

Scratch Hacks!!

[Scratchと遊ぼう:NT山城版](#)

はじめに

みなさん、Scratchはご存知ですか？ブロックを積み重ねてプログラムを創るScratchは、ひょっとしたら学校やCoderDojoなどで使ったことがあるかもしれません。

本書では、Scratchの基礎などについては、一切説明しません。良書がたくさんでているので、それらを参考にしてください。

私が初めてScratchと出会ったのは、「Scratch: A Sneak Preview」という論文を読んだことでした。この時は、まだアルファ版で、今とはかなり雰囲気の異なる環境になっています。この論文ではあまりそのスゴさ理解できていなかったのが、私の先見の明がなかったところです。

当時は、Squeakというブロックプログラミングに似たタイルスクリプティングベースの環境で遊んでいました。C5という京都大学で行われていた国際会議なども定期的に行われており、Squeakに関連した人たちが多く参加していました。

Squeakの時代から、その言語環境内で閉じるのではなく、外部のデバイスと一緒に使う試みがたくさんありました。Scratchの場合は公式拡張機能を使ってLEGOシリーズやmicro:bitと一緒に使えることを知っている人も多いかもしれません。こういうデバイスと一緒にScratchを使うことは、私には大好物です。

本書ではこれらを踏まえて、Scratchの少し(?)変わった使い方の紹介をします。

以下のようなデバイスやサービスをScratchと一緒に使用します。

デバイスとしては、以下ののようなものを使用します。各デバイスの詳しい使い方についても、もうしわけありませんが、本書では説明しません。動作に必要な最低限の説明だけを行います。

- M5Stack
 - スタックチャン
- micro:bit
- AkaDako
- hapStak
- LEGO
- Seeeduino XIAOシリーズ, WioTerminal

また、以下のようなサービスを利用します。

- IFTTT
- [not yet]ChatGPT
- [not yet]TeachableMachine
- [not yet]音声認識
- [not yet]画像分類器
- [not yet]ポーズ認識

みなさんのScratchライフが楽しいものになれば筆者としてはうれしいです。

Happy Scratch Hacking!!

Scratchってなあに?

ScratchはMITメディアラボが開発したプログラミング言語です。プログラムは、ブロックを積み上げて創っていきます。しばしば、小学校などのようなプログラミング学習の入門用の教育用途で利用されています。作者たちのスローガンは、「全ての年齢の子どもたちに!!」であり、大人の利用も視野に入っています。

Scratchは教育用途という先入観があると思いますが、作者たちは、以下のような多様な用途に利用できるように設計をしています。

- 低い床: はじめやすく
- 高い天井: 高度なこともでき
- 広い壁: いろいろなものを作れる

個人的には、少し変わった変態言語と感じています。詳しくは説明しませんが、例えば以下のようないくつかの特徴があります。

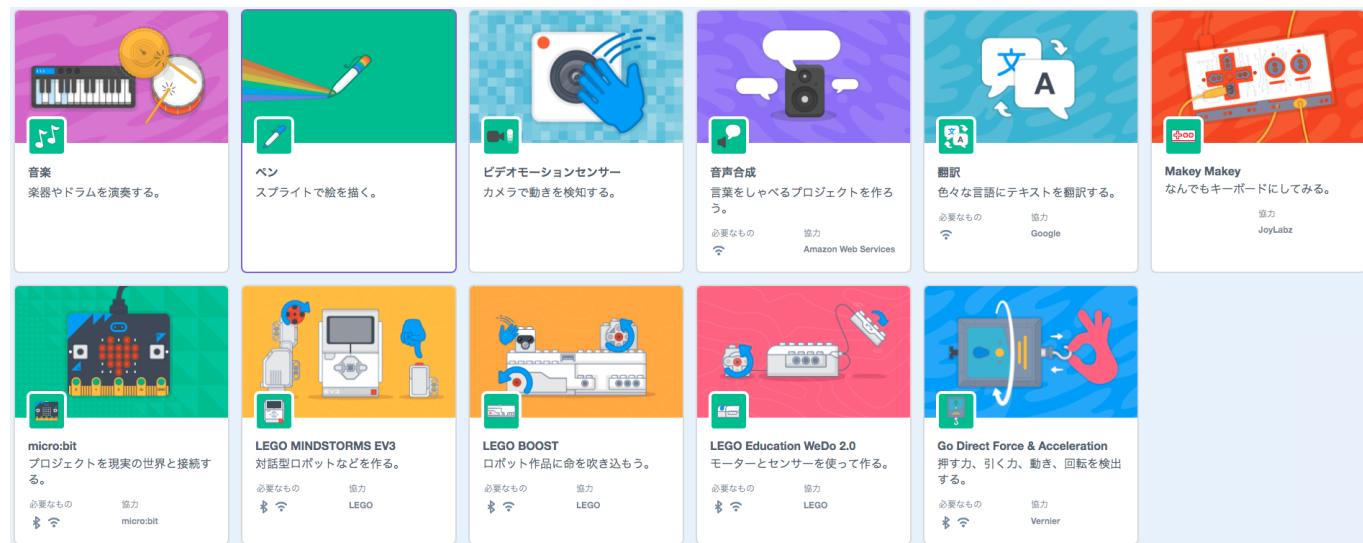
- 並列実行: プログラムはごく自然に並列に実行するように書かれます。
- イベントベース: プログラムは、イベントの送受信を使って書かれます。
- オブジェクトベース: 全てはオブジェクトです。
 - スプライト: 表示されているキャラクターはオブジェクトです。
 - クローン: スプライトは複製を作ることができ、これも自律的に動かせます。

