

Scratch Hacks!!

筆者:武藤武士 (X:@610t)

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、まだアルファ版で一般公開されておらず、今とはかなり雰囲気の異なる環境でした。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、私にとっては大好物です。

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
- ChatGPT
- [TBD]TeachableMachine
- [TBD]音声認識
- [TBD]画像分類器
- [TBD]ポーズ認識

みんなの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。

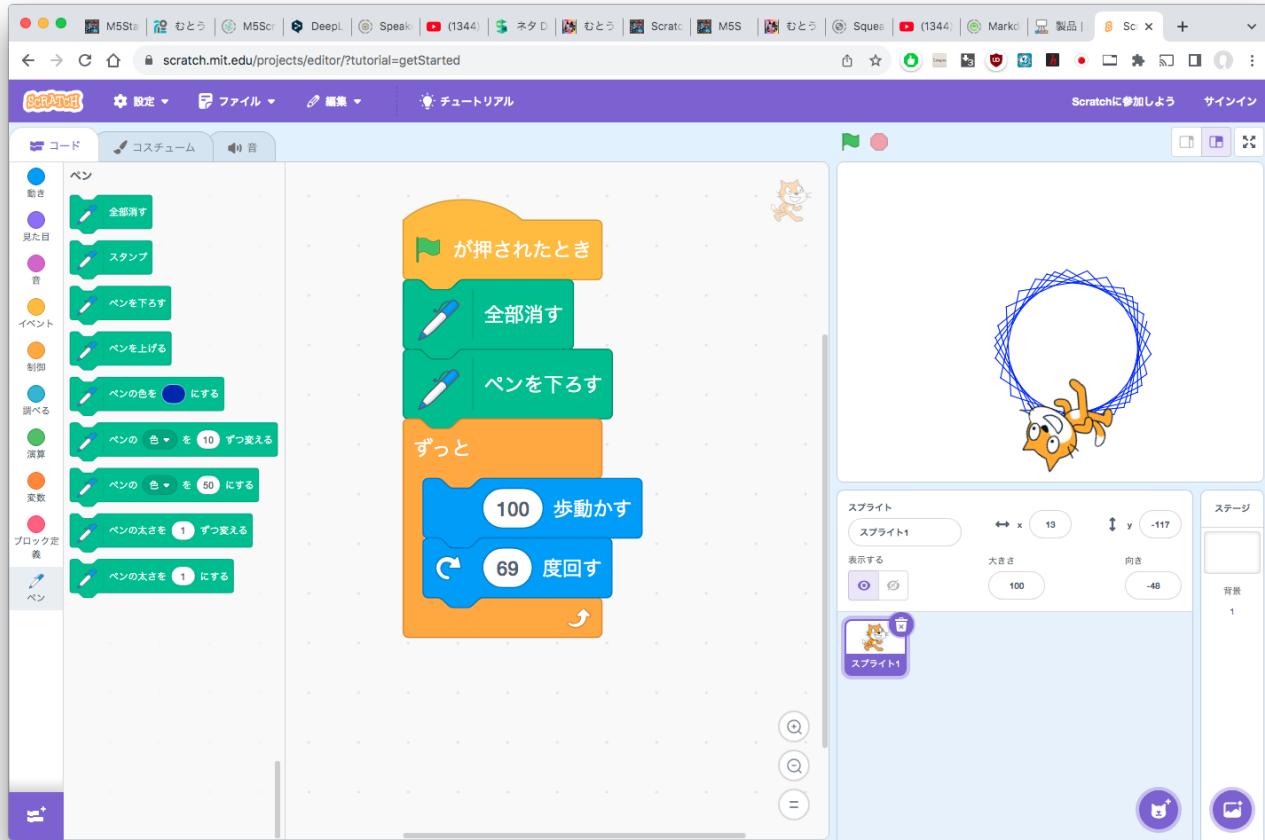
Happy Scratch Hacking!!

目次

- はじめに
- 目次
- Scratchってなあに?
 - Scratch拡張機能
 - 独自の拡張機能を追加したScratch拡張サーバー
- Stretch3(ストレッチスリー)
 - Microbit More拡張機能
 - AkaDako拡張機能
 - 参考文献
 - 音声認識拡張機能(Speech2Scratch)
 - IFTTT Webhooks拡張機能
 - IFTTT側での設定
 - Scratchでのプログラミング
 - デモ: 侵入検知器
 - デモ: 音声入力翻訳システム
 - デモ: りんごを食べようゲーム
 - ChatGPT拡張機能(CHATGPT2Scratch)
 - ポーズ認識に関する拡張機能
 - Posenet2Scratch : 姿勢認識
 - Handpose2Scratch:手の姿勢認識
 - Facemesh2Scratch : 顔のメッシュ分割
 - 地図で遊ぼう!!: Geo Scratch
 - Geo Scratchを使った例
 - Get Scratchで作るナビシステム
 - TeachableMachineに関する拡張機能
 - TMpose2Scratch
 - TM2Scratch
 - ML2Scratch
 - 画像分類器拡張機能(ImageClassifier2Scratch)
- M5Stackと遊ぼう!!
 - M5Stackってなあに?
 - M5StackとScratchで遊ぶための参考文献
- M5bitLess = M5Stack x Scratch:M5StackでMicrobit Moreを使う
 - 参考文献
 - その他のマイコンボードで*bitLess系列を使う
 - 参考文献
- Scratch1.4と遠隔センサープロトコル
 - Scratch 1.4
 - 遠隔センサープロトコル(RSP)
 - AndroidでのRSPの利用
 - Scratch x M5Stack UIFlow
- M5Scratch = Scratch x M5Stack:M5StackでScratch遠隔センサーを使う
 - 参考文献

- スーパーかわいいロボットスッタックチャン
 - 音声入出力を使ってスッタックチャンと遊ぼう
 - 参考文献
 - M5bitLessを使ってスッタックチャンと遊ぼう
- Scratchサーバー自分で立ち上げる
 - 拡張機能を追加する
 - 他のPCからこのサーバの全機能を利用する
 - 参考文献
- 拡張機能の作成
 - 通常の拡張機能を作成する
 - xcratch用拡張機能
 - xcratchを利用する
 - xcratchの拡張機能を作成する
 - 例: ambient用の拡張機能
 - ambient拡張機能の使い方
 - ambient拡張機能の解説
 - 参考文献
- おわりに
- 奥付

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つていきます。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。

独自の拡張機能を追加したScratch拡張サーバー

Scratchの拡張機能を独自に追加したサーバー(以下、拡張サーバー)も各所で提供されています。

例えば、以下のような拡張サーバーがあります。

- [Stretch3](#):AI系に強い
- [Xcratch](#): 拡張機能動的追加型
- [blicklife](#): LEGOに強い
- [toioDo](#): Sony 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

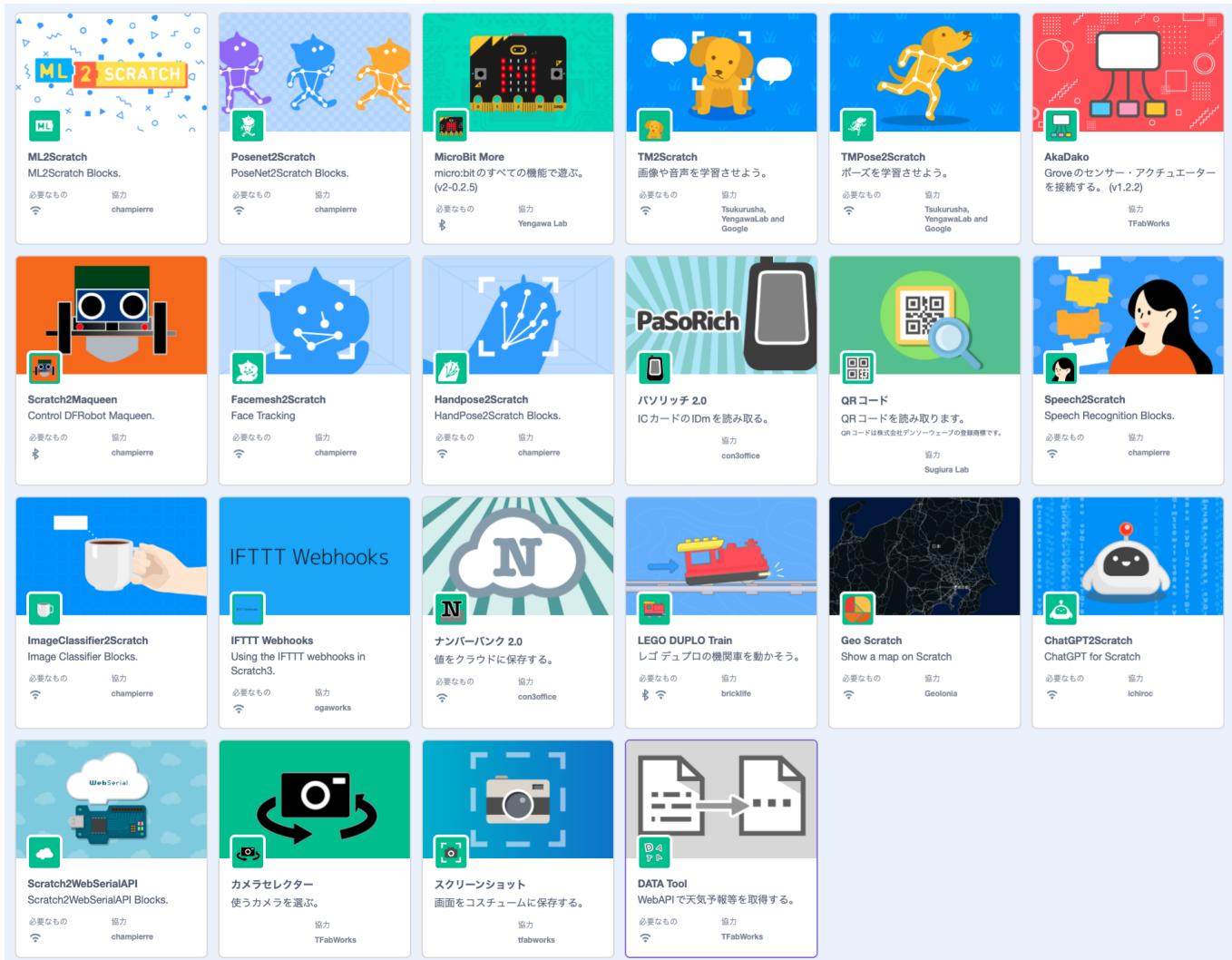
本書では、Stretch3を使っています。

[!CAUTION] 拡張サーバーでは、クラウドへのコード保存ができませんので、明示的に自分でダウンロードして保存する必要があります。

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
- ChatGPT(ChatGPT2Scratch)
- [TBD]TeachableMachine(TM2Scratch, TMPose2Scratch)
- [TBD]音声認識(Speech2Scratch)
- [TBD]画像分類器(ImageClassifier2Scratch)
- [TBD]ポーズ認識(Posenet2Scratch, Facemesh2Scratch, Handpose2Scratch)

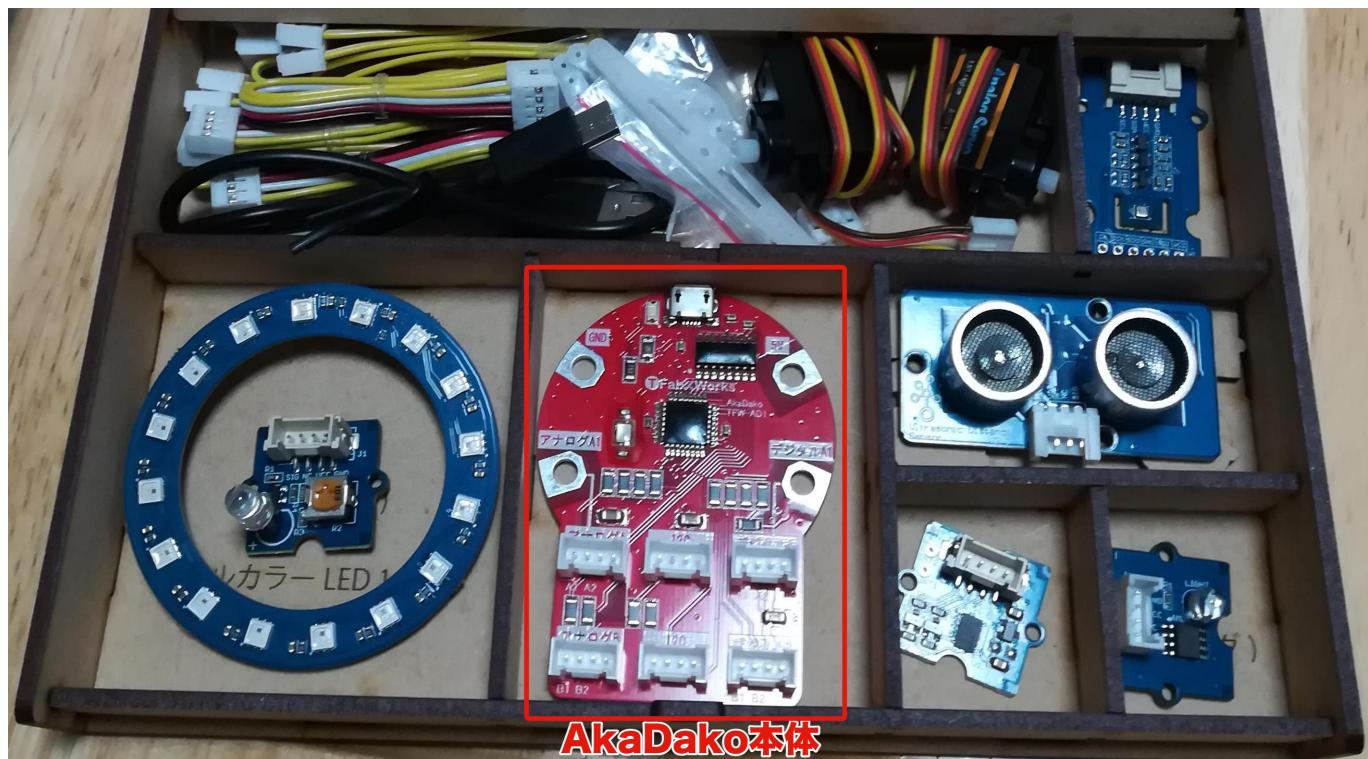
Microbit More拡張機能

[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	○	○
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。 詳細は、[M5bitLess = M5Stack x Scratch:M5StackでMicrobit Moreを使う](#)でご紹介します。

AkaDako拡張機能



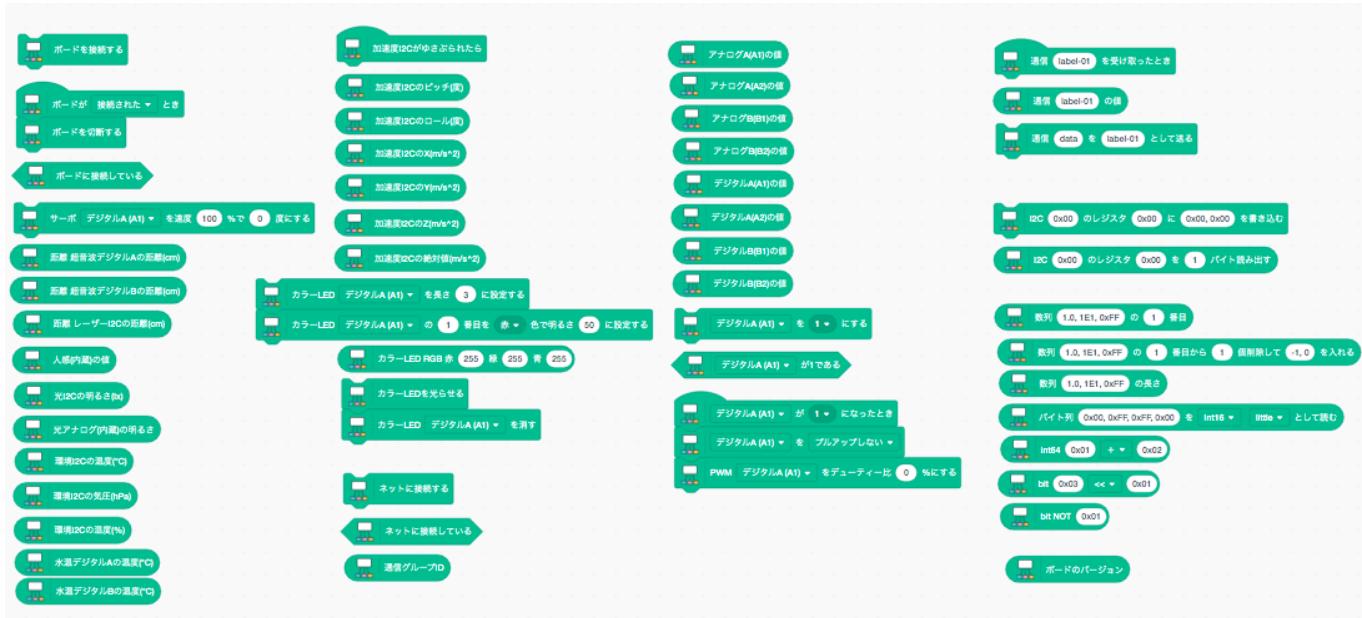
AkaDakoは[TFabWorks](#)によって開発および販売されている、Scratch用のGroveシールドです。USBで有線接続するため、Bluetoothなどの無線を使った接続のものより、安定して利用することができます。教育向け用途を志向しており、各種学習単元に沿った実験ができるようになっています。

