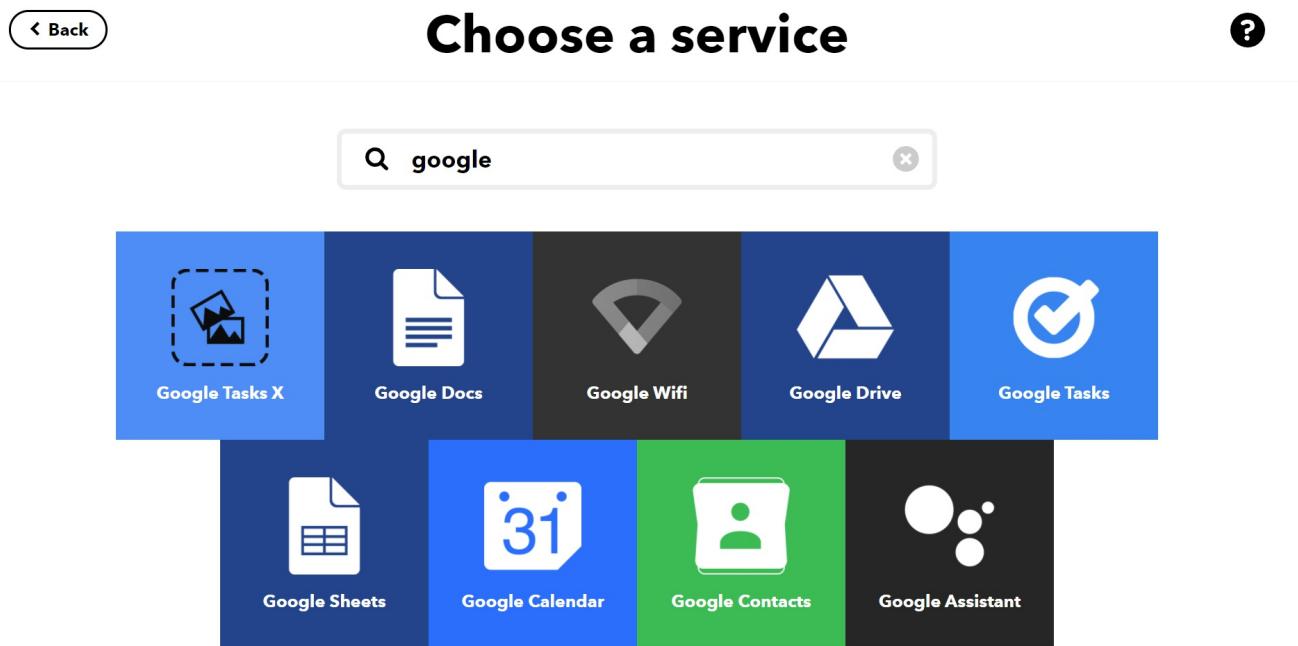
1. **Create** ボタンをクリック



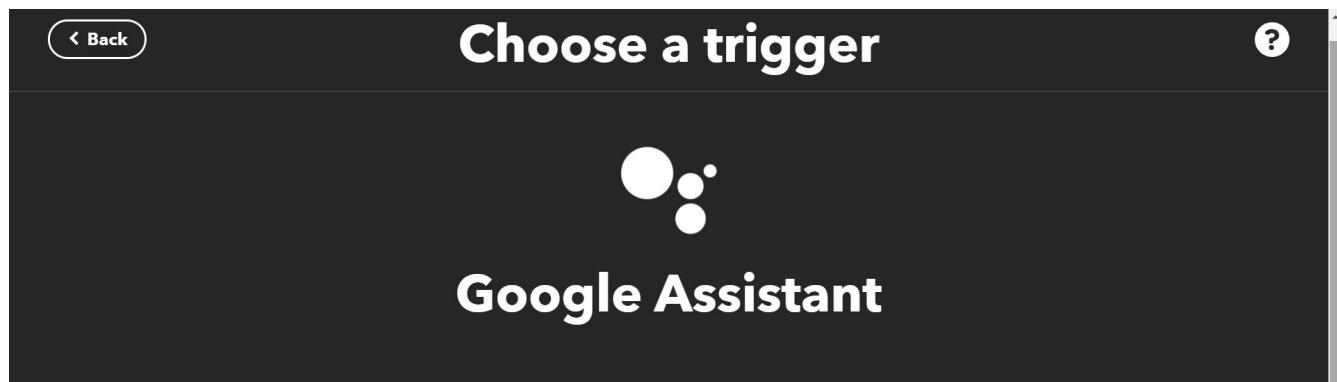
2. **If This** をクリック



3. 検索窓に **google** と入力し、結果から **Google Assistant** を選択



4. **Say a phrase with a text ingredient** を選択



Say a simple phrase	Say a phrase with a number	Say a phrase with a text ingredient	Say a phrase with both a number and a text ingredient
This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase you choose. For example, say "Ok Google, I'm running late" to text a family member that you're on your way home.	This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Set Nest thermostat to 68." **Use the # symbol to specify where you'll say the number ingredient	This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Post a tweet saying 'New high score.'" **Use the \$ symbol to specify where you'll say the text	