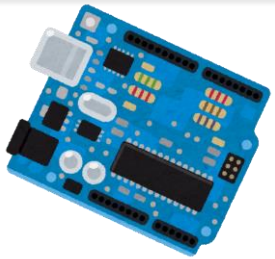
デザイン工学:プロトタイピング論

秋田純一（金沢大）
akita@is.t.kanazawa-u.ac.jp
@akita11

（準備）マイコンとコンピュータ

マイコンは、
演算性能は低めだが、
実世界（物理世界）との
連携が得意



	PC (Personal Computer)	スマホ	マイコン (Micro-controller)
演算性能	◎	○～◎	△～○
インターネット接続	○（WiFi経由）	◎	△～◎
省サイズ	△	○	◎
低価格	△	○	◎
実世界連携	△	○	◎

（準備）使用するマイコン：M5stack

- 中国M5stack社の製品
  - ESP32（WiFi/BluetoothLEつきマイコン）
    （240MHz Dual Core, 600MIPS）
  - センサ等の接続ができるコネクタ
  - 小型液晶モニタ
  - プッシュスイッチ（3個）
  - 単体で「完成品」に近いプロトタイプを作成可能
  - プログラミング環境：MicroPython / ブロック型言語等
  - 積み重ねて機能拡張が可能

（準備）M5stackのプログラミング

「Scratch」に似た
ブロック型プログラミング
Blockly

UI Flow DesktopIDE（USB版）
https://m5stack.com/pages/download

UI Flow（Web版）
~~http://flow.m5stack.com/~~
http://flow01.m5stack.com/

MicroPythonでコーディング
（プログラミングに慣れている
人は便利かも）

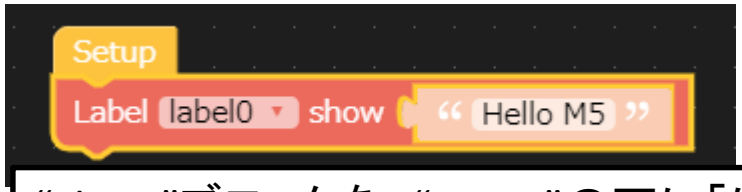
Blockly→MicroPythonへの
変換が可能（※逆は×）

1.お約束の”Hellow World”

- UI Flow DesktopIDEを開く

※不要なLabel等はドラッグ時に現れる「ゴミ箱」へ入れれば消去

“UI”内の”Label”に、Labelに対する処理ブロックが現れる



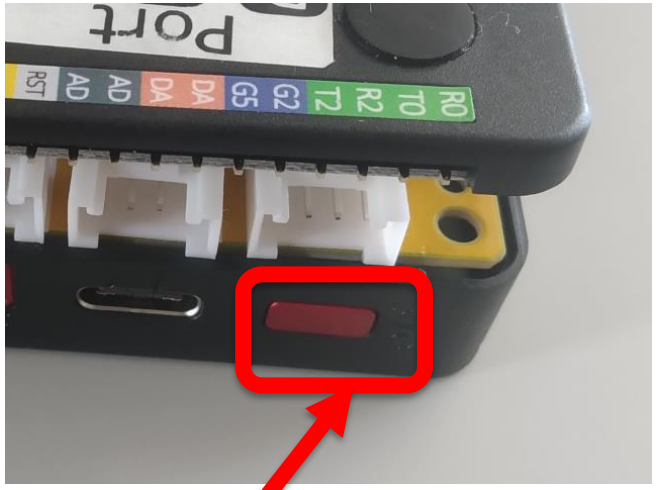
“Show”ブロックを、“Setup”の下に「はめる」（「はまる」と、グレーから色が変わる）

※”Setup”は、起動後に実行される処理を「はめる」場所

1.M5stackの電源のON/OFF

- PC等にUSB-Cケーブルで接続して給電
  - 使用後はケーブルを抜けば電源OFF（ブチ切OK）
  - 内蔵バッテリもある（この授業では普段は使わない）

主電源
（普段はOFFにしておく）
※ONにするとバッテリから給電＆充電



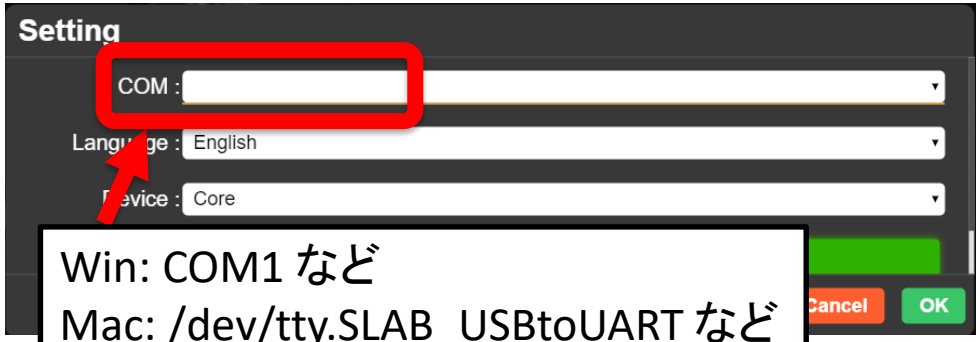
本体電源SW（赤）
・**1回押し→リセット（※よく使う）**
・2回押し→電源ON/OFF（バッテリ動作時）

1.プログラムの転送・実行

初回のみ行う設定

本体ディスプレイの上部に
「USB Mode」と出ているのを確認

なっていなかったらリセット
→すぐ右ボタンを押し続け
→メニューから”USB Mode”を選択

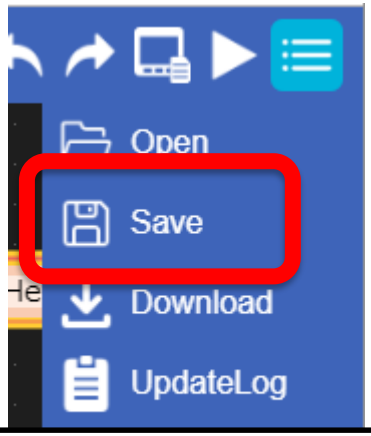


Win: COM1 など
Mac: /dev/tty.SLAB_USBtoUART など

これらが選べなかったら、
左上の「Driver Installation」からドライバを入れる

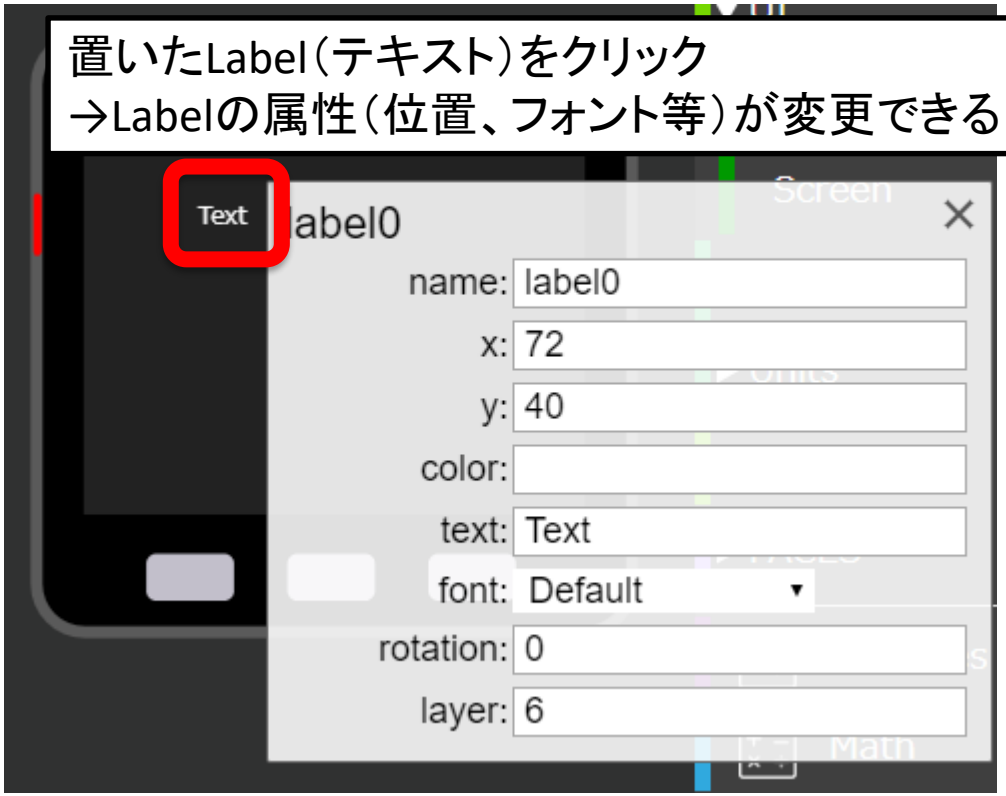
この部分が个で指定した名称になる

”Run”ボタンで、M5stackに
プログラムを転送し、実行



”Save”でプログラムを保存できる
※「1a.m5f」のように、作業ごとに名前を
　変えて保存しておくこと
※”Load”で読み込むこともできる
※画面キャプチャもあわせて
　保存しておくと便利

2.テキストの属性を変更する

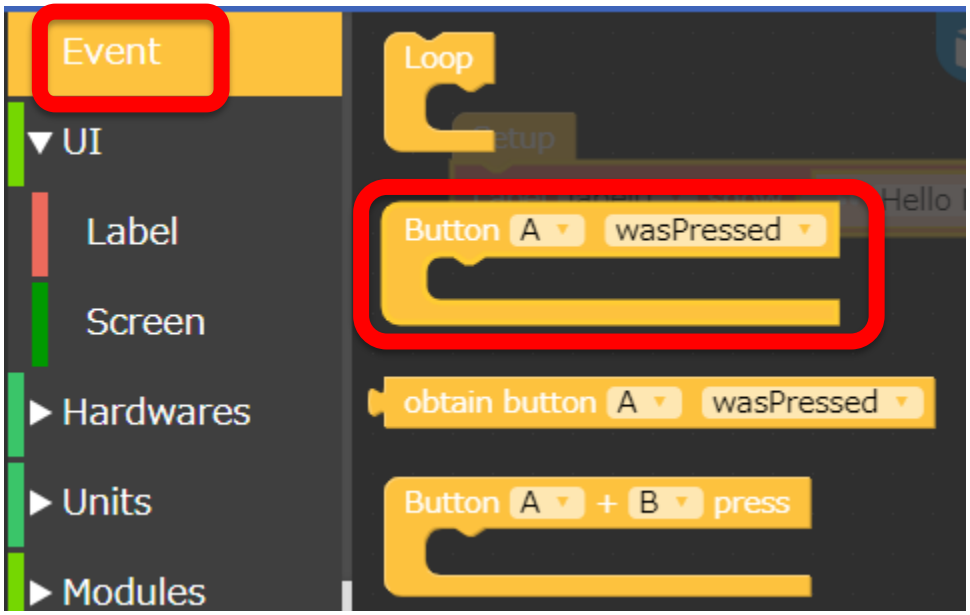


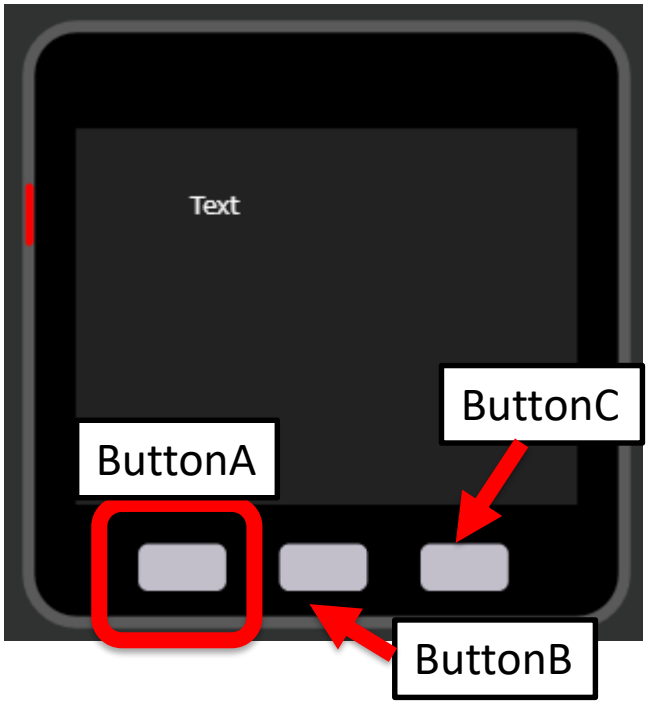
黒枠白字の内容は
各自で実施すること
（結果（写真等）・プログラムも保存しておくとよい）

