

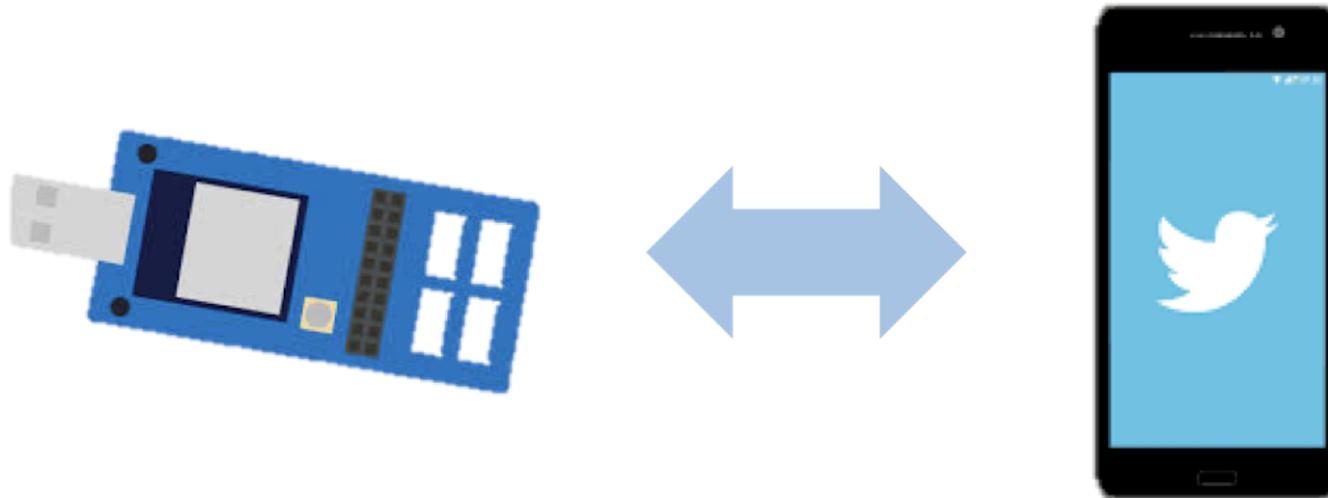
# IoTハッカソン～business～事前勉強会 【NefryでIoTハンズオン】Day 1

# 目次

1. ノンプログラミングでTwitter投稿を行ってみよう
2. LEDを光らせるプログラムを書いてみよう
3. 光センサの情報を取得して記録してみよう

# 1.NefryとIFTTTを用いて ノンプログラミングでTwitter投稿を行おう

<https://dotstud.io/docs/nefrybt-ifttt/>



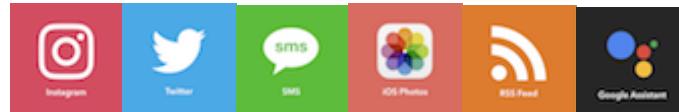
# Nefry BTとは？

IFTTT等のインターネットのサービスとハードウェアを簡単に接続できるIoTデバイス。Wi-Fi / BLE通信モジュール「ESP-WROOM-32」を搭載



## IFTTT (if this, then that) とは？

LINE, Twitter, Slack など、様々なWebサービス同士を簡単に連携(レシピと呼ぶ)できるサービス

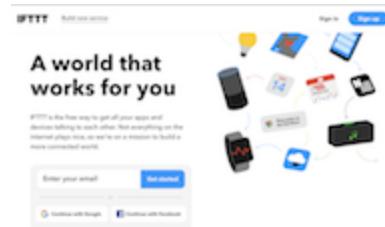


## NefryとIFTTT連携の手順

- (1) IFTTTにレシピを作成 → (2) Secret Key を取得  
→ (3) Nefry BT の設定ページでSecret Key を登録

# (0) IFTTT,twitter のアカウント作成

- IFTTTのアカウントを作成 <https://ifttt.com>



- (アカウントを取得していない人のみ)  
twitter のアカウントを作成 <https://twitter.com/>



# (1) IFTTTにレシピを作成

レシピ：

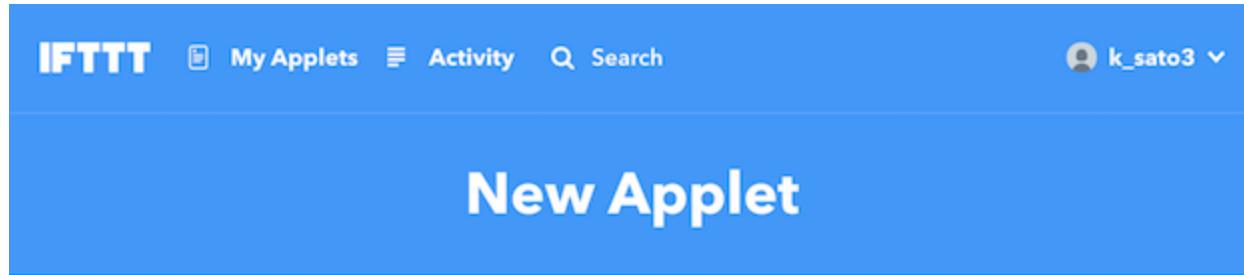
if **this** then **that**

○○ が起きたら △△ する

Nefryのボタンを押し たら IFTTTのWebhooksにアクセス する

# (1) IFTTTにレシピを作成

[Thisの設定] 「+this」 をクリック



if **+this** then **that**

Want to build your own service? Build on the platform [↗](#)

[About](#)   [Blog](#)   [Help](#)   [Jobs](#)   [Terms](#)   [Privacy](#)   [Trust](#)

Build your own service and Applets

 **IFTTT** Platform

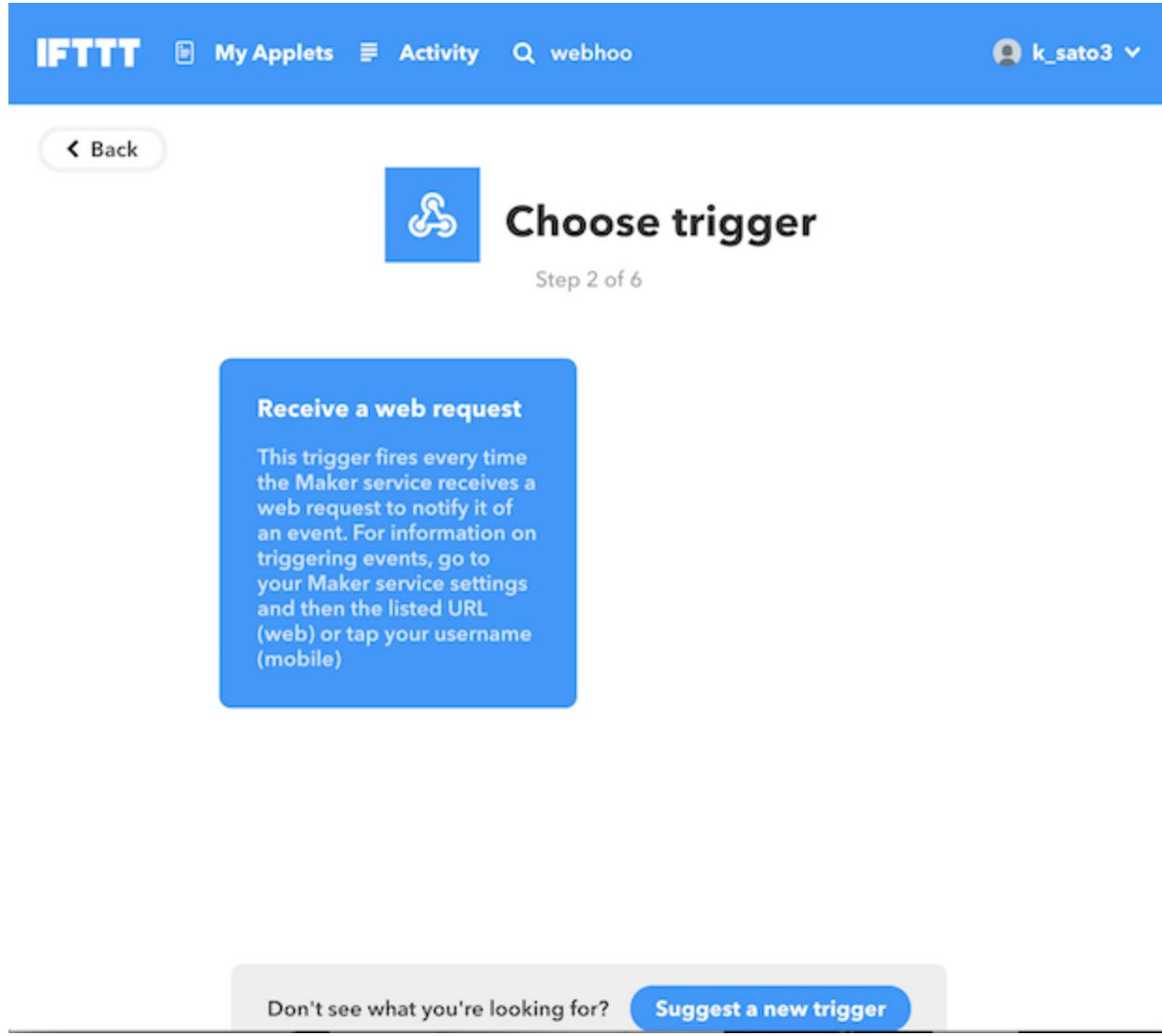
# (1) IFTTTにレシピを作成

[Thisの設定] webhooksを検索し、 webhooksのアイコンをクリック

The screenshot shows the IFTTT platform's "Choose a service" step. At the top, there is a blue header bar with the IFTTT logo, "My Applets", "Activity", "Search", and a user profile icon. Below the header, a "Back" button is visible. The main title "Choose a service" is centered above a progress indicator "Step 1 of 6". A search bar contains the query "webhooks", which is highlighted with a blue border. Below the search bar, a blue square icon for the "Webhooks" service is displayed, featuring a white three-headed arrow symbol and the word "Webhooks" underneath. At the bottom of the page, there is a horizontal navigation bar with links: "About", "Blog", "Help", "Jobs", "Terms", "Privacy", and "Trust". Below this, a tagline reads "Build your own service and Applets". At the very bottom, there is a button labeled "IFTTT Platform".

# (1) IFTTTにレシピを作成

[Thisの設定] 「Receive a web request」 をクリック



The screenshot shows the IFTTT web interface. At the top, there's a blue header bar with the IFTTT logo, 'My Applets', 'Activity', and a search bar containing 'webhook'. On the right, there's a user profile icon and the text 'k\_sato3'. Below the header, a back button labeled 'Back' is visible. The main title 'Choose trigger' is centered above 'Step 2 of 6'. A large blue button labeled 'Receive a web request' is prominently displayed. To its right, a detailed description of the trigger is provided: 'This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)'. At the bottom of the screen, there's a grey footer bar with the text 'Don't see what you're looking for?' and a blue button labeled 'Suggest a new trigger'.

