

さくらの IoT Platform を使ってみよう

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

DAY

2017/1/28

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)
- ・毎月5試合ぐらいあり



- 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

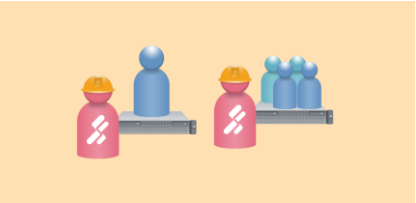


商号	さくらインターネット株式会社 (SAKURA Internet Inc.)
代表取締役	田中 邦裕
設立	1999年8月17日 (サービス開始: 1996年12月23日)
資本金	8億9,530万円
事業内容	インターネットでのサーバの設置およびその管理業務 電気通信事業法に基づく電気通信事業 マルチメディアの企画ならびに製作・販売 インターネットに関するコンサルティング
従業員数	339名 (2016年3月末)
所属団体	特定非営利活動法人日本データセンター協会 (JDCC) 社団法人コンピュータソフトウェア協会(CSAJ) 社団法人日本ネットワークインフォメーションセンター (JPNIC) 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

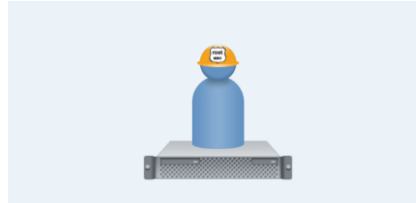
仮想サーバ



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



ホスティング



コロケーション



さくらのIoT Platform

さくらのサービス上で稼働



さくらのIoT Platform

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

各プレイヤーが既存の事業領域やスキルセットを大幅に変更せずに
モノ/サービスづくり、連携へ注力できるようにしたい

“IoT”とは何か？

Internet of Things Landscape 2016



“インターネット”と同じで広すぎる範囲





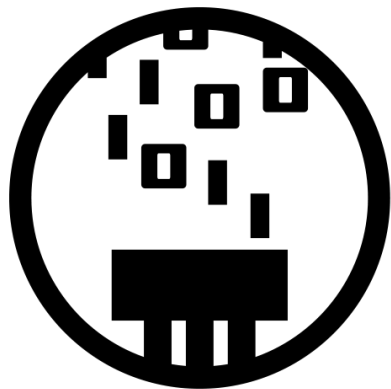
IoTが「ビジネスモデル転換」の前段となる「収集・蓄積」を担う



【Trillion Sensors Universe】

米国の起業家であるDr.Janusz Bryzek氏が2013年に提唱したビジョン。
「毎年1兆個のセンサーが活用される世界が2023年までに実現する」
というもの。

どうやって
データ
集めるの？



さくらのIoT
Platform

さくらのIoT Platform

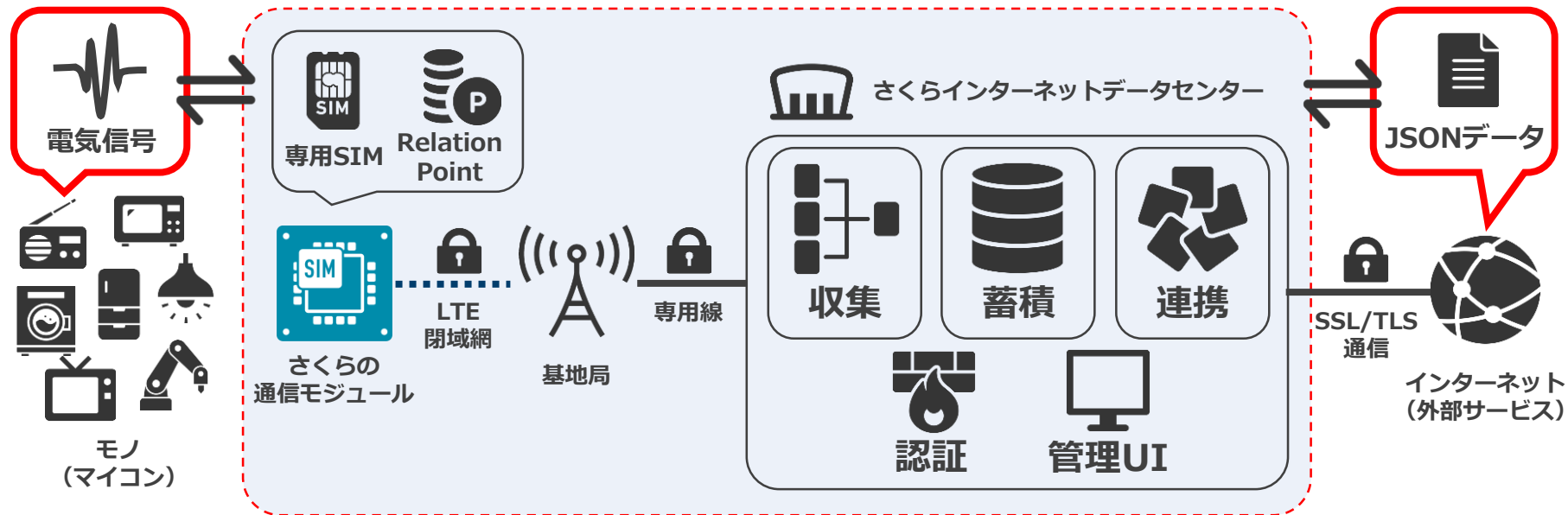
概要



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

さくらのIoT Platform ご提供範囲



モノ側の「通信モジュール」からサービス側の「連携」までを「電気信号とJSONデータの相互変換」をプラットフォームとして提供



	さくらのIoT Platform	他のIoT プラットフォーム
企画・アイデア		
モノ(製造)		とりあえずアプリやスマホで！ 設定は利用者側で！
センサー		安全性の担保は 開発者の負担に…
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/API/分析)	●	



	さくらのIoT Platform	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)

「データを迎えに行く」 という発想

- 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
- モノに組み込めば、電源を入れるだけで利用可能
- 利用者に接続の知識や現地の有線/無線LAN環境も不要





- 2015/12/24 「さくらの聖夜」にてしれっと発表
- 2016/02/08 α 版を発表(記者説明会/さくらの夕べ)
- 2016/10/05 CEATECにて β 版を発表
- 2016/11/01 β 版提供開始
- 2016年度内 正式版を提供予定