This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Block time for 'exercise' at 6 PM." **Use the # symbol to specify where you'll say

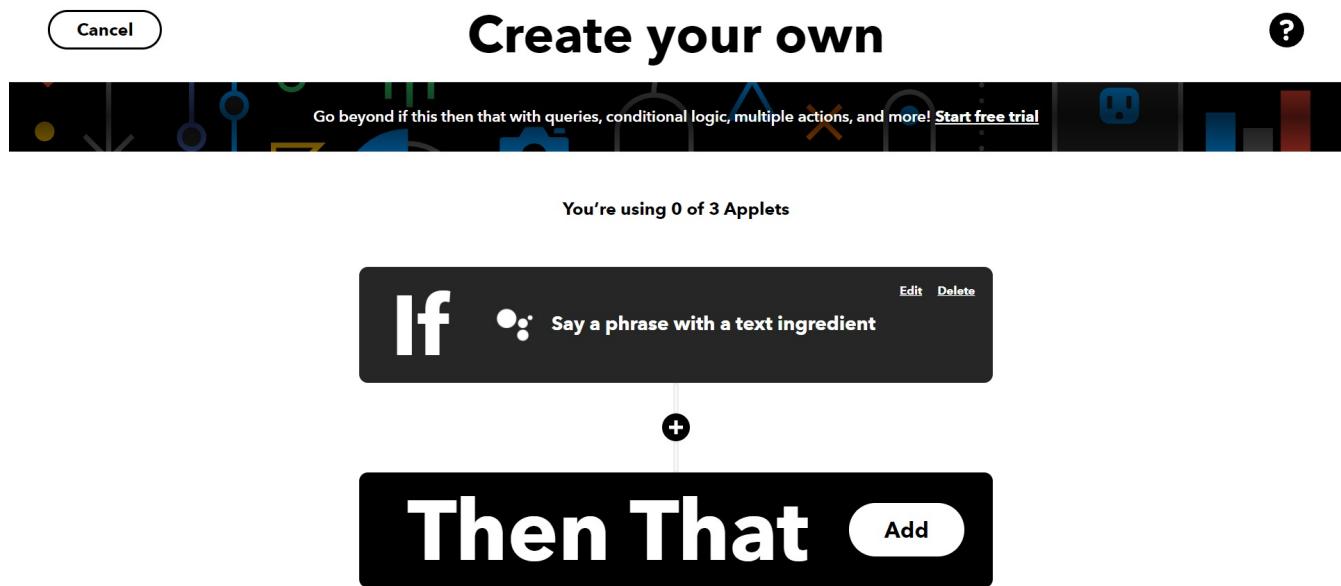
5. **What do you want to say?** に ドローン \$ と入力

The screenshot shows the 'Complete trigger fields' page. At the top, there's a back button and a help icon. Below that is a large 'Og.' logo. The main title is 'Say a phrase with a text ingredient'. A note below the title says: 'This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Post a tweet saying "New high score."** Use the \$ symbol to specify where you'll say the text ingredient'. There are two input fields: one for 'What do you want to say?' containing 'ドローン \$' and another for 'What's another way to say it? (optional)'. Both fields have placeholder text: 'Enter a \$ where you'll say the text ingredient'.

