センサを活用したIoT
アプリケーション開発技術

ポリテクセンター熊本

本セミナーの目的

- まだまだ新規性の高い「IoTシステム」を，いきなり本番稼働まで想定して設計・導入していくには，不確実性が高く効果的な導入となるかは不明確である.

- そこで， IoT事例の多くはPoC(Proof of Concept・概念実証)からスタートし，効果測定を行った後に本番稼働を見越したフェーズへ移行していく事が多い

- 今回はPoC/プロトタイピング(試作開発)/社内治具としての利用を想定し，短期間かつ低予算でIoTシステムを開発する為の要素技術を学ぶ

本セミナーの目標

- IoTの基本である「可視化」と「通知」の実現を目標とする

- 目標
  - IoTデバイスからのセンサ値の取得が行える
  - 取得したセンサ値を元にグラフを作成し「可視化」する
  - 取得したセンサ値を元に閾値を上回った／下回った際に「通知」する
    - IoTデバイスを用いた**物理空間にいる人**への通知
    - メール等を用いた**特定の人物(の持つ端末)**への通知

本セミナーで使用する機材

- 機材
  - PC
    - Win10 1904以降
    - Core i5 – 6世代以上
    - メモリ：8GB程度
    - ストレージ：40GB空き容量
  - Anrdroid タブレット
  - WioNode
  - Grove LEDモジュール
  - Grove 光センサモジュール
  - その他のGroveモジュール一式
- Visual Studio 2019
- 外部サービス
  - IFTTT
  - Ambient
- WiFi
  - 2.4GHzであること
  - DHCPが有効になっていること

「組込み」と「IoT」

- IoT：Internet Of Thingsの略
  - 直訳すると「モノのインターネット」
  - 家電や生産設備など各種製品がインターネットと繋がる事で新たな付加価値を生み出す
- デバイス側では組込みの領域、Web側ではWebプログラミングやデータベース、ネットワーク等の領域
  - 加えてココにセキュリティなども
  - 求められる知識・技術の幅が非常に広い
- 組込み屋も今後はネットワークやWeb関連の知識・技術が求められる

IoT事例の紹介

IoTの事例：みまもりホットライン

- http://www.mimamori.net/（2001年にスタート）

IoTの事例：コマツのIoT(KOMTRAX)

コマツ IoT センタ

「なるほど、これがスマートコンストラクションか」
そんな驚きと、発見に出会えます。

コマツＩｏＴセンタでは、最新のICT建機や話題のドローン(無人ヘリ)の性能を目の当たりに出来る
説明会やデモンストレーション見学会など、
「スマートコンストラクション」をより深く理解していただくための様々な体験をご用意しています。

■スマートコンストラクション説明会

スマートコンストラクションの概要を講師が丁寧に解説。スマートコンストラクションが現場の何をどう変えるのかをわかりやすくお伝えします。

■デモンストレーション見学

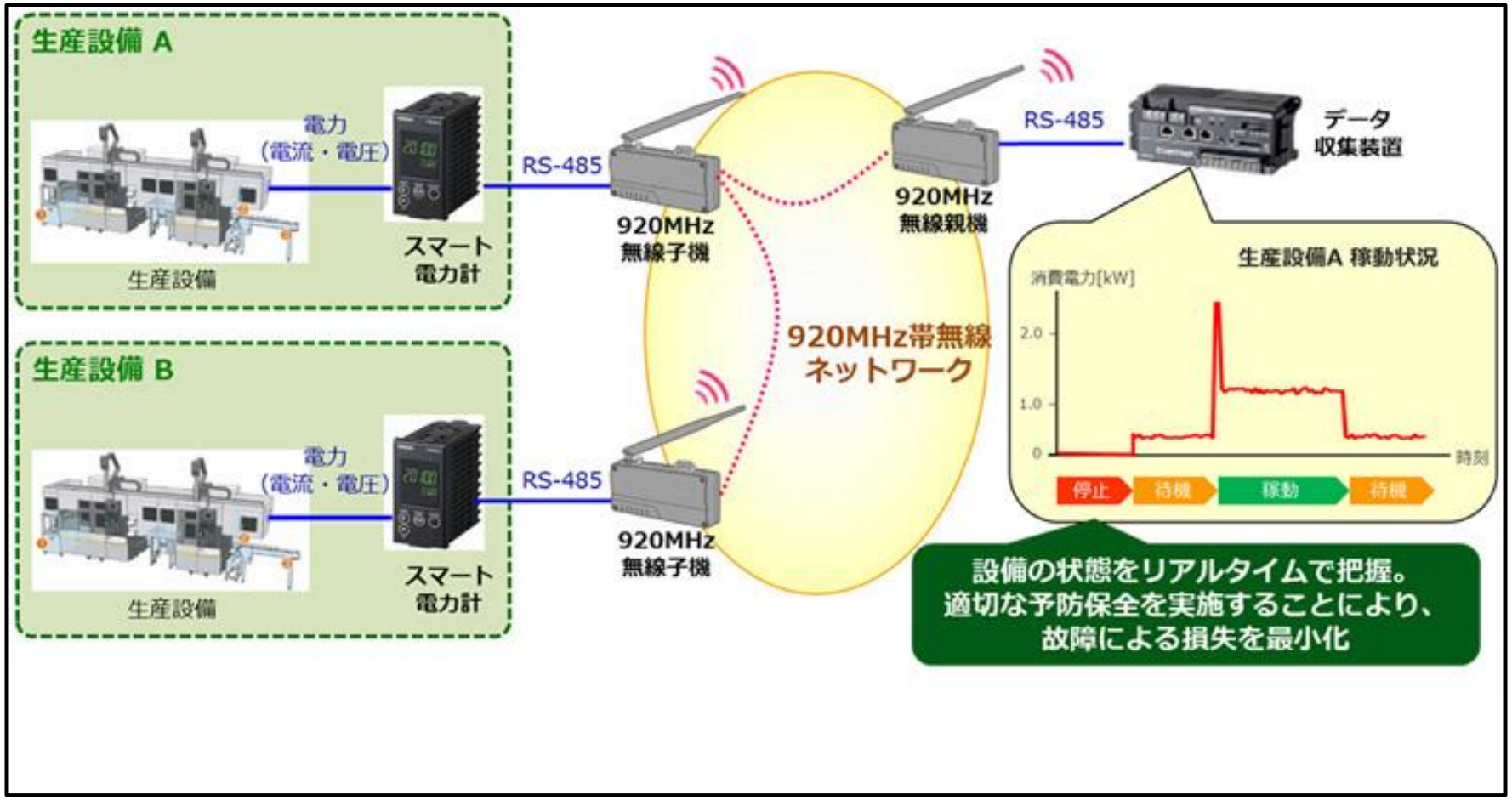
ボタン操作ひとつでドローン(無人ヘリ)が現況を測量するさまや、ICT建機(油圧ショベル・ブルドーザ)が実際に施工する様子を見学します。

■ICT建機試乗

油圧ショベルPC200i、ブルドーザD61PXiに体験試乗し、ICT建機の優れた操作性を実感していただきます。

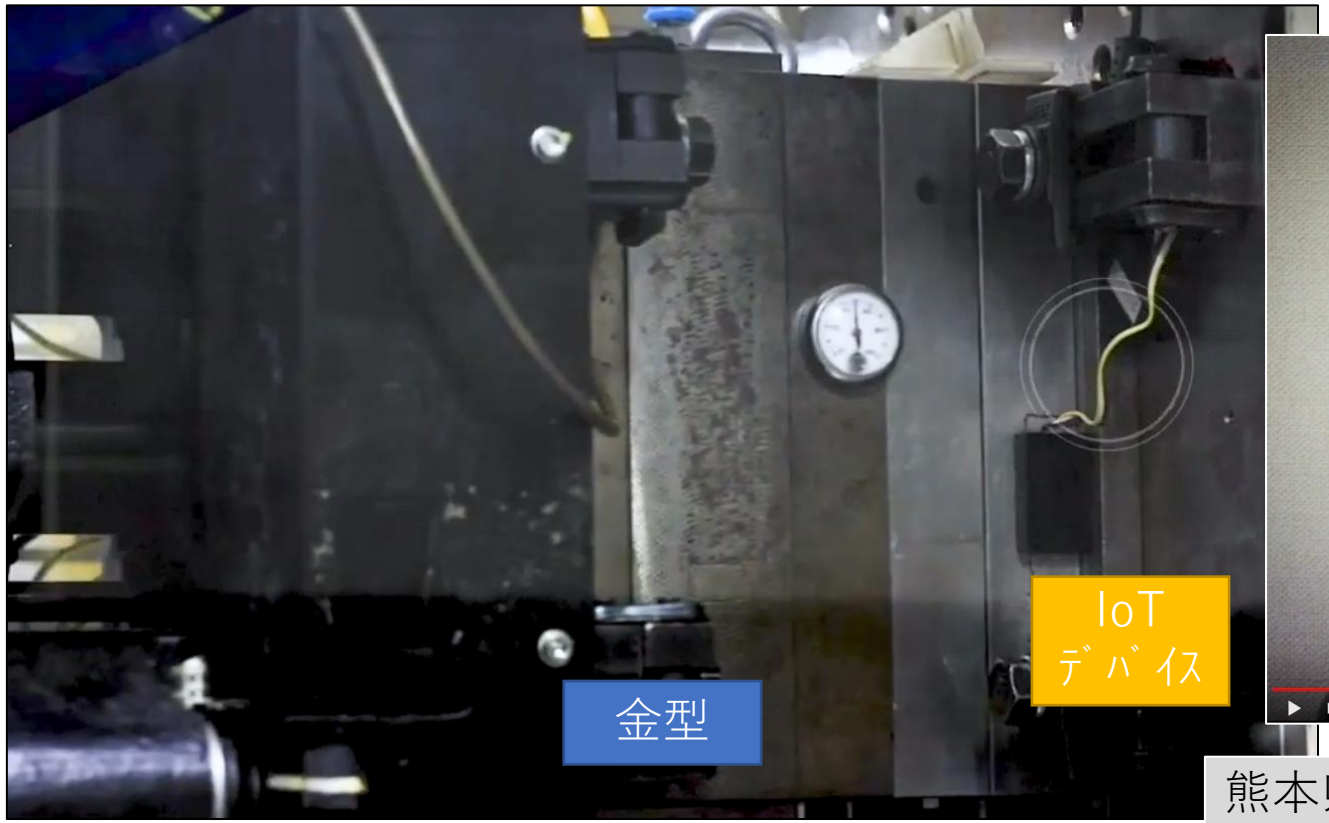
http://smartconstruction.komatsu/seminar.html （2001年スタート）

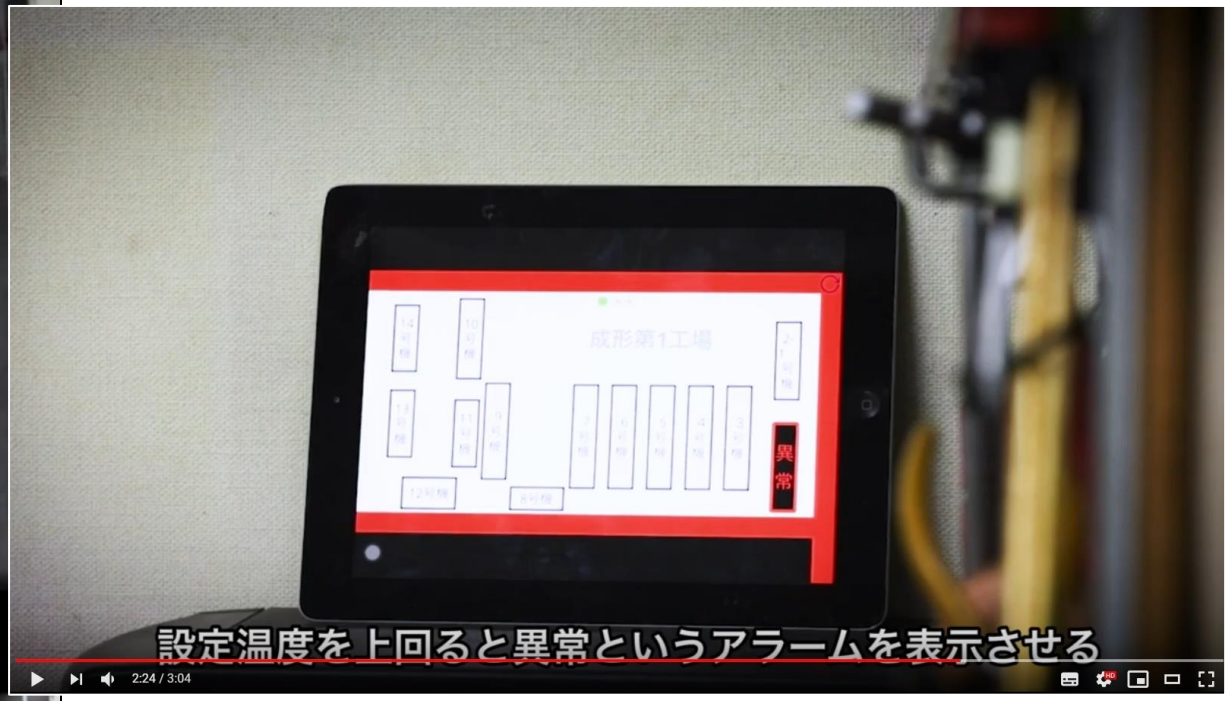
IoT事例：OKIのIoT



https://www.oki.com/jp/iot/doc/2016/16vol_02.html

IoT事例：射出成型機の金型温度モニタリング





- https://project.nikkeibp.co.jp/atclmono/vision/042400027/
- https://youtu.be/NscvrwHJQQ4

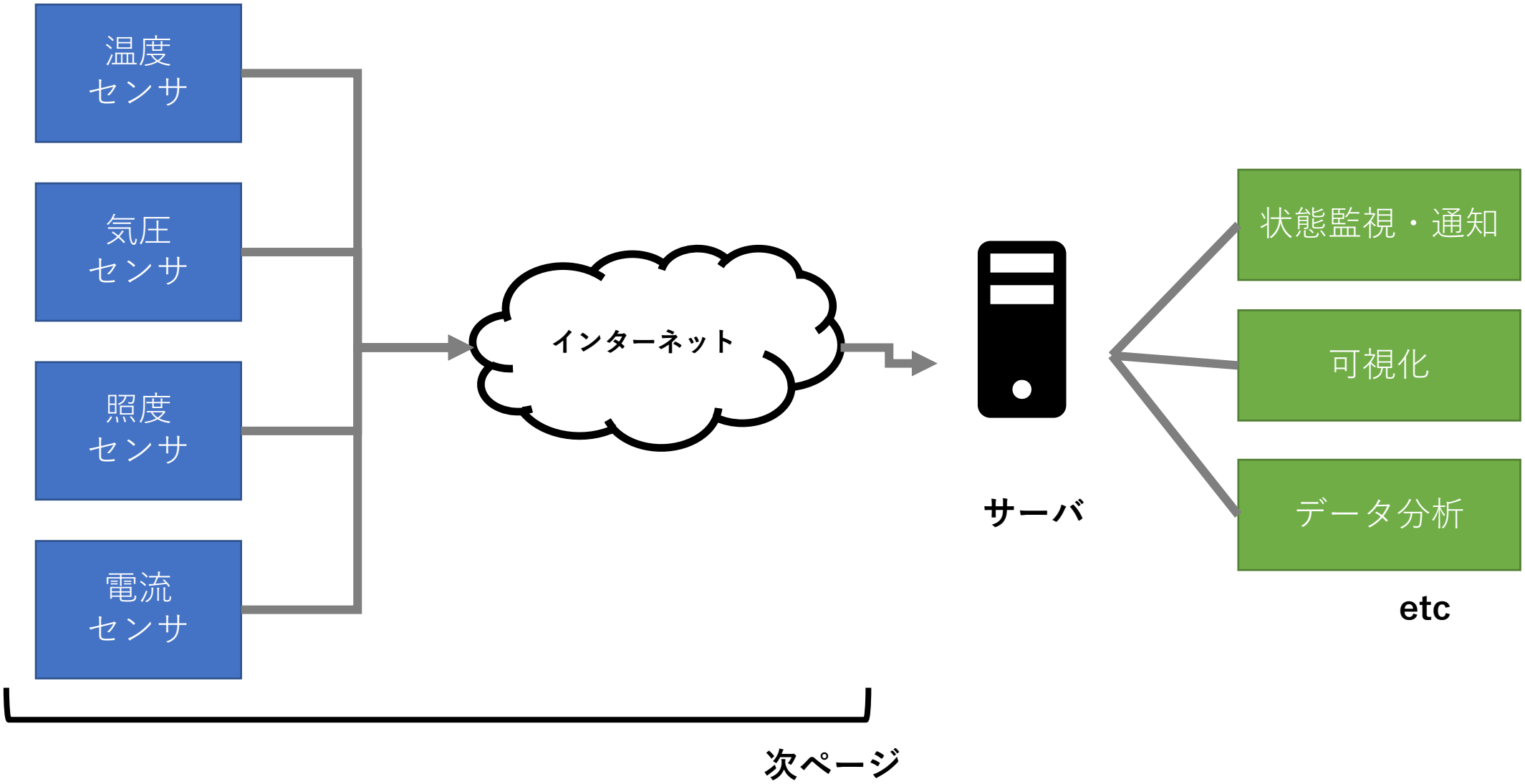
熊本県の樹脂メーカー、小森プラスチック工業の方です。射出成形機の金型温度をモニタリングする装置を開発し、これを工場の射出成形機に実装しました。これによって、1日6回、工場内を巡回しながら28台のアナログ温度計のデータを読み取る作業が不要になり、1日に180分も担当者の作業時間を短縮できました。

主流はセンシングとデータ活用

- 世の中のIoTの基本はセンシング
- センシングしたデータから何を生み出すか？
  - より確かな品質保証
  - 監視業務の工数削減
    - 重大なアラートの即時通知による被害提言
  - 需要予測・ジャストインタイム　　などなど

- 新しい市場であり、まずは作って検証が大事！
  - PoC(概念実証)・プロトタイピング

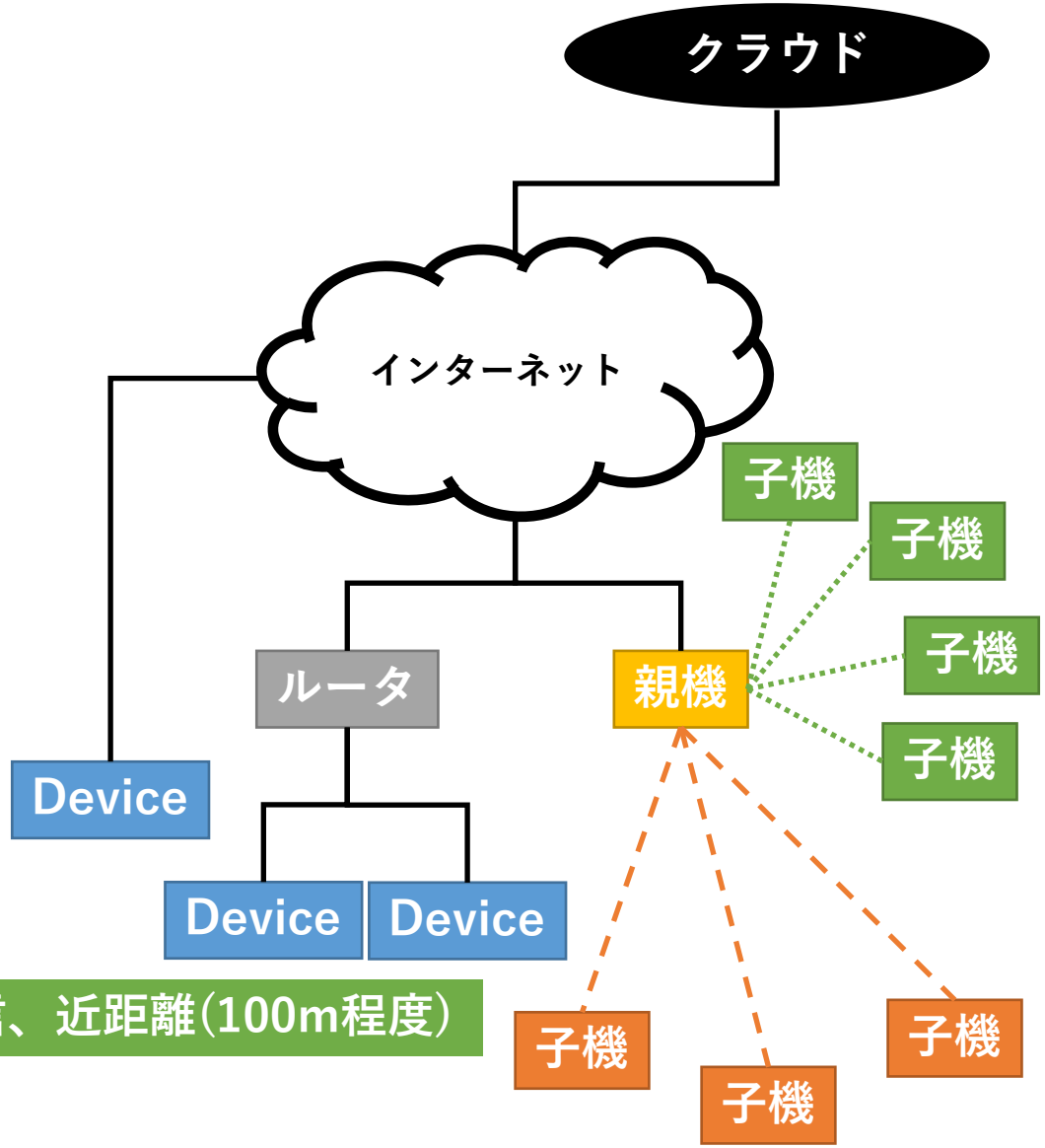
IoTシステムのイメージ



エッジ側のシステム構成

- IoTデバイスから直接Webに繋がる
  - 3G/4G/LTEといった携帯回線の活用
  - WiFi/Ethernetの活用

- センサデバイス(子機)の情報を親機に一旦集約してWebに繋がる
  - 親機はRaspberryPiのようなLinuxが動作するボードを用いる事が多い
  - 親機ー子機間の通信
    - WiFi/Ehernet/Bluetooth
    - Zigbee/twe-lite
    - LoRa/SigFox/Wi-SUN



2.4GHz帯：高速通信、近距離(100m程度)

920MHz帯: (比較的)低速な通信、長距離(1km程度)

今回は 「IoTデバイスからWiFi経由で直接Webに繋がるパターン」をやってみます

クラウドコンピューティングの分類

SaaS
Software as a Serice

PaaS
Platform as a Serice

IaaS
Infrastructure as a Serice

・安価な傾向
・自由度は低い
・求められるスキルは低め

・自由度は高い
・要スキル

例：Gmail, Microsoft365,
Dropbox, サイボウズ, Ambient

例：AWS Lamda, AWS IoT Core
Azure IoT Hub,他

例：AWS EC2, Azure VM他

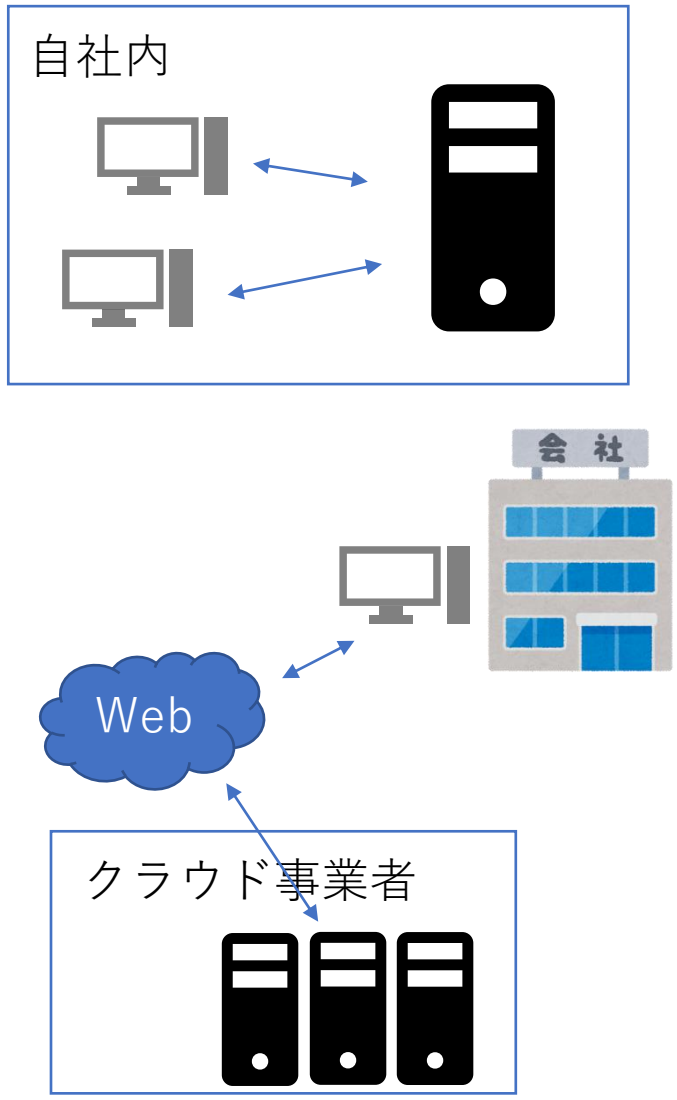
クラウド上で提供されるソフトウェア
特定の用途を実現する為のソフトウェアが完成された状態で提供される

クラウド上で提供される実行環境
アプリやDB等が動作する実行環境などが提供される

仮想サーバや外部ストレージ,
ファイアウォール等のインターネット上のサービスとして提供する

[参考] オンプレミスとクラウド

- サーバをどこで運用管理するか？の違い
- オンプレミス
  - 会社の敷地内にサーバを置いての運用
  - 従来からの運用方法
  - 秘匿性が非常に高いデータなどの扱い向き
    - 自社内の閉じたネットワーク内で管理できる
  - ハードウェアの購入など大きな初期費用が掛かる
  - 障害対応時に、自社で行う必要あり
- クラウド
  - サーバを貸し出ししてくれる事業者がおり、そのサーバを活用する
    - ハードウェアはクラウド事業者のデータセンターに置いての運用
    - サーバ構築やWebアプリ開発等は、Web経由で実施
  - 月額制／年額制
  - 初期費用は一般的に無料。スケールアップ・スケールダウンもコストが変わるだけで基本的に自由が効くことが多い
  - 障害対応時にクラウドの事業者が行う