公式Scratchサーバーは <https://scratch.mit.edu/> で公開されています。

Scratchのサーバーはオープンソースで公開されています。そのため、比較的簡単に自分好みのサーバを創ることもできるようになっています。これを自分の手元で立ち上げる方法も本書では解説します。

Scratch拡張機能

Scratchには拡張機能という仕組みがあり、その機能を簡単に追加していくことができるようになっています。公式のScratchサーバーで利用できる拡張機能は以下の図のようなものです。



一見基本的にみえる「ペン」や「音楽」などの他に、文字列を翻訳する「翻訳」や文字列を音声合成して読み上げる「音声合成」、ビデオ入力から動きを検知する「ビデオモーションセンサー」などがあります。ハードウェアとして

は、「micro:bit」やLEGO(MINDSTORMS EV3, BOOST, WeDo)などが利用可能になっています。

拡張機能は、ユーザーから見ると他のScratchの基本的な機能と変わりなく利用可能です。 サーバー開発者視点から見ると、Scratchに機能を追加するためのフレームワークになっています。 これを使うことで、比較的簡単にScratchに機能を追加することができるのです。

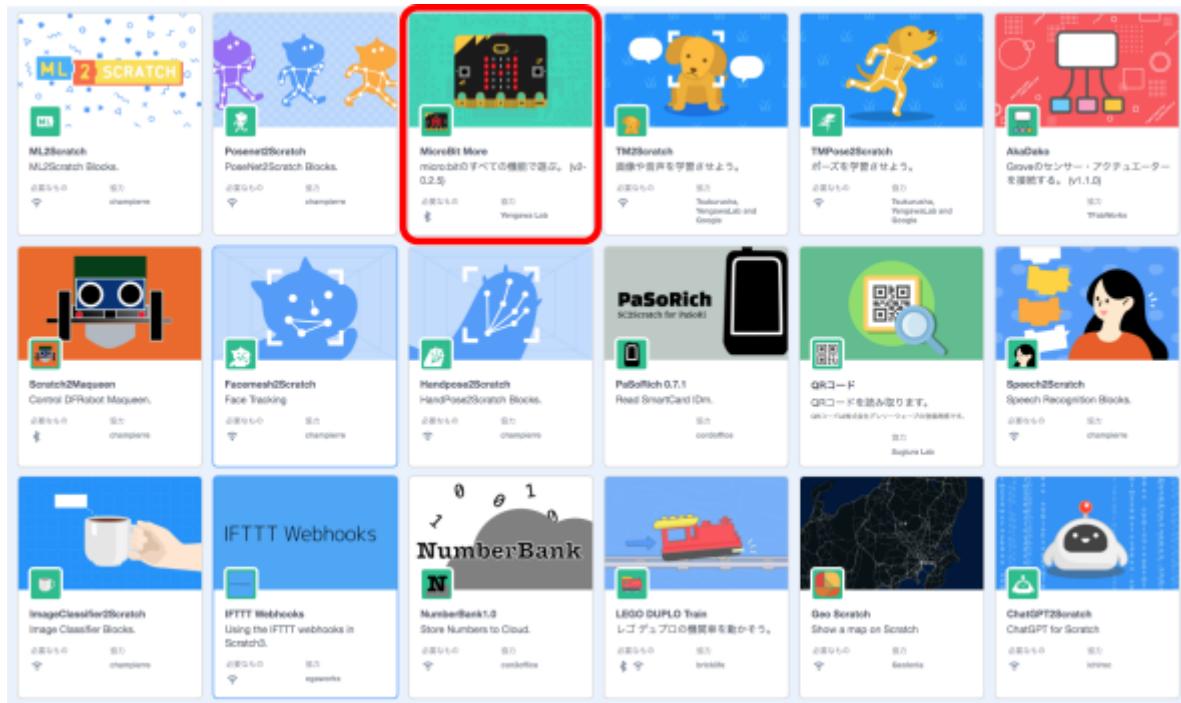
Scratchの拡張機能を独自に追加したサーバーも各所で提供されています。 例えば、以下のようなサーバーがあります。 本書では、Stretch3を使っています。

- [Stretch3](#): AI系に強い
- [Xcratch](#): 拡張機能動的追加型
- [blicklife](#): LEGOに強い
- [toioDo](#): toio用環境
- [CodeSkool](#): なんかいっぱい
- [つくるっち](#): ハードウェア系に強い

Stretch3(ストレッチスリー)

Stretch3(ストレッチスリー)は、公式の拡張機能以外に主に日本の開発者たちが開発した拡張機能が利用できるようにしたScratchサーバーです。これらの拡張機能は、主に日本の開発者たちが開発しており、オープンソースで公開されています。

Stretch3の拡張機能には、以下の図のようなものがあります。



今回、本書で解説しようと考えている拡張機能は、以下のとおりです。

- Microbit More
- AkaDako
- LEGO DUPLO Train
- IFTTT
- [not yet]ChatGPT(ChatGPT2Scratch)
- [not yet]TeachableMachine(TM2Scratch, TMPose2Scratch)
- [not yet]音声認識(Speech2Scratch)
- [not yet]画像分類器(ImageClassifier2Scratch)
- [not yet]ポーズ認識(Posenet2Scratch, Facemesh2Scratch, Handpose2Scratch)

Microbit More拡張機能

Microbit Moreは、Scratchから micro:bit のフル機能を利用できるようにした拡張機能です。「公式にも拡張機能があるのでは?」と思うかもしれません、こちらはかなり機能が限定されており、私はMicrobit Moreを利用することをおすすめします。以下のような違いがあります。

