



# さくらの IoT Platform を使ってみよう



<http://www.sakura.ad.jp/>

DAY

2016/11/27

COMPANY

さくらインターネット株式会社

DEPARTMENT

コミュニティマネージャー

NAME

法林 浩之



 Facebook 法林 浩之

 Twitter @hourin

### どんな人？

- ・日本UNIXユーザ会 幹事
- ・フリーランスエンジニア
- ・さくらインターネット コミュニティマネージャー
- ・くわしくは「法林浩之」で検索

### さくらでやっていること

- ・当社主催イベントの運営
- ・社外イベント対応(協賛/出展/登壇/取材など)
- ・10月末～12月上旬にかけて20連戦敢行中



- さくらのIoT Platformの概要
  - 開発経緯
  - 主な機能/システム構成/パートナー連携
  - 事例
  - β版の販売について
- 実際に使ってみる
  - マイコンおよびプログラムの構築
  - さくらのIoT Platformの設定
  - Webサービスとの連携



大阪本社



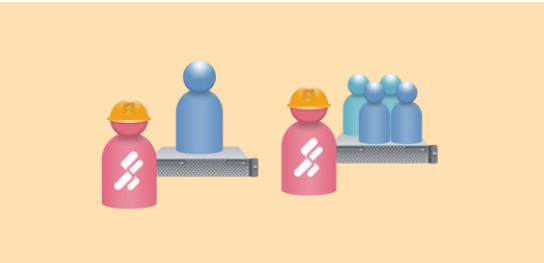
東京支社



<b>商号</b>	さくらインターネット株式会社 (SAKURA Internet Inc.)
<b>代表取締役</b>	田中 邦裕
<b>設立</b>	1999年8月17日 (サービス開始: 1996年12月23日)
<b>資本金</b>	8億9,530万円
<b>事業内容</b>	インターネットでのサーバの設置およびその管理業務 電気通信事業法に基づく電気通信事業 マルチメディアの企画ならびに製作・販売 インターネットに関するコンサルティング
<b>従業員数</b>	339名 (2016年3月末)
<b>所属団体</b>	特定非営利活動法人日本データセンター協会 (JDCC) 社団法人コンピュータソフトウェア協会(CSAJ) 社団法人日本ネットワークインフォメーションセンター (JPNIC) 社団法人インターネットプロバイダー協会 (JAIPA) グリーン・グリッド (The Green Grid) IPv6普及・高度化推進協議会 社団法人電子情報技術産業協会 グリーンIT推進協議会 ASP・SaaSインダストリ・コンソーシアム ホスティングビジネス研究会

データセンターにまつわるサービスのすべてを提供

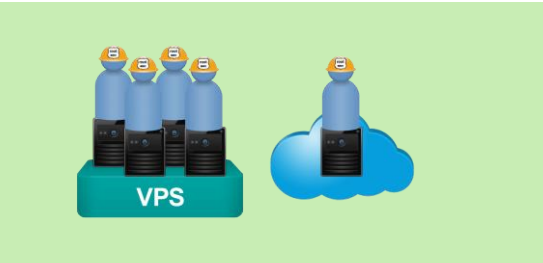
レンタルサーバ



さくらのレンタルサーバ  
さくらのマネージドサーバ

1台のサーバを複数の契約者で共有して利用するサービス

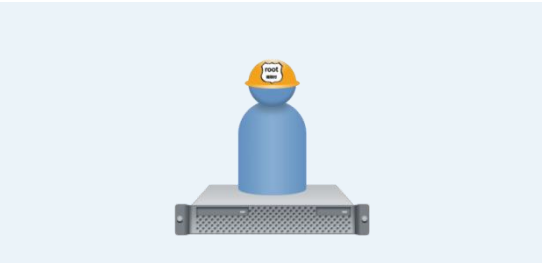
仮想サーバ



さくらのVPS  
さくらのクラウド

1台のサーバを仮想的に分割し、分割された領域を占有できるサービス

専用サーバ



さくらの専用サーバ

顧客が物理サーバ1台を丸ごと占有するサービス

ハウジング



ハウジング  
リモートハウジング

顧客が所有する機器類を設置するスペースと回線、電源などを貸与するサービス




**さくらのIoT Platform<sup>β</sup>** **NEW**

さくらのサービス上で稼働



# さくらのIoT Platform<sup>β</sup>

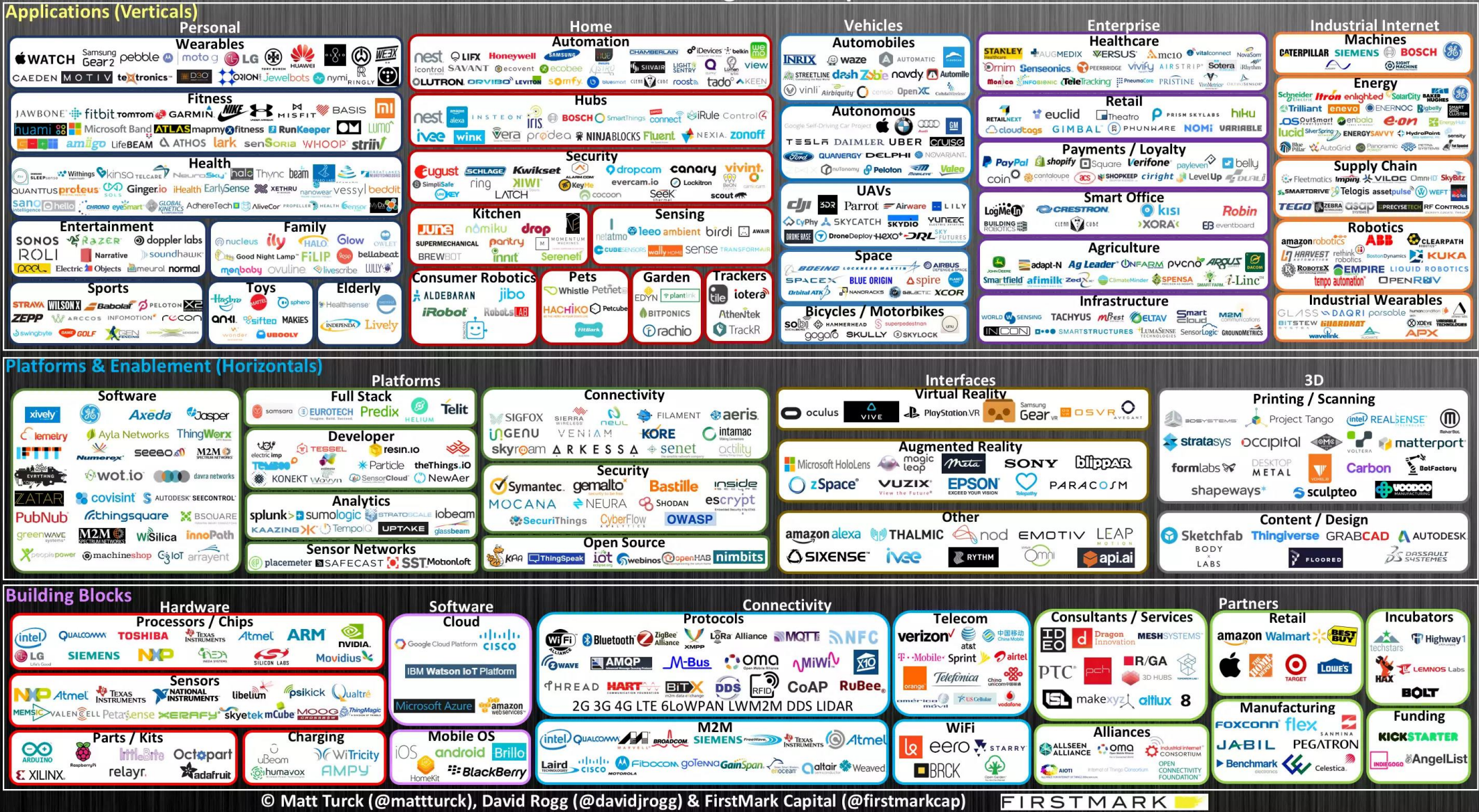
これまで気付けなかった「モノ・コト」の  
相関性や関係性を見出し、それを世界で  
シェアできるプラットフォームを目指す。



「モノがつぶやけばいいのに…」  
という会話がきっかけ

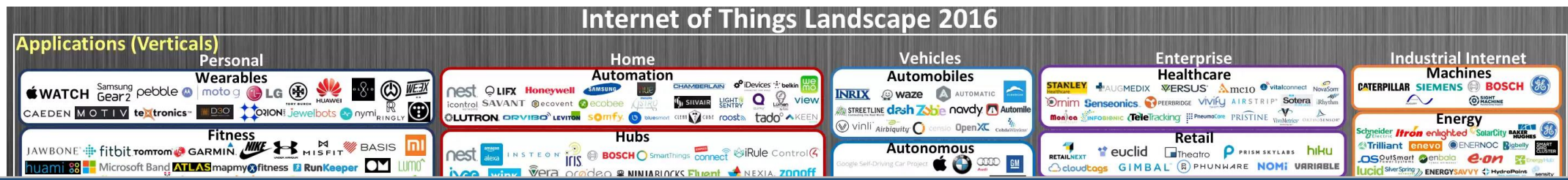
"IoT"とは何か？

## Internet of Things Landscape 2016



© Matt Turck (@mattturck), David Rogg (@davidjrogg) & FirstMark Capital (@firstmarkcap) **FIRSTMARK**

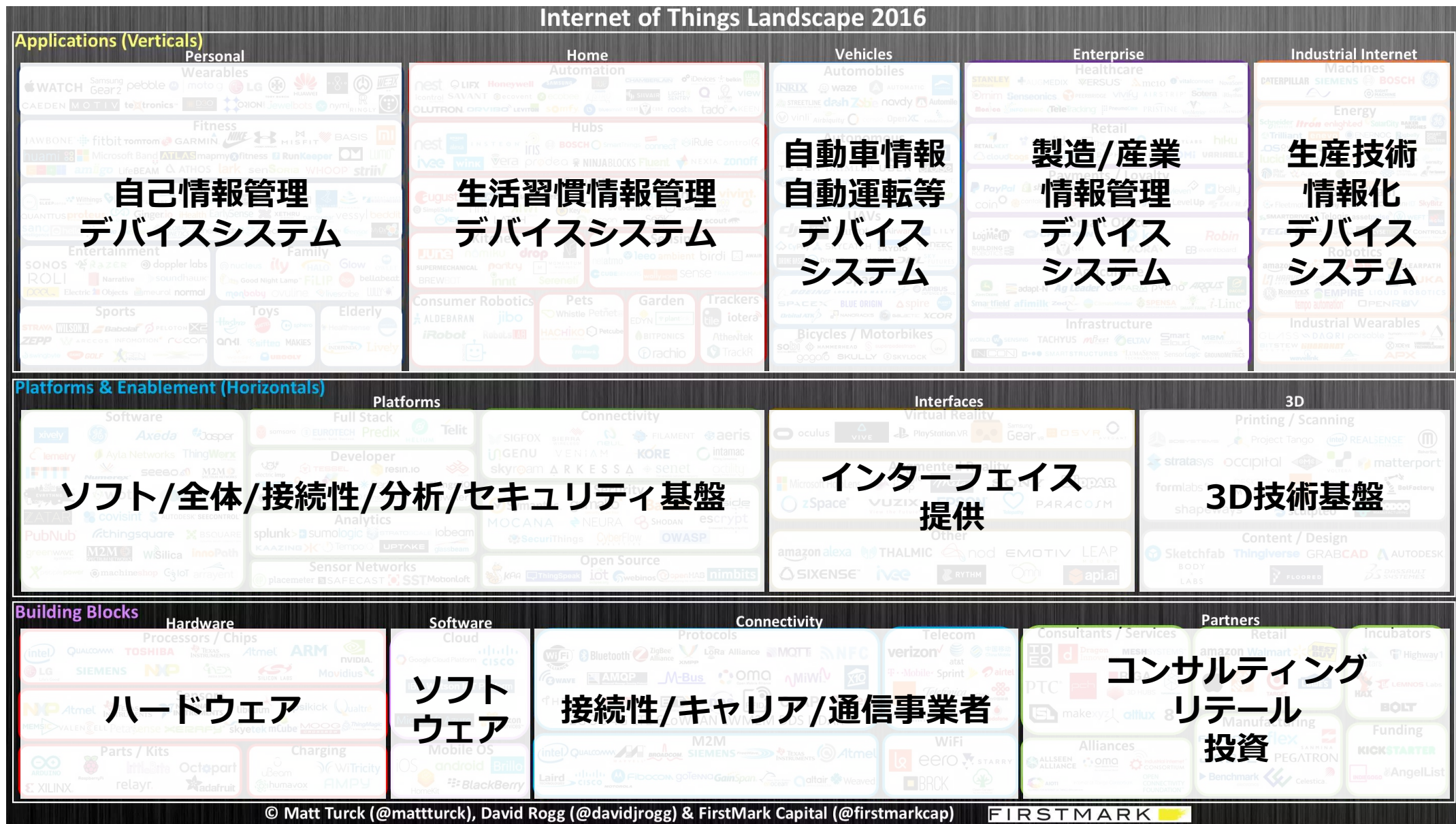




"インターネット"と同じで広すぎる範囲



© Matt Turck (@mattturck), David Rogg (@davidjrogg) & FirstMark Capital (@firstmarkcap) FIRSTMARK



Internet of Things Landscape 2016

Applications (Verticals) Personal Home Vehicles Enterprise Industrial Internet

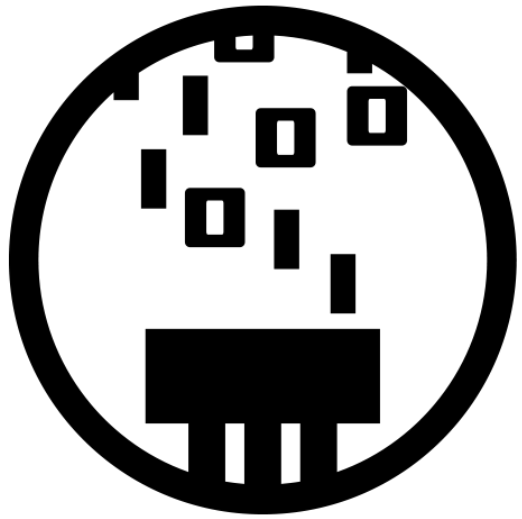
全部やる！ "垂直統合基盤"

Platforms & Enablement (Horizontals) Platforms Interfaces 3D

汎用的な仕組みを提供 "水平統合基盤"

Building Blocks Hardware Software Connectivity Partners

各種構成要素を提供 "要素技術"