AkaDako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エ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3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、2つのアナログ入力端子と、2つのデジタル入力端子がGrove以外にも用意されています。更に、初めから光センサーや加速度計や距離などのセンサーや、サーボモータやLEDアレイなどのアクチュエーターがブロックで簡単に利用可能になっています。

AkaDakoで利用可能なブロックは、以下の図のようになります。



標準で多くのセンサーやアクチュエーターに対応しています。もちろん、オンボードでこれらが用意されているものは、すぐに利用可能です。

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、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ってみました。

音声認識拡張機能(Speech2Scratch)

音声認識拡張機能(Speech2Scratch)は、その名前のとおりで、マイクから入力した音声をテキストに変換してくれる機能です。 Scratchにはテキストを翻訳したり、音声で読み上げたりする機能があるので、組み合わせると面白い作品が作れます。

ブロックは2種類しかありません。



少し使い方にコツがいるので、動作例を紹介します。はじめに、[音声認識開始]ブロックで音声認識をはじめます。音声が認識されるまでは、[音声]ブロックには何も文字列が入らない空文字列になるので、文字列が入るまで待機します。文字列が入ったら、その文字列に対して処理を行うという手順になります。



[音声入出力を使ってスクリプチャンと遊ぼう](#)では、音声認識を使ってスクリプチャンと遊ぶ例もご紹介します。

IFTTT Webhooks拡張機能

IFTTTは、IoT(Internet of Thinks)でよく使われるサービスです。IoTとは、なんでもインターネットにつないで、何かをしようということです。ここで、つなぐものは後で出てくるM5Stackのようなマイコンだったり、スマートセンサー類だったりします。

IFTTTでは、その名のとおり、IFに相当する特定のイベント(トリガー)が発生した時に、THENに相当する色々なサービス(アクション)を行なうというような動作をします。サービスの部分では、例えばLINEや電子メールなどにメッセージを送ったり、スプレッドシートに書き込んだりなど色々なことができます。

[IFTTT Webhooks拡張機能](#)はScratchからIFTTTのWebhook機能を利用するための拡張機能です。トリガーとしてWebhookを利用しておらず、3つまでの値が送れるようになっています。

[!NOTE] 現在、IFTTTの無料サービスでは、Webhookは利用できません。有料プランを利用するか、無料トライアルを利用する必要があります。

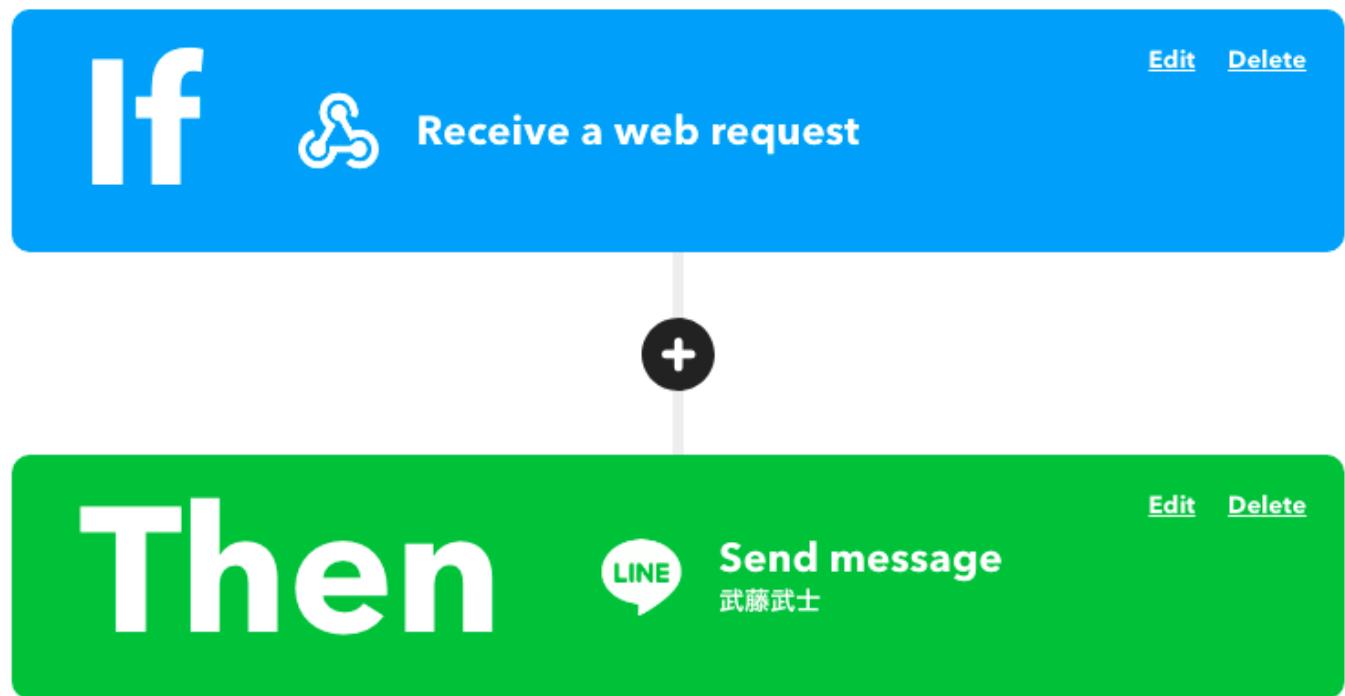
IFTTT側での設定

今回のデモでは、すべてトリガーとしてWebhookを使い、アクションとしてはLINE Notifyを使います。ここでは、その設定に関して説明していきます。

アプレットの構成は、以下の図のようになります。

Edit Applet

Want to publish this Applet so anyone can use it? [Click here ↗](#)



はじめに、トリガーであるWebhookを設定します。後で使うキーは、WebhookのURLの末尾に記述されています。

Edit trigger fields



Receive a web request

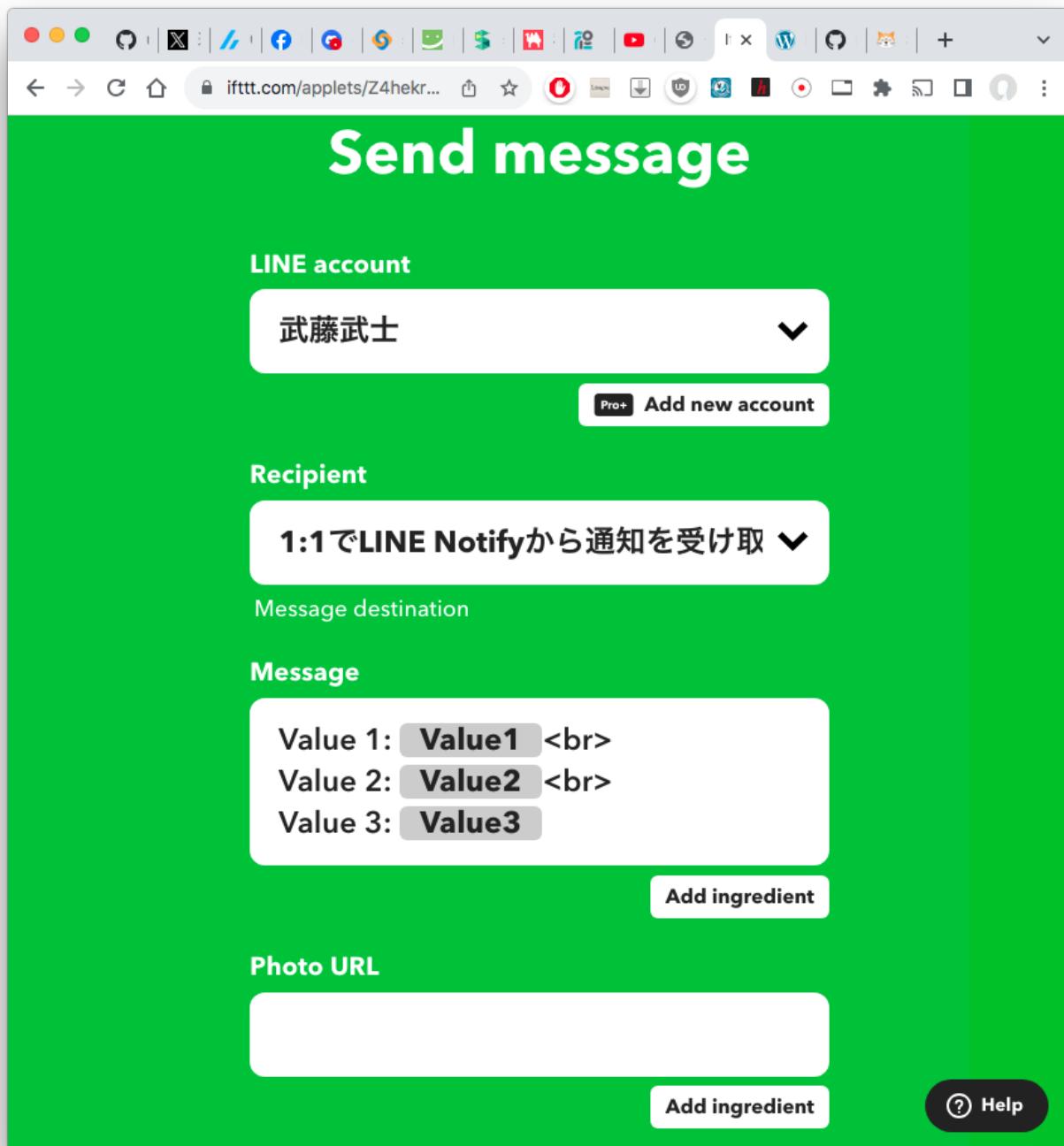
Event Name

Stretch3

The name of the event, like "button_pressed" or "front_door_opened". Use only letters, numbers, and underscores

Update trigger

次にアクションであるLINE Notifyの設定です。 LINEとの接続部分に関しては省略しますが、指示どおりに設定していけば大丈夫です。



Scratchでのプログラミング

IFTTT Webhooks拡張機能には、以下のようなブロックがあります。



使い方は、以下のような流れになります。

- IFTTT Webhookアプレットを利用する場合keyが必要なので、これを与える必要があります([IFTTT key:(key)] ブロック)。keyはWebhook URLの最後にある文字列です。
- イベント名でどのアプレットが使われるか指定するので、これを適切に設定する必要があります([IFTTT event:(event_name)] ブロック)。値は3つまで利用でき、それぞれvalue1, value2, value3([value1を(value1)にする]などのブロック)が利用できます。
- 最後に [送る] ブロックでデータが送られます。

以下、Scratchでのプログラムの例を示します。

デモ: 侵入検知器

The screenshot shows a LINE Notify message from IFTTT with several triggers and their values, followed by a Scratch script.

LINE Notify

[IFTTT] Value 1: 敵機発見!!
Value 2: 100
Value 3: 175.5052103451511

[IFTTT] Value 1: 敵機発見!!
Value 2: 100
Value 3: 153.2287753548061

[IFTTT] Value 1: 敵機発見!!
Value 2: 100
Value 3: 108.88419362966084

[IFTTT] Value 1: 敵機発見!!
Value 2: 91
Value 3: -81.31230964159789

午前 10:19

Scratch Script:

```
when green flag clicked
  [flag was pressed v] do
    video set [入 v]
    video motion > [90 v] do
      value1 set [敵機発見!! v]
      value2 set [スプライト v] of [ビデオ的运动 v]
      value3 set [スプライト v] of [ビデオ的向き v]
      send [送る v] to [all v]
```

ビデオモーションセンサーを使って、動きを検知し、動きが大きい時には侵入されたと考えて、LINE Notifyで通知を行います。

デモ： 音声入力翻訳システム



The screenshot shows a mobile application interface for IFTTT. On the left, there are three notifications from the 'IFTTT' channel:

- [IFTTT] Value 1: 翻訳してみます
Value 2: 我会试着翻译的
Value 3: I'll try to translate
- [IFTTT] Value 1: これはどうでしょうか
Value 2: 这个怎么样
Value 3: How about this
- [IFTTT] Value 1: 私の名前は 武藤 剛 です
Value 2: 我叫武藤刚
Value 3: My name is Tsuyoshi Mutou

On the right, the IFTTT recipe is displayed:

```
が押されたとき
IFTTT event : Stretch3
音声認識開始
音声 = ではない まで待つ
text を 音声 にする
value1 を text にする
value2 を text を 中国語（簡体）に翻訳する にする
value3 を text を 英語 に翻訳する にする
送る
```

The recipe triggers when a button is pressed (Stretch3 event). It starts voice recognition and waits until the sound is not (is not) detected. It then converts text to speech, sets value1 to text, translates value2 to Chinese (Simplified), translates value3 to English, and finally sends the message.