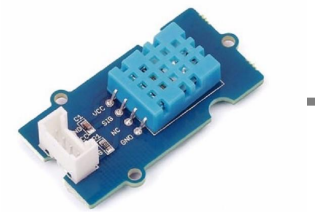


IoTは総合格闘技

- 技術的に非常に幅広い分野のスキルが必要
  - Web、組込み、ネットワーク、ビッグデータ、AIなどなど

今回実現するシステム

モニタリングによる通知と可視化

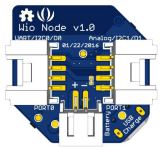


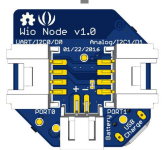
温湿度センサ

光センサ

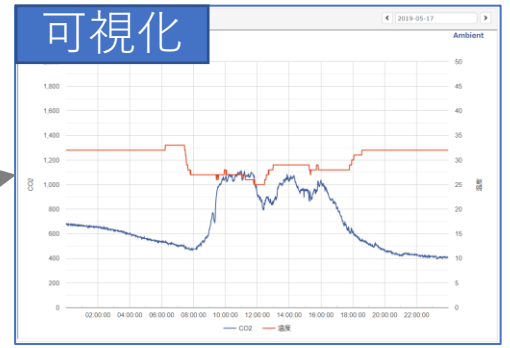
LED

ブザー



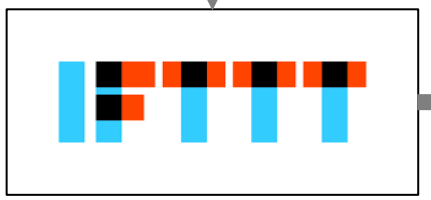


インターネット

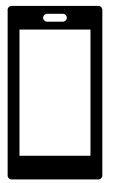


可視化

Ambient

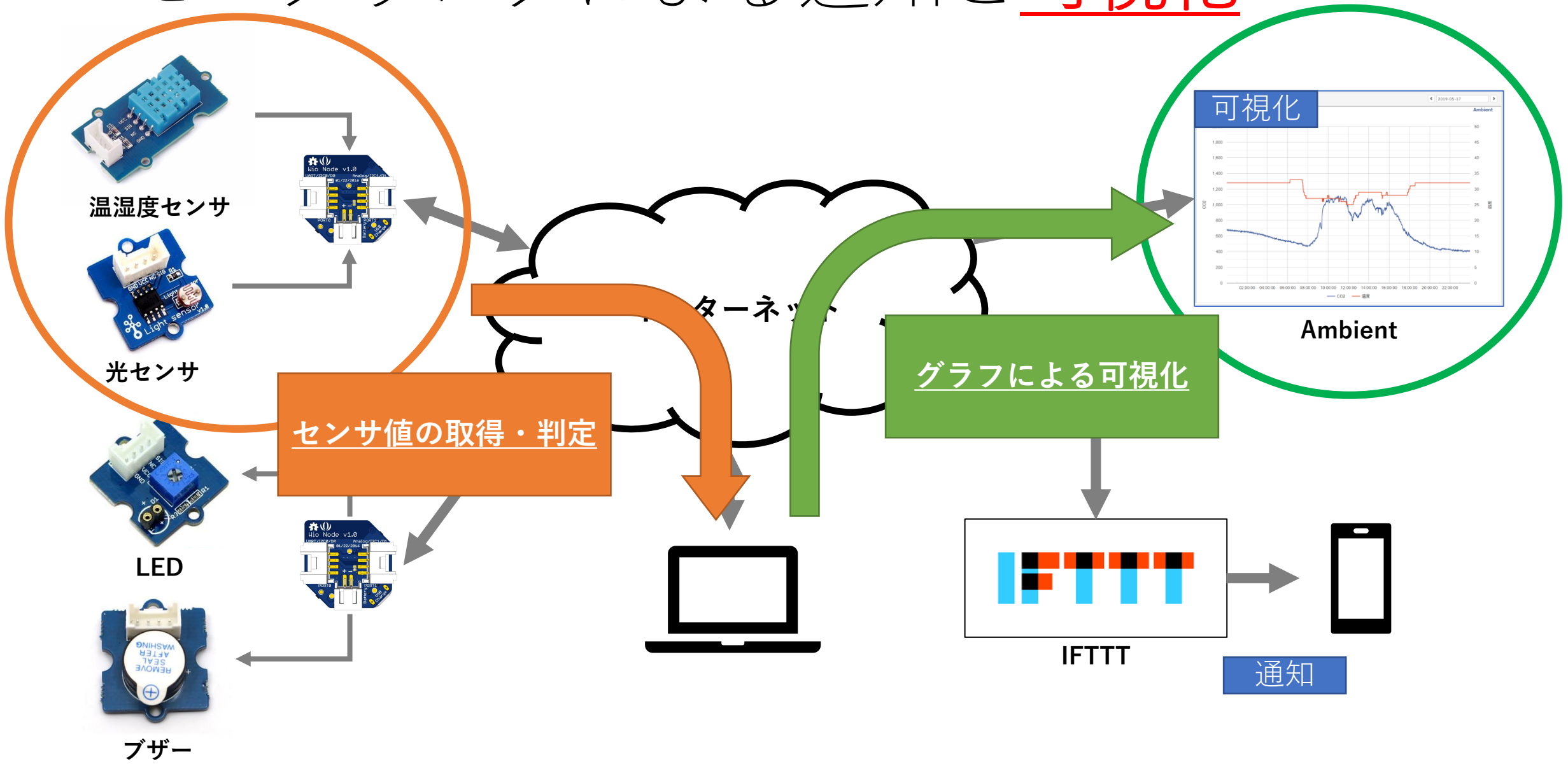


IFTTT

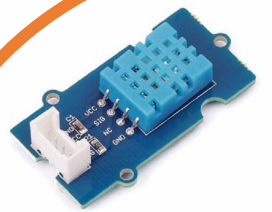


通知

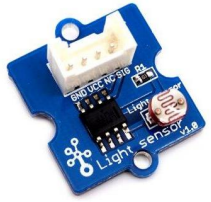
モニタリングによる通知と可視化



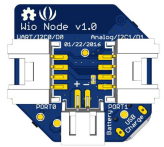
モニタリングによる通知と可視化



温湿度センサ

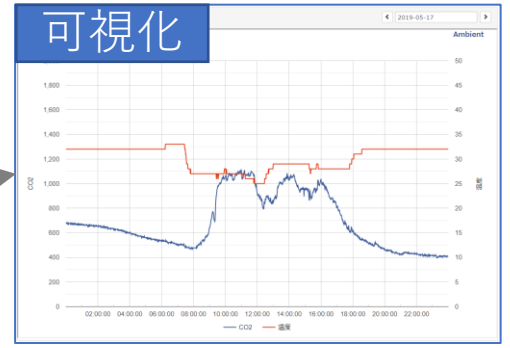


光センサ



センサ値の取得・判定

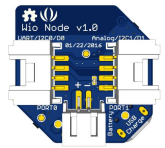
ターネット



Ambient

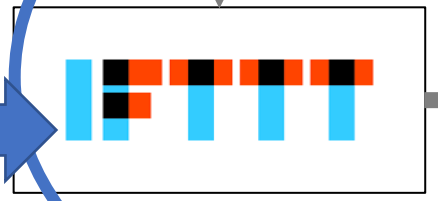
情報端末への通知

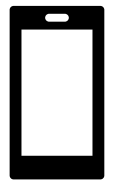
LED



ブザー

空間への通知





通知

実現すること

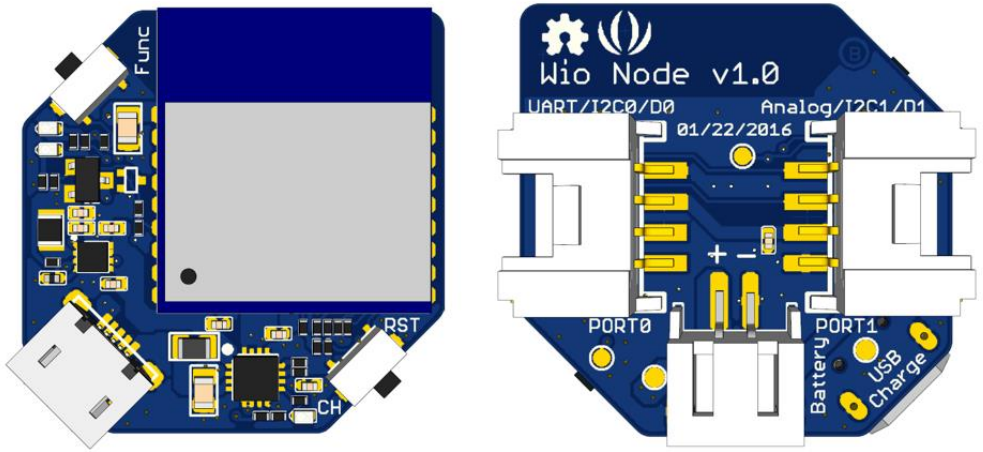
- **グラフによる可視化**
  - センサにより対象の空間の温度や湿度の変化を可視化する

- **情報端末への通知**
  - センサ値が一定の値を超えた時，メールにより端末を持つ人へアラートを通知する
- **空間への通知**
  - センサ値が一定の値を超えた時，LEDの点灯やブザーの鳴動により，対象の空間にいる人へアラートを通知する

WioNodeについて

Internet of **Things**

Wio Nodeとは？



- 本実習では容易に扱う事ができるIoTデバイス「Wio Node」を活用して演習を行う
- Wio Nodeは深センに本社を置くSeeed社が開発したIoTデバイス。同社が提供するGroveに対応したセンサ・アクチュエータと併用して動作させる。
- ファームウェアの開発が一切不要で、RESTful APIからセンサ・アクチュエータの制御が可能であり、非組込み系のエンジニアでも容易にHWデバイスの制御ができる事が特徴。

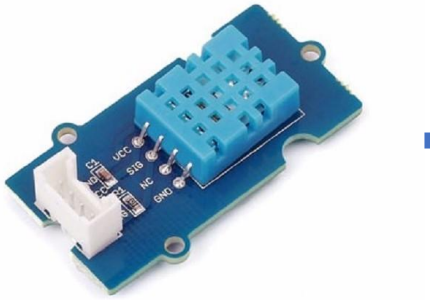
WioNodeの特徴

- とにかくお手軽！
  - 安価なデバイス
  - 豊富なHWモジュール
  - ファームウェア開発不要
    - タブレットからの簡単セットアップ
  - REST APIによる制御の為、言語依存が少ない

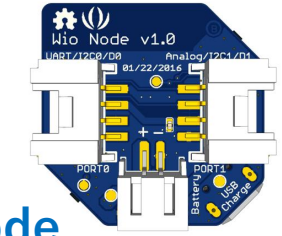
IoTへのファーストトライや
PoC・プロトタイピング用途には
最適なツールの1つ！！

- その反面のイマイチなところ
  - 基本的には数秒以上のサンプリング周期
  - 本番運用には複数の課題アリ
    - 安定稼働、デバイスの供給面、セキュリティなど

WioNodeのシステムイメージ



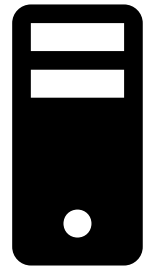
各種センサ



Wio Node

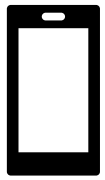
無線LAN
ルータ

インターネット



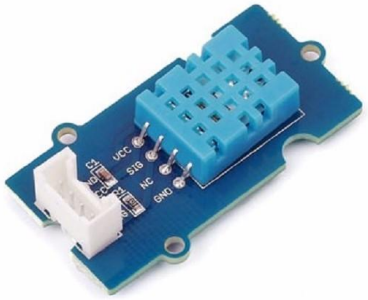
APIサーバ
(Seeed社)

各種アクチュエータ、
各種ディスプレイ

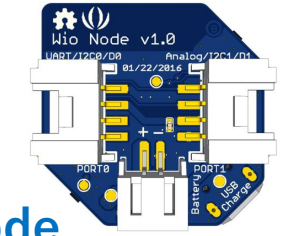


スマホ・タブレット
(設定用途)

WioNodeのシステムイメージ(今回)



各種センサ



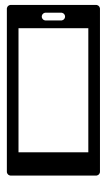
Wio Node

無線LAN
ルータ

インターネット

APIサーバ
(Seeed社)

各種アクチュエータ、
各種ディスプレイ



スマホ・タブレット
(設定用途)

クラウド
サービス
(SaaS)

Wio Nodeのハードウェア構成

No	名称	機能	No	名称	機能
1	Function	設定ボタン。Wio Nodeの動作設定に使用する。	4	PORT0	UART/I2C0/D0
2	Reset	デバイスのリセットを行うボタン	5	PORT1	Analog/I2C1/D1
3	Micro USB	電源の供給専用。通信機能無し。バッテリへの充電機能あり	6	Battery Holder	3.7V LiPoバッテリを接続可能

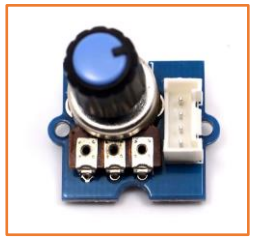
接続可能なセンサ＆アクチュエータ

ボタン

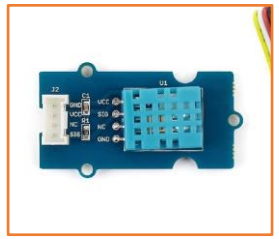
光センサ

温度センサ

マイク



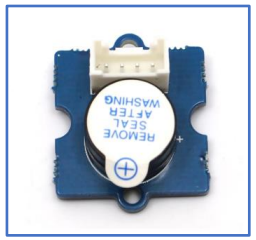
ボリューム



*温湿度センサ

*超音波距離センサ

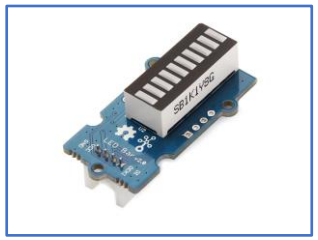
LED



ブザー

リレー

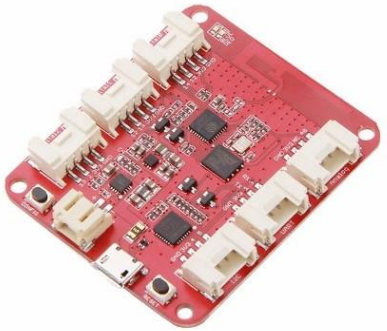
サーボモータ



*LEDバー

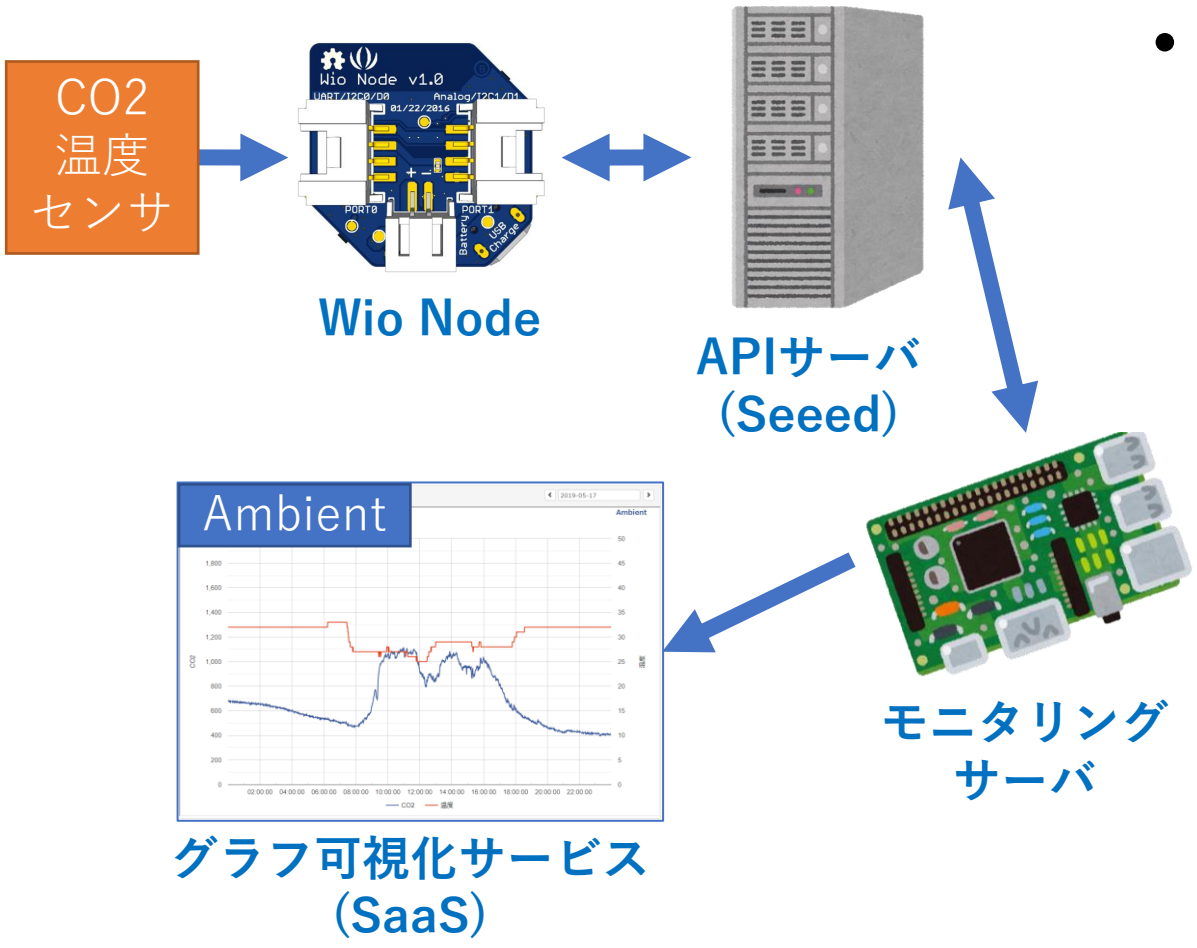
- ポリテクセンター熊本にあるセンサ＆アクチュエータ(2018年7月現在)
- *がついたモジュールは少量
- Seeed社のGroveシステムに適合したセンサ、アクチュエータ

Wioシリーズ一覧

	Wio Link	Wio Node	Wio LTE
			
価格	---円(日本では未発売)	1200円程度	12500円程度
Grove	6ポート	2ポート	6ポート
通信	WiFi	WiFi	LTE
Wioアプリ対応	○	○	×
Arduino化	○	○	○

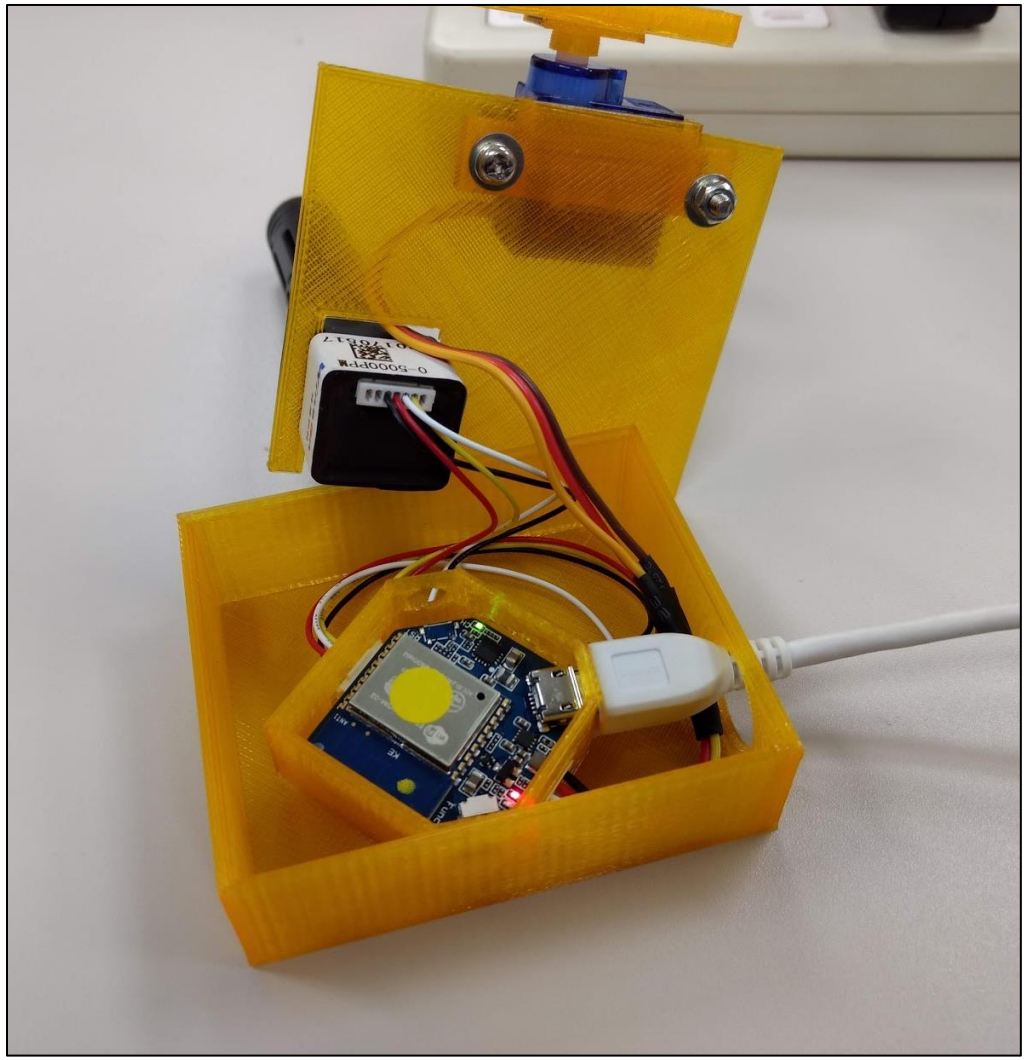
- 2018年7月現在、国内で販売されているWioシリーズはWio NodeとWio LTEの2種類
- Wio LinkはWioシリーズの最初のプロダクト
  - 2015年末にKickStarterで登場
  - Wio Nodeに比べてサイズが大きい分、Groveコネクタが6つあり
  - 技適等の問題により2018年7月現在、日本国内での販売無し

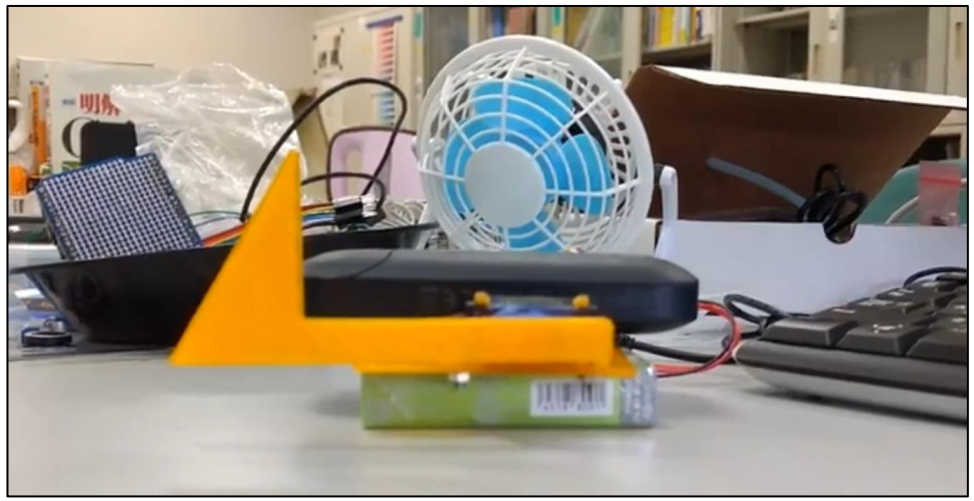
WioNodeを活用したシステム事例

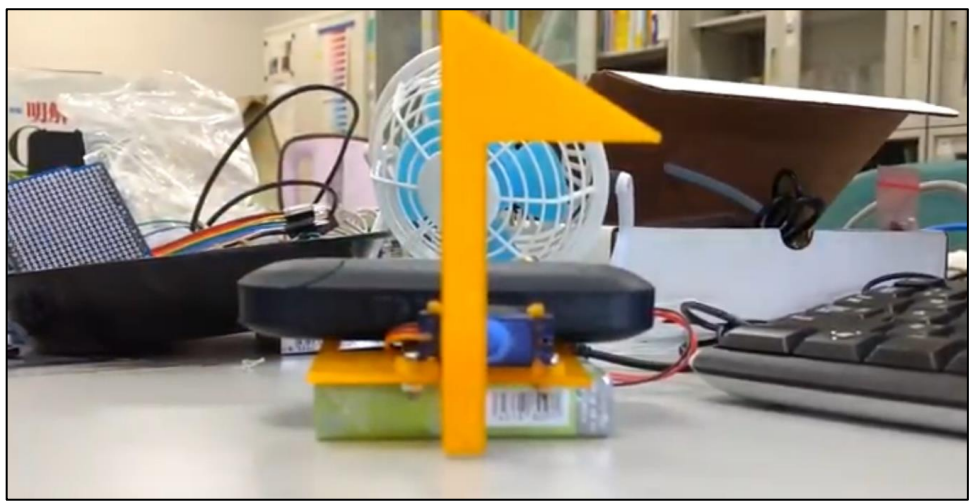


- とある教室の”CO2濃度”と”温度”をロギングするシステム

[成果]エアコンの稼働時期の前倒しに貢献
温度データを定期的にサンプリングしている為,
過去数日の室温の推移を把握可能に！
→ たまたま，その日だけが暑かったわけではない







WioNodeのセットアップ

Wio Nodeセットアップ①

基本的な手順は以下の通り

[参照] http://wiki.seeedstudio.com/Wio_Node/

1. スマートフォン or タブレットにWioアプリをインストール

   スマートフォンでセットアップする場合，SIMを用いたモバイルデータ通信(3G/4G/LTE回線)とWiFiを併用した状態では失敗する事が多いです.
   一時的に，モバイルデータ通信をオフにしWiFi接続のみで実施下さい

2. Wioアプリ用のアカウントを作成
   - メールアドレス　　：「　　」
   - パスワード　　：「　　」

Wio Nodeセットアップ②

3. Wio Nodeとアプリとの接続
   - **<u>複数台のWio Nodeを一斉に接続すると、自分のWio Nodeをアプリ上から見失う為、1人ずつ設定を行うこと</u>**

   - 設定時のWioNodeのSSID名を控える.
     - WioNode①：「　　」
     - WioNode②：「　　」
   - WioNodeの名前は「SemiXXYY」とすること。
     - XX：自分の使用しているPCに貼られたテプラを参照
       - 例：テプラの番号が”3”の場合は，”03”とする
     - YY：WioNode①か②かを指定
       - 例：①の場合は，”01”とする