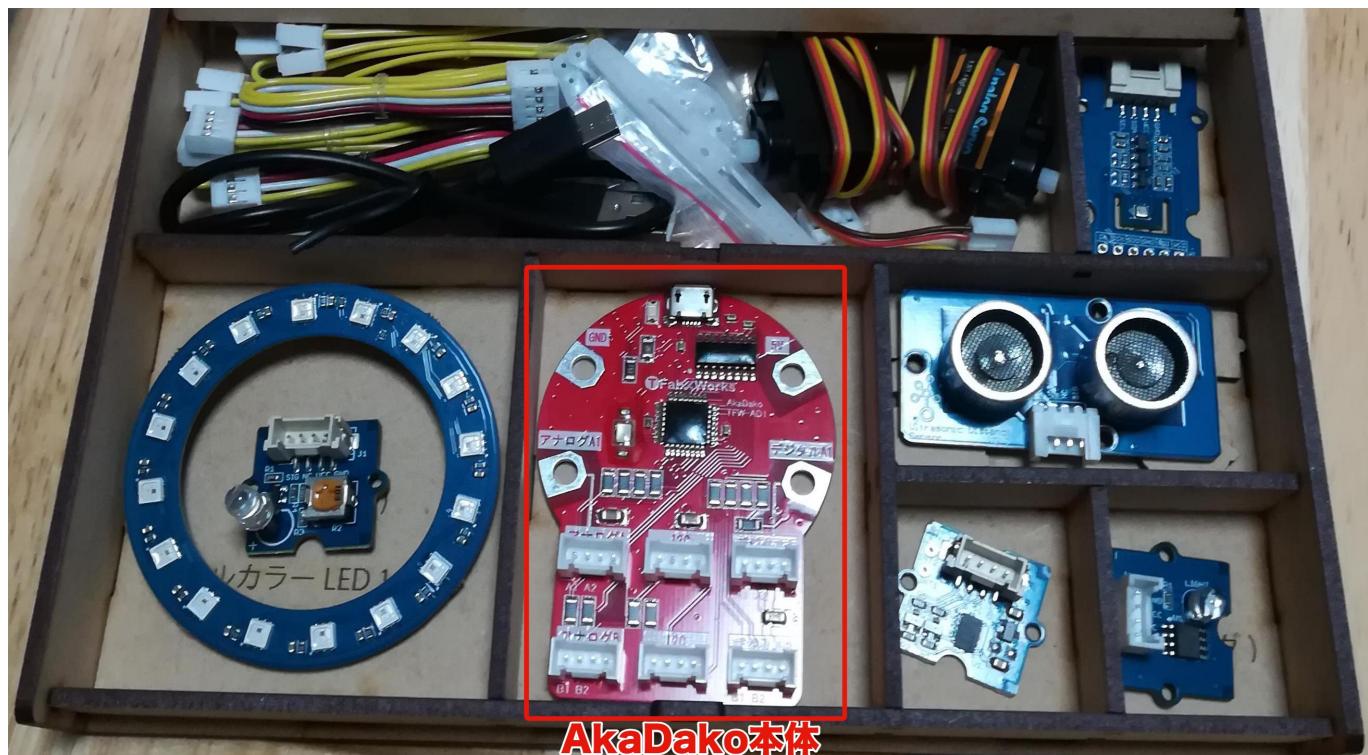
micro:bit拡張機能 Micorbit More

ボタン	○	○
-----	---	---

	micro:bit拡張機能	Micorbit More
LED表示	○	○
簡易な状態による操作	○	○
加速度の利用	×	○
ピンの利用	限定的	完全に利用可能

Microbit Moreはmicro:bitでの利用の他に、拙作のM5bitLessを使って、M5Stackシリーズのマイコンで利用することも可能です。

AkaDako拡張機能



AkaDakoはTFab Worksによって開発および販売されている、Scratch用のGroveシールドです。USBで有線接続をします。教育向け用途を指向しており、各種学習単元に沿った実験ができるようになっています。機種によっては、amazonやTFab Worksオンラインショップ、スイッチサイエンスなどで一般向け販売も行われています([製品ページ](#))。

Groveとは、Seeedが中心となって進めているハードウェアを接続するためのインターフェースです。Groveには、デジタルI/O、アナログI/O、I2Cの種類があり、Grove対応の様々なデバイスを接続することが可能になっています。

AkaDakoでは、初めから光センサーや加速度計や距離などのセンサーヤ、サーボモータやLEDアレイなどのアクチュエータがブロックで簡単に利用可能になっています。更に、I2Cに対する操作ができるブロックも用意されているため、自分で対応されていないI2Cデバイスを利用することも可能になっています。

例えば、I2C接続の環境光センサーのBH1750で明るさを取得するコードは、以下の図のようになります。



参考文献

- ビジュアルプログラミングでブルブルブルッ:AkaDakoとhapStakを使って、入力でブルブル震えるシステムを作成してみました。

IFTTT

[IFTTT](#)

M5Stackと遊ぼう!!

M5Stack

M5Stackってなあに?