音声入力した言葉をテキストに変換し、そのテキストを翻訳してLINE Notifyで送ります。

デモ: りんごを食べようゲーム



Stretch3のMicrobit Moreを使って、猫を加速度センサーで動かし、りんごを捕まえましょう。りんごを捕まえると、LINE Notifyで通知が送られます。

今回は、後述のM5bitLessを使って、M5Stackの加速度センサーを使いましたが、もちろんmicro:bitでも大丈夫です。

ChatGPT拡張機能(CHATGPT2Scratch)

ChatGPT拡張機能(CHATGPT2Scratch)は、Scratchから話題のAIであるChatGPTを使うことのできる拡張機能です。

CHATGPT2Scratchのブロックは6種類ですが、実際に利用する時は2種類のブロックだけで十分です。

CHATGPT2Scratch



ChatGPTを利用するためには、アカウント作成やAPIキーの取得、利用料金支払いのためのクレジットカード情報の入力などかなり手間がかかります。今のところ、この手順に関しては本書では解説しません。ここでは、既にこれらの準備が終了しており、APIキーが取得できていることを前提に話を進めます。

ChatGPTのアカウント作成には、13歳以上であることが必要なので、それに満たない人は保護者の方と一緒に使ってください。また、クレジットカードの情報も必要なので、こちらも保護者の方にご相談してください。

CHATGPT2Scratchの[APIキーをセット]ブロックが実行されると、図のようにAPIキーの入力がうながされます。ここに、取得したAPIキーを入力してあげます。



利用は簡単で(())の答え)ブロックにChatGPTに問い合わせしたいテキストを入力するだけです。

ChatGPTは、[音声入出力を使ってスクリプトと遊ぼう](#)や[M5bitLessを使ってスクリプトと遊ぼう](#)でも利用します。

ポーズ認識に関する拡張機能

ここでは、Stretch3で利用可能なポーズ認識の拡張機能について紹介します。ポーズ認識は、例えば鼻や口などの位置を認識して、その場所を返してくれるような機能になります。

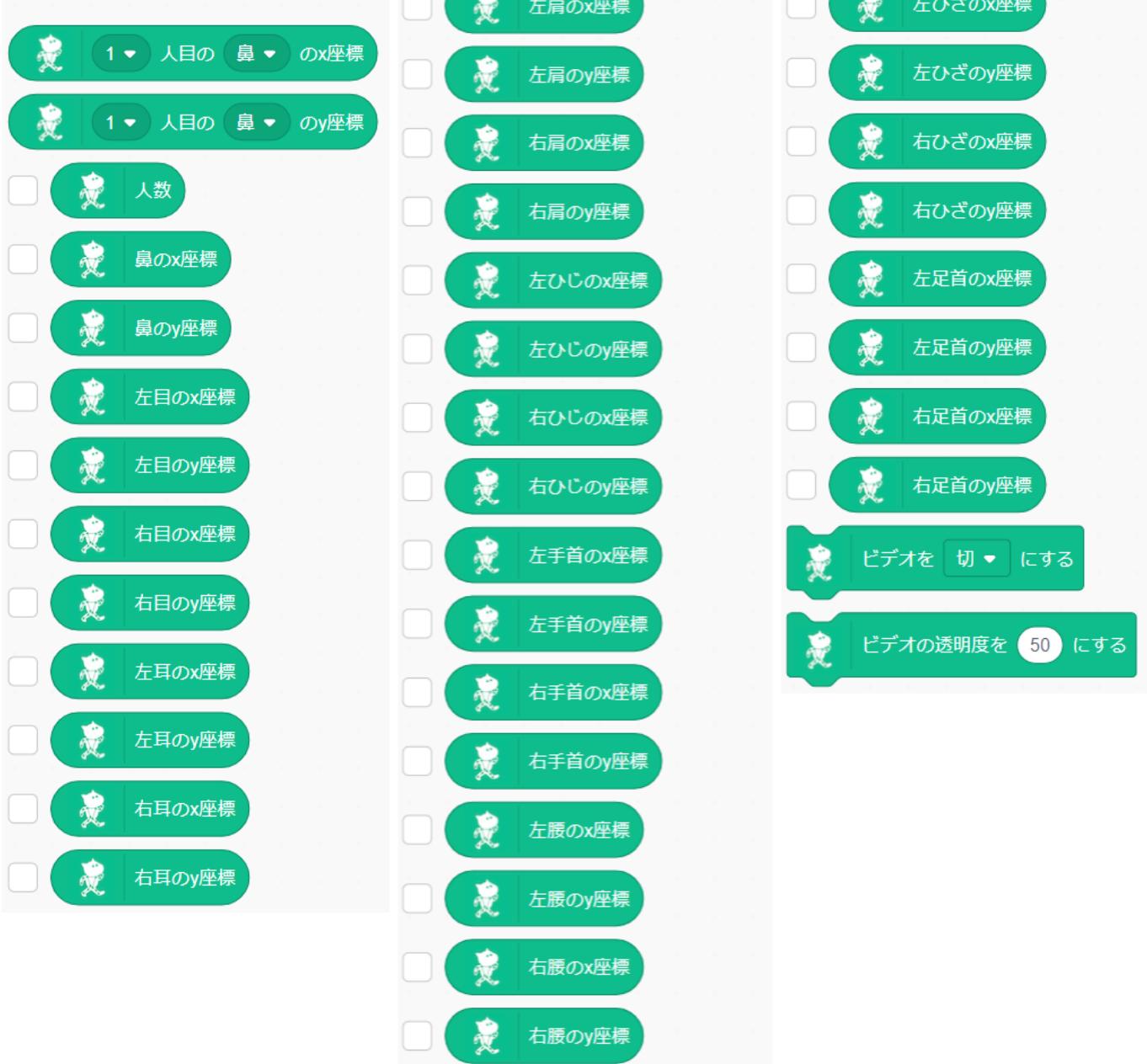
Posenet2Scratch : 姿勢認識

Posenet2Scratchは、ScratchでWebカメラの映像から体の姿勢を見つけて、その顔を部品ごとに追跡できる拡張機能です。姿勢認識には、[PoseNet](#)が利用されています。

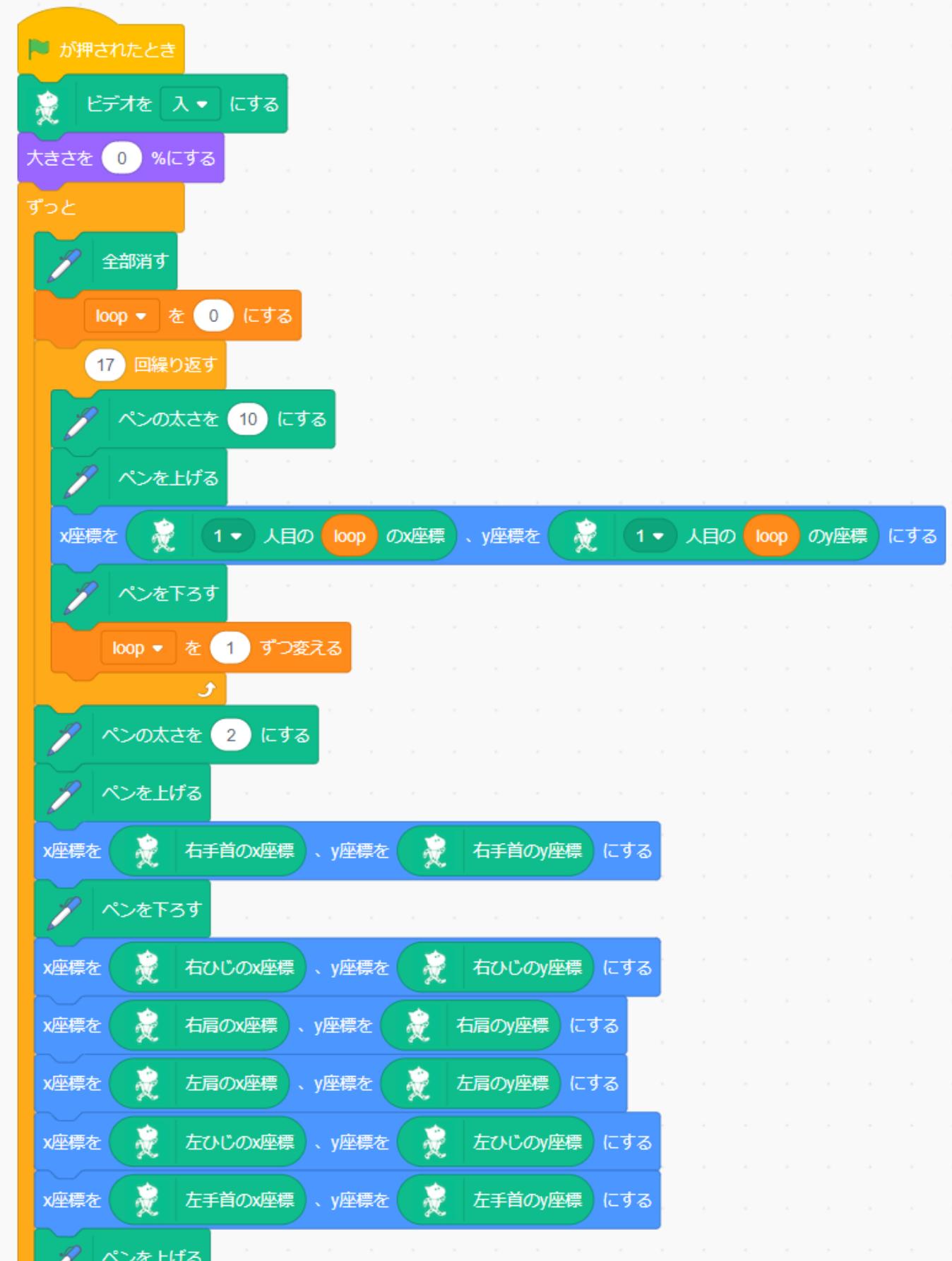
Posenet2Scratchでは、以下のブロックのように部品ごとに位置がわかるようになっているため、あとで紹介するFacemesh2Scratchより用途によっては使いやすいと思います。

Posenet2Scratchで提供されているブロックは以下のとおりです。

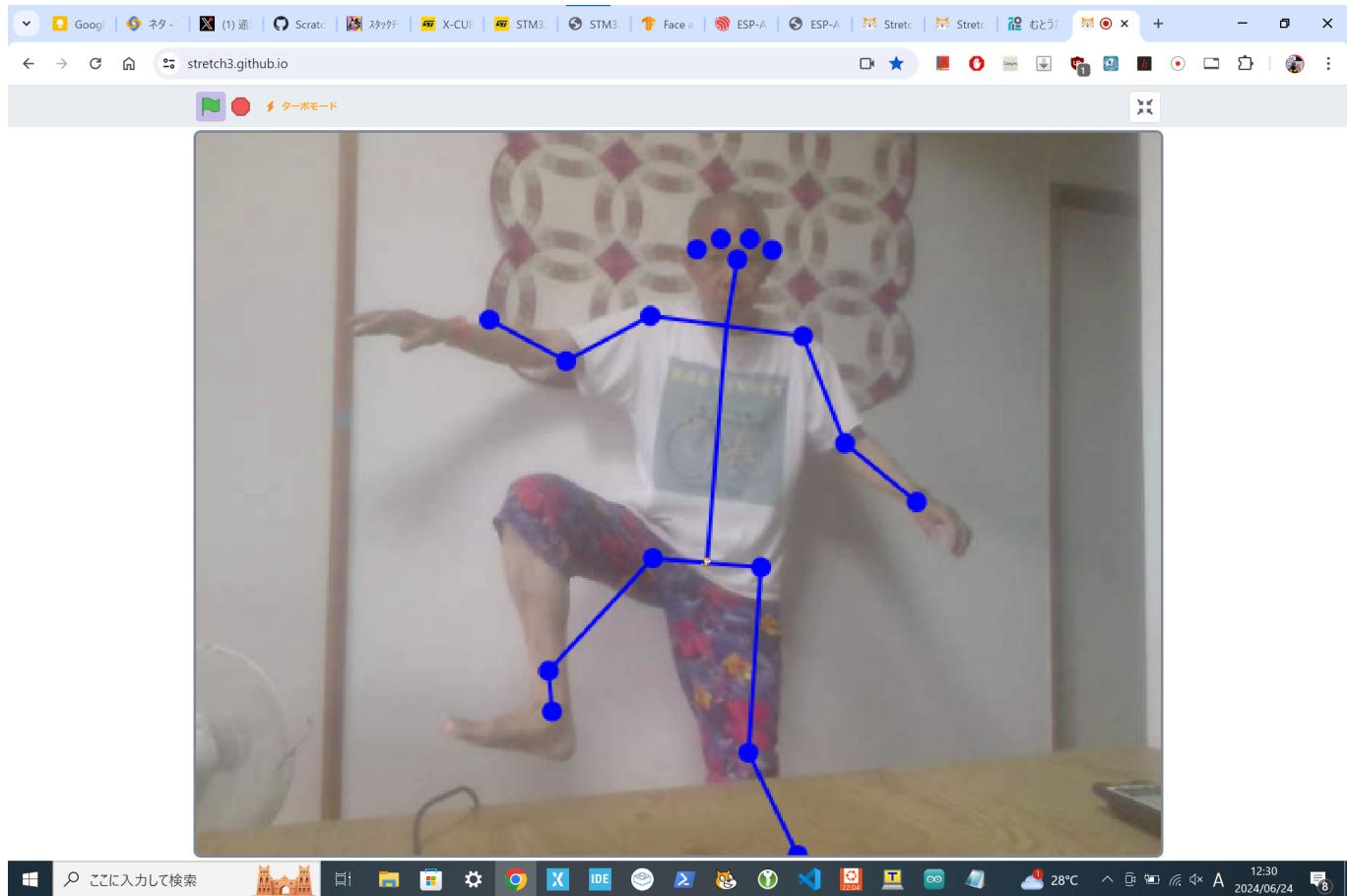
Posenet2Scratch



以下のようなコードを書くことで、一般的にスティック線図と呼ばれる姿勢が良くわかる図を描くことができます。コードは図ではスティック線図を描く部分を一部を省略していますが、実際のコード全体は [examples/Posenet2Scratch_demo.sb3](#) として提供しています。通常のモードでは遅いため、ターボモードを利用することをお勧めします。



デモでは、以下のように体の姿勢がスティック線図として表示されます。



Handpose2Scratch:手の姿勢認識

[Handpose2Scratch](#)は、ScratchでWebカメラの映像から手の姿勢を見つけて、その手を部品ごとに追跡できる拡張機能です。

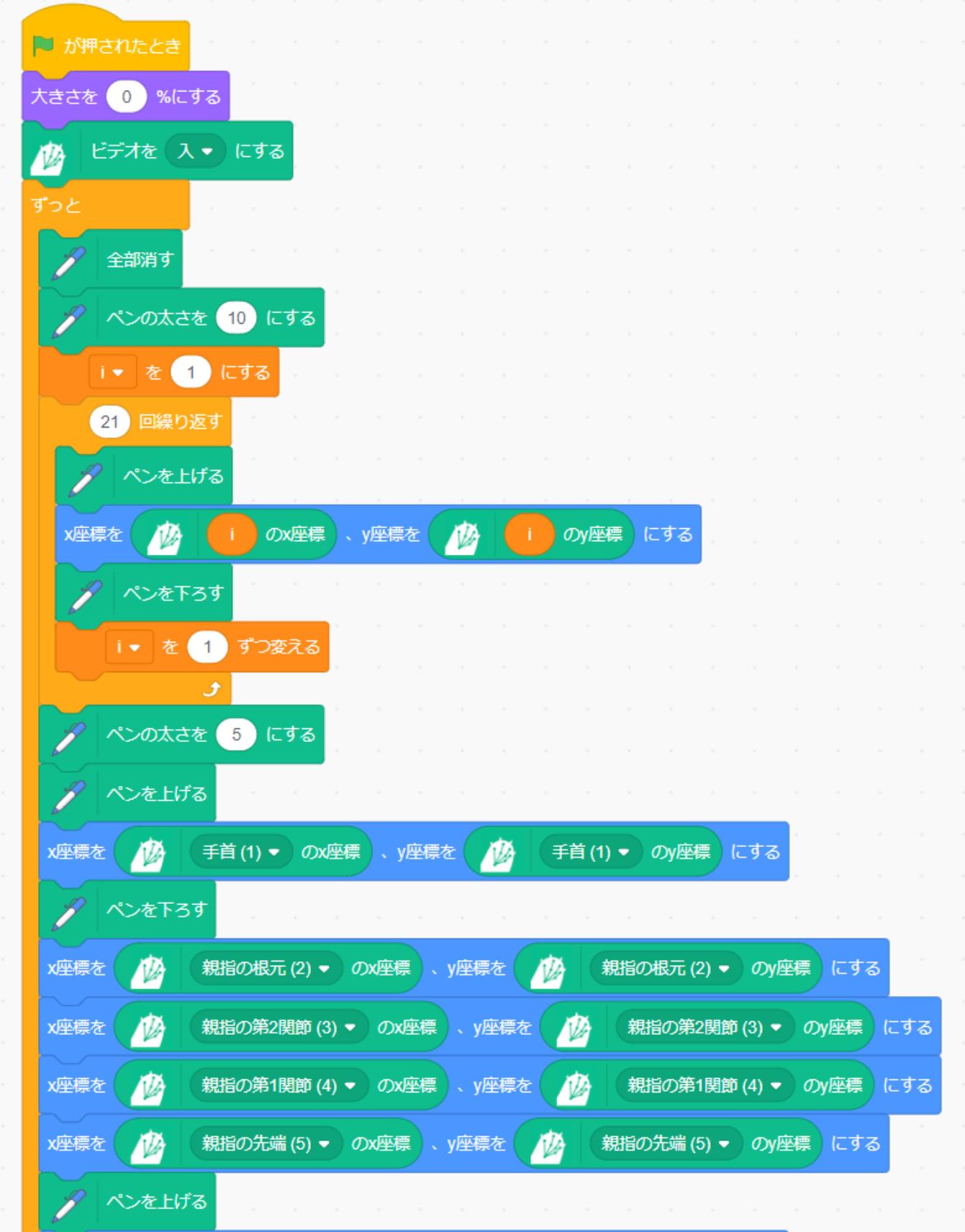
技術に関する詳細は、[Face and hand tracking in the browser with MediaPipe and TensorFlow.js](#)に記述されています。