Wio Nodeセットアップ③

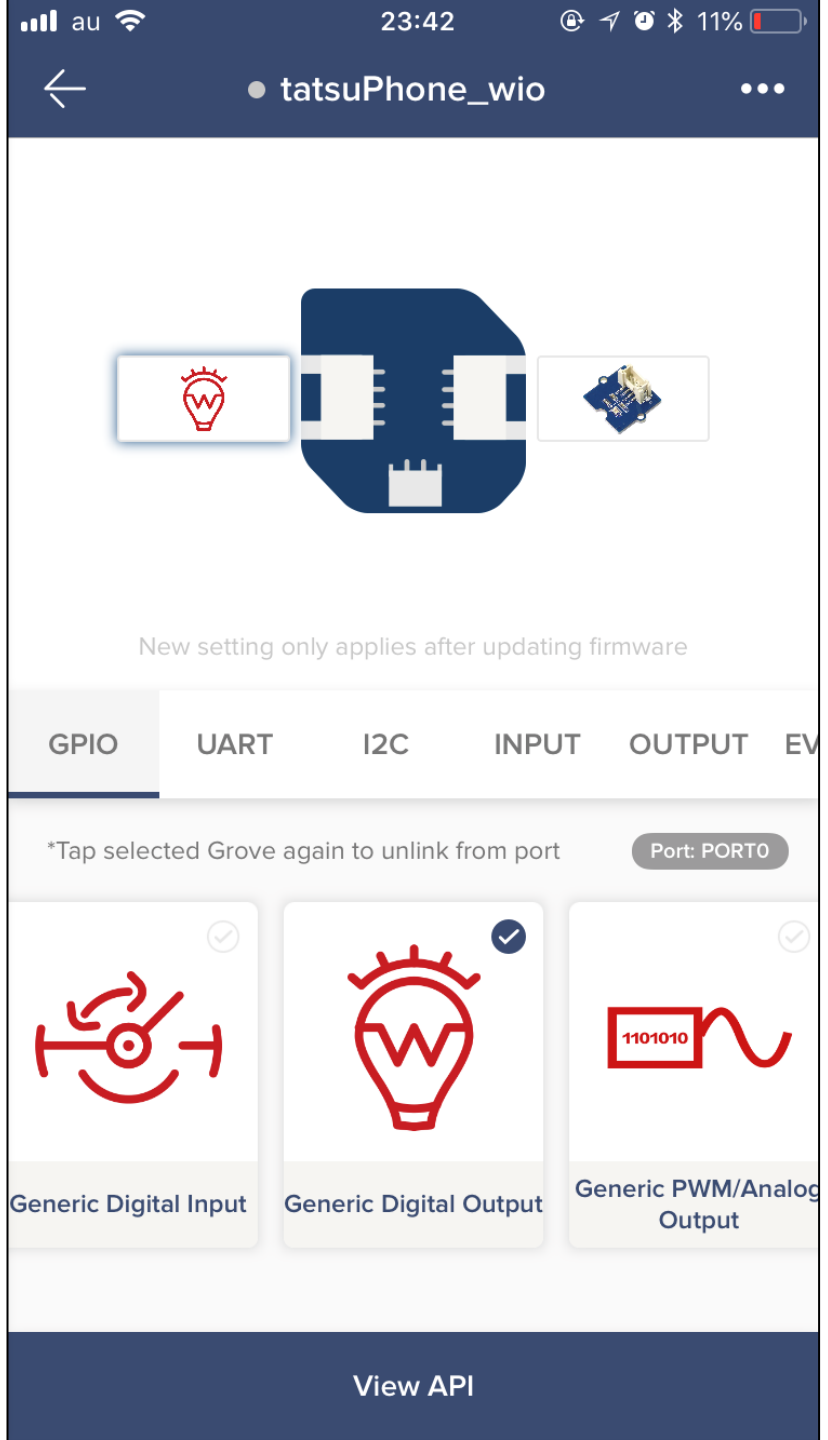
4. 接続するHWを選択し、ファームウェアをアップデートする

[D0(左)]

OUTPUT＞LED (Generic Digital Output)

[D1(右)]

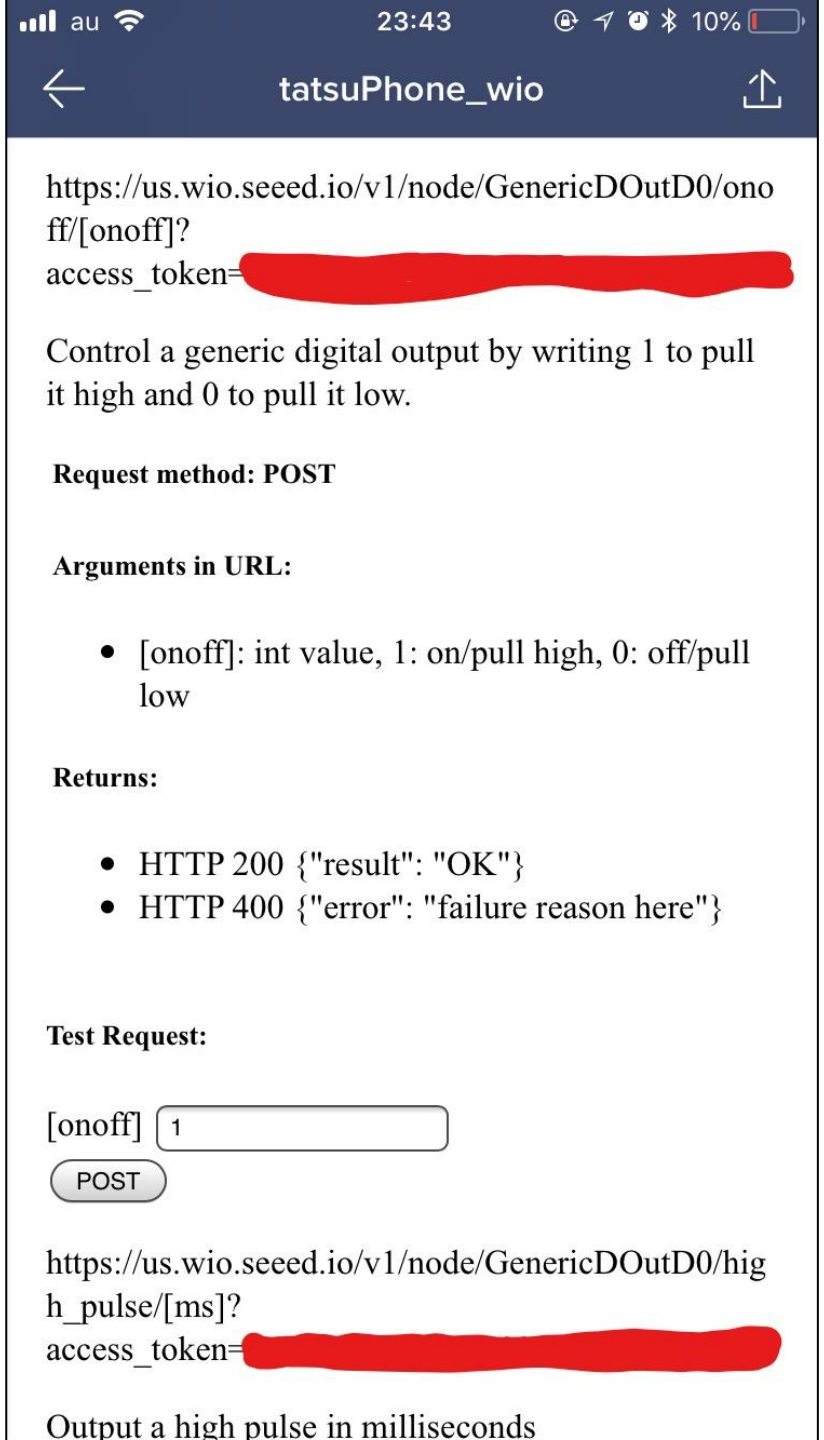
INPUT＞Luminace Sensor



Wio Nodeセットアップ④

5. テストアプリ(ViewAPI)から動作確認
   - ViewAPIボタンをタップすると、テスト用のWebアプリが開かれます
   - 接続中のセンサからのデータ取得、アクチュエータの制御が簡易に行える機能を持ちます

   - 以下が確認できればセットアップ完了
     - LEDの点灯／消灯制御
     - 光センサの値の取得



ViewAPIをPCのブラウザで見る方法

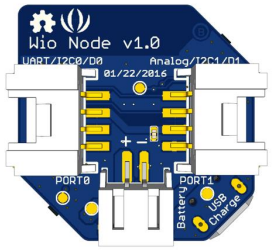
- ViewAPIはWebページである為、PCのブラウザからURLでアクセスすれば閲覧が可能
  - IEやEdgeからは動作しない事がある為，Chrome/Firefoxを推奨

- URLのタブレット→PCへの転送方法
  - URLが非常に長い為、メールによる転送もしくはURL短縮を行うこと
  - WebブラウザからURL短縮サービス「bitly」を開き、ViewAPIのURLを短縮。その後、短縮されたURLをPCのブラウザに打ち込む
    - https://bitly.com

WioNodeとViewAPIの紐付き

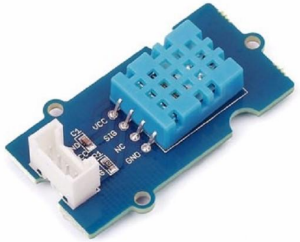
ViewAPI **WioNode** **Groveモジュール** **REST API**

ViewAPI



WioNode

光センサ



温湿度センサ

光量の取得

湿度の取得

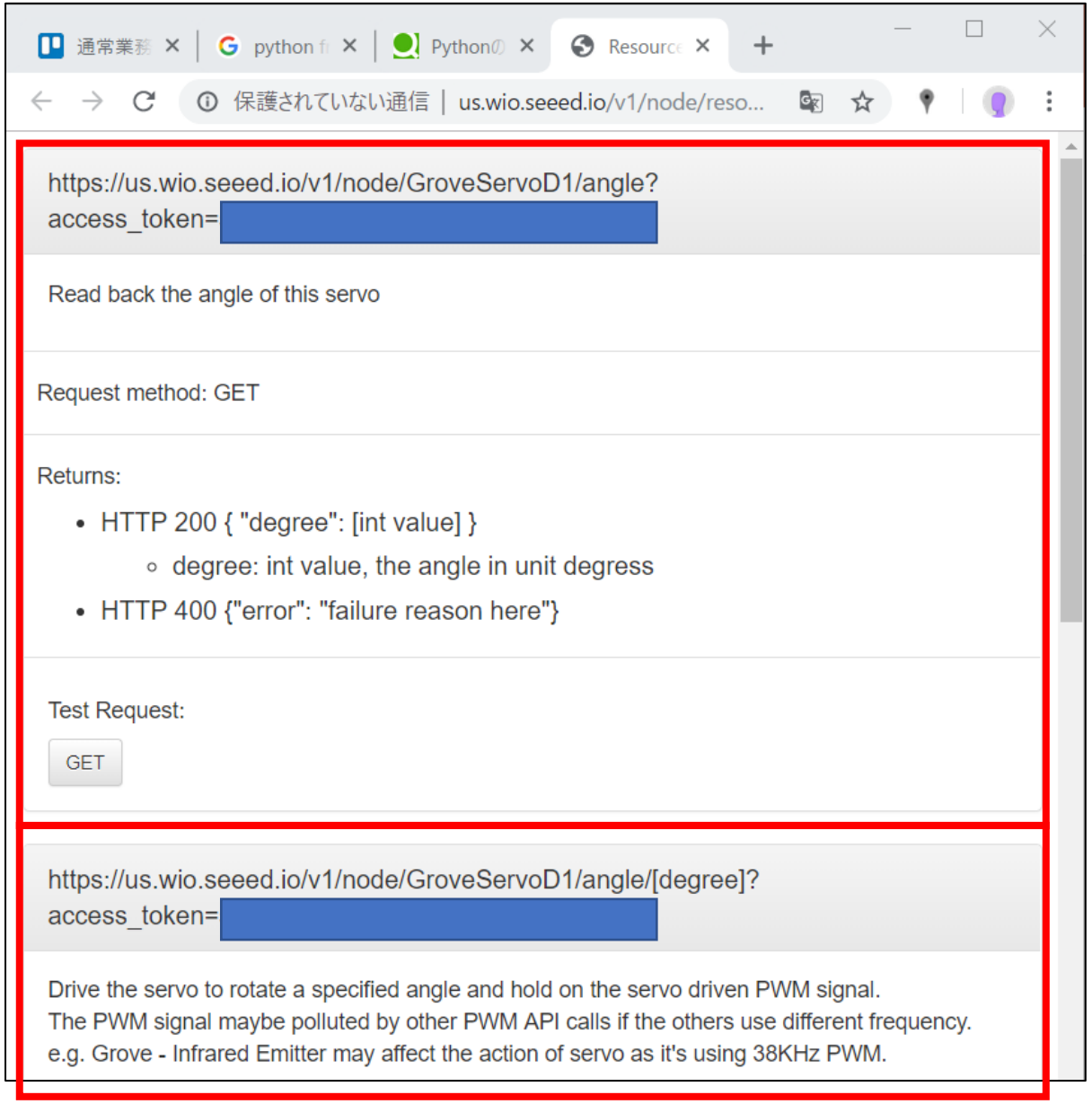
温度(セ氏)の取得

湿度(カ氏)の取得

Sleep機能

- WioNodeとViewAPIは1対1の関係
- WioNodeとGroveモジュールは1:2の関係
- Groveモジュールと制御用のREST APIは1対1*の関係

ViewAPIの読み方①



ViewAPIはWebページであり，URLがあります
→　PCからの閲覧も可能です(Chrome, Firefox推奨)

ViewAPIのページは，WioNodeのアクセストークンと1対1に紐づいています
→ 接続するGroveモジュールを変更してもURLは変わらない

ViewAPIのページ内に，接続したGroveモジュールを制御する
為のREST APIのドキュメントおよびテスト実行環境が揃っています.
→ ここで記載のあること以上の制御は出来ません

プログラムの開発手順は以下の通りです
① Groveモジュールを繋ぐ
② Wioアプリから接続したGroveモジュールを設定
③ WioアプリからFirmwareUpdate
④ ViewAPIよりREST APIの仕様確認．動作確認テスト
⑤ 任意の言語でプログラミング

ViewAPIの読み方②

https://us.wio.seeed.io/v1/node/GroveLuminanceA0/luminance?
access_token=[redacted]

← REST API

Read the intensity of the ambient light.

← REST APIの解説

Request method: GET

← 方式：GET/POST．値を取得する場合はGET.

Returns:

- HTTP 200 { "lux": [float value] }
    - lux: float value, light intensity
- HTTP 400 {"error": "failure reason here"}

← 戻り値の事例.

Test Request:

GET

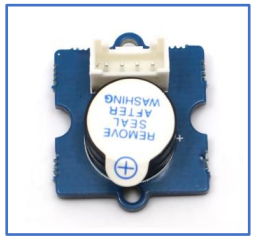
← REST APIをテスト実行する

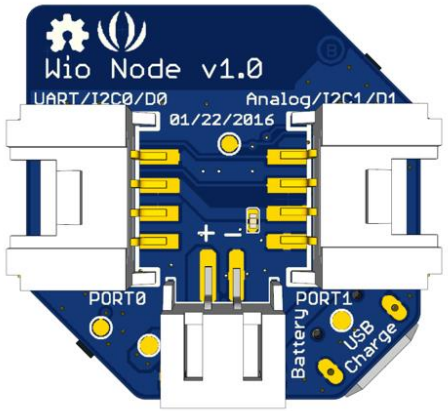
Curl example:
curl -k https://us.wio.seeed.io/v1/node/GroveLuminanceA0/luminance?
access_token=[redacted]

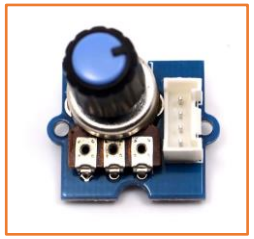
Response:
{"lux": 567.21}

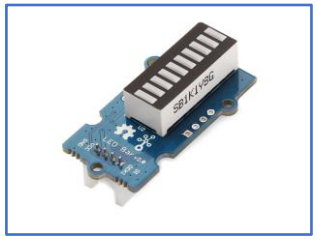
← Curlコマンドを使った場合の事例とその応答

練習①：いろいろなHWを繋いでみる









- 手元にあるGroveモジュールを接続して、どのような制御ができるのかViewAPIから確認してみましょう！
- 手順
  - 手元にあるHW(センサ・アクチュエータ)をWio Nodeに接続し、タブレットからファームウェアアップデートを行って下さい。
  - アップデート後、View APIより、接続したHWの制御を行い動作を確認してください
  - 最後は、元の「LED」と「光センサ」の構成に戻すこと

練習①：いろいろなHWを繋いでみる(続き)

- PORT0, PORT1でそれぞれ接続できるHWが異なります
  - 接続するGroveモジュールとどのようなインターフェースで接続するかよく確認しましょう！
  - Wio Node側の仕様
    - PORT0　　: Digital IO, UART, I2C
    - PORT1　　: Digital IO, Analog Input, I2C
  - Groveモジュール側の仕様
    → Seeed社のWikiを参照しましょう！
    - センサ系　　：　http://wiki.seeedstudio.com/Sensor/
    - アクチュエータ系　　：　http://wiki.seeedstudio.com/Actuator/
    - ディスプレイ系　　：　http://wiki.seeedstudio.com/Display/

- 【動作がうまくいかなくなった時は？】
  - WioNodeの電源を再投入してください
  - Wioアプリを一旦終了後、再起動してください
  - それでもダメな場合は、タブレットの電源を再投入してください

触ったモジュールについてはメモしておきましょう！

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

触ったモジュールについてはメモしておきましょう！

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

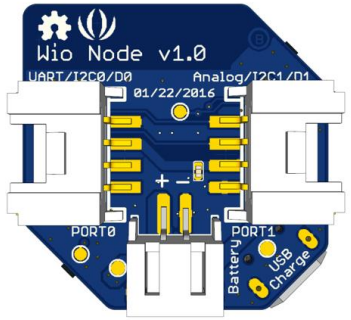
- 注意点　　 ：
- 備考　　 ：

- モジュール名 ：
- できること　 ：

- 注意点　　 ：
- 備考　　 ：

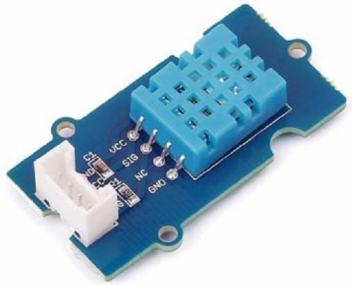
練習②：最後は以下の構成に

LED

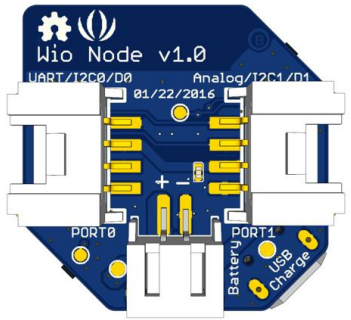


WioNode①

ブザー



温湿度センサ



WioNode②

光センサ

- 以後の実習を行う為に左図の構成にしてください
- WioNode①の構成
  - PORT0 ：LED
  - PORT1 ：ブザー
- WioNode②の構成
  - PORT0 ：温湿度センサ
  - PORT1 ：光センサ

[参考] 周期的なアクセスの安定度

- 10秒前後の周期でアクセスだと，ちらほら通信に失敗する
- では，どの程度の通信の安定度なのか？

- [参考]とある1日のデータ
  - 動作条件
    - 60sec周期に2度，特定のWioNodeのセンサ値に対するアクセス
    - 60sec周期に1度，ambientに対してデータをpush
    - 同一ネットワーク内のWioNodeの個数は1つ
  - 24時間で考えると1440件のデータをpush
    - 実績値：1424件
    - **成功率：98.89%**

Python入門

Pythonについて

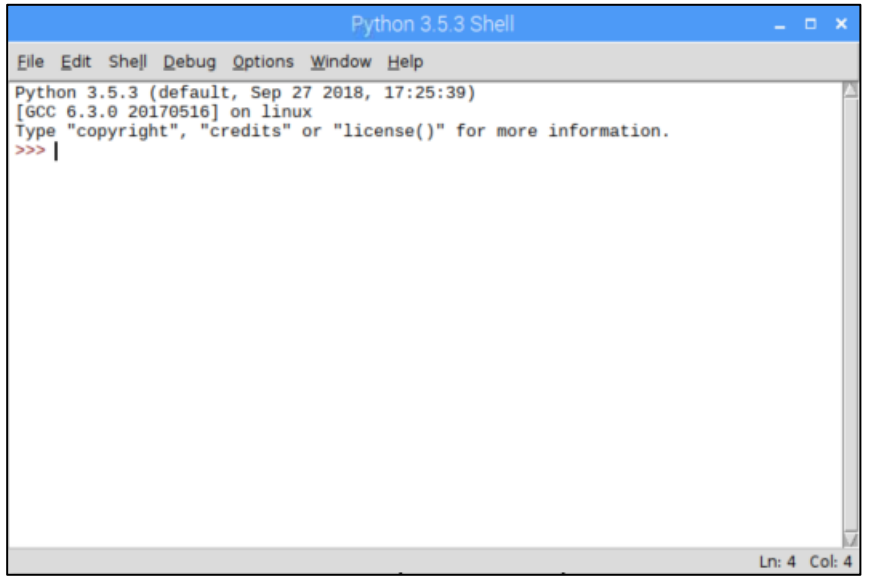
- 1991年グイド・ヴァンロッサムによって開発
- 特徴
  - コードがシンプルで扱いやすい
  - 少ないコード行数で書ける
  - コードが読みやすい
- インデントが重要…オフサイドルール
  - ブロックを字下げで指定する
  - 結果、誰が書いても似た記述に。→読みやすい
- 主な活用先
  - 機械学習、クロール＆スクレイピング
  - IoT
    - RaspberryPiが代表格
    - 一部のマイコンではPythonをサポートしていることも

Pythonに入門したい人は…

- 本実習はPythonコードを基本的に提供→解説の流れ
- Pythonを言語として学習したい方は…
  - Python Boot Campが初学者向けの入り口
    - https://bootcamp-text.readthedocs.io/textbook/index.html

Pythonの開発環境

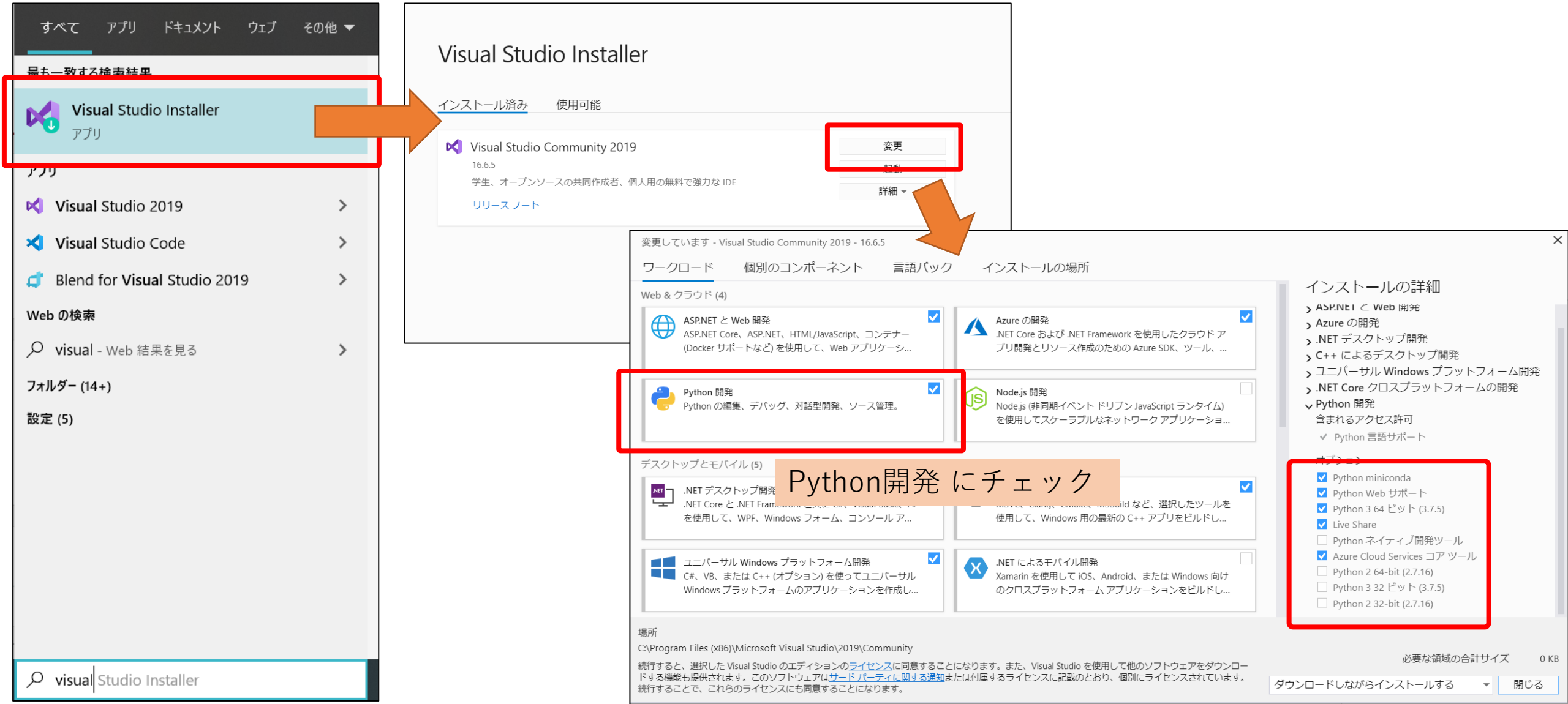
- IDLE
  - Python標準のエディタ
  - 軽量であるが、簡易な機能まで
- PyCharm
  - 有償版と無償版あり
    - 無償版は一部機能が制限されている模様
  - サポートOSが豊富…Win, Mac, Linux
  - 補完機能アリ
- VisualStudio
  - VSが公式でサポート
    - ブレイクポイントを活用したデバッグ可能
    - インテリセンスも使用可能
  - Windowsを使用する開発者には馴染みのあるUI
    - プロジェクトのテンプレートもあり