さくらのIoT

**Platform**<sup>*β*</sup>



通信モジュール  
LTE閉域網  
データ保存/連携サービス

統合型プラットフォーム

他のIoTプラットフォームとの違い



AlphaGo

# Ponanza

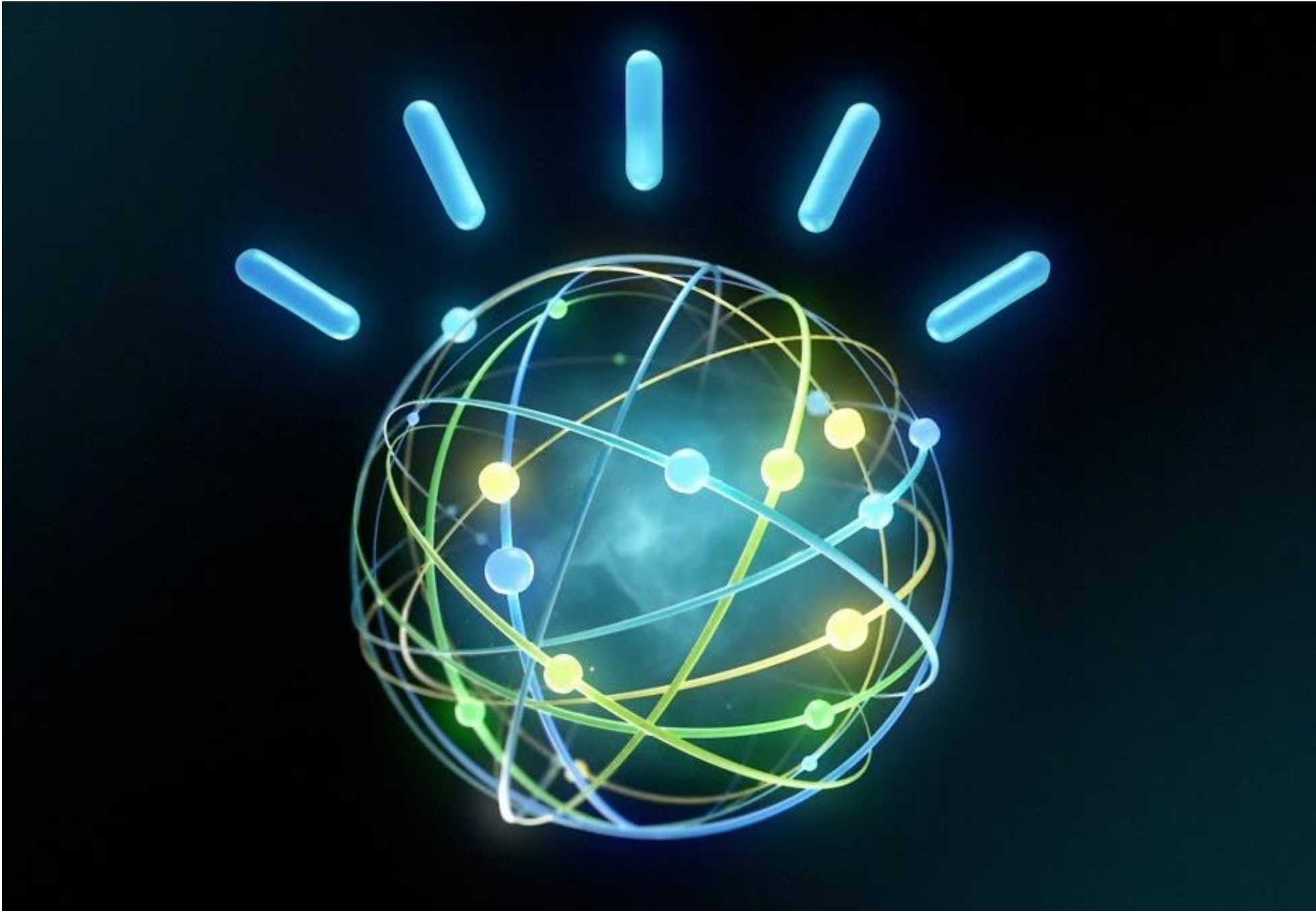


※さくらのサーバで機械学習しています  
<http://ascii.jp/elem/000/001/171/1171630/>

AKI

PONANZA





特化型AI（囲碁/将棋）  
コグニティブコンピューティング



データさえあれば  
何でも分析できます！！

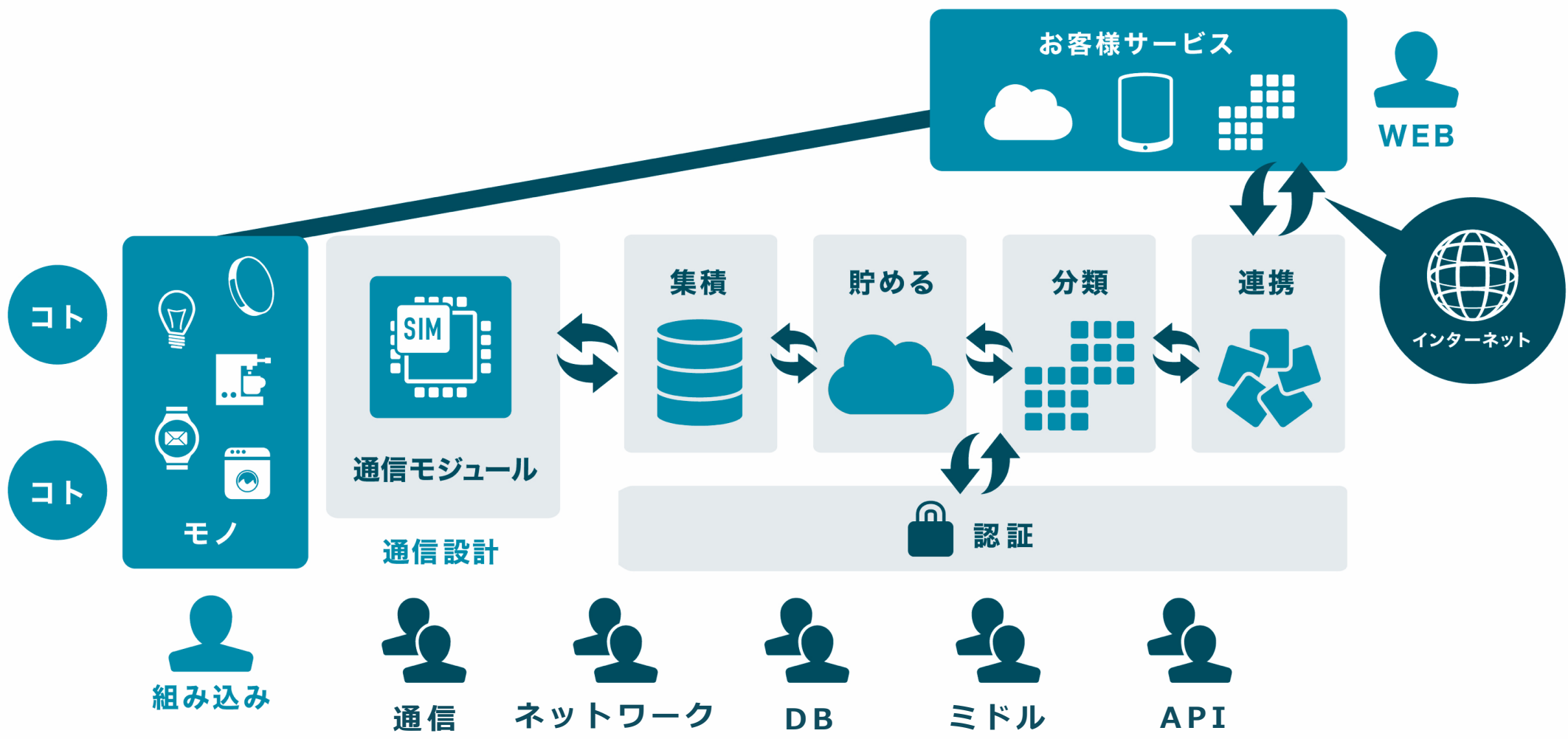
どうやって  
データ

集めるの？

そこに、さくら

さくらのIoT Platform<sup>β</sup>

概要

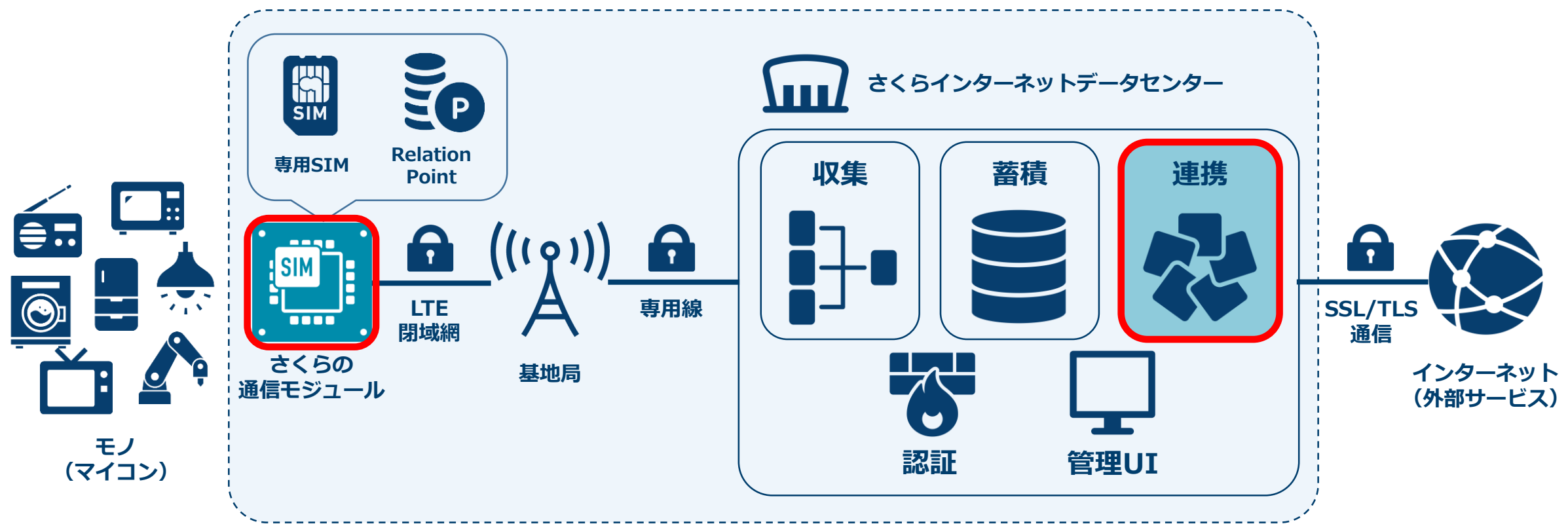


ネットワークとデータをやり取りしたいだけなのに...  
やらなければならないことが多い



既存の事業領域/スキルセットの大幅な変更なく  
モノ/サービスづくり、連携に注力可能

## さくらのIoT Platform<sup>β</sup> ご提供範囲



モノに組み込む「さくらの通信モジュール」から  
データを取り出す「連携サービス」までを統合して提供








	さくらのIoT Platform <sup>β</sup>	他のIoTプラットフォーム
企画・アイデア		
モノ（製造）		
センサー		
データの送受信手段		
安全な通信経路		
プラットフォーム (集める/貯める/分ける)		
管理UI		
連携API		
サービス (Web/AP/分析)		

さくらのIoT Platform<sup>β</sup>側には、プラットフォーム層からサービス層まで、緑色の縦棒が連続して表示されています。

他のIoTプラットフォーム側には、プラットフォーム層に緑色の丸が1つ、管理UI層に緑色の丸が1つ、連携API層に緑色の丸が1つ表示されています。また、データ送受信手段層には破線の丸が1つ表示されています。

黄色い吹き出しのコメント:

- さくらのIoT Platform<sup>β</sup>側: とりあえずアプリやスマホで！ 設定は利用者側で！
- 他のIoTプラットフォーム側: 安全性の担保は開発者の負担に…

	さくらのIoT Platform <sup>β</sup>	他のIoTプラットフォーム
企画・アイデア	<p><b>「データを迎えに行く」</b> という発想</p> <ul style="list-style-type: none"> <li>モノからのアウトプットだけでなくモノへのインプットも可能</li> <li>モノに組み込めば、電源を入れるだけで利用可能</li> <li>利用者に接続の知識や現地の有線/無線LAN環境も不要</li> </ul>	
モノ (製造)		
センサー		
データの送受信手段		
安全な通信経路		
プラットフォーム (集める/貯める/分ける)		
管理UI		
連携API		
サービス (Web/AP/分析)		

さくらのIoT Platform<sup>β</sup>

機能詳細

1. IoTデバイス、サービスの**“開発工数削減”**が可能
2. **“SDカードほぼ2枚分”**に収まるコンパクトサイズ
3. 閉域網を利用した**“Secure & Safety”**なネットワーク設計
4. プラットフォームサービスだから**“設計や運用は考慮不要”**
5. **“時刻提供機能”**でマイコンに現在時刻を提供
6. **“ファイル配信機能”**でマイコン側のアップデートも実現

# IoTデバイス、サービスの“開発工数削減”が可能

要開発項目

- アプリケーション
- コマンドI/F実装
- 上位プロトコル実装
- TCP/IPスタック
- モデムコマンド制御
- UART制御
- 省電力制御

従来の通信手法  
I<sup>2</sup>C/SPI

さくらの  
通信モジュール



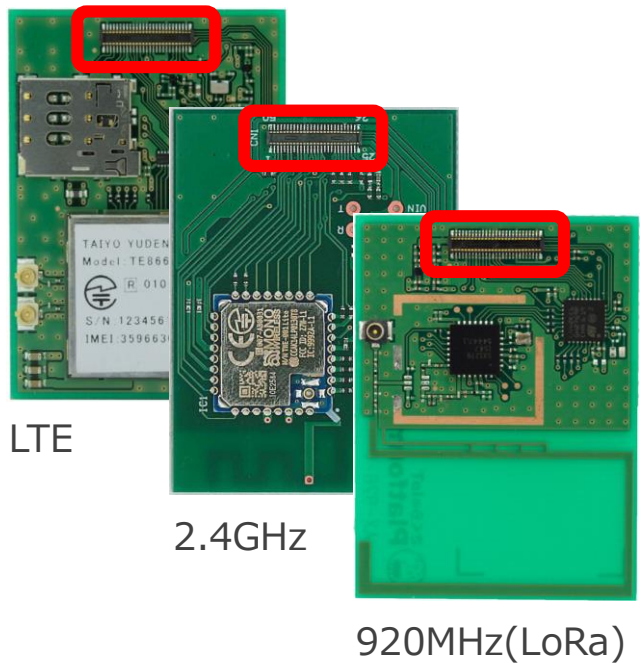
削減工数

複数の無線規格に対応

「作らなければならないもの」より「作りたいもの」に注力が可能

# IoTデバイス、サービスの“開発工数削減”が可能

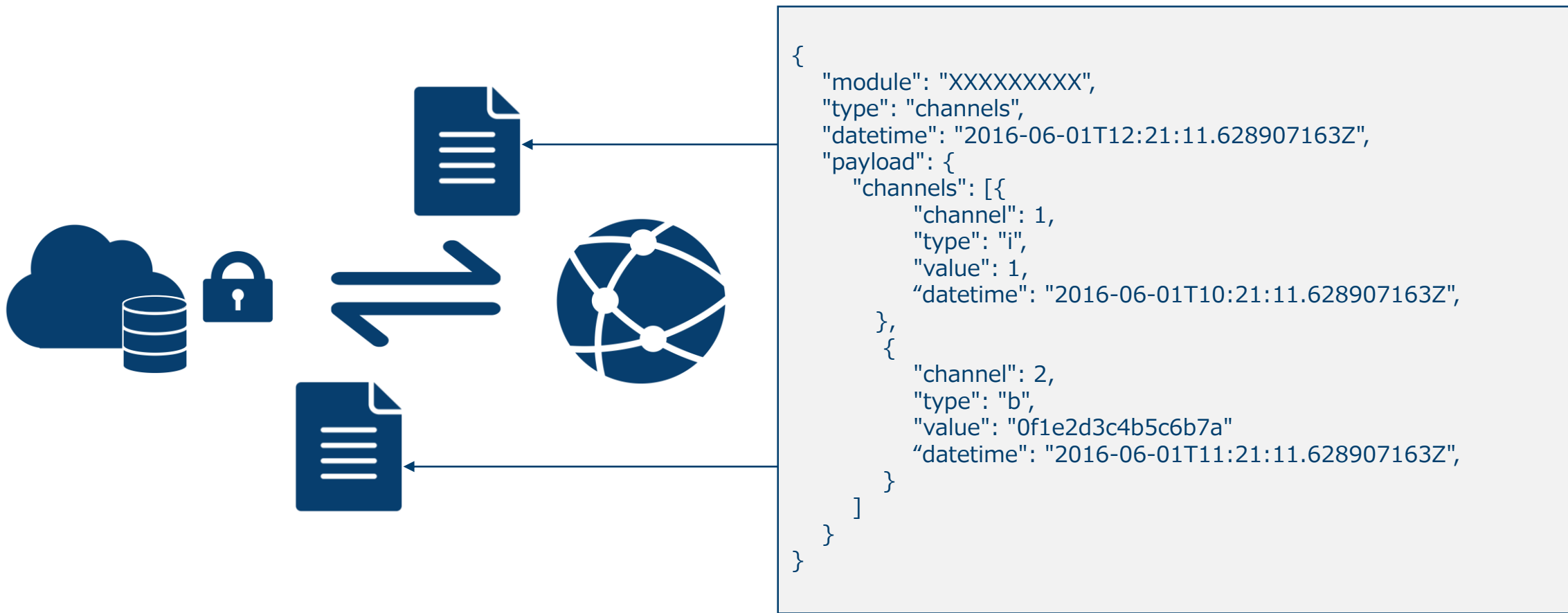
量産性に配慮した  
“基板間コネクタ”を採用



方式	GW	特徴	通信可能レンジ	伝送速度	消費電力
LTE	不要	単独 使用可	キャリア網内 どこでも	速い	大きい
2.4GHz帯	必要	短距離 大容量	数百メートル (最大1km程度)	速い	小さい
920MHz帯	必要	長距離 小容量	数キロメートル (最大10km程度)	遅い	小さい

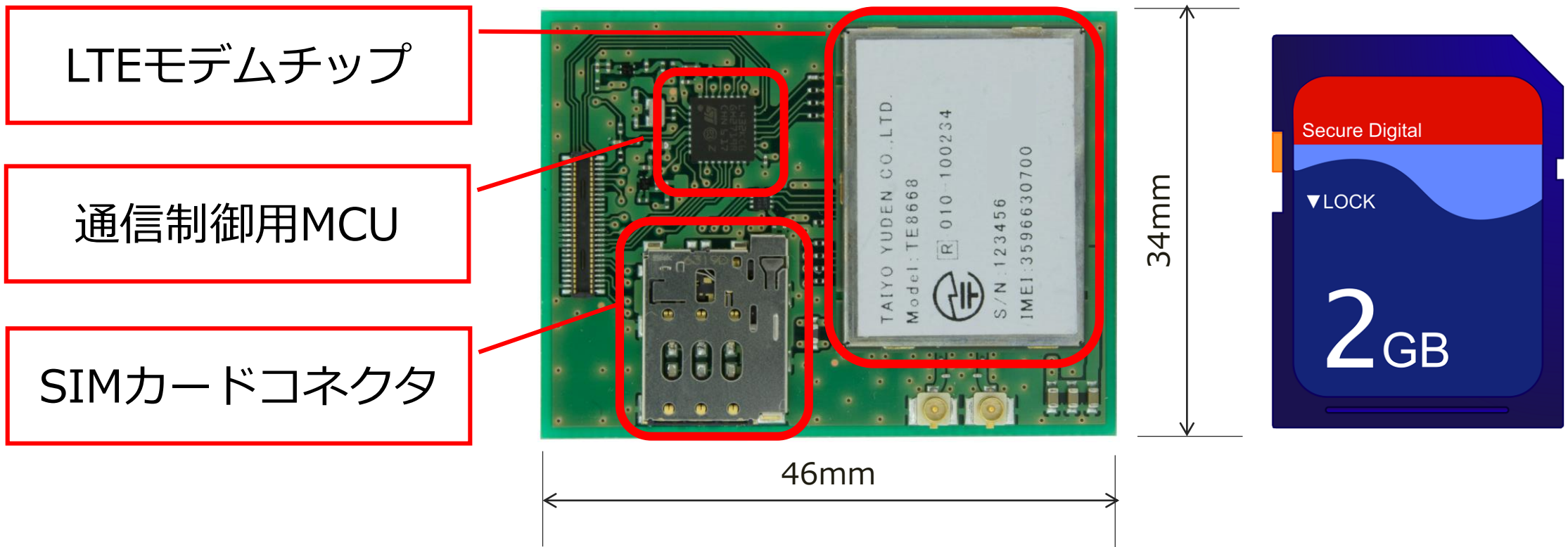
共通I/Fおよび寸法のため複数規格のモジュールの対応が容易

# IoTデバイス、サービスの“開発工数削減”が可能



データ活用は暗号化された経路から扱いやすいJSONフォーマットで

# “SDカードほぼ2枚分”に収まるコンパクトサイズ



“さくらの通信モジュール”にモノ側の通信に必要なすべてを凝縮



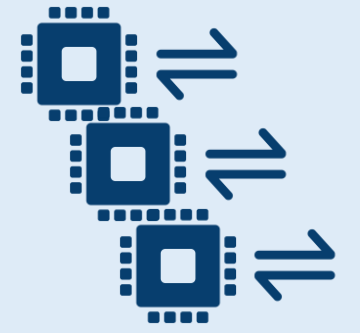
# 閉域網を利用した"Secure & Safety"なネットワーク設計



通信モジュール+SIMカードをセットで提供することで安全性を確保

# プラットフォームサービスだから“設計や運用は考慮不要”

サービスの  
“設計”



データの収集



データの蓄積



データの連携



セキュリティ

サービスの  
“運用”