Handpose2Scratchで提供されるブロックは以下のとおりです。 (手首(1))となっている部分は場所を表し、1-21で"小指の先端(21)"などの場所を表しています。ここには、数字のブロックを入れることも可能で、全ての点を描画したい時などに利用することができます。

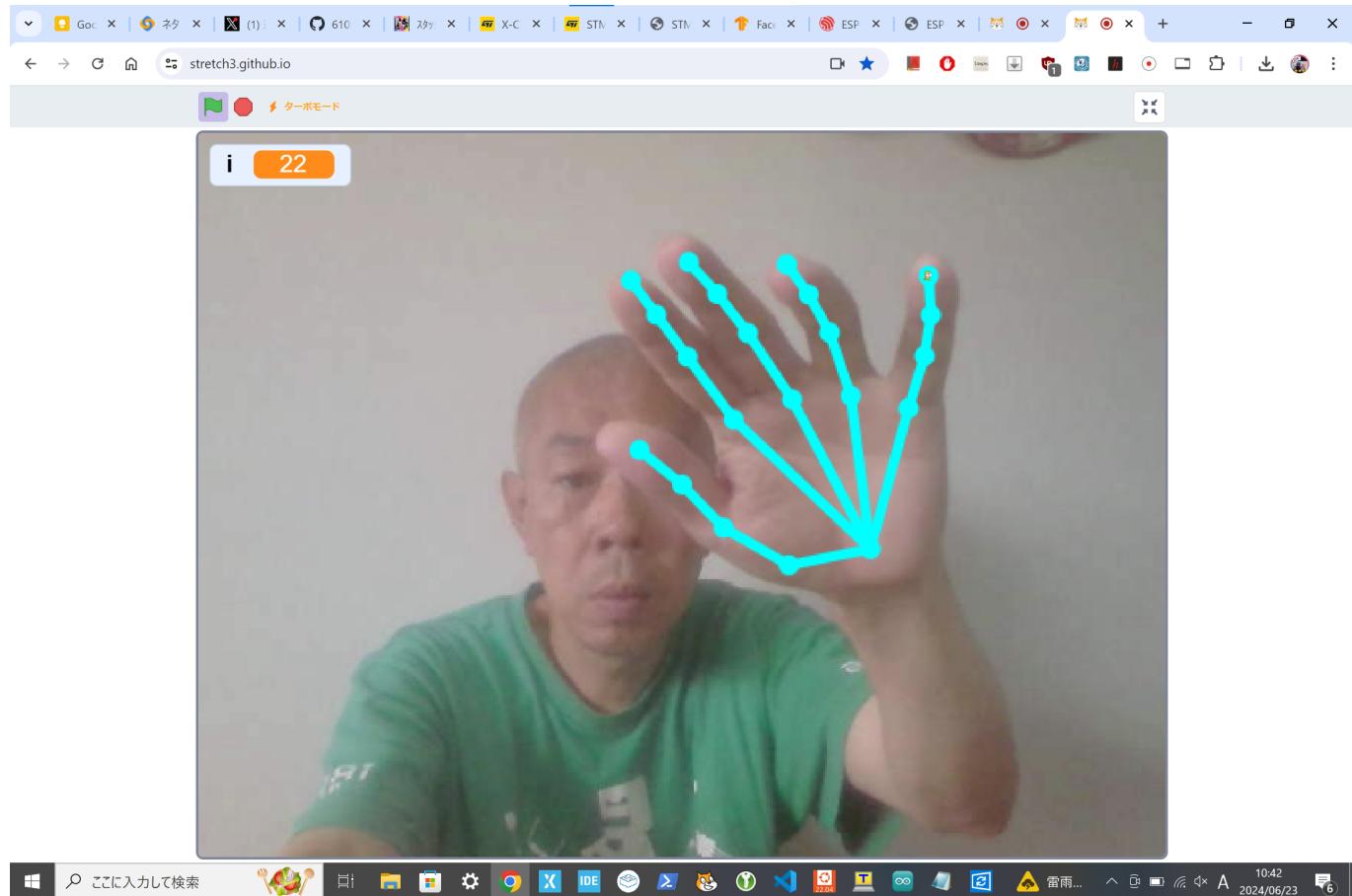
Handpose2Scratch



デモのコードは以下のとおりです。このデモでは、手の姿勢を認識して、スティック線図を描きます。コードは図ではスティック線図を描く部分を一部を省略していますが、実際のコード全体は [examples/Handpose2Scratch_demo.sb3](#) として提供しています。通常のモードでは遅いため、ターボモードを利用することをお勧めします。



デモでは、以下のように手の姿勢がスティック線図として表示されます。



Facemesh2Scratch : 顔のメッシュ分割

[Facemesh2Scratch](#)は、ScratchでWebカメラの映像から顔を見つけて、その顔を部品ごとに追跡できる拡張機能です。

顔が468点のメッシュに分割されその位置がわかるようになっています。複数人数にも対応しており、人ごとにメッシュが作成されます。

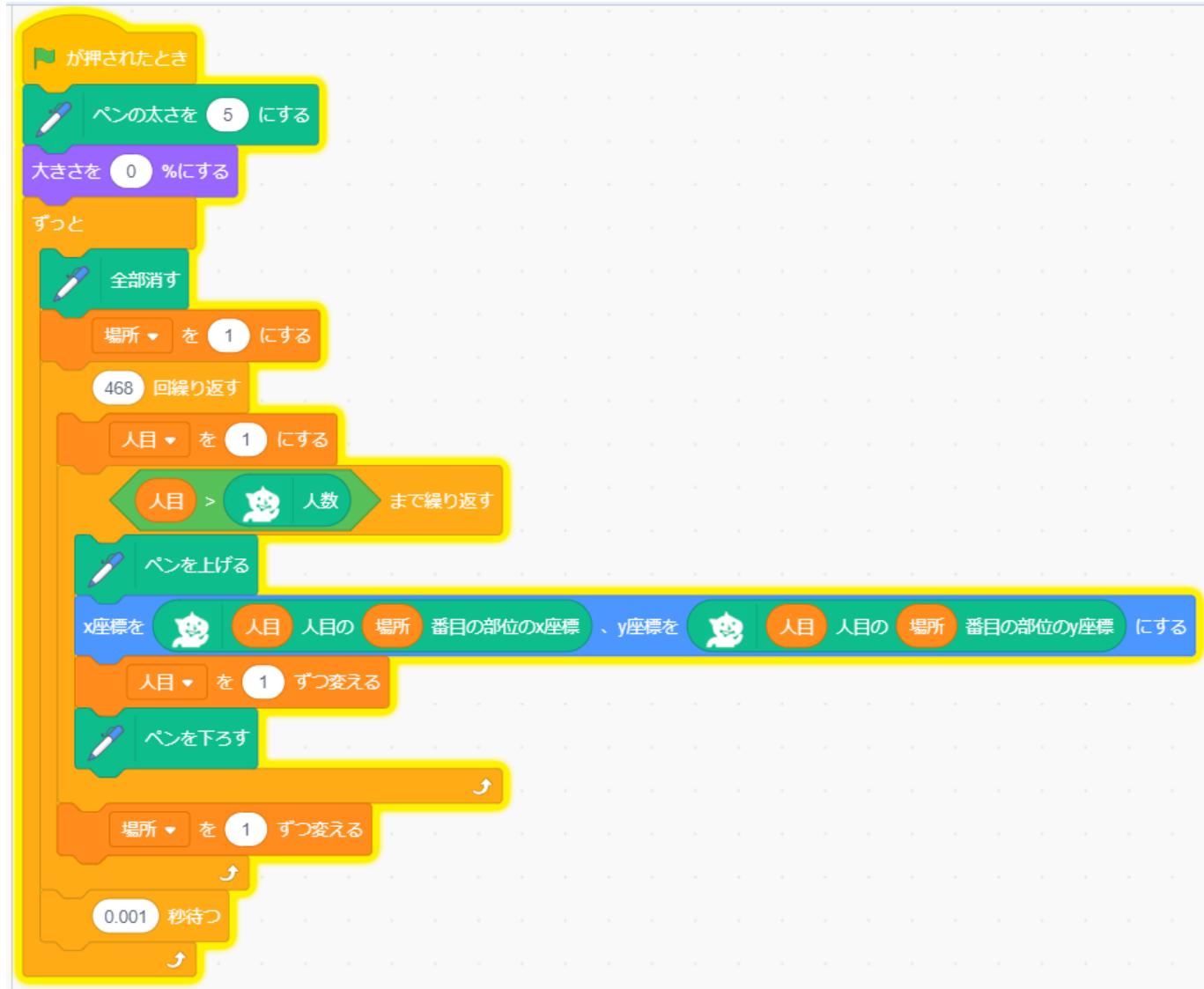
技術に関する詳細は、[Face and hand tracking in the browser with MediaPipe and TensorFlow.js](#) に記述されています。

Facemesh2Scratchで提供されているブロックは以下の通りです。ほとんどの場合、((1)人目の(1)番目の部位のx座標)と((1)人目の(1)番目の部位のy座標)を使って特定の部位の位置を検出するのに使うことになると思います。複数人数での応用では、人目部分で人を区別することができます。"(1)人目"や"(1)番目"部分には、数字のブロックを入れることも可能で、全ての点を描画したい時などに利用することができます。

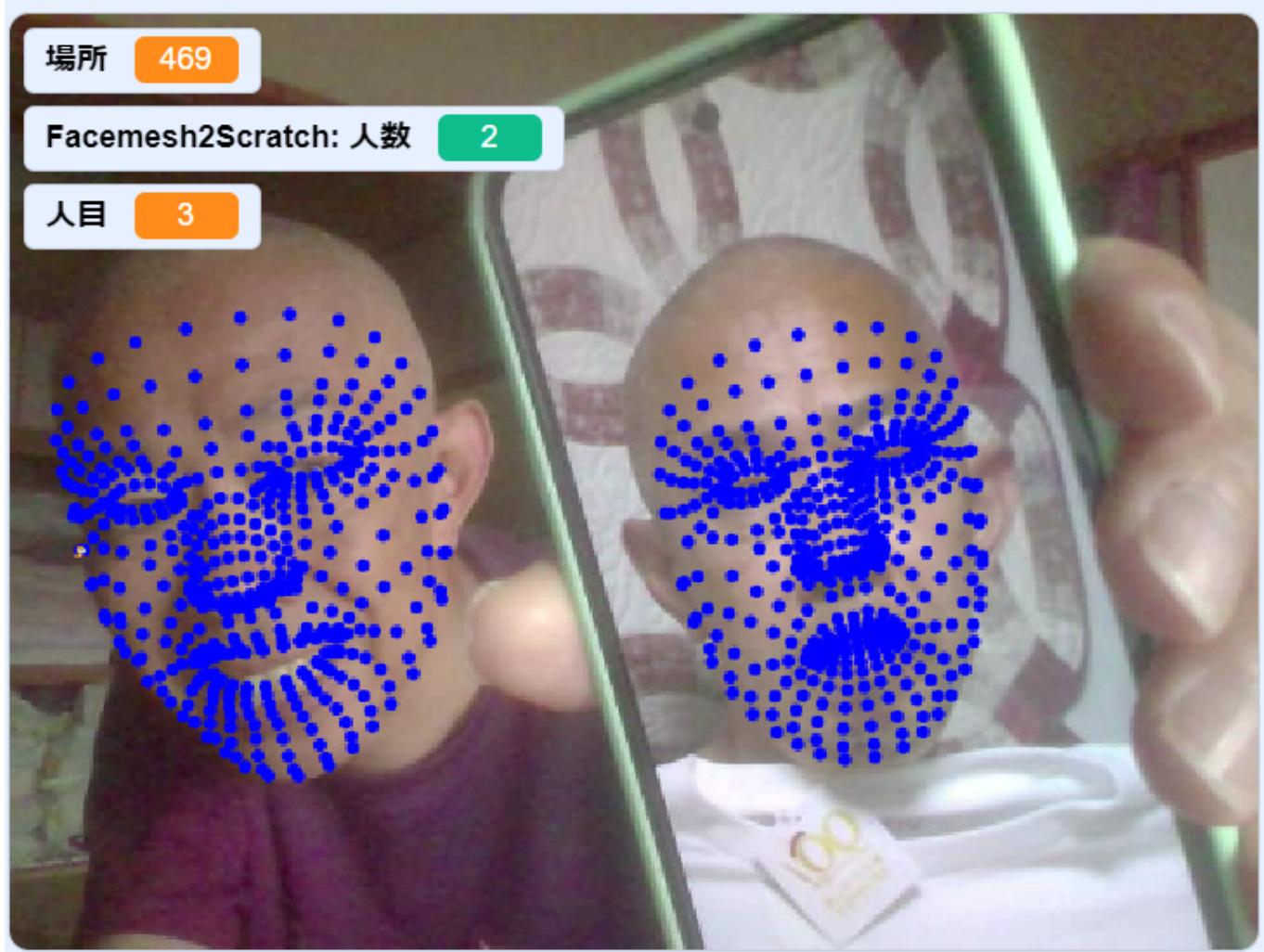
ちなみに、1番目の場所は口の上部にあたるので、(x座標を()y座標を()にする)と使うことで、スプライトが口を追いかけるコードが簡単に書けます。



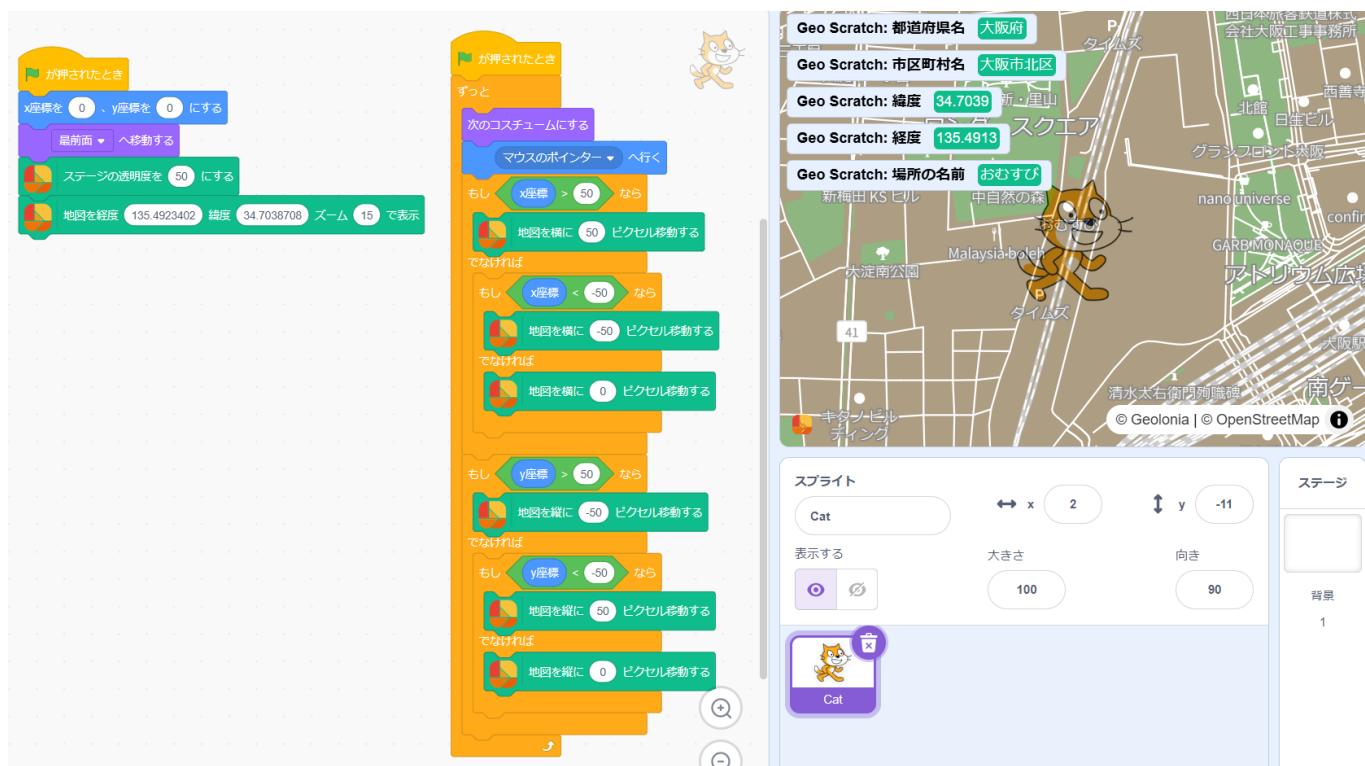
以下のようなコードを書くことで、提供されているメッシュのすべての点を表示することができます。実際のコード全体は[examples/Facemesh2Scratch_demo.sb3](#)として提供しています。通常のモードでは遅いため、ターボモードを利用することをお勧めします。