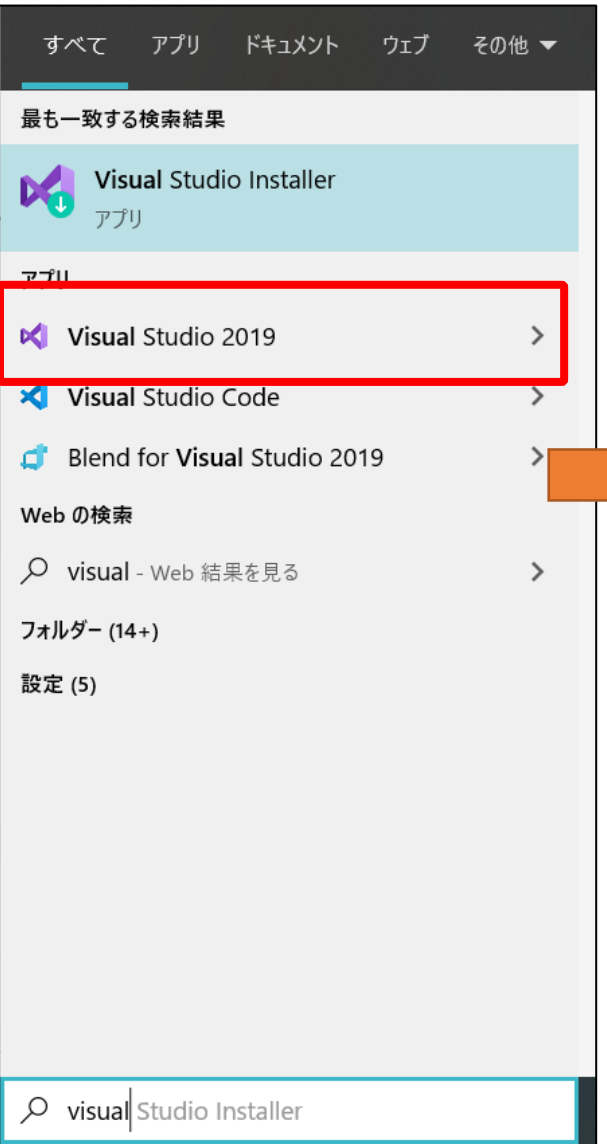
ドキュメント情報：https://docs.python.jp/3/library/index.html

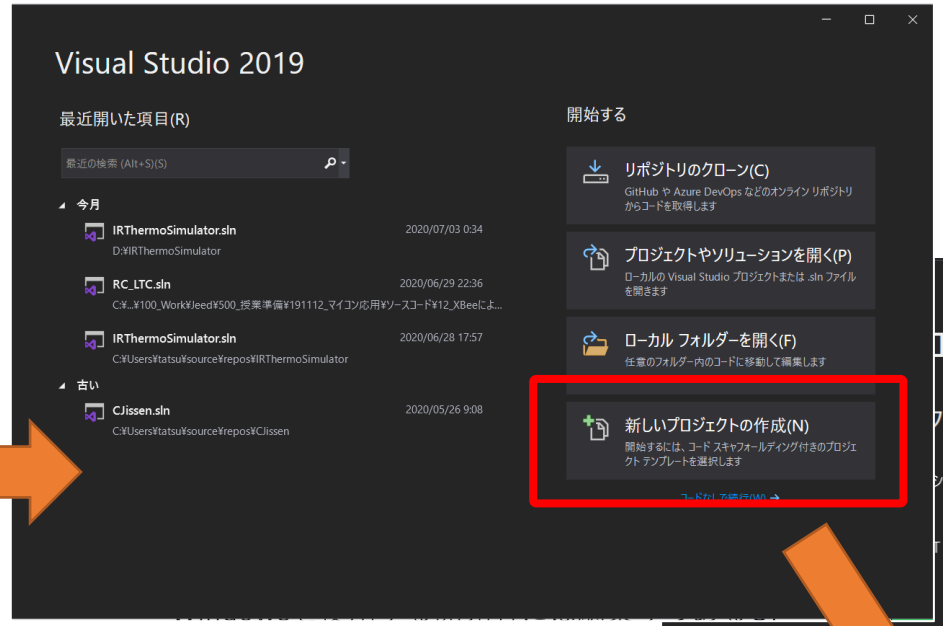
① はじめての Pythonプログラミング

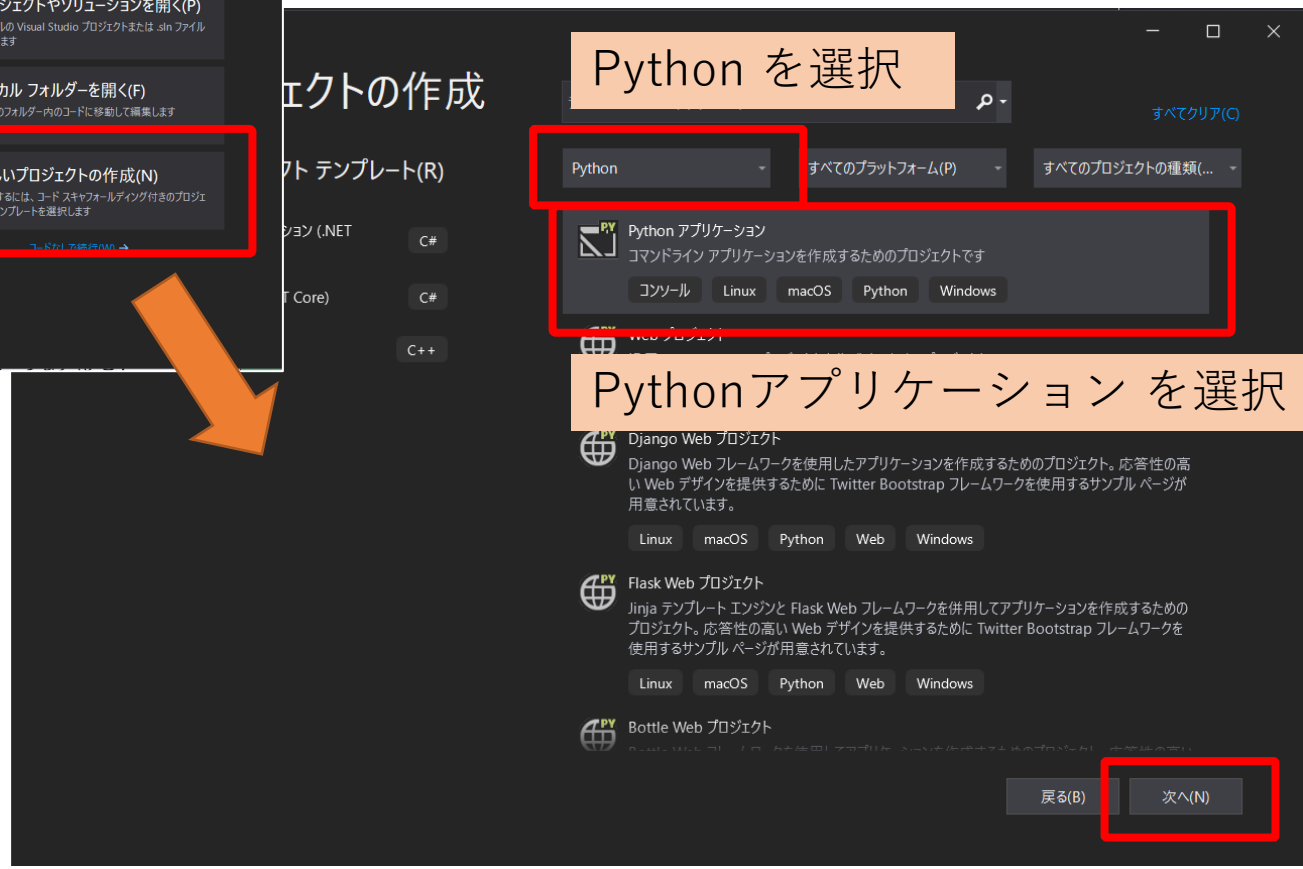
1．VisualStudioでPython開発環境の確認



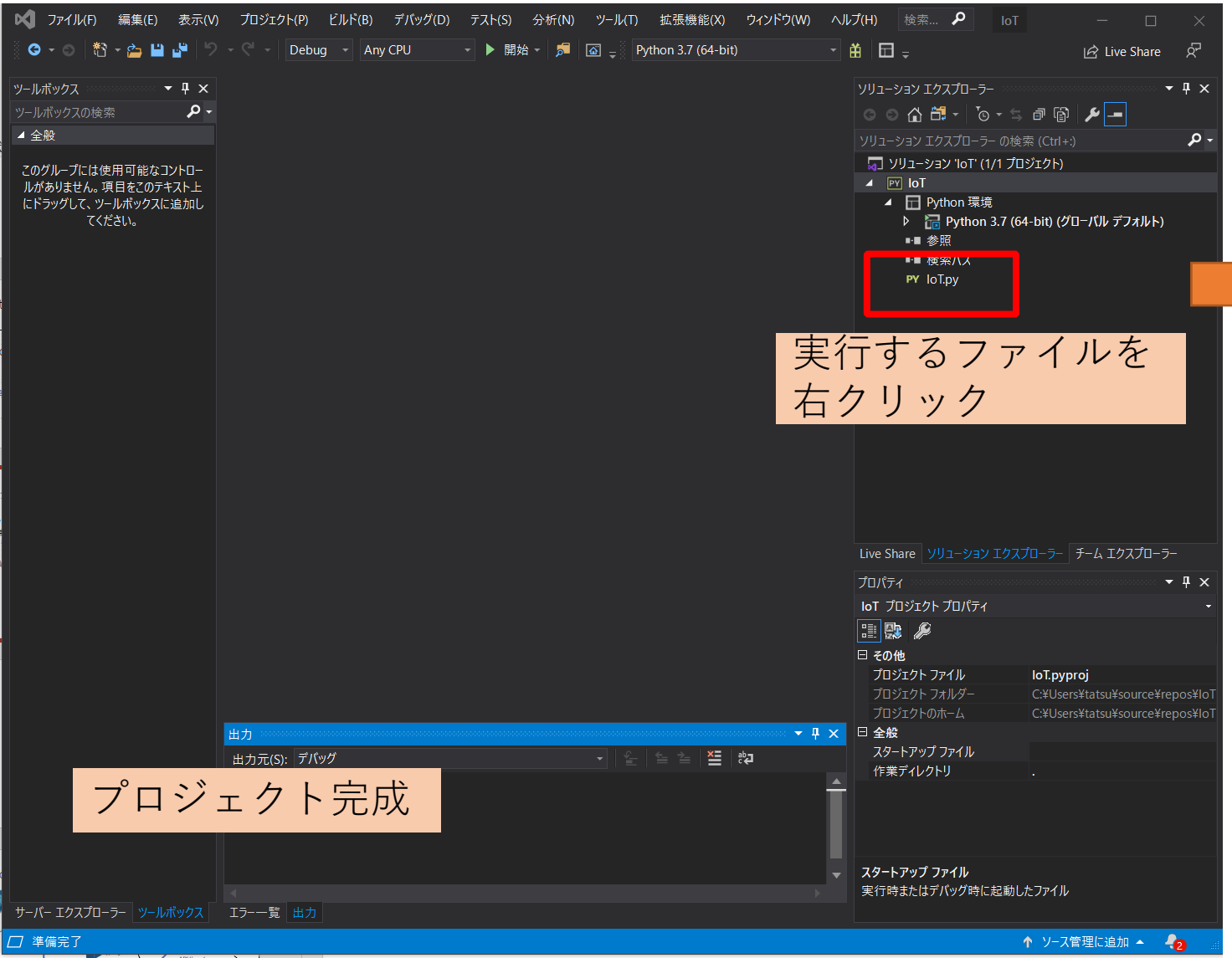
2．Pythonプロジェクトの作成

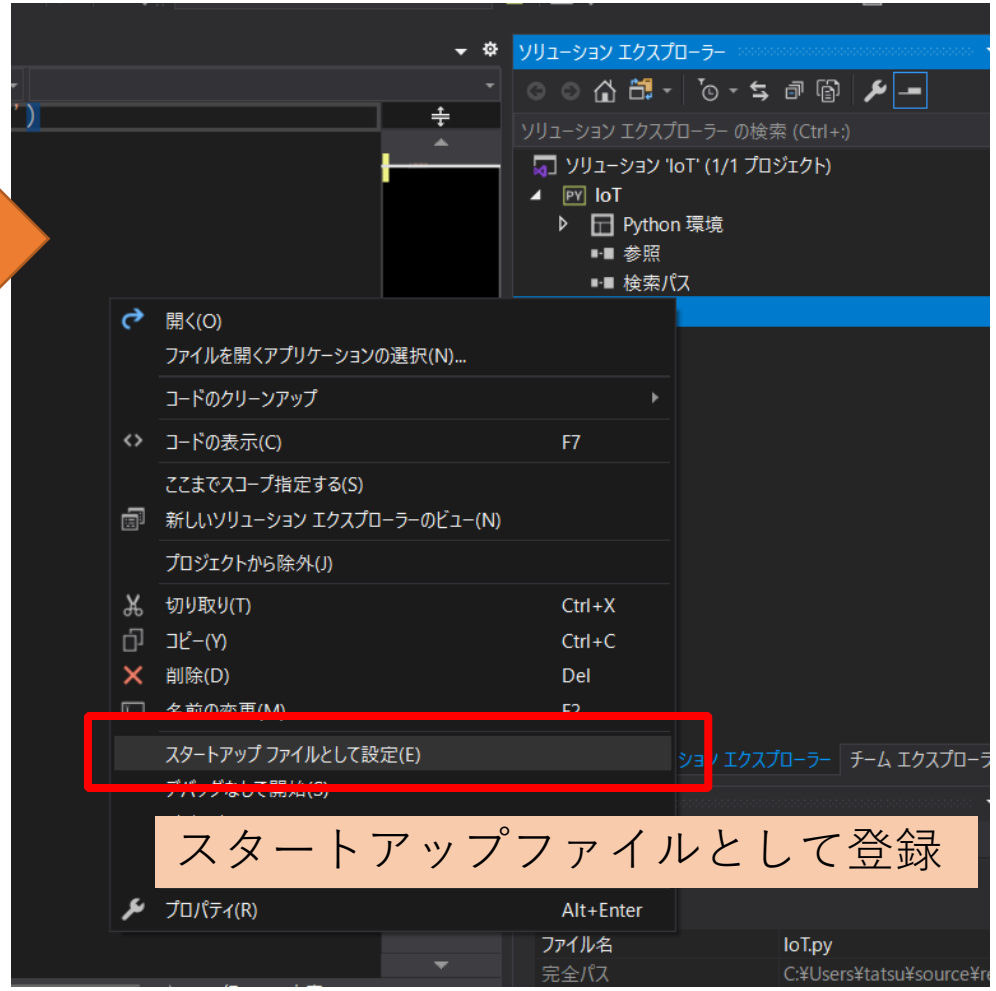




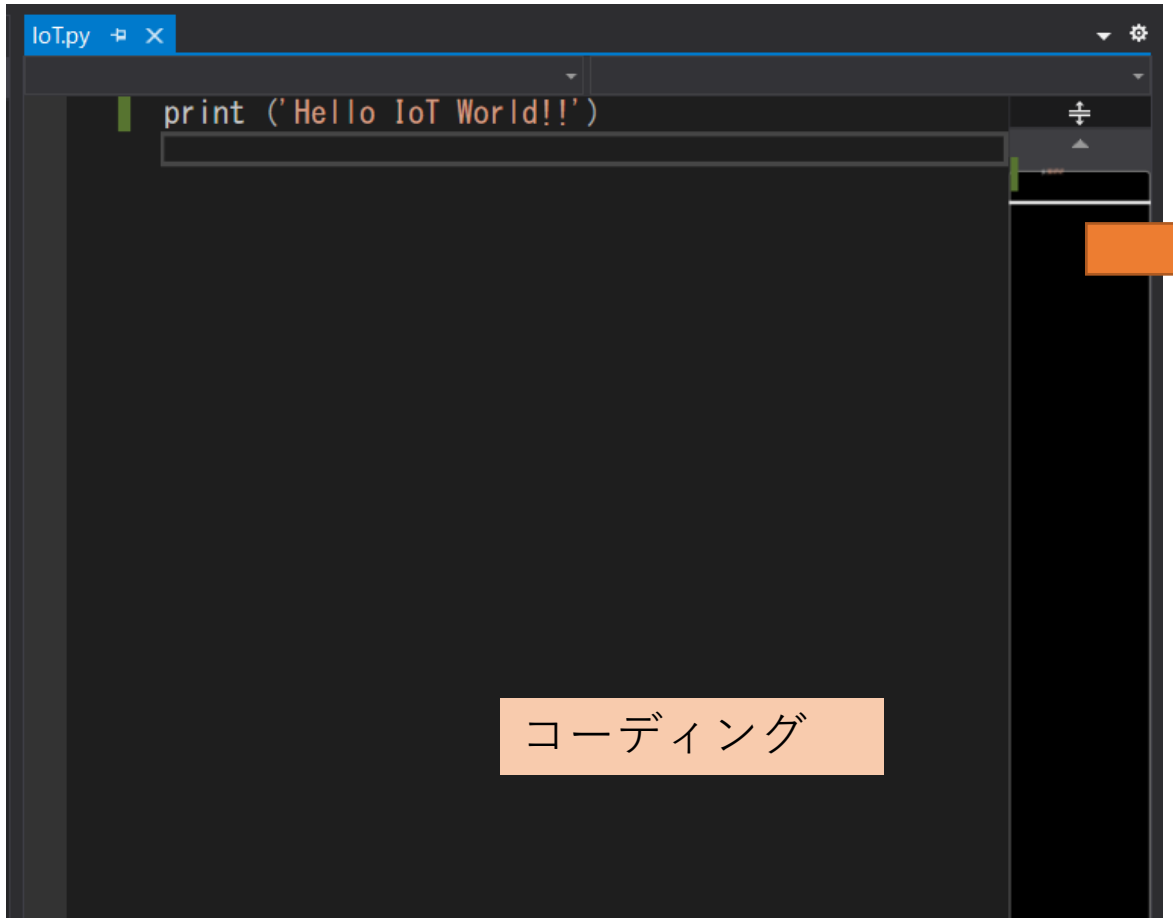


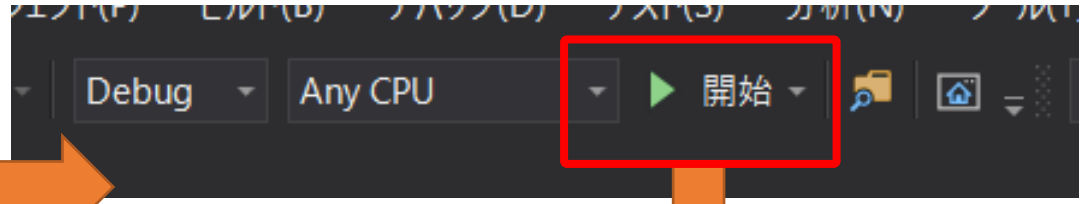
3．Pythonプロジェクトの作成(続き)





４．実装＆動作確認


IoT.py
print ('Hello IoT World!!')
コーディング


Debug
Any CPU
開始


C:¥Program Files (x86)¥Microsoft Visual
Shared¥Python37_64¥python.exe
Hello IoT World!!
Press any key to continue . . .
実行結果が表示

② 値の扱い

print()関数とは

- ()内に含めた値を画面上に表示する機能(関数)
- 値は「数値」でも「文字列」でも可

```
print('Hello World')
print(123)
```

- 値の種類
  - 文字列　…文字，文字列．“” or ‘’ で囲った文字を指す
  - 整数　　…小数点を含まない数値
  - 実数　　…小数点を含む数値(浮動小数点とも呼ぶ)
  - ブール　…True / Falseの二値
  - 配列　　…(今回は含めない)
  - タプル　…(今回は含めない)
  - 辞書　　…(今回は含めない)

算術演算

```
print(10 + 5)  # 足し算
print(10 - 5)  # 引き算
print(10 * 5)  # 掛け算
print(10 / 5)  # 割り算

print(10 % 6)  # 余り算

print(10 + 0.5)# 整数と実数の計算
```

- ファイル名
  - 02_01_calc.py

- 四則計算の他に余り算などが存在する
  - %　　… 余り算
  - //　　… 割り算(小数点以下切捨)
  - **　　… 左辺の右辺乗
    - 例： 6**2 だと 6の2乗 = 36

演算子の優先順位

```
print(10 + 4 - 2)　# 普通の計算
print(10 + 4 / 2)　# 足し算より割り算の方が優先順位が高い

print((10 + 4) / 2）# 括弧で優先順位を制御しよう
```

- ファイル名
  - 02_02_calc_long.py

- 通常の四則計算と同様に優先順位の概念がある為，注意が必要
  - 気になるなら，括弧を使って優先順位を上げる方法も．

https://qiita.com/tTAGUCHI/items/1dee92cd058f975ffce3

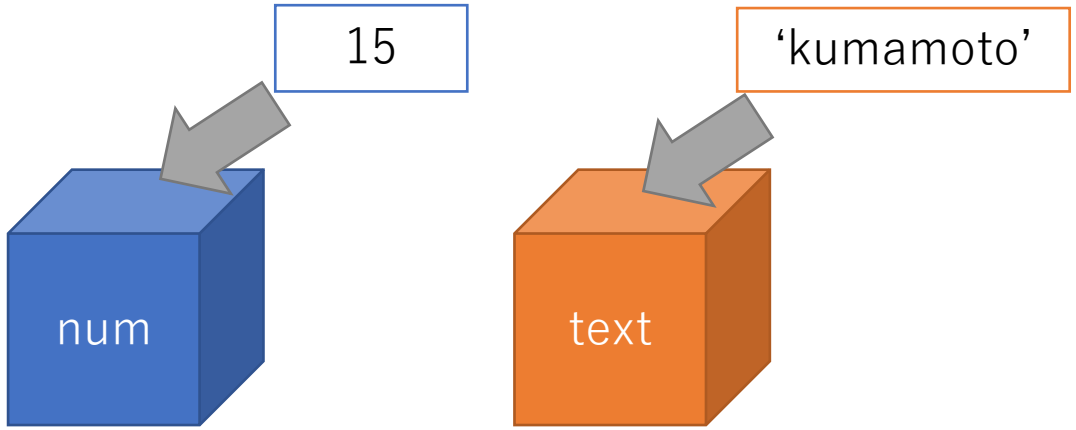
③ 変数の扱い

変数

- ファイル名
  - 02_03_val.py

```
# 数値を扱う変数
num = 10 + 5
print(num)

# 文字列を扱う変数
text = 'kumamoto'
print(text)
```

- 変数は値を一時的に保存する箱
  - 値が必要な場面での代わりに変数を使用可



num = 10

変数numに整数10を格納している処理の意
→ 数学における =(等価) ではなく，代入を意味する為，初学者には注意が必要.