※ここの”text”属性で、画面上に表示される文字列を設定できるが、
プログラムの”Setup”内で設定してもよい（結果は同じになる）

位置や色、フォント（文字サイズ）を変更して実行してみる
また、背景の色の変更や、適当な図形（矩形・円）なども描画してみる

3.本体ボタンの動作





ButtonAを押した時に実行される処理を書くブロック
※検出するボタンの種類・押し方などを選べる

いずれかのボタンを、いろいろな押し方で押して
Labelを変更するプログラムを実行してみる

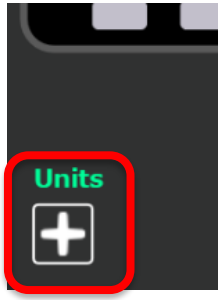
4.本体ボタンの動作：Wait

Timer内の”Wait”で、指定時間だけ待機する

Waitで指定した時間待つ動作を確認する

5.センサの使い方：Button

- M5stack純正の各種センサ等の「Unit」を接続し、UI Flowの”Unit”から使用できる



Unitを追加

DualButtonを選択
（接続するPortをBとする）

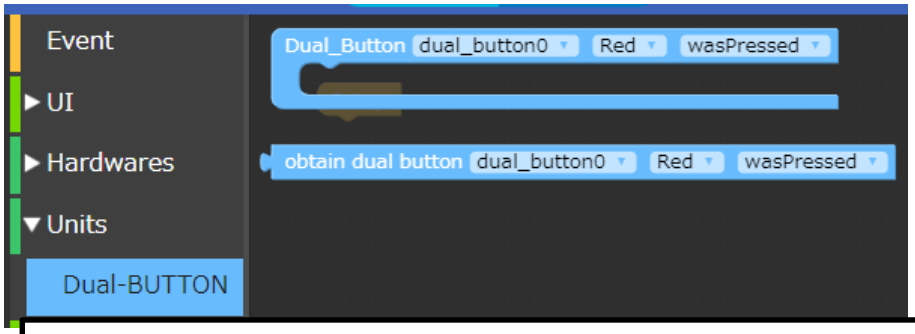
DualButtonユニットが
追加された

やや見にくいが、
スクロールバーでスクロール

実際にDualButtonユニットを、
指定した”PortB”に接続する

コネクタの片面の両側にヤマが
あるので、ソケットにあわせる

5.センサの使い方：Button



“Units”内に、追加したUnitに関する
ブロックが現れる

接続したButtonユニットの
赤ボタンを押した時に
label0を”Hello M5”とする
プログラム

これを参考に、以下の仕様のプログラムを作成して動作させる

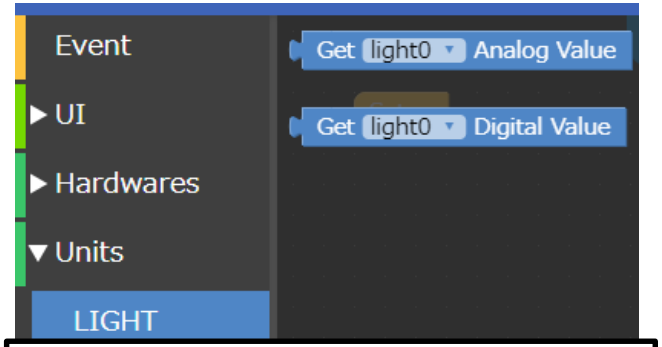
- 動作仕様
  - ラベルを1つおく
  - 初期状態では、label0=“”とする
  - 赤ボタンを押したら、label0を1秒間だけ”Red”に変更し、その後””に戻す
  - 青ボタンを押したら、画面中央(160,120)に半径30の円を描画し、1秒後に消す

6.センサの使い方：Light（明るさ）

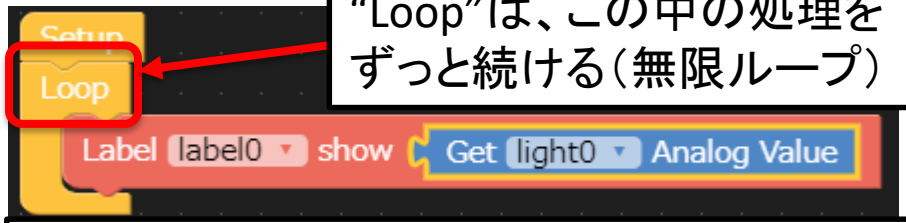
Lightを選択
（接続するPortをB or Dとする）

Lightユニットは、センサ面の明るさに応じた
電圧が出力され、それを取得できる

Unitで設定した
Portに接続する



Lightユニットで使える処理
（“Analog Value”でアナログ値）



“Loop”は、この中の処理を
ずっと続ける（無限ループ）

Lightsユニットで取得した明るさの値を表示する例

Lightユニットの値（明るさ）をディスプレイに表示し、
明るさに応じて値が変化することを確認する

6.条件分岐

Logic（論理）から持ってくる

Math（数学）から持ってくる

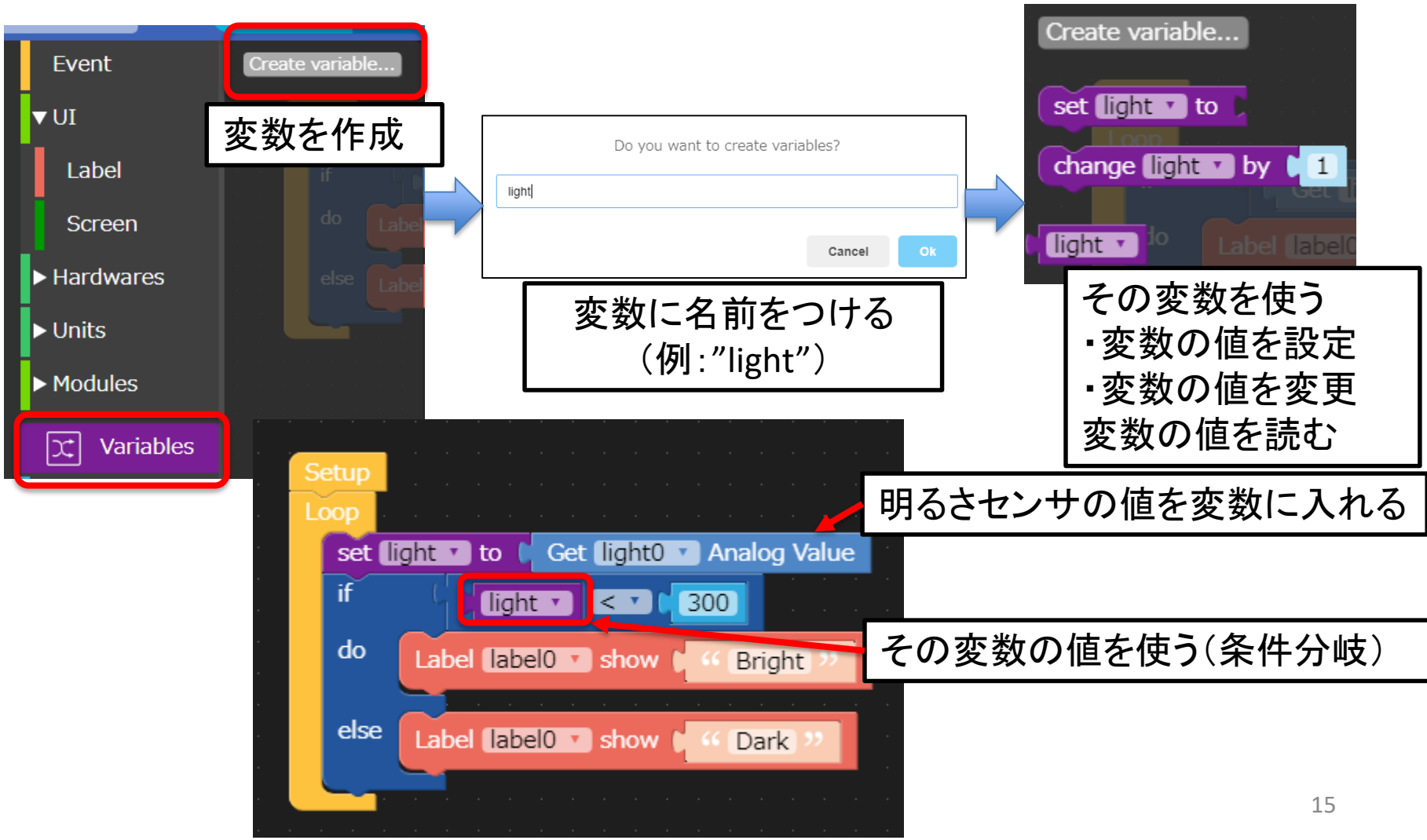
設定したい条件にあわせて変更

If以下の条件が成り立つ場合に実行

If以下の条件が成り立たない場合に実行

6.変数（Variables）

- 値を一時的に保管しておく「箱」＝変数



7.センサの使い方（ToF）

- “ToF”ユニット＝センサ面から対象物までの距離を計測する
  - ToF = Time of Flight
    ※光の往復時間から距離を計測する
  - Unitから”TOF”を追加（PortAを選択）
  - PortA（複数あるがいずれでもOK）に接続する