さくらのIoT Platform

機能詳細

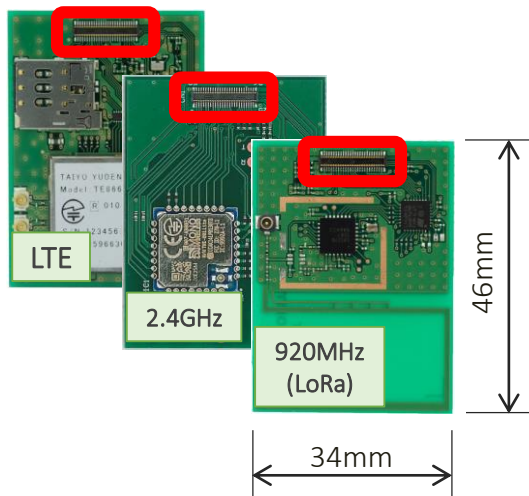


1. IoTデバイス、サービスの**“開発工数削減”**が可能
2. 閉域網を利用した**“Secure&Safety”**なネットワーク設計
3. プラットフォームサービスだから**“設計や運用は考慮不要”**
4. **“時刻提供機能”**でマイコンに現在時刻を提供
5. **“ファイル配信機能”**でマイコン側のアップデートも実現



IoTデバイス、サービスの“開発工数削減”が可能

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の無線規格への対応が容易



IoTデバイス、サービスの“開発工数削減”が可能

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

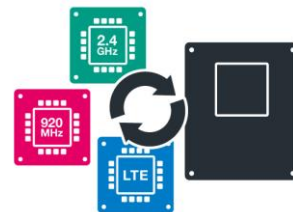
省電力制御

従来の通信手法
I²C/SPI

削減工数

さくらの
通信モジュール

アンテナ

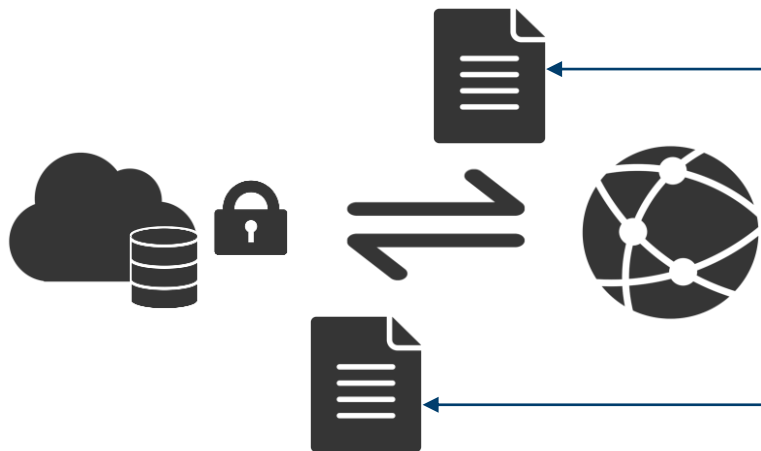


複数の無線規格に対応

「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



```
{
  "module": "XXXXXXXXX",
  "type": "channels",
  "datetime": "2016-06-01T12:21:11.628907163Z",
  "payload": {
    "channels": [{
      "channel": 1,
      "type": "i",
      "value": 1,
      "datetime": "2016-06-01T10:21:11.628907163Z"
    }, {
      "channel": 2,
      "type": "b",
      "value": "Of1e2d3c4b5c6b7a",
      "datetime": "2016-06-01T11:21:11.628907163Z"
    }
  ]
}
```

暗号化された経路上を扱いやすいJSONフォーマットでやり取り



閉域網を利用した”Secure & Safety”なネットワーク設計

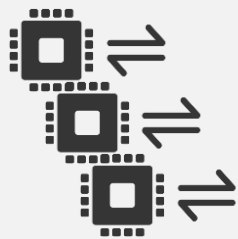


通信モジュール+SIMカードをセットで提供することで安全性を確保



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

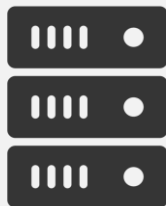


データの連携

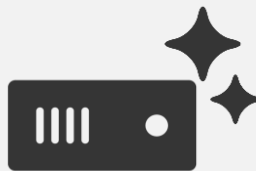


セキュリティ

サービスの
“運用”



ラージスケール対応



アップデート



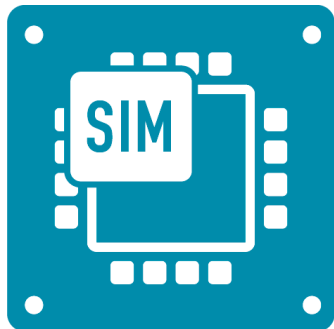
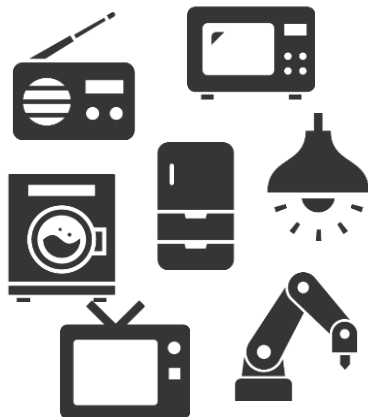
障害切り分け、復旧

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

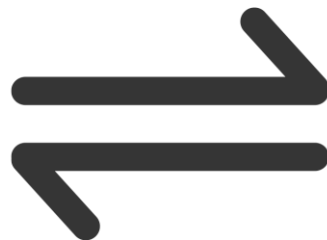


“時刻提供機能”でマイコンに現在時刻を提供

マイコンへの適用



時刻情報の要求



正確な時刻の提供



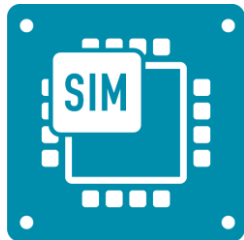
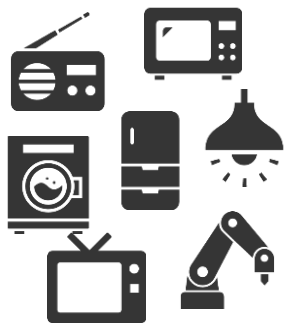
マイコン側でのログ記録等が求められるデバイスやサービスに

※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。

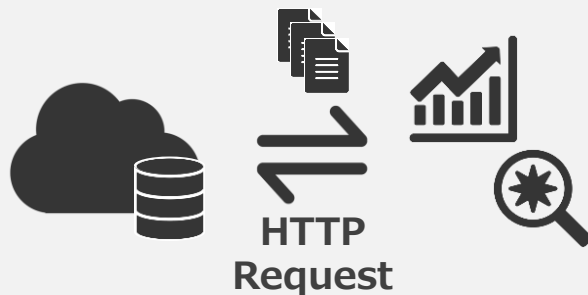
Outgoing Webhook/Incoming Webhook/WebSocket/MQTTなど