ラージスケール対応



アップデート

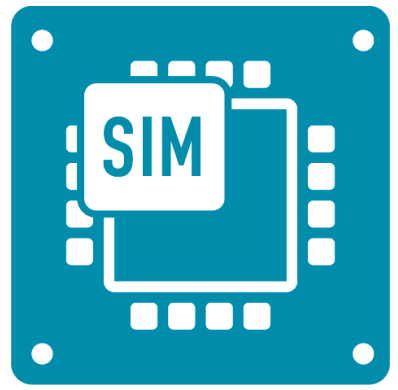


障害切り分け、復旧

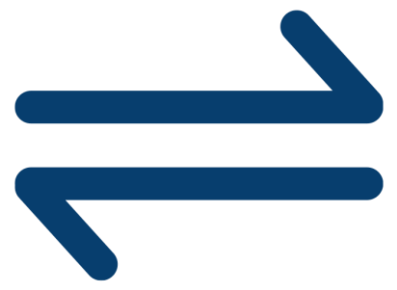
IoTサービスに不可欠な“設計”や“運用”はさくらインターネットが対応

# “時刻提供機能”でマイコンに現在時刻を提供

マイコンへの適用



時刻情報の要求



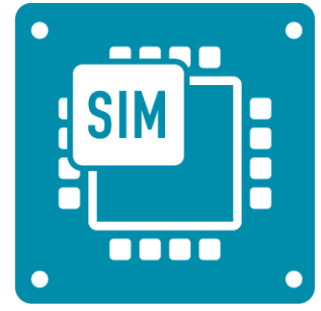
正確な時刻の提供

マイコン側でのログ記録等が求められるデバイスやサービスに

※マイコン側で時刻情報要求の制御が必要です

# “ファイル配信機能”でマイコン側のアップデートも実現

マイコンへの適用



さくらの  
通信モジュール

モジュールへの配信



さくらの  
IoT Platform

ファイルのアップロード



管理者

## ソフトウェア的な問題はアップデートで対応可能

※マイコン側でのファームウェア書換制御が必要です



オープン

オープンデータとして活用可能にいただくことでデータ保存のコストをゼロにできます。



クローズ

公開したくないデータはユーザのみ閲覧権限を付与いただくことができます。



プライベート

コンプライアンス対策としてデータを専用領域に保存いただくことができます。

データ保護ポリシーに応じて適切な保存先を選択可能

## リアルタイム連携

### <即時性が求められるサービス向け>

通信モジュールからの受信データを即連携先に送信します。  
連携先からの送信データも即通信モジュールに送信します。  
提供は以下を含め、随時他社サービスも追加されます。

Outgoing Webhook/Incoming Webhook/WebSocket/MQTT(予定)



## 保存データの一括取得

### <分析や集計等のバッチ処理向け>

通信モジュールから受信したデータを要求された時に  
要求された期間分まとめて送信します。  
提供はHTTP REST APIにより行われます。



利用用途に応じて適切な連携方式を選択可能

共創パートナー



他社クラウドサービスや自社環境にも用途に応じて連携可能



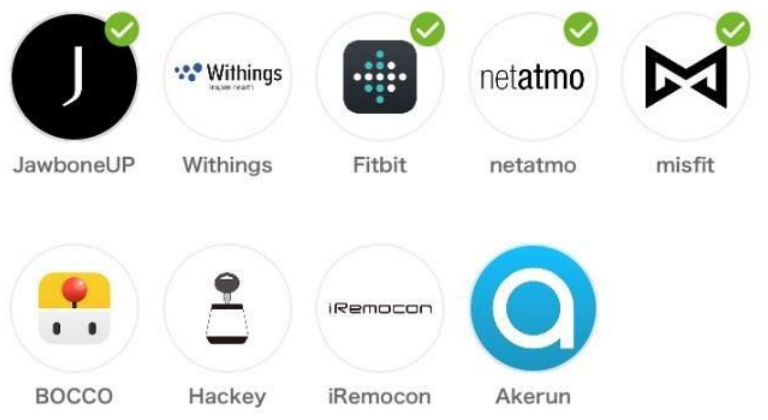


# myThings

インターネットサービスやIoTプロダクトを利用して、  
あなたにあった便利な組み合わせをつくることのできるアプリ

その他 チャンネル一覧

ガジェット



スマート家電



開発者向け



**チャンネルリクエスト**  
あなたの欲しいチャンネルを教えてください

myThingsに  
さくらのIoT Platform  $\beta$  の  
チャンネルが作成される予定

作成

トリガー

防災速報

地震情報が出たら

地域 千葉県  
震度 5弱以上

アクション

アクションを追加してください



作成

# トリガーを設定

例：防災速報  
地震情報が出たら  
熱中症の危険度が高まったら  
津波警報・注意報が発令されたら

トリガー


**防災速報**
×

---

地震情報が出たら

地域 千葉県
 >

震度〇以上だったら 5弱以上

アクション


**さくらのIoTプラットフォーム**
×

---

データを送る

サービス test-webhook

モジュール uf3tJqUZ83Yr
 >

チャンネル 1

型 符号あり32bit整数

値 1



実行タイミング

時間指定

# アクションを設定

## 例：さくらのIoTプラットフォームの1チャンネルに1という値を送信



トリガーごとに異なるデータを送信可能

例：熱中症の危険度が高まったら さくらのIoTプラットフォームに 2チャンネルに2という値を送信

トリガー

防災速報

津波注意報・警報が出たら

地域 千葉県

アクション

さくらのIoTプラットフォーム

データを送る

サービス test-webhook

モジュール uf3tJqUZ83Yr

チャンネル 3

型 符号あり32bit整数

値 3

実行タイミング

時間指定

時間指定しない場合は15分間隔で実行します

例：津波警報・注意報が出たら  
 さくらのIoTプラットフォームの  
 3チャンネルに3という値を送信

さくらのIoTプラットフォームを  
 トリガーに使うことも可能

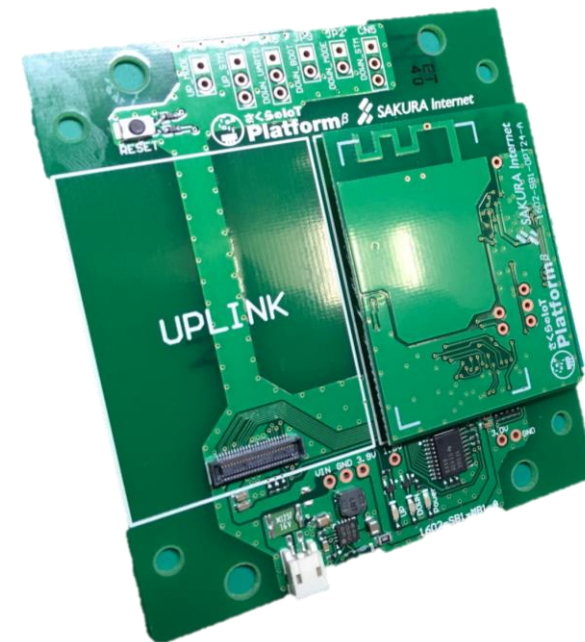
# 事例



私たちは  
IoTの“T”を知らなすぎる

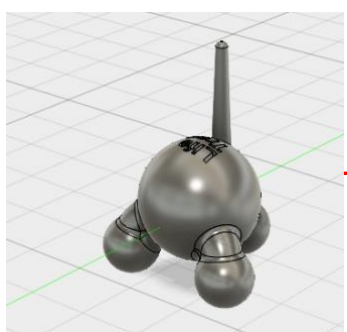


「なぜそれが必要なのか」をパートナーと協力し追求、  
量産化対応、新機能、GW方式モジュール開発につなげ、  
継続している。

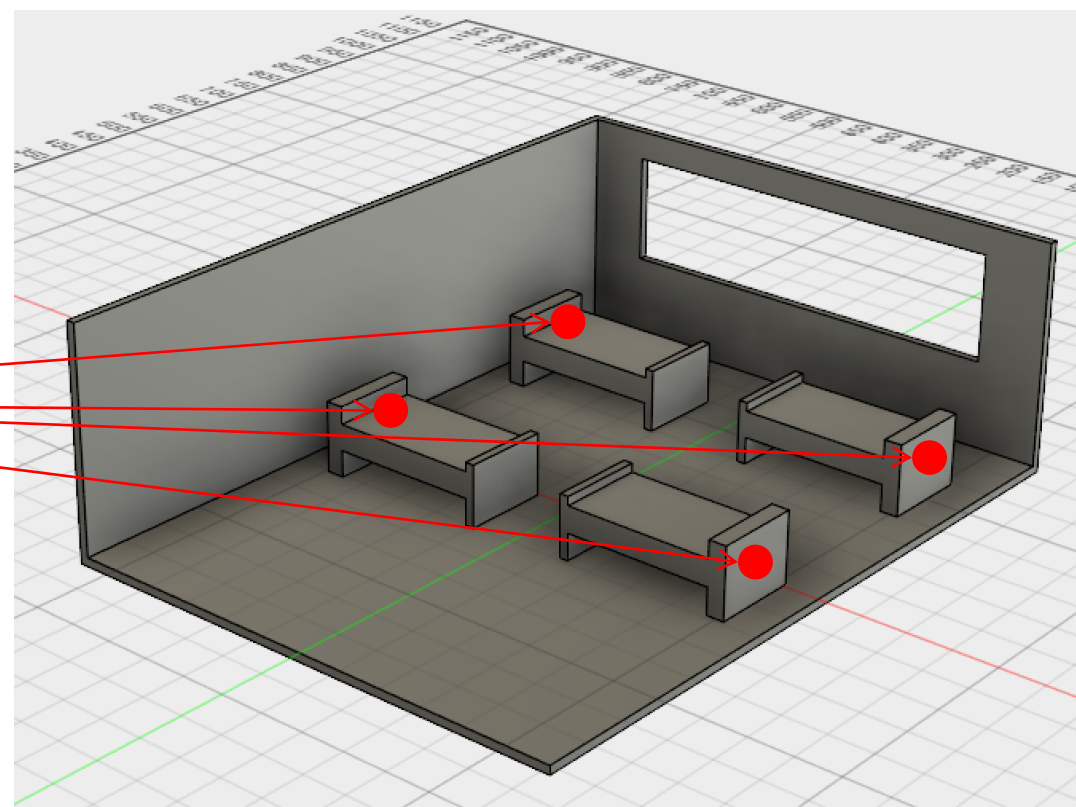


# primesap

## 病院での実証実験



Intel Edision  
加速度/温度/湿度/照度センサー  
さくらの通信モジュール



## ベッド毎に入眠後の身体の動揺を計測

- 気圧や温度などの環境要因と安眠度の相関性について理解が進む
- 睡眠に適切な環境とさらにその個人差を把握することで環境改善に貢献



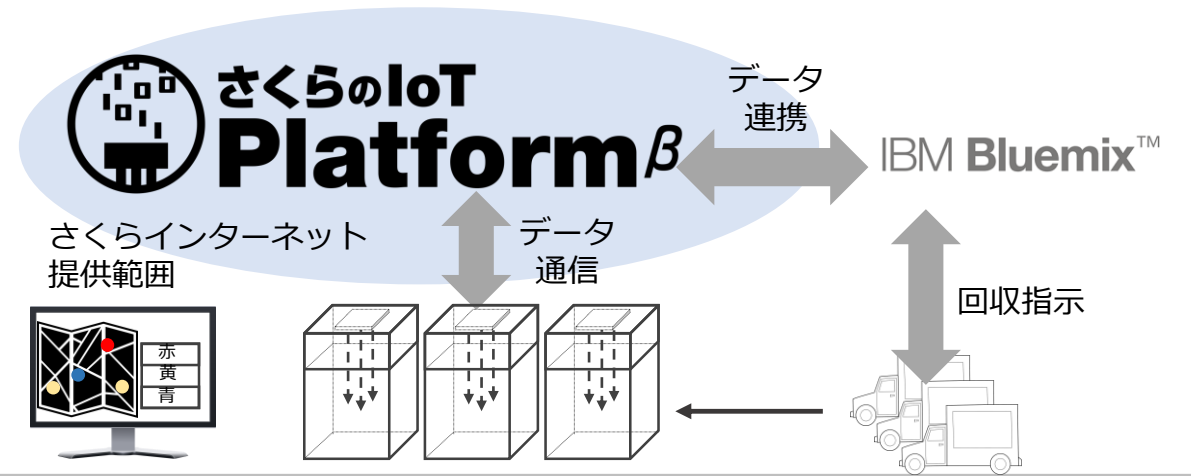
シェアリングエコノミーを加速させる  
スマートロックを中心とした  
プラットフォームカンパニー



# ハウステンボスとのPoC



※ハウステンボスのIoTへの取り組み  
ハウステンボス社では、IoTやAI、ロボティクスを活用しユニークな顧客体験の創造を目指しており、さくらインターネットの通信モジュールを活用した実証実験もその一環



■ イメージ

パーク内に設置されたゴミ箱に計測機器を取り付け、さくらのIoTプラットフォームに送付、コグニティブコンピューティングシステムに送付し、解析する。将来的にはAIによって常に最適な回収経路を自動回収車に指示し、収集が自動的に実施される仕組みを目指す。

パーク内ゴミ箱に、集積量を測定できるセンサーを取り付け、自動的に回収する仕組みを構築する実証実験を行うため、計測機器の通信およびデータ連携システムにさくらのIoTプラットフォームを採用

※LTE通信モジュールの他、さくらインターネットで開発中の920MHz (LoRa) および2.4GHzのゲートウェイ型モジュールも活用し、より良い利用方法についても検証を進める

御提供価格

# さくらの通信モジュール

単体方式：SCM-LTE-beta

**定価 9,960円**

100万回プラットフォームとデータ送受信可能なポイント※付き

## RP=RelationPoint

さくらのIoTプラットフォーム利用時に消費されるポイント

※1つデータ(RM)をプラットフォームとやり取りすると1RP消費

※上記のみを1分間に1回行うと約2年間利用することが可能



# さくらの通信モジュール オプション

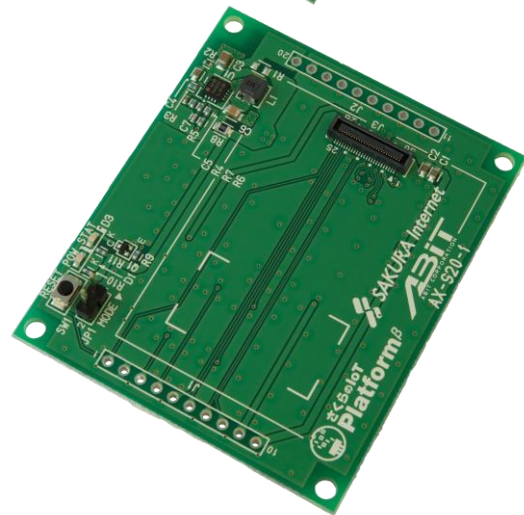


## Arduinoシールドボード

SCO-ARD-01

Arduino マイコンボードを  
利用したプロトタイプ開発に

**定価8,000円**



## ブレイクアウトボード (検証ボード)

SCO-BB-01

その他のマイコンボードを  
利用したプロトタイプ開発に

**定価5,000円**

# さくらのIoT Platform β 期間

β 期間中プラットフォーム利用料無料  
RP消費なし



# $\beta$ 提供記念 キャンペーン

# さくらの通信モジュール及びオプション

半  
額



単体方式  
SCM-LTE-beta

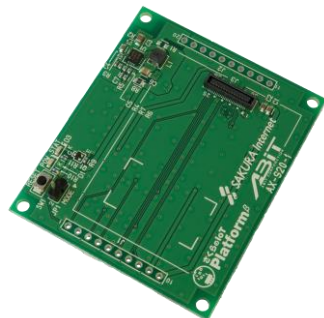
~~9,960円~~ → 4,980円

100万回プラットフォームと  
データ送受信可能なポイント※付き



Arduino  
シールドボード  
SCO-ARD-01

~~8,000円~~ → 4,000円



ブレイクアウトボード  
(検証ボード)  
SCO-BB-01

~~5,000円~~ → 2,500円

## ツイート



田中邦裕  
@kunihirotanaka

さくらのIoTプラットフォームは、SPIやI2Cで専用通信モジュールにつなぐと、LTE回線経由でREST APIからデータを取れるサービスです。SIMが最初から入っていて2年分のLTE料金が入ってます。

[iot.sakura.ad.jp/service/](http://iot.sakura.ad.jp/service/) #procon27

2016/10/09 16:31 場所: 三重 伊勢市

15 リツイート 22 いいね

実際に使ってみよう



# さくらの IoT Platform β版ハンズオン



<http://www.sakura.ad.jp/>

DAY

2016/11/27

COMPANY

さくらインターネット株式会社

DEPARTMENT

コミュニティマネージャー

NAME

法林 浩之

# ハンズオンの資料に沿って 実際に使っている 様子を紹介

マイコンおよびプログラムの構築

温湿度センサ (HDC1000)    マイコン (Arduino Uno)    さくらの通信モジュール



さくらのIoT Platformの設定

さくらのIoT Platform<sup>β</sup>

Webサービス連携 (さくらのクラウド)

node JS  
仮想サーバ

Webサービス連携 (Arukas)

node JS  
コンテナ

## 1. マイコンおよびプログラムの構築

- マイコン (Arduino) による開発環境の準備
- 温湿度センサおよびさくらの通信モジュールの繋ぎ込み
- 試験用プログラムの流し込み

## 2. さくらのIoT Platformの設定

- プロジェクトの作成
- さくらの通信モジュールの登録
- 連携サービスの設定

## 3. Webへのデータ連携 (さくらのクラウド)

- Node-REDサーバの作成
- WebSocketを利用したデータ連携フロー作成
- Zabbixサーバへのデータ連携とグラフ化※ハンズオン対象外



# マイコンおよび プログラムの構築

## マイコンおよびプログラムの構築

温湿度センサ  
(HDC1000)



マイコン  
(Arduino Uno)