ToFセンサから取得した値（単位：mm）をディスプレイに表示するプログラムを作成し、実行させる

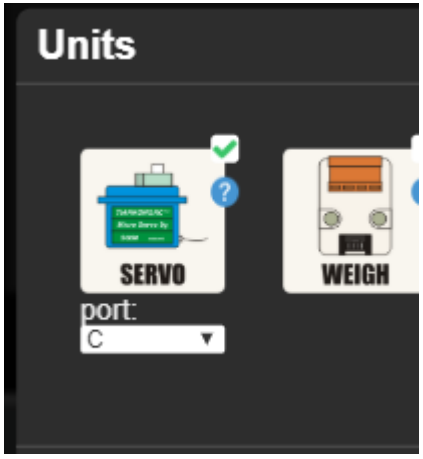
8.センサの使い方（Color）

- “Color”ユニット＝対象物の色（RGB値）を計測
  - Unitから”COLOR”を追加（PortAを選択）
  - PortA（複数あるがいずれでもOK）に接続する

Colorセンサから取得した値（R, G, Bそれぞれ）をディスプレイに表示するプログラムを記述し、実行させる

9.アクチュエータの使い方（サーボ）

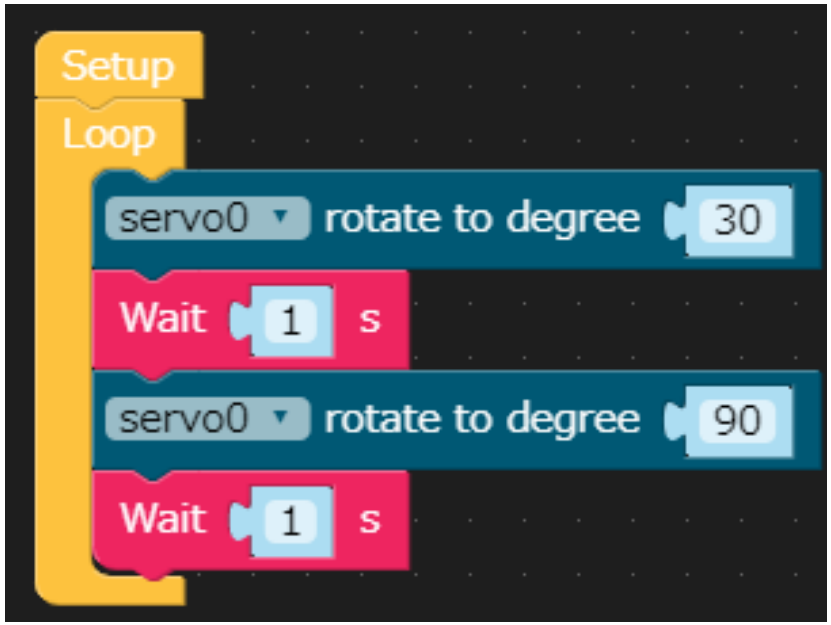
- 指定した位置にアームを移動できる
  サーボモータ
- Unitから追加し、指定したPortに接続



※アクチュエータ (actuator)
　＝環境に作用を与えるもの
　（センサ (sensor)の逆の機能）

9.アクチュエータの使い方（サーボ）

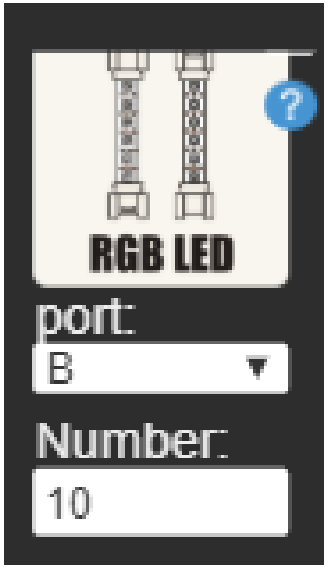
- サーボの角度は”角度[度]”で指定できる



このプログラムの動作を確認する。
またサーボの移動順序を自分で決め、それを実現するプログラムを記述して実行する

10.アクチュエータの使い方（LEDテープ）

- コマンドで複数LEDのRGB値を個別に指定できるLED（NeoPixel）

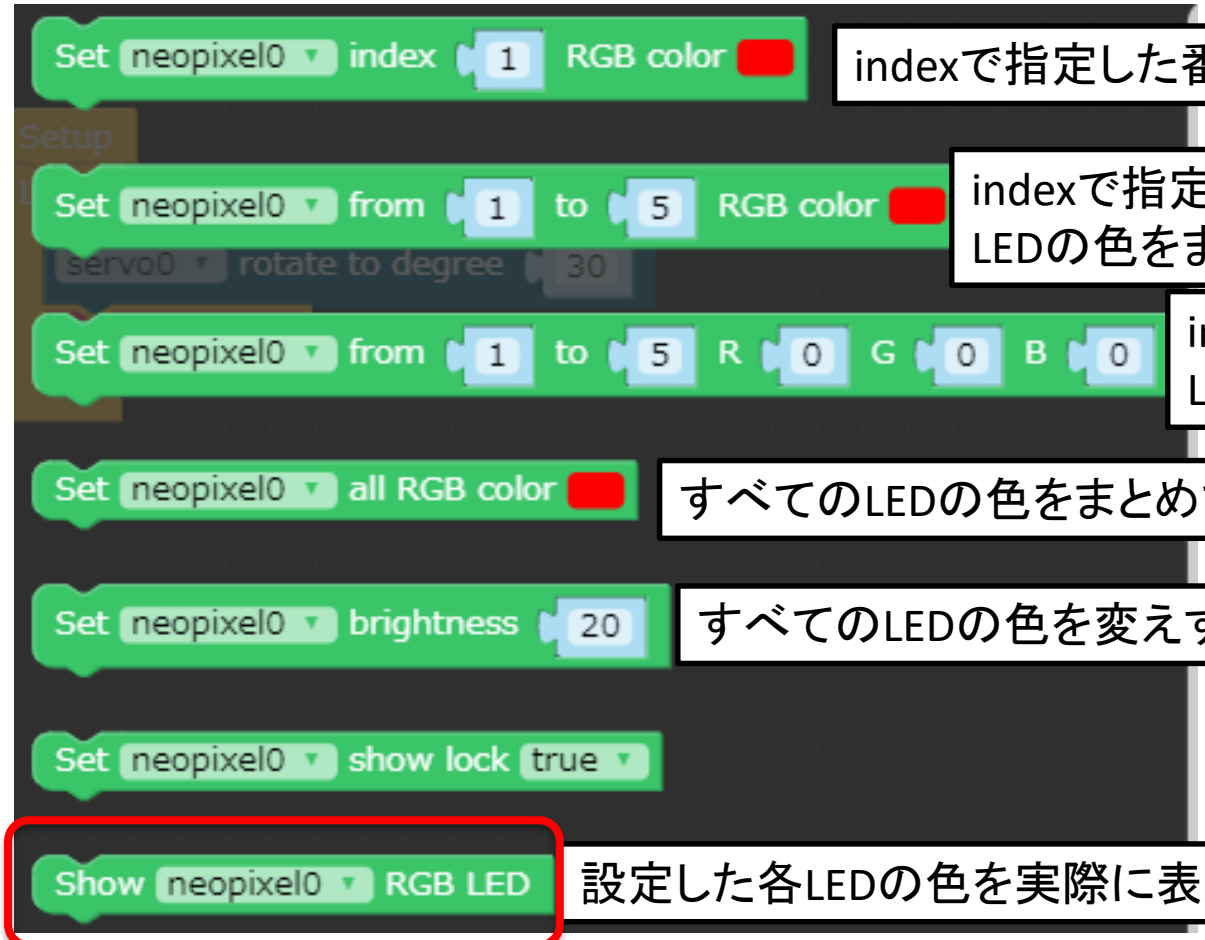


接続するPortと
LED数（この実験の
機材では10個）を指定

指定したPortに接続

10.アクチュエータの使い方（LEDテープ）

Blocklyのブロックを参考に、自分でLED点灯の仕様を決め、
それを実現するプログラムを作成して実行する



indexで指定した番号のLEDの色を設定

indexで指定した範囲の
LEDの色をまとめて設定

indexで指定した範囲の
LEDの色をRGB値で設定

すべてのLEDの色をまとめて設定

すべてのLEDの色を変えずに明るさのみを設定

設定した各LEDの色を実際に表示する

※これを実行した時点で、実際のLEDの色が変わる

アイディアの具体化

- ものをつくるのためには「仕様」が必要
  - 「仕様」を決めてからものを作る？
  - 作って、使ってみて、「仕様」を決める？
  - 「完璧な完成品」を作ってから試す？
  - 「とりあえず動く完成品」を作り、試して、「アイディアが正しかったか」を検証し、改良して作って・・・を繰り返す？

プロトタイプ
(Prototype)

デザイン思考
(Design Thinking)

作る＝思考の道具

アイディアの具現化（レポート）

- 「あったら便利なもの、生活がよくなるもの」を1つ考え、M5stackで試作する
  - 実用レベルの完成度である必要はない
  - あくまでも試作品・動作検証ができればよい
- 以下の内容を含むこと
  - 作ったものの概要
  - それを作ろうと思った理由
  - 試作結果（実機の写真、プログラム（できれば））
  - 実際に使ってみての感想・改善点
- 例：※これと全く同じものは×、ヒントにするはOK
  - 部屋に誰か入ってきたらスマホに通知（ToFセンサ・Remoteを使えば作れそう）
  - 視覚障害者向けの、対象物の色を音で表現する機器（Colorセンサ・スピーカを使えば作れそう）
  - 暗くなったら部屋の電灯をつける（Lightセンサとサーボモータ（吊り紐を引く）でできそう）

成果発表について

- 以下の内容を含むこと
  - 作ったものの概要
  - それを作ろうと思った理由
  - 試作結果（実機の写真、Blocklyのプログラム）
  - 実際に使ってみての感想・改善点

プロタイピングの心得

- 機能やコンセプトを検証するための最低限の実装
  - つくって触ってみることで初めてわかることがある
- プロトタイピングに必要なもの
  - 検証したい機能やコンセプト
  - 実装の迅速さ
- プロトタイピングに
  必ずしも必要ないもの
  - 耐久性
  - バグのないコード

プロタイピングの心得

「似たものが既にある」は
気にしなくてOK！
（むしろ改善・差別化のチャンス）

西餅「ハルロック」（講談社コミック）より

おまけ:さらにやってみたくなった人は

マルツ(金沢西インター店)

https://www.switch-science.com/

以降、オプションの発展的内容
（興味のある人、余裕のある人向け）

実装上の注意

- PortA～Eには、以下の通りIOピンが接続されている。
- 以下のIOピンは、PortA～Eコネクタと本体コネクタに接続されていて、同時に使えないので注意
  - PortA: IO21, IO22
  - PortB: IO26, IO36
  - PortC: IO17, IO16
  - PortD: IO35, IO34
  - PortE: IO13, IO5

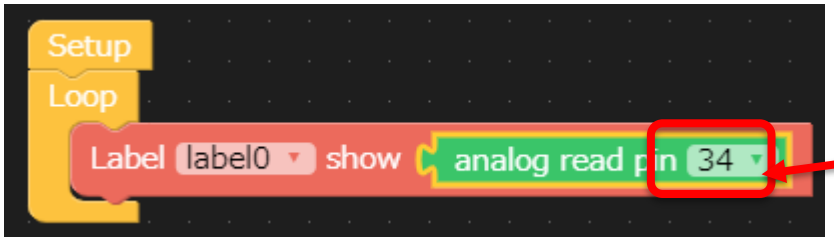
E1.センサの使い方（ジョイスティック）

- ジョイスティックは、Unitにないので、信号を直接読み取る
- PortD＝2本の信号線（IOピン34番と35番）
  （IO=Input/Ouput, 入出力のこと）
- スティックの傾き（X軸・Y軸）に応じて、それぞれの電圧が変化する