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



Microsoft



myThings



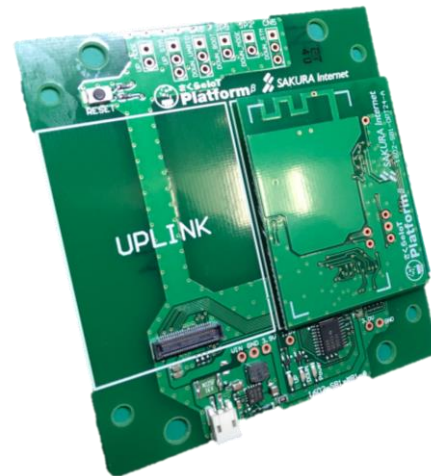
BOT TREE
for IoT

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

事例

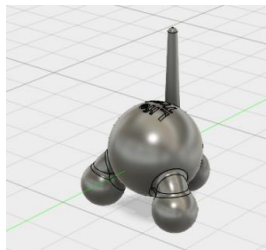


「なぜそれが必要なのか」をパートナーと協力し追求、
量産化対応、新機能、GW方式モジュール開発につなげ、
継続している。





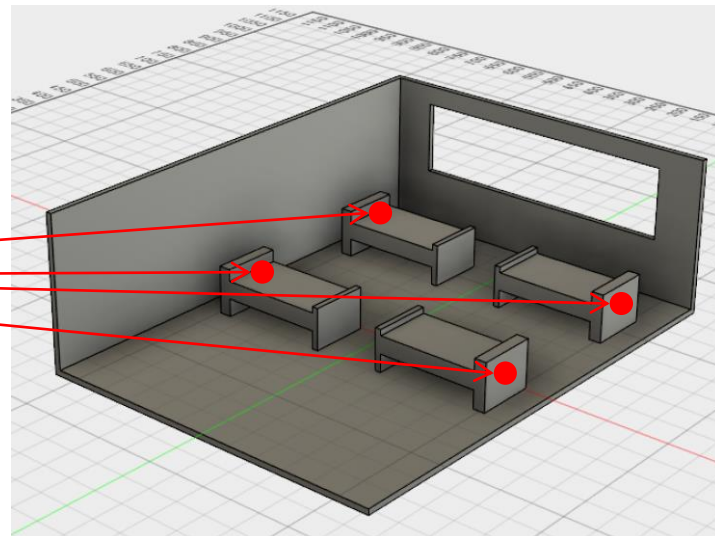
primesap



Intel Edison

加速度/温度/湿度/照度センサー
さくらの通信モジュール

病院での実証実験



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

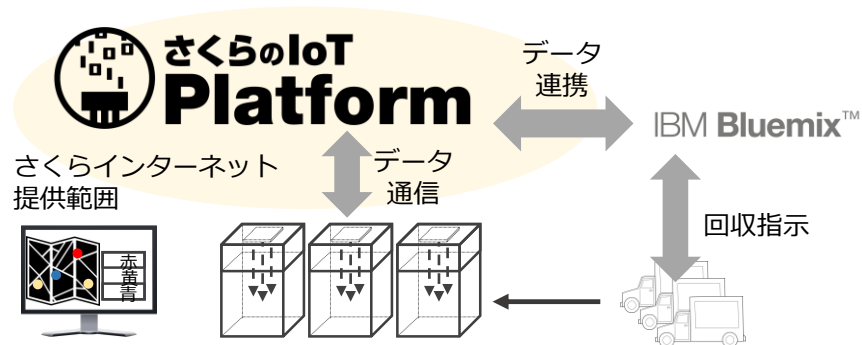


ハウステンボスとのPoC



※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

ご提供価格

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

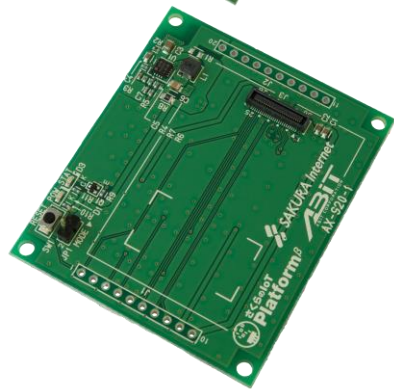


Arduinoシールドボード

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 (税別)



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 (税別)

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

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がると、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/ #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

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

DAY

2017/1/28

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

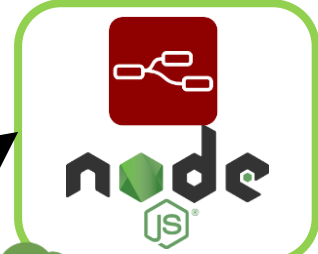
温湿度センサ (HDC1000) マイコン (Arduino Uno) さくらの通信モジュール



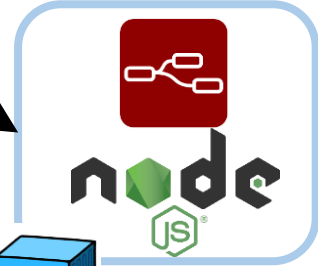
さくらのIoT Platformの設定



Webサービス連携 (さくらのクラウド)



仮想サーバ



コンテナ

Webサービス連携 (Arukas)



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)



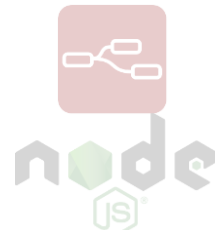
さくらの通信
モジュール



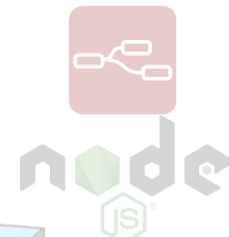
さくらのIoT
Platformの設定

さくらのIoT
Platform

Webサービス連携
(さくらのクラウド)



仮想サーバ



コンテナ

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。
2017/1/12時点での最新版は【1.8.1】となります。
Windowsは【Windows Installer】、Macは【 Mac OS X 10.7 Lion or newer】を選択します。