M5Stackは、色々な機能が一つになった、とても便利なマイコンデバイスです。

M5Stackには、機種によって違いますが、以下のような機能があります。各機種のリンクは、日本の正規代理店のひとつであるスイッチサイエンスの販売ページになっています。

- CPU: ESP32C, ESP32S, K210など
 - ディスプレイ: 320x240TFT([Core](#), [Core2](#), [CoreS3](#)), 80x160TFT(M5StickC), 135x240TFT([C Plus](#), [C Plus2](#)), 240 x 135TFT([Cardputer](#)), 5x5フルカラーLED([ATOM Matrix](#))
 - バッテリー: 150mAh(Core), 390mAh(Core2), 500 mAh(CoreS3), 80or95mAh(C), 120mAh(C Plus), 120 mAh+1400 mAh(Cardputer)
 - ネットワーク: Wi-Fi(2.4G) + BLE
 - センサー: 加速度センサー、ジャイロ、ボタンスイッチ、温度センサーなど
 - オーディオ: マイク, スピーカー

- カメラ: 30万画素
- その他I/O: Grove A(I2C), B(A/D,GPIO), C(UART), M-BUSなど
- 付属品: キーボード、腕時計バンド、LEGOマウント

初めて買うときの私のおすすめは、[M5StickC Plus2 ウォッチアクセサリキット](#)です。腕に巻いて遊べます!!

ただ、後で説明するスタックチャンで遊びたい場合は、現状ではCore2シリーズ(Core2, [Core2aws](#))を買っておいた方が良いです。

M5StackとScratchで遊ぶための参考文献

- [M5StackとScratchで遊ぶたった3つの冴えたやり方](#): M5StackとScratchで遊ぶためのUIFlow, M5Scratch, M5bitLessの3(+1)つの方法について解説しています。
- [ScratchとM5Stackで遊ぶ](#): Scratch遠隔センサーについての説明が少し詳しめです。

M5bitLess: M5StackでMicrobit Moreを使う

参考文献

- [M5bitLess: M5Stack x Scratch3 = So Fun!!](#): M5bitLessのシステム全体を知るのに良いと思います。
- [M5bitLess label & data extension](#): データをやり取りするLabelとDataというしくみの説明です。
- [M5bitLessのI/Oサポート](#): M5bitLess外部のハードウェアを利用するための拡張についての説明です。
- デモ類
 - [M5StackとScratchとhapStakでスポーツの秋に挑戦!!](#): M5bitLessを使って、運動するゲームを作りました。

その他のボードで*bitLess系列を使う

- Seeeduino XIAOシリーズ (ESP32C3, nRF52480(Sense)): [XIAO32bitLess](#)
- Seeeduino WioTerminal: M5bitLessに対応コードあり
- IoT Algyan XIAOGYAN: [XIAOGYANbitLess](#)

参考文献

- [いろんなボードからScratchを使おう](#): 色々なボードでScratchを使っている例です。

Scratch1.4 遠隔センサープロトコル

Scratch 1.4

Scratch 1.4は、最初に公開されたバージョンのScratchです。 アプリケーションとして提供されており、インターネット接続がないオフラインでも利用可能です。 Smalltalk([Squeak](#))で記述されています。

Scratch 1.4はいまでもよく利用されています。 例えば、Raspberry Piの標準OSには、Scratch 1.4が含まれています。

遠隔センサープロトコル(RSP)

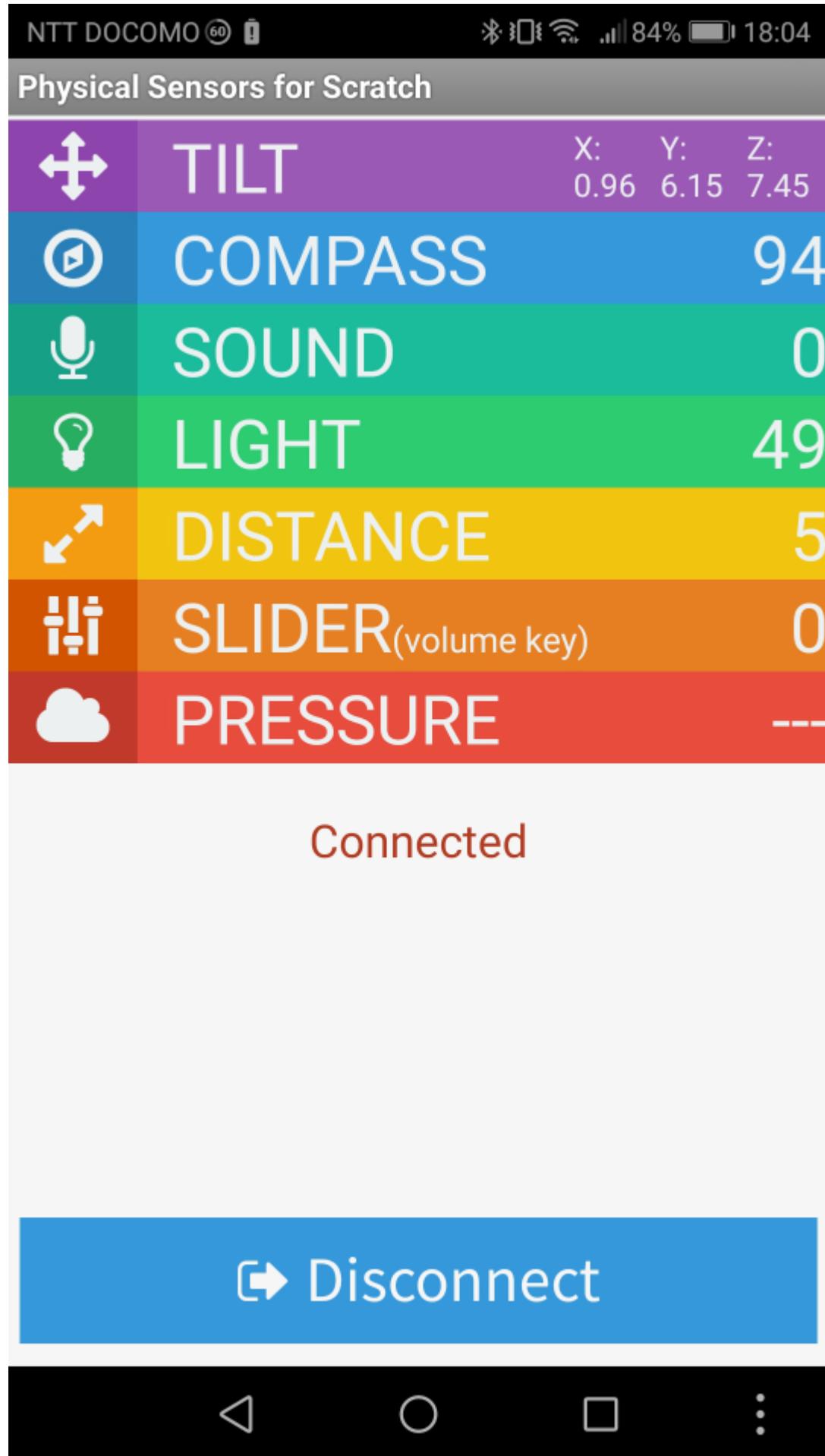
Scratch遠隔センサーパロトコル([Remote Sensors Protocol](#):以下RSP)は、Scratchとネットワークでつながった外部の何かと情報をやり取りするための仕組みです。 Scratch同士の通信も可能になっています。 基本的にはTCP/IP 42001を利用してやり取りしますが、 UDP/IP 42001も利用可能です。