# (1) IFTTTにレシピを作成

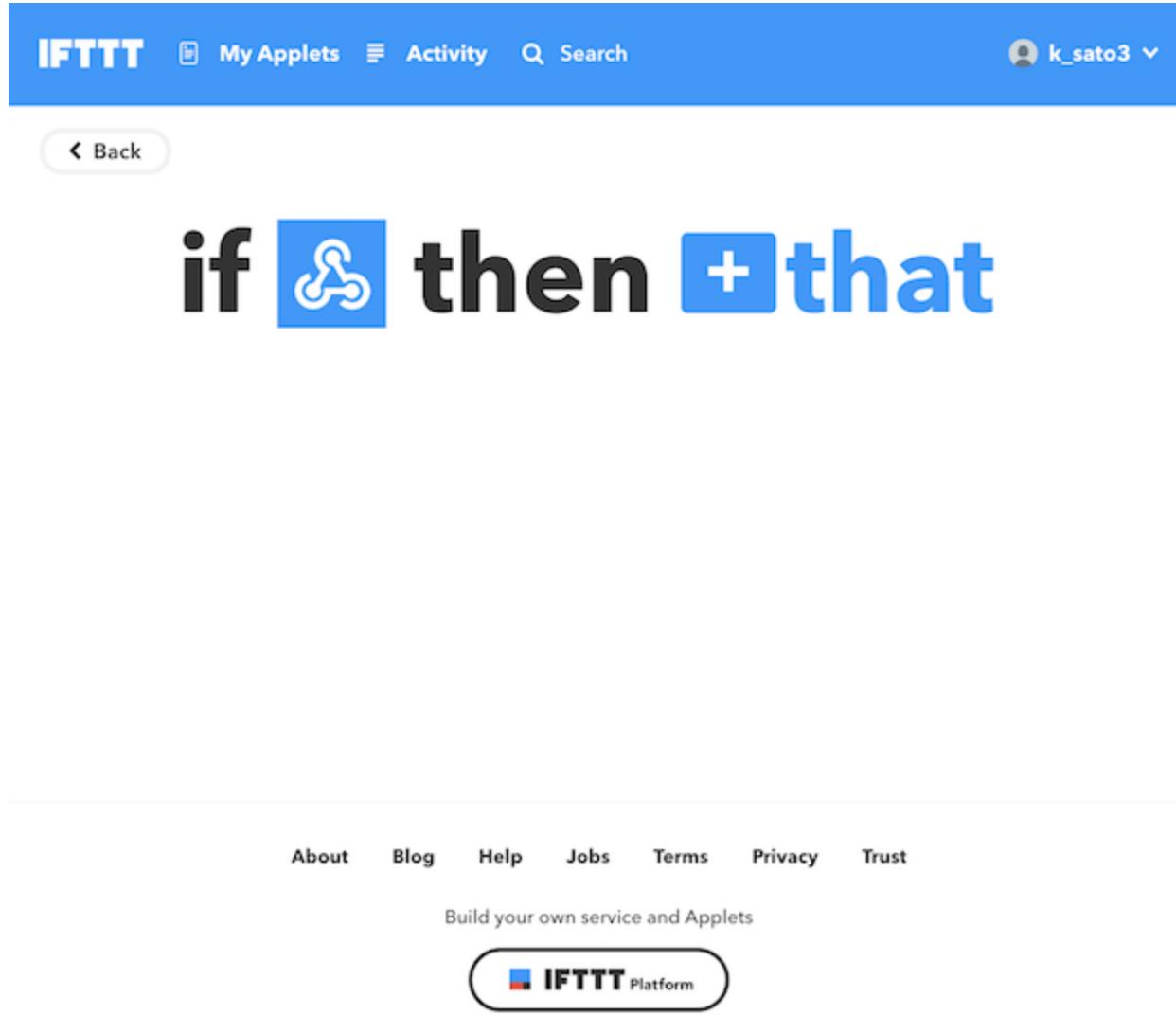
[Thisの設定]

Event Nameに Nefry\_IFTTT と記入し、「Create trigger」をクリック

The screenshot shows the IFTTT web interface. At the top, there's a blue header bar with the IFTTT logo, 'My Applets', 'Activity', and a search bar. On the right, it shows a user profile icon and the name 'k\_sato3'. Below the header, a back button is visible. The main content area has a blue background and features a large blue button with a white 'a' icon and the text 'Complete trigger fields'. Underneath, it says 'Step 2 of 6'. A central box is titled 'Receive a web request' with a sub-instruction: 'This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)'. It has a text input field labeled 'Event Name' containing 'Nefry\_IFTTT'. Below the input field is a note: 'The name of the event, like "button\_pressed" or "front\_door\_opened"' followed by a large blue 'Create trigger' button.

# (1) IFTTTにレシピを作成

[Thatの設定] 「+that」 をクリック



# (1) IFTTTにレシピを作成

[Thatの設定]twitter を検索し、twitterのアイコンをクリック

The screenshot shows the IFTTT platform interface at Step 3 of 6. The top navigation bar includes links for 'My Applets', 'Activity', and a search bar with the query 'webhoo'. A user profile icon for 'k\_sato3' is also present. Below the navigation, a 'Back' button is visible. The main title 'Choose action service' is centered above a search bar containing the text 'twitter'. A large blue button with the Twitter logo and the word 'Twitter' is prominently displayed below the search bar.

IFTTT

My Applets Activity webhoo

k\_sato3

Back

Choose action service

Step 3 of 6

Q twitter

Twitter

About Blog Help Jobs Terms Privacy Trust

Build your own service and Applets

IFTTT Platform

# (1) IFTTTにレシピを作成

[Thatの設定] 「Post a tweet」 をクリック

The screenshot shows the IFTTT web interface at Step 4 of 6. The top navigation bar includes 'IFTTT', 'My Applets', 'Activity', 'Q webhoo', and a user profile 'k\_sato3'. Below the navigation is a back button labeled 'Back'. The main title 'Choose action' is displayed next to a Twitter icon. The step number 'Step 4 of 6' is shown below the title.

**Post a tweet**  
This Action will post a new tweet to your Twitter account. NOTE: Please adhere to Twitter's Rules and Terms of Service.

**Post a tweet with image**  
This Action will post a new tweet to your Twitter account with a linked pic.twitter.com image. NOTE: Please adhere to Twitter's Rules and Terms of Service.

**Send a direct message to yourself**  
This Action will send a direct message to your Twitter account. NOTE: Please adhere to Twitter's Rules and Terms of Service.

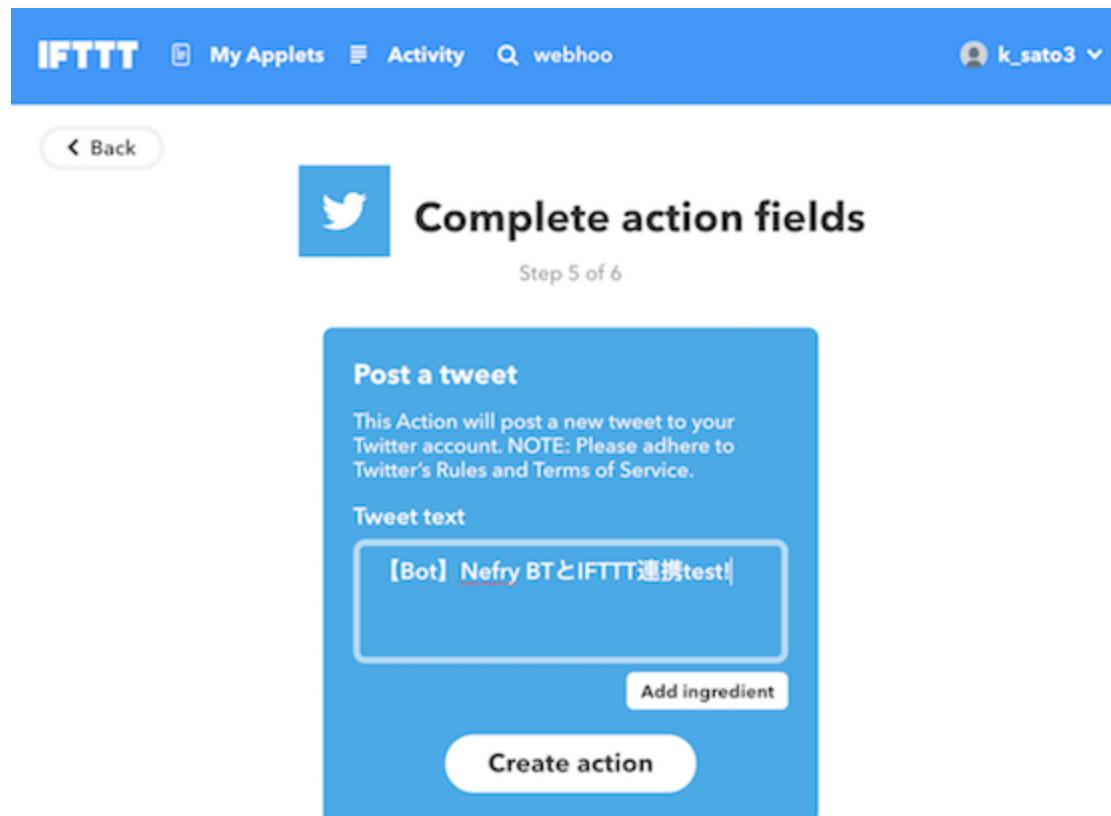
**Update profile picture**  
This Action will update your profile picture from the image URL you specify and optionally tweet about it. NOTE: Please adhere to Twitter's Rules and Terms of Service.