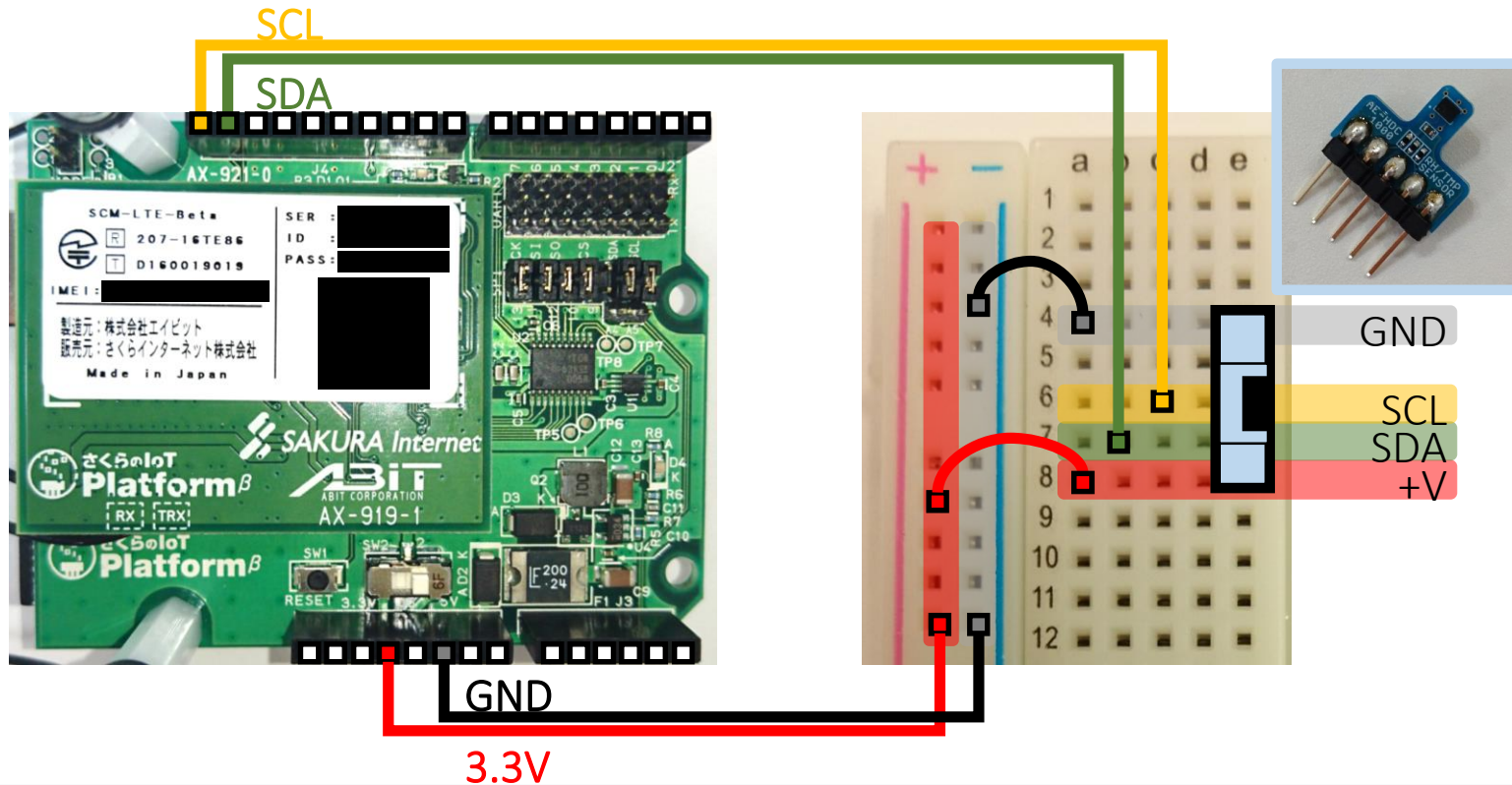
ARDUINO 1.6.13

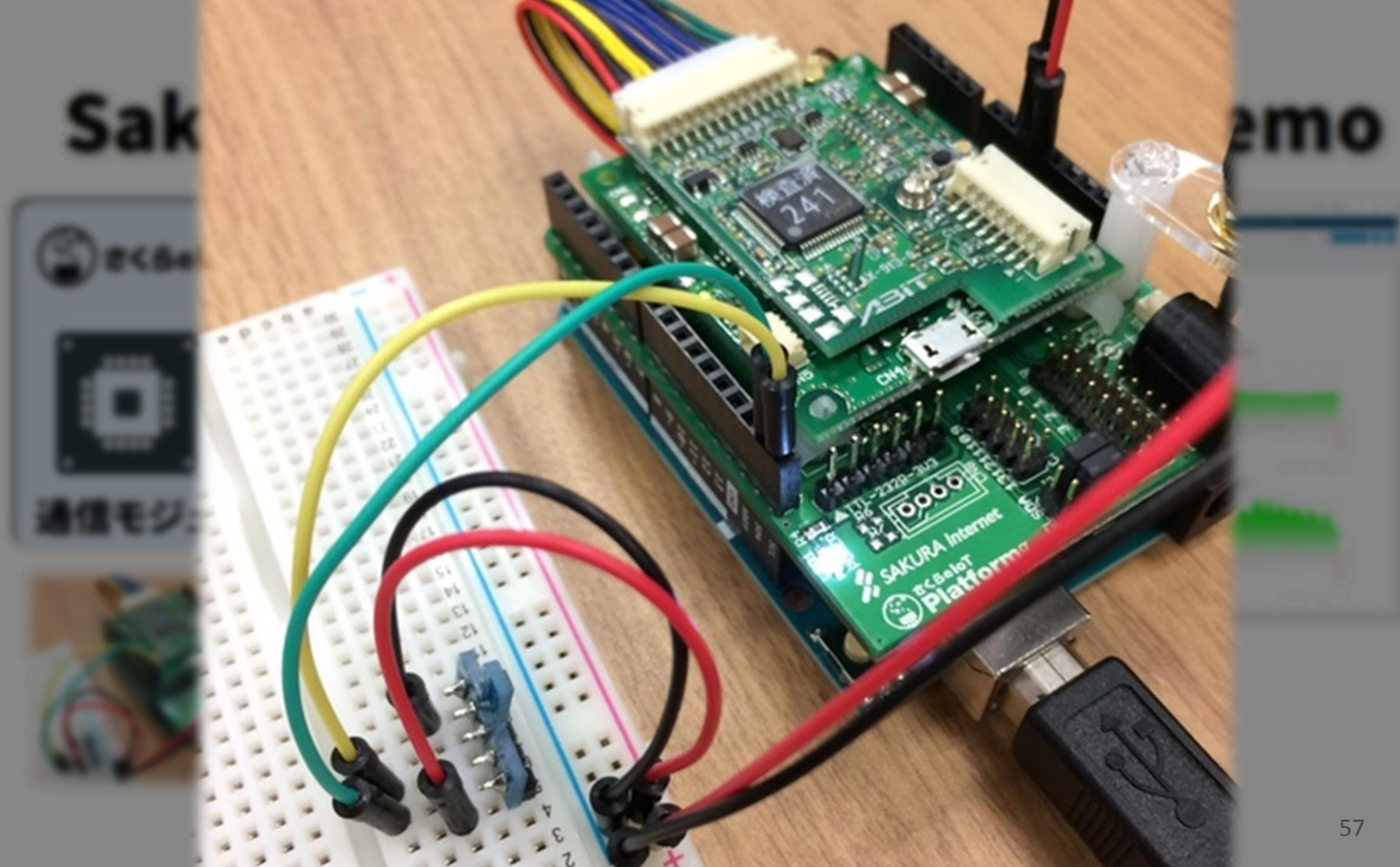
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows installer
Windows app
Mac OS X 10.7 Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM (experimental)