変数の命名規則

- 半角アルファベットの大文字・小文字，アンダースコア，数字により構成が可能
  - A～Z, a～z, _, 0～9は使用可能
- 変数名が数字のみ・先頭に数字は不可
- 予約語と同一の名前は不可
  - True, False, if, for, while, class, and…などなど

数値と文字列の組み合わせ

```
# 電圧値を変数に入れる
voltage = 1.23
print(voltage)

# センサ名を変数に入れる
sensor = 'volume:'
print(sensor)

# 組み合わせて表示
#  ポイント:文字の結合 と 数値->文字列変換
print(sensor + str(voltage) + 'v')
```

- ファイル名
  - 02_04_print.py

- 数値→文字列変換　　：str(数値)
- 文字列→数値変換　　：int(文字列)
  float(文字列)

- 文字列同士の結合
  文字列 + 文字列　で可能に

④ 分岐と繰り返し

分岐と繰り返し

03_IfLoop.py

```
# 正方形の描画
print('input square : ', end='')  # end='' で改行なし
sq_str = input()

# 0からint(sq_str)まで繰り返し
for i in range(int(sq_str)):
    for j in range(int(sq_str)):
        print('*', end='')
    print('')

# 三角形の描画
print('input triangle : ', end='')
tr_str = input()
tr_num = int(tr_str)

for i in range(tr_num):
    for j in range(tr_num):
        if i < j:                  # if文
            print('*', end='')
    print('')
```

```
# おみくじ
print('input number(0～2):', end='')
o_str = input()
o_num = int(o_str)

if o_num == 0:
    print('大吉')
elif o_num == 1:
    print('中吉')
elif o_num == 2:
    print('小吉')
else:
    print('凶')
```

```
if 式や関数など:
    実行する処理
```

```
if 式や関数など:
    実行する処理A
else:
    実行する処理B
```

```
if 式や関数など:
    実行する処理A
elif 式や関数など:
    実行する処理B
else:
    実行する処理C
```

比較演算子

演算子	例示	説明
<	A < B	AはB未満
<=	A <= B	AはB以下
>	A > B	AはBより大きい
>=	A >= B	AはB以上
==	A == B	AはBと等しい
!=	A != B	AはBと等しくない

ブール演算子

演算子	例示	説明
and	比較式1 and 比較式2	比較式1,2の**どちらも** Trueの時にTrueとなる
or	比較式1 or 比較式2	比較式1,2の**どちらか**が Trueの時にTrueとなる
not	not(比較式)	比較式がTrueの時にFalse 比較式がFalseの時にTrue

```
for 変数 in range(終了値):
    繰り返す処理
```

```
for 変数 in range(開始値, 終了値):
    繰り返す処理
```

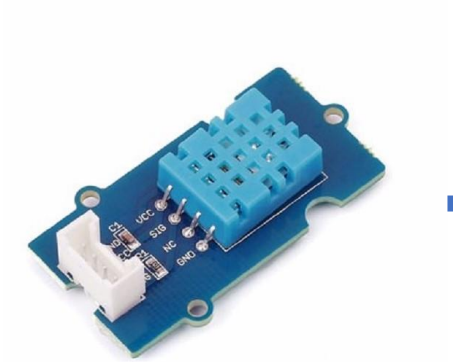
```
while 継続条件 :
    繰り返す処理
```

Memo

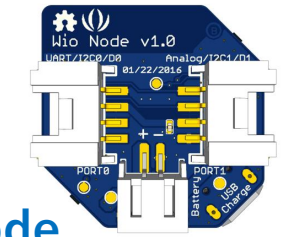
PythonでWioNode制御

センサ編

WioNodeの制御イメージ



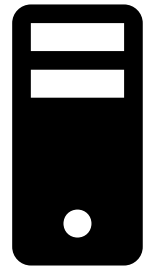
各種センサ



Wio Node

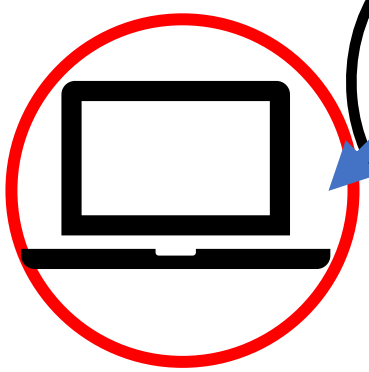
無線LAN
ルータ

インターネット



APIサーバ
(Seeed社)

各種アクチュエータ、
各種ディスプレイ



PCでプログラミング

REST APIとは

- RESTful API / REST APIと呼ばれるWebAPIの一種
- **Webシステムを外部から利用する為のAPI**
- HTTPメソッドによってWebシステムへアクセスする
  - GET　…　データの取得
  - POST　…　データの作成・追加
  - PUT　…　データの更新
  - DELETE　…　データの削除
- APIを実行した結果はXML, HTML, JSONなどで返される

- Twitterなど様々なWebサービスで提供されている
  - 海外・国内の便利なAPI一覧： http://smsurf.app-rox.com/api/

参考： http://e-words.jp/w/RESTful_API.html

Wio Nodeの制御のポイント(output系)

LEDのON/OFF制御の場合

①　②

https://us.wio.seeed.io/v1/node/GenericDOutD0/onoff/[onoff]?access_token=XXXX

https://us.wio.seeed.io/v1/node/制御対象/制御方法/値?access_token=値2

- パラメータを送って制御する場合
  - **POSTメソッド**での制御
  - [onoff]の箇所に0 or 1を入れてLEDの点灯/消灯制御
    - 0 : LED消灯
    - 1 : LED点灯

←①

  - どのWio Nodeを制御するかの識別子。セットアップしたWio Nodeごとに異なる為、取り扱いには注意。

←②

  - 戻り値は以下の2パターン。制御の成否が返る
    - HTTP 200 {"result": "OK"}
    - HTTP 400 {"error": "failure reason here"}

Wio Nodeの制御のポイント(input系)

光センサの光量取得の場合

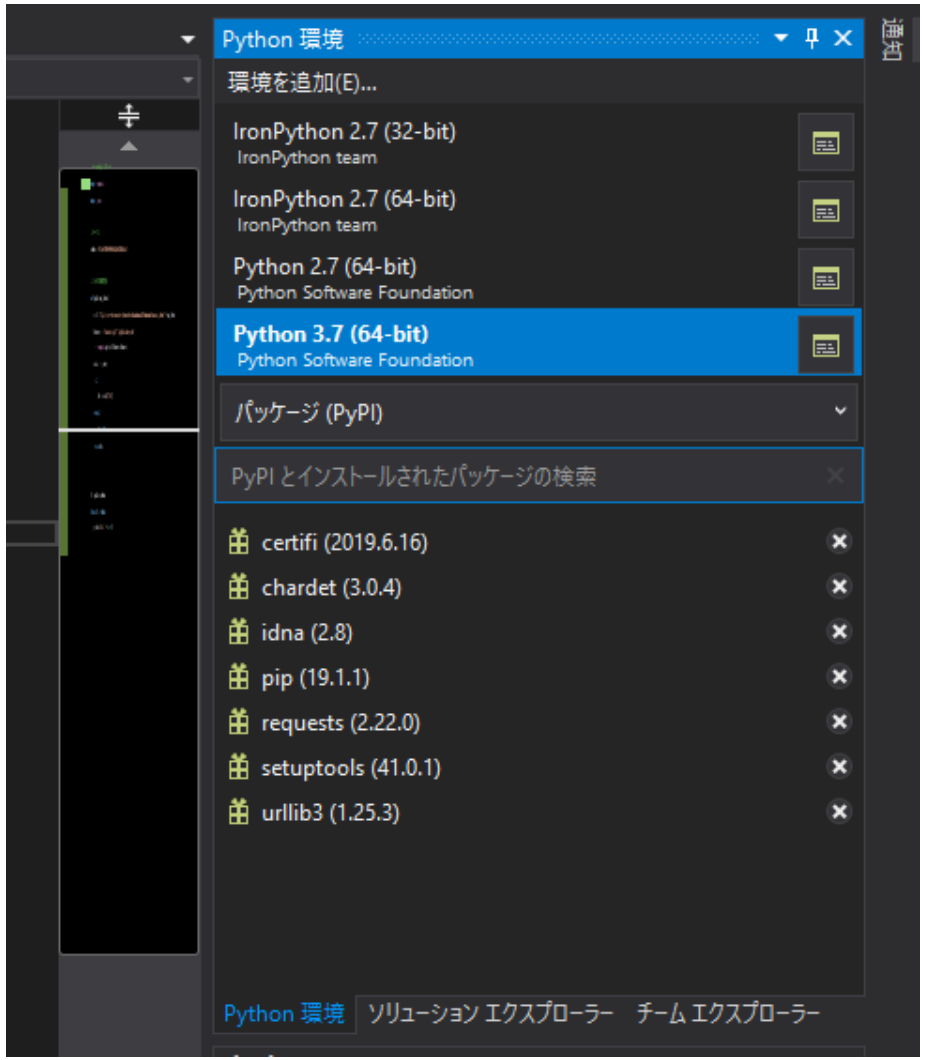
①

https://us.wio.seeed.io/v1/node/GroveLuminanceA0/luminance?access_token=XXXX

https://us.wio.seeed.io/v1/node/制御対象/取得データ?access_token=値

- センサ値などを取得する場合
  - **GETメソッド**での制御
  - どのWio Nodeを制御するかの識別子。セットアップしたWio Nodeごとに異なる為、取り扱いには注意。 ←①
  - 戻り値は以下の2パターン。制御の成否が返る
    - HTTP 200 { "lux": **[float value]** }
      - lux: float value, 光の強さを返す
    - HTTP 400 {"error": "failure reason here"}

VisualStduioにrequestsを追加

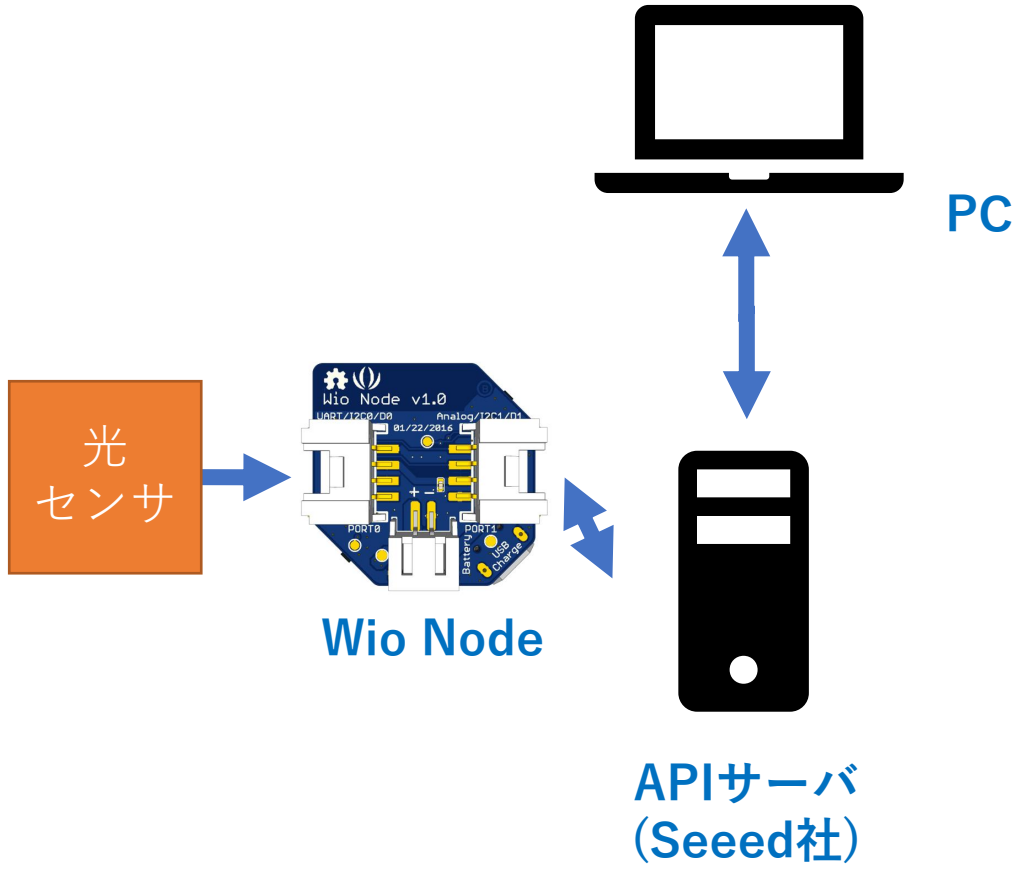


- Pythonの環境をクリック
- PyPIとインストールされたパッケージの検索に**”requests”**と入力
- 次のコマンドを実行する:pip install requestsをクリック

練習①：光量の取得

- 例題
  - PythonからWio Nodeから光センサの値を取得しよう！
- 練習
  - 繰り返し光量を取得するプログラムに改変しよう！
    - ヒント①
      - 無限ループは while 1:
      - 繰り返す際には待ち処理を入れよう

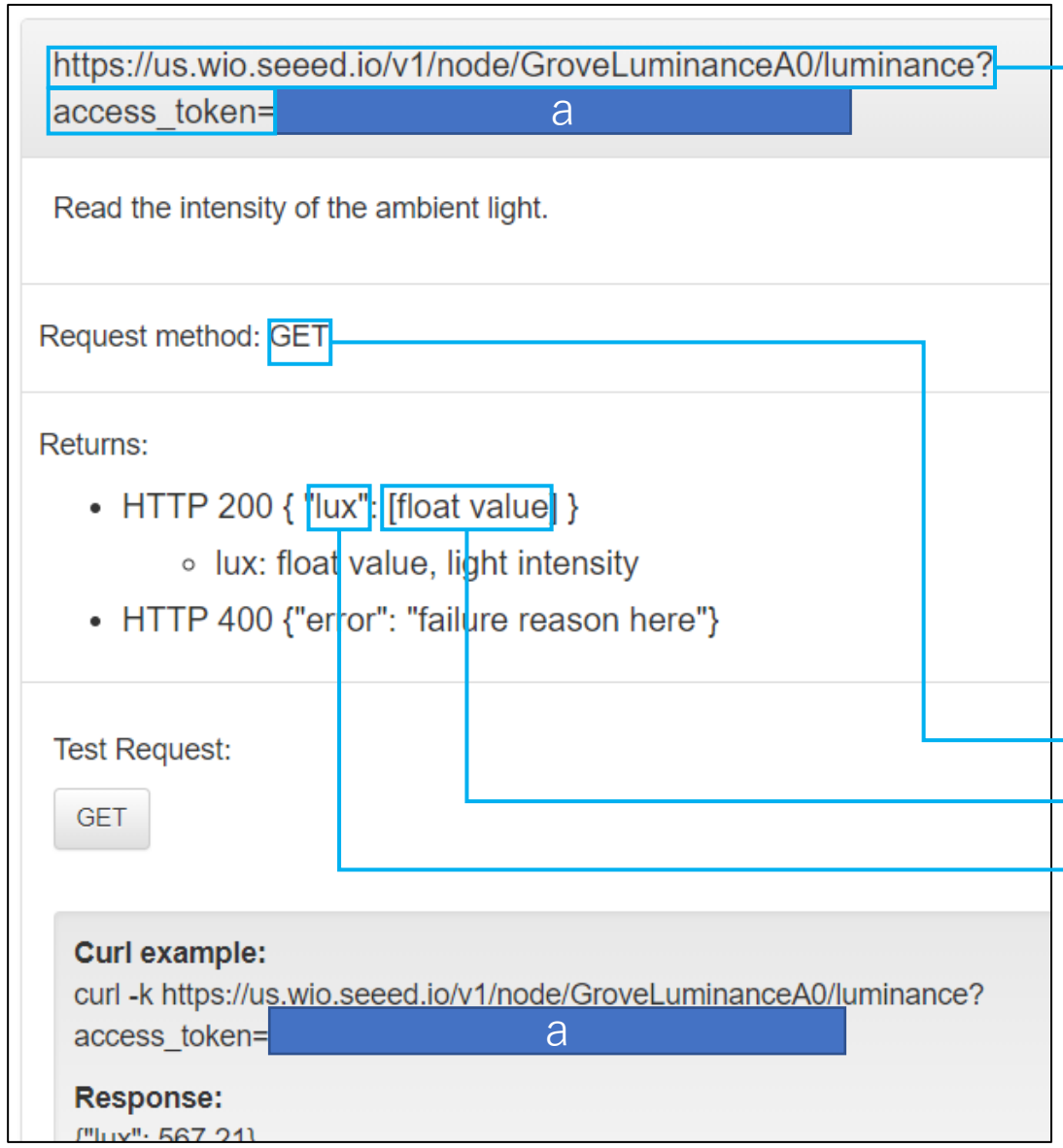
```
from time import sleep
sleep(5)                # 5sec待つ
```



- 例題(10_WioLight.py):ソースコード

```
# -*- coding: utf-8 -*-
import requests
import json

# トークン
token = [redacted]

# 明るさの取得関数
def getLux(my_token):
    url = "https://us.wio.seeed.io/v1/node/GroveLuminanceA0/luminance?access_token=" + my_token
    headers = {"content-type": "application/json"}
    r = requests.get(url, headers=headers)
    data = r.json()
    try:
        lux = data["lux"]
    except:
        lux = None
    return lux


l = getLux(token)
if not l == None:
    print('lux:' + str(l))
```

ViewAPIとソースコードの対応




```
# -*- coding: utf-8 -*-
import requests
import json

# トークン
token = '                                a                                '

# 明るさの取得関数
def getLux(my_token):
    url =
"https://us.wio.seeed.io/v1/node/GroveLuminanceA0/luminance?access
_token=" + my_token
    headers = {"content-type": "application/json"}
    r = requests.get(url, headers=headers)
    data = r.json()
    try:
        lux = data[ "lux" ]      ← HTTP200(成功)の時はコチラの処理
    except:
        lux = None               ← HTTP400(失敗)の時はコチラの処理
    return lux

l = getLux(token)
if not l == None:                ← REST APIが失敗していなければ
    print('lux:' + str(l))
```

JSONとは

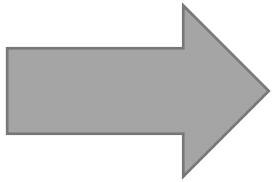
- JavaScript Object Notationの略
  - JavaScriptのオブジェクトの書き方を元にしているらしい
- テキストベースのデータフォーマット
  - データのやりとりをする時のフォーマットの一種
  - 似た用途のフォーマットとして「XML」がある
  - 最近では非常にスタンダードなフォーマット
- JSONデータのエンコーダ／デコーダ
  - 変数→JSONデータへの変換
  - JSONデータ→変数への変換　が可能

```
{
   “通常":“hoge",
   “配列":["1","2","test"],
   “配列の中に構造":[
      {
         “生徒A":{
            “数学":“90",
            “英語”:“68”,
            “電磁気”:“70”,
         }
      },
      {
         “生徒B":{
            “数学":“45",
            “英語”:“98”,
            “電磁気”:“55”,
         }
      }
   ]
}
```

JSONデータのデコード

- Pythonの場合、request.get()の戻り値のインスタンスよりjson()メソッドでJSON形式からのデータ変換が可能
- JSON形式のデータは連想配列に変換される

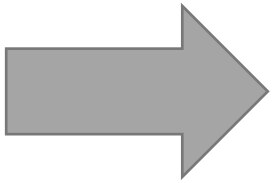
```
{
   “math": 87,
   “english": 49,
   “science": 78
}
```



```
data = r.json ()
print( data[“math”])        // 87と出力
print( data[“english”])     // 49と出力
print( data[“science"])     // 78と出力
```

- 今回の場合、APIサーバからの応答がJSON形式で返ってくる
  - 光センサの事例

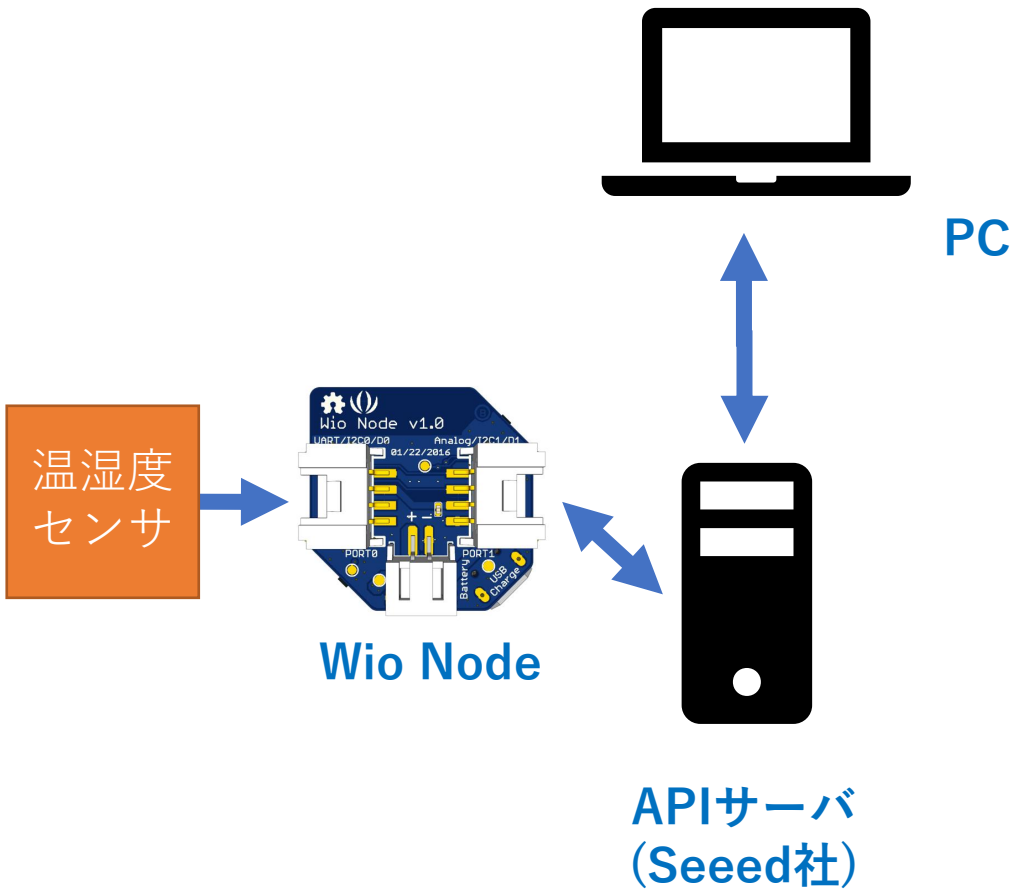
```
{"lux": 976.83}
```



```
data = r.json()
print( data[“lux”])          // 976.83と出力
```

練習②：温湿度センサも制御

- 温湿度センサも制御して値を表示してみましょう！



[参考] C言語やPHPの場合はcURLでOK

- cURL(カール)と読みます
- フリーで扱いやすい、様々なプロトコルをサポートするデータ転送ライブラリのプロジェクト
- libcurl
  - クライアントサイドのURL転送ライブラリ
  - 様々なプロトコルをサポート(FTP, HTTP, HTTPS, TELNETなど)
  - PHPにもcURLライブラリがあり
    - 今回はこれを使用し、REST APIを叩く！
- cURL
  - コマンドラインツール
  - Linuxはもちろん、Windowsでも動作可

Ambientでグラフ化

センサデータをグラフ化する

クラウドコンピューティングの分類

SaaS
Software as a Serice

PaaS
Platform as a Serice

IaaS
Infrastructure as a Serice

・安価な傾向
・自由度は低い
・求められるスキルは低め

・自由度は高い
・要スキル

例：Gmail, Microsoft365,
Dropbox, サイボウズ, Ambient

例：AWS Lamda, AWS IoT Core
Azure IoT Hub,他

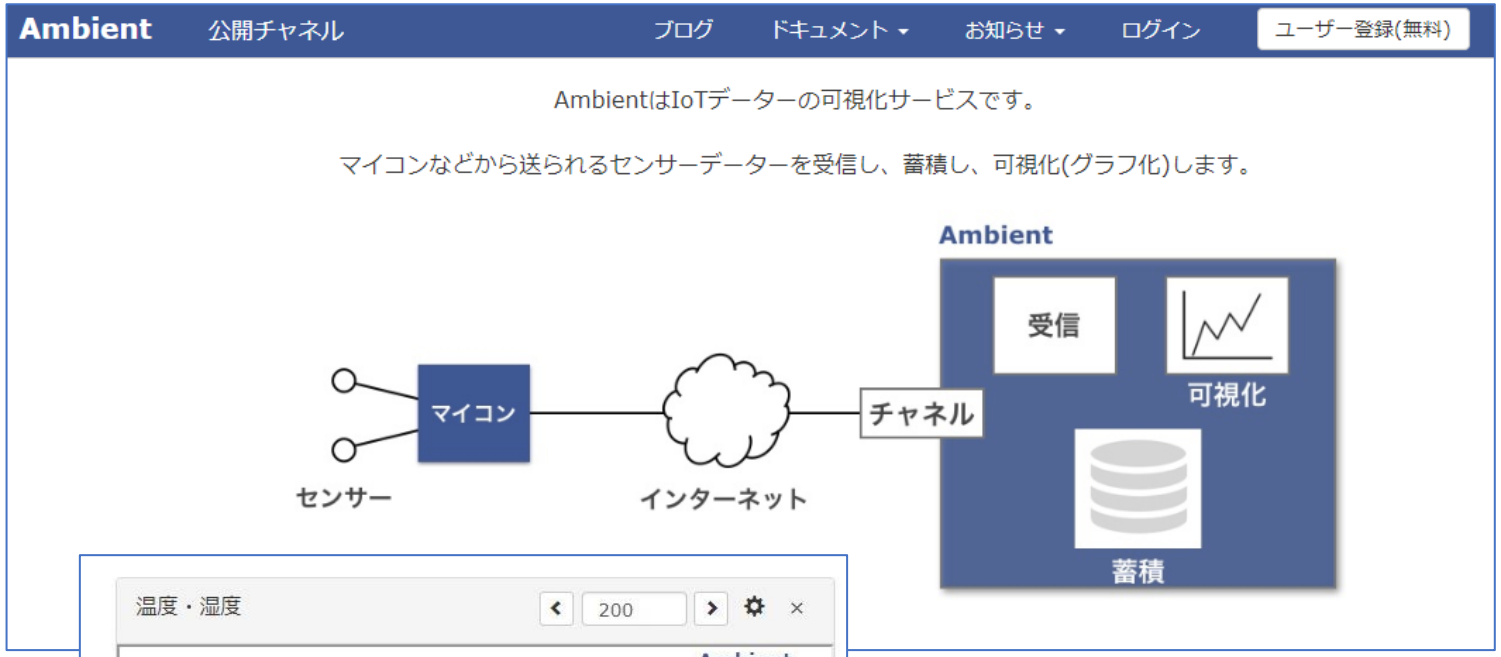
例：AWS EC2, Azure VM他

クラウド上で提供されるソフトウェア
特定の用途を実現する為のソフトウェアが完成された状態で提供される

クラウド上で提供される実行環境
アプリやDB等が動作する実行環境などが提供される

仮想サーバや外部ストレージ,
ファイアウォール等のインターネット上のサービスとして提供する

ambientによるお手軽IoT



IoTデータの可視化サービス

- 簡単かつ強力なグラフ化
- ユーザ登録・無料

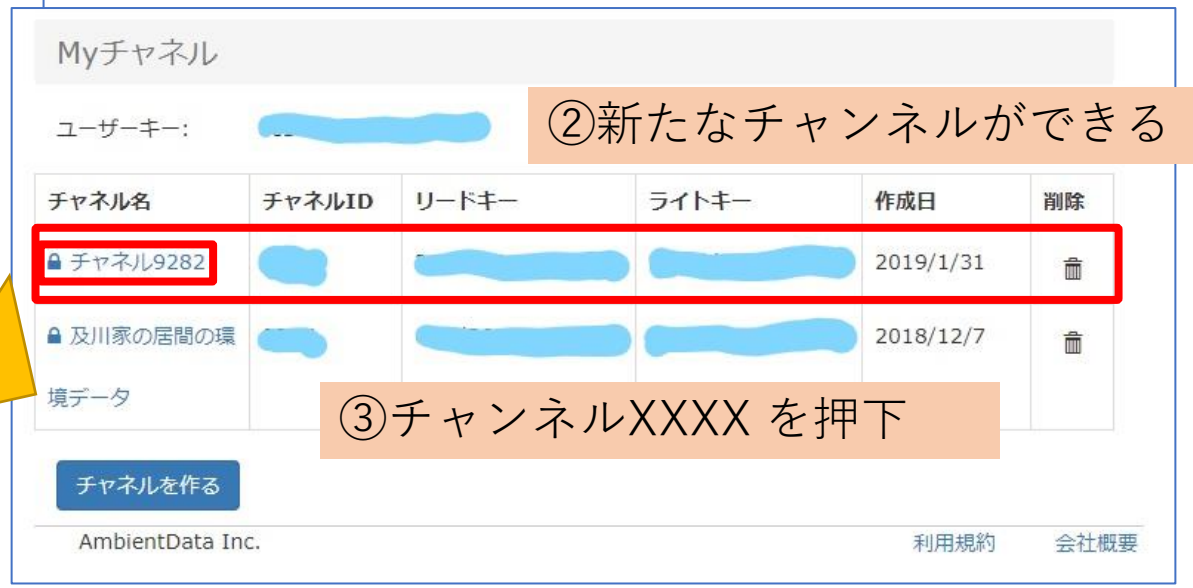
- 各種マイコンへ対応
  (ライブラリ＆リファレンス)
  - ESP8266, mbed LPC1768, M5Stack
  - Raspberry Pi(JavaScript, Python)