RSPを利用するためには、少し準備が必要です。 Scratch側で遠隔センサーを有効にする必要があります。 そのためには以下の図のように、調べる->"スライダセンサーの値"を右クリック->"遠隔センサー接続を有効にする"を選びます。



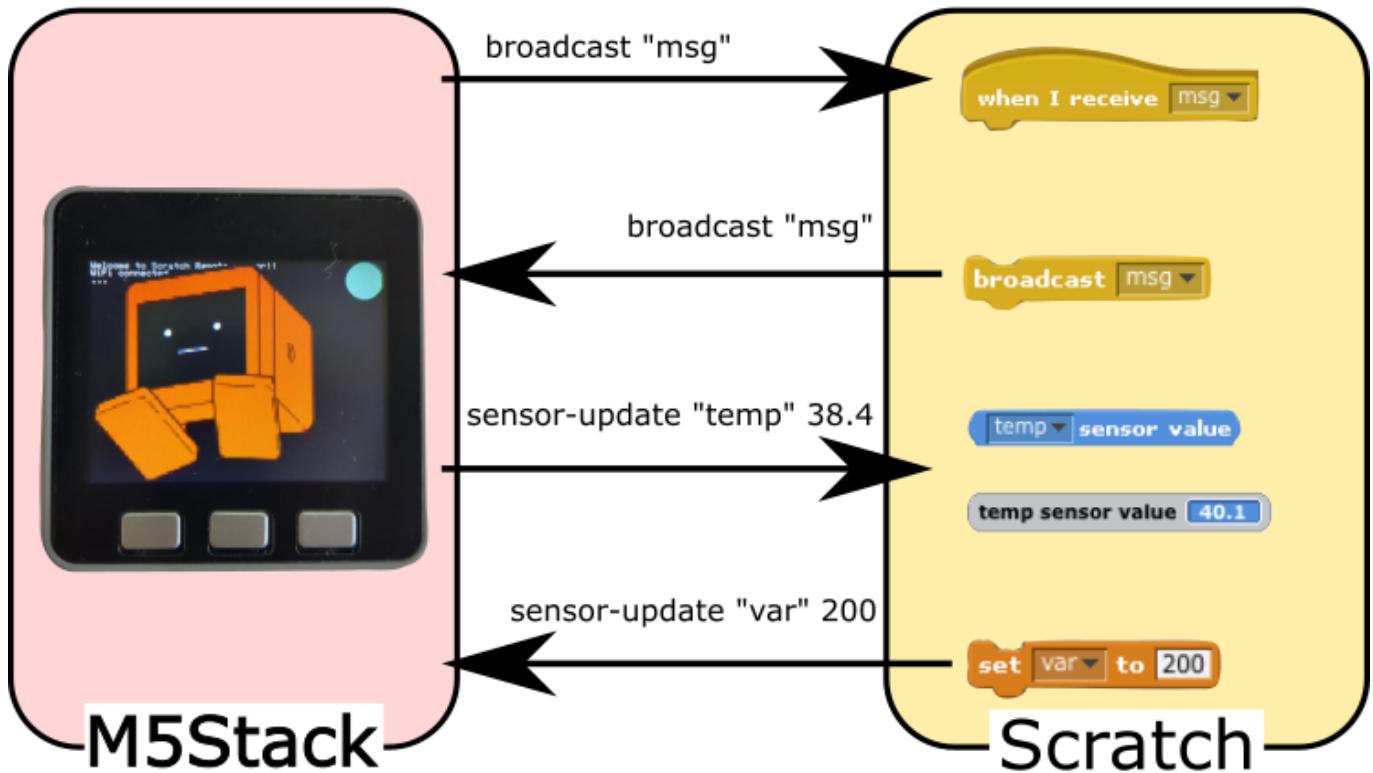
RSPをスマートフォンから利用するためのアプリケーションもいくつか提供されています。 これらのアプリケーションでは、スマートフォン内蔵の加速度センサーなどの情報を、Scratchから利用できるようになります。

- Android
 - [Physical Sensors for Scratch](#)
 - [Scratch Sensor](#)
- iOS: 軽く探した範囲では見つかりませんでした。情報求む!!