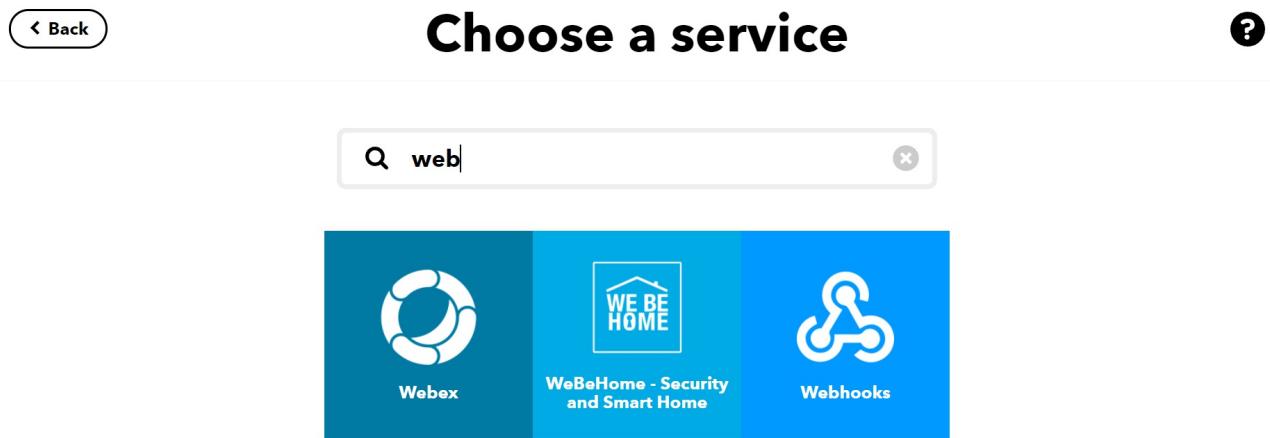
6. **What do you want the Assistant to say in response?** に ドローンを操作します と入力し、**Language** で Japanese を選択し、**Create trigger** ボタンをクリック

The screenshot shows the trigger configuration page with the response setup section. It includes fields for 'What's another way to say it? (optional)', 'And another way? (optional)', and 'What do you want the Assistant to say in response?' containing 'ドローンを操作します'. A note below the response field says: 'You can enter a \$ where you want to hear the text ingredient in the response'. There's a 'Language' dropdown set to 'Japanese' with a dropdown arrow. At the bottom is a large 'Create trigger' button.

7. **Then That** を選択



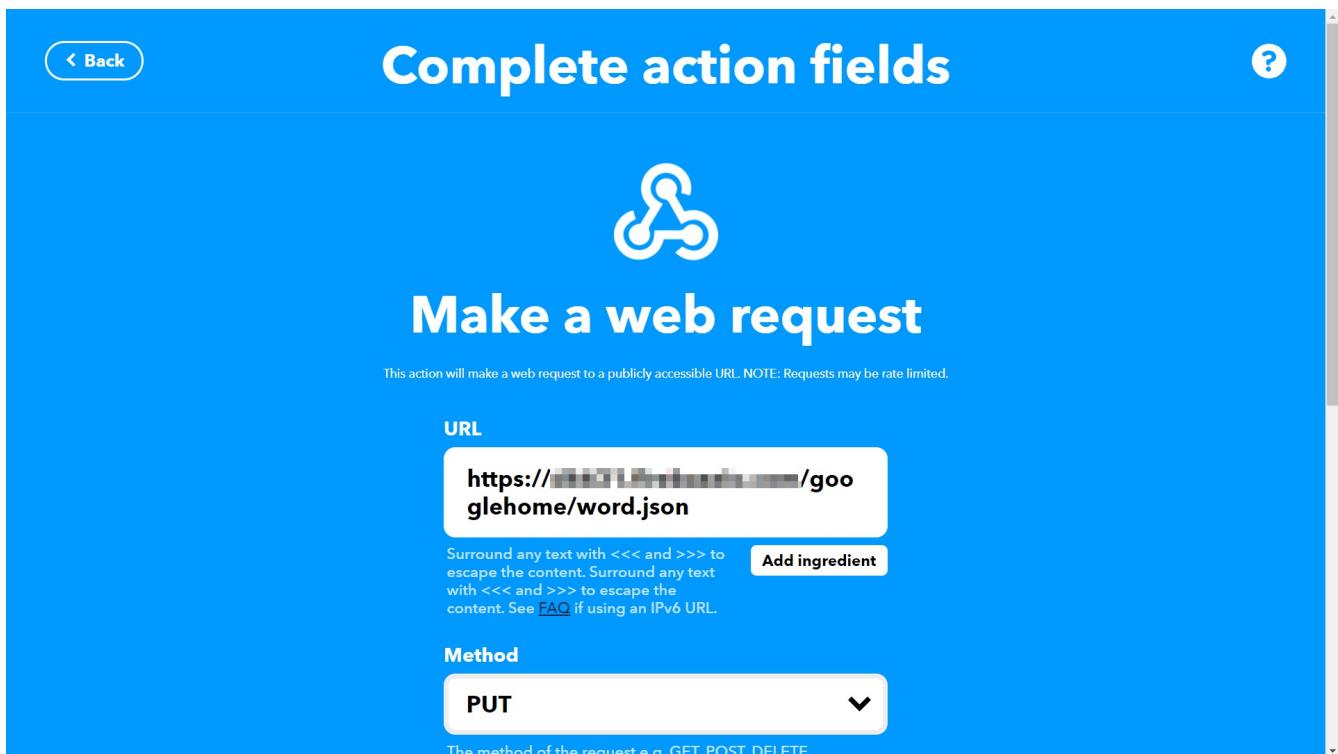
8. 検索窓に **web** と入力し、結果から **Webhooks** を選択