E1.センサの使い方（ジョイスティック）

- IOピンの電圧は、”analog read”で取得できる

下のほうの”Advanced”内の
“Easy I/O”にある



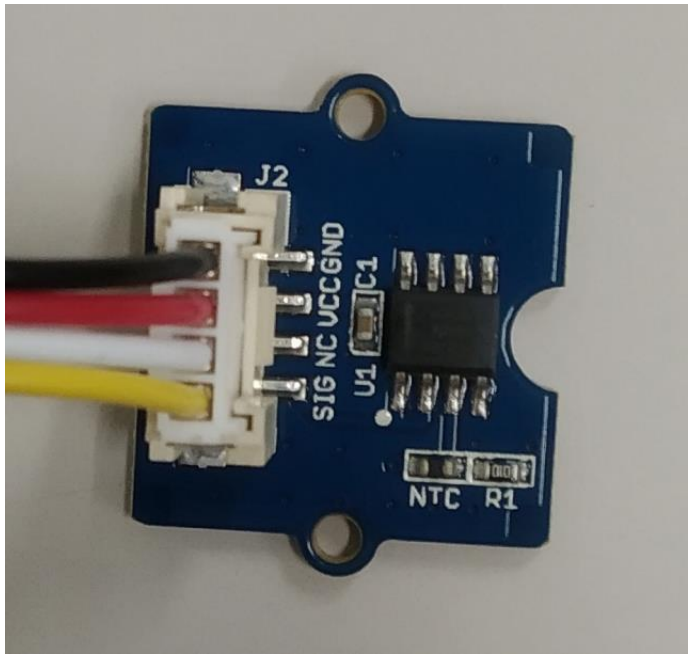
ジョイスティックのX軸は、
IOピン34番につながっているので
そのピン番号の電圧を取得する

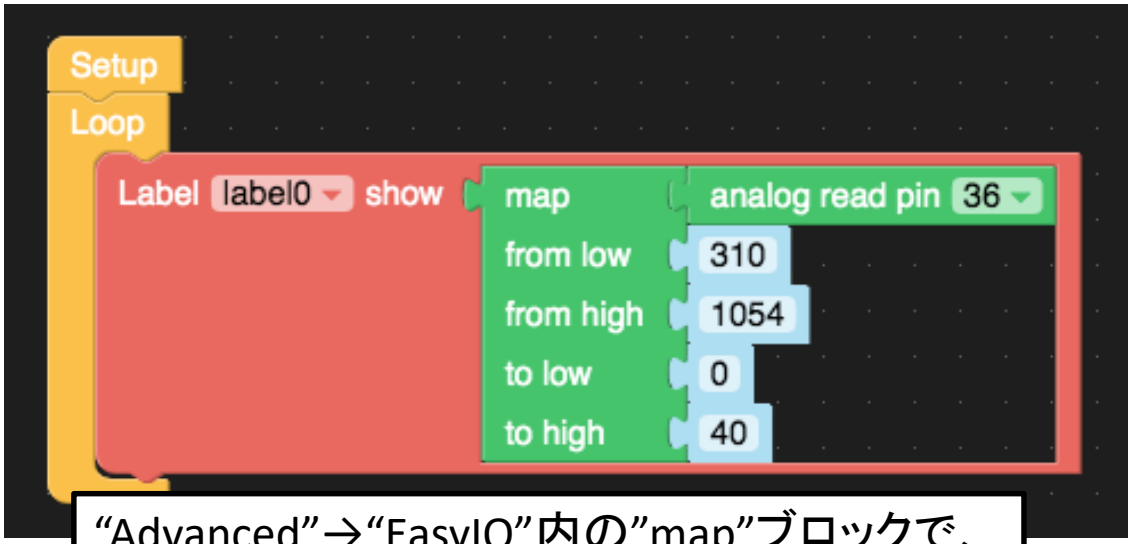
このプログラムを実行させて、スティックを傾けたときに表示される値の変化を確認する。
またY軸（IOピン35番に接続されている）の値もあわせて表示させてみる。