このデモコードで以下のように、顔がメッシュに分けられた状態で表示されます。



地図で遊ぼう!!: Geo Scratch



Geo Scratch拡張機能は、ScratchからOpenStreetMapの地図を簡単に扱えるようにした拡張機能です。地図を表示し、動かしたり、回転したりができます。更に、都道府県名や市町村、場所の名前、現在の緯度、経

度などの情報が取得可能です。

Geo Scratchで提供されるブロックは、以下の通りです。

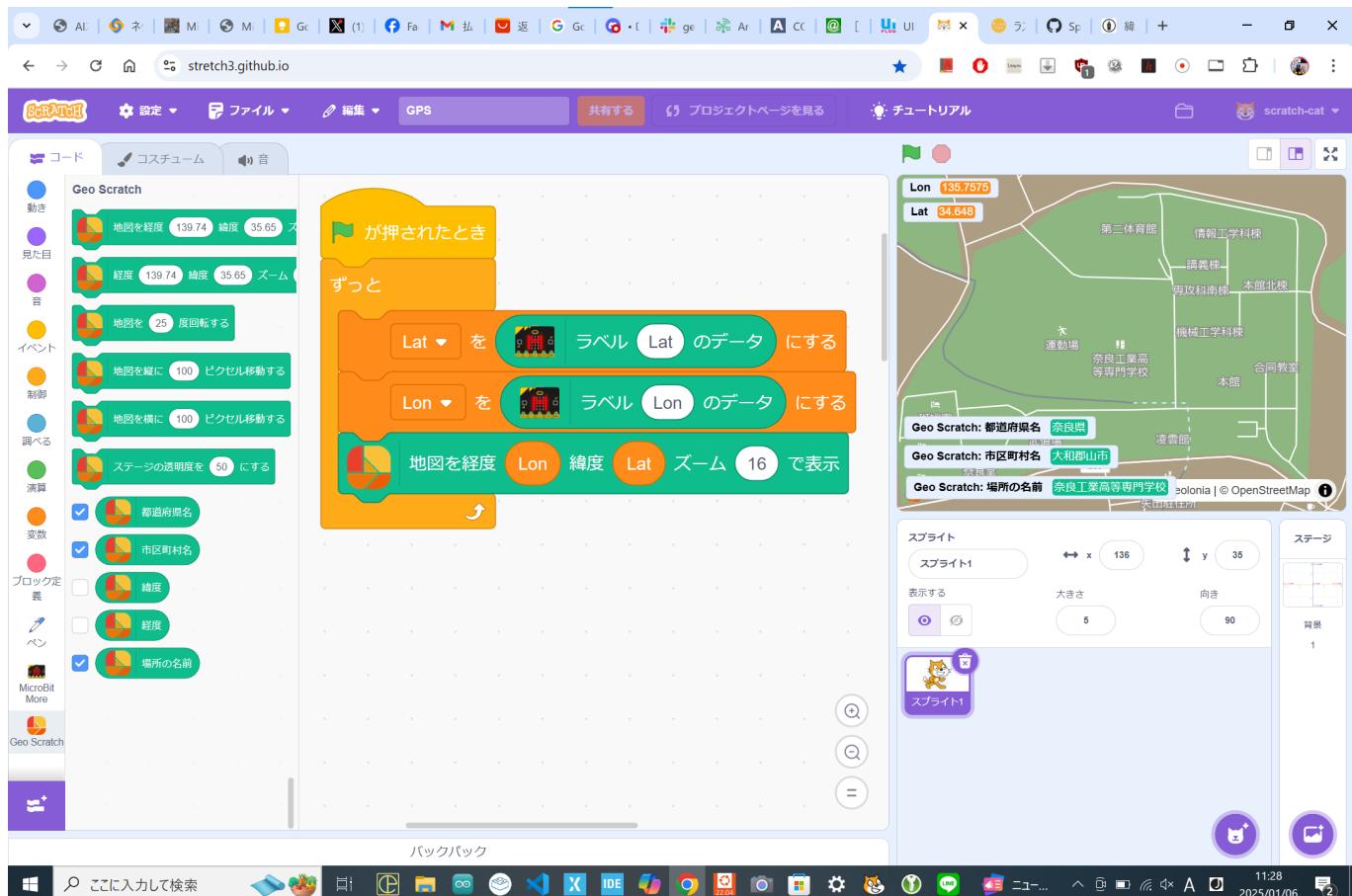


Geo Scratchを使った例

ここでは、利用例を二つご紹介します。ともに、なんらかの操作で地図をスクロール表示します。最初の例では、zとxキーで回転表示もできます。

- キーボードの入力で地図をスクロール表示: [examples/GeoScratchKey.sb3](#)
- マウスの操作で地図をスクロール: [examples/GeoScratch.sb3](#)

Get Scratchで作るナビシステム



何らかの方法で、GPSのデータを取得できる場合、簡単にナビシステムを構築することができます。上の図のコードでは、経度と緯度がそれぞれ変数`Lon`と`Lat`に入っていることを仮定しています。この状態で、[地図を経度(Lon)緯度(Lat)ズーム(16)で表示]ブロックを使うことで、その場所に地図を移動させています。

例えば、[SPRESENSE](#)というマイコンのGPS機能を使って、このようなシステムを作った例は [SpreM5ScratchSense: SPRESENSE x M5Stack x Scratch](#)として紹介していますので、興味のある方はご覧ください。

TeachableMachineに関する拡張機能

TMpose2Scratch

TM2Scratch

ML2Scratch

画像分類器拡張機能(ImageClassifier2Scratch)

M5Stackと遊ぼう!!

本章では、[M5Stack](#)というマイコンについて解説していきます。

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です。組み込み用途で必要になってくる、ディスプレイやボタン、筐体などがあるため、大変扱いやすいです。毎週新製品が発表され、活発に動きがあるのも楽しいところです。

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](#)), 240x135TFT([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), 120mAh+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

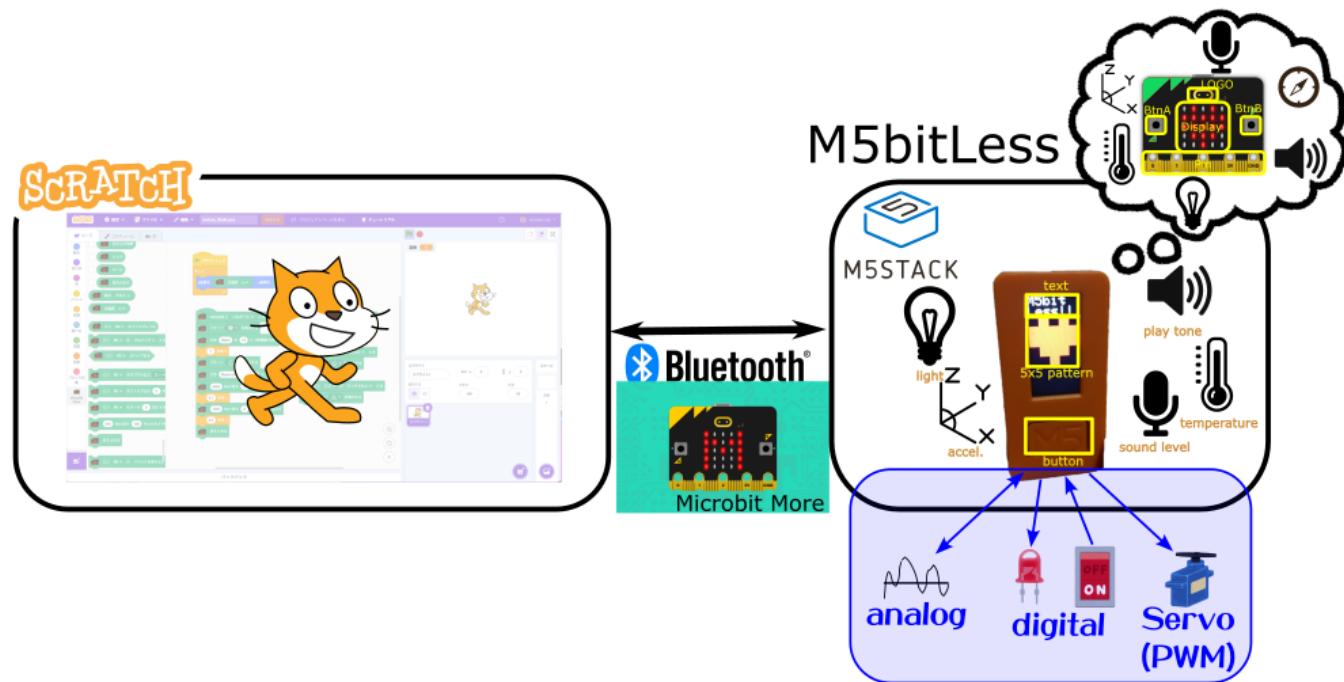
初めて買うときの私のおすすめは、[M5StickC Plus2 ウォッチアクセサリキット](#)です。 腕に巻いて遊べます。

ただ、後で説明するスタッフやで遊びたい場合は、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

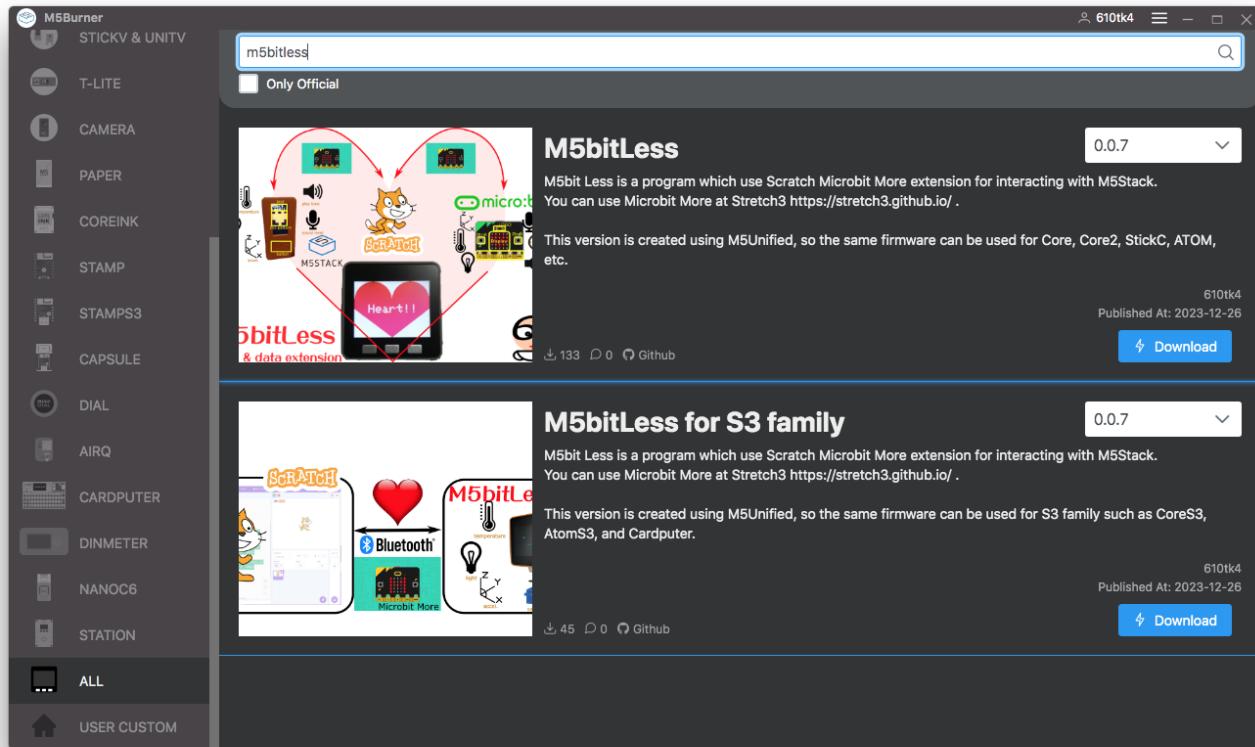
- [M5StackとScratchで遊ぶたった3つの冴えたやり方](#):M5StackとScratchで遊ぶためのUIFlow, M5Scratch, M5bitLessの3(+1:つくるっち)つの方法について解説しています。
- [ScratchとM5Stackで遊ぶ](#):Scratch遠隔センサーについての説明が少し詳しみです。

M5bitLess = M5Stack x Scratch:M5StackでMicrobit Moreを使う

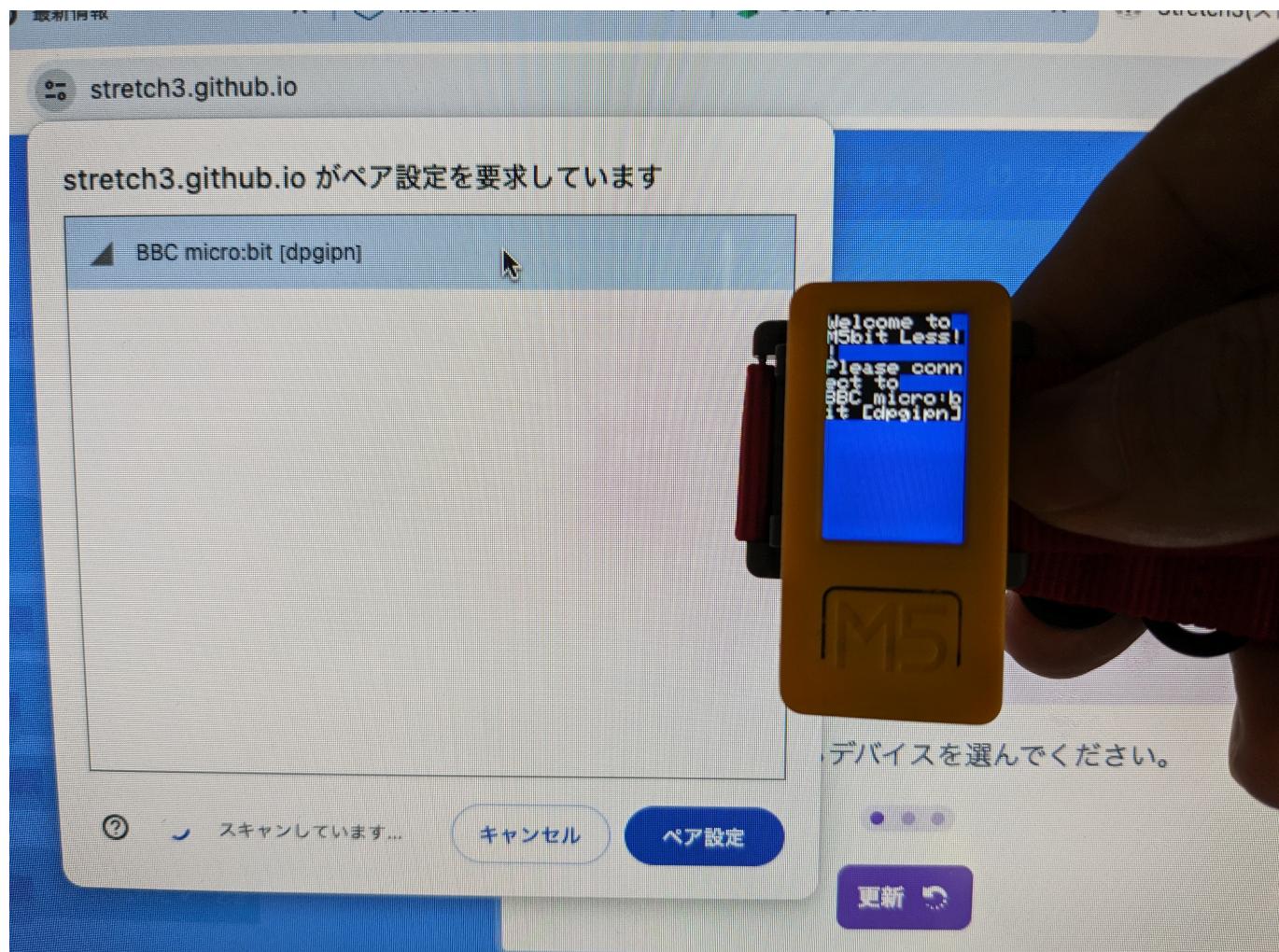


M5bitLessは、M5StackでStretch3+Microbit Moreの使うための拙作のプログラムです。M5Stackがあたかもmicro:bitのようにふるまうことことで、動作しています。M5bitLessは、Arduinoプログラムです。