# (1) IFTTTにレシピを作成

[Thatの設定] Tweet text 欄にツイートしたい内容を入力し,

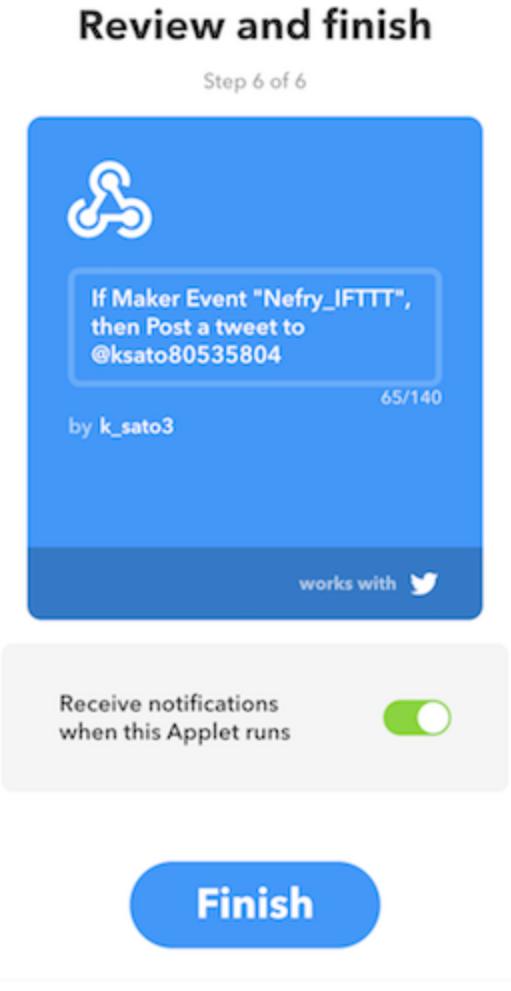
(例: 【Bot】 Nefry BTとIFTTT連携test )

「Create action」 をクリック



# (1) IFTTTにレシピを作成

[Thatの設定] 「Finish」をクリック



これで、「**Nefryのボタンを押したらIFTTTのWebhooksにアクセスする**」レシピの作成完了です。

## (2) webhooksのSecret Key を取得

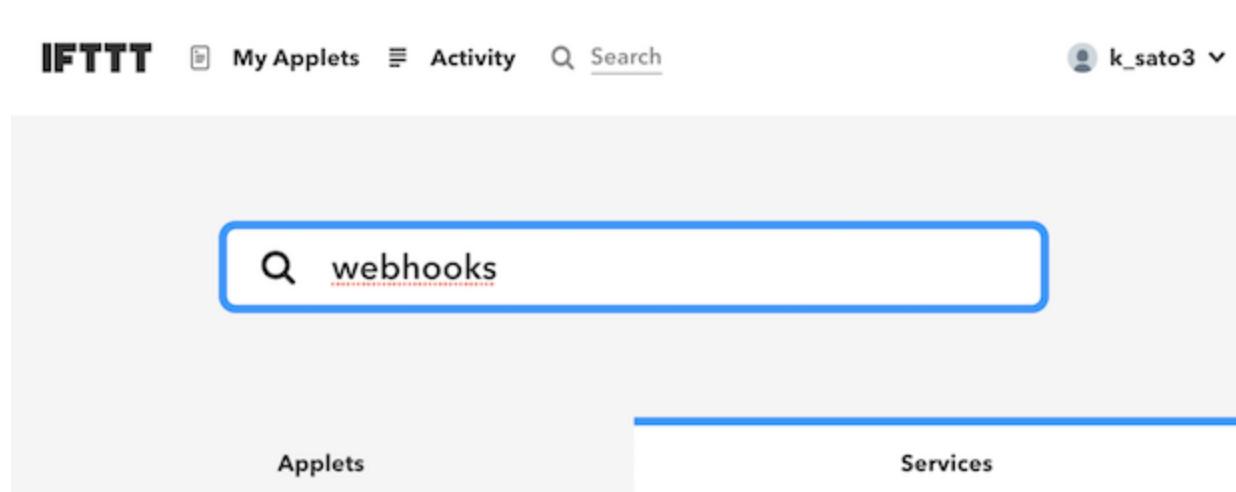
## (2) Secret Key を取得

webhooks で検索

The screenshot shows the IFTTT web interface. At the top, there's a navigation bar with 'IFTTT' logo, 'My Applets', 'Activity', a search bar containing 'webhooks', and a user profile 'k\_sato3'. Below the navigation, the path 'My Applets > Webhooks' is visible. The main content area displays a blue applet card for an 'If Maker Event "Nefry\_IFTTT", then Post a tweet to @ksato80535804' applet. The card features a gear icon, the IFTTT logo, and the applet's description. It includes a large green 'On' toggle switch. Below the switch, it says 'Created on Jul 30 2018' and 'Never run'. A note states 'This Applet usually runs within an hour' with a 'Check now' button. The entire interface has a clean, modern design with white space.

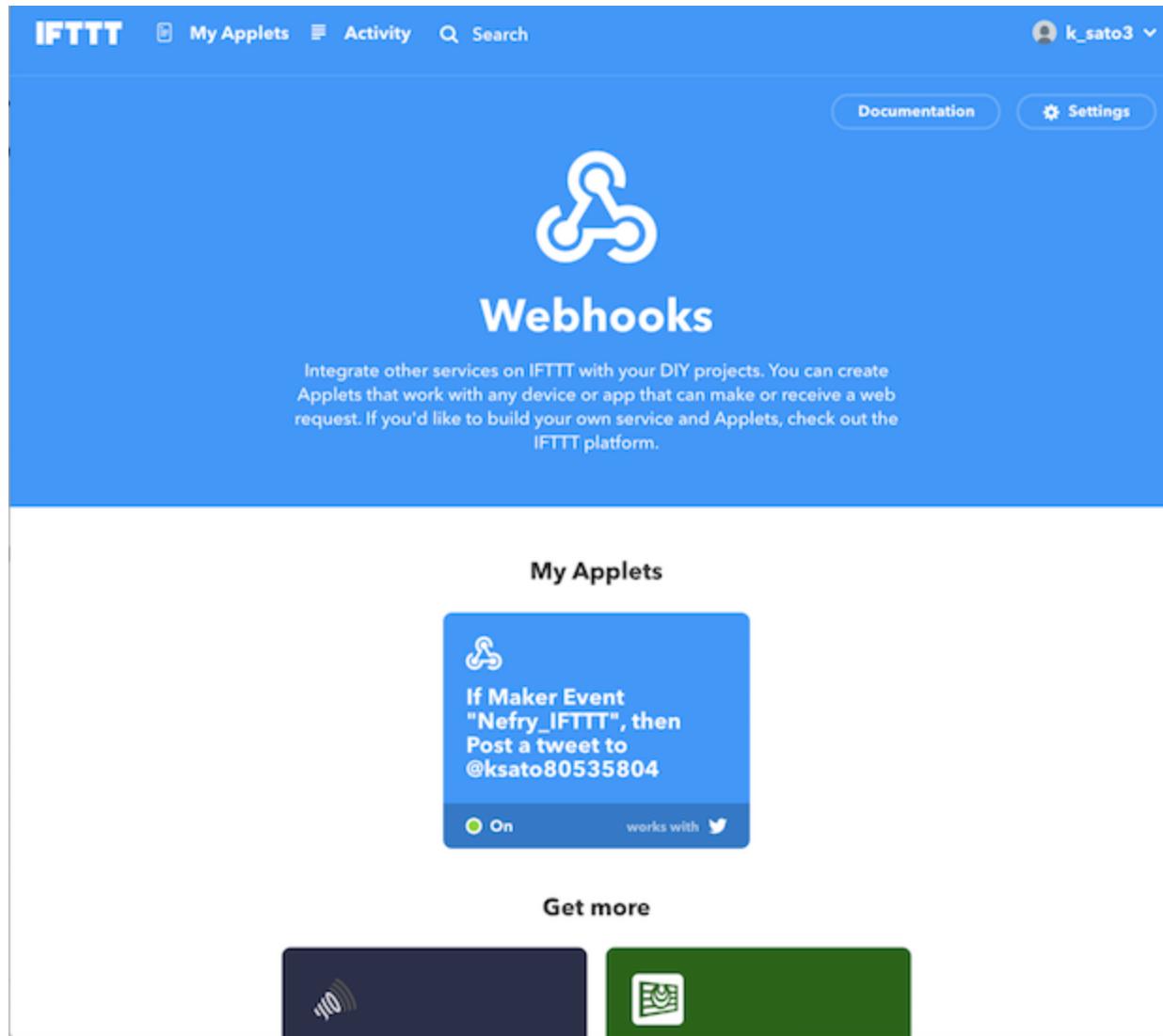
## (2) Secret Key を取得

「Services」をクリックし、 webhooksのアイコンをクリック



## (2) Secret Key を取得

右上の「Settings」をクリック



## (2) Secret Key を取得

URLの末尾の自身のSecret Keyをコピー

The screenshot shows the IFTTT Webhooks settings page. At the top, there's a navigation bar with 'My Applets', 'Activity', and 'Search' links, and a user profile icon 'k\_sato3'. Below the navigation, the URL 'My Applets > Webhooks' is shown. A blue icon representing Webhooks is displayed, followed by the text 'Webhooks settings' and a link 'View activity log'. The 'Account Info' section contains the following details:

- Connected as: [REDACTED]
- URL: <https://maker.ifttt.com/use/> [REDACTED]
- Status: active

A button labeled 'Edit connection' is present. Further down the page, a red link 'Disconnect Webhooks' is visible. At the bottom, there are links for 'About', 'Blog', 'Help', 'Jobs', 'Terms', 'Privacy', and 'Trust', along with a tagline 'Build your own service and Applets' and the 'IFTTT Platform' logo.

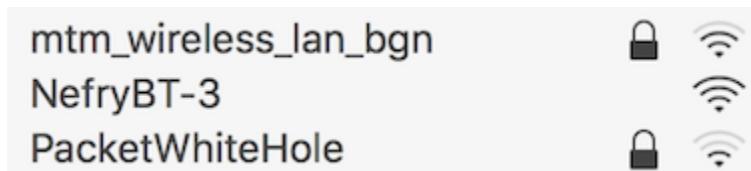
次に、コピーしたWebhooksの Secret Key を Nefry に登録します。

**(3)Nefry BTの設定ページでSecret Keyを登録**

# (3) Nefry BT の設定ページでSecret Key を登録

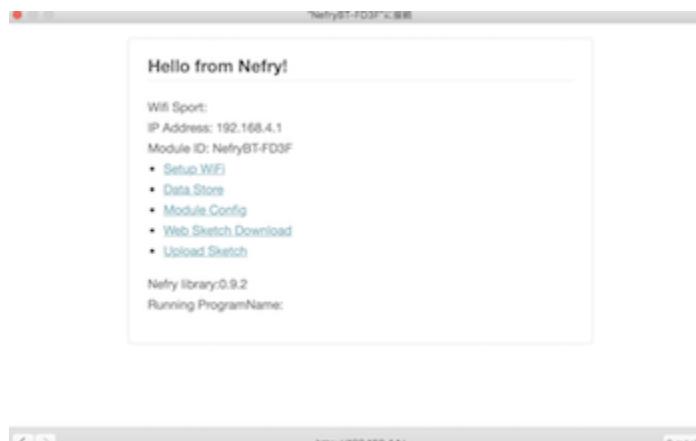
## wifi設定

- Nefry BTを自分のPCに接続.
- "NefryBT-(数字)" というWi-Fiに接続.