- 公開/非公開設定が可能

Ambient活用の為の準備①

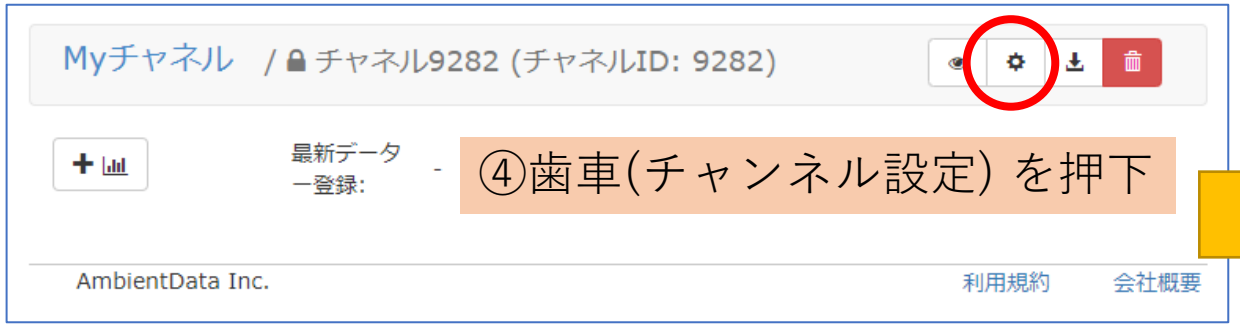
1. Ambientのアカウントを取得
2. Ambientでチャンネル作成
   - ログイン後、「MYチャンネル」を押下

①チャンネルを作る を押下



②新たなチャンネルができる

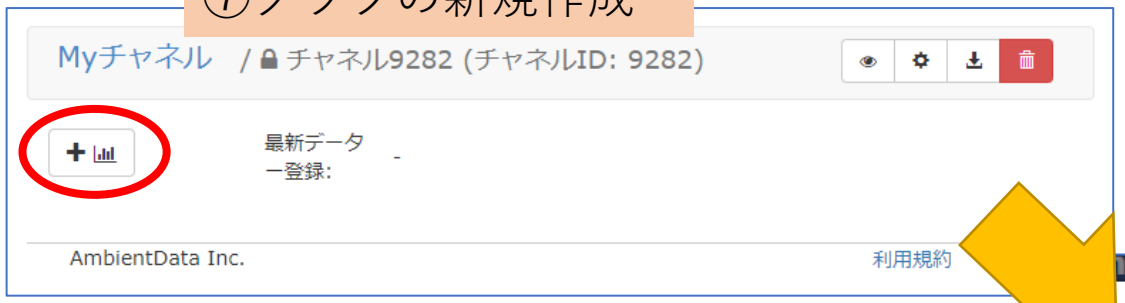
③チャンネルXXXX を押下



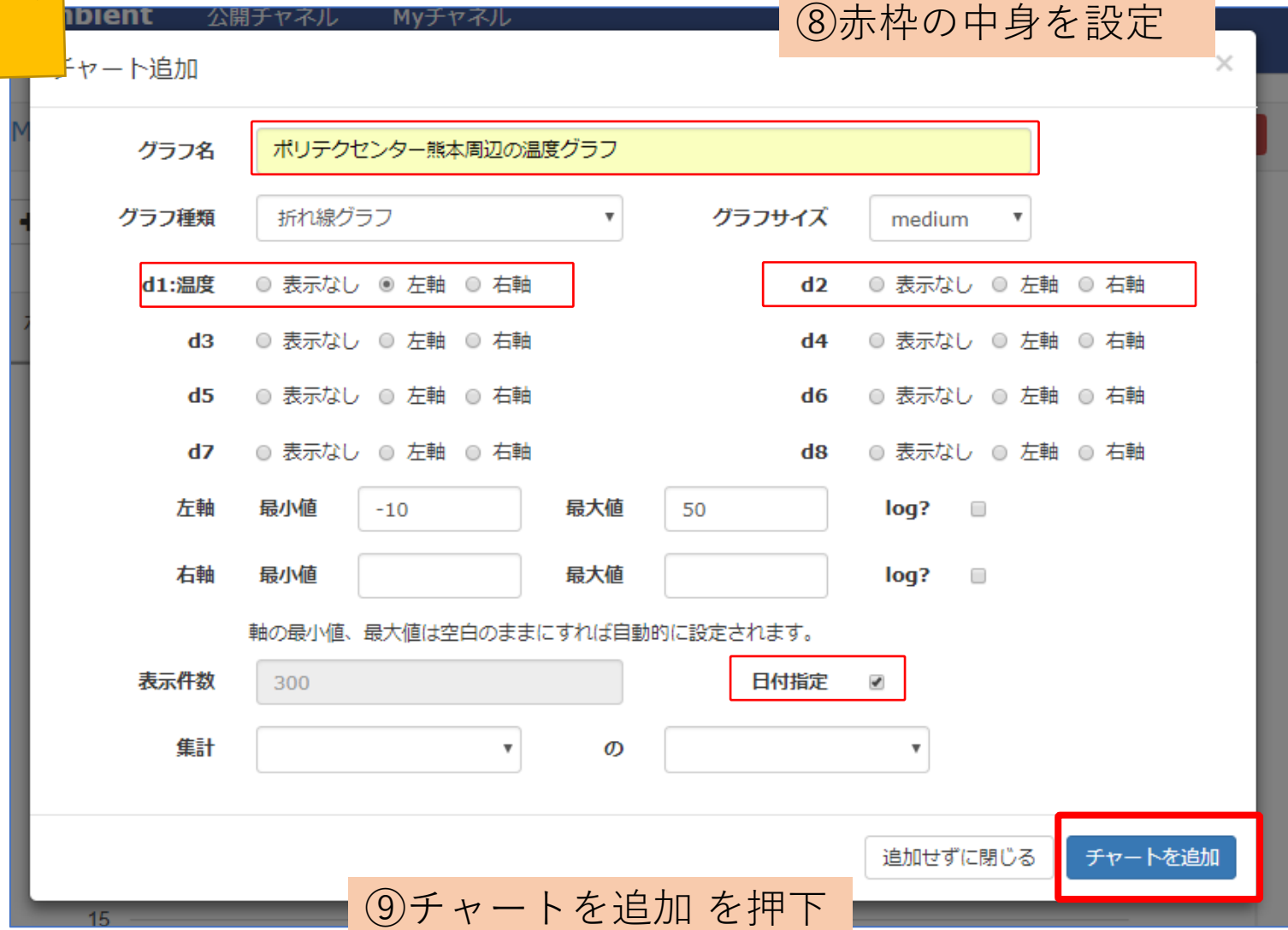
⑤赤枠の中身を設定

チャンネル名	：	WioNodeセミナー
説明	：	（任意）
データー１	：	温度
データー２	：	湿度
データー３	：	明るさ
公開チャンネル？	：	レ
場所を表示	：	レ
緯度・経度	：	任意

⑦グラフの新規作成



⑧赤枠の中身を設定



⑧赤枠の中身を設定

グラフ名　：　部屋Aの温湿度グラフ
D1　：　左軸
D2　：　右軸
D3　：　表示なし
日付指定　：　レ

⑨チャートを追加 を押下

Ambientのpythonライブラリの追加①

- Pythonからambientへデータプッシュする為に
  - まずは、**Git for Windows**のインストール
    - https://gitforwindows.org/
  - Ambientのpythonライブラリをインストール
    - Pythonの環境をクリック
    - PyPIとインストールされたパッケージの検索に **”git+https://github.com/AmbientDataInc/ambient-python-lib.git”**と入力
    - 次のコマンドを実行する:pip install git+https://github.com/AmbientDataInc/ambient-python-lib.git
    - をクリック

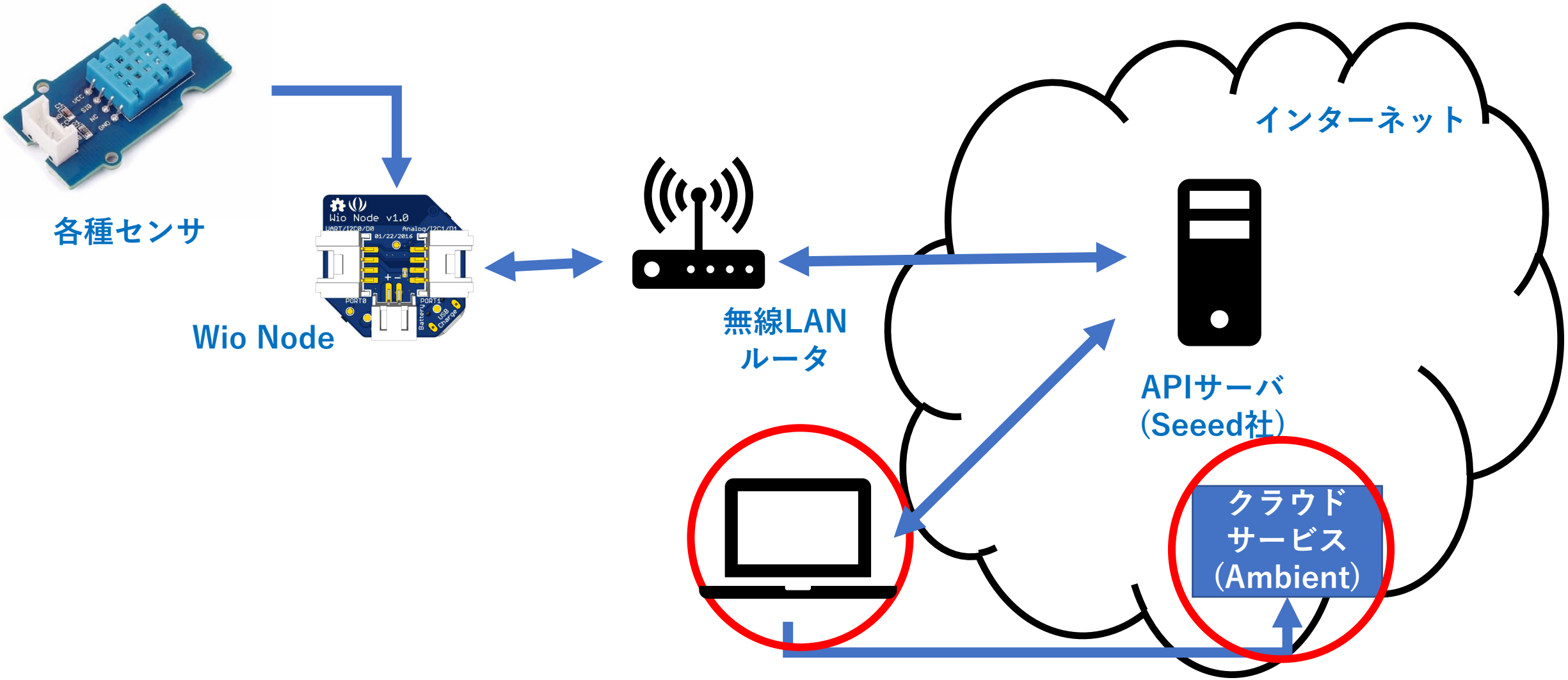
Ambientのpythonライブラリの追加②

- Pythonでの使い方(最低限)

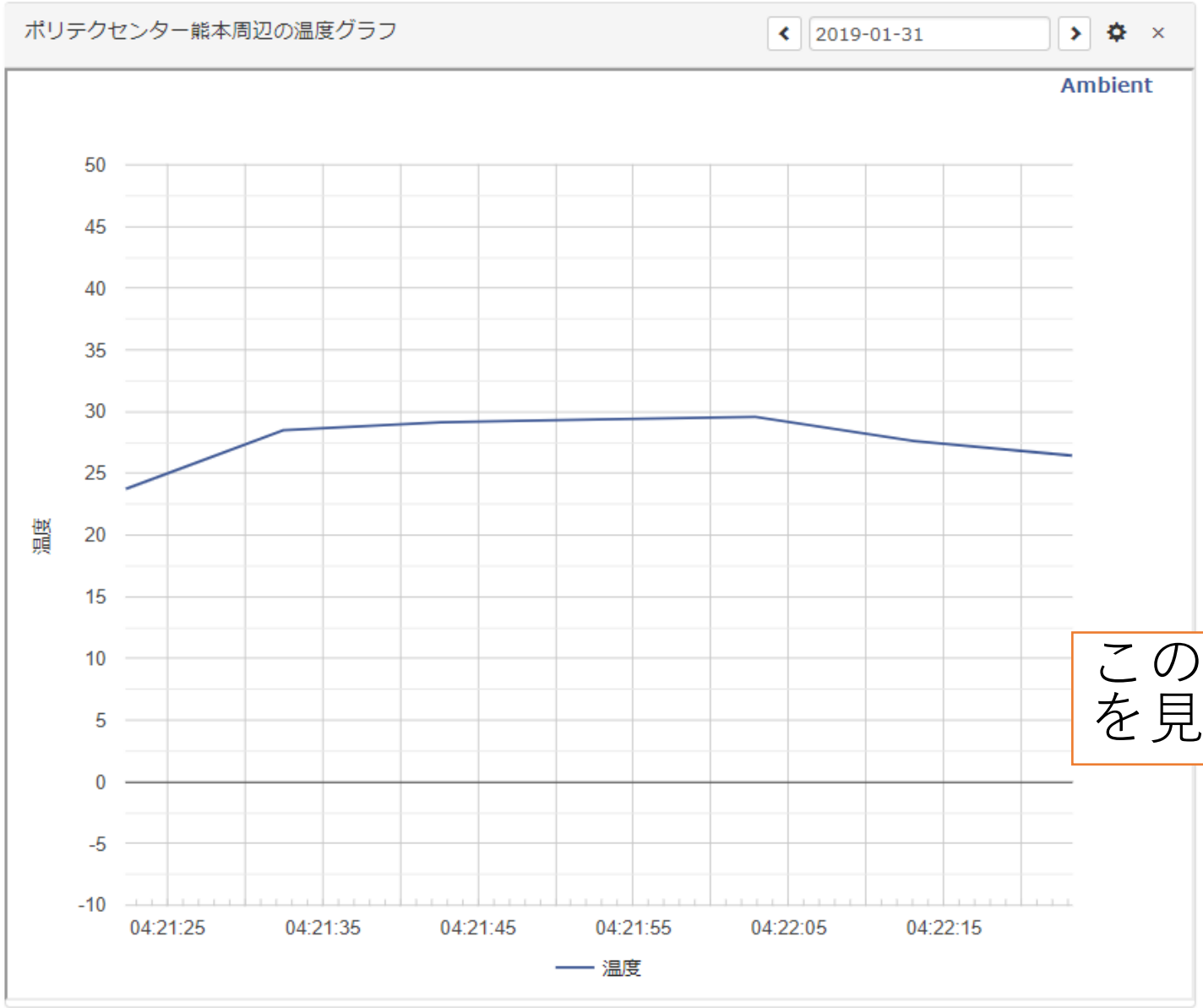
```
import ambient

am = ambient.Ambient(100, "your_writeKey") # 自分のチャネルID、ライトキーに置き換え

r = am.send({"d1": temp, "d2": humid})
```

- その他の機能についてはGithubを参照
  - https://github.com/AmbientDataInc/ambient-python-lib

WioNode+Ambientの制御イメージ



完成すると…

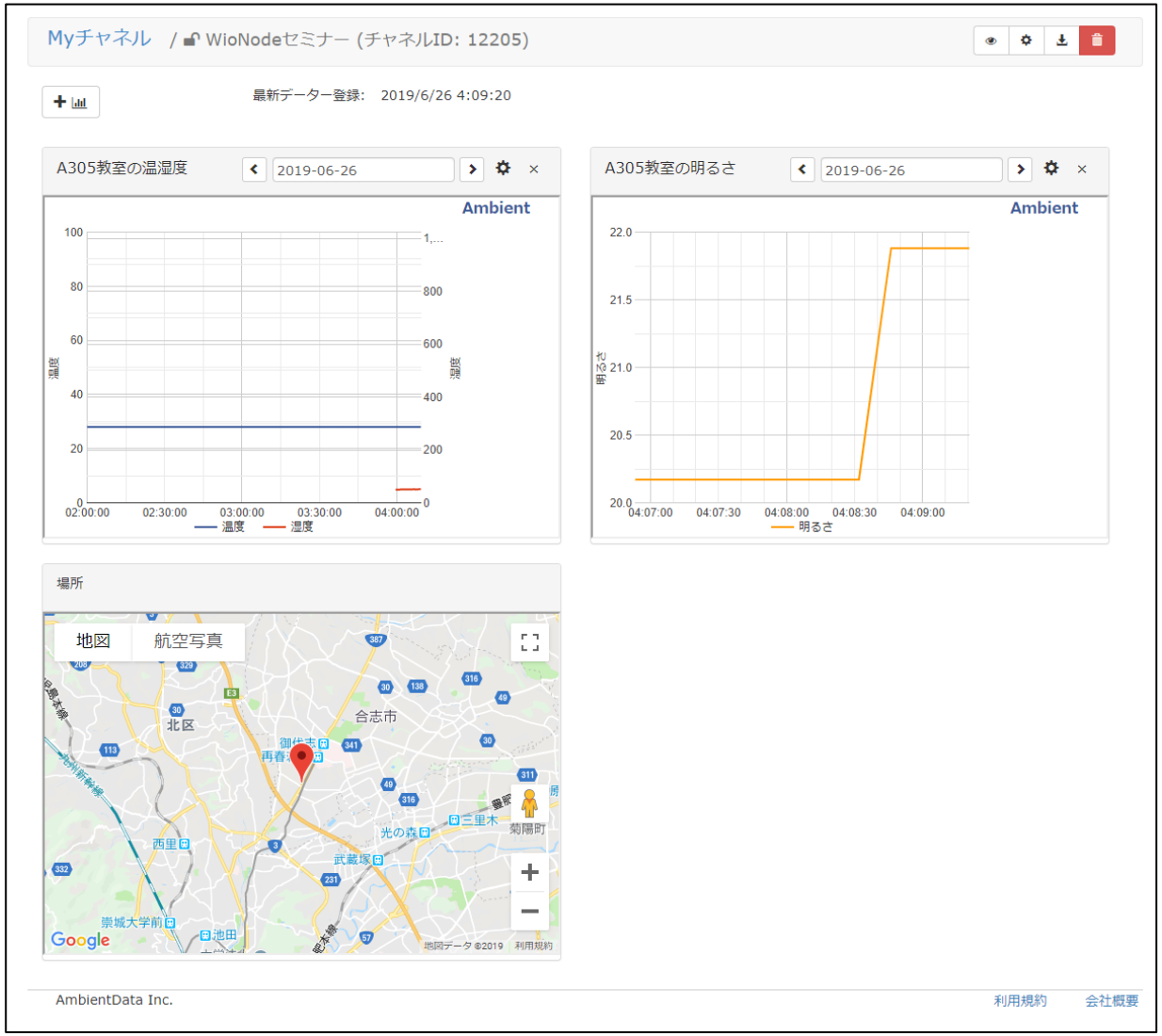


このようなイメージでグラフを見ることができます

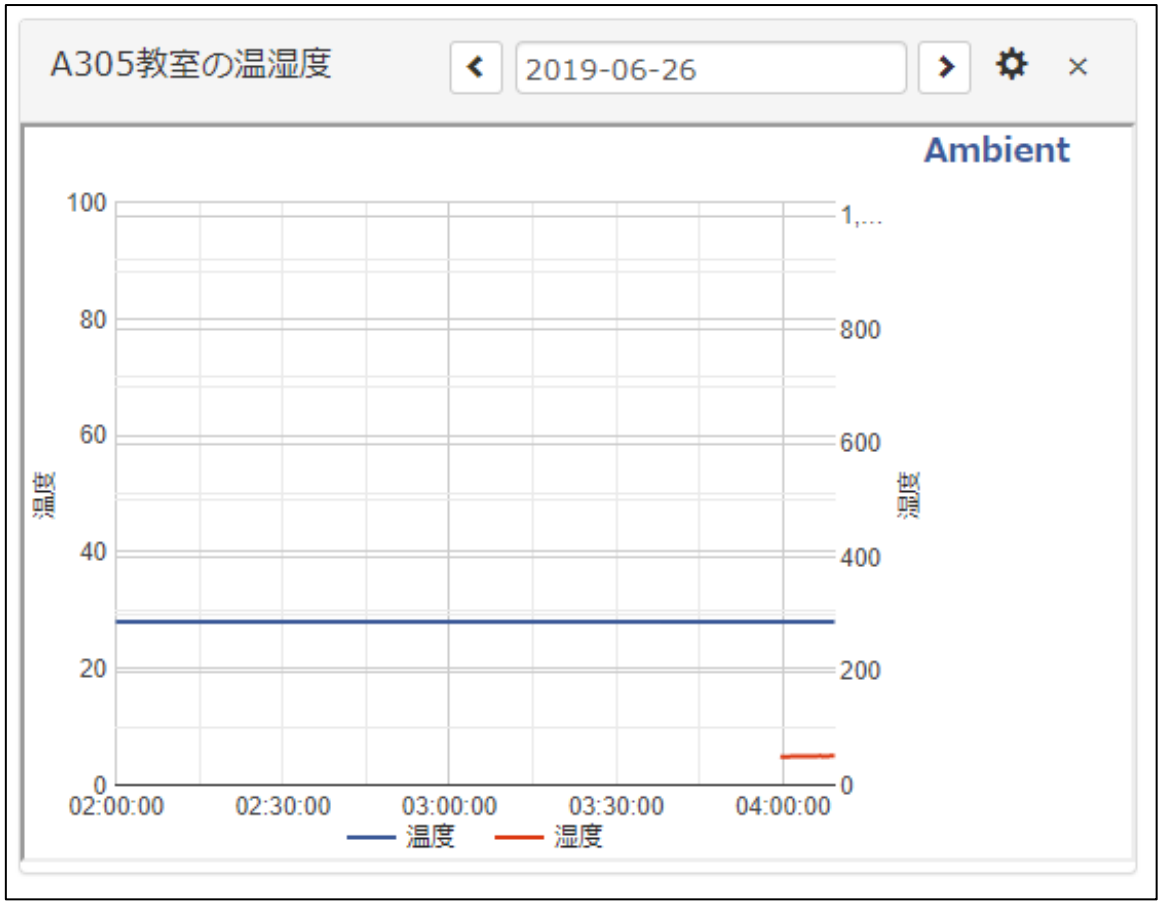
練習：湿度や明るさもグラフ化(可視化)

1. 既存のグラフのd2の項目に湿度の項目も足してみましょう
   → グラフは右軸で表示すること
2. 新規にグラフを作成し、明るさを可視化してみましょう

今回、Ambientで作成するグラフ



今回、Ambientで作成するグラフ(拡大)

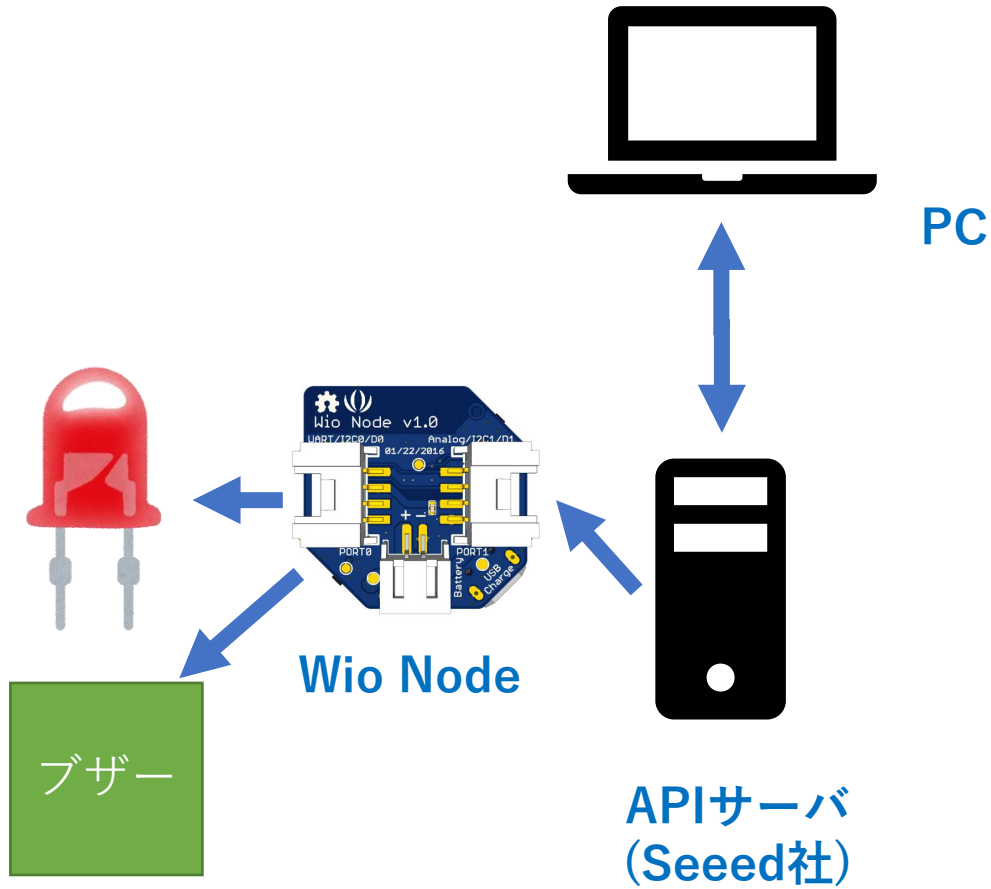


PythonでWioNode制御

アクチュエータ編

練習①：LEDを点灯／消灯 制御する

- 例題
  - PythonよりWio NodeのLEDの点灯/消灯制御をしよう！
- 課題
  - LEDをx秒のhighパルスで制御しよう
    - ヒント： Wio NodeのREST APIに該当の機能がある
  - ブザー制御も実現しよう！



- 例題(30_WioLED.py):ソースコード

```
# -*- coding: utf-8 -*-
import requests
import json
from time import sleep

# トークン
token = 'f0f7e685afd3d9d8174ff950c13eff91'

# LEDのON/OFF制御関数
def setLed(my_token, onoff):
    url =
"https://us.wio.seeed.io/v1/node/GenericDOutD1/onoff/"
    url += str(onoff)
    url += "?access_token=" + my_token
    r = requests.post(url)
    data = r.json()
    try:
        result = data["result"]
    except:
        result = None
    return result
```