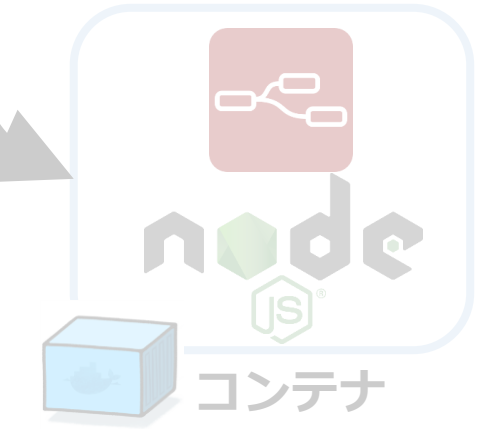
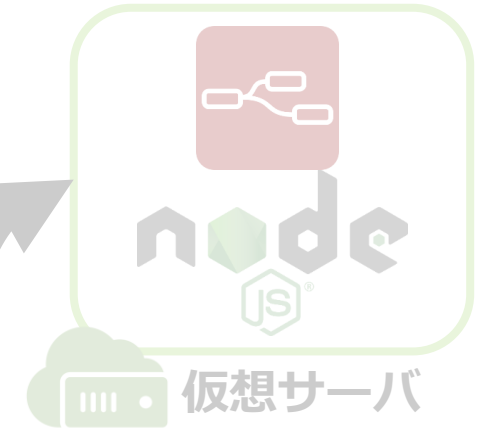
さくらの通信  
モジュール



さくらのIoT Platformの設定



Webサービス連携  
(さくらのクラウド)



Webサービス連携  
(Arukas)

https://www.arduino.cc/en/Main/Software から開発環境（Arduino IDE）を入手します。  
2016/10/18時点での最新版は【1.6.12】となります。

ARDUINO 1.6.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

- Windows Installer
- Windows ZIP file for non admin install
- Mac OS X 10.7 Lion or newer
- Linux 32 bits
- Linux 64 bits
- Linux ARM (experimental)

[Release Notes](#)  
[Source Code](#)  
[Checksums](#)

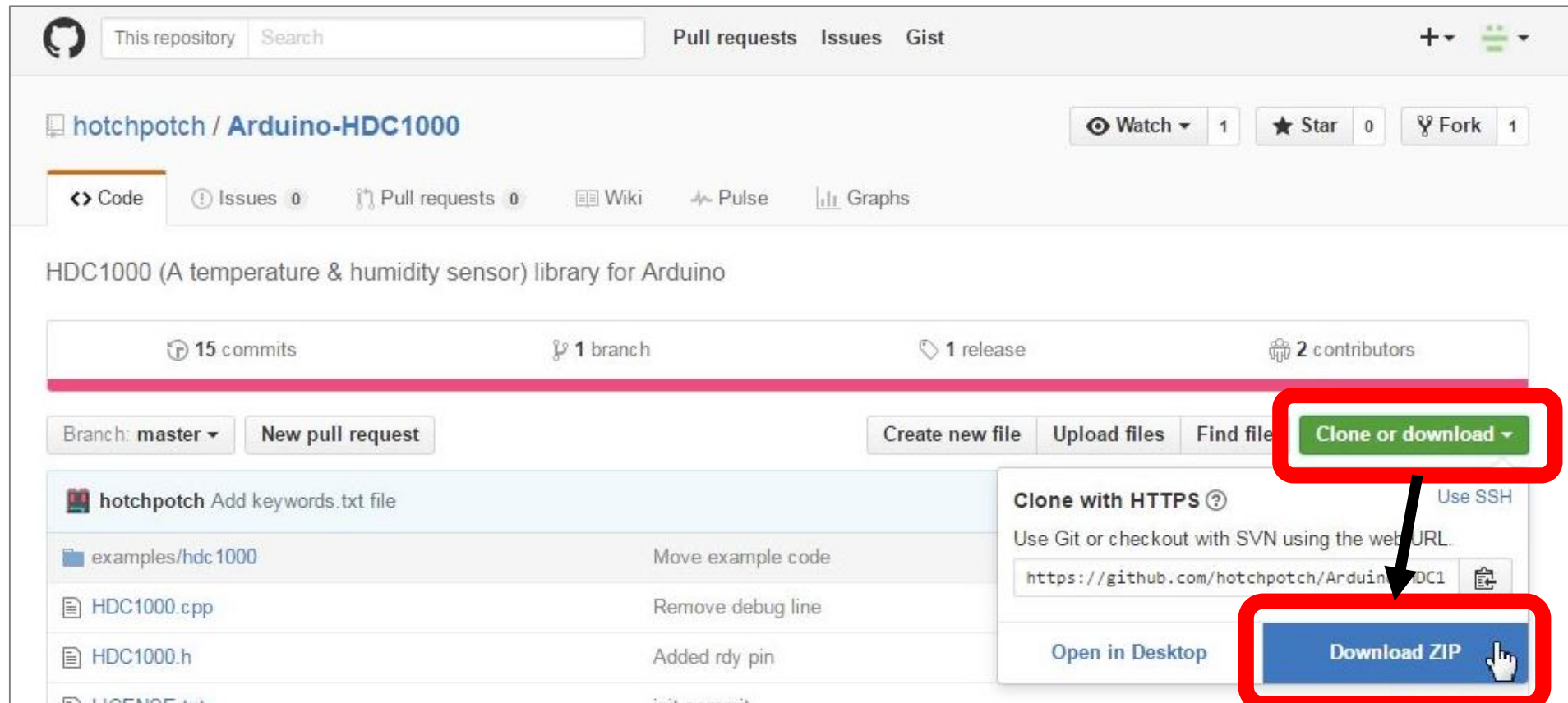
Googleにて「github hdc1000」を検索し、ライブラリを入手します。

<https://github.com/hotchpotch/Arduino-HDC1000>

入手したZipファイルは**解凍**し、「Arduino-HDC1000-master」フォルダごと以下へ格納します。

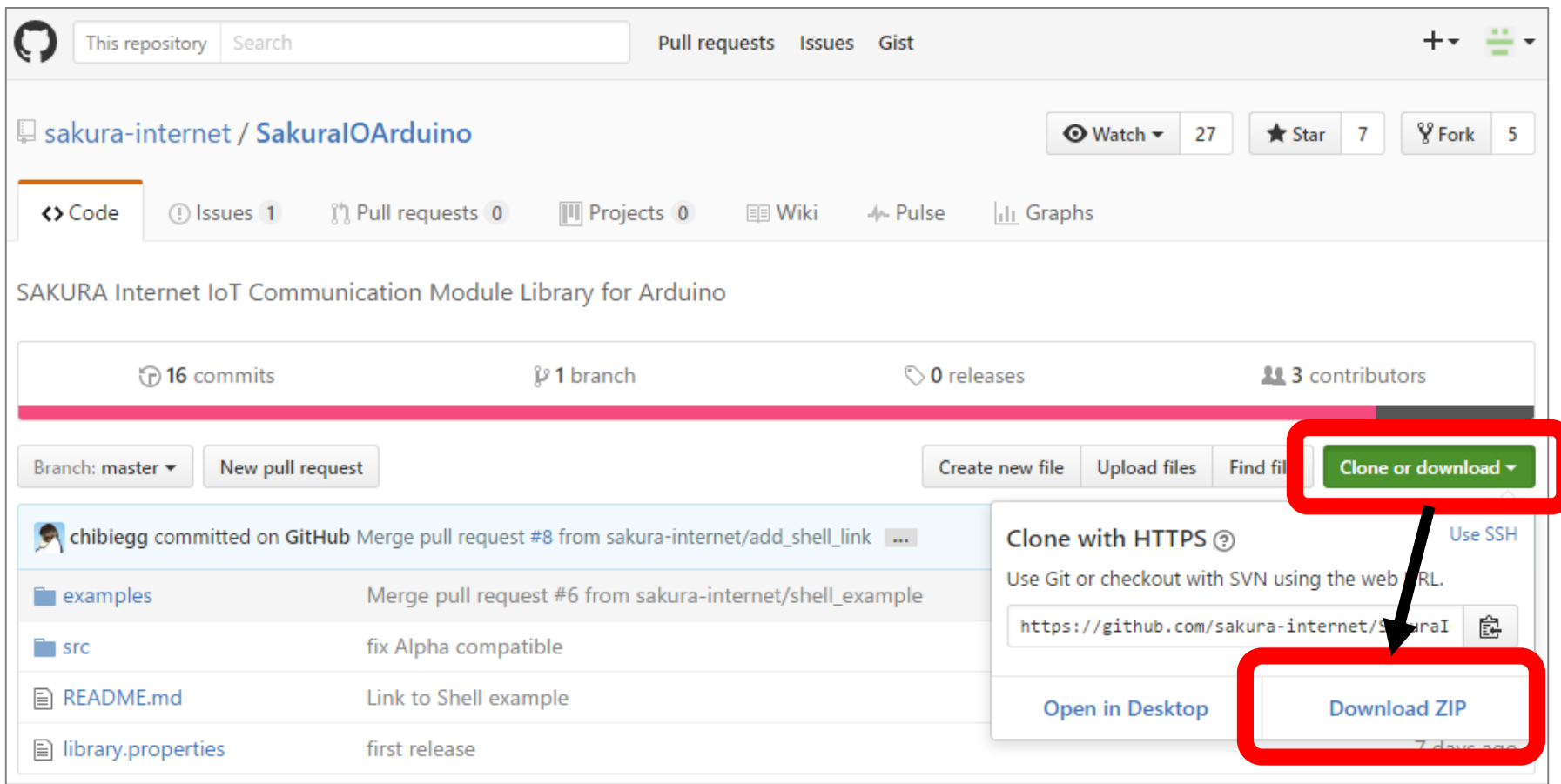
Windows C:¥Users¥《ユーザ名》¥Documents¥Arduino¥libraries

Mac 書類¥Arduino¥libraries



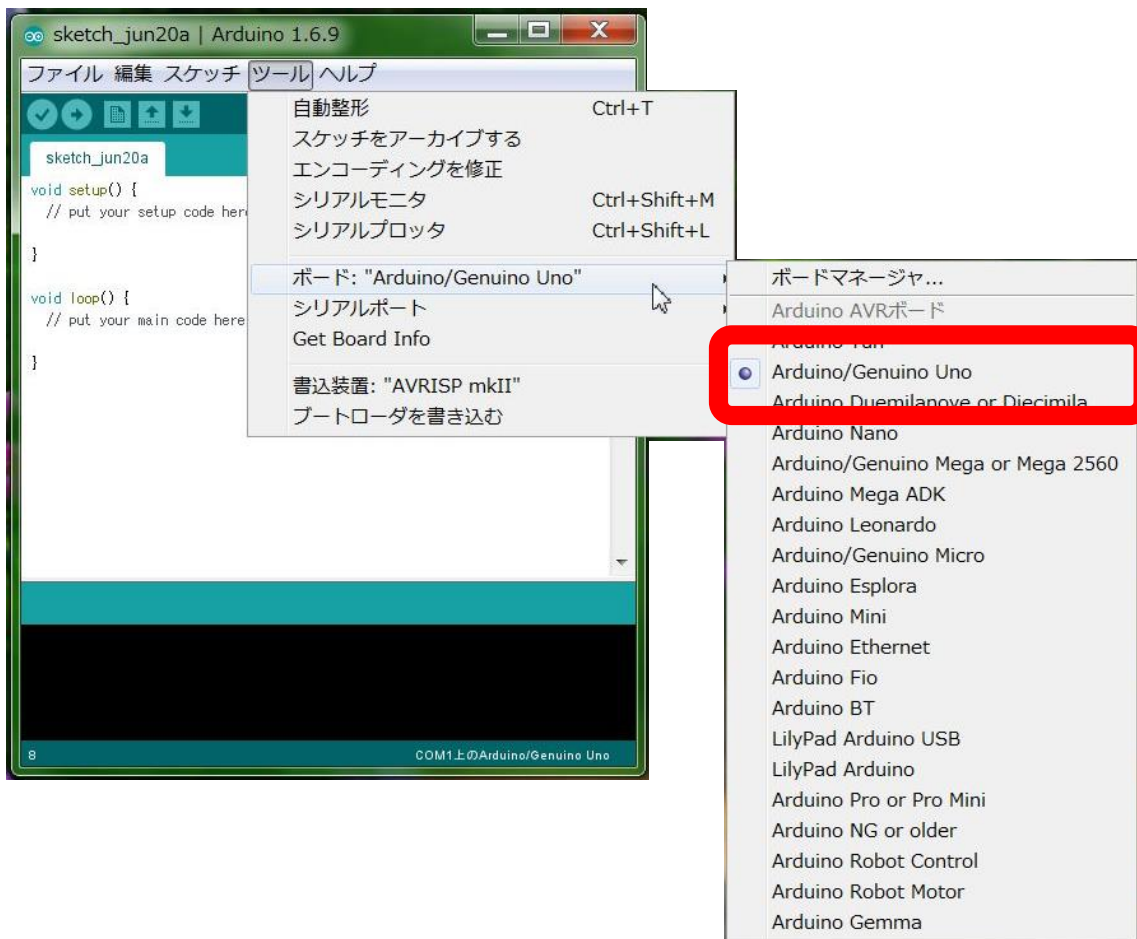
Googleにて「github sakuraioarduino」を検索し、ライブラリを入手します。  
<https://github.com/sakura-internet/SakuraIOArduino>

入手したZipファイルは**解凍**し、「SakuraIOArduino-master」フォルダごと以下へ格納します。  
Windows C:¥Users¥《ユーザ名》¥Documents¥Arduino¥libraries  
Mac 書類¥Arduino¥libraries

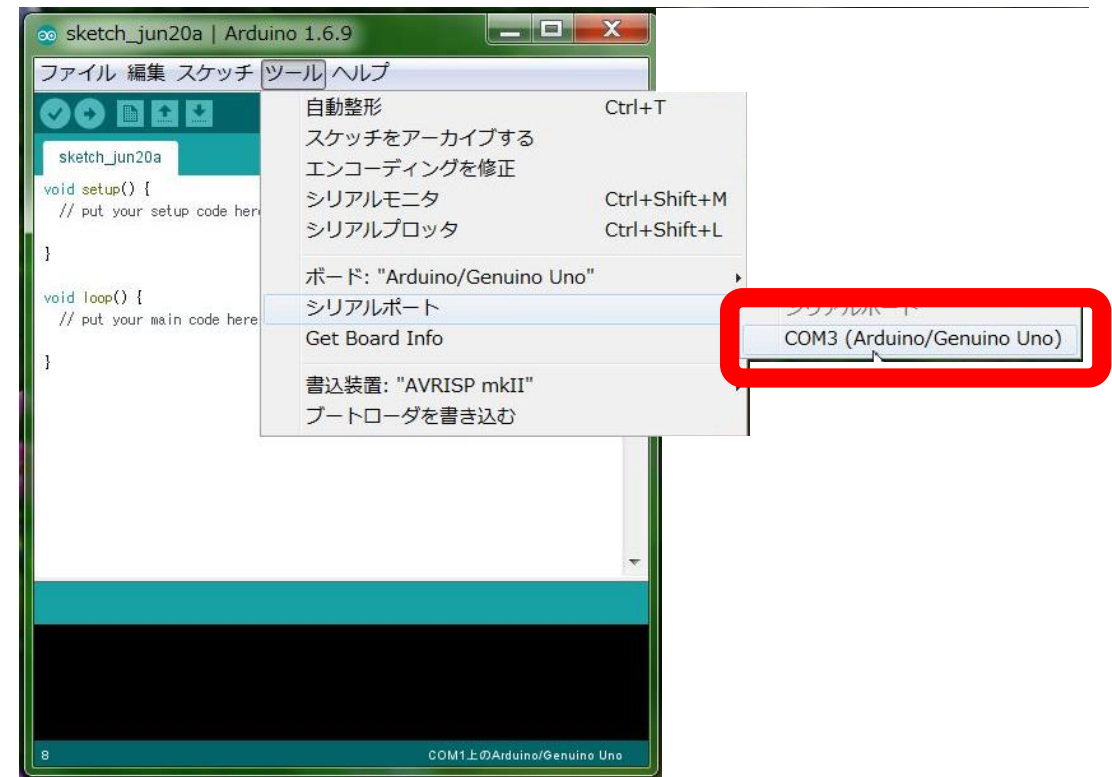


Arduino IDEが起動したら、Arduino本体をPCに接続します。  
ボードは【Arduino/Genuino Uno】、シリアルポートは出てきたCOMポートを選択します。  
Macの場合はシリアルポート内の「〜〜（Arduino/Genuino Uno）」となるものを選択します。