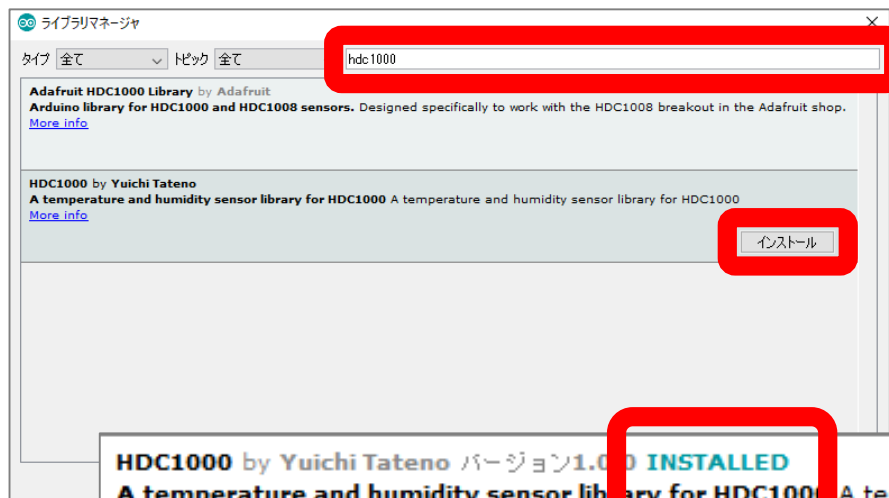
Release Notes
Source Code
Checksums (sha512)

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

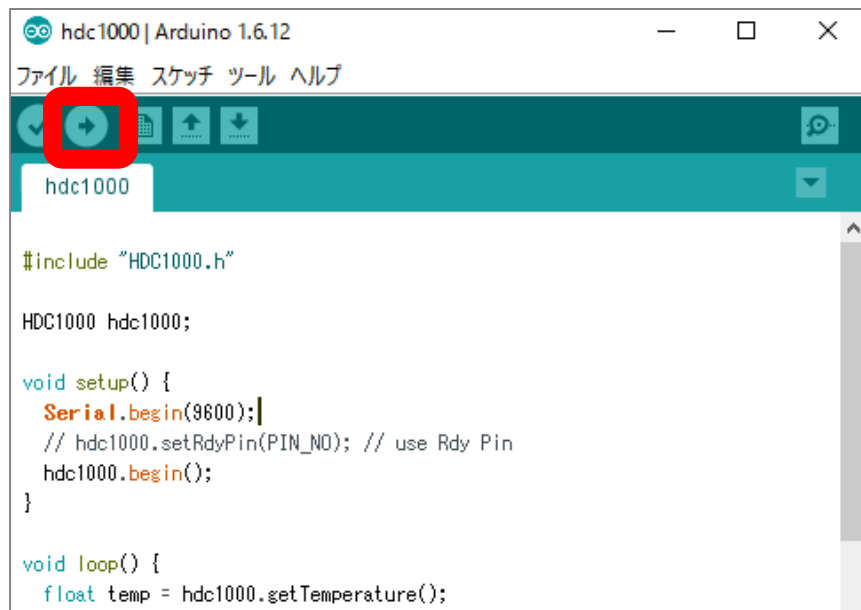




[スケッチ]→[ライブラリをインクルード]→ [ライブラリを管理...]をクリックし、
右上検索窓から【hdc1000】を検索すると、[HDC1000 by Yuichi Tateno]がヒットします。
インストールをクリックすると自動的に該当ライブラリが格納され、[INSTALLED]が表示されます。



[ファイル]→[スケッチの例]→[HDC1000]→[hdc1000]→【→】ボタンをクリックします。
[ツール]→[シリアルモニタ]よりTemperature & Humidity情報が取得されることを確認します。
何らかの問題があった場合、スケッチ下部にオレンジ色のエラーが表示されます。

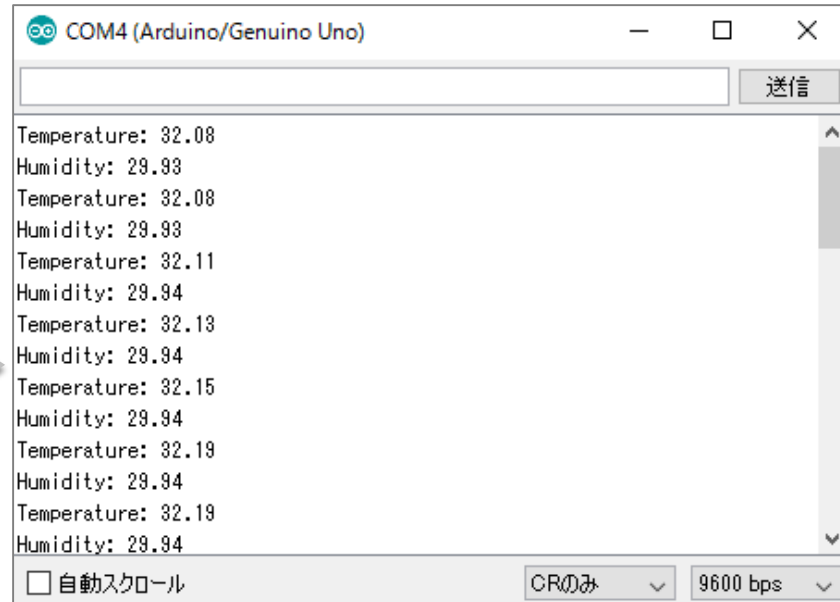


```
hdc1000 | Arduino 1.6.12
ファイル 編集 スケッチ ツール ヘルプ
hdc1000
#include "HDC1000.h"

HDC1000 hdc1000;

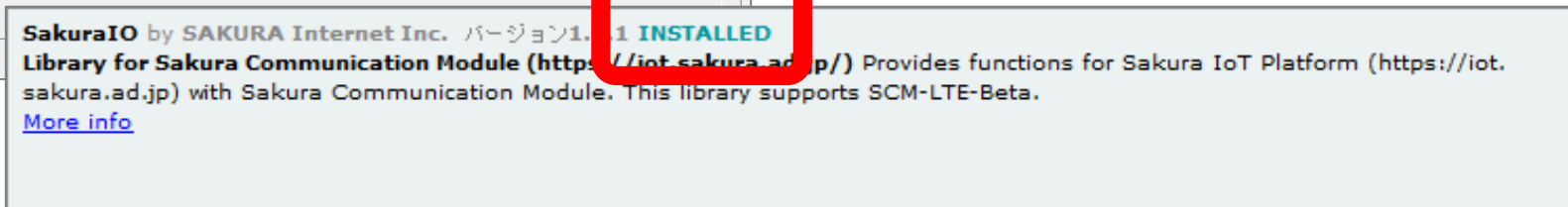
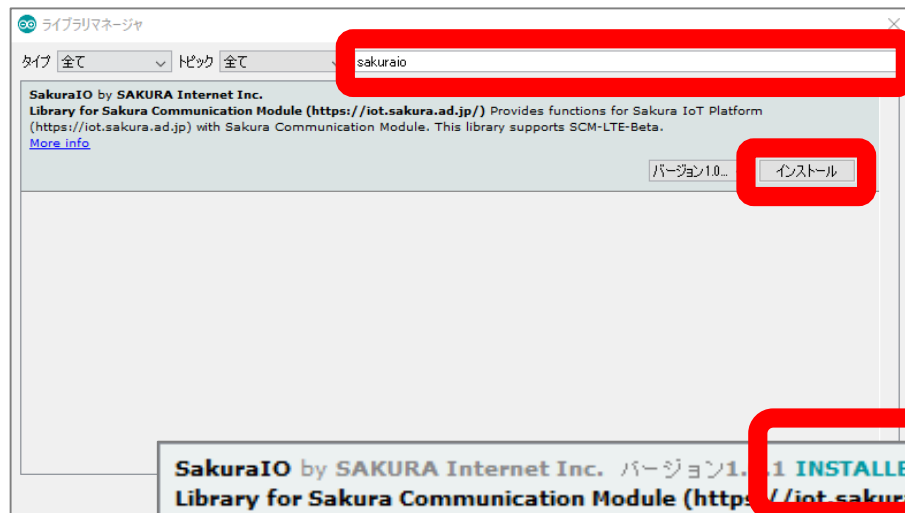
void setup() {
  Serial.begin(9600);
  // hdc1000.setRdyPin(PIN_NO); // use Rdy Pin
  hdc1000.begin();
}

void loop() {
  float temp = hdc1000.getTemperature();
```