E2.温度センサ

- 温度に応じた電圧が出力される
- 理論計算は複雑だが、以下の値の範囲を変換すればよい

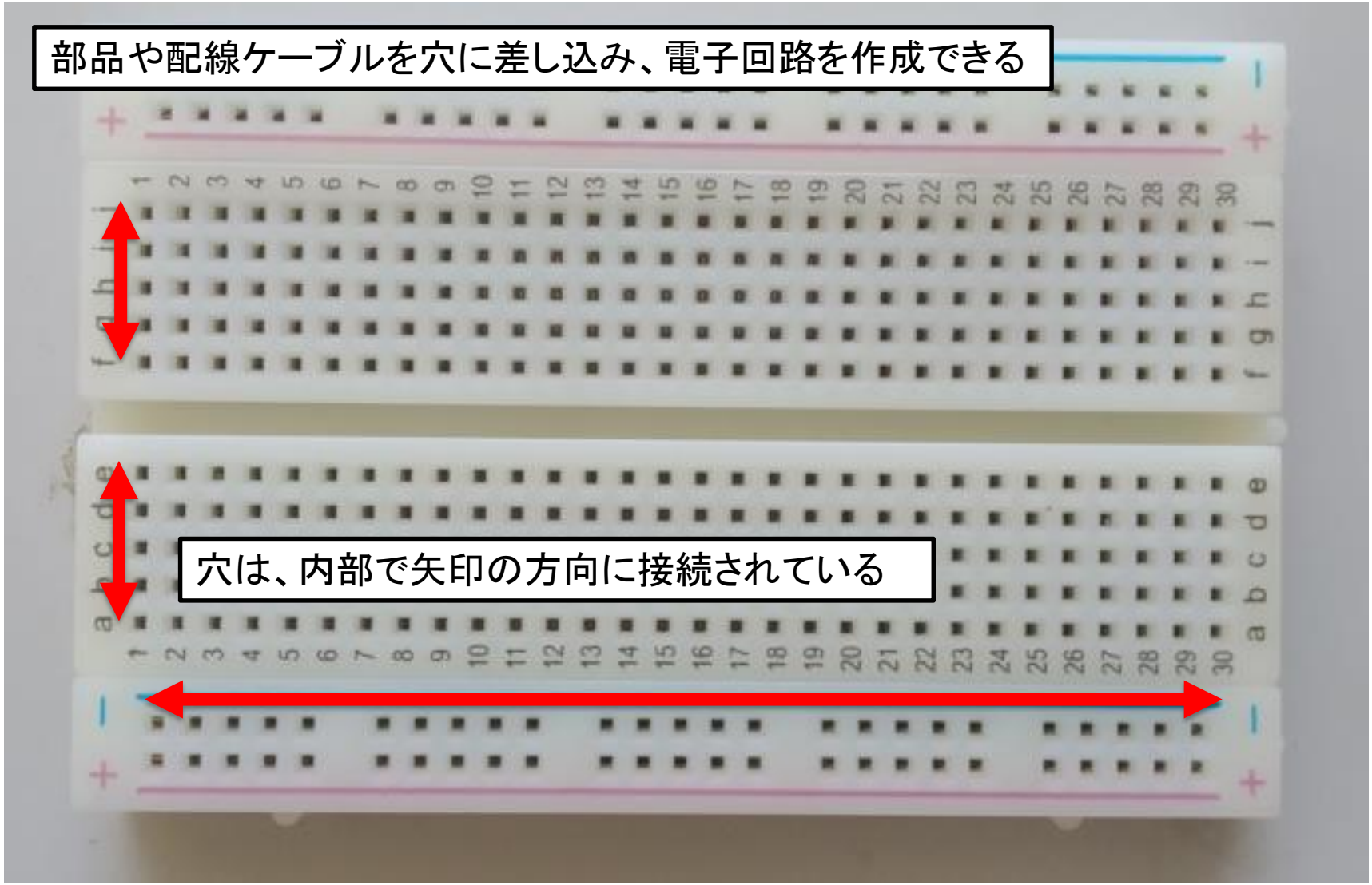
analog readの値：310～1054 ←→ 温度：0℃～40℃





“Advanced”→“EasyIO”内の”map”ブロックで、この変換を行える

E3.ブレッドボードの使い方



E3.LEDの接続と回路

長い方

赤線の箇所が内部で接続されていることに注意

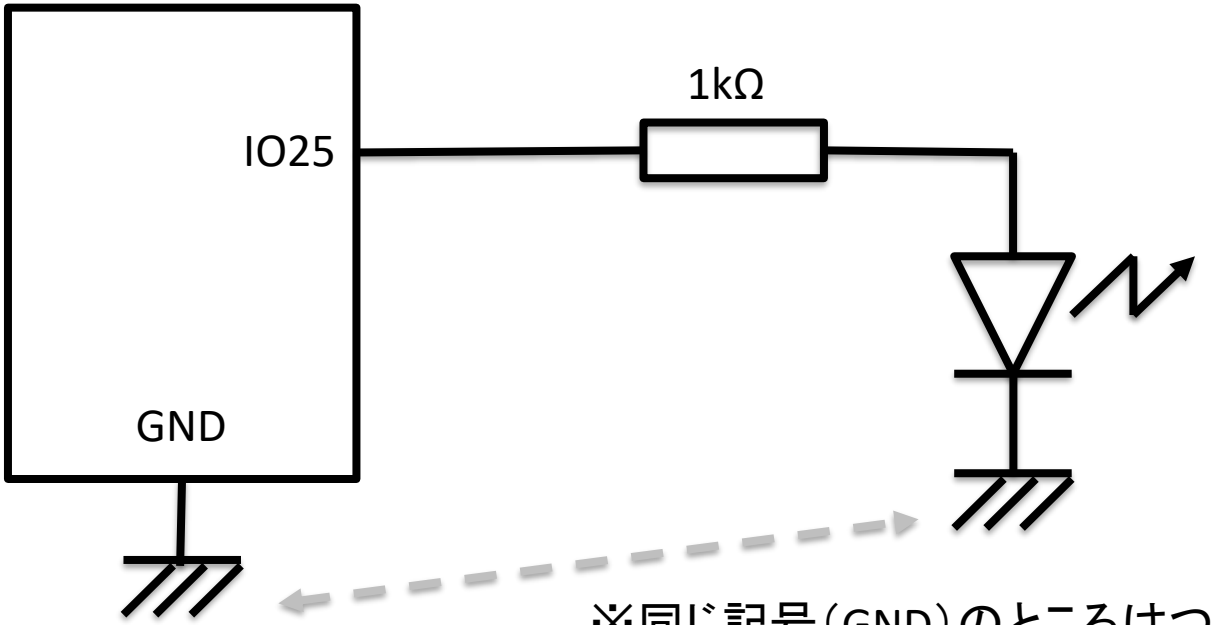
IOピン25番

電源のマイナス（GND; G）

配線ケーブルはどの色も機能は同じだが、
電源のマイナス（GND; Gと表記）側は
黒い配線ケーブルを使うのが通例

E3.LEDの接続と回路

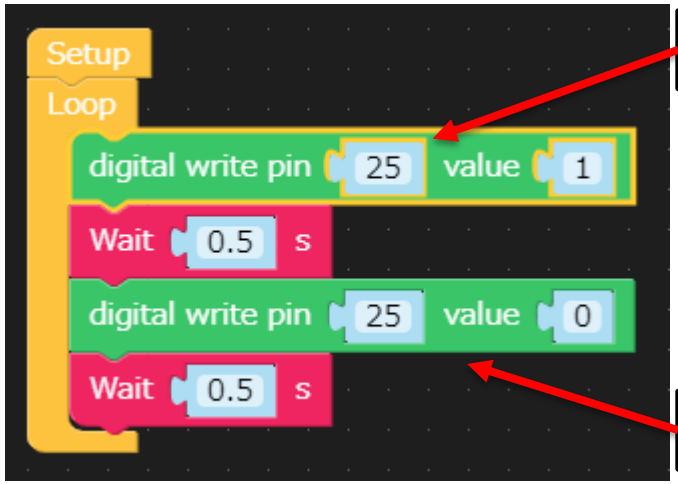
2b.でブレッドボード上に作成した回路の回路図



※同じ記号（GND）のところはつながっている

2b.で作成した回路が、上記の回路図となることを理解する

E3.LED点滅（Lチカ）



IOピン25番に”1”を出力＝3.3Vが出力され、LEDが点灯

IOピン25番に”0”を出力＝0Vが出力され、LEDが消灯

このプログラムを実行させ、LEDを点滅させる。
またwait()を変更してLEDの点滅周期を変える。

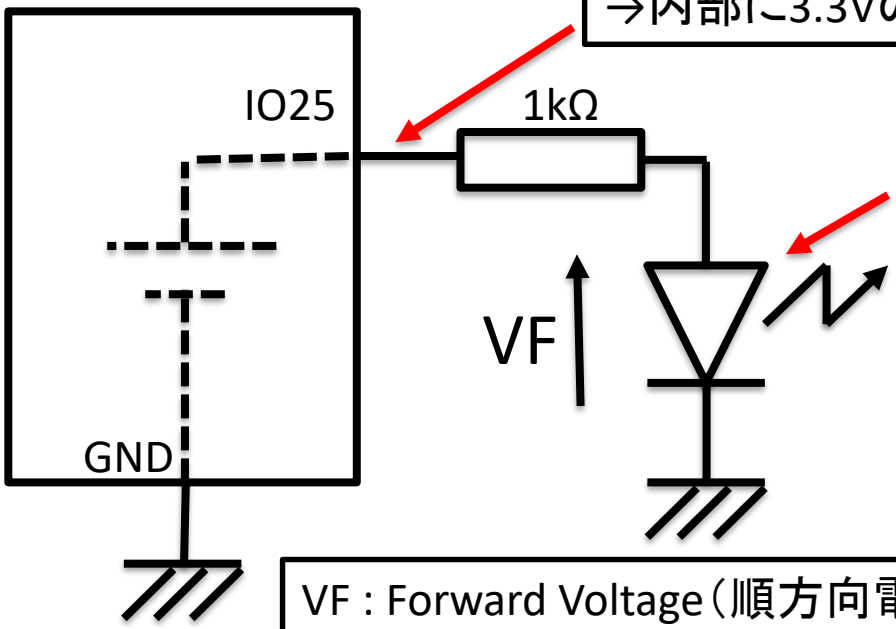
E4.複数のLチカ

直接、以下の動作をするPythonプログラムを記述し、実行させる

- 動作仕様
  - 赤LEDを、抵抗を通してIO25に接続
  - 緑LEDを、抵抗を通してIO26に接続
  - 以下の順序でLEDを点滅させる
    1. 赤のみ点灯（0.5秒間）
    2. 緑のみ点灯（0.5秒間）
    3. 赤と緑を点灯（0.5秒間）
    4. いずれも消灯（0.5秒間）

E5.LEDに流れる電流を求める

“1”のときは、電源電圧の3.3Vとなる
→内部に3.3Vの電圧源（点線）があると考えればよい



OptoSupply社OSDR5113A
http://akizukidenshi.com/catalog/g/gI-00624/
※データシートはここからダウンロードできる

VF : Forward Voltage（順方向電圧）
→この電圧の電圧源と考えればよい

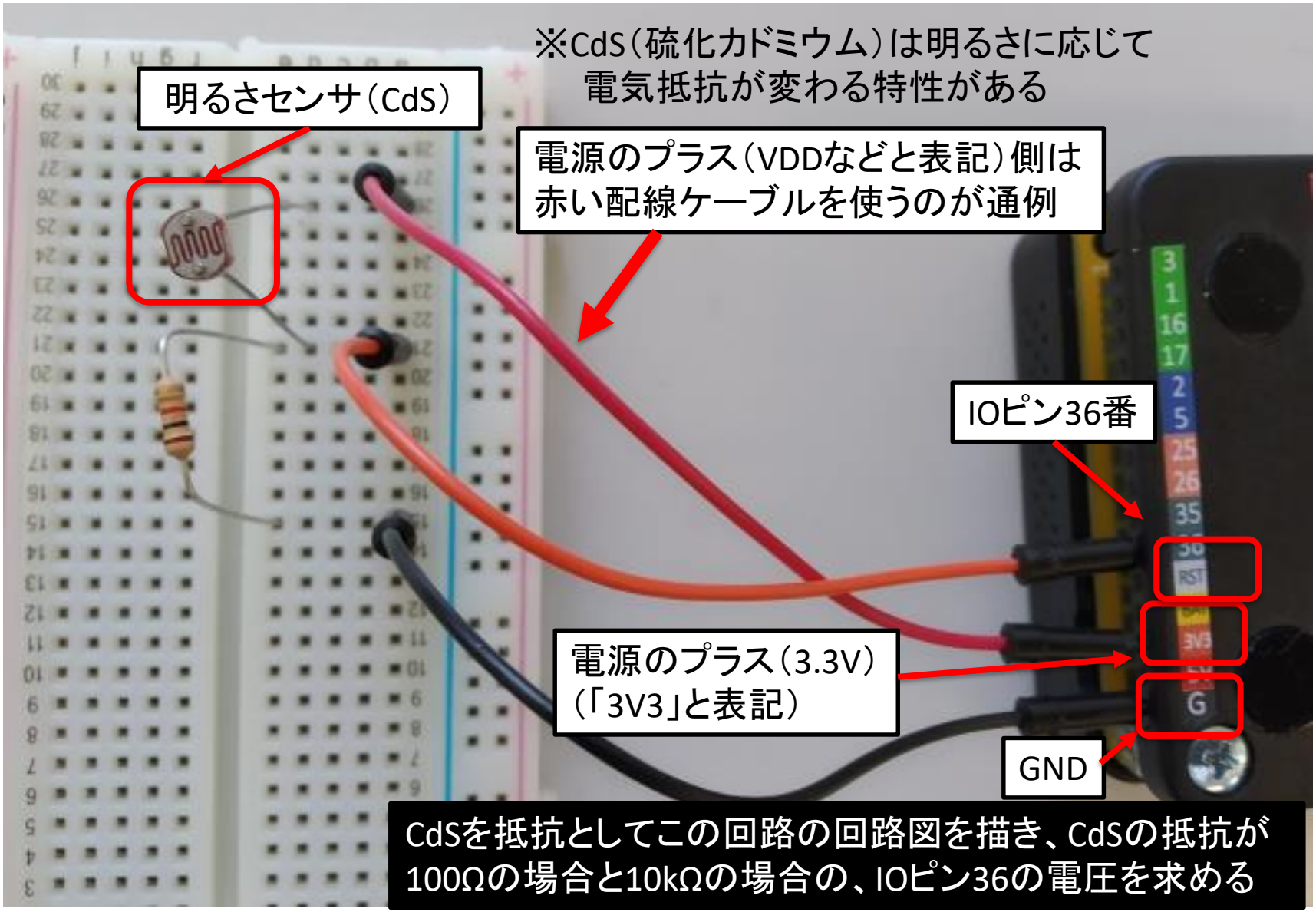
■Electrical -Optical Characteristics

Item	Symbol	Condition
DC Forward Voltage	V_F	I_F=20mA
DC Reverse Current	I_R	V_R=5V
Domi. Wavelength	λ_D	I_F=20mA
Luminous Intensity	Iv	I_F=20mA
50% Power Angle	$2\theta_{1/2}$	I_F=20mA

※英文のデータシートだが、得るべき情報は一部なので、慣れれば楽（になるはず）

LEDのデータシートをダウンロードし、VFを読み取り、それをもとにLEDに流れる電流を求める

E6.明るさセンサ（CdS）の接続と回路



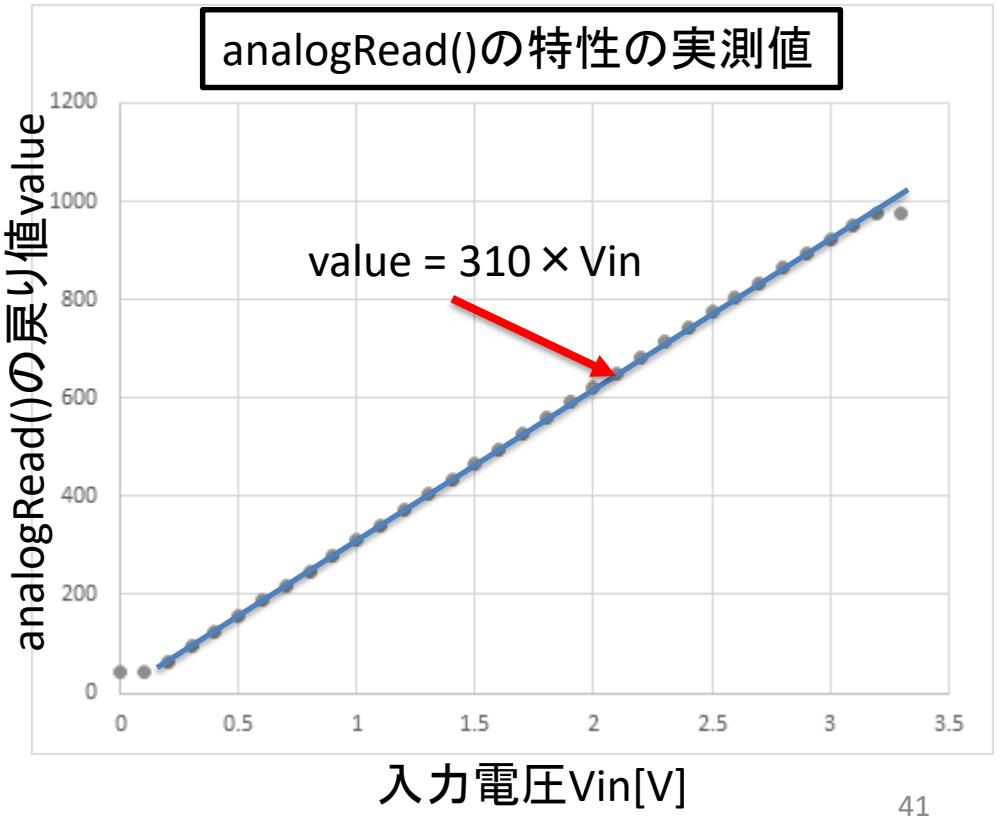
E6.明るさセンサ（CdS）の読み取り

IOピン36番の電圧を読み取り、label0に設定

このプログラムを実行させ、以下の3つの場合で値を読み取り、IOピン36の電圧を求める。
※電圧の計算は次項を参照
(a)手で覆って暗くした場合、(b)部屋の照明下、(c)スマホのライト等の照明を直接照射