## ボードの選択

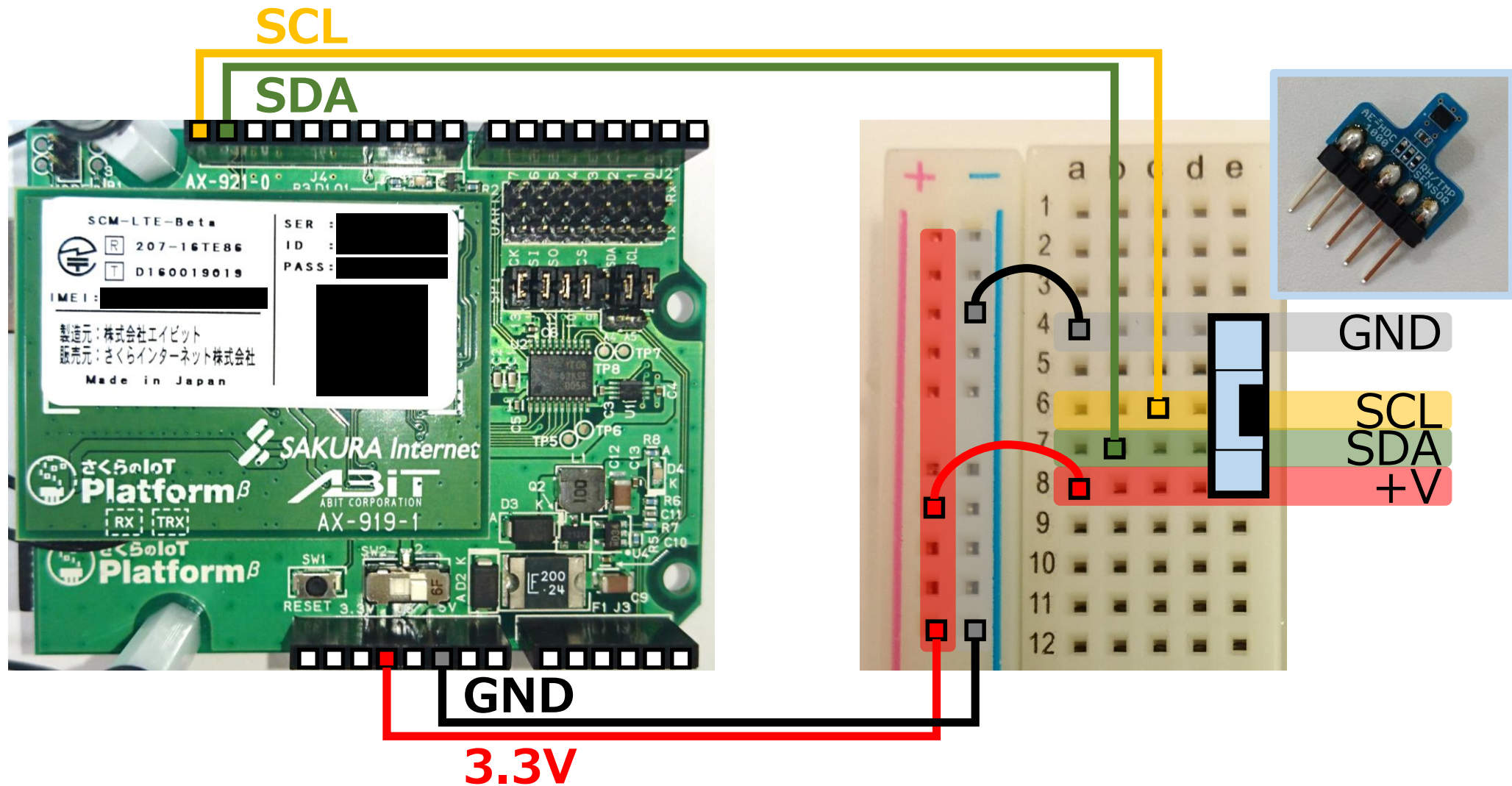


## シリアルポートの選択



## → 温湿度センサの繋ぎ込み

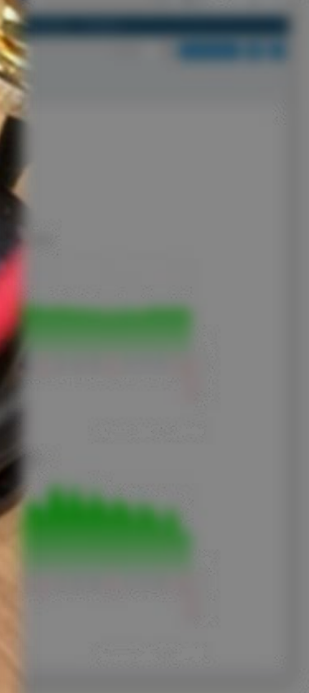
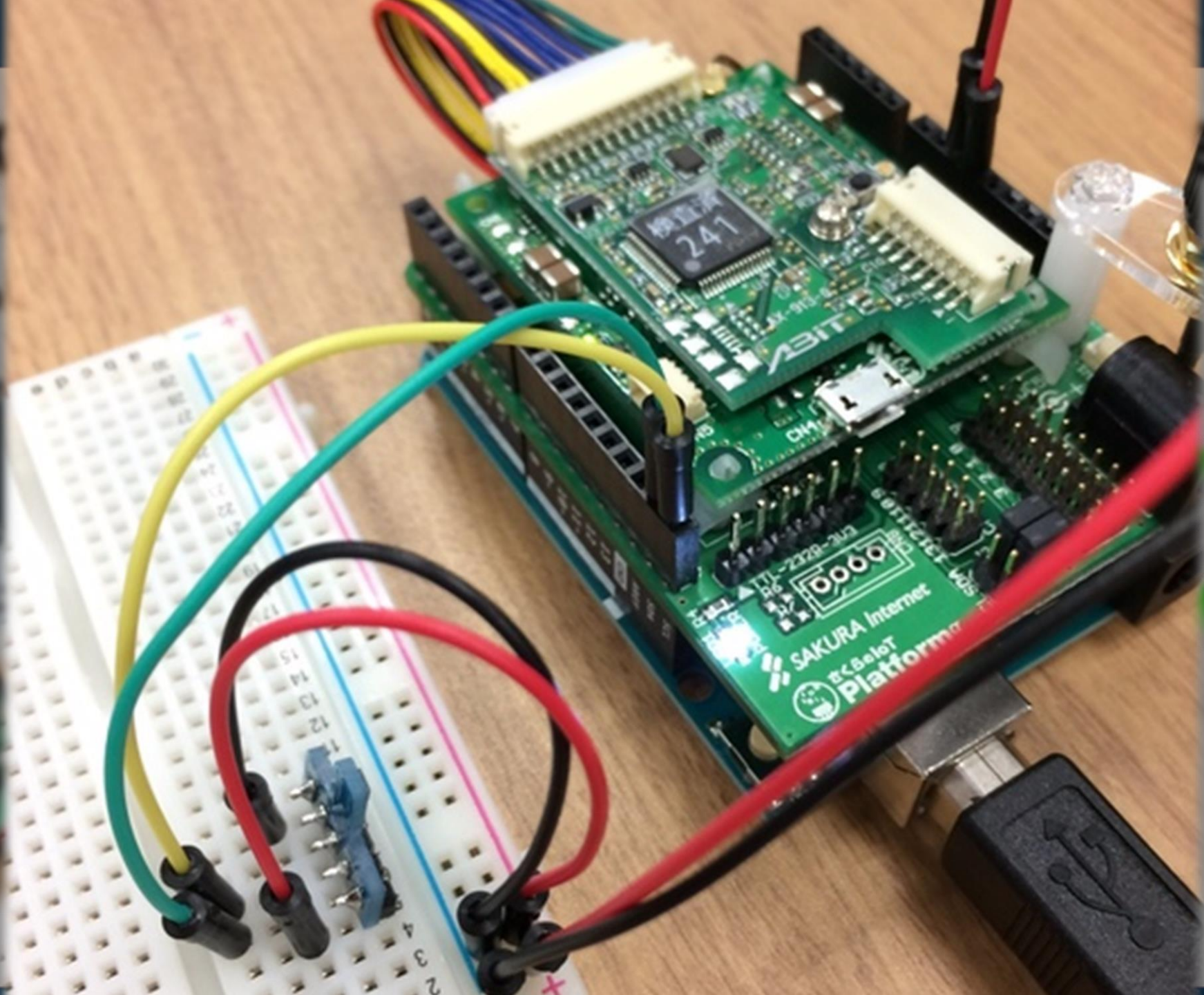
結線を行うため、ArduinoをPCから外します。  
その後、図に従ってジャンパーコードを接続します。





Sak

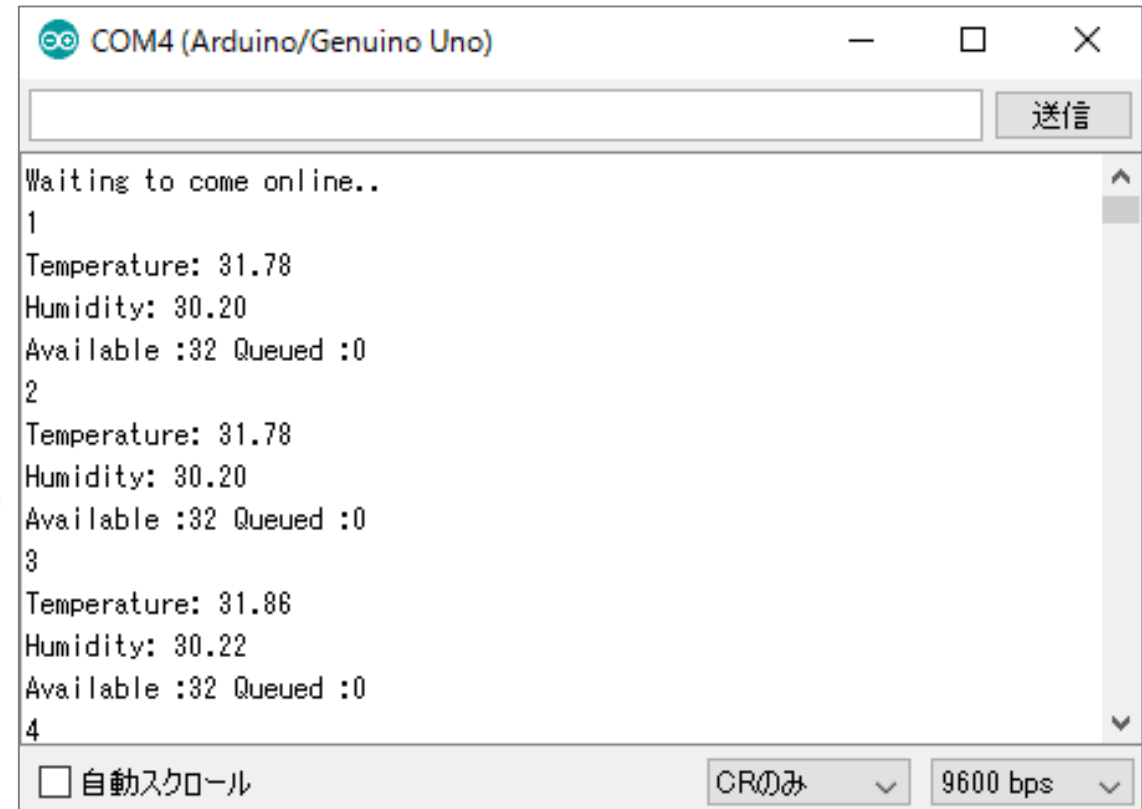
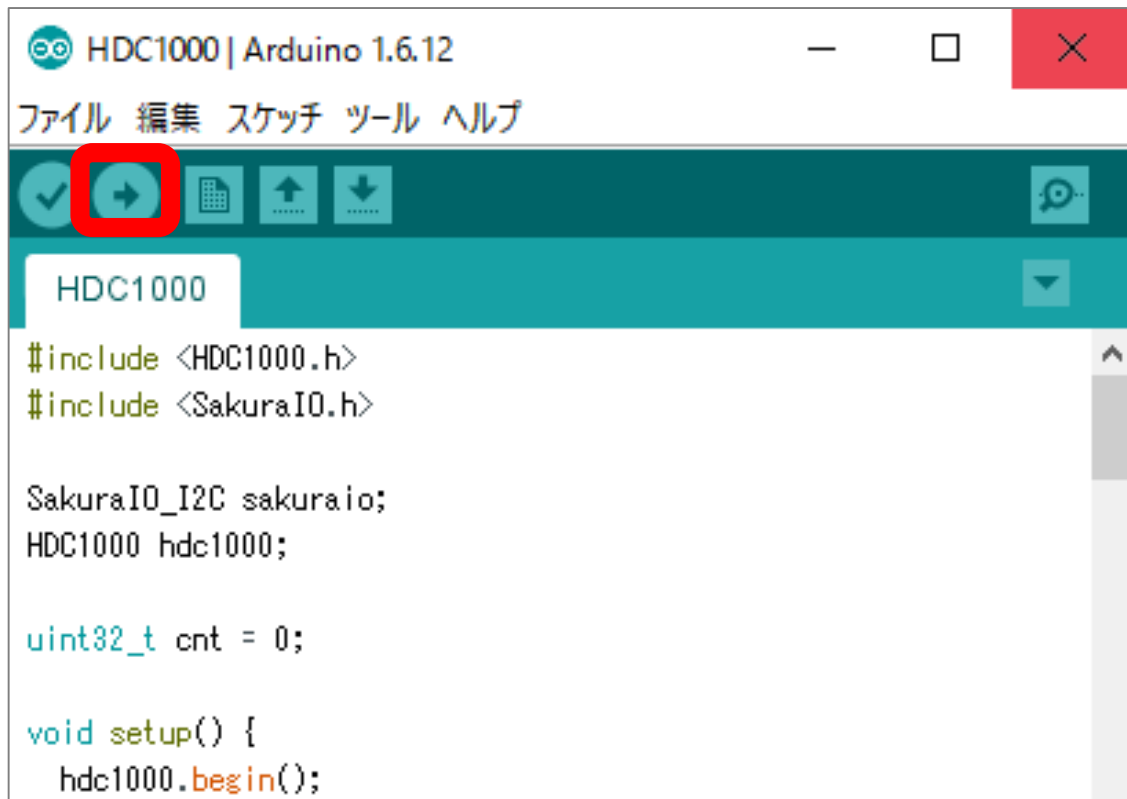
emo





## → 試験用プログラムの流し込み

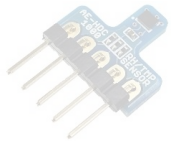
[スケッチの例]→[SakuraIO]→[[HDC1000](#)] → 【→】 ボタンをクリックします。  
[ツール]→[シリアルモニタ]より「Waiting to come online」表記の後、カウント値、Temperature、Humidityに加え、Available（キューイング可能なチャンネル数）と Queued（キューで送信待ちになっているチャンネル数）が表示されることを確認します。



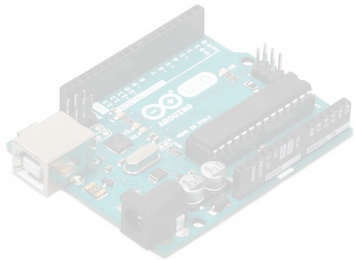
# さくらのIoT Platform の設定

マイコンおよび  
プログラムの構築

温湿度センサ  
(HDC1000)



マイコン  
(Arduino Uno)



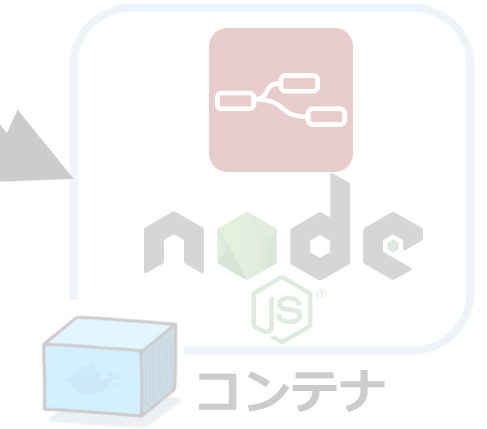
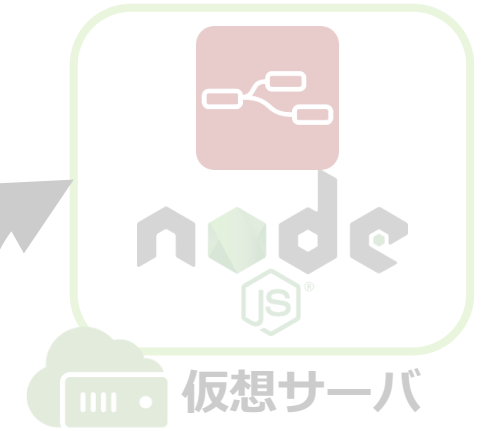
さくらの通信  
モジュール



さくらのIoT  
Platformの設定

さくらのIoT  
Platform<sup>β</sup>

Webサービス連携  
(さくらのクラウド)



Webサービス連携  
(Arukas)

Googleにて「さくらインターネット iot 開発者」を検索し、開発者向けページからコントロールパネル（<https://secure.sakura.ad.jp/iot-alpha/>）にログインします。

## さくらインターネット 会員認証

「会員ID」と「会員メニューのパスワード」をご入力ください

※ 会員メニューのパスワードはお客さまにてお決めいただいたパスワードです

会員ID

例: nnn12345

パスワード

ログイン(認証)

- 会員メニューのパスワードをお持ちでない方・お忘れの方は[こちら](#)
- 会員IDをお忘れの方は[こちら](#)

プロジェクトを作成し、通信モジュールを登録し、連携サービスを設定します。

プロジェクト→

温湿度展示デモ

モジュール→

名称	ID	接続
温湿度セン サー	u6Sh4uh5EuXU	オフライン

連携サービス

温湿度 WebSocket

+ サービス追加

モジュール登録



作成した連携サービスで、シリアルポートで表示される情報（温度/湿度/シリアル値）と同様の情報が画面上で受信データとして表示されていることを確認します。

送信されたデータの  
タイムスタンプ

データを送信した  
通信モジュールのID

データが格納された  
チャンネル番号

データの型

送信された値

受信データ

時刻	モジュール	チャンネル	型	値
2016-11-16T08:26:36.592526903Z	[REDACTED]	2	l	22
2016-11-16T08:26:36.592526903Z	[REDACTED]	1	f	36.120605
2016-11-16T08:26:36.592526903Z	[REDACTED]	0	f	28.995056
2016-11-16T08:26:35.51272341Z	[REDACTED]	2	l	21
2016-11-16T08:26:35.51272341Z	[REDACTED]	1	f	36.120605
2016-11-16T08:26:35.51272341Z	[REDACTED]	0	f	28.964844

単一メッセージで  
送信された値は  
同一時刻で表示

→カウント値

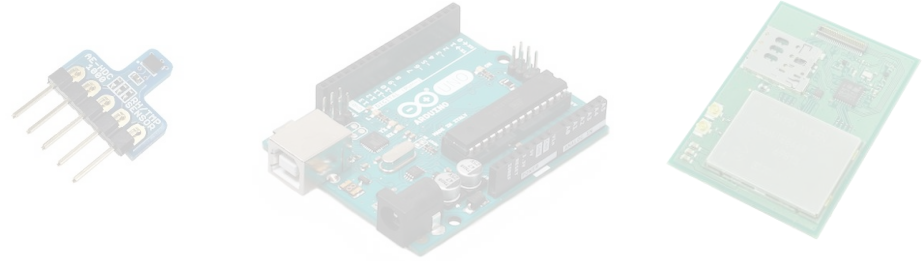
→湿度

→温度

# Webへのデータ連携 (Node-RED編)

マイコンおよびプログラムの構築

温湿度センサ (HDC1000)    マイコン (Arduino Uno)    さくらの通信モジュール



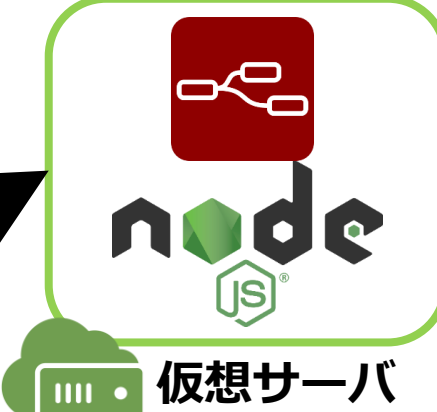
さくらのIoT Platformの設定

さくらのIoT Platform<sup>β</sup>



Webサービス連携 (さくらのクラウド)

node JS  
仮想サーバ



node JS  
コンテナ  
Webサービス連携 (Arukas)





以下URLより、「さくらのクラウド」のコントロールパネルにログインします。  
「<https://secure.sakura.ad.jp/cloud/>」

SAKURA Internet さくらのクラウド ホーム

さくらインターネット会員としてログイン:

会員ID パスワード ログイン

[新規登録はこちら](#) [会員IDを忘れた](#) [パスワードを忘れた](#)