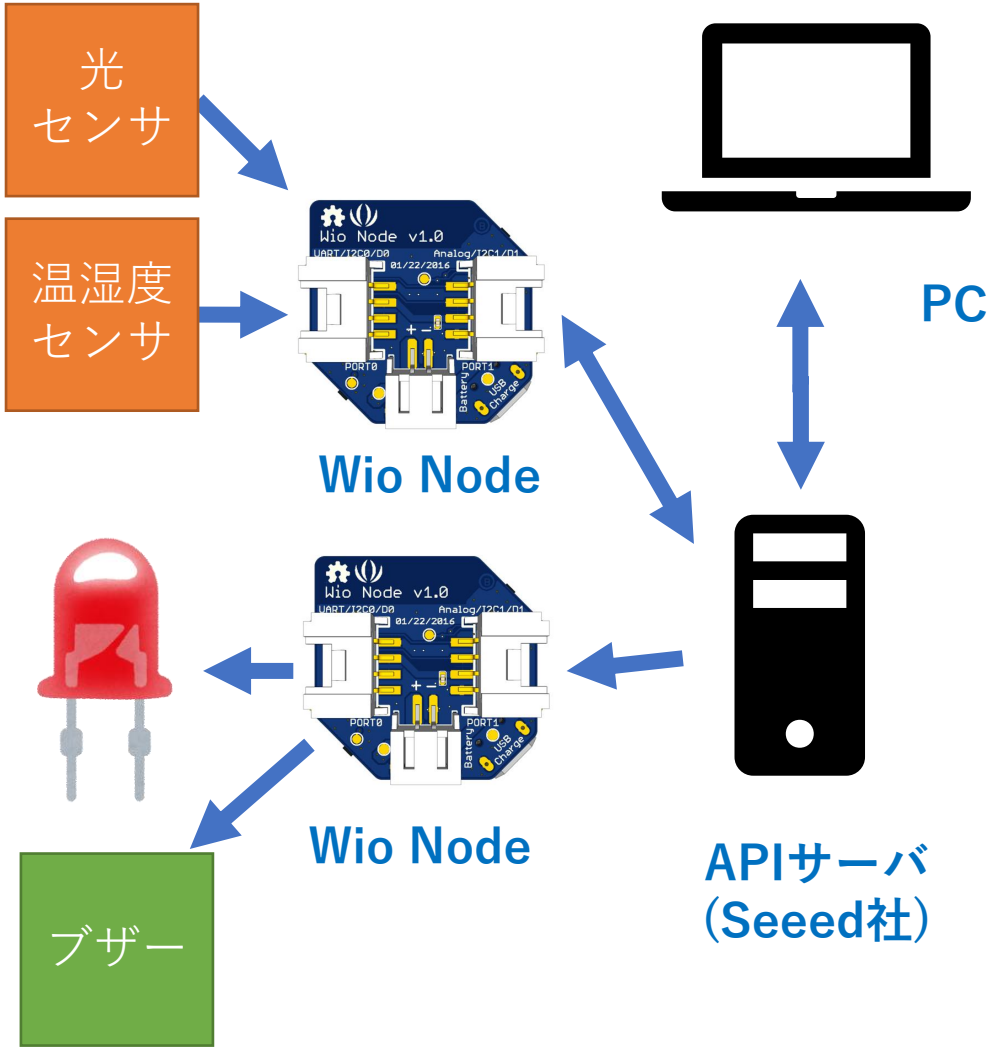
```
while 1:
    print("plese iput command(0:off, 1:on, 9:exit) :",
end='');
    cmd = input()
    if cmd == "0":
        r = setLed(token, 0)
    elif cmd == "1":
        r = setLed(token, 1)
    elif cmd == '9':
        print('exit...')
        break
    else:
        print('error')
```

アラート機能の実装

センサが一定の値を超えたら通知

練習①：光量の取得及びLED点灯／消灯

- 練習A
  - 室内が暗くなったらLEDが点灯するようなプログラムを作成せよ
- 練習B
  - 練習Aにブザーの鳴動機能も追加せよ
- 練習C
  - 室内の明るさを室温に変えて実現せよ。35度を超えたらLEDおよびブザーをオンにする

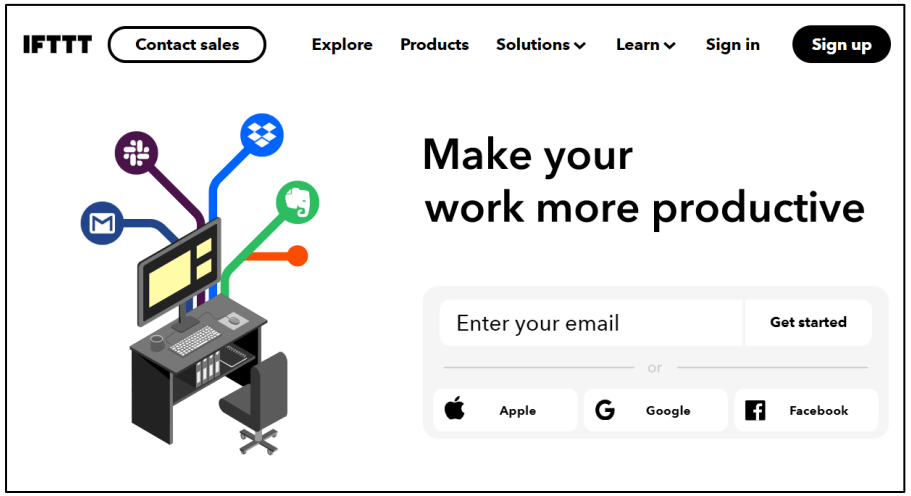


高機能アラート機能の実装

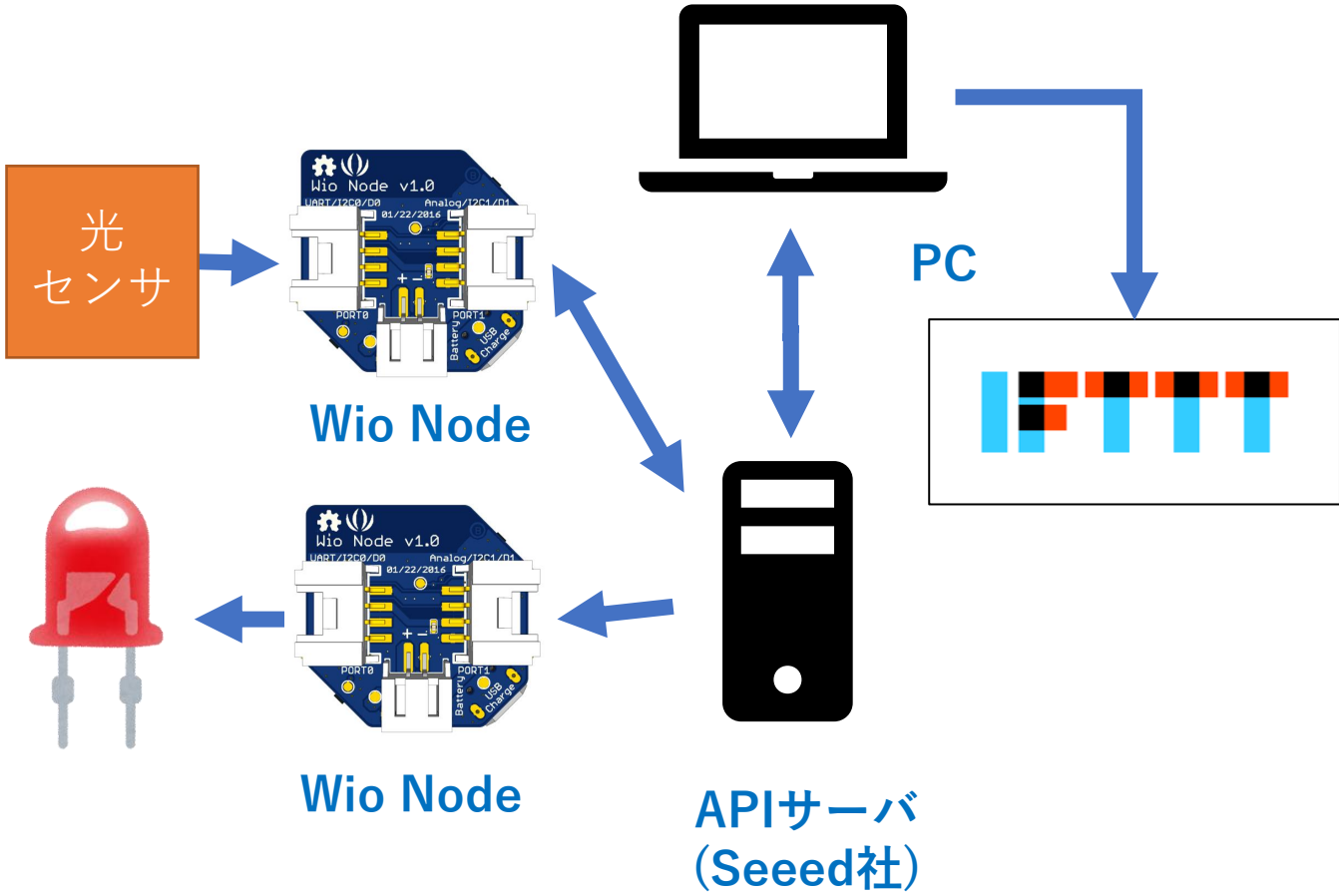
センサが一定の値を超えたら

メール通知

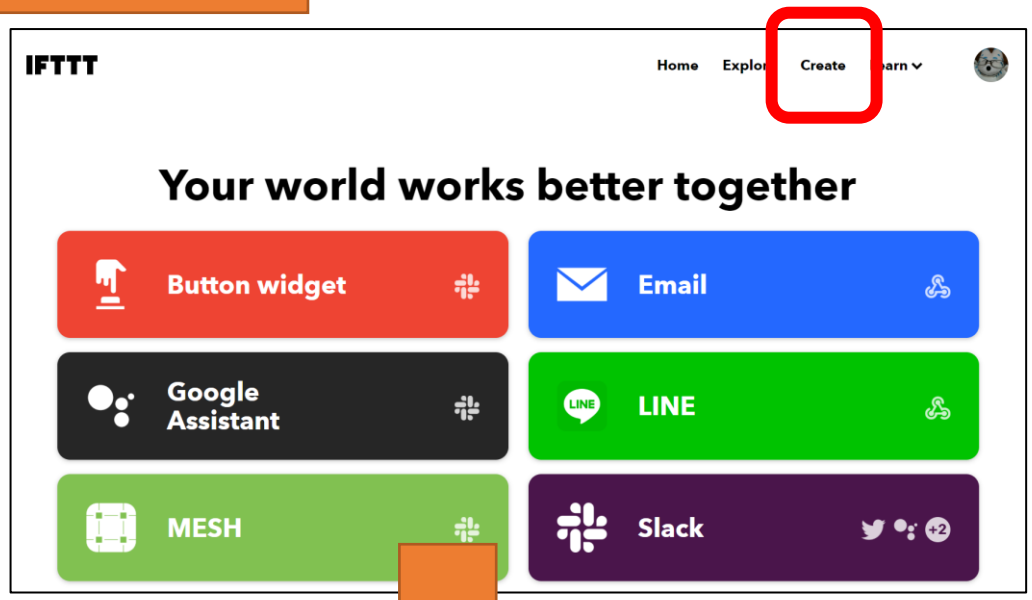
IFTTTを活用してメール通知する

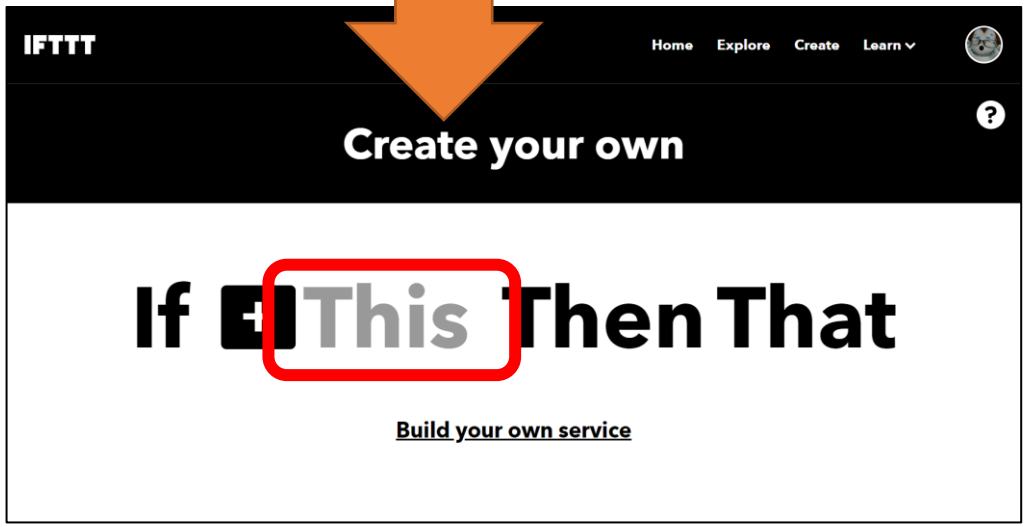


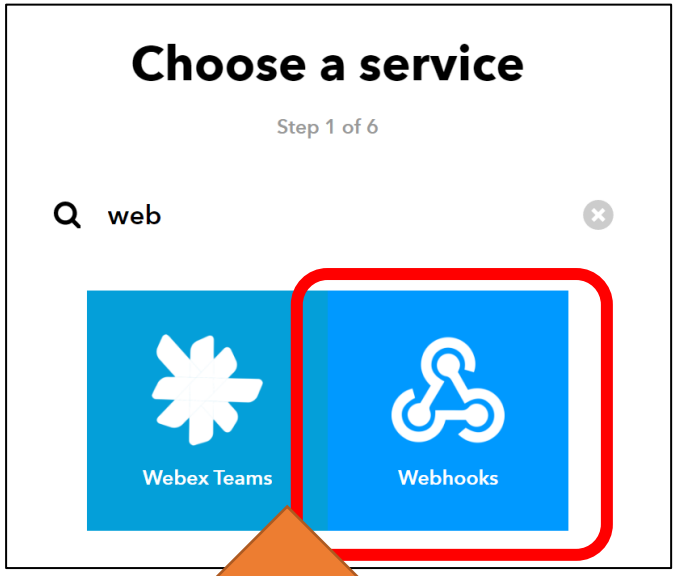
- IFTTTと呼ばれるWebサービスを活用して，アラートをメール送信により通知する

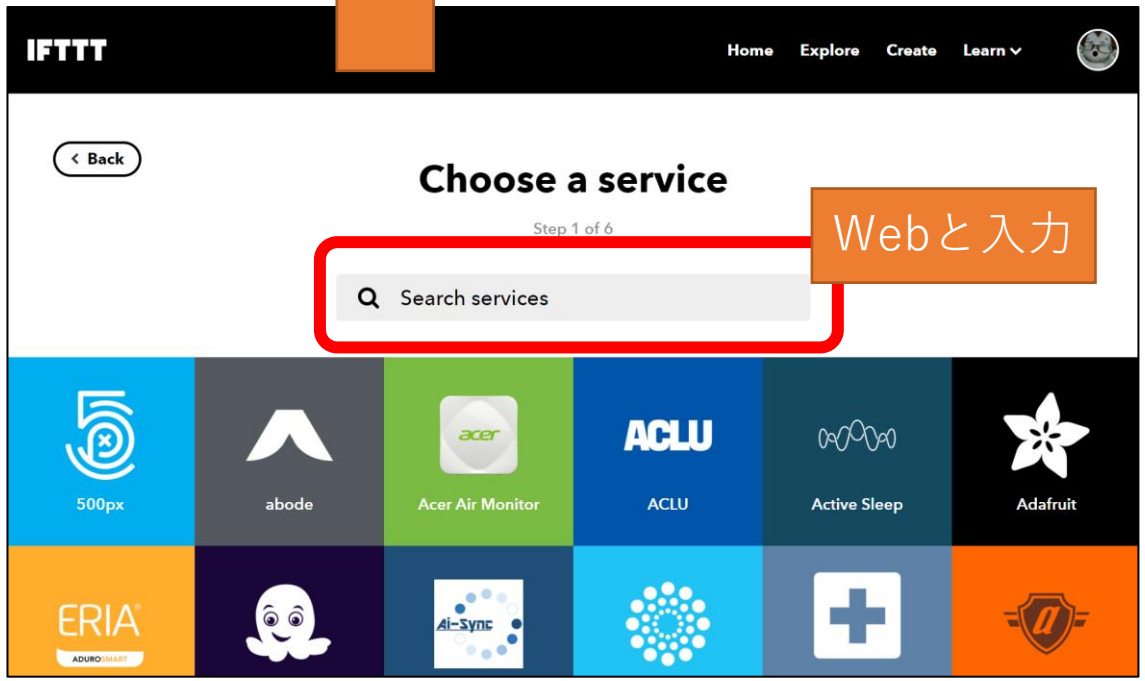


IFTTT設定①


IFTTT
Home
Explor
Create
Your world works better together
Button widget
Email
Google Assistant
LINE
MESH
Slack

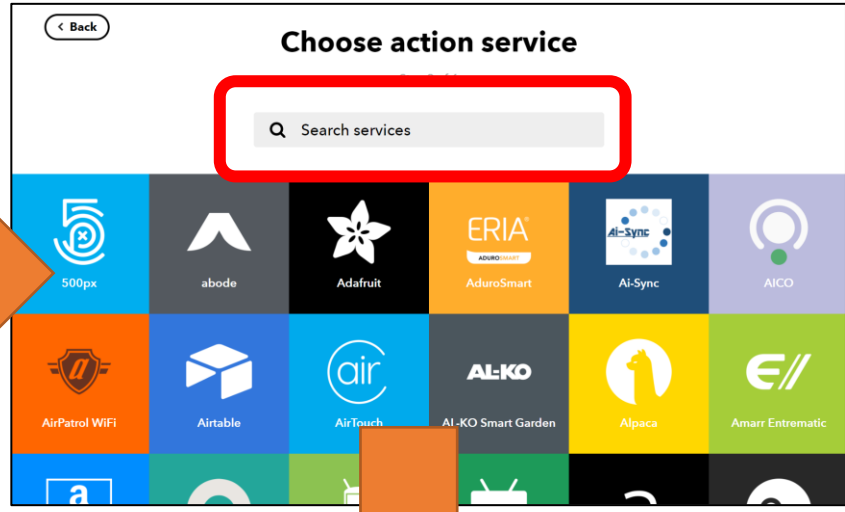

IFTTT
Home
Explore
Create
Learn
Create your own
If This Then That
Build your own service


Choose a service
Step 1 of 6
web
Webex Teams
Webhooks

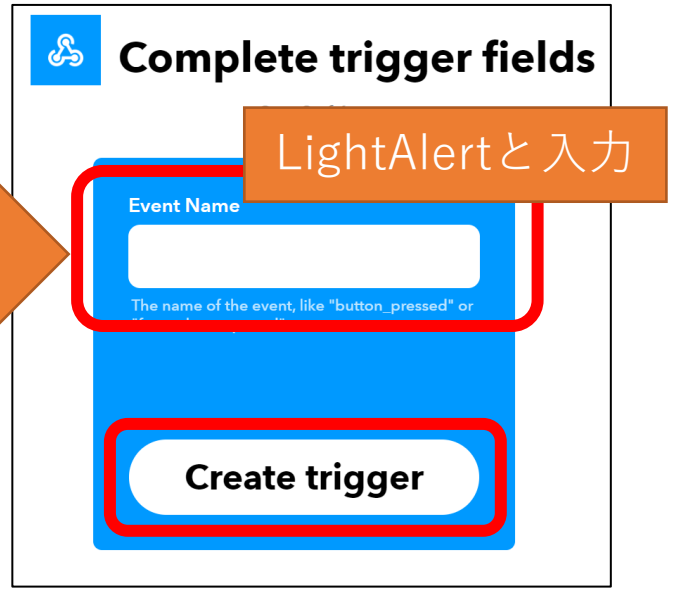

IFTTT
Home
Explore
Create
Learn
Back
Choose a service
Step 1 of 6
Search services
Webと入力
500px
abode
Acer Air Monitor
ACLU
Active Sleep
Adafruit
ERIA

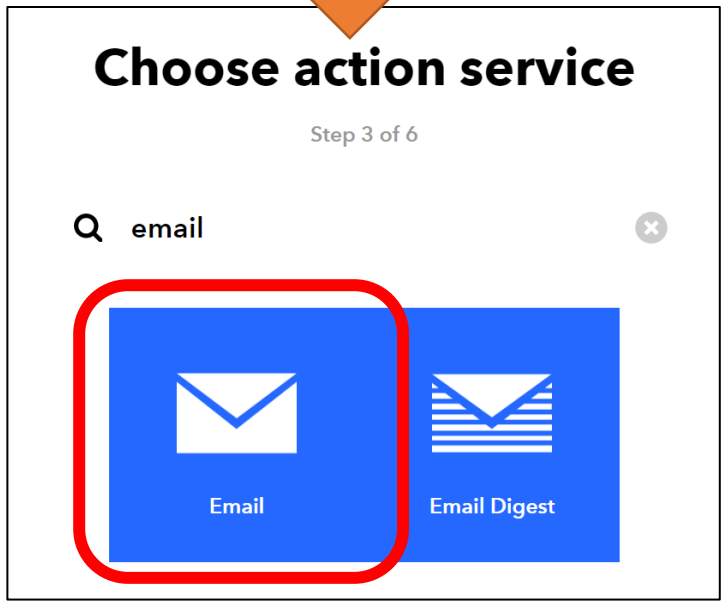
IFTTT設定②


Back
If
Then
That

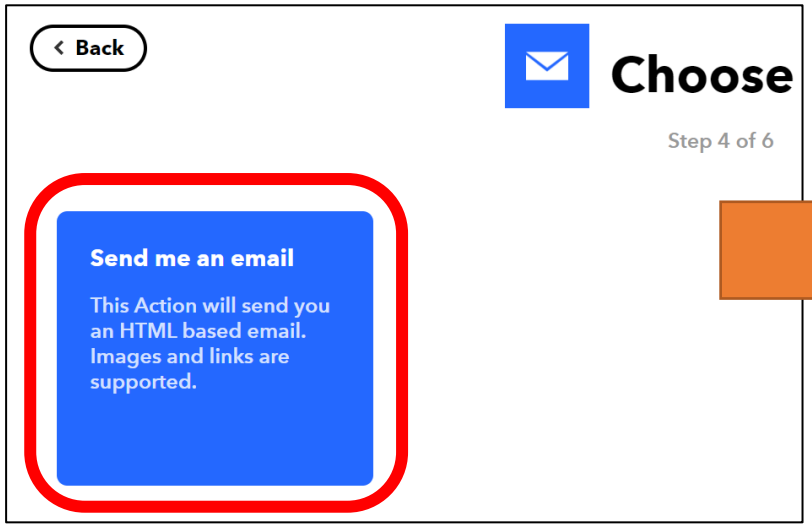

Back
Choose action service
Search services
500px
abode
Adafruit
ERIA
AduroSmart
Ai-Sync
AICO
AirPatrol WiFi
Airtable
AirTouch
AI-KO Smart Garden
Alpaca
Amarr Entrematic

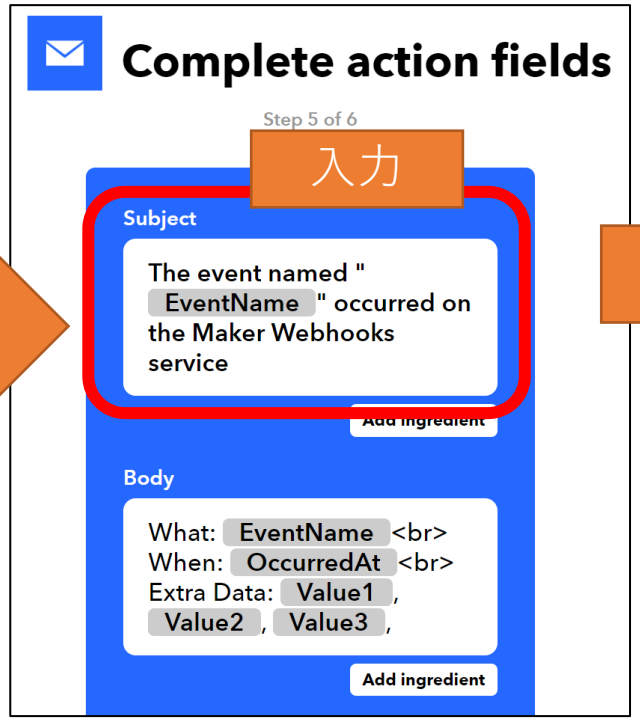

Back
Choose trigger
Step 2 of 6
Receive a web request
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

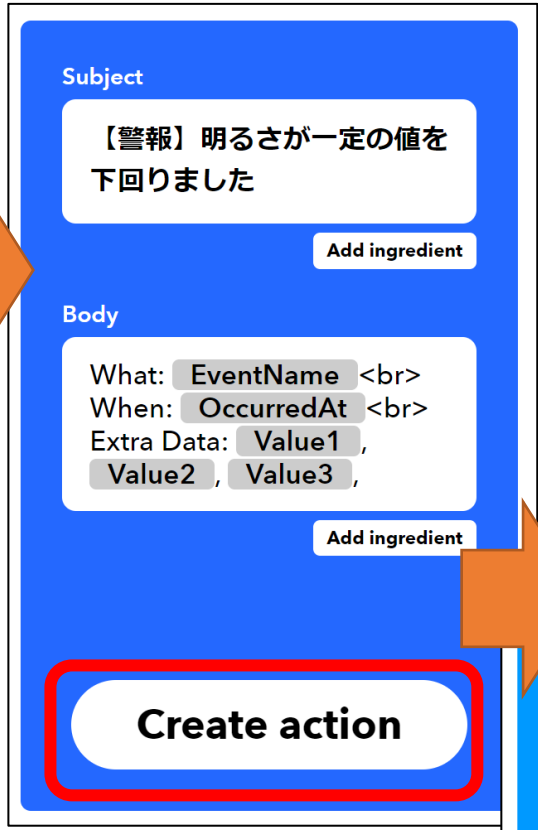

Complete trigger fields
LightAlertと入力
Event Name
The name of the event, like "button_pressed" or
Create trigger

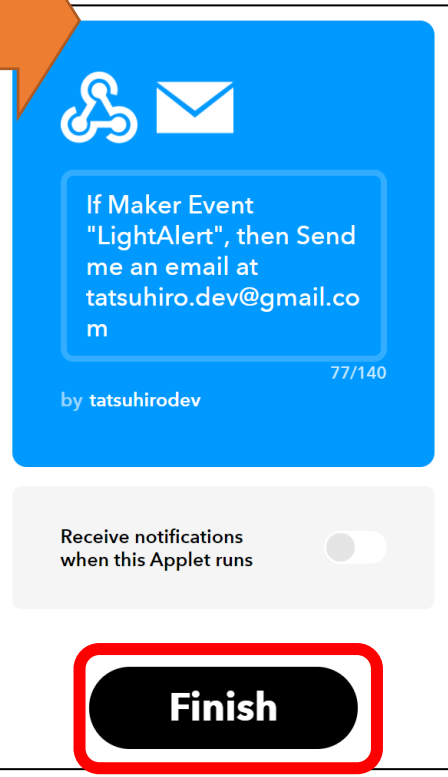

Choose action service
Step 3 of 6
email
Email
Email Digest

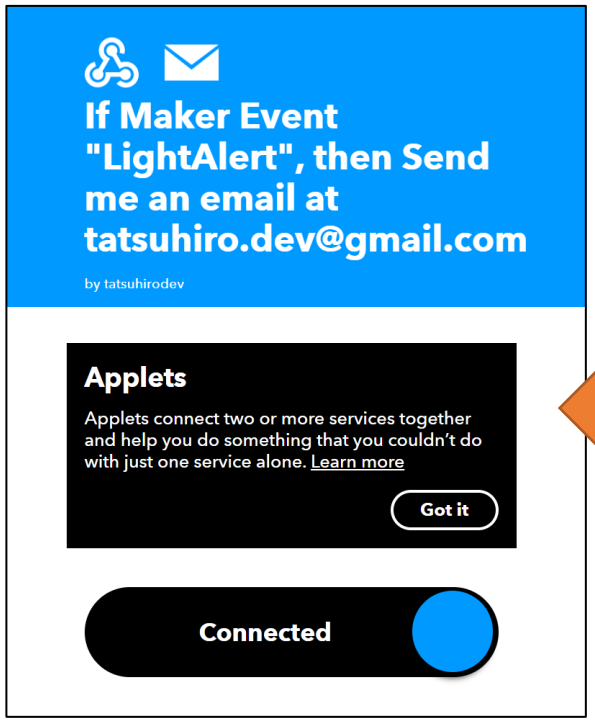
IFTTT設定③


Back
Choose
Step 4 of 6
Send me an email
This Action will send you an HTML based email. Images and links are supported.