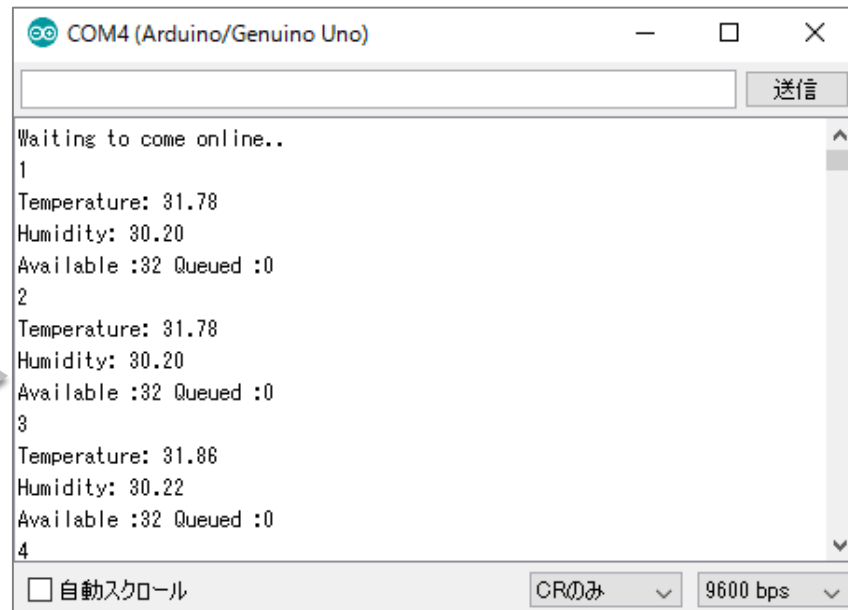
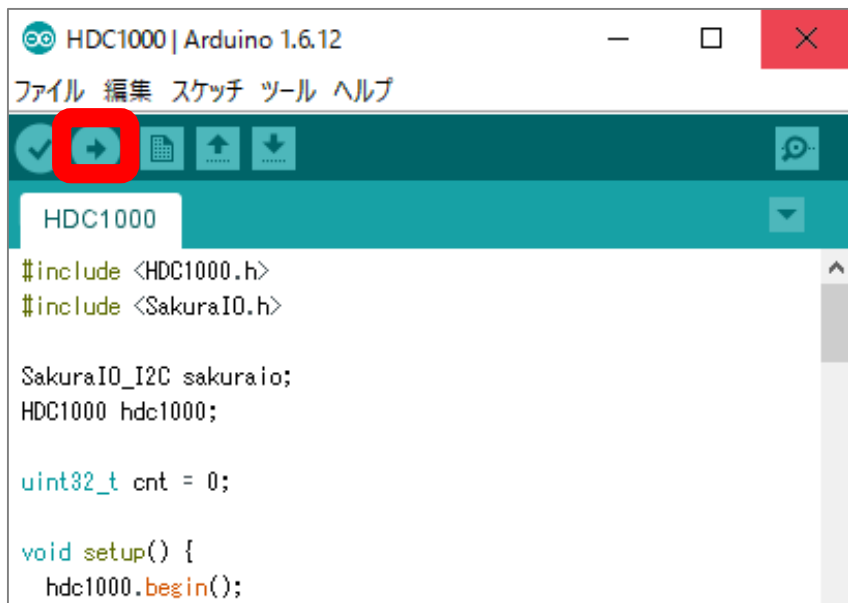
```
COM4 (Arduino/Genuino Uno)
送信
Temperature: 32.08
Humidity: 29.93
Temperature: 32.08
Humidity: 29.93
Temperature: 32.11
Humidity: 29.94
Temperature: 32.13
Humidity: 29.94
Temperature: 32.15
Humidity: 29.94
Temperature: 32.19
Humidity: 29.94
Temperature: 32.19
Humidity: 29.94
 自動スクロール
CRのみ 9600 bps
```

[スケッチ]→[ライブラリをインクルード]→ [ライブラリを管理...]をクリックし、
右上検索窓から【sakuraio】を検索すると、[SakuraIO by SAKURA Internet Inc.]がヒットします。
インストールをクリックすると自動的に該当ライブラリが格納され、[INSTALLED]が表示されます。





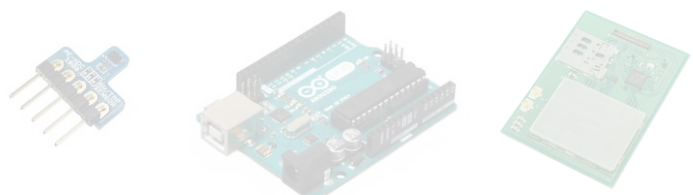
[スケッチの例]→[SakuraIO]→[HDC1000] →【→】ボタンをクリックします。
[ツール]→[シリアルモニタ]より「Waiting to come online」表記の後、カウント値、Temperature、Humidityに加え、Available(キューイング可能なチャンネル数)と Queued(キューで送信待ちになっているチャンネル数)が表示されることを確認します。



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



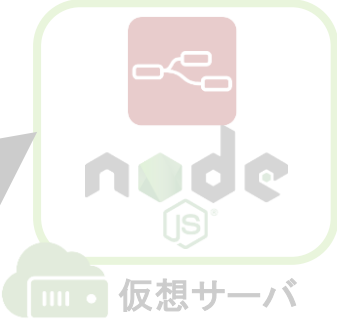
さくらのIoT Platformの設定

さくらのIoT Platform

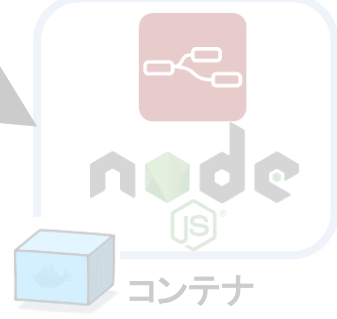


Webサービス連携 (さくらのクラウド)

node JS
仮想サーバ



node JS
コンテナ



Webサービス連携 (Arukas)

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/>)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

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。

The screenshot shows the Sakura IoT Platform control panel. The browser address bar indicates the URL is <https://secure.sakura.ad.jp/iot/>. The page title is "さくらのIoT Platformβ" and the user is logged in as "cod51192".