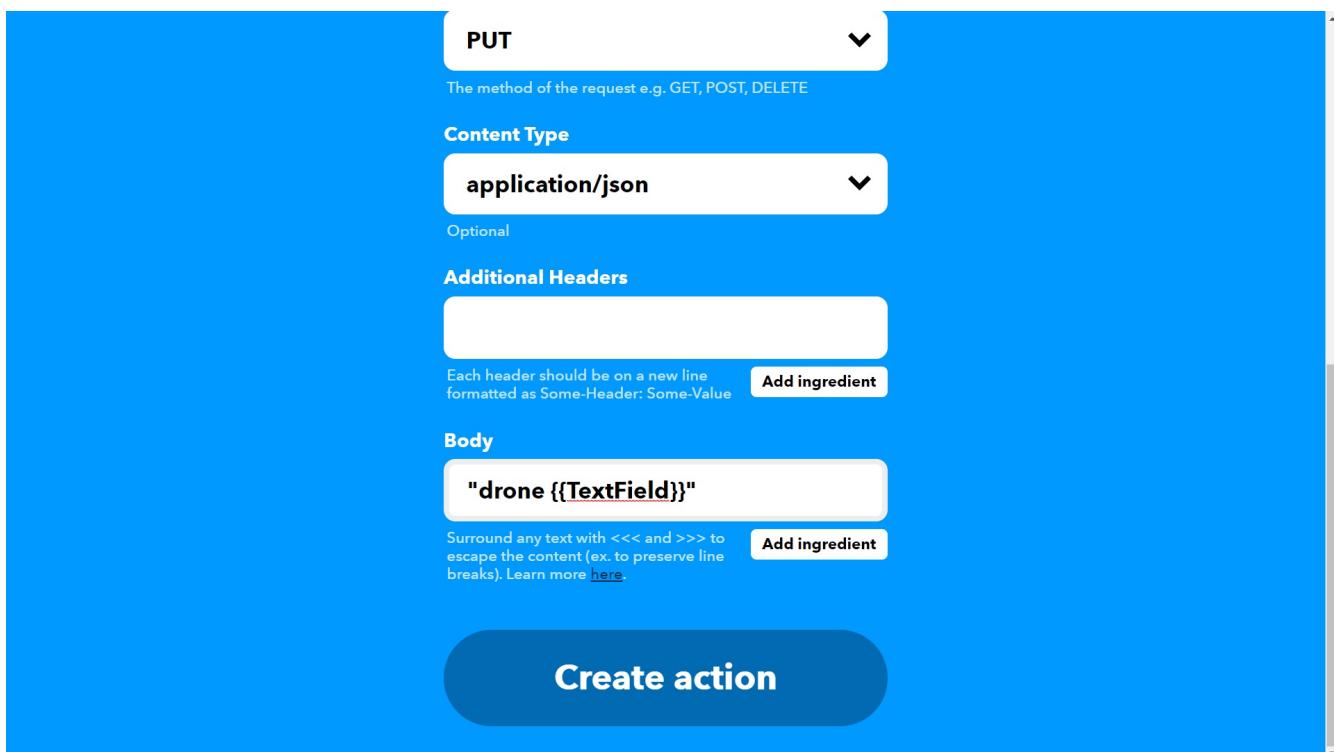
9. `Make a web request` を選択



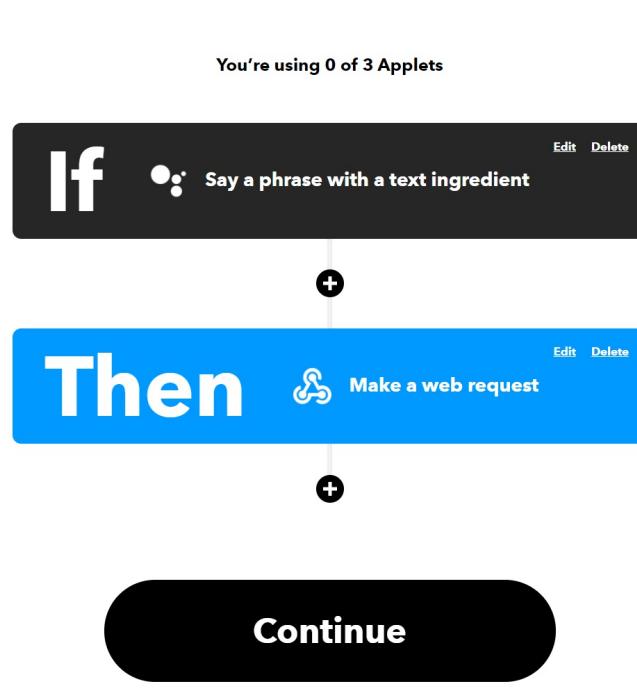
10. `URL` に
`https://{FirebaseのプロジェクトID}-default-rtdb.firebaseio.app/googlehome/word.json`