会員IDを保存

さくらのクラウドユーザとしてログイン:

ユーザコード @ 会員ID

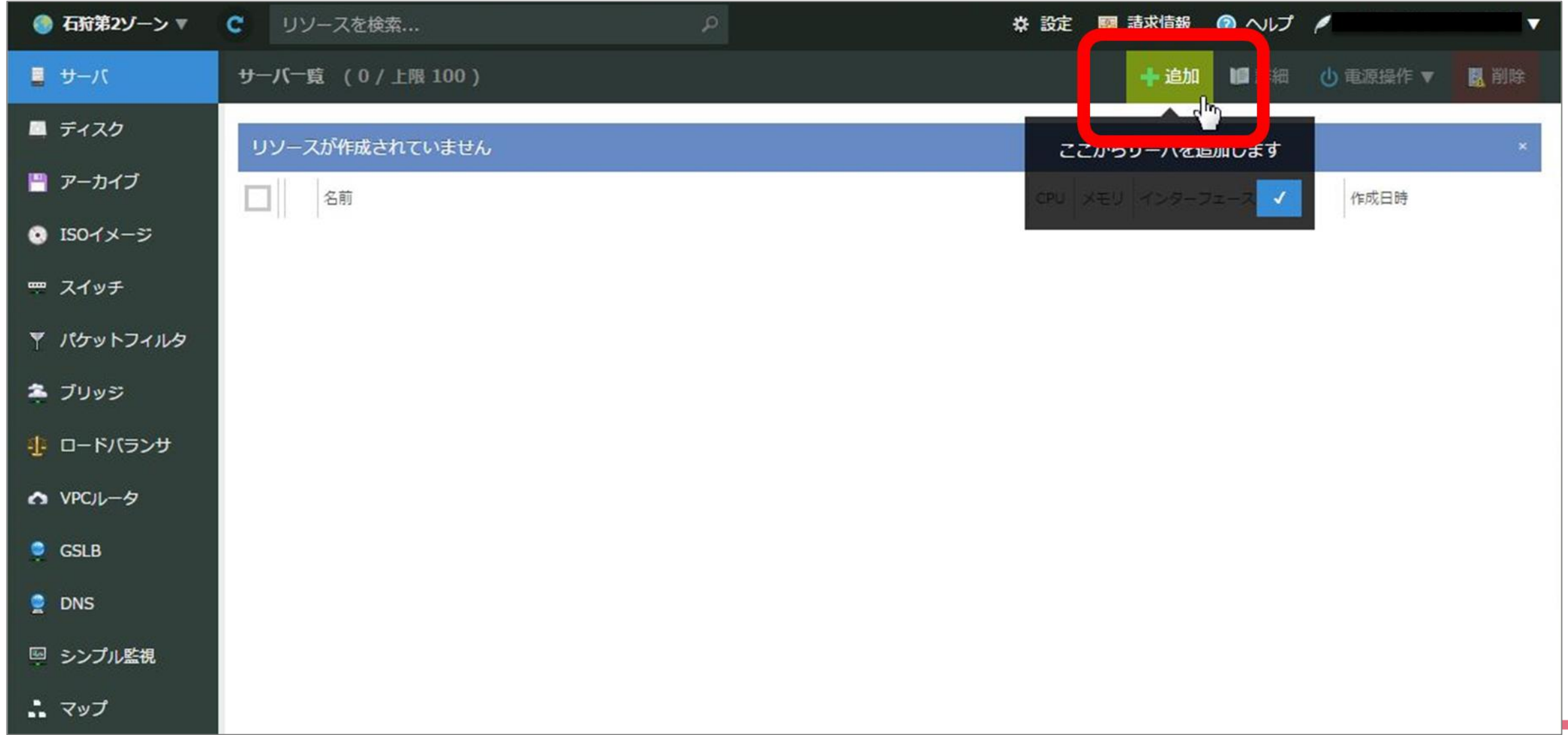
パスワード ログイン

ユーザコードと会員IDを保存

ユーザコード・パスワードが分からない場合はさくらインターネット会員としてログインするか、ユーザの管理者にお問い合わせください。

個人情報保護ポリシー 約款 © 2016 SAKURA Internet Inc.

左側のペインのサーバを選択し、右上の【追加】ボタンをクリックします。  
はじめはサーバ追加の案内が出る場合があります。





## 作業概要

- **Gitの入手**
- **NVM(Node Version Manager)の入手と設定**
- **Node.jsの入手**
- **Node-REDの入手と設定と起動**

まずはyumコマンドでGitのパッケージをインストールします。

```
-----  
#Gitのインストール  
-----
```

```
[root@test ~]# yum -y install git
```

```
Loaded plugins: fastestmirror, priorities, security
```

```
Setting up Install Process
```

```
Determining fastest mirrors
```

```
epel/metalink | 5.2 kB 00:00
```

```
* elrepo: ftp.ne.jp
```

```
* epel: www.ftp.ne.jp
```

```
～中略～
```

```
Updated:
```

```
git.x86_64 0:1.7.1-4.el6_7.1
```

```
Dependency Updated:
```

```
perl-Git.noarch 0:1.7.1-4.el6_7.1
```

```
Complete!
```

次にGitを使用してNode Version Manager(NVM)をインストールします。

```
-----  
#Node Version Manager(NVM)のインストール  
-----
```

```
[root@test ~]# git clone https://github.com/creationix/nvm.git ~/.nvm
```

```
Initialized empty Git repository in /root/.nvm/.git/
```

```
remote: Counting objects: 4732, done.
```

```
remote: Total 4732 (delta 0), reused 0 (delta 0), pack-reused 4731
```

```
Receiving objects: 100% (4732/4732), 1.27 MiB | 544 KiB/s, done.
```

```
Resolving deltas: 100% (2814/2814), done.
```

```
[root@test ~]# source ~/.nvm/nvm.sh
```

```
[root@test ~]# nvm help
```

```
Node Version Manager
```

```
～中略～
```

```
to remove, delete, or uninstall nvm - just remove the ` $NVM_DIR ` folder (usually ` ~/.nvm `)
```

NVMを使用し、Node-REDが稼動するためのNode.jsをインストールします。

```
-----  
#Node.jsのインストール  
-----
```

```
[root@test ~]# nvm ls-remote
```

```
  v0.1.14
```

```
  v0.1.15
```

```
~中略~
```

```
  v6.2.1
```

```
  v6.2.2
```

```
[root@test ~]# nvm install v6.2.2
```

```
Downloading https://nodejs.org/dist/v6.2.2/node-v6.2.2-linux-x64.tar.xz...
```

```
##### 100.0%
```

```
Now using node v6.2.2 (npm v3.9.5)
```

```
Creating default alias: default -> v6.2.2
```

```
[root@test ~]# node -v
```

```
v6.2.2
```

Node-REDをインストールし、起動します。以降はWebブラウザで作業します。

```
-----  
#Node-Redのインストール  
-----
```

```
[root@test ~]# npm install -g node-red
```

```
npm WARN deprecated i18next-client@1.10.3: you can use npm install i18next from version 2.0.0  
/root/.npm/versions/node/v6.2.2/bin/node-red ->
```

```
～中略～
```

```
`-- xmlbuilder@4.2.1  
  `-- lodash@4.13.1
```

```
[root@test ~]# node-red
```

```
Welcome to Node-RED
```

```
～中略～
```

```
21 Jun 14:29:05 - [info] Started flows
```

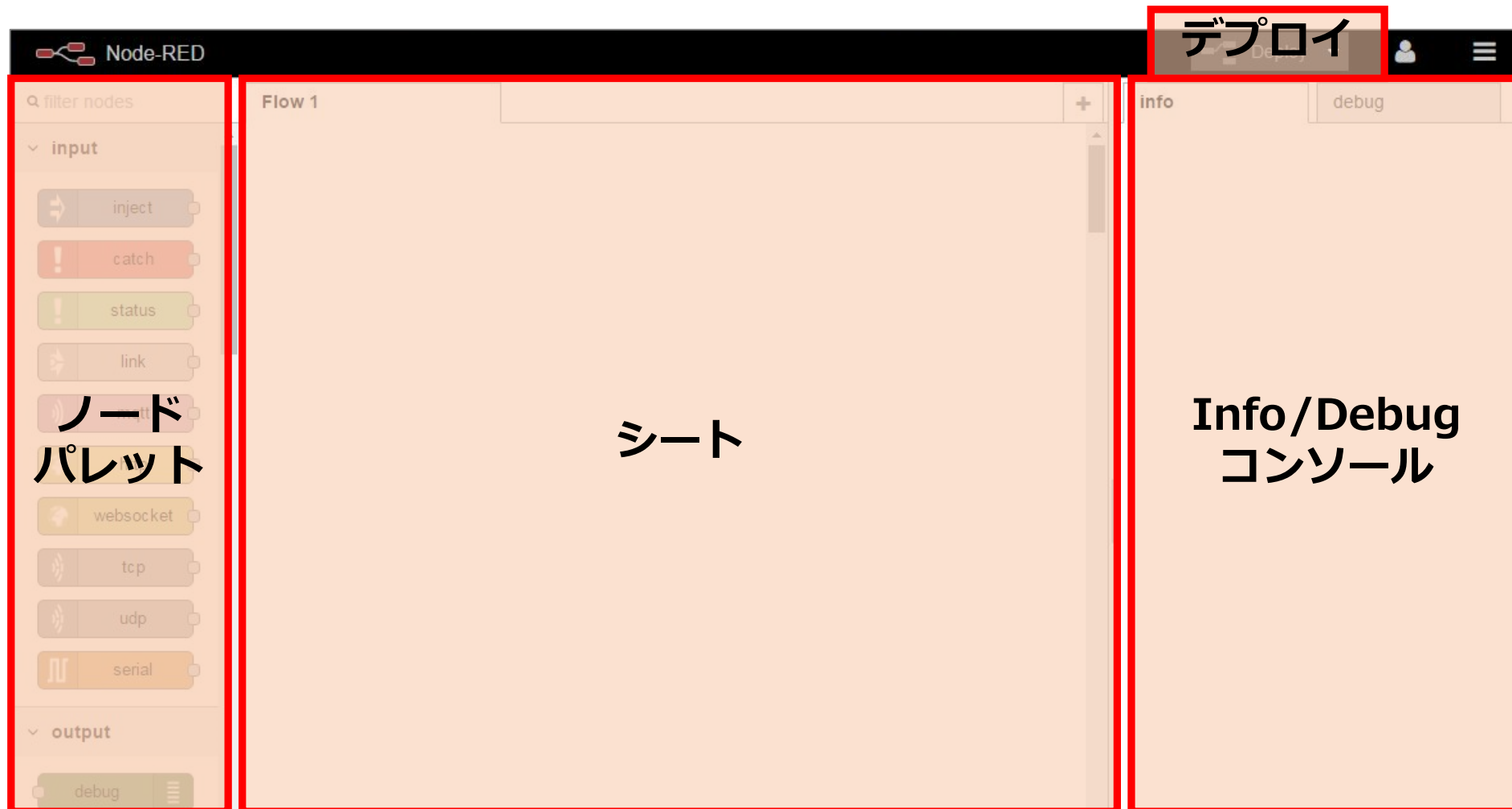
```
21 Jun 14:29:05 - [info] Server now running at http://127.0.0.1:1880/
```

このIPアドレスをサーバのグローバルIP  
アドレスに変更してブラウザでアクセス



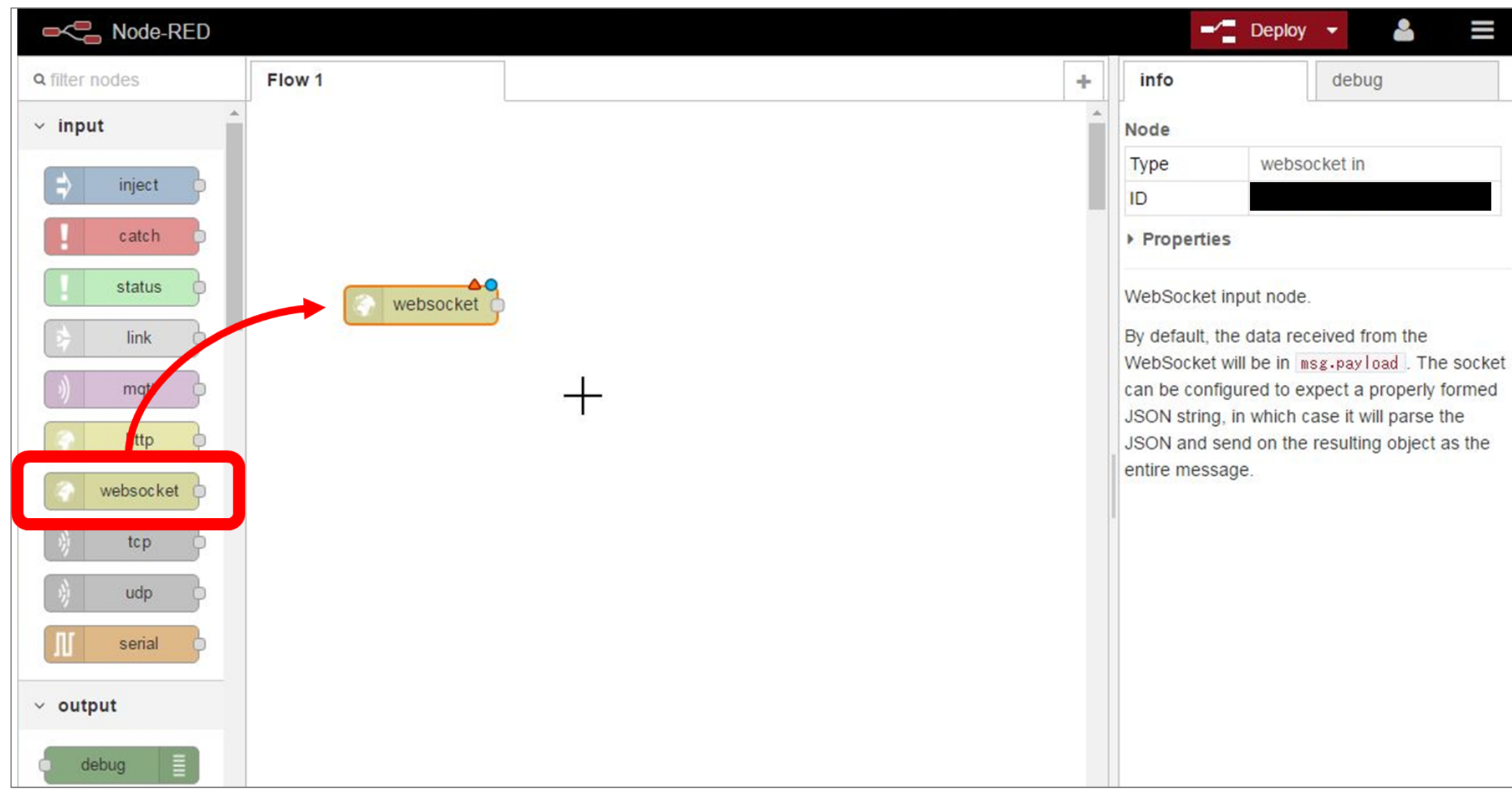
**http://127.0.0.1:1880/**

Node-REDは「ノード」と呼ばれる機能の固まりをシート上で組み合わせ、ひとつの「フロー」にすることで、ほとんどプログラミングを知らない人でもプログラムを構築することができるツールとなります。

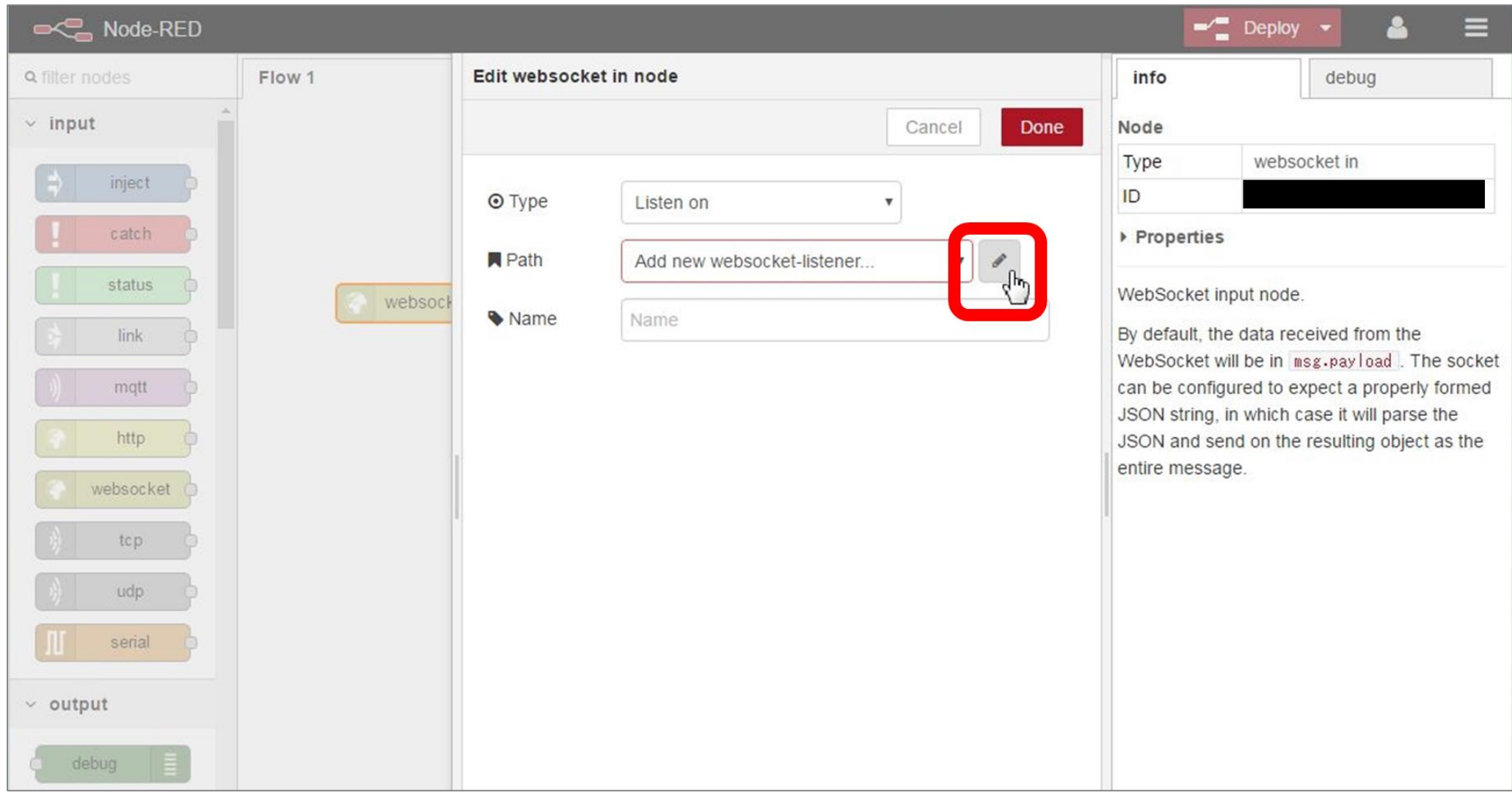




まずはWebSocketからのデータを受け取るノードを追加します。  
ノードパレットから「websocket」ノードをシートにドラッグ&ドロップします。



ドラッグ&ドロップされたWebSocketノードをWクリックし、設定画面に移ります。  
まずはPathの行にある鉛筆マークをクリックします。



指定するPathの値は、コンパネの連携サービスで確認できる赤枠部分となります。  
赤枠部分の情報をコピーして、WebSocketノードのPath部分にペーストします。

さくらのIoT Platform  $\alpha$

## テスト WebSocket

名前  
テスト

Token  
[Redacted]

WebSocket  
wss://secure.sakura.ad.jp/iot-alpha/ws/[Redacted]

受信データ

時刻	モジュール	チャンネル	型	値	
2016-06-20T09:17:08.514197428Z	[Redacted]	2	l	181	→シリアル値
2016-06-20T09:17:08.514197428Z	[Redacted]	1	f	46.246338	→湿度
2016-06-20T09:17:08.514197428Z	[Redacted]	0	f	29.41803	→温度

次に、ノードパレットから「debug」ノードをシートにドラッグ&ドロップします。  
Debugノードは自動で「msg.payload」に名前が変わり、特に設定は不要です。

The screenshot shows the Node-RED interface. On the left, the 'output' section of the node palette is expanded, and the 'debug' node is highlighted with a red box. A red arrow points from this node to the 'msg.payload' node in the main workspace. The workspace contains a 'sakura-websocket' node connected to the 'msg.payload' node. On the right, the sidebar is open to the 'debug' tab, displaying the node's properties and a detailed description of its functionality.