On the left, the text "プロジェクト" (Project) has an arrow pointing to a red-bordered box containing the project name "温湿度展示デモ".

Below that, the text "モジュール" (Module) has an arrow pointing to a red-bordered box containing a table of modules:

名称	ID	接続
温湿度セン サー	u6Sh4uh5EuXU	オフライン

Below the table is a button labeled "モジュール登録".

To the right of the table is a red-bordered box labeled "連携サービス" (Linked Service). It contains a table with one row:

名称	接続
温湿度	WebSocket

Below this table is a button labeled "+ サービス追加".

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	██████████	2	l	22
2016-11-16T08:26:36.592526903Z	██████████	1	f	36.120605
2016-11-16T08:26:36.592526903Z	██████████	0	f	28.995056
2016-11-16T08:26:35.51272341Z	██████████	2	l	21
2016-11-16T08:26:35.51272341Z	██████████	1	f	36.120605
2016-11-16T08:26:35.51272341Z	██████████	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

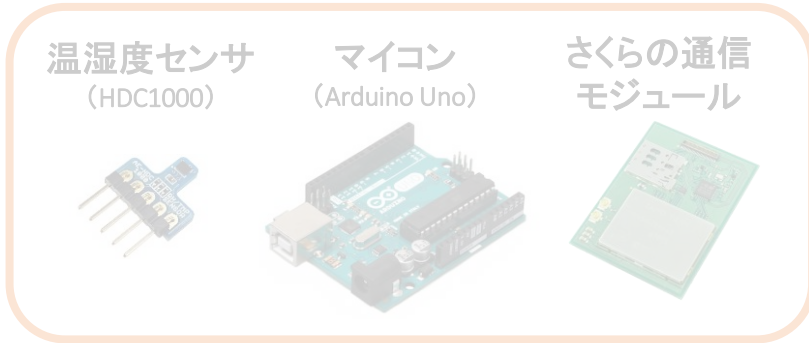
→湿度

→温度

Webへのデータ連携 (Node-RED編)

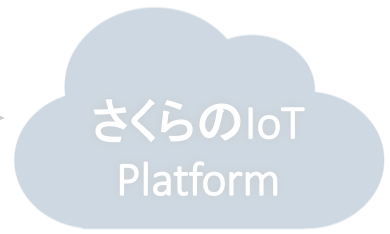
マイコンおよびプログラムの構築

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



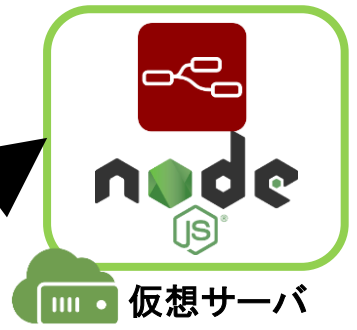
さくらのIoT Platformの設定

さくらのIoT Platform

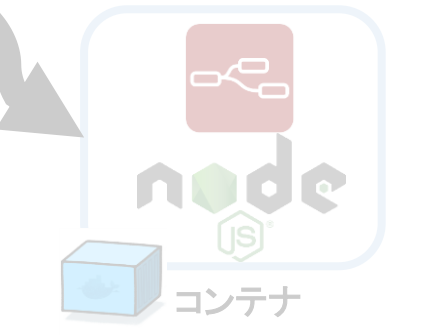


Webサービス連携 (さくらのクラウド)

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/」

The screenshot shows the login page for Sakura Cloud. It features two main sections. The left section is for logging in as a member, with fields for '会員ID' (Member ID) and 'パスワード' (Password), and a 'ログイン' (Login) button. The right section is for logging in as a user, with fields for 'ユーザコード' (User Code) and '会員ID' (Member ID) in the top row, and 'パスワード' (Password) and a green 'ログイン' (Login) button in the bottom row. A red rectangle highlights the user login section. At the bottom, there are links for '個人情報保護ポリシー' (Privacy Policy) and '約款' (Terms of Service), and a copyright notice for © 2016 SAKURA Internet Inc.