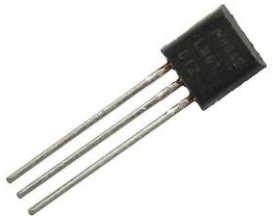
E7.明るさセンサ（CdS）の電圧の計算

- “analog read”で取得される値は、加える電圧[V]に（ほぼ）比例
  - 0V付近と3.3V付近は、やや線形性が崩れるが、ほぼ無視できる
- 今回の回路では、抵抗分圧から関係が求められる



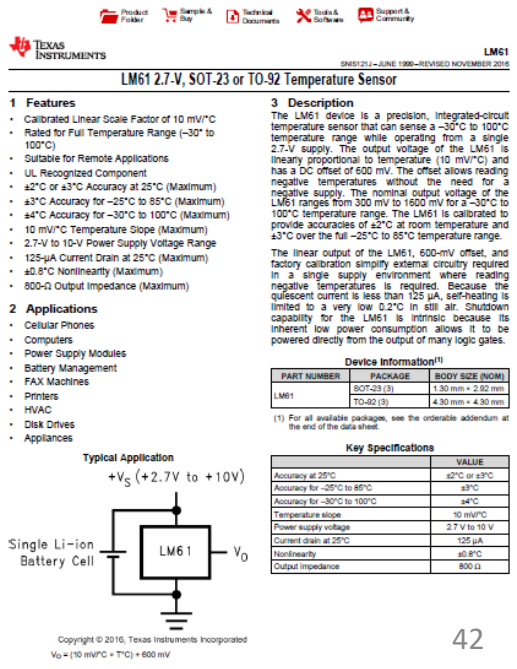
E8.温度センサの接続と回路

- 使用する温度センサ「LM61」のデータシートを型番で検索するなどしてダウンロードする
- データシートから、動作に必要な回路接続を読み取る（TO-92パッケージ）



http://akizukidenshi.com/catalog/g/gI-11160/

- 電源、GND、出力(VOUT)をブレッドボードでM5stackに接続
  - 電源は5Vを供給
  - VOUTはIO35へ



TEXAS INSTRUMENTS

LM61

SNIS125J – JUNE 1999 – REVISED NOVEMBER 2016

LM61 2.7-V, SOT-23 or TO-92 Temperature Sensor

1 Features

- Calibrated Linear Scale Factor of 10 mV/°C
- Rated for Full Temperature Range (–30° to 100°C)
- Suitable for Remote Applications
- UL Recognized Component
- ±2°C or ±3°C Accuracy at 25°C (Maximum)
- ±3°C Accuracy for –25°C to 85°C (Maximum)
- ±4°C Accuracy for –30°C to 100°C (Maximum)
- 10 mV/°C Temperature Slope (Maximum)
- 2.7-V to 10-V Power Supply Voltage Range
- 125-µA Current Drain at 25°C (Maximum)
- ±0.8°C Nonlinearity (Maximum)
- 800-Ω Output Impedance (Maximum)

2 Applications

- Cellular Phones
- Computers
- Power Supply Modules
- Battery Management
- FAX Machines
- Printers
- HVAC
- Disk Drives
- Appliances

3 Description

The LM61 device is a precision, integrated-circuit temperature sensor that can sense a –30°C to 100°C temperature range while operating from a single 2.7-V supply. The output voltage of the LM61 is linearly proportional to temperature (10 mV/°C) and has a DC offset of 600 mV. The offset allows reading negative temperatures without the need for a negative supply. The nominal output voltage of the LM61 ranges from 300 mV to 1600 mV for a –30°C to 100°C temperature range. The LM61 is calibrated to provide accuracies of ±2°C at room temperature and ±3°C over the full –25°C to 85°C temperature range.

The linear output of the LM61, 600-mV offset, and factory calibration simplify external circuitry required in a single supply environment where reading negative temperatures is required. Because the quiescent current is less than 125 µA, self-heating is limited to a very low 0.2°C in still air. Shutdown capability for the LM61 is intrinsic because its inherent low power consumption allows it to be powered directly from the output of many logic gates.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM61	SOT-23 (3)	1.30 mm × 2.92 mm
	TO-92 (3)	4.30 mm × 4.30 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Key Specifications

	VALUE
Accuracy at 25°C	±2°C or ±3°C
Accuracy for –25°C to 85°C	±3°C
Accuracy for –30°C to 100°C	±4°C
Temperature slope	10 mV/°C
Power supply voltage	2.7 V to 10 V
Current drain at 25°C	125 µA
Nonlinearity	±0.8°C
Output impedance	800 Ω

Typical Application

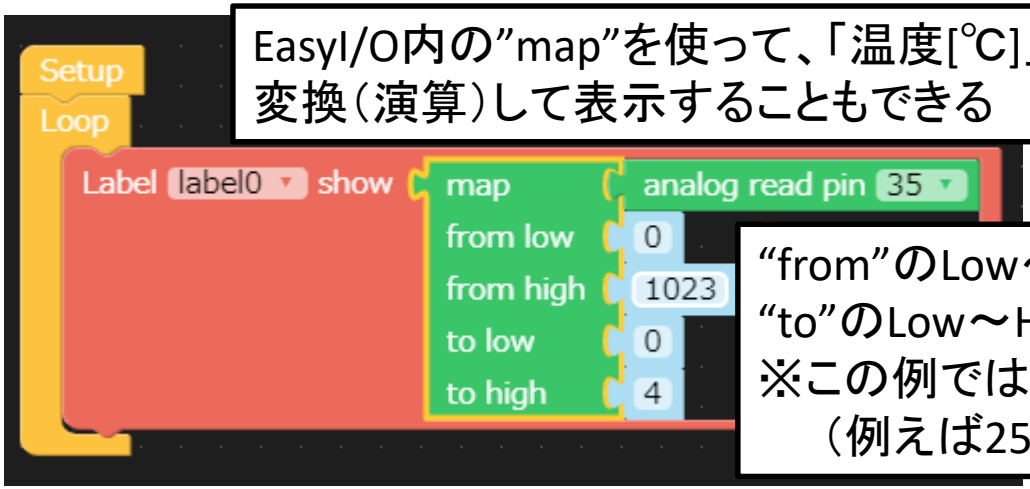
V_O = (10 mV/°C × T°C) + 600 mV

※データシートの読み取りに便利な主な英単語

- power Supply＝電源
- output＝出力
- ground＝接地（電源のマイナス(negative)側、「G」と表記
- bottom view＝底面図（部品を裏側（足側）から見た図）

E8.温度センサの読み取り

IOピン35番の電圧を読み取り、label0に設定



EasyI/O内の”map”を使って、「温度[℃]」の値に変換（演算）して表示することもできる

“from”のLow～Highの範囲の数値を、
“to”のLow～Highの範囲に（線形に）換算する
※この例では「0～1023」を「0～4」に換算
（例えば255→1.0、511→2.0、など）

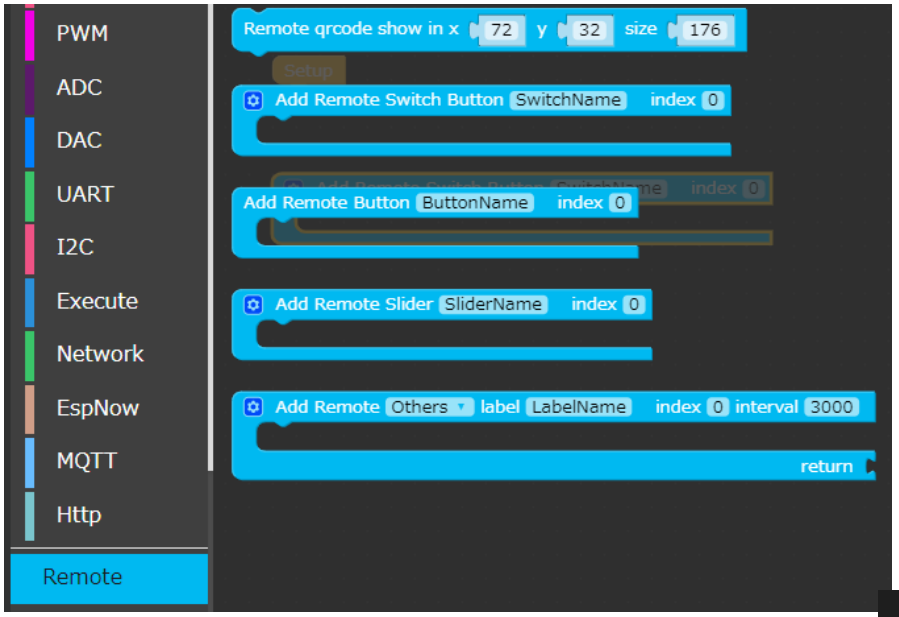
このプログラムをもとに、温度を[℃]単位で表示する

E9.インターネット連携：IoTへ

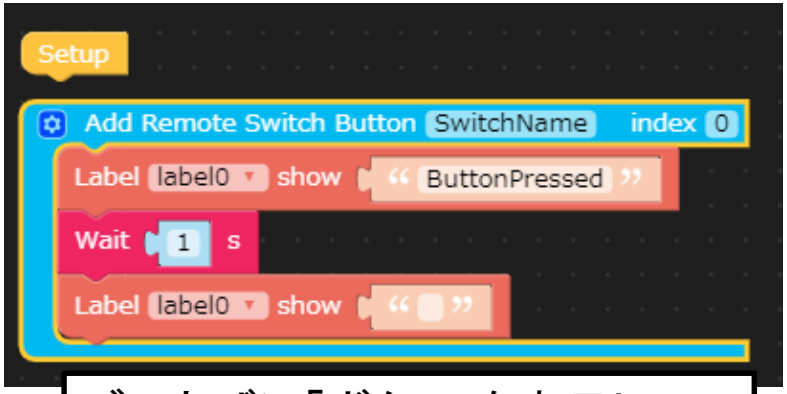
- PCやスマホだけでなく、「いろいろなもの」がインターネットにつながる世界
  ＝IoT (Internet of Things)　→　AIの出番