M5bitLessは、M5Burnerからも焼くことができるため、開発環境がない場合でも使えるようになっています。

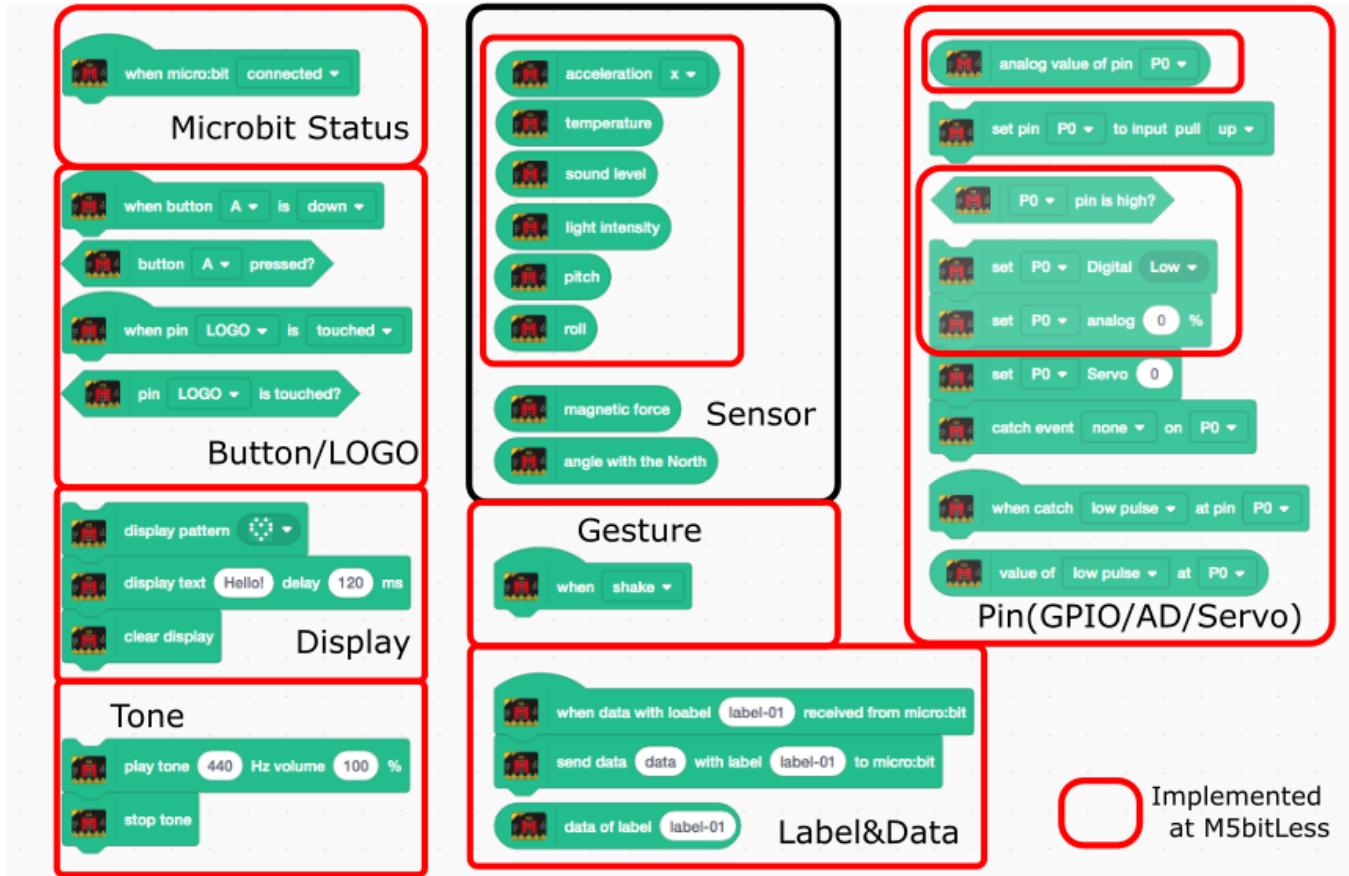


M5bitLessを拡張機能から追加すると、以下の画面のようにM5Stackに接続する設定が表示されます。M5Stackに表示されたのと同じIDを選択してください。



あとは、普通のScratchのように利用することができます。ただし、以下のように、Microbit Moreの全ての機能が実装されているわけではないので、注意してください。

Microbit More Blocks



参考文献

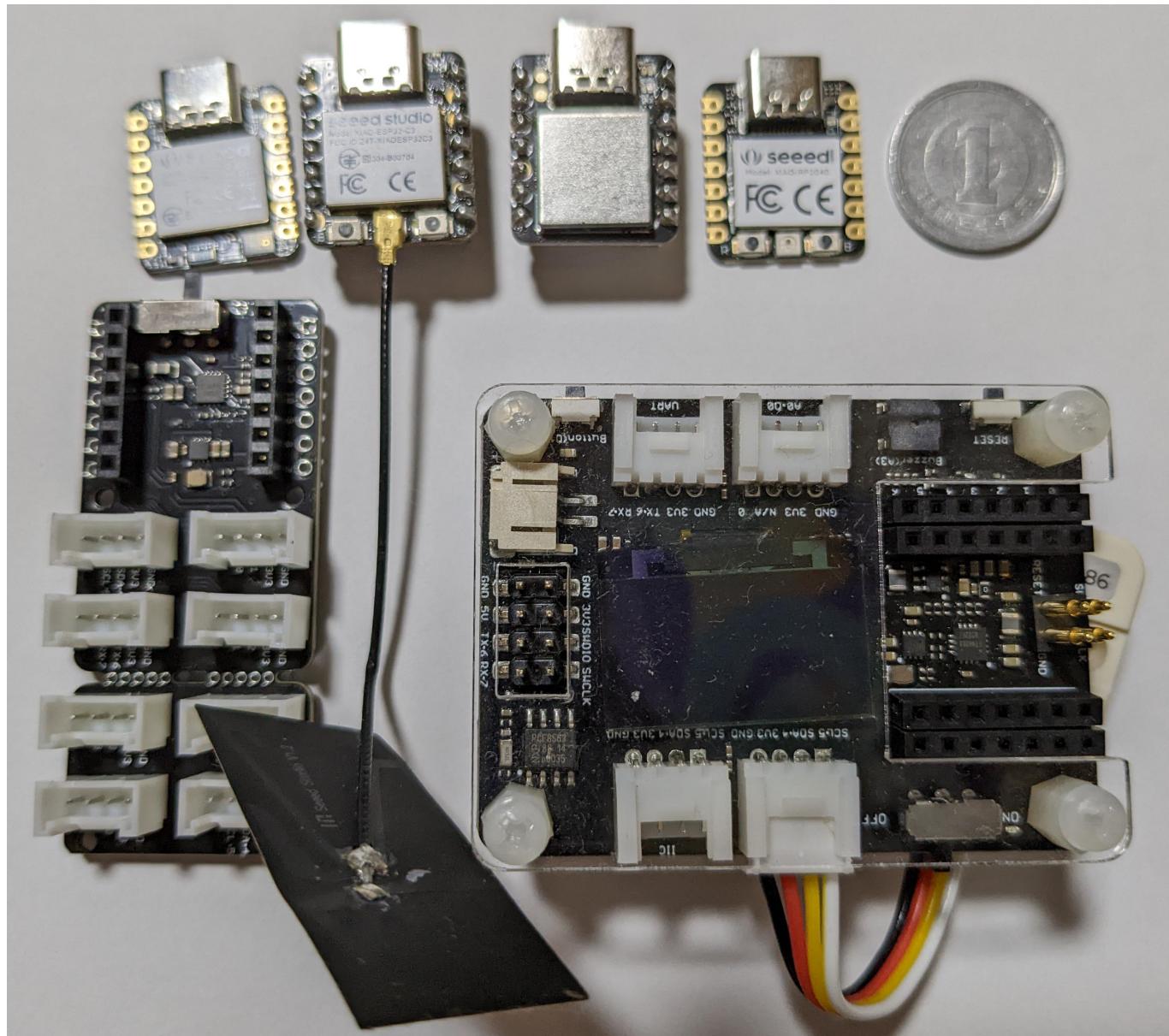
- [M5bitLess: M5Stack x Scratch3 = So Fun!!](#): M5bitLessのシステム全体を知るのに良いと思います。
- [M5bitLess label & data extension](#): データをやり取りするLabelとDataというしくみの説明です。
- [M5bitLessのI/Oサポート](#): M5bitLess外部のハードウェアを利用するための拡張についての説明です。
- デモ類
 - [M5StackとScratchとhapStakでスポーツの秋に挑戦!!](#): M5bitLessを使って、運動するゲームを作つてみました。

その他のマイコンボードで*bitLess系列を使う

M5Stack以外にも、色々なマイコンボードでStretch3+Microbit Moreの構成でScratchを利用することができます。

- 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](#)

SeeedのXIAOファミリーは、以下のようなボードになります。とても小さくて色々なものに組み込みやすく、マイコンの種類も色々あって、使い勝手は大変良いです。

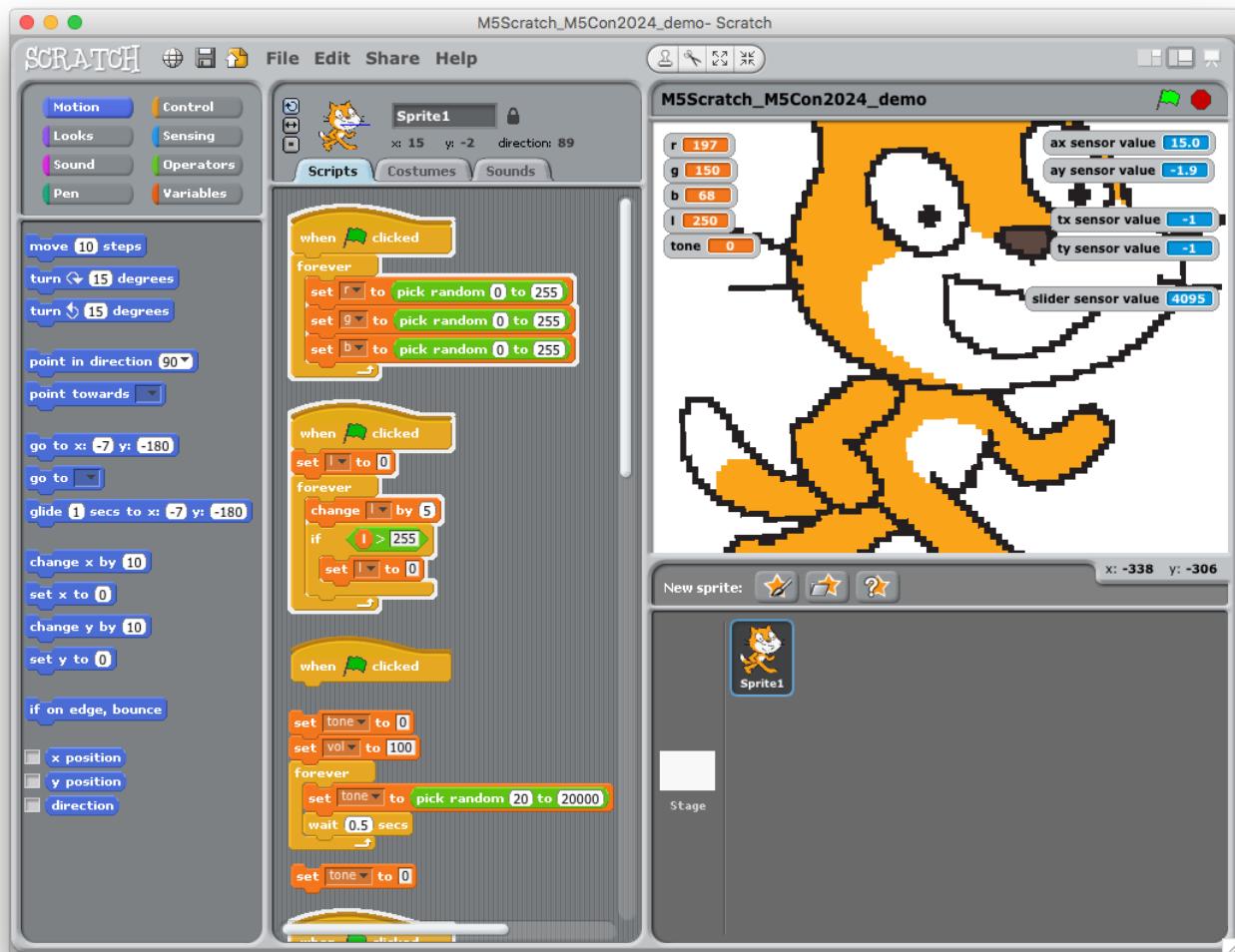


参考文献

- [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と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。 これは、800x480という狭い画面でも動作できたり、軽く動く、オフラインで使えるなどの特徴があるためです。 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)