Wi-Fiに接続すると自動で設定ページのウィンドウが立ち上がる。  
(立ち上がらない場合はブラウザでhttp://192.168.4.1にアクセス)

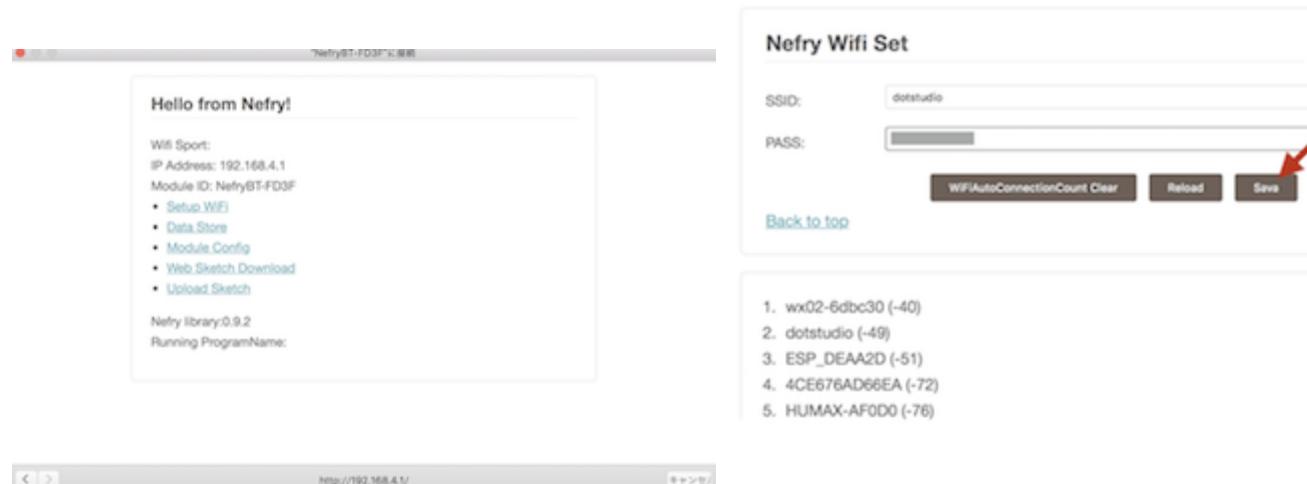
このページからNefry BTに関する様々な設定が可能.



### (3) Nefry BT の設定ページでSecret Key を登録

#### wifi設定

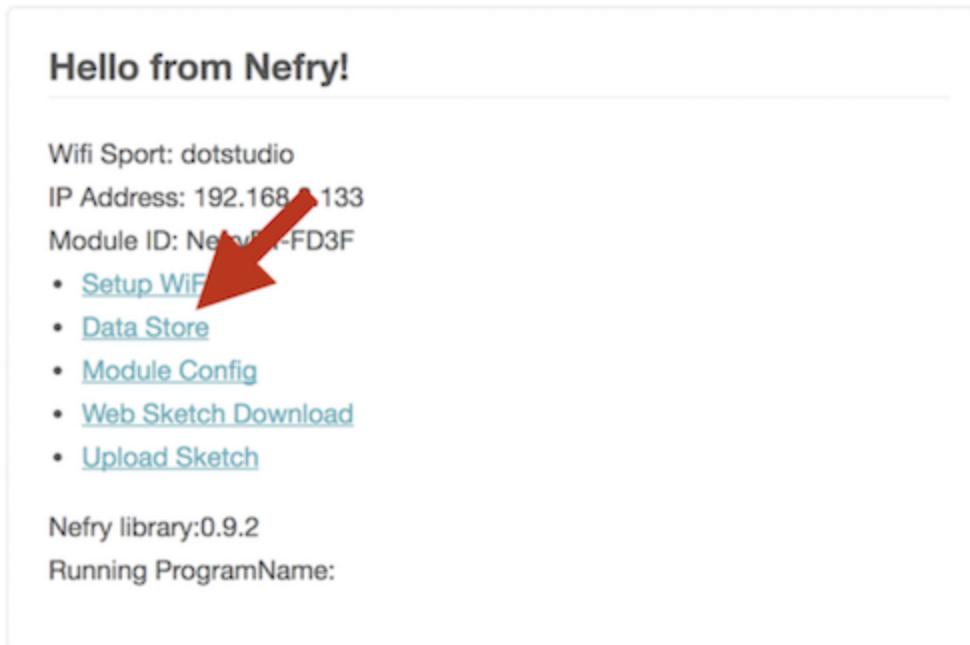
「Setup WiFi」をクリックし、利用するWi-FiのSSIDとパスワードを入力し、「Save」で保存。



再度、Nefry BTのWi-Fiにつなぎ、トップ画面を開くと、利用するWi-FiのSSIDとNefry BTのIPアドレスが表示される。

## (3) Nefry BT の設定ページでSecret Key を登録

トップ画面から「Data Store」に移動。



## (3) Nefry BT の設定ページでSecret Key を登録

先ほど取得したWebhooksの「Secret Key」と「Event Name」をそれぞれ入力し、保存。

Nefry DataStore Setup

このページはプログラム内から読み書きした値を表示、編集することができます。

わざわざプログラムを書き換えずに値を変更できるためWebサービスでアクセスキーが必要になる場合やモードを切り替える時など環境変数として扱うことができます。

- setStoreValue or setStoreStr : 値の保存
- getStoreValue or getStoreStr : 値の取得
- setStoreTitle : 内容の表示

それぞれの関数の使い方はNefry公式サイトをご覧になるか、サンプルプログラムを参考にしてください。

SecretKey	<input type="text" value="XXXXXXXXXX"/>
Event	<input type="text" value="NefryIFTTT"/>
LINE Auth	<input type="text"/>
LINEMessage	<input type="text"/>

[Back to top](#)

### (3) Nefry BT の設定ページでSecret Key を登録

Nefry BTのボタンを押して試してみましょう！



Nefryのボタンを押すとTweetすることができました！  
(IFTTTのサーバ状況により送信に時間がかかる場合があります。)

## 2.LEDを光らせるプログラムを書いてみよう

<https://dotstud.io/docs/nefrybt-led/>

## 2.LEDを光らせるプログラムを書いてみよう

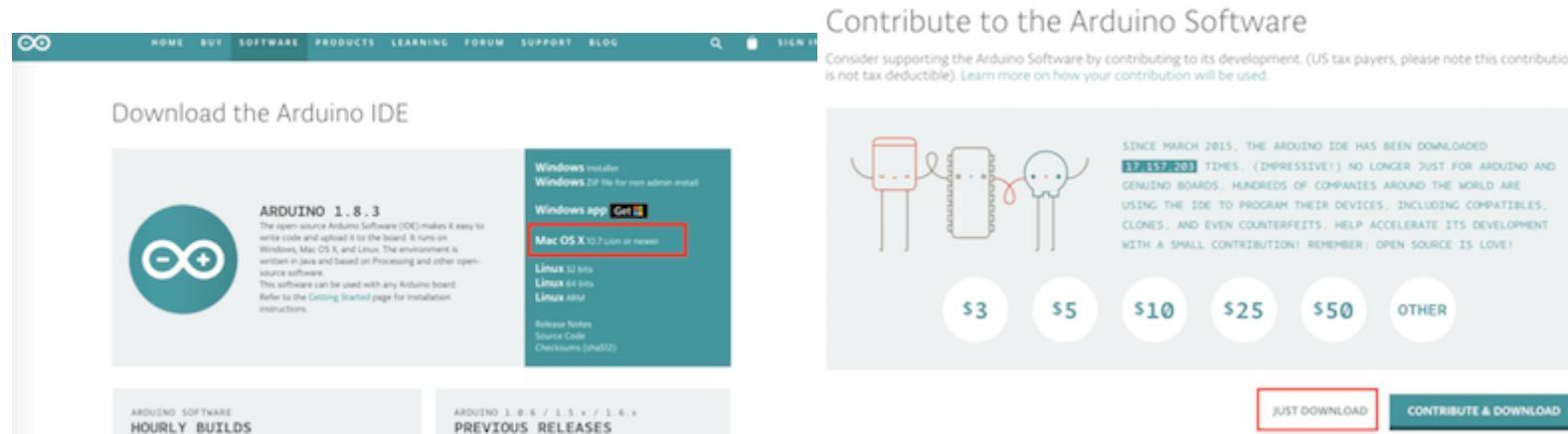
### (0)準備

- Arduino インストール <https://dotstud.io/docs/arduinoide-setup/>
- Arduino セットアップ <https://dotstud.io/docs/nefrybt-arduino-ide-setup/>

## 2.LEDを光らせるプログラムを書いてみよう

### (0)準備 Arduino インストール

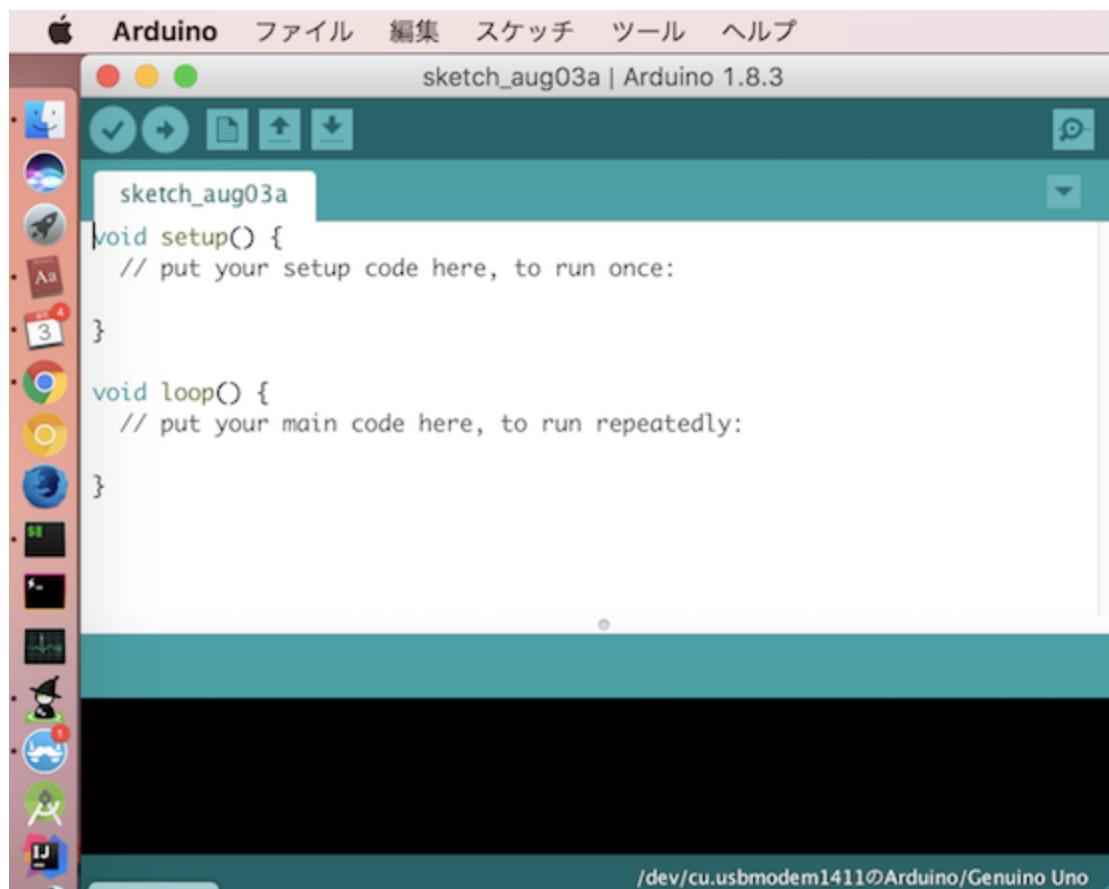
Arduinoのサイトから自身の環境に合わせたダウンロードリンクを選択。次のページでドネーション（寄付金）を求められますが、特に気にせず「JUST DOWNLOAD」を選択し、インストール。