RSPはテキストベースのプロトコル、コマンドが2種類だけ提供されています。メッセージのやり取りを行う
broadcast "メッセージ" と 変数のやり取りを行うsensor-update "変数名" "値" ... です。

Scratch Remote Sensor Protocol



Scratch x M5Stack UIFlow

[UIFlow](#)はM5Stackの標準開発環境のひとつです。 様々なM5Stack純正のGroveデバイス類が簡単に利用できるようになっています。

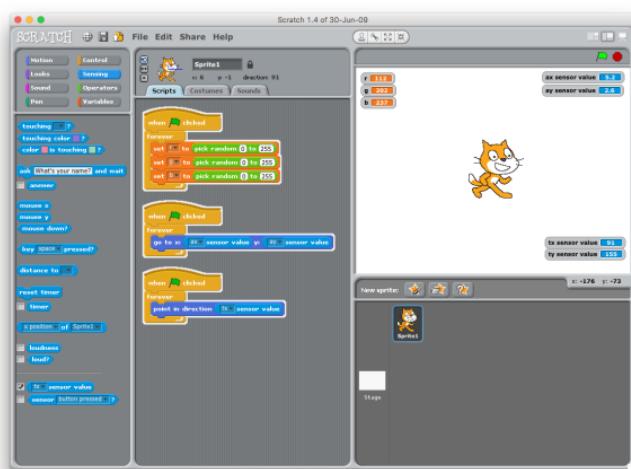
提供されている機能の中にUDPによる通信があるため、RSPを利用することが可能になります。

例えば、以下の例では、M5Stackの加速度センサーの値に応じて猫を動かすことと、ボタンスイッチの値をM5StackからScratchへ伝えることができます。 ただ、なぜかScratchからM5Stack方向の通信はうまくできないようです。



M5Scratch = Scratch x M5Stack

M5Stack



参考文献

- M5Scratch: M5Stack x Scratch1.4:システム全体の説明です。すみません英語です。

- デモ類

- [M5Scratchを使って、M5Scratchの仲間たちとScratchを使ったゲームを作ろう!!:M5Scratchでゲームを作ってみた例です。](#)
- [M5Stack Christmas with M5Scratch:これもゲームを作ってみた例です。すみません英語です。](#)

スーパーかわいいロボットスタックチャン



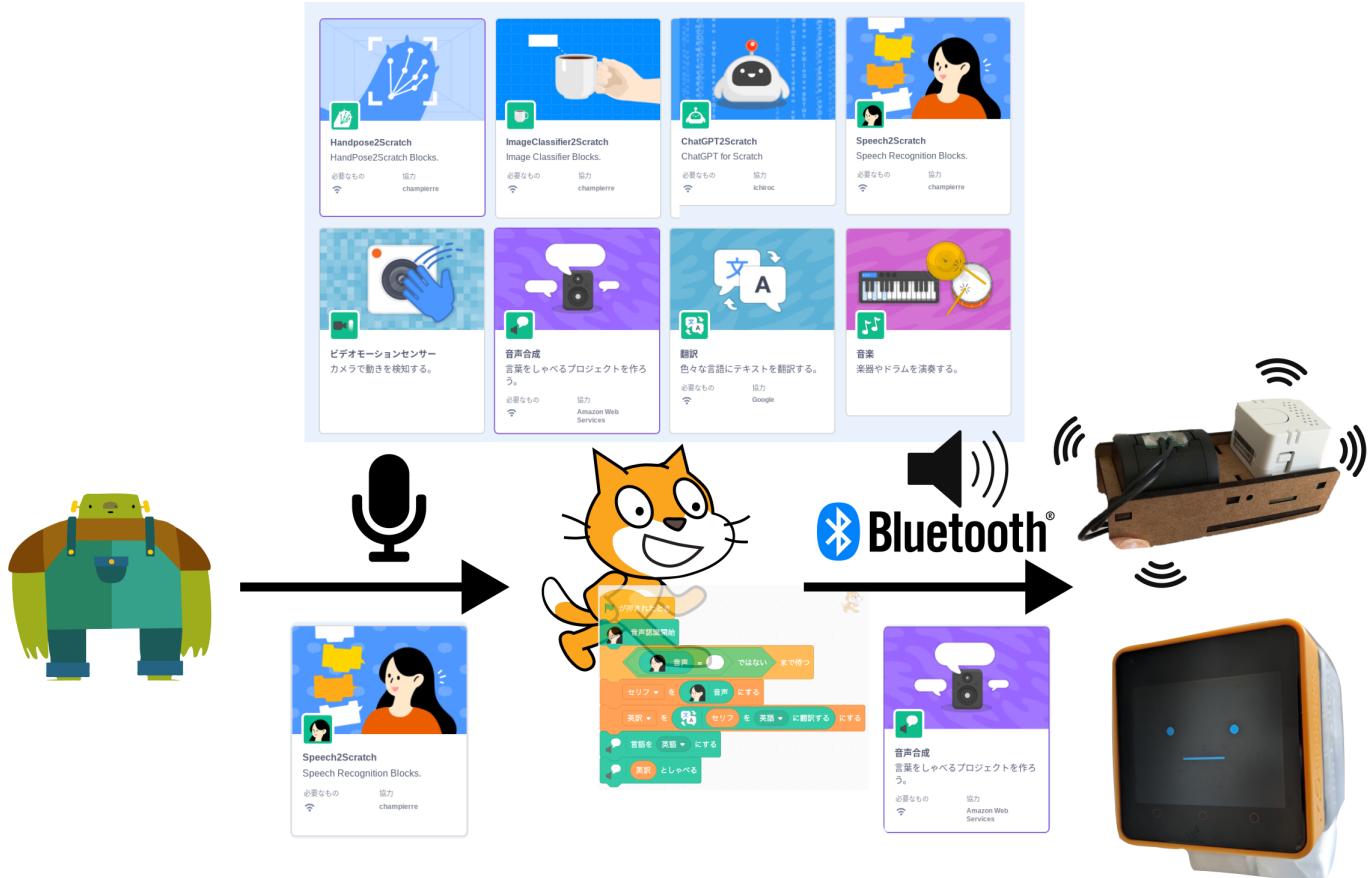
スタックチャン (Stack-chan) は、しあわさんが開発、公開している手乗りサイズのスーパーかわいいコミュニケーションロボットです。スタックチャンはオープンソースで開発されており、ハードウェア(サーボのドライバーや体(筐体)の3Dデザインなど)もソフトウェア(AI用やBluetoothスピーカ版など)もオープンなものがたくさん存在します。