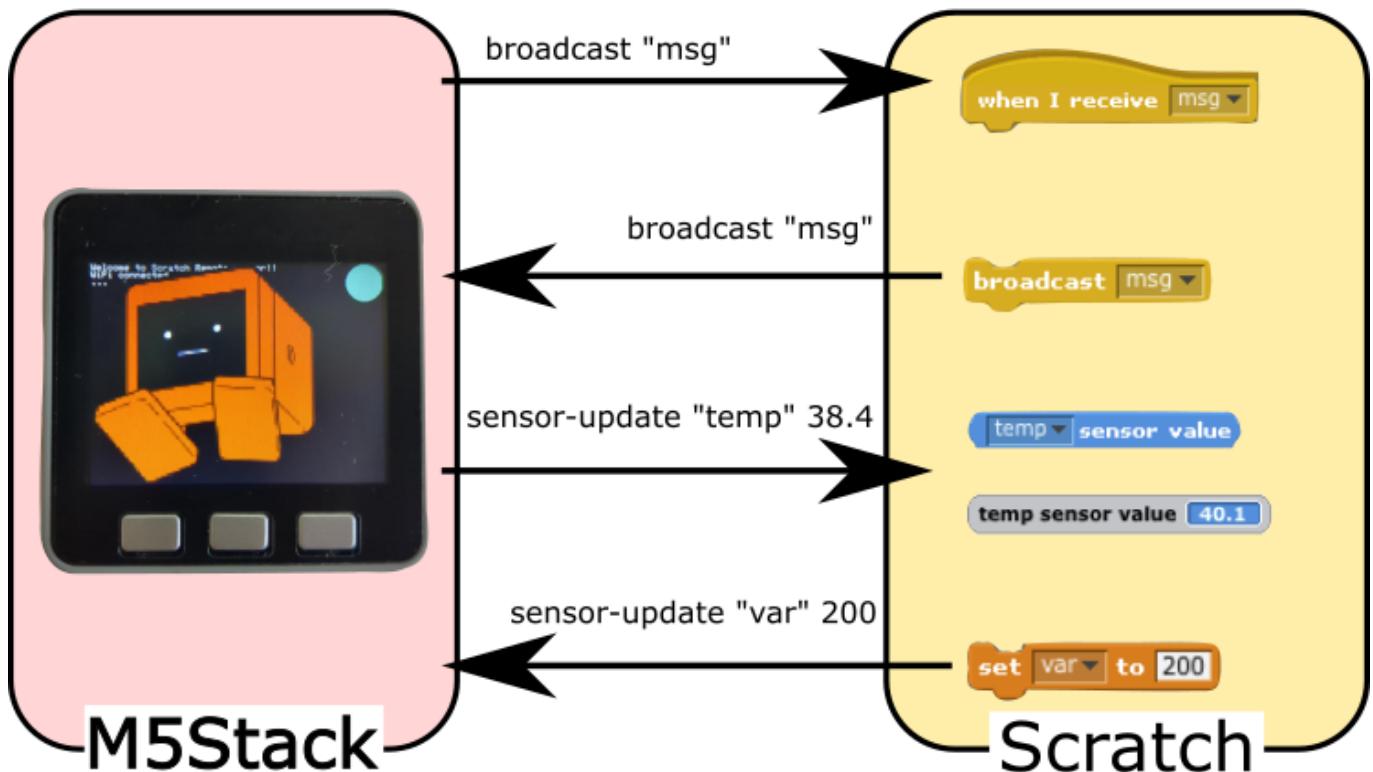
Scratch遠隔センサープロトコル([Remote Sensors Protocol](#):以下RSP)は、Scratchとネットワークでつながった外部の何かと情報をやり取りするための仕組みです。 「何か」には色々なデバイス以外にもソフトウェアやサービスも含まれており、Scratch同士の通信も可能になっています。 基本的にはTCP/IP 42001を利用してやり取りしますが、UDP/IP 42001も利用可能です。

RSPを利用するためには、少し準備が必要で、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はテキストベースのプロトコルであり、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

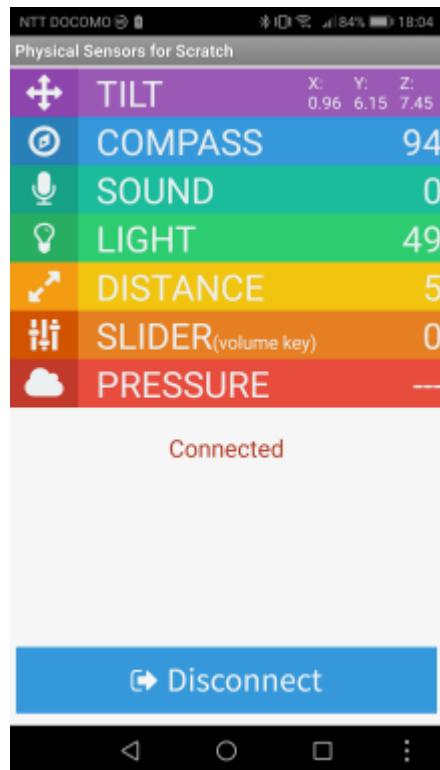


AndroidでのRSPの利用

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っています。

- Android
 - [Physical Sensors for Scratch](#): 現在利用できません。
 - [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!!

下の図は、Physical Sensors for ScratchのUIです。スマートフォンの傾きセンサーやコンパス、音入力などが利用可能になっています。



Scratch x M5Stack UIFlow

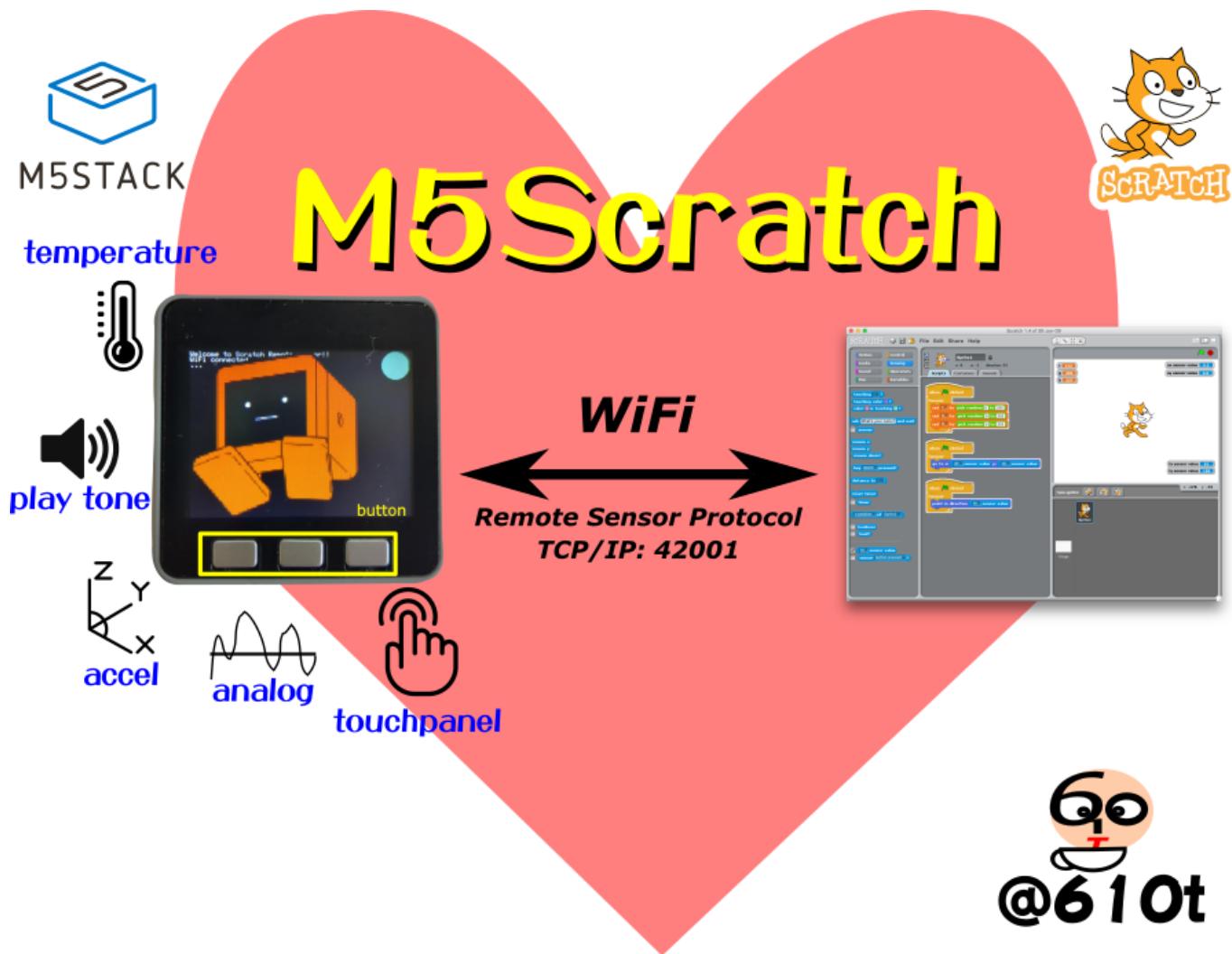
UIFlowはM5Stackの標準開発環境のひとつです。 様々なM5Stack純正のGroveデバイス類が簡単に利用できるようになっています。

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

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ています。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。

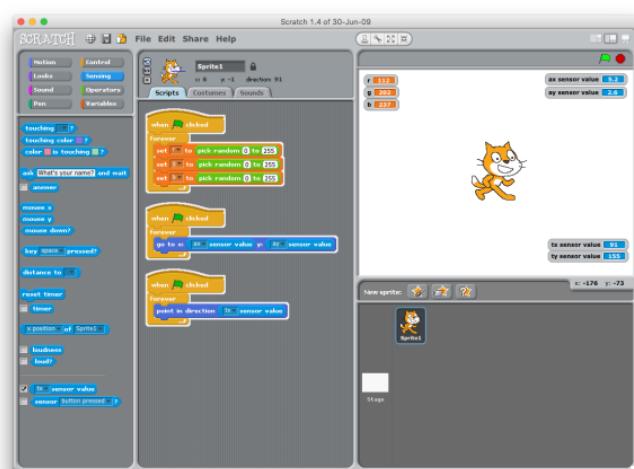


M5Scratch = Scratch x M5Stack:M5StackでScratch遠隔センサーを使う



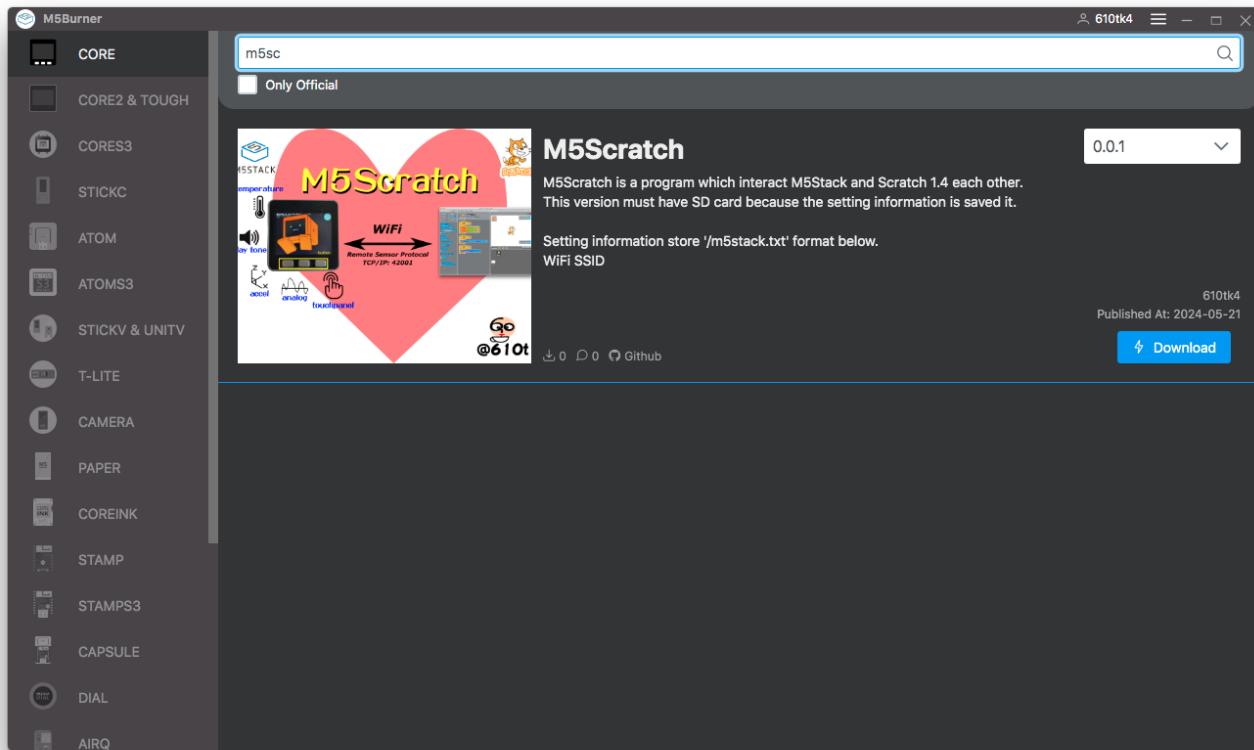
M5Scratchは、M5StackとScratchで、Scratch遠隔センサー(RSP)を使って情報のやり取りを行うためのプログラムです。M5Scratchは、Arduino言語で書かれています。

M5Stack



Scratch 1.4

M5Scratchは、上の図のように、WiFiを使ってScratchと通信します。このため、RSPを利用するためにはIPアドレスや無線LAN(WiFi)の情報を指定する必要があります。SDカードがある機種では、SDカードに設定情報を入れておくことができるようにしてあるのですが、SDカードのない機種ではこれらの設定情報をプログラムの中に入れて、再コンパイルする必要があります。SDカードで設定可能な機種のために、M5Burnerでも提供していますが、この制限には注意してください。



SDカードで設定する場合、SDカードの一番上の場所(/)の`m5scratch.txt`というファイルに以下の形式で設定情報を入れてください。

```
WiFiのSSID
WiFiのパスワード
Scratchが動いているパソコンのIPアドレス
```

Scratch側の最初の設定に関しては、[遠隔センサープロトコル\(RSP\)](#)を参照してください。

M5Scratchでは、はじめにScratch側から変数を送らないとRSPによるデータの交換がはじまらないので、注意してください。

参考文献

- [M5Scratch: M5Stack x Scratch1.4](#):システム全体の説明です。
- 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でゲームを作つた例です。](#)
 - [M5Stack Christmas with M5Scratch](#):これもゲームを作つた例です。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、ししかわさんが開発、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。

スタックチャンの情報は[スタックチャンのScrapbox](#)に現在進行形で集められています。

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普通に販売しているものとしては存在しません。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ロボット 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。

スタッキチャンのソフトウェアは、M5Burnerでもいくつか提供されており、簡単に使うことができます。これには、Bluetoothスピーカー版や、AI版などがあります。M5Burnerで提供されているもので、"スタッキチャン"で検索できるものには、以下の図のようなものがあります。他にも"Stack-chan"や"Stackchan"などでも登録されているので、探してみてください。



CoreS3 AIスタッキチャン (顔検出)

CoreS3の内蔵カメラで顔検出するAIスタッキチャンです。
(robo8080さんのAIスタッキチャンをベースとしています。)

- ・顔検出すると「御用でしょうか?」と音声認識を起動します。
- ・LCD左上隅にカメラ画像が表示されます。画像部分をタッチすると表示ON/OFFできます。

Published At: 2023-06-21
motol

[Remove](#) [Burn](#)



スタッキチャン BTスピーカー

アフコさん仕様 Ver3.1 (Port.B LEDx70 & Port.C LEDx70)

**bluetooth-simple
(CORE2 for AWS &
CORE2 V1.1)**

Ver3.1 : LEDエフェクトをマルチタスク対応にし負荷分散化しました。

アフコさん仕様をアップロードしました。LED 70弾に対応しました。ネコミミLEDで遊べます。
エフェクト切り替えは、スタッキチャンの顔中央左側をタップして変更します。(それぞれのエフェクトはGitHubにリンク貼りました。)

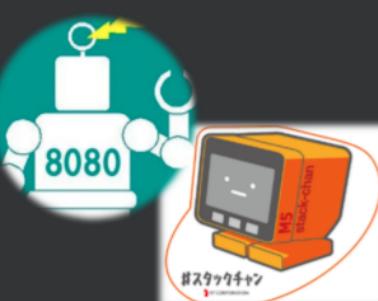


AIスタッキチャン2 KnightRider season2

for
AWS(& core2+Go Bottom2, &core2(V1.1)+Go Bottom2)
ChatGPT-API,
VOICEVOX-API,
Google Cloud Speech to Text-API

Published At: 2023-06-01
yama2010

[Download](#) [Upload](#)



**AIスタッキチャン2 (FY24/7月 証明書
更新+ChatGPT-4o mini)**

～～【AIスタッキチャン2：バージョン情報】～～
0.0.10 : モデル GPT-4o mini に更新しました。(サーポ Port.A / Port.C版を別ファームにて提供)
0.0.9 : rootCA証明書を更新しました。(2024/07/10 ~)
*GitHubリンクは暫定で私の更新したプランチを指定しています。
↓↓↓

Published At: 2024-09-29
t_mac116

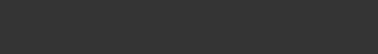


スタッキチャンRadikoプレイヤー

【注意】 Core2 Ver.1.1では動きません。
You can listen to radiko with Stack-Chan.
Note: Can only be used in Japan.
M5Stack Core2スタッキチャンで動きます。
サーポはPORT.Aに接続してください。

Published At: 2023-05-08
robo8080

[Download](#)



スタッキチャン(ランダムwav再生)