E9.インターネット連携：リモートボタン

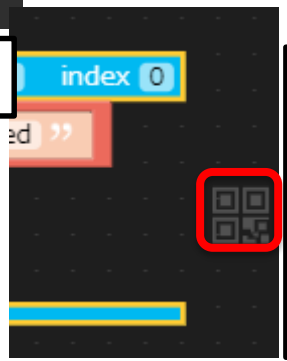
- スマホ（ブラウザ）からM5stackを操作してみる



“Remote”内に、スマホ連携の処理がある



ブラウザに「ボタン」を表示し、
それを押したらM5stackで処理を
行うプログラム



（ちょっと薄くて見にくいが）
「QRコード表示アイコン」で
QRコードを表示し、
それをスマホ等でスキャンして
ブラウザで制御ページを表示

このプログラムを実行し、スマホからM5stackを制御してみる

E9.インターネット連携：リモートラベル

- 逆にM5stackからスマホ(ブラウザ)を制御

スマホ(ブラウザ)側で表示を自動更新する間隔[ms]（この例では3秒）

スマホ(ブラウザ)上にラベルを表示し、そこに情報(この例では、ToFセンサで取得した距離)を表示する例

ToFセンサで取得した距離情報

変数の作成

変数の更新（値の取得）

変数(Variable)を作成し（この例では”distance”という変数）、それに必要な値を代入して表示などに使用することもできる

このプログラムを参考に、M5stackで取得したセンサ情報をスマホに表示してみる

E10.インターネット連携（Webスクレイピング）

※E10, E11の内容はEdelWorksの古川光氏の協力で作成されました

- Webページを取得し、その中の情報を抜き出して利用する（スクレイピング）
  - 例として、無料Webサービスから天気情報を取得

https://openweathermap.org/

E10.インターネット連携（Webスクレイピング）

- 取得したAPI Keyと都市名を使って、以下のURLで天気情報などを取得できる（JSON形式）

http://api.openweathermap.org/data/2.5/weather?q=[都市名]&appid=[APIキー]


```
{
 - coord: {
      lon: 136.66,
      lat: 36.56
   },
 - weather: [
    - {
         id: 520,
         main: "Rain",
         description: "light intensity shower rain",
         icon: "09d"
      }
   ],
   base: "stations",
 - main: {
      temp: 296.15,
      pressure: 1017
      humidity: 83,
      temp_min: 295.
      temp_max: 297.
   },
   visibility: 10000,
 - wind: {
      speed: 2.1,
      deg: 40
   },
 - clouds: {
      all: 75
   },
   dt: 1569716359,
 - sys: {
      type: 1,
      id: 8017,
      message: 0.0074,
      country: "JP",
      sunrise: 1569703568,
      sunset: 1569746499
   },
   timezone: 32400,
   id: 1857470,
   name: "Kanazawa",
   cod: 200
}
```


ここに現在の天気が書いてある

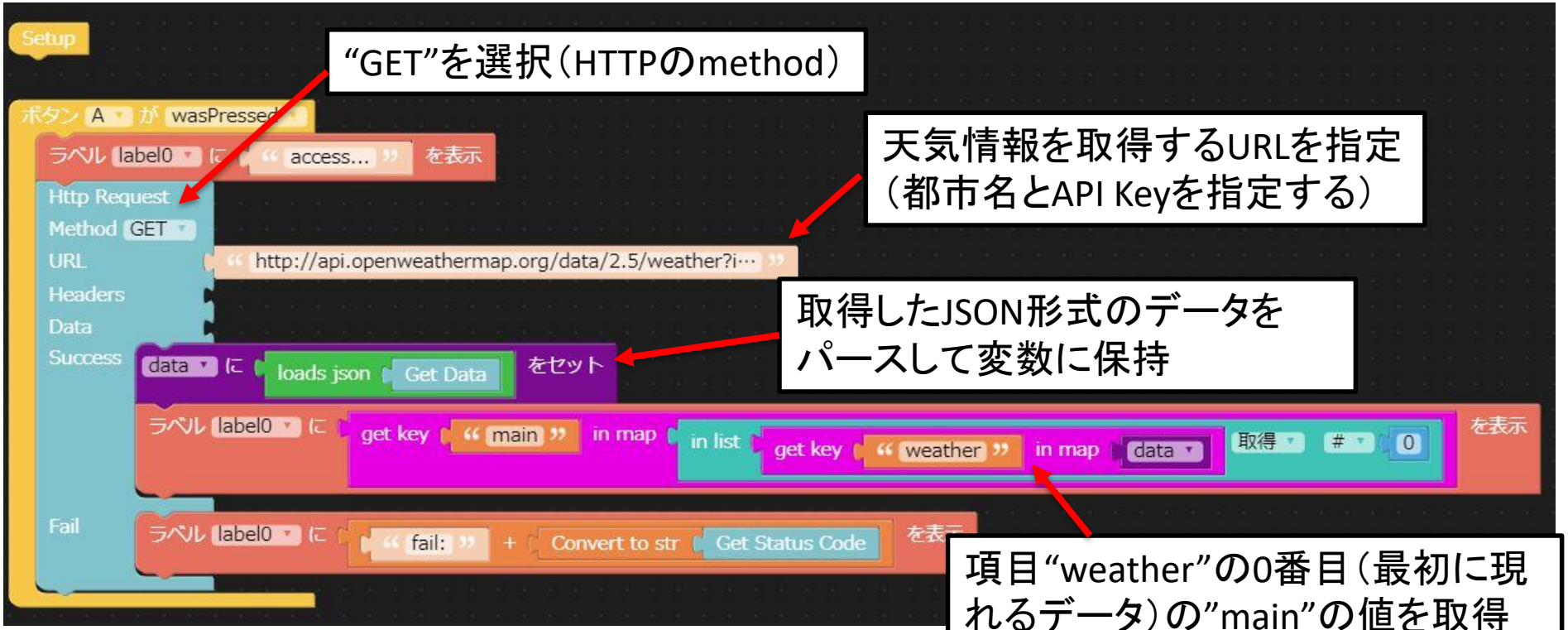
※[都市名]のところは、kanazawa,jp などが入る
※APIキーは、先ほど記録したAPI Key

このURLへアクセスすると、このようなデータが返送される（JSON形式）

※JSON（JavaScript Object Notation）
Webからのデータの取得などでよく使われるデータ形式
見れば、なんとなく意味はわかるはず
詳しくは→https://www.json.org/json-ja.html

E10.インターネット連携（Webスクレイピング）

- 例として以下のプログラムで、指定したURLからJSONデータ取得→そこからmain部分を切り出し（パース）→それをlabel0（置いておく）で表示



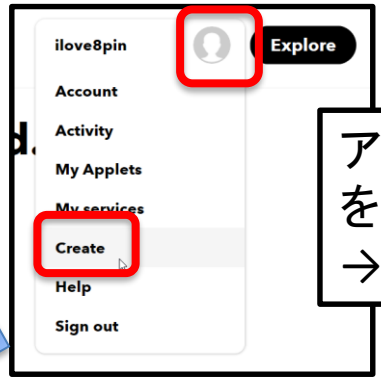
このプログラムを参考に、M5stackで取得した天気や気温などの情報を表示してみる

E11.インターネット連携（IFTTT：準備1）

- Webサービスどうしを連携するサービスIFTTT
  ※IFTTT=IF This Then That：条件で行う動作を定義
- 例として、「M5stackからWebアクセスしてGoogleスプレッドシートに情報を記録する」という動作をさせてみる