スタックチャンには、様々な亜種が存在します。例えば、ソフトウェア的にはAIが使えるバージョンやスピーカーになるバージョンがあります。ハードウェア的には、タンク形状のものや、かなり小型化されたいるものなど実に多様です。[みんなのスタックチャン作例集](#)に色々なスタックチャンが紹介されています。なんと、等身大で自律的に動くスタックサンというヒューマノイドまであります。

残念なことに、2024年5月時点では、これだけを買えばすぐに使えるというものは存在しません。体(筐体)の種類もいろいろあり、動かすためのサーボモーターの種類や、M5Stackの機種による違いなどがあり、簡単に説明することができない状態です。これは、将来的には解決されると思うのですが、こういう現状のためここではスタックチャンの作り方については説明しません、というかできません。AIスタックチャンの作成方法に関しては、動画「[知識ゼロで作る！ 手乗りサイズのスーパーかわいいボット AIスタックチャン2PLUS版](#)」を見ると作成作業の手順がつかめると思います。

スタックチャンは動きが素敵だと思うのですが、動かなくても良いと割り切れる場合はCore2またはCore2awsだけを使って、そのスゴさを確認することはできると思います。

音声入出力を使って遊ぼう



[スタックチャンとScratchチャン](#)

参考文献

- [Scratch V2 Programming:スタックチャンとhapStakを音声で使っています。](#)

M5bitLessを使って遊ぼう

[スタックチャン meets Scratch with M5bitLess](#)

Scratchサーバーを自分で立ち上げる

Scratchのサーバーはオープンソースになっており、誰でも手元で立ち上げることが可能です。以下では、自分で立ち上げるサーバーのことを、オレオレサーバーと呼びます。

本章では、実際にオレオレサーバーを立ち上げる手順を説明していきます。

具体的に手順に関しては、適宜[大人のためのScratch Scratch を改造しようの 2. Scratch 3を自分のパソコンで動かしてみよう](#)(Windowsでの手順)を参照してください。

事前に必要な準備は以下のとおりです。

- `git`コマンドのインストール
- Node.jsの環境構築

[!NOTE] Ubuntuでの手順は、以下のようになります。

```
$ sudo apt install -y git nodejs npm
```

[!NOTE] FreeBSDでの手順は、以下のようになります。

```
$ sudo pkg install -y git npm
```

追加拡張機能の無いScratch3サーバーを準備する手順は以下のとおりです。

```
# Scratchサーバーのリポジトリを取得する。  
$ git clone --depth 1 https://github.com/LLK/scratch-gui  
$ cd scratch-gui  
# 実行準備を行う。  
$ npm install
```

サーバーを起動するには、以下のようにコマンドを実行します。

```
$ npm start
```

これで、<http://localhost:8601/> にアクセスすると、いつものようなScratchの画面が表示されるはずです。

更に進んだScratch3サーバーを立ち上げる方法として、githubを使ってオレオレScratchサーバーを立ち上げることも可能です。具体的には、[GitHub Actions で独自 Scratch を動かす](#)を参照してください。

拡張機能を追加する

オレオレサーバーに公式でない拡張機能を追加することも可能です。

Stretch3サーバーで実際にどのようにインストールされているかは、
<https://github.com/stretch3/stretch3.github.io/blob/source/.github/workflows/deploy.yml> を参照してください。