0.0.1



【注意】Core2 Ver.1.1では動きません。
独り言を言うスタッキちゃんです。セリフをSDカードにwavファイルで入れておくとランダムに再生します。
【使い方】
* サーボはPORT.Aに接続してください。
* SDカードに、wavというディレクトリを作りそこにwavファイルを入れておきます。

robo8080
Published At: 2023-05-11

↓ 117 □ 0 GitHub Download



Stackchan-AIスタッキチャン-ミニマル(Minimal) v0.0.2

==GPT-4o miniと多言語対応βを追加2024/7/19==
【誰でも気軽に遊べる“おもちゃのスタッキちゃん” Japanese only】
ボタンを押して話しかけると、対話型AIとおしゃべりできます。またWeb設定画面から、キャラクター音声/AIモデルの切替えができ、長い文章もテキスト入力で会話することができます
*Atom Echo単体でも動きますが、ディスプレイと外装ケースでよりかわいい姿になります

Aoya-Uta
Published At: 2024-06-14

↓ 53 □ 0 GitHub Download



AIスタッキチャン2 (LEDエフェクト実装版) 内蔵LED+ネコミミLED ※Ver 0.0.11 (ChatGPT-4o mini)

~~~ 【AIスタッキチャン2：バージョン情報】 ~~  
0.0.11：初期リリース (2024/08/16) \*ネコミミLEDはわしひさんのNekomimi LEDに最適化しています  
robo8080さんのAIスタッキちゃん2にLEDエフェクトを実装しました。LEDの動作確認等にお使いください。  
AIスタッキちゃん2は、ChatGPT問い合わせやおしゃべりにメモリを費やす為、LED実装にてシビアなチューニングを施しています。

t\_mac16  
Published At: 2024-08-16

0.0.1

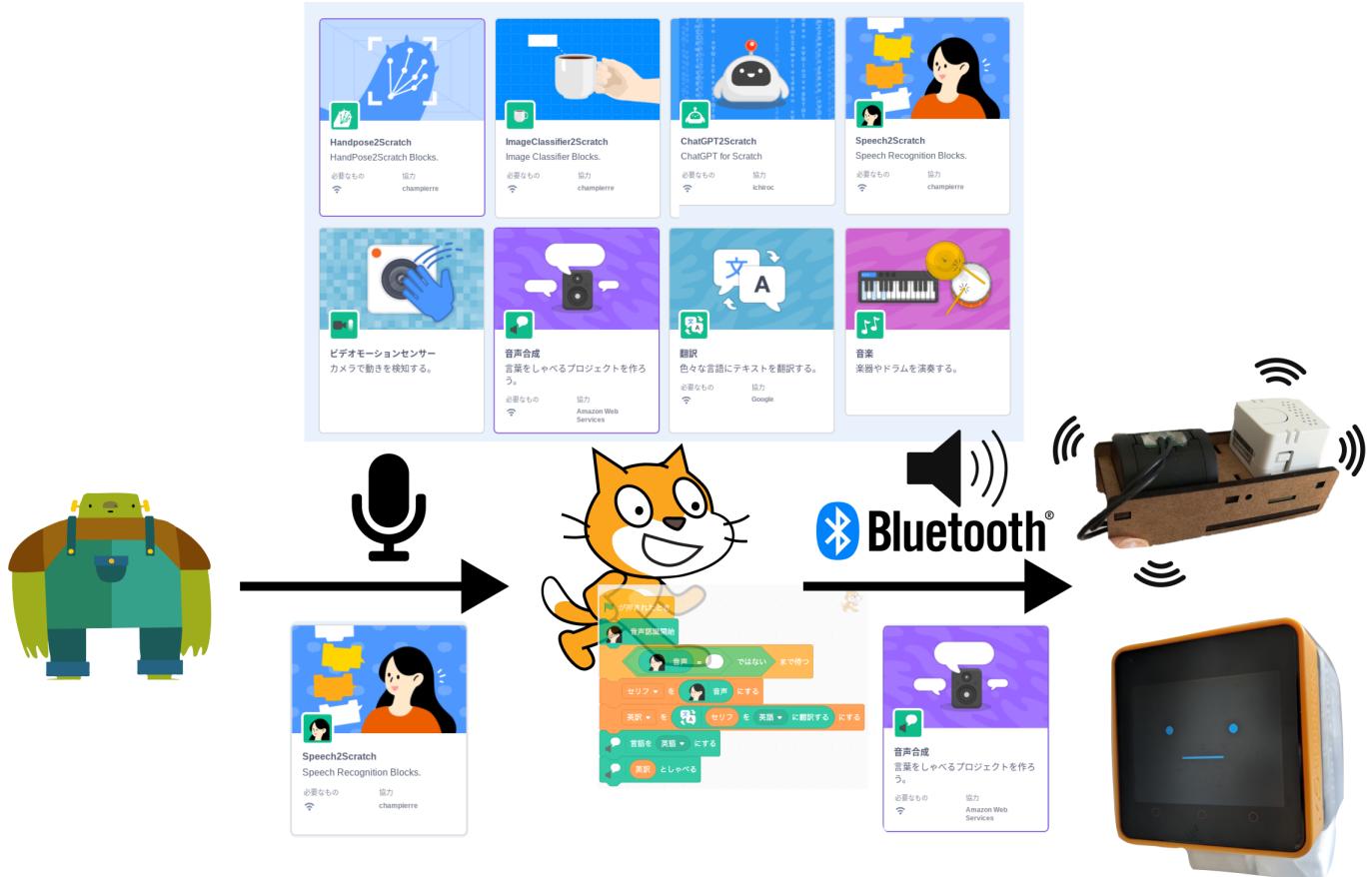
AIスタッキチャン2\_起動デバッグ用

AIスタッキちゃん2を元にデバッグ用にカスタマイズしたものです。  
1. 起動時に読み込んでいるWifiの情報やAPIキーを視認性良く確認できる  
2. 簡単操作でVOICEVOX/ChatGPT接続/ChatGPTとの会話の3段階のどこまで動作するかを確認できる  
3. エラーがあったときにどこが不具合になってそうかを答えてくれます。  
詳細な仕様・使い方はこちら -> [https://github.com/u-tanick/AI\\_StackChan2\\_for\\_DEBUG](https://github.com/u-tanick/AI_StackChan2_for_DEBUG)

u-tanick  
Published At: 2024-09-30

↓ 33 □ 0 GitHub Download

音声入出力を使ってスタッキちゃんと遊ぼう



スタッキちゃんには、Bluetoothスピーカー版があり、M5Burnerで提供されています。このバージョンは、Bluetoothスピーカーとして動作し、その音に応じて顔の表情が変わり、動くようになっています。音声を流してあげると、まさにスタッキちゃんがしゃべっているように見えます。

[スタッキちゃんとScratchちゃん](#)では、このような使い方について説明しています。

## 参考文献

- [Scratch V2 Programming](#):スタッキちゃんとhapStakを音声で使っています。

## M5bitLessを使ってスタッキちゃんと遊ぼう

[スタッキちゃん 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。

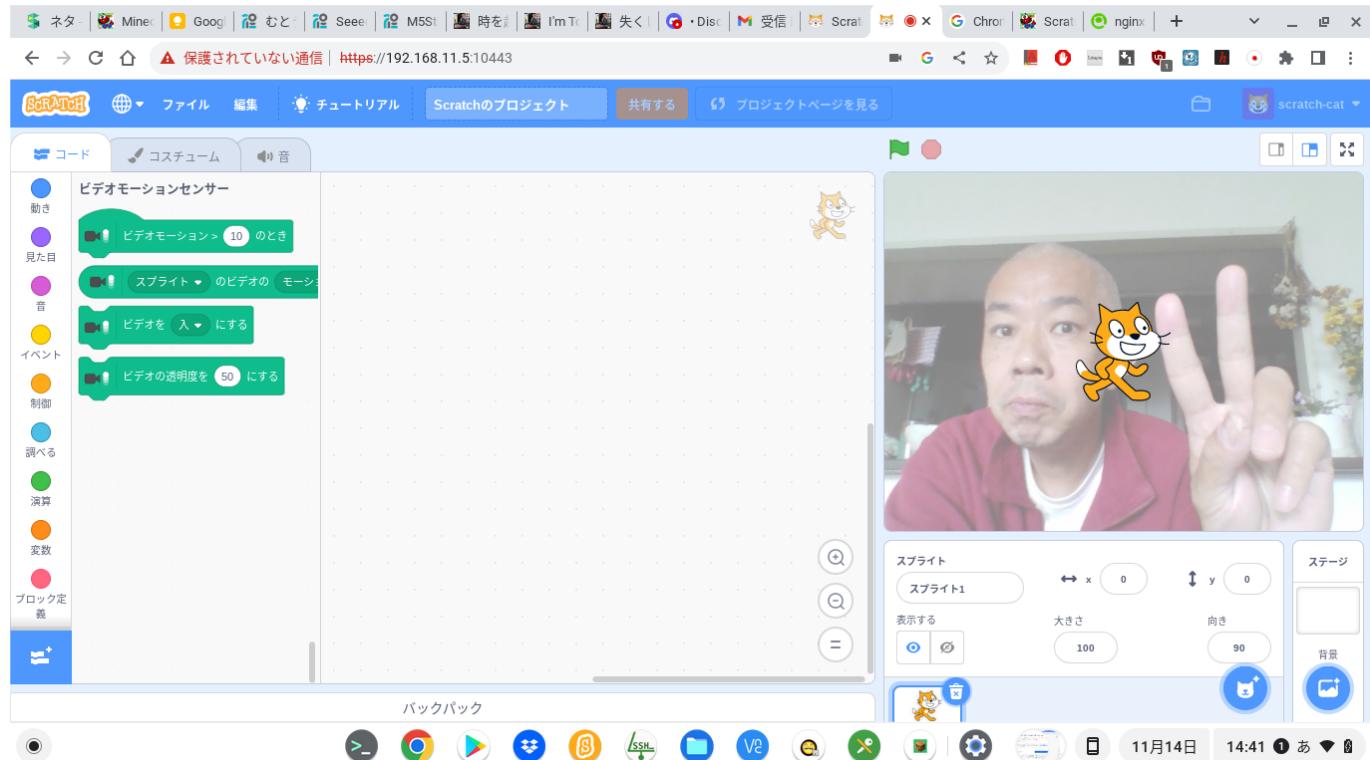
Microbit Moreを追加するためには、以下のようにします。このコマンドは、[scratch-gui](#)で行ってください。

```
# 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。 設定がうまく行つていれば、<https://サーバーのIPアドレス: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。

# 拡張機能の作成

Scratchでは、既に書いたように、サーバーのソースが公開されており、自分の好きなように変更してサーバーを立ち上げることができます。ただ、やみくもにソースを変更して機能を追加したりすることは大変な作業になります。

拡張機能は自分の足したい機能を決まったフォーマットで実装するための仕組みで、比較的簡単に機能追加ができます。

## 通常の拡張機能を作成する

通常の拡張機能を作成する方法に関しては、有料(500円)ですが[Scratch を改造しよう](#)が大変参考になります。

これに関しては、今後解説を追加したいと思います。

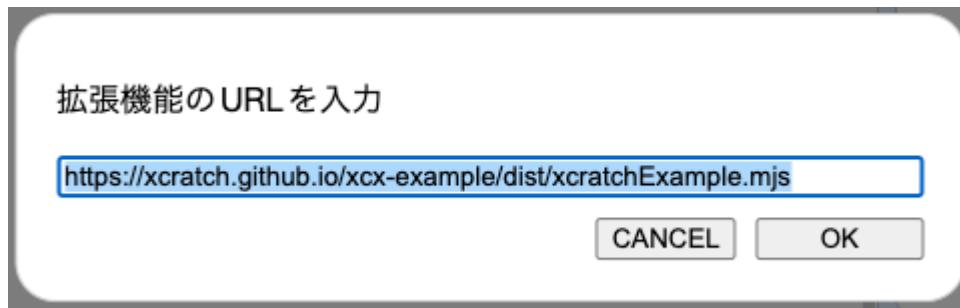
## xcrash用拡張機能

[xcrash](#)では、xcrash用に作成された拡張機能を動的に読みこむことができる拡張機能が提供されています。xcrash拡張機能用のモジュールを作成するには、決まったフォーマットで拡張機能を作成する必要があります。



## xcrashを利用する

xcrashの"拡張機能を読み込む"を選択すると、以下のようなURLを入力するウインドウが立ち上がります。ここで、xcrash用に作成されたモジュールのURLを入力します。



xcratch用に提供されているモジュールには、例えば以下のようなものがあります。

- [Microbit More](https://microbit-more.github.io/dist/microbitMore.mjs): <https://microbit-more.github.io/dist/microbitMore.mjs>
- [NumberBank](https://con3code.github.io/xcx-numberbank/dist/numberbank.mjs): <https://con3code.github.io/xcx-numberbank/dist/numberbank.mjs>
- [Ambient](https://610t.github.io/ambient/dist/ambient.mjs): <https://610t.github.io/ambient/dist/ambient.mjs>

## xcratchの拡張機能を作成する

ここでは、xcratchの拡張機能の作成方法を解説します。 xcratch拡張機能の作成方法は、[Interface 2022/2: 大人のためのスクラッチHack](#)が詳しいです。

実際の作成方法に関しては、後日記述します。

### 例: ambient用の拡張機能

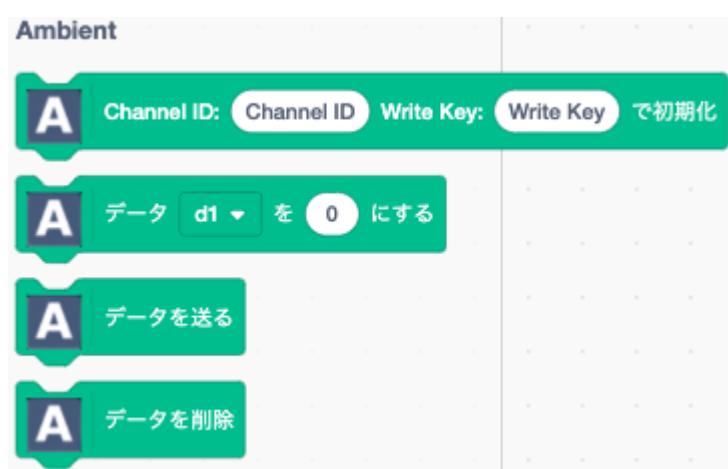
ambient拡張機能は、私が作成した[Ambient](#)というデータ蓄積およびグラフ化サービス用のxcratch拡張機能です。

ambient拡張機能を利用するには、xcratch拡張機能でURLとして <https://610t.github.io/ambient/dist/ambient.mjs> を指定してください。ソースコードは、githubの <https://github.com/610t/ambient/> で提供しています。

### ambient拡張機能の使い方

はじめに使い方を簡単に説明します。

ambient拡張機能には、以下のようなブロックがあります。

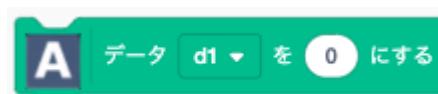


はじめにambientでアカウントを作成します。これに関しては、ここでは省略します。

ambientを利用するためには、自分のアカウントのChannel IDとWrite Keyを準備しておきます。



ブロックでこのChannel IDとWrite Keyを設定します。



送るデータは、  
で設定します。 設定しただけではデータが送られないことに注意が必要です。

実際にデータを送るには



ブロックを使います。 この時、設定したデータはクリアされます。

強制的にデータのクリアを行いたい時には、



ブロックを使います。

### ambient拡張機能の解説

今後作成予定です。

### 参考文献

- [Scratch を改造しよう](#)
- [Interface 2022/2: 大人のためのスクラッチHack](#)

# おわりに

---

とりあえず、Scratch Days in Tokyou 2024中にbeta0版として完成させました。 まだまだ内容を追加して、いつかはちゃんとした同人誌として完成させたいと思います。 特に、Stretch3の拡張機能については、書きたいことがもっとたくさんあります。

皆さんからのご意見、ご要望をお待ちしております。

- X(twitter): @610t
- Email: takeshi.mutoh@gmail.com

# 奥付

---

初版(beta0): 2024年5月26日(日) [Scratch Day 2024 in Tokyo](#)が行われている日に