Node	
Type	debug
ID	[REDACTED]

**Properties**

The Debug node can be connected to the output of any node. It can be used to display the output of any message property in the debug tab of the sidebar. The default is to display `msg.payload`.

Each message will also display the timestamp, `msg.topic` and the type of property chosen to output.

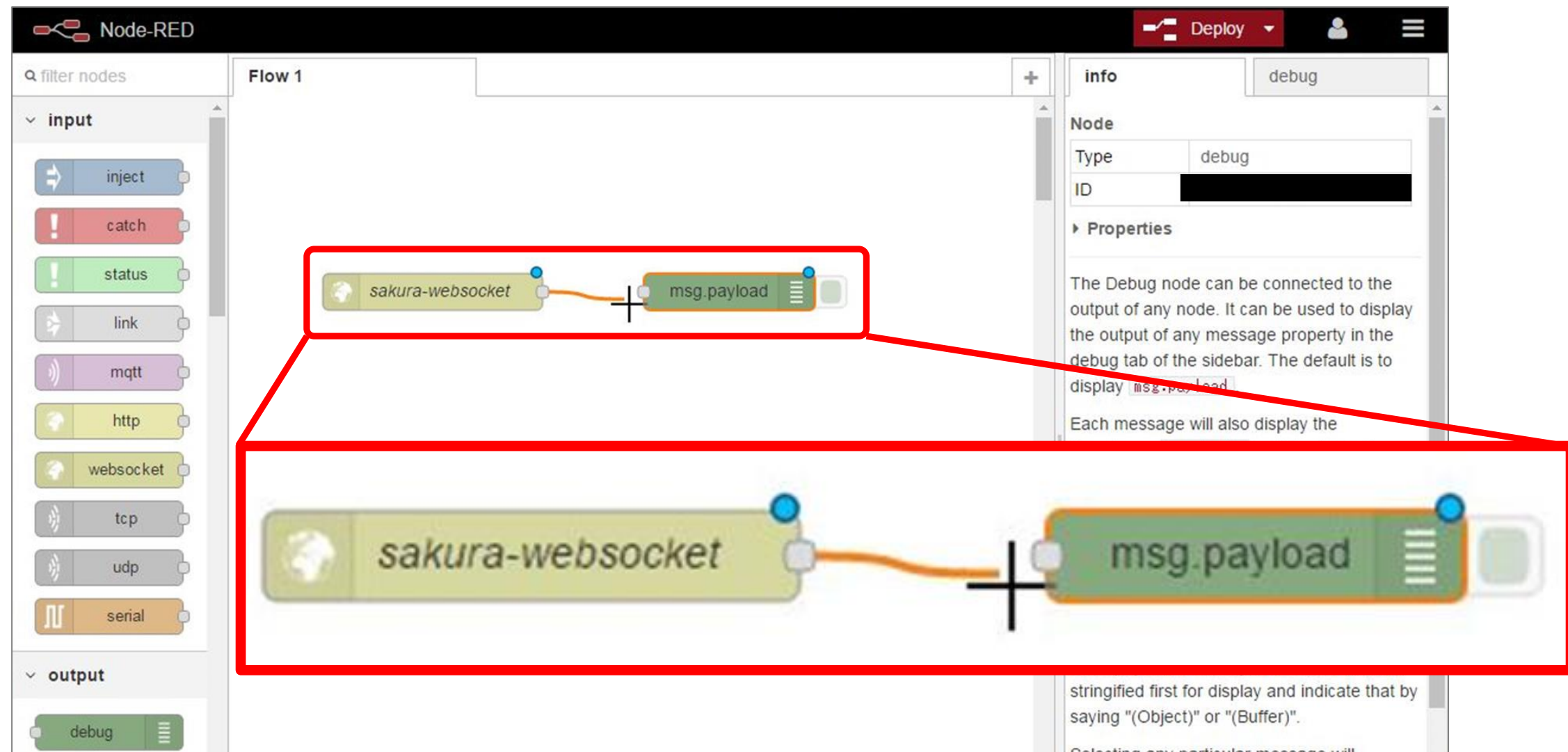
The sidebar can be accessed under the options drop-down in the top right corner.

The button to the right of the node will toggle its output on and off so you can de-clutter the debug window.

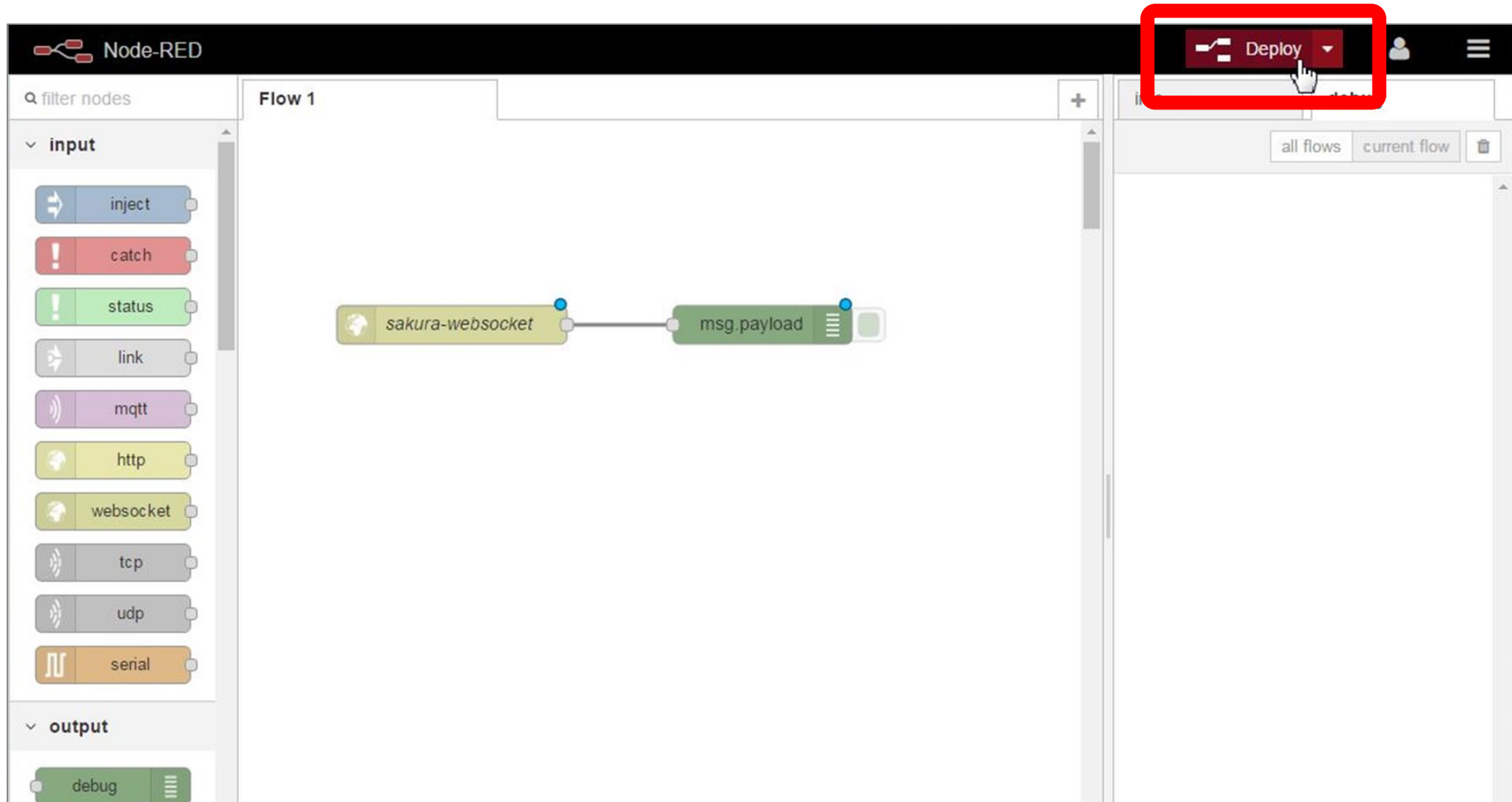
If the payload is an object or buffer it will be stringified first for display and indicate that by saying "(Object)" or "(Buffer)".

Selecting any particular message will highlight (in red) the debug node that

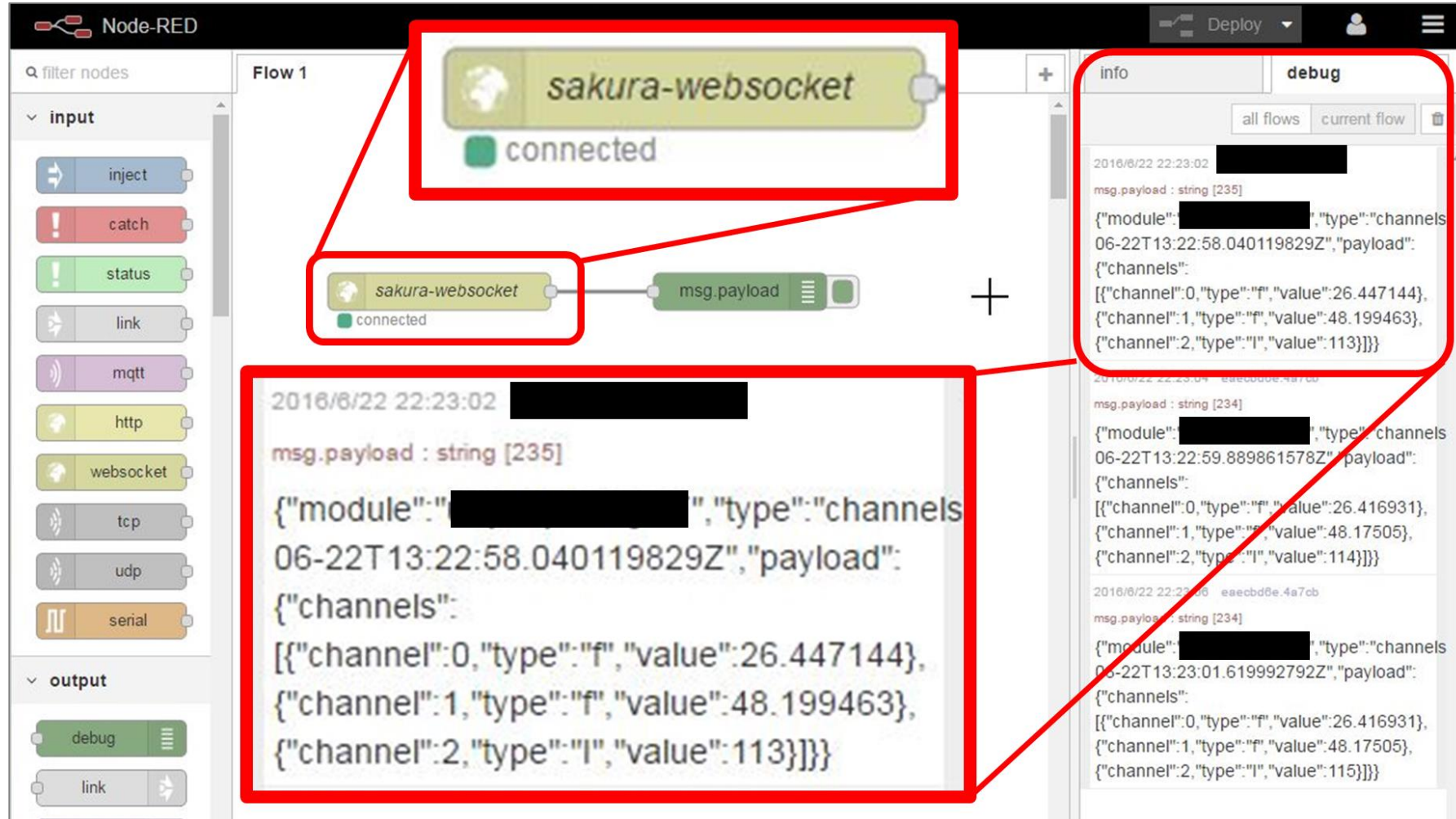
各ノードでの動作を処理として繋げるために、WebSocketノードの右端とDebugノードの左端をドラッグ&ドロップで線を繋ぎます。



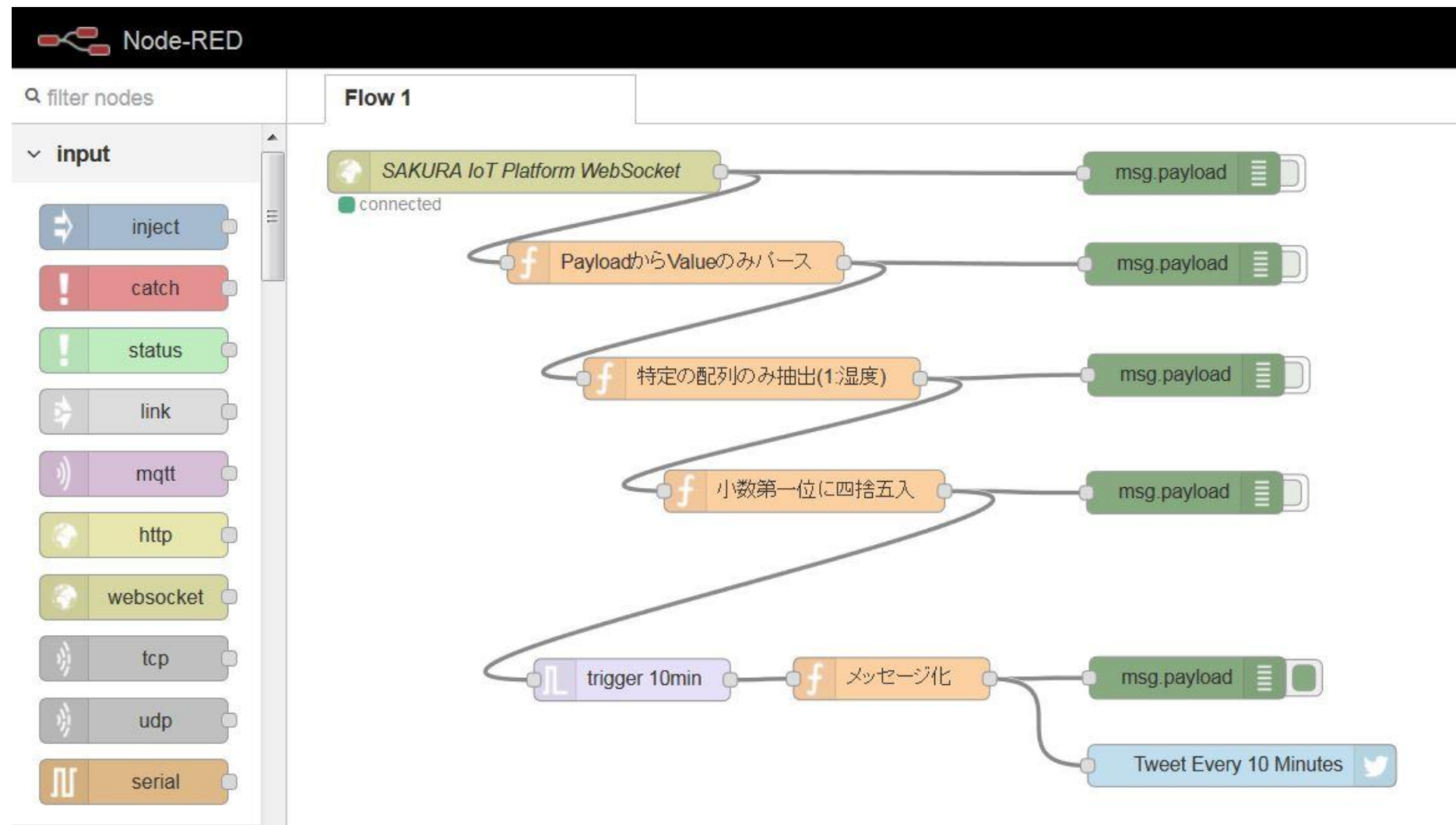
各ノードを接続し、準備が完了したら、右上部の【Deploy】をクリックします。  
デプロイが完了するとDeployボタンがグレースアウトされ、設定した内容を元に処理が開始されます。