Complete action fields
Step 5 of 6
入力
Subject
The event named " EventName " occurred on the Maker Webhooks service
Add ingredient
Body
What: EventName
When: OccurredAt
Extra Data: Value1 , Value2 , Value3 ,
Add ingredient

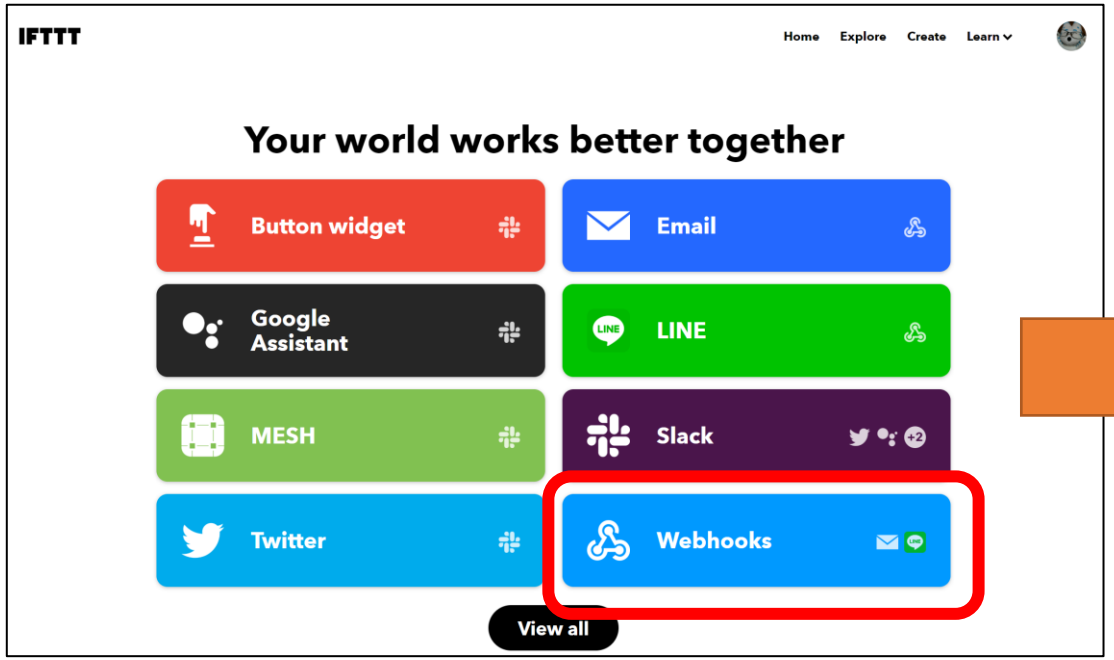

Subject
【警報】明るさが一定の値を下回りました
Add ingredient
Body
What: EventName
When: OccurredAt
Extra Data: Value1 , Value2 , Value3 ,
Add ingredient
Create action

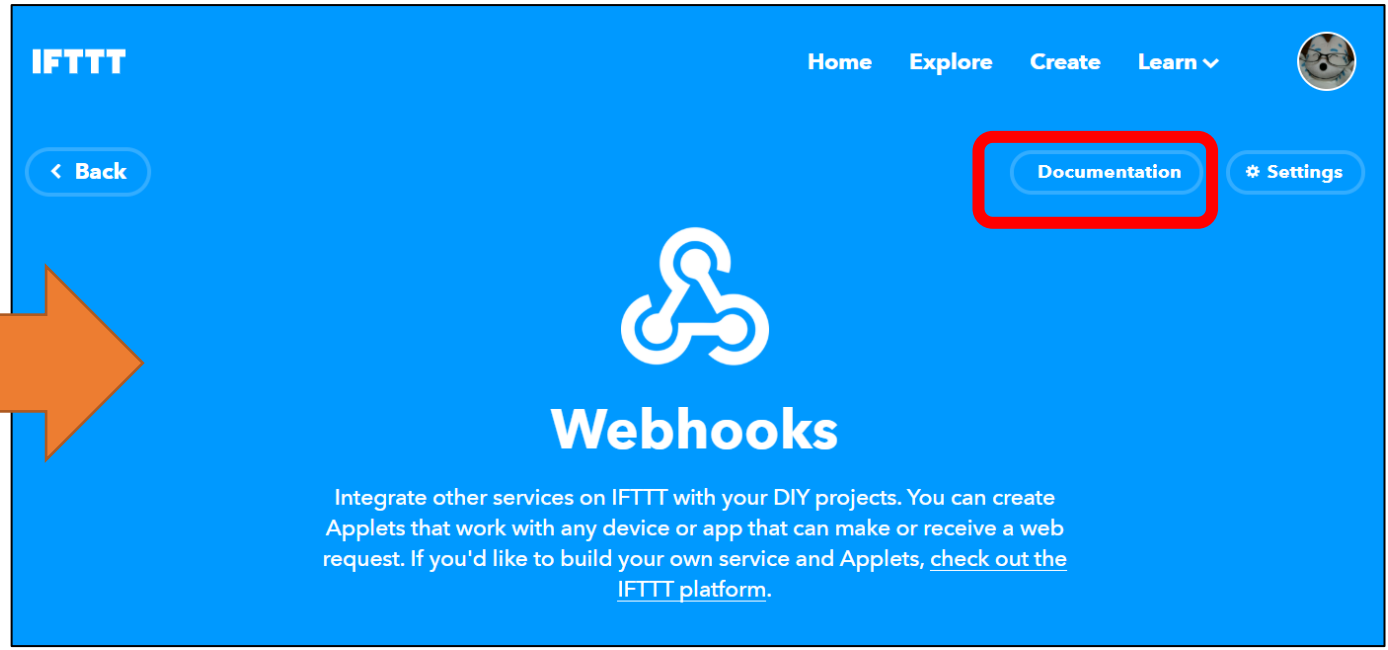

If Maker Event "LightAlert", then Send me an email at tatsuhiro.dev@gmail.com
77/140
by tatsuhirodev
Receive notifications when this Applet runs
Finish


If Maker Event "LightAlert", then Send me an email at tatsuhiro.dev@gmail.com
by tatsuhirodev
Applets
Applets connect two or more services together and help you do something that you couldn't do with just one service alone. Learn more
Got it
Connected

完成

IFTTTの動作確認





- Homeから「Webhook」を選択
- もしくは，URLから直接アクセス
  - https://ifttt.com/maker_webhooks

Your key is: [redacted]

Back to service

To trigger an Event

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigge / LightAlert / ith/key/ [redacted]
```

With an optional JSON body of:

```
{ "value1" : " 100 ", "value2" : " kumamoto ", "value3" : " " }
```

The data is completely optional, and you can also pass `value1`, `value2`, and `value3` as query parameters or form variables. This content will be passed on to the Action in your Recipe.

You can also try it with `curl` from a command line.

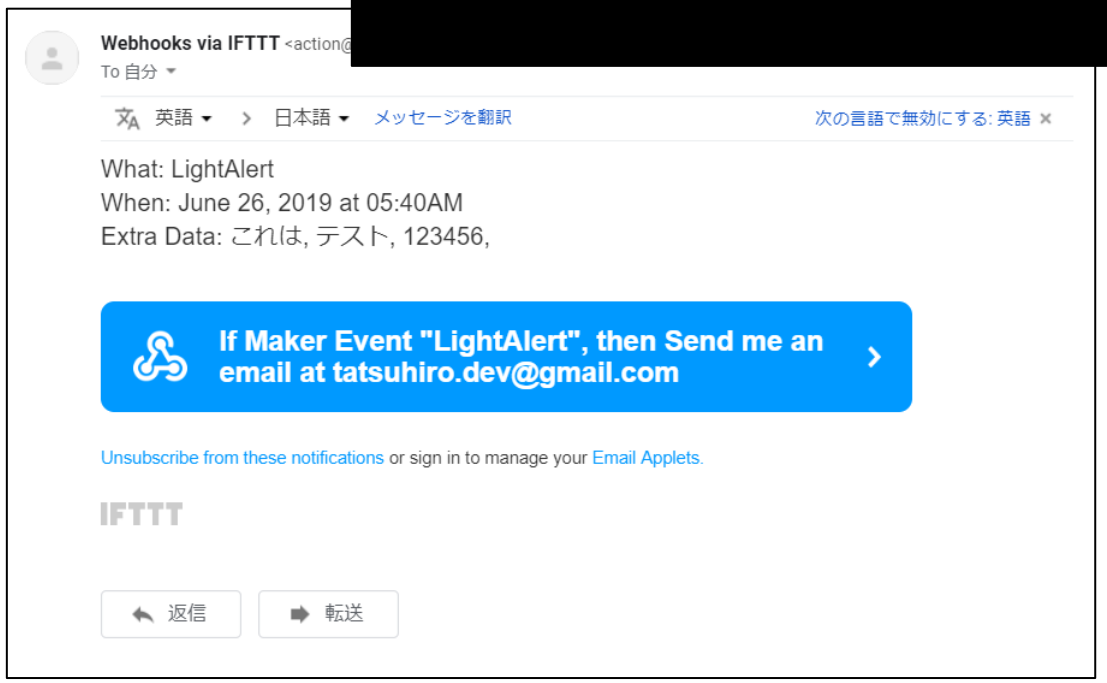
```
curl -X POST -H "Content-Type: application/json" -d '{"value1":"100","value2":"kumamoto"}'
https://maker.ifttt.com/trigger/LightAlert/with/key/ [redacted]
```

Test It

メールが届けばIFTTTの動作確認OK!!

練習：IFTTTにPOSTするだけの処理

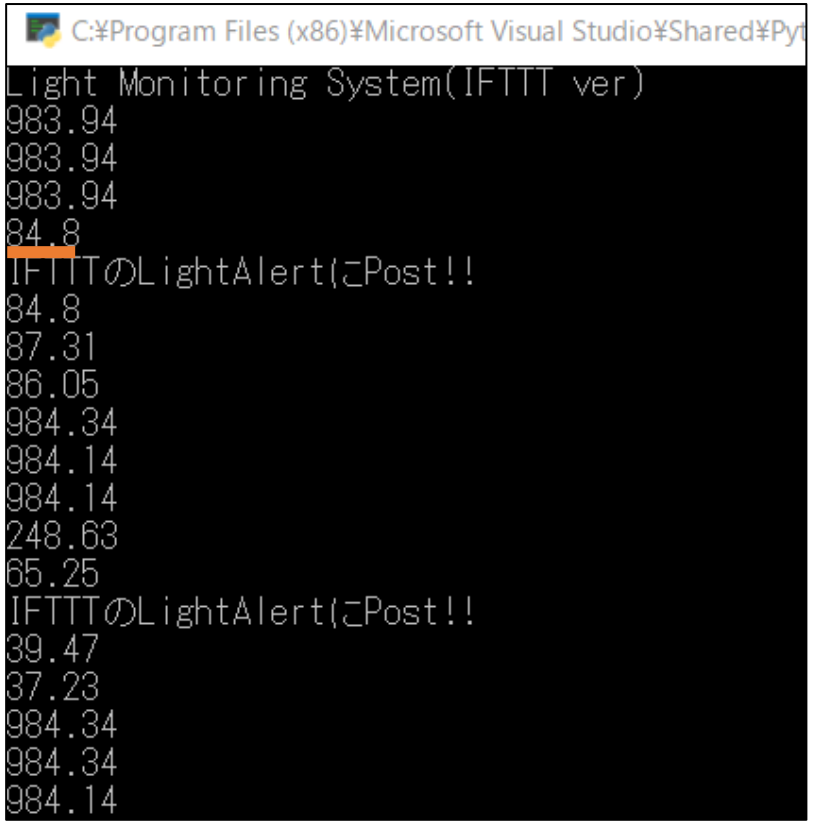
- PythonからIFTTTのWebhook(REST API)を叩くだけのプログラムを作ります

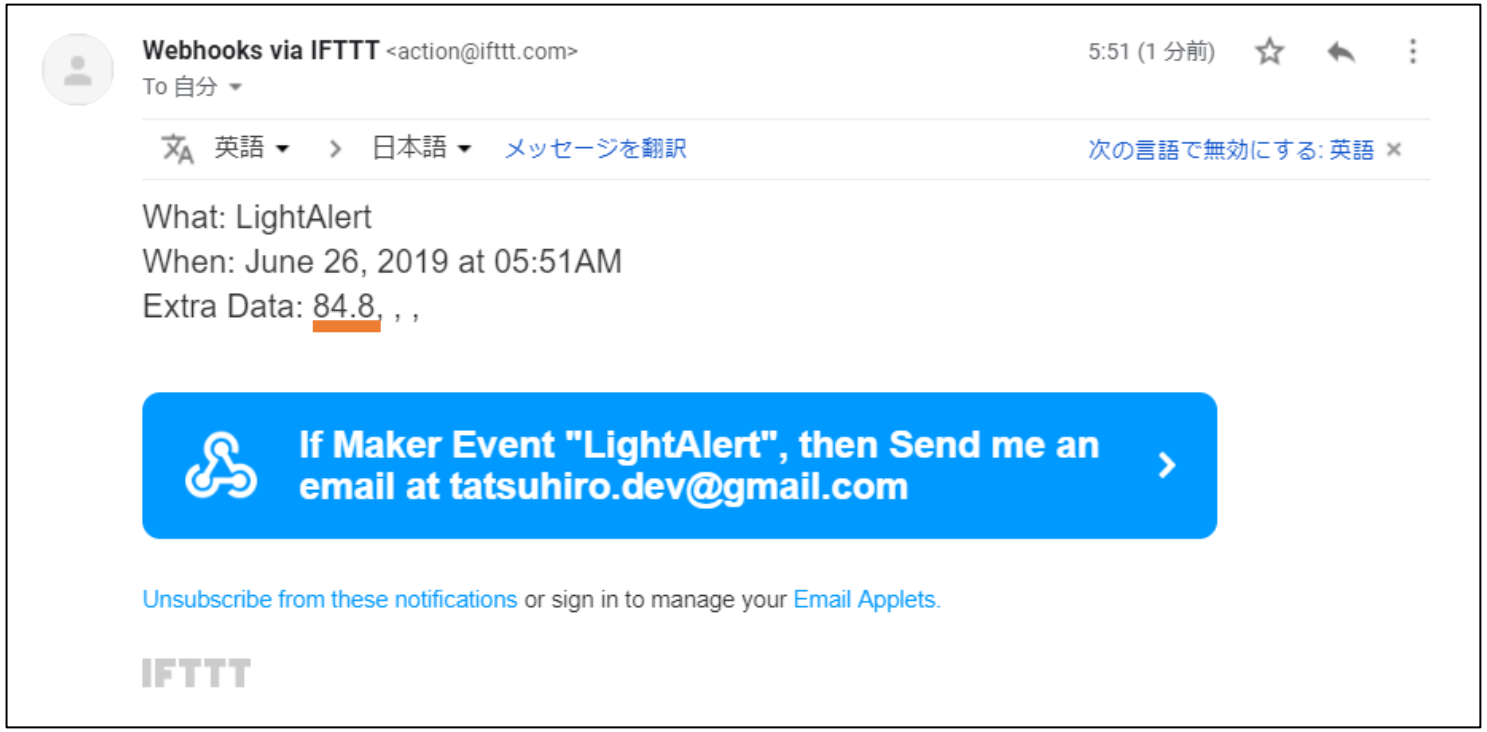



```
import requests

# IFTTTのWebhookにポストする関数
def fire_ifttt_webhook(eventid):
    payload = {"value1": "これは",
               "value2": "テスト",
               "value3": 123456 }
    url =  "https://maker.ifttt.com/trigger/"  +
eventid + "/with/key/ IFTTT Webhookのキー "
    r = requests.post(url, data=payload)


print ("IFTTTのLightAlertにPostする")
fire_ifttt_webhook("LightAlert")
```

課題：明るさ検出とアラート通知

- 前章で作ったLightMonitoringシステムを改修し、暗くなったらメールアラートでお知らせするシステムを開発しましょう！

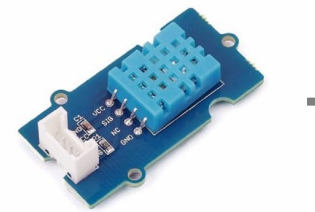




まとめ

今回実現したIoTシステム

モニタリングによる通知と可視化

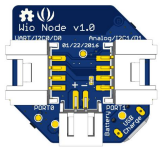


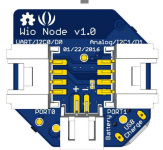
温湿度センサ

光センサ

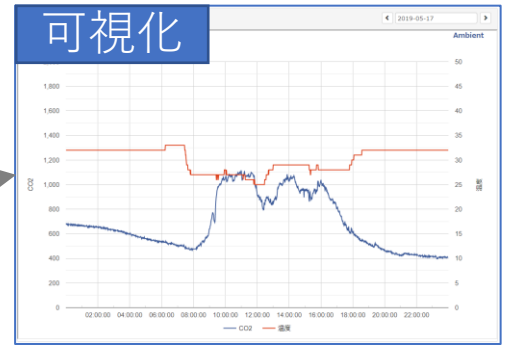
LED

ブザー



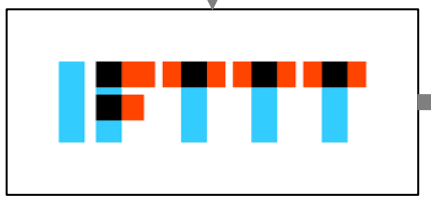


インターネット

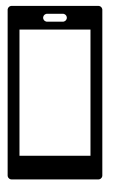


可視化

Ambient

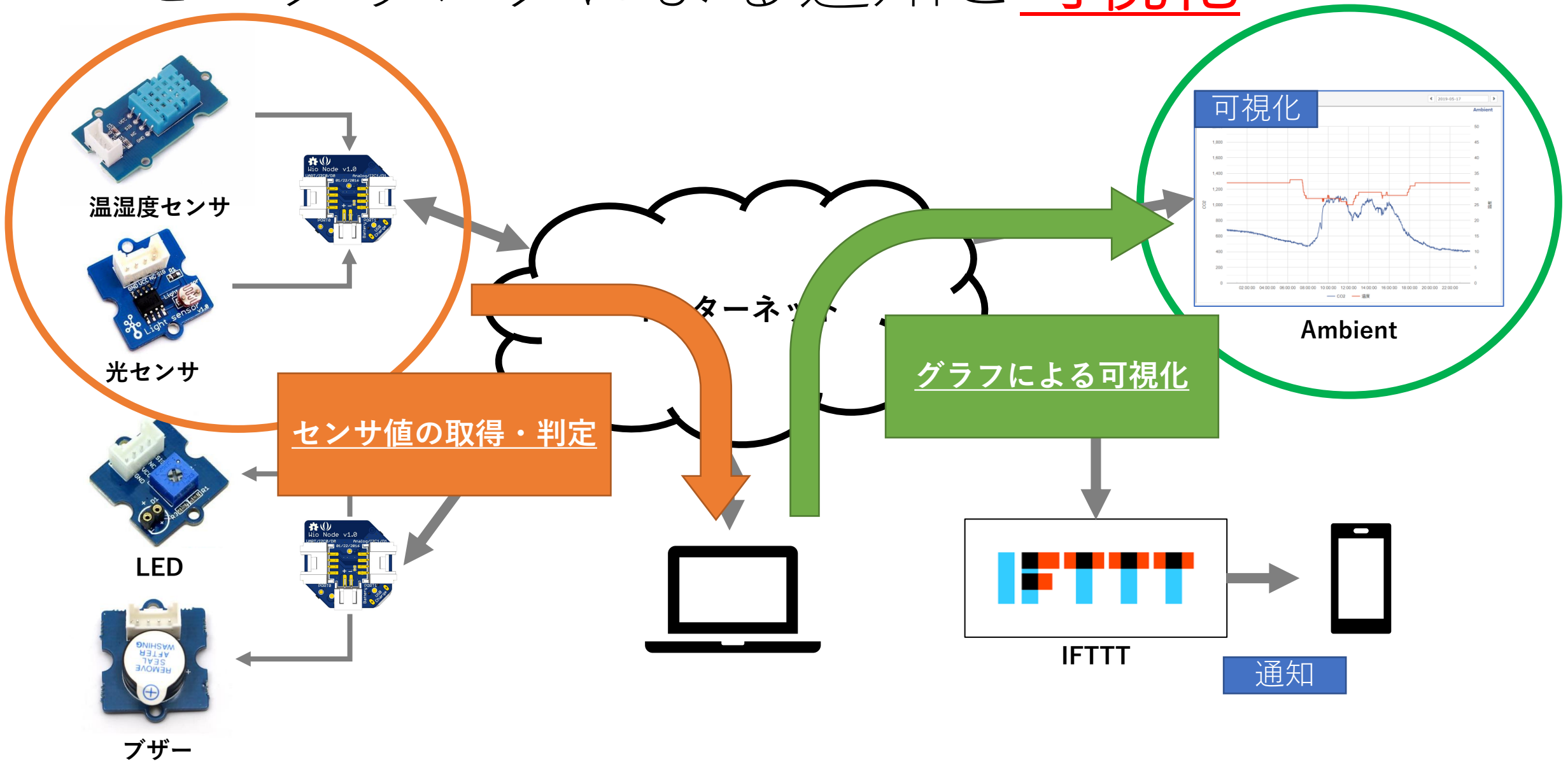


IFTTT

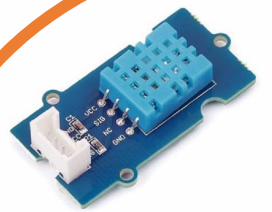


通知

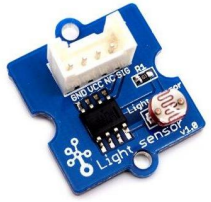
モニタリングによる通知と可視化



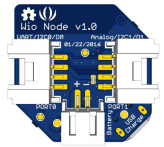
モニタリングによる通知と可視化



温湿度センサ

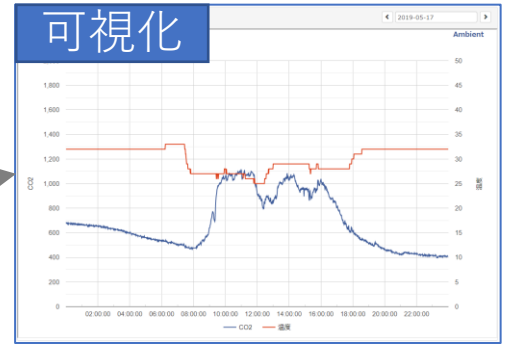


光センサ



センサ値の取得・判定

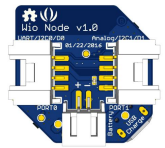
ターネット



Ambient

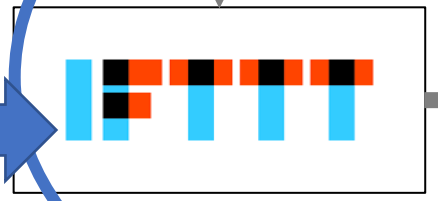
情報端末への通知

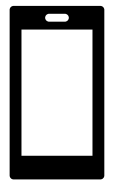
LED



ブザー

空間への通知





通知

実現したこと

- **グラフによる可視化**
  - センサにより対象の空間の温度や湿度の変化を可視化する

- **情報端末への通知**
  - センサ値が一定の値を超えた時，メールにより端末を持つ人へアラートを通知する
- **空間への通知**
  - センサ値が一定の値を超えた時，LEDの点灯やブザーの鳴動により，対象の空間にいる人へアラートを通知する

まとめ

- IoTはまだまだ市場的にも新しい
  - PoCやプロトタイピングを行って，構想したシステムが有用であるのか？検証する事が大事

- お手軽に実現する為のSaaS/iPaaSの活用
  - 目的を達成する為には，便利なサービス・ツールを活用していく
  - 実運用まで想定するのであれば，SaaS/iPaaSでも有償サービスに切り替えていくべき
    - IFTTT → PowerAutomateほか
    - Ambinent → Azure IoT Centralほか

[オマケ] APIサーバを自前で建てる

- WioNodeを使う場合，Seeed社のAPIサーバを介する事になる
- このAPIサーバを自分で用意する事も可能
  - メリット
    - より短い周期でのサンプリングが可能になる
    - APIサーバを自身らでコントロールできる
      - 意図しないサーバダウンを防ぐ
- Server Development Guideを参照
  - https://github.com/Seeed-Studio/Wio_Link/wiki/Server%20Deployment%20Guide
  - 「2. Deploy Directly into Host Filesystem」がオススメ方法
  - Ubuntu18.04がターゲット

参考文献および画像等の引用元

- 画像引用元
  - Seeed (https://www.seeedstudio.com)
  - Seeed Wiki (http://wiki.seeedstudio.com)
  - いらすとや (https://www.irasutoya.com)