## 2.LEDを光らせるプログラムを書いてみよう

### (0)準備 Arduino インストール

Arduinoファイルを起動し、下記のような画面が表示されれば起動完了。

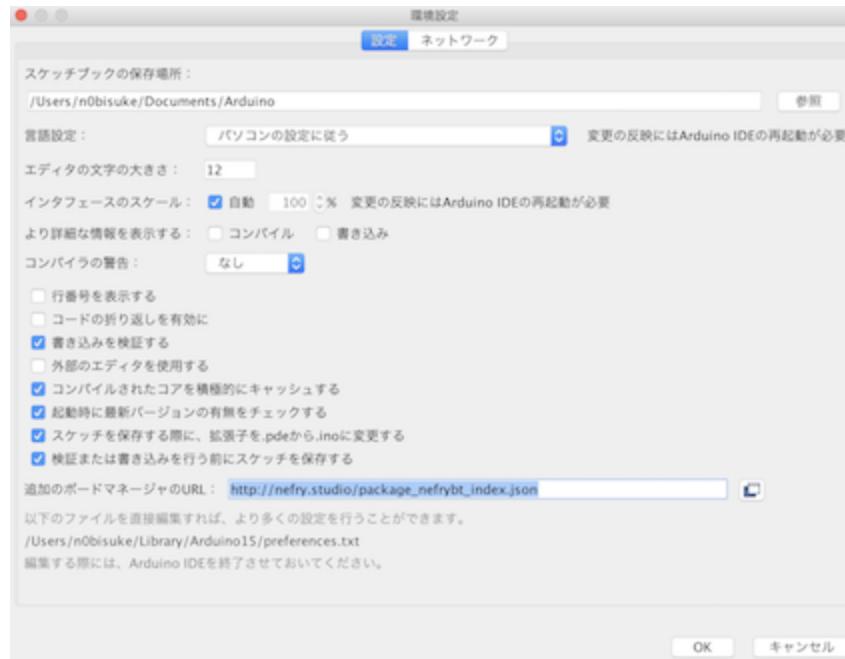


## 2.LEDを光らせるプログラムを書いてみよう

### (0)準備 Arduino セットアップ

メニューのArduino > Preferencesを選択。追加のボードマネージャのURLのフォームに以下のURLを追加し、OKを選択。

`http://nefry.studio/package_nefrybt_index.json`



## 2.LEDを光らせるプログラムを書いてみよう

### (0)準備 Arduino セットアップ

メニューのツール > ボード > ボードマネージャを選択。



## 2.LEDを光らせるプログラムを書いてみよう

### (0)準備 Arduino セットアップ

フォームにNefryと入力して検索する。

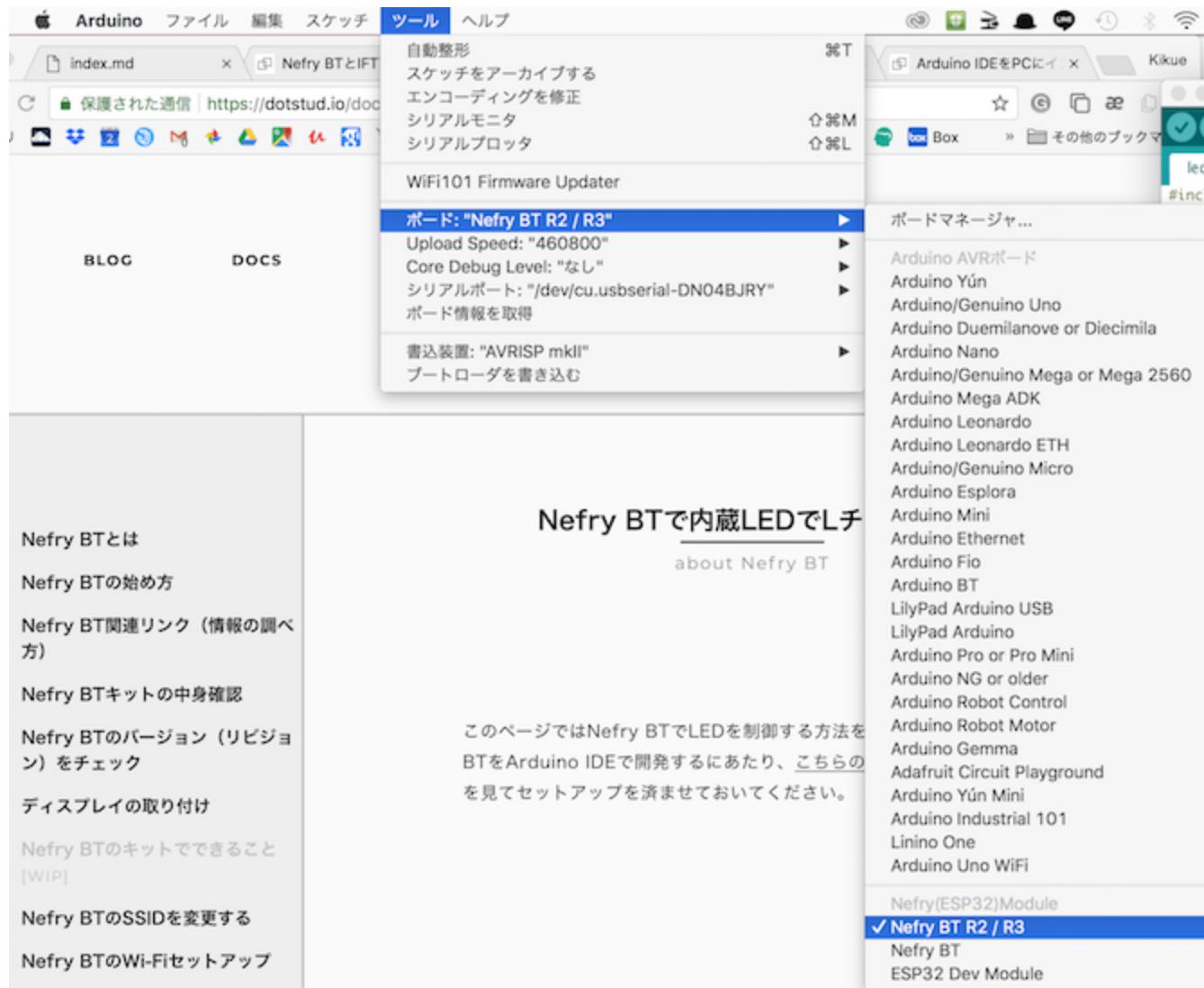


バージョン1.4.0を選択し、インストールを押す  
(バージョンは1.2.2以上のものを選択)

# 2.LEDを光らせるプログラムを書いてみよう

## (1)ボードの選択

メニューのツール > ボード > Nefry BT R2/R3を選択。



## 2.LEDを光らせるプログラムを書いてみよう

### (2)Nefry BTをPCのUSBポートに挿入

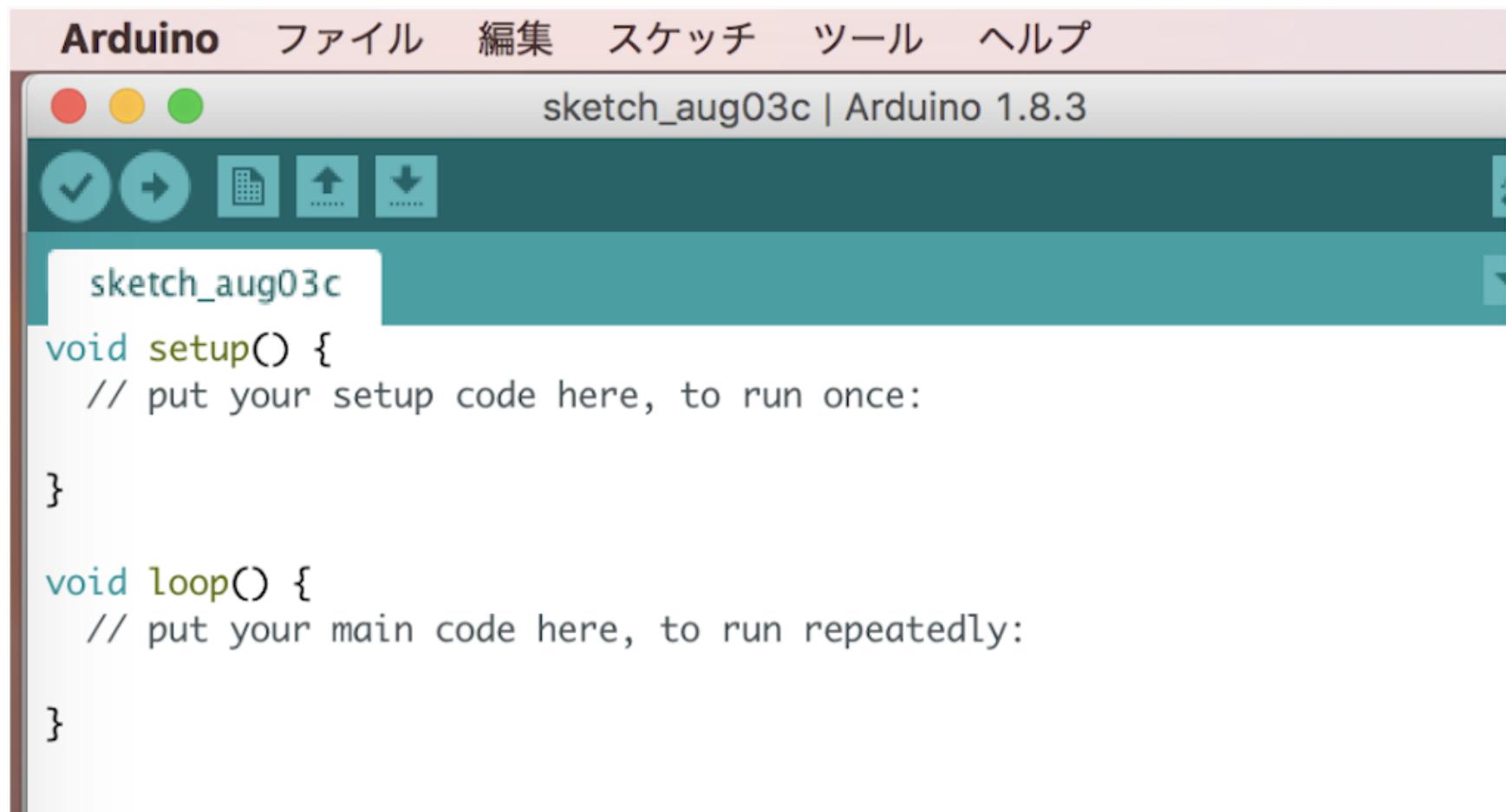
- Nefry BTをPCのUSBポートに挿入.
- メニューのツール > シリアルポート > /dev/cu.usbserial-xxxxxx を選択. (環境によっては /dev/tty.usbserial-xxxxxxなどの場合あり)