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

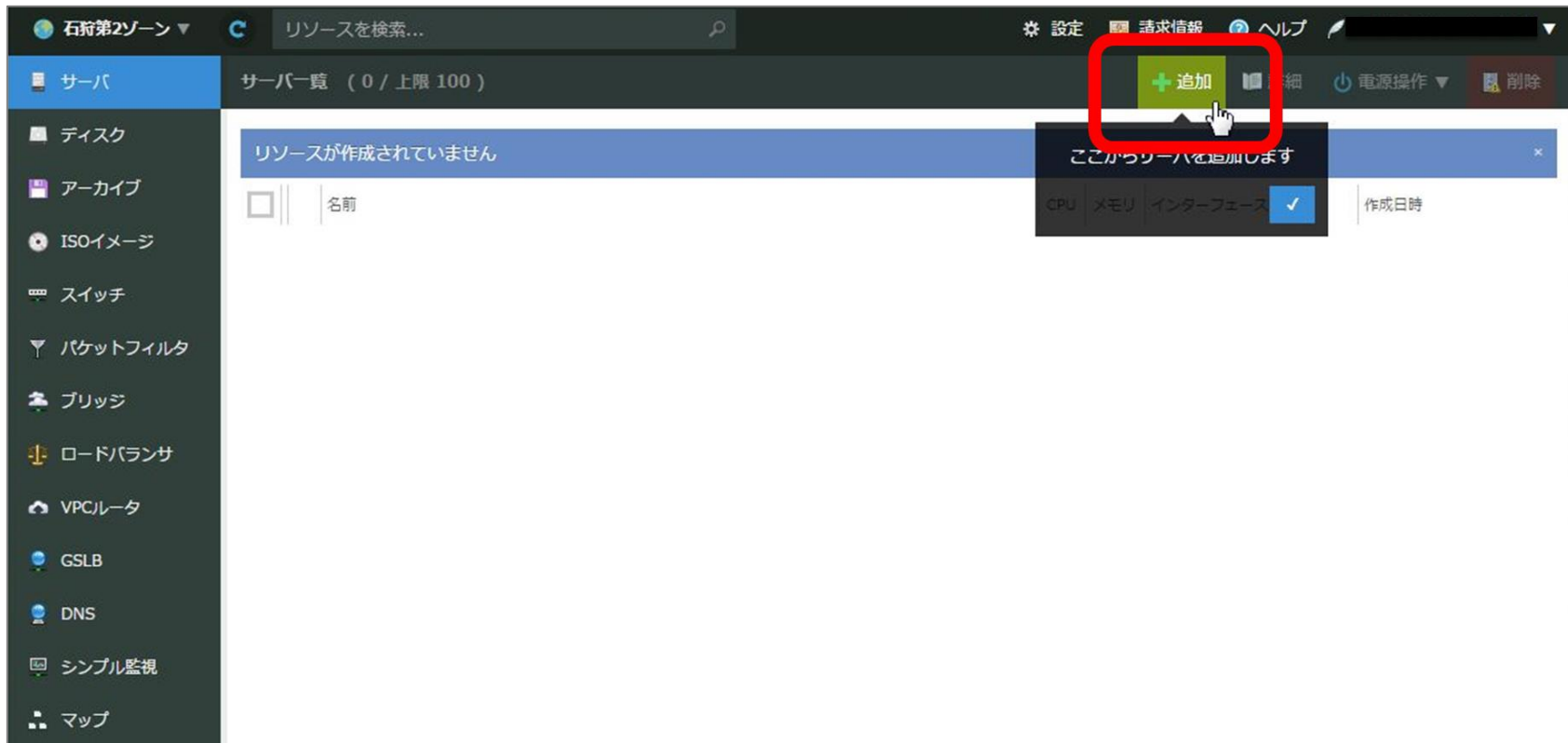
パスワード **ログイン**

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` folder
```



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～

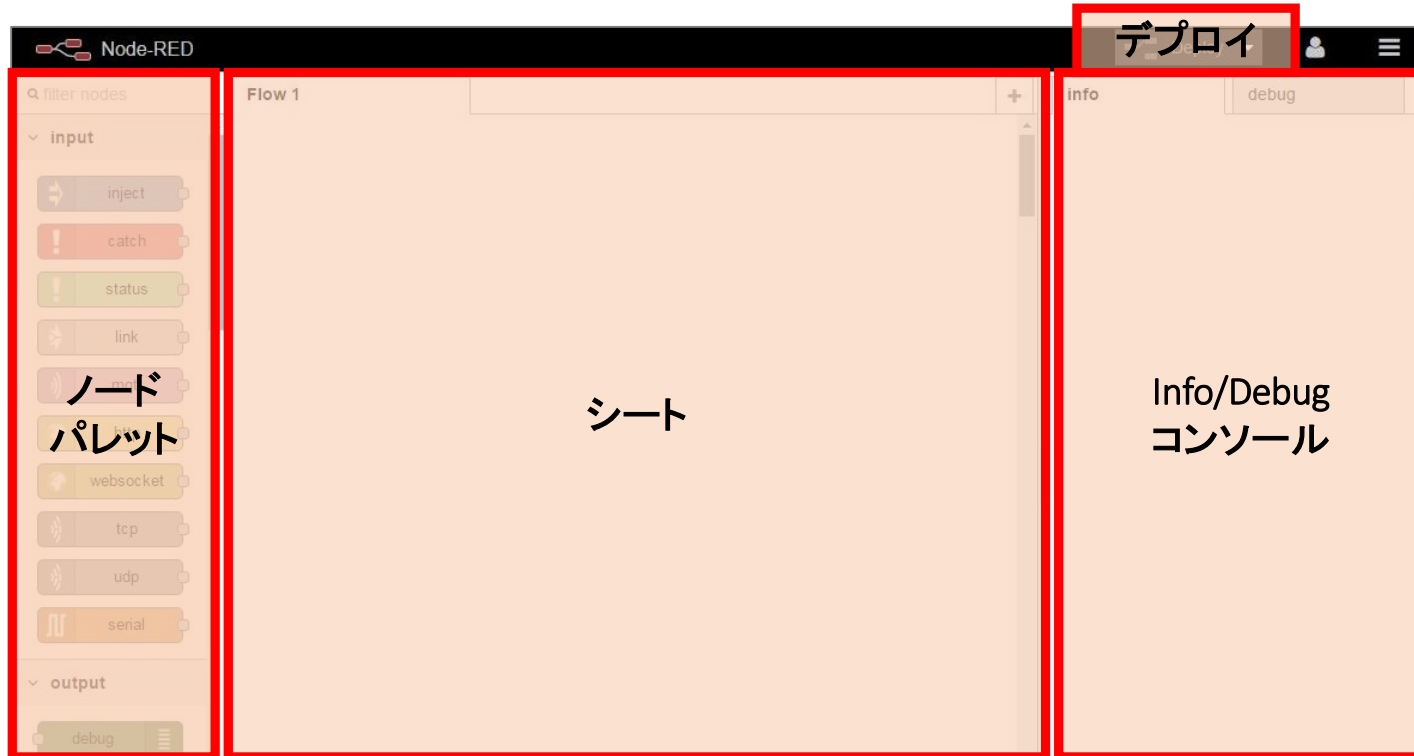
```
21 Jun 14:29:05 - [info] Started flows
```

```
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



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です。





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。

The screenshot shows the Node-RED interface. On the left, the 'input' section of the node palette is visible, with the 'websocket' node highlighted by a red box. A red arrow points from this node to a 'websocket' node already placed in the workspace. The workspace shows a flow with a single 'websocket' node. On the right, the 'info' tab is active, displaying the node's details:

Node	
Type	websocket in
ID	[REDACTED]

Properties

WebSocket input node.

By default, the data received from the WebSocket will be in `msg.payload`. The socket can be configured to expect a properly formed JSON string, in which case it will parse the JSON and send on the resulting object as the entire message.



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。
Typeは「Connect to」、Nameは「任意の名称」を入力の上、URLの行にある鉛筆マークをクリックします。

The screenshot shows the Node-RED interface with a 'websocket' node in the flow. The configuration panel for the 'websocket out' node is open, showing the following settings:

- Type: Connect to
- URL: Add new websocket-client... (with a pencil icon)
- Name: sakura-websocket

The right sidebar shows the 'info' tab with the following details:

Node	
Type	websocket out
ID	[REDACTED]

Properties

WebSocket out node.

By default, `msg.payload` will be sent over the WebSocket. The socket can be configured to encode the entire `msg` object as a JSON string and send that over the WebSocket.

If the message arriving at this node started at a WebSocket In node, the message will be sent back to the client that triggered the flow. Otherwise, the message will be broadcast to all connected clients.

If you want to broadcast a message that started at a WebSocket In node, you should delete the `msg._session` property within the flow.



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。

テスト WebSocket

名前

Token

WebSocket

受信データ

時刻	モジュール	チャンネル	型	値
----	-------	-------	---	---

< 戻る

削除する

+ 保存する



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」に名前が変わります。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 highlighted with a red box, and the 'debug' node is also highlighted with a red