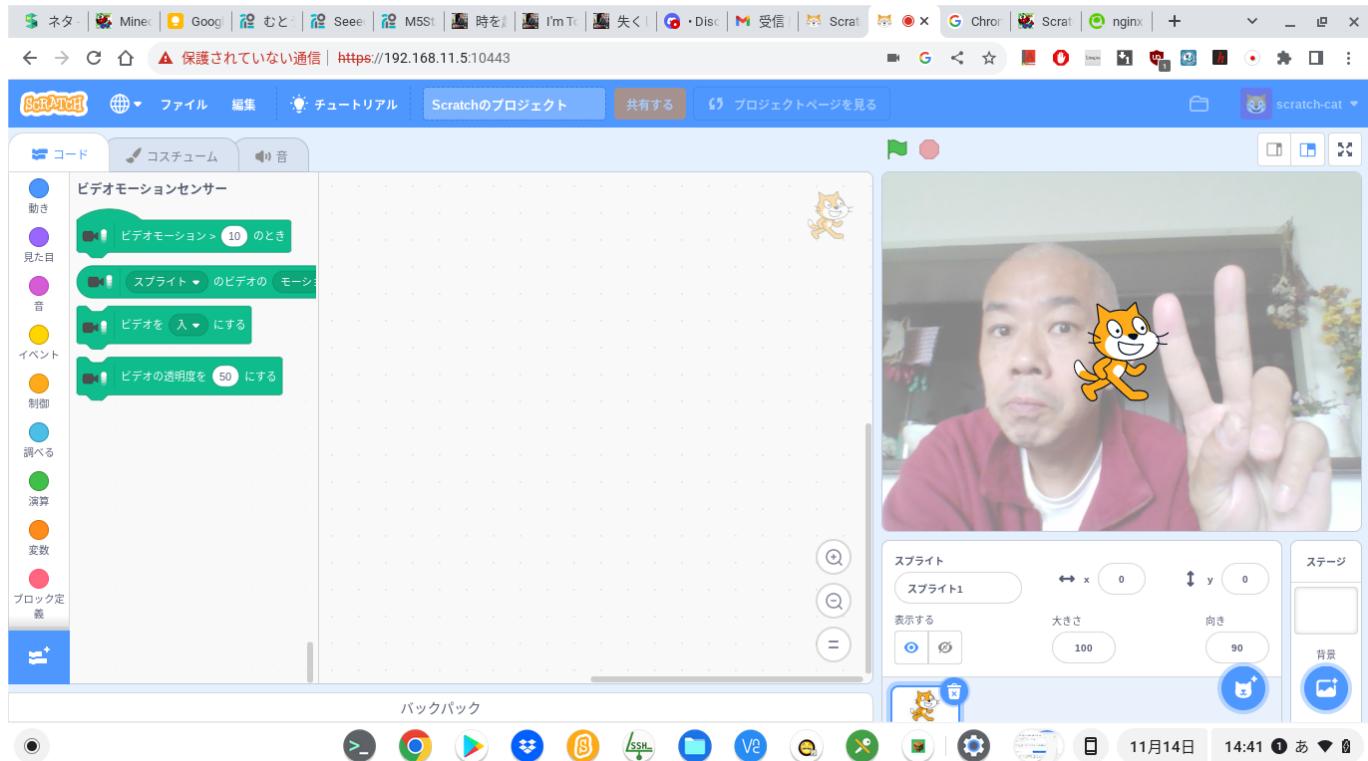
Microbit Moreを追加するためには、以下のようにします。このコマンドは、[scratch-gui](#)で行ってください。

```
# リポジトリからソースを取得する。
$ git clone --depth 1 https://github.com/microbit-more/mbit-more-v2
# おまじない
$ ln -s mbit-more-v2 microbitMore
# インストールコマンド
$ sh ./microbitMore/install-stretch3.sh
```

AkaDako拡張機能をインストールするためには、以下のようにします。

```
# AkaDako
$ git clone --depth 1 https://github.com/tfabworks/xcx-g2s
$ sh ./xcx-g2s/scripts/stretch3-install.sh
```

他のPCからこのサーバの全機能を利用する



仕様上、ブラウザでは、カメラやマイクを使う機能やWebUSBやWebBluetoothなどを使う拡張機能は、[localhost](#)ではない外部からはhttp経由ではなくhttps経由でしか使えないようになっています ([Chrome 47 WebRTC: Media Recording, Secure Origins and Proxy Handling](#))。

したがって、httpサーバー自分で用意して、httpsが使えるように設定する必要があります。

このためには、nginxやapacheなどのWebサーバーを用意して、httpsが使えるようにする、つまりSSLが使えるように設定する必要があります。SSLを利用するためには証明書が必要ですが、Let's encryptなどを使って正規の証明書を使う方法や自己署名証明書(通称、オレオレ証明書)を使う方法などがあります。

その後、SSLでの接続をScratchサーバーにProxyするための設定を行うことになります。

nginxを利用する場合の手順は、以下の通りになります。

自己署名証明書(オレオレ証明書)を作成します。

```
# オレオレ証明書を保存するディレクトリを作成する
$ mkdir -p /usr/local/etc/nginx/ssl
$ cd /usr/local/etc/nginx/ssl
# キーの生成
$ sudo openssl genrsa -out server.key 2048
$ sudo openssl req -new -key server.key -out server.csr
##### (適切に項目を埋める)
$ sudo openssl x509 -days 3650 -req -signkey server.key -in server.csr -out server.crt
```

nginxの設定ファイルを調整します。以下の例では、10443ポートでSSLを受けて、Scratchサーバーデフォルトの8601にproxyするように設定しています。

```
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;
    server {
        listen 10443 ssl;
        server_name localhost;
        ssl_certificate /usr/local/etc/nginx/ssl/server.crt;
        ssl_certificate_key /usr/local/etc/nginx/ssl/server.key;
        location / {
            proxy_pass http://localhost:8601/;
        }
    }
    include servers/*;
}
```

ここで、nginxを起動します。設定がうまく行っていれば、<https://localhost:10443/>でScratchにアクセスできるようになっているはずです。

[!NOTE] このサーバにアクセスした時、Chromeブラウザで警告が出た場合、以下のどちらかでアクセス可能になります。

- "詳細情報"ボタンを押して、出てきた"にアクセスする（安全ではありません）"リンクを押す
- "thisisunsafe"と入力する

参考文献

- [Scratchサーバーを手元で立ち上げる](#):Scratchサーバーを自分で作るための概略説明です。
- [Scratch at FreeBSD](#):実際にFreeBSDというOSでサーバーを立ち上げた例です。他のOSでも参考になるかと思います。

おわりに