※Windowsだと表記が異なる可能性あり

## 2.LEDを光らせるプログラムを書いてみよう

### (3) プログラムの書き込み

Arduinoを選択し、メニューのファイル > 新規ファイルを選択。



The screenshot shows the Arduino IDE interface. The menu bar at the top includes 'Arduino', 'ファイル' (File), '編集' (Edit), 'スケッチ' (Sketch), 'ツール' (Tools), and 'ヘルプ' (Help). The title bar indicates the file is named 'sketch\_aug03c | Arduino 1.8.3'. Below the title bar is a toolbar with icons for saving, running, opening, uploading, and downloading. The main code editor window displays the following C++ code:

```
sketch_aug03c

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

初期状態で書かれているコードを削除し,以下のコードに差し替える

```
#include <Nefty.h>
//フルカラーLED ランダムにカラーが変わります.
#define SEED_PIN A0

void setup() {
    randomSeed(analogRead(SEED_PIN));
}

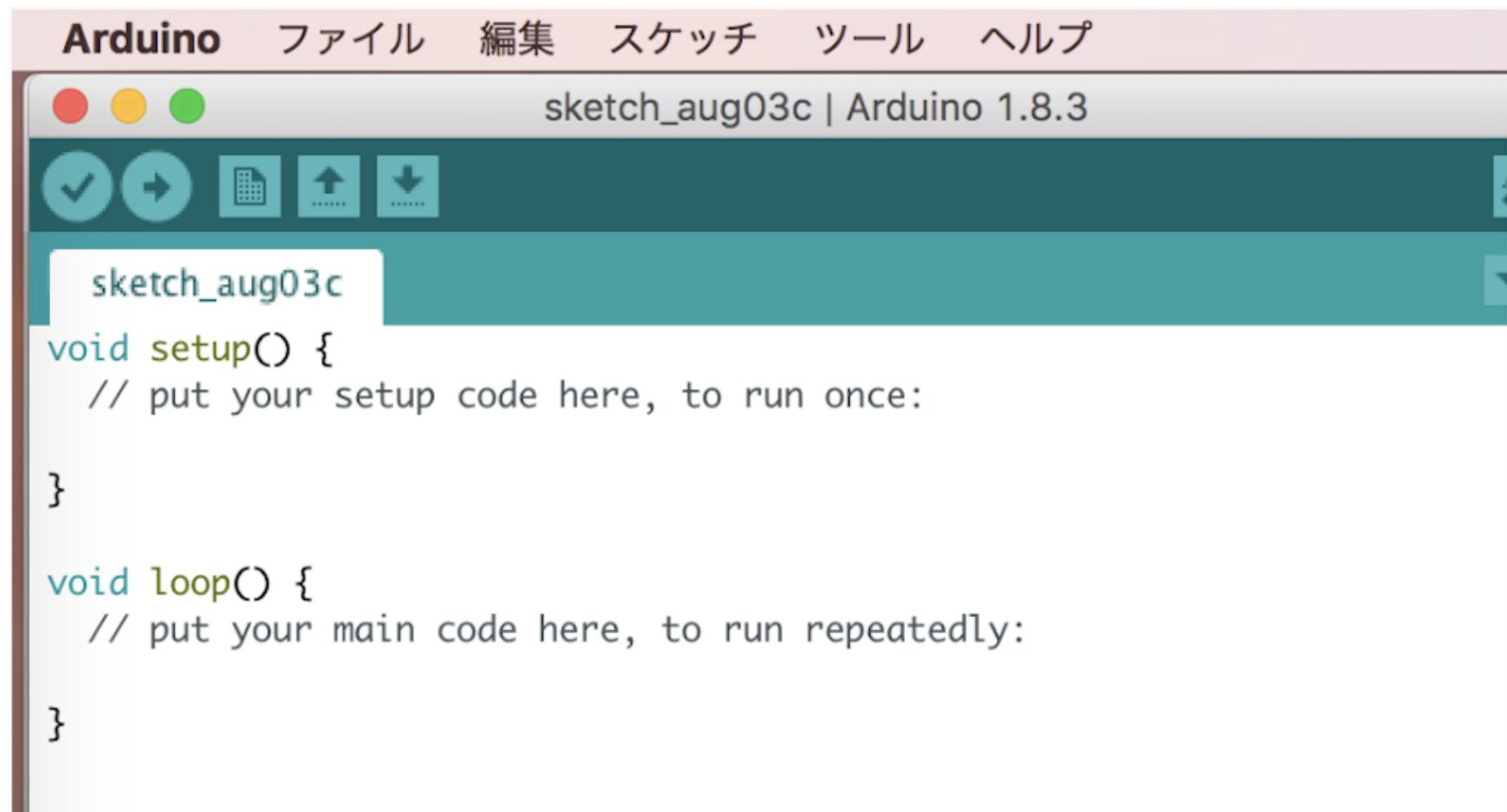
int red,green,blue;
void loop() {
    red=random(255); //random関数は0-255の数値をランダムに返します
    green=random(255);
    blue=random(255);
    Nefny.setLed(red,green,blue); //LEDがランダムに点灯します.
    String color="Red:";color+=red;
    color+=" Green:";color+=green;
    color+=" Blue:";color+=blue;
    Nefny.ndelay(1000); //1秒待つ
}
```

\*A0を指定していますが、乱数のSeed用に使っているだけで、内蔵のLEDがA0という訳ではありません。

## 2.LEDを光らせるプログラムを書いてみよう

### (4) プログラムの保存とボードに書き込み

左上の → ボタンを押し、ボード（Nefry BT）にプログラムを書き込む。



The screenshot shows the Arduino IDE interface. The menu bar includes 'Arduino', 'ファイル' (File), '編集' (Edit), 'スケッチ' (Sketch), 'ツール' (Tools), and 'ヘルプ' (Help). The title bar displays 'sketch\_aug03c | Arduino 1.8.3'. The toolbar below has icons for save, upload, and download. The code editor contains the following C-like pseudocode:

```
sketch_aug03c

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 2.LEDを光らせるプログラムを書いてみよう

### (5)確認

無事にプログラム書き込みが終わると、 Nefry BTに内蔵してあるLEDがカラフルに光ります。

\*プログラムが書き込めない場合は、 設定を以下のように変えてみましょう

ツール > Upload Speed : 460800

### 3.光センサの情報を取得してfirebaseに記録 してみよう

## **firebaseとは**

BaaS (Backend as a Service) の1つ.

Webアプリケーションやモバイルアプリケーションのバックエンドで行う機能を提供するクラウドサービス. 実装者がバックエンド側の実装をすることなく, 決められたAPIを叩くだけでクラウド上に用意された機能群を使うことが出来る.

## **使用するセンサ：光センサ**



### 3.光センサの情報を取得してfirebaseに記録してみよう

#### (1) 光センサの値を確認する

Nefry BTのA2 と書かれたソケットに光センサを接続する.



Arduinoを起動し、メニューのファイル>新規ファイルを選択。

新規ファイルに以下のプログラムを書き込み、左上の→ボタンを押し、実行。

```
#include <Nefry.h>
#define PIN A2

void setup()
{
    pinMode(PIN, INPUT);
}

void loop()
{
    long sum = 0;
    // 32回分の合計をする
    for(int i=0; i<32; i++)
    {
        sum += analogRead(PIN);
    }
    // 合計を割って平均を取る
    sum >>= 5;
    Nefry.println(sum);
    Nefry.ndelay(10);
}
```

### 3.光センサの情報を取得してfirebaseに記録してみよう

#### (1) 光センサの値を確認する

- ツール > シリアルモニタ(command+shift+M) を選択し,  
115200bpsに設定すると値を確認できる。
- ツール > シリアルプロット(command+shift+L) を選択し,  
115200bpsに設定するとグラフで可視化される

\*注意：同時に2つ開くことはできない

\*音センサも同じプログラムで実行可能

## (2) Firebaseへの記録

1. Firebase を開く <https://firebase.google.com/?hl=ja>
2. スタートガイドをクリック

The screenshot shows the official Firebase website. At the top, there's a navigation bar with the Firebase logo, a search bar, and links for 'プロダクト', '使用例', '料金', 'ドキュ >', 'コンソールへ移動', and a user profile icon. Below the navigation is a dark banner for 'Firebase Crashlytics' with the tagline 'Prioritize and fix issues with powerful, realtime crash reporting.' A large yellow graphic features a smartphone displaying various app icons like a lock, money, and a file, all labeled with the word 'Firebase'. The main text on the page reads 'Firebase helps mobile app teams succeed.' Below this are two blue buttons: 'スタートガイド' and '動画を見る'. At the bottom, there are three columns of text: 'アプリをすばやく作成、インフラストラクチャの管理は不要', 'Google のインフラが支える、多数の人気アプリが信頼するサービス', and '連係する全プロダクトを1つのコンソールで管理'. The footer contains the text 'Firebase のプロダクトは詳細な'.