、`Method` に `PUT` と入力



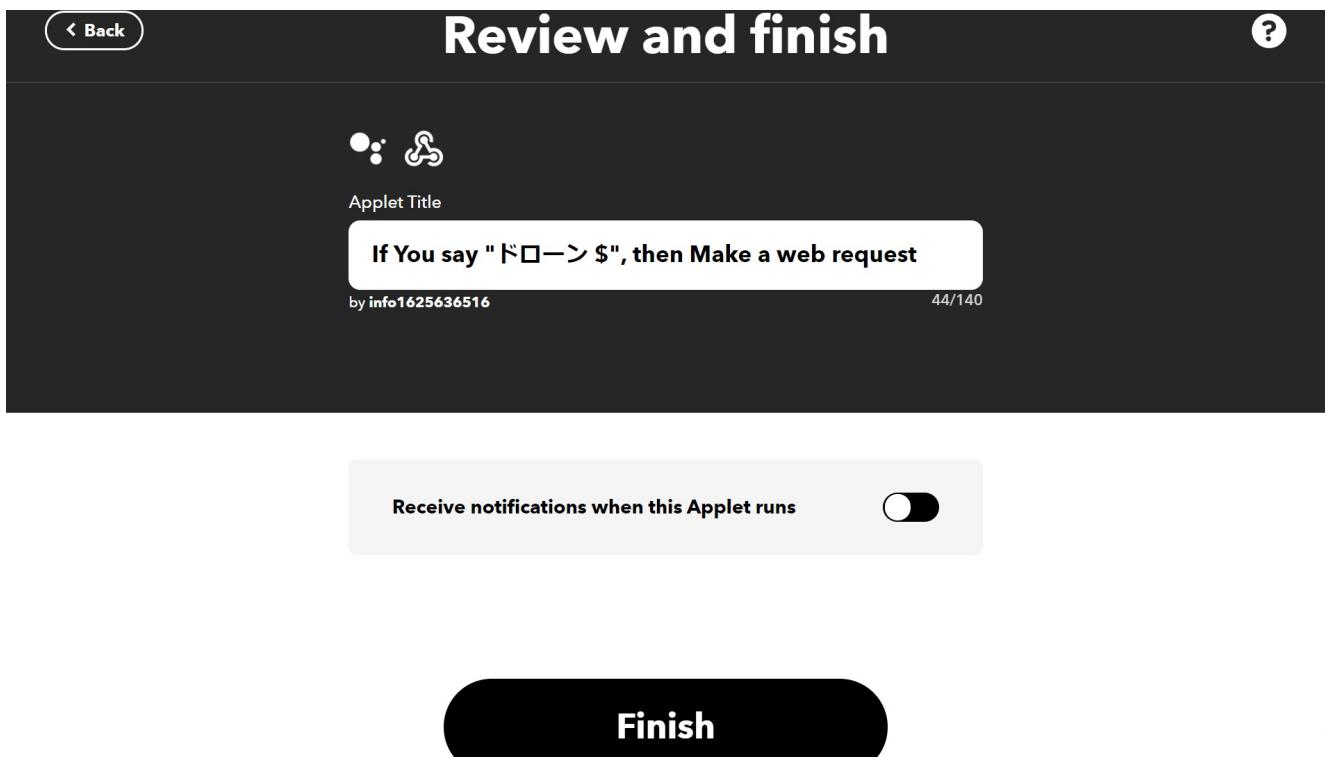
11. `Content Type` で `application/json` を選択、`Body` に `"drone {{TextField}}"` と入力し `Create action` ボタンをクリック