https://ifttt.com/

アカウントを作成



アカウント（顔マーク）をクリック
→”Create”を選択

動作を起こす条件を定義する（クリック）

使うサービスとして“webhooks”を検索して選択

E11.インターネット連携（IFTTT：準備2）

条件「M5stackからWebアクセス」のイベントを設定する

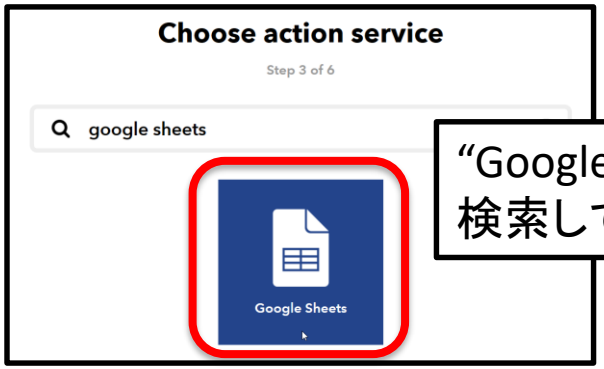
「Webアクセスを受けたとき」を条件(trigger)に設定する

このイベント（Webアクセスを受けたとき）に適当な名前をつけておく
（このイベント名はあとで使う）

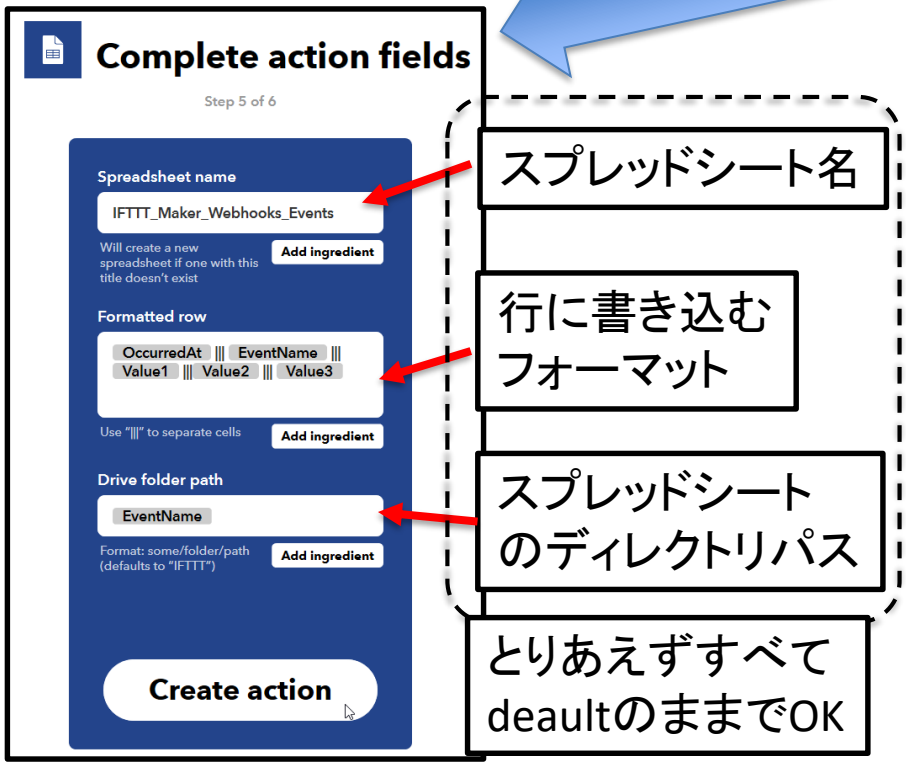
E11.インターネット連携（IFTTT：準備3）

条件（M5stackからアクセス）が起こった時に行う動作（GoogleSpreadsheetへ記録）を指定する

条件が起こった(trigger)ときに
行う動作を指定する



“Google sheets”を
検索して指定



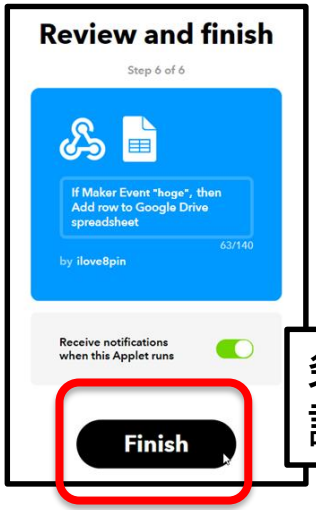
スプレッドシート名

行に書き込む
フォーマット

スプレッドシート
のディレクトリパス

とりあえずすべて
deaultのままでOK

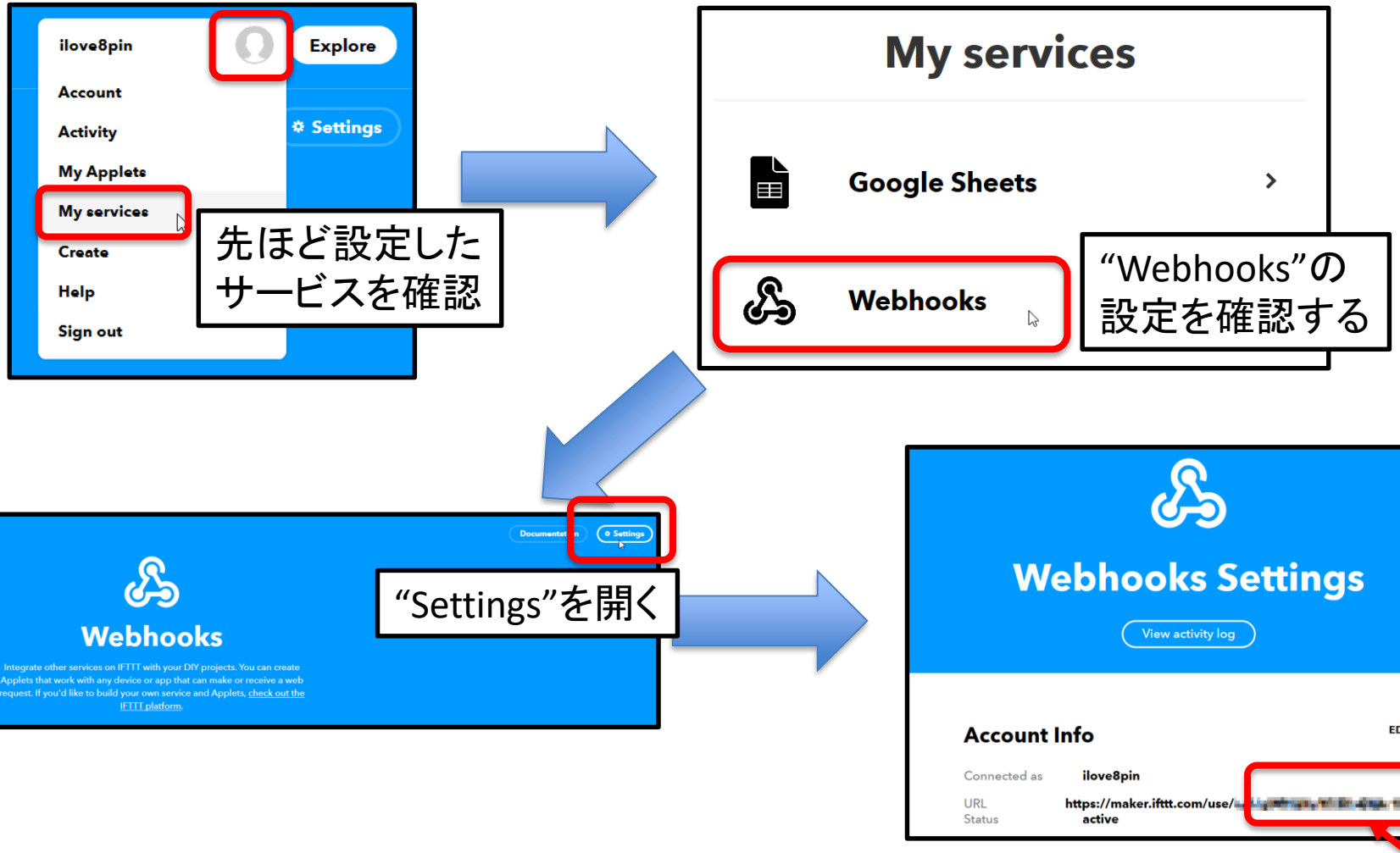
接続の確認やアクセス許可
などを設定（OKで進む）
※Actionは”Add row to spreadsheet”を指定



条件と動作の
設定を完了

E11.インターネット連携（IFTTT：準備4）

M5stackからWebアクセスするためのAPI Keyを取得する

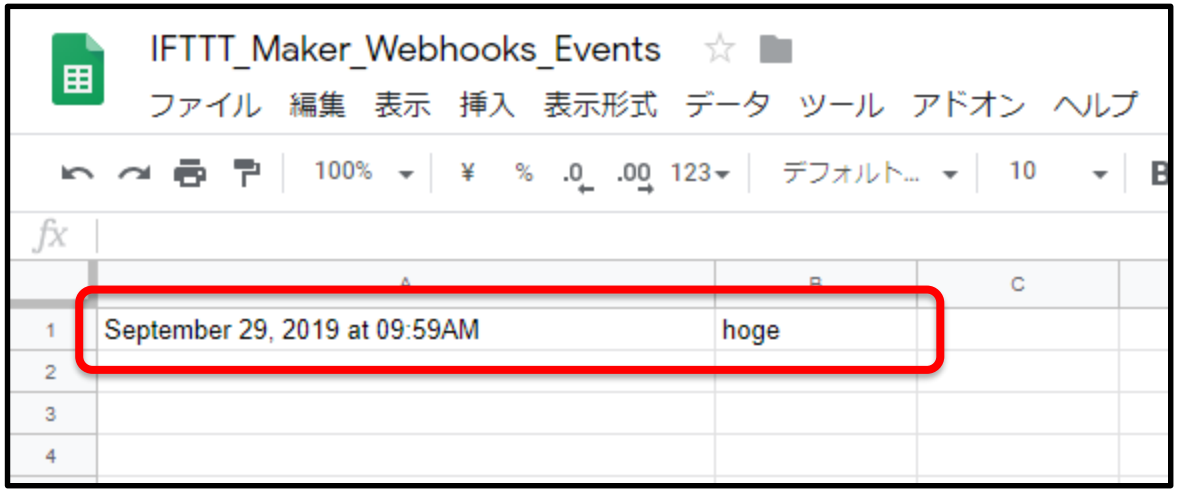


E11.インターネット連携（IFTTTアクセス）

- 以下のURLで、Googleスプレッドシートに記録されていく（はず）のを確認する

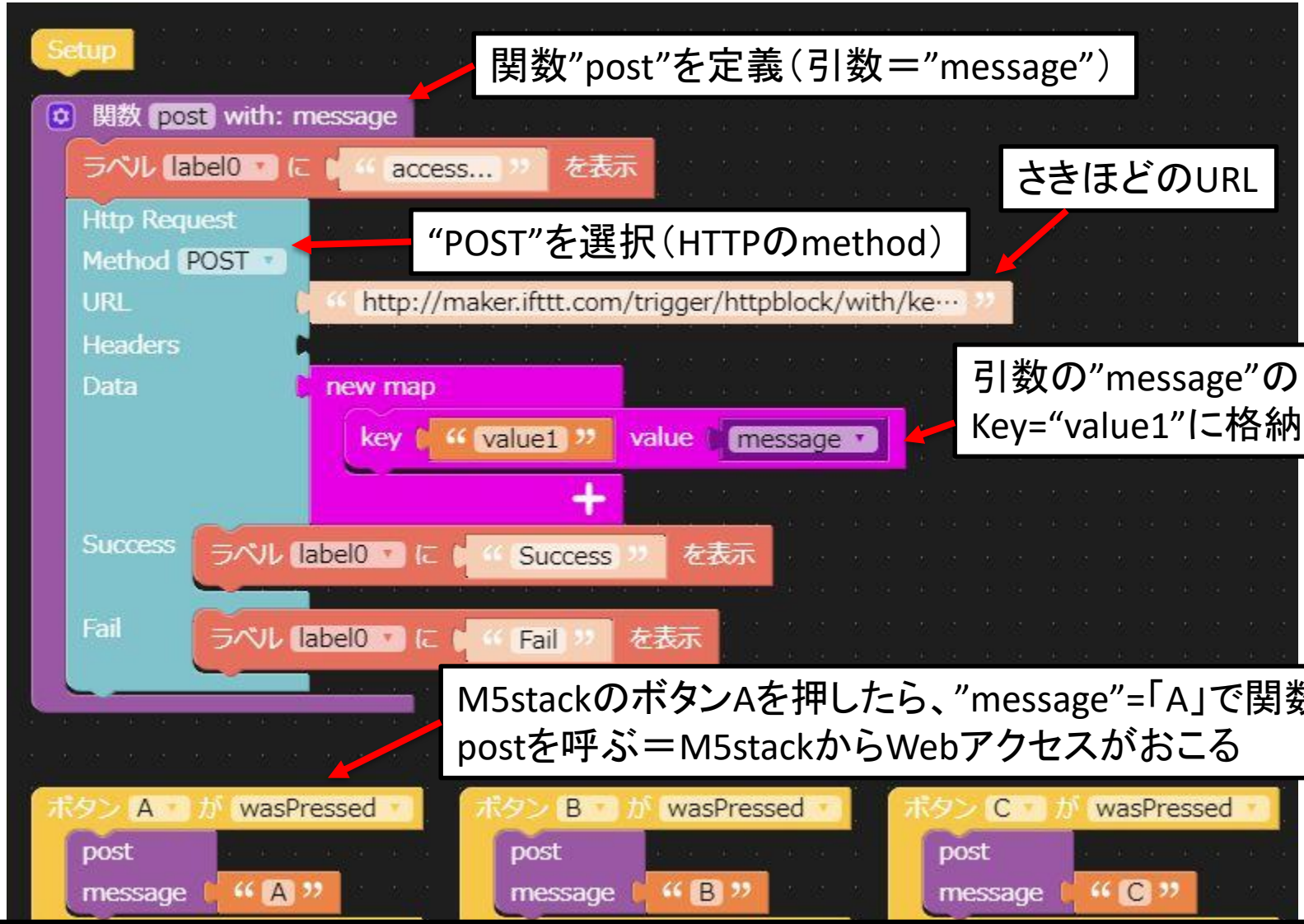
http://maker.ifttt.com/trigger/[イベント名]/with/key/[APIキー]

※[イベント名]のところは「準備2」(p.47)で指定したイベント名
※APIキーは、先ほど記録したAPI Key



まずはPCやスマホのWebブラウザから、このURLを直接入力し、Googleスプレッドシートに記録されるかを確認する

E11.インターネット連携（IFTTTアクセス）



このプログラムを参考に、M5stackからIFTTT経由でGoogleスプレッドシートに記録をしてみる