スタートガイド 動画を見る

アプリをすばやく作成、  
インフラストラクチャの  
管理は不要

Google のインフラが支  
える、多数の人気アプリ  
が信頼するサービス

連係する全プロダクトを  
1つのコンソールで管理

Firebase のプロダクトは詳細な

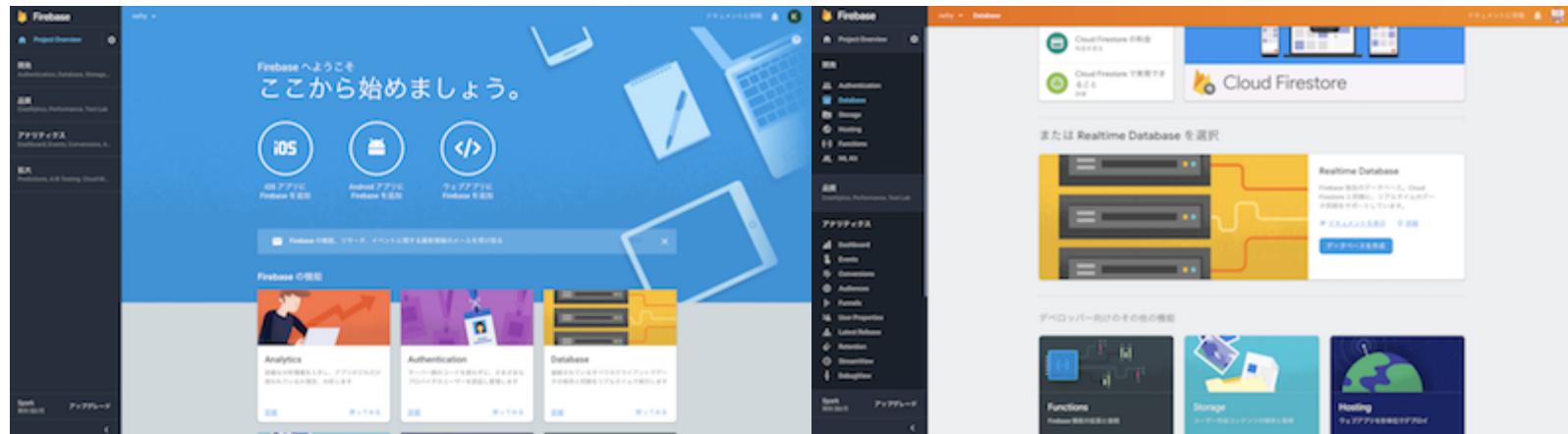
## (2) Firebaseへの記録

3.   ○ プロジェクト名 nefry
- アナリティクスと課金の地域 日本  
     と設定し、プロジェクトを作成



## (2) Firebaseへの記録

4. 「Database」を選択し、「Realtime Database」を選択



## (2) Firebaseへの記録

5. 「テストモードで開始」を選択し、「有効にする」をクリック

The screenshot shows the Firebase console interface. On the left, the navigation sidebar includes 'Project Overview', 'Authentication', 'Database' (selected), 'Storage', 'Hosting', 'Functions', and 'ML Kit'. Below these are sections for '品質' (Crashlytics, Performance, Test Lab) and 'アナリティクス' (Dashboard, Events, Conversions, Audiences, Funnels, User Properties). At the bottom, there are links for 'Spark' (無料 \$0/月) and 'アップグレード'.

The main area displays the 'Cloud Firestore' dashboard with sections for 'Cloud Firestore の料金' (Billing) and 'Cloud Firestore で実現できること' (What can be achieved with Cloud Firestore).

A modal window titled 'Realtime Database のセキュリティ ルール' (Realtime Database Security Rules) is open. It contains two radio button options:

- ロックモードで開始  
読み取りと書き込みをすべて拒否し、データベースを非公開で作成します。
- テストモードで開始  
読み取りと書き込みをすべて許可し、設定をすばやく行います。

The 'Test mode' option is selected. Below it is the JSON security rule code:

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

A yellow callout box highlights a note: 'データベース参照を所有しているユーザーなら誰でも、データベースの読み取りや書き込みを行えるようになります' (Any user who owns a database reference can read or write to it).

At the bottom of the modal are 'キャンセル' (Cancel) and '有効にする' (Enable) buttons.

## (2) Firebaseへの記録

6.databaseのURLをコピーしておく

The screenshot shows the Firebase Realtime Database console. On the left, there's a sidebar with navigation links for Authentication, Database (which is selected), Storage, Hosting, Functions, and ML Kit. Below that are sections for Quality (Crashlytics, Performance, Test Lab) and Analytics (Dashboard, Events, Conversions, Audiences, Funnels, User Properties, Latest Release, Retention, StreamView). At the bottom of the sidebar, it says "Spark 無料 \$0/月 アップグレード". The main area is titled "Database" and "Realtime Database". It has tabs for データ (Data), ルール (Rules), バックアップ (Backup), and 使用状況 (Usage). A modal window is open, displaying the database URL: <https://nefry-cef61.firebaseio.com/>. The modal also contains a warning message: "セキュリティルールが公開として定義されているため、誰でもデータベース内のデータを窃取、変更、削除できます。" (Security rules are publicly defined, so anyone can read, write, or delete data from the database.) There are "Go", "詳細 (Details)", and "閉じる (Close)" buttons at the bottom of the modal.

## (2) Firebaseへの記録

7. Nefryに以下のコードを書き込む

FIREBASE\_HOSTは先程コピーしたものと記述(https:// は除く)

```
#include <Nefry.h>
#include <NefryFireBase.h>
#define PIN A2
#define FIREBASE_HOST "xxxx.firebaseio.com"

NefryFireBase firebase;

void setup()
{
    pinMode(PIN, INPUT);
    firebase.begin(FIREBASE_HOST);
}

void loop()
{
    DataElement elem = DataElement();
    elem.setValue("sensor", analogRead(PIN));
    firebase.write("Nefry", &elem); //FireBaseのデータを書き込む
    Nefry.ndelay(1000);
}
```

## (2) Firebaseへの記録

8. 光センサの値がFireBaseでリアルタイムに更新されるのが確認できる

The screenshot shows the Firebase Realtime Database console interface. On the left is a sidebar with icons for Home, Project Overview, Functions, Storage, Firestore, Cloud Firestore, and a Recent Projects section. The main area has a blue header bar with the project name "nefry" and a dropdown for "Realtime Database". Below the header are tabs for "データ" (Data), "ルール" (Rules), "バックアップ" (Backup), and "使用状況" (Usage). A prominent orange warning banner at the top states: "セキュリティ ルールが公開として定義されているため、誰でもデータベース内のデータを窃取、変更、削除できます。" (A security rule is defined publicly, so anyone can read, write, or delete data from the database). It includes "詳細" (Details) and "閉じる" (Close) buttons. The main content area displays the database structure under "nefry-b2994":

- Nefry
  - sensor: 2295

**(3) webページを作成し,ブラウザで光センサの値を表示**

以下のソースコードのdatabaseURLを変更し、htmlページを作成。

```
#define PIN D2
#include <Nefry.h> // Nefryのライブラリをインクルード
// LEDCのパラメータ設定
//   LEDC_CHANNEL      : チャンネル : 0
//   LEDC_RESOLUTION_BITS: 目盛数    : 10bit (0~1023)
//   LEDC_FREQUENCY    : 周波数    : 50Hz (= 20ms周期)
#define LEDC_CHANNEL 0
#define LEDC_RESOLUTION_BITS 10
#define LEDC_FREQUENCY 50

// 30/1024*20ms = 0.59ms
// 77/1024*20ms = 1.50ms
// 122/1024*20ms = 2.38ms
uint32_t pulse[3] = {30, 77, 122};
int n = 0;

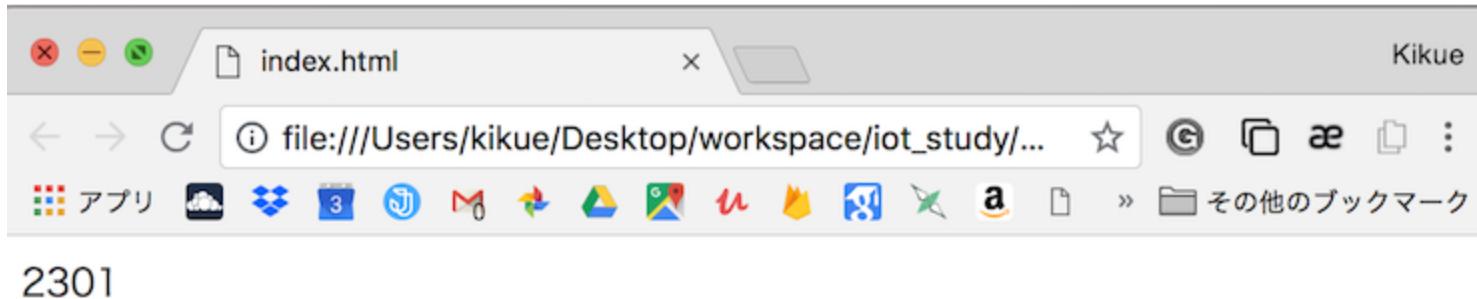
void setup() {
    ledcSetup(LEDC_CHANNEL, LEDC_FREQUENCY, LEDC_RESOLUTION_BITS);
    ledcAttachPin(PIN, LEDC_CHANNEL);
    Serial.println(pulse[n]);
    ledcWrite(0, pulse[n]);
    Nefry.enableSW();
}

void loop() {
    if (Nefry.readSW()) {
        n = (n+1) % 3;
        Serial.println(pulse[n]);
        ledcWrite(0, pulse[n]);
    }
}
```

### (3)webページを作成し、ブラウザで光センサの値を表示

Command+R or 左上の更新ボタンで更新すると、

光センサの値が確認できる。



## 付録(他のセンサの使い方)

光センサ以外のセンサーや出力装置のサンプルコードを紹介

サンプルによってはArduino IDEに外部ライブラリのインストールが必要

参考ページ <http://wiki.seeedstudio.com/Sensor/>

センサー一覧 <http://wiki.seeedstudio.com/>

サーボモータ



光センサ/音センサ/傾きセンサ/水センサ



モーションセンサ/温度湿度センサ/LEDbarセンサ



その他→ <https://dotstud.io/docs/grove/>

## 音センサ



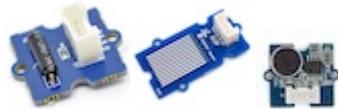
[http://wiki.seeedstudio.com/Grove-Sound\\_Sensor/](http://wiki.seeedstudio.com/Grove-Sound_Sensor/)

A0に接続し、下記プログラムを実行。

ツール > シリアルモニタ or シリアルプロットで値を確認できる

```
const int pinAdc = A0;
void setup()
{
    Serial.begin(115200);
}
void loop()
{
    long sum = 0;
    for(int i=0; i<32; i++)
    {
        sum += analogRead(pinAdc); //32回計測
    }
    sum >>= 5; // 平均を算出
    Serial.println(sum);
    delay(10); //10ms周期で計測
}
```

## 傾向センサ/水センサ(/音センサ)



```
#define PIN D2

void setup()
{
    Serial.begin(115200);
    pinMode(PIN, INPUT);
}

void loop()
{
    int v = digitalRead(PIN);
    Serial.println(v);
    delay(10);
}
```

ツール > シリアルモニタ or シリアルプロットで値を確認できる

## モーションセンサ (動きのある人を検知)



```
#define PIN D2

void setup()
{
    Serial.begin(115200);
    pinMode(PIN, INPUT);
}

void loop()
{
    if(digitalRead(PIN))//if it detects the moving people?
        Serial.println("Hi, people is coming");
    else
        Serial.println("Watching");

    delay(200);
}
```