フローに問題がない場合、Websocketノード下部に「connected」と表示され、コンソールのdebug内にプラットフォームから取得したJSONデータを確認できます。Debugノード右端の緑マークをクリックするとdebugへの表示が停止されます。



Websocketでデータを入力 → 温度と湿度を抽出 → 小数第一位で四捨五入  
→ 10分ごとにトリガー発生 → メッセージ作成 → Twitterに投稿





## 展示ブースにて実演中

ツイート ツイートと返信



法林浩之(バースト用) @hourin\_burst · 11月16日

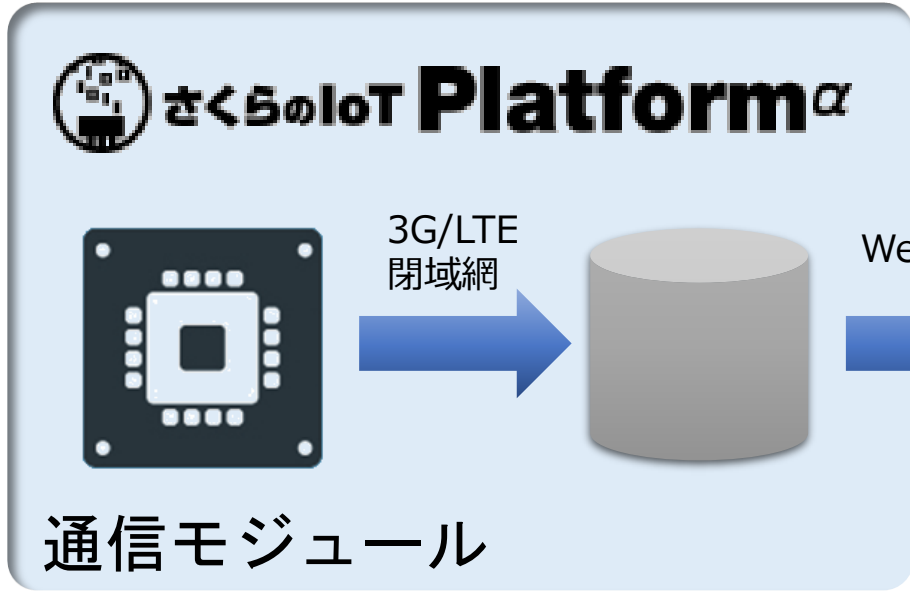
さくらインターネットのハンズオンで湿度情報を取得中！ただ今の現地湿度は56度だよ。β版もよろしくね！ #さくらのIoTPlatform #さくらインターネット



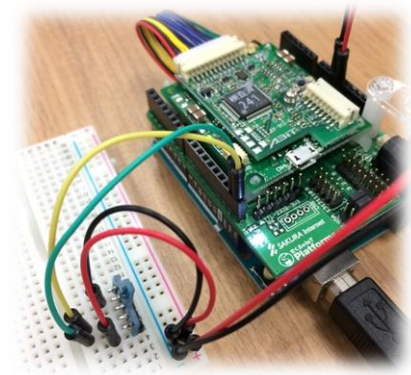
法林浩之(バースト用) @hourin\_burst · 11月16日

さくらインターネットのハンズオンで湿度情報を取得中！ただ今の現地湿度は53.4度だよ。β版もよろしくね！ #さくらのIoTPlatform #さくらインターネット

# Webへのデータ連携 (Zabbix編)

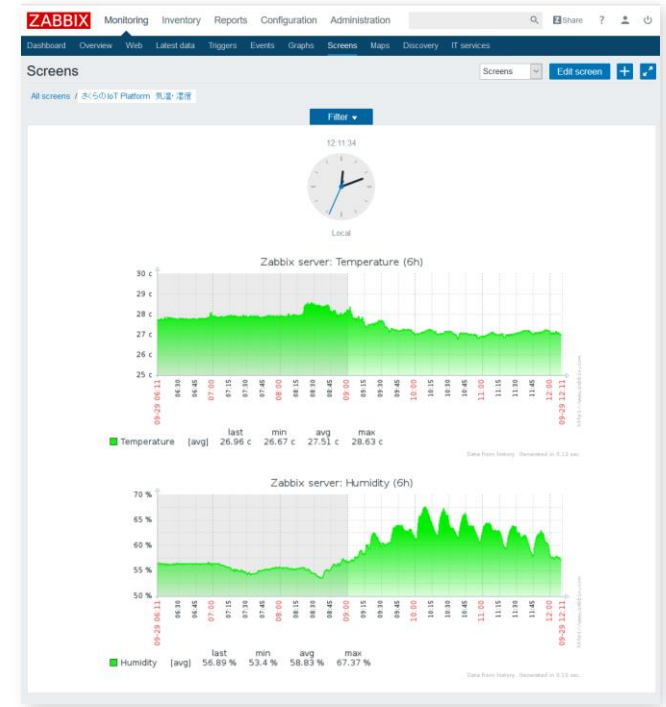


Linux (CentOS 7.2)  
<http://zabbix.sakura-pr.jp/>  
 (59.106.222.91)



Arduino  
 温湿度センサ

WebSocket 受信スクリプト(Perl)  
 ↓  
 zabbix\_sender で Zabbix に送信  
 ↓  
 Zabbix でデータ収集・グラフ化・アクション



- **サーバの作成**
- **Zabbixサーバの構築**
- **Zabbixにおける監視の設定**
- **さくらのIoT Platformからのデータ取得  
プログラムの設置**

- **田中さん@ZABBIX-JPの記事を参照**
  - **Zabbix 3.0をCentOS 7にインストール**
  - <http://qiita.com/atanaka7/items/294a639effdb804cfdaa>
- **作業の概略**
  - **CentOS 7のサーバを作成**
  - **firewalldの設定**
  - **Webサーバ(Apacheなど)の入手と設定**
  - **MariadbのインストールとDB作成**
  - **Zabbixのインストールと設定**



## 監視の有効化 → アイテムの作成 → グラフの追加 → スクリーンの追加

The screenshot shows the Zabbix web interface. At the top, there is a navigation bar with the ZABBIX logo and menu items: 監視データ, インベントリ, レポート, 設定, 管理. Below this is a sub-navigation bar with: ホストグループ, テンプレート, ホスト, メンテナンス, アクション, イベント, 相関関係, ディスカバリ. The main content area is titled 'アイテム' (Items). Below the title, there is a breadcrumb: すべてのホスト / Zabbix server. To the right of the breadcrumb are several filters: 有効 (Active), ZBX (selected), SNMP, JMX, IPMI, アプリケーション 12, and アイテム 73. A red box highlights the configuration form for a new item. The form fields are: 名前 (Name): Temperature; タイプ (Type): Zabbix-ラッパー (Zabbix wrapper); キー (Key): sakura\_iot\_temp; データ型 (Data type): 数値 (浮動小数) (Numeric (floating point)); 単位 (Unit): °C.

処理の流れ：

1. さくらの Iot Platform からの WebSocket を受信 (Mojoliciousを使用)
2. 温度と湿度のデータを zabbix\_sender で送信

```
#!/usr/bin/perl

use strict;
use warnings;
use Mojo::UserAgent;

my $ua = Mojo::UserAgent->new;
$ua->websocket('wss://secure.sakura.ad.jp/iot-alpha/ws/xxxxxxxxxx/' => sub {
  (略)
    open(CMD, "zabbix_sender -z 127.0.0.1 -s ¥"Zabbix server¥" -k
sakura_iot_temp -o $dat |");
    print "Temp:",$dat,"¥n";
    print "zabbix_sender -z 127.0.0.1 -s ¥"Zabbix server¥" -k sakura_iot_temp -o
",$dat,"¥n";
```

さくらのIoT Platform $\alpha$

### websocket-test WebSocket

名前  
websocket-test

Token  
WebSocket  
wss://secure.sakura.ad.jp/iot-alpha/ws/ad92d3a5-812b-4179-8164-e9

受信データ

時刻	モジュール	チャンネル	型	値
2016-09-29T05:36:02.76660137Z	u74ekA2GQJme	2	l	10615
2016-09-29T05:36:02.76660137Z	u74ekA2GQJme	1	f	52.404785
2016-09-29T05:36:02.76660137Z	u74ekA2GQJme	0	f	27.887268

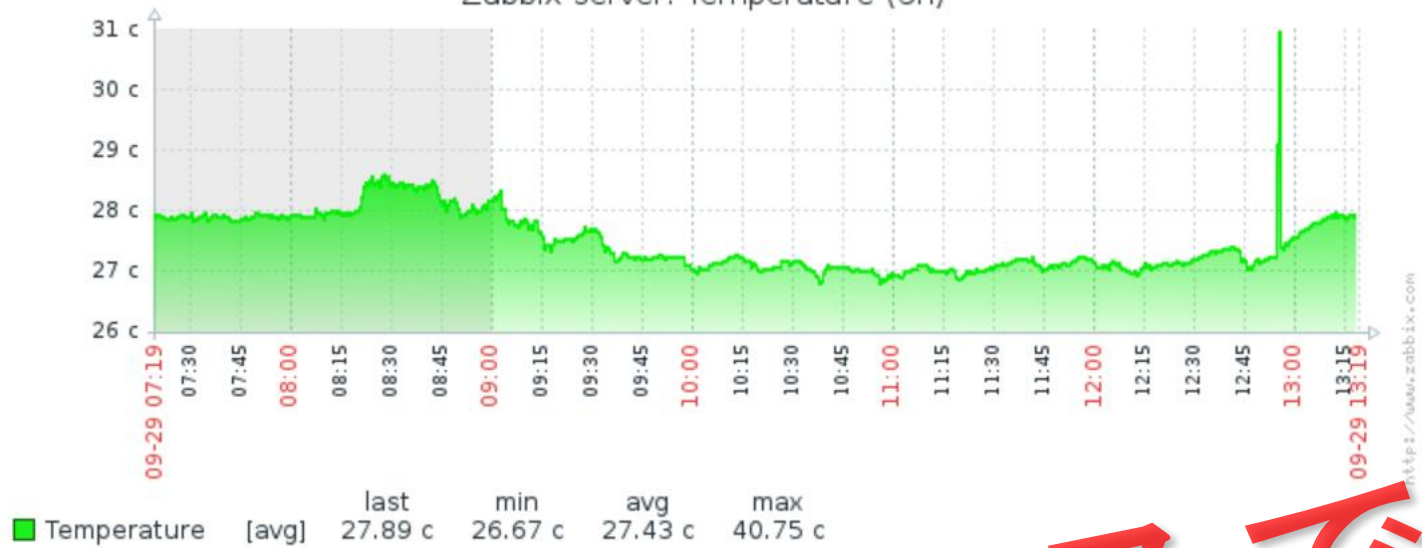
< 戻る

削除する 保存する

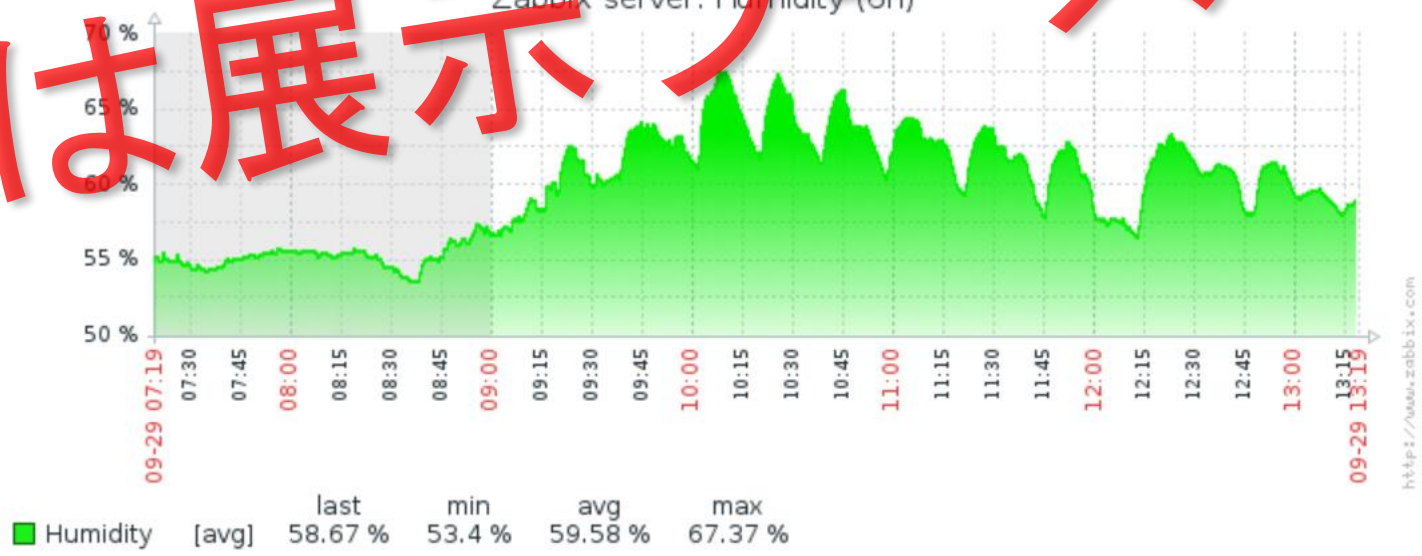




Zabbix server: Temperature (6h)



Zabbix server: Humidity (6h)



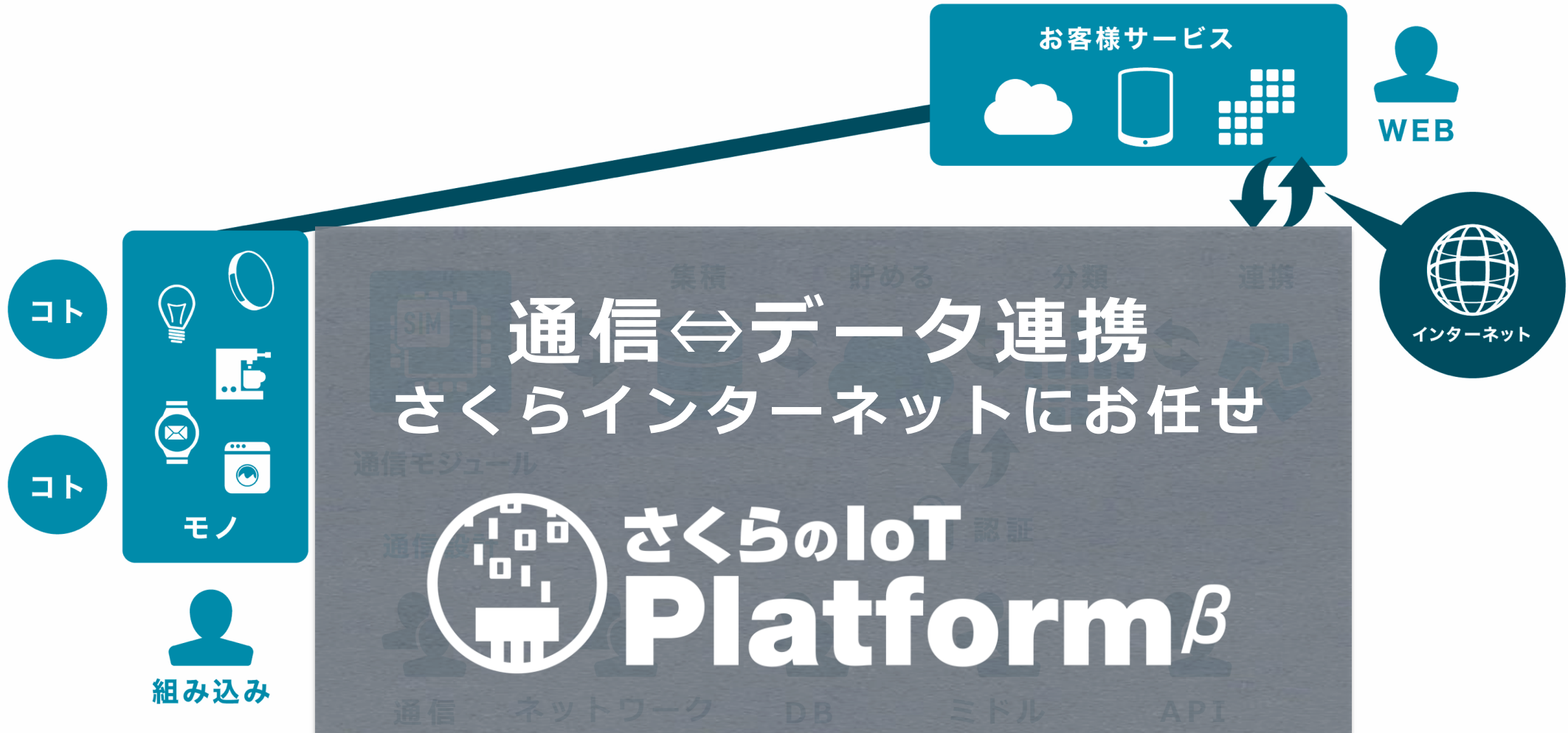
実機は展示ブースで

!!

まとめ




- さくらのIoT Platformの概要
  - 開発経緯
  - 主な機能/システム構成/パートナー連携
  - 事例
  - β版の販売について
- 実際に使ってみる
  - マイコンおよびプログラムの構築
  - さくらのIoT Platformの設定
  - Webサービスとの連携



既存の事業領域/スキルセットの大幅な変更なく  
モノ/サービスづくり、連携に注力可能

マイコンおよびプログラムの構築

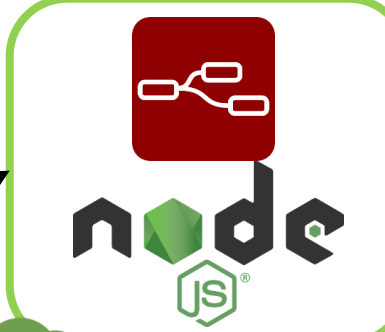
温湿度センサ (HDC1000)    マイコン (Arduino Uno)    さくらの通信モジュール




さくらのIoT Platformの設定



Webサービス連携 (Node-RED)



仮想サーバ



仮想サーバ

Webサービス連携 (Zabbix)

「さくらのクラウド」2万円クーポンも配布中！  
ステッカーやグッズも配布中！

- **広島でもさくらのイベントを開催したい！**
  - さくらのIoT Platformのハンズオン
  - さくらのクラウドやArukasなどのハンズオン
  - さくらのタベ / さくらクラブ など…
- **協力者求む！**
  - **会場の提供**
  - **参加者集め**
  - **地元コミュニティとの共催も可**

そこに、さくら