12. `Continue` ボタンをクリック

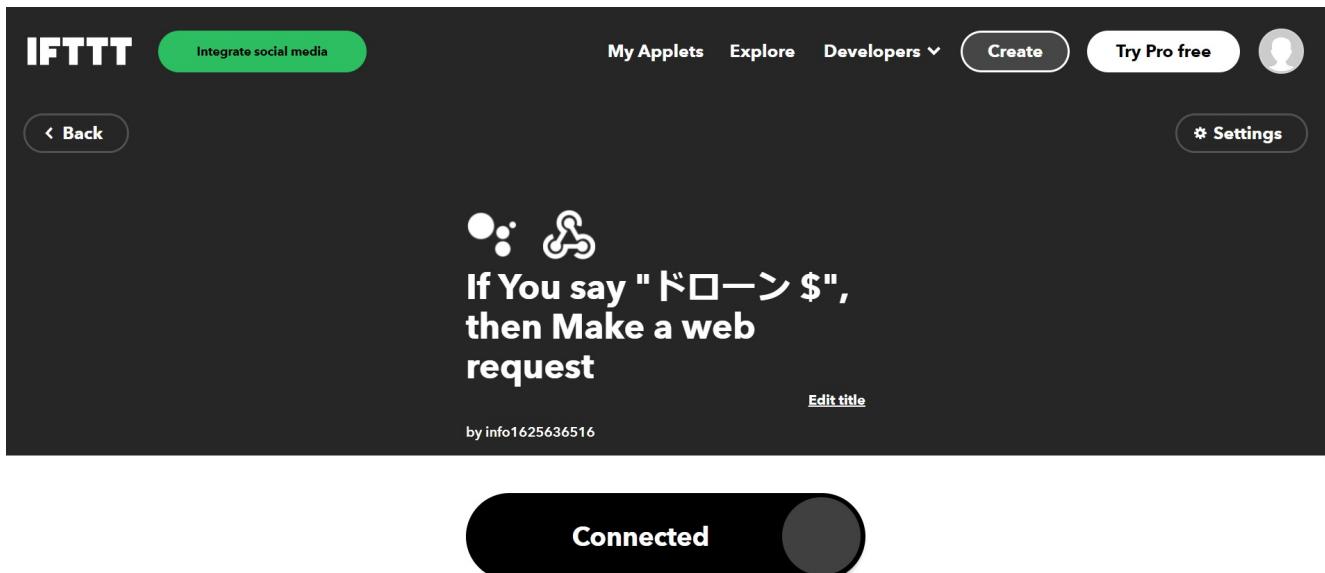


13. **Finish** ボタンをクリック



The screenshot shows the 'Review and finish' step in the IFTTT applet creation process. At the top, there's a back button and a help icon. The main title 'Review and finish' is displayed in large white font. Below it, the applet title 'If You say "ドローン \$", then Make a web request' is shown, along with the creator's name 'by info1625636516' and character count '44/140'. A toggle switch for 'Receive notifications when this Applet runs' is turned off. At the bottom, a large black button with the word 'Finish' in white is centered.

14. 完成



The screenshot shows the completed applet details page. At the top, there's a back button and a settings icon. The applet title 'If You say "ドローン \$", then Make a web request' is displayed again, with an 'Edit title' link. The creator's name 'by info1625636516' is shown below. A large black button with the word 'Connected' in white is centered. Below it, a toggle switch for 'Get notifications when this Applet is active' is turned off. At the bottom, there's a list of activity history: 'Connected Oct 08, 2021' and 'Never run', with a 'View activity' button.

音声制御のための Node.js プログラム

Firebase パッケージ

```
npm i firebase@8.10.0
```

index.js

`index.js` に以下を記述

```
"use strict"

// firebase
const firebase = require("firebase")
const firebaseProjectId = "ここに Firebase のプロジェクト ID を記入する"
const firebasePath = "/googlehome/word"
firebase.initializeApp({databaseURL: `https://${firebaseProjectId}-default.firebaseio.app`})

// udp
const dgram = require("dgram")
const sock = dgram.createSocket("udp4")

// Tello
const tello = {
  address: "192.168.10.1",
  port: 8889
}

// Telloへcommand送信
const sendTello = buf => {
  const command = new Buffer(buf)
  setTimeout(() => sock.send(command, 0, command.length, tello.port, tello.address), 500)
}

// database更新時
const db = firebase.database()
db.ref(firebasePath).on("value", snapshot => {
  // 値取得
  let value = snapshot.val()
  if (!value) return
  console.log(value)

  // option word index
  let index = 1
  // 助詞を除外
```

```

value = value.replace(/ ([がのをにへとでや] | より | から)/g, "")

// コマンド定義
const commandJson = {

    // drone
    "drone": () => {
        // 距離情報取得
        const distanceWord = / \d+ (CM|cm|センチ)/
        const distanceMatch = value.match(distanceWord)
        const distance = distanceMatch ? distanceMatch[0].match(/\d+/)[0] : 0
        value = value.replace(distanceWord, "")

        // 角度情報取得
        const angleWord = / \d+ (°|度|ドル)/
        const angleMatch = value.match(angleWord)
        const angle = angleMatch ? angleMatch[0].match(/\d+/)[0] : 0
        value = value.replace(angleWord, "")

        // スペースで区切られた単語を修正
        value = value.replace(" 旋回", "旋回")
        value = value.replace(" 旋回", "旋回")
        value = value.replace(" フリップ", "フリップ")
        value = value.replace(" グリップ", "フリップ")

        sendTello("command")
        const option = {
            "離陸": "takeoff",
            "着陸": "land",
            "上昇": `up ${distance}`,
            "上": `up ${distance}`,
            "下降": `down ${distance}`,
            "下": `down ${distance}`,
            "右": `right ${distance}`,
            "左": `left ${distance}`,
            "前": `forward ${distance}`,
            "前進": `forward ${distance}`,
            "バック": `back ${distance}`,
            "後ろ": `back ${distance}`,
            "後退": `back ${distance}`,
            "右旋回": `cw ${angle}`,
            "左旋回": `ccw ${angle}`,
            "右フリップ": "flip r",
            "左フリップ": "flip l",
            "前フリップ": "flip f",
            "バックフリップ": "flip b",
        }[value.split(" ")[index]]
        return typeof option === "string" ? () => sendTello(option) :
false
    },
    }[value.split(" ")[0]]

    // コマンド取得
    if (!commandJson) return
    const command = commandJson()

    // コマンド実行
    if (!command) return
}

```

```
command()

// firebase clear
db.ref(firebasePath).set("")  
})
```

実行

1. Google Home Mini を Wi-Fi で繋ぐ
2. Tello 起動
3. RaspberryPi を有線 LAN でネットに繋ぐ(Wi-Fi は Tello との連携に使うため)
4. RaspberryPi と Tello を Wi-Fi で繋ぐ
5. `node index.js` コマンドを打ち込む
6. 音声で制御

命令サンプル

- OK グーグル、ドローン離陸
- OK グーグル、ドローン前に80センチ
- OK グーグル、ドローン後ろに80センチ
- OK グーグル、ドローン90度右旋回
- OK グーグル、ドローン90度左旋回
- OK グーグル、ドローン着陸