## サーボモータ



```
#define PIN D2
#include <Nefry.h> // Nefryのライブラリをインクルード
// LEDCのパラメータ設定
//   LEDC_CHANNEL      : チャンネル : 0
//   LEDC_RESOLUTION_BITS: 目盛数    : 10bit (0~1023)
//   LEDC_FREQUENCY    : 周波数    : 50Hz (= 20ms周期)
#define LEDC_CHANNEL 0
#define LEDC_RESOLUTION_BITS 10
#define LEDC_FREQUENCY 50

// 30/1024*20ms = 0.59ms
// 77/1024*20ms = 1.50ms
// 122/1024*20ms = 2.38ms
uint32_t pulse[3] = {30, 77, 122};
int n = 0;

void setup() {
    ledcSetup(LEDC_CHANNEL, LEDC_FREQUENCY, LEDC_RESOLUTION_BITS);
    ledcAttachPin(PIN, LEDC_CHANNEL);
    Serial.println(pulse[n]);
    ledcWrite(0, pulse[n]);
    Nefry.enableSW();
}

void loop() {
    if (Nefry.readSW()) {
        n = (n+1) % 3;
        Serial.println(pulse[n]);
        ledcWrite(0, pulse[n]);
    }
}
```

## 温度湿度センサ



以下のライブラリをインストールする

[https://github.com/adafruit/Adafruit\\_Sensor](https://github.com/adafruit/Adafruit_Sensor)

<https://github.com/adafruit/DHT-sensor-library>

# 外部ライブラリのインポートの方法

## 1.zip形式でダウンロード

The screenshot shows the GitHub repository page for 'adafruit / Adafruit\_Sensor'. The repository is described as a 'Common sensor library' for Arduino, containing code, issues, pull requests, ZenHub, projects, a wiki, and insights. It has 35 commits, 1 branch, 3 releases, and 6 contributors. A commit by microbuilder from Oct 27, 2016, is listed. The 'Clone or download' button is highlighted in green. Below the repository details, there's a section titled 'Adafruit Unified Sensor Driver' with a brief description of its purpose and challenges. The URL at the bottom of the page is [https://github.com/adafruit/Adafruit\\_Sensor/archive/master.zip](https://github.com/adafruit/Adafruit_Sensor/archive/master.zip).

adafruit / Adafruit\_Sensor

Common sensor library

35 commits 1 branch 3 releases 6 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

microbuilder committed on 27 Oct 2016 Updated link to DHT repo

Adafruit\_Sensor.h mark unimplemented virtual functions as abstract

README.md Updated link to DHT repo

library.properties Remove unused avr/pgmspace.h reference to fix #8.

README.md

### Adafruit Unified Sensor Driver

Many small embedded systems exist to collect data from sensors, analyse the data, and either take an appropriate action or send that sensor data to another system for processing.

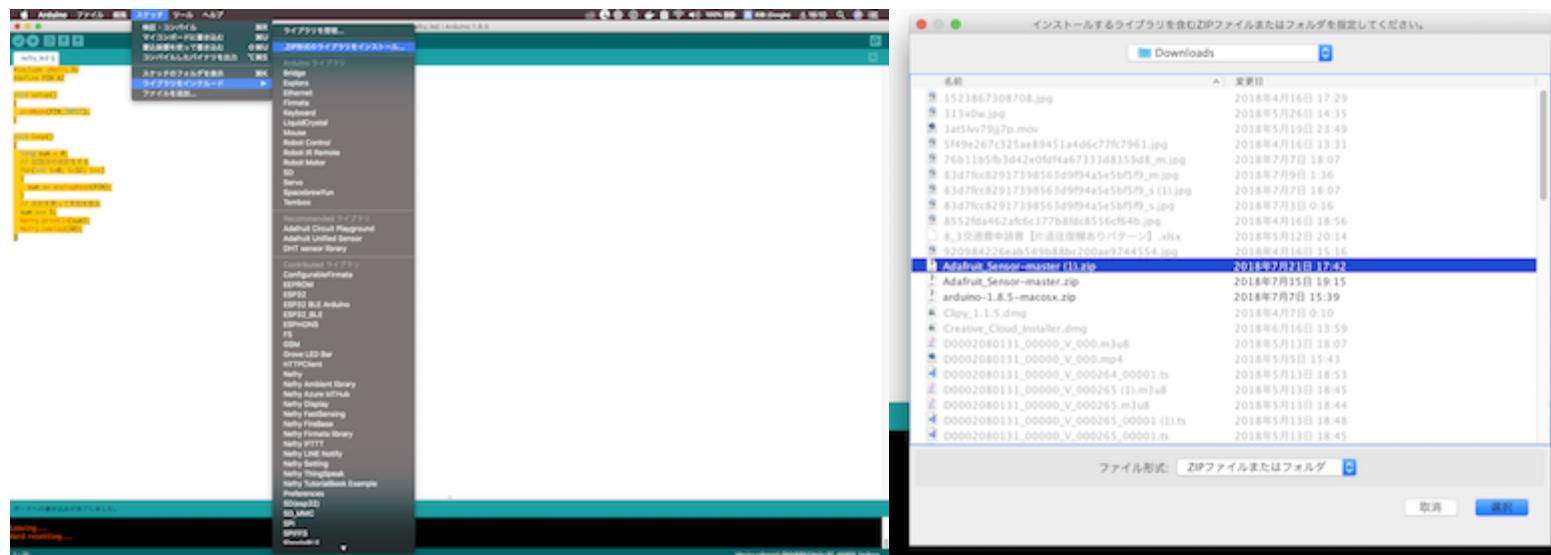
One of the many challenges of embedded systems design is the fact that parts you used today may be out of production tomorrow, or system requirements may change and you may need to choose a different sensor down the road.

Creating new drivers is a relatively easy task, but integrating them into existing systems is both error prone and time consuming since sensors rarely use the exact same units of measurement.

By reducing all data to a single `sensors_event_t` 'type' and settling on specific, standardised SI units for each sensor family, the same sensor types return values that are comparable with any other similar sensor. This enables you to switch

[https://github.com/adafruit/Adafruit\\_Sensor/archive/master.zip](https://github.com/adafruit/Adafruit_Sensor/archive/master.zip)

## 2.Arduino IDEを起動し、スケッチ > ライブラリをインクルード > ZIP形式のライブラリをインストール

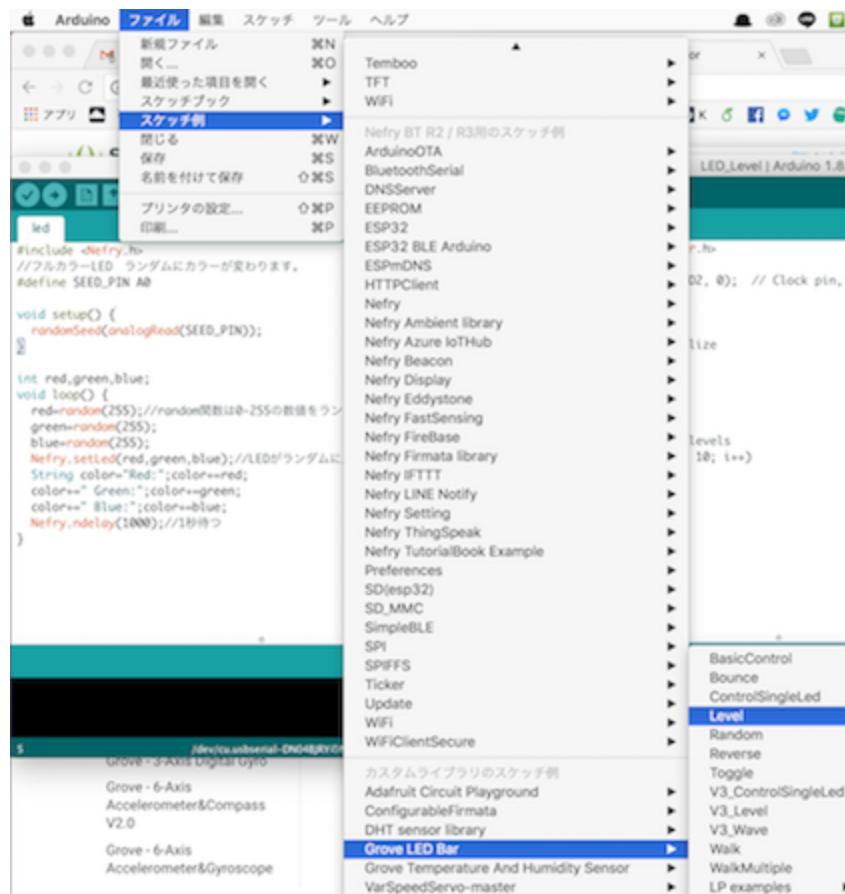


次のページのプログラムを実行

```
1 #include <Nefry.h> // Nefryのライブラリをインクルード
2
3 #include "DHT.h"
4 #define DHTPIN D2      // 値をD4に変更（接続するピンに応じて変更）
5
6 #define DHTTYPE DHT11 // DHT 11に変更
7
8 DHT dht(DHTPIN, DHTTYPE);
9
10 void setup() {
11     //Serial.begin(9600); 削除
12     Nefry.println("DHTxx test!"); // SerialをNefryへ変更
13     dht.begin();
14 }
15
16 void loop() {
17     Nefry.ndelay(2000); // Serial.delayをNefry.ndelayへ変更
18
19     float h = dht.readHumidity();
20     float t = dht.readTemperature();
21     float f = dht.readTemperature(true);
22
23     if (isnan(h) || isnan(t) || isnan(f)) {
24         Nefry.println("Failed to read from DHT sensor!"); // SerialをNefryへ変更
25         return;
26     }
27
28     float hif = dht.computeHeatIndex(f, h);
29     float hic = dht.computeHeatIndex(t, h, false);
30     Nefry.print("Humidity: ");    // SerialをNefryへ変更
31     Nefry.print(h);              // SerialをNefryへ変更
32     Nefry.print(" %\t");        // SerialをNefryへ変更
33     Nefry.print("Temperature: "); // SerialをNefryへ変更
34     Nefry.print(t);             // SerialをNefryへ変更
35     Nefry.print(" *C ");        // SerialをNefryへ変更
36     Nefry.print(f);             // SerialをNefryへ変更
37     Nefry.print(" *F\t");       // SerialをNefryへ変更
38     Nefry.print("Heat index: "); // SerialをNefryへ変更
39     Nefry.print(hic);           // SerialをNefryへ変更
40     Nefry.print(" *C ");        // SerialをNefryへ変更
41     Nefry.print(hif);           // SerialをNefryへ変更
42     Nefry.println(" *F");       // SerialをNefryへ変更
43 }
```

# LEDbarセンサ

1. ライブラリ [https://github.com/Seeed-Studio/Grove\\_LED\\_Bar](https://github.com/Seeed-Studio/Grove_LED_Bar) をダウンロード
2. Arduino IDEを起動し、スケッチ > ライブラリをインクルード > ZIP形式のライブラリをインストール
3. スケッチ例 > Grove LED Bar > Levelをクリック



## LEDbarセンサ



### 4. 下記プログラムを実行

```
#include <Grove_LED_Bar.h>
//Clock pin, Data pin, Orientation
Grove_LED_Bar bar(D3, D2, 0);

void setup()
{
    // nothing to initialize
    bar.begin();
}

void loop()
{
    // Walk through the levels
    for (int i = 0; i <= 10; i++)
    {
        bar.setLevel(i);
        delay(100);
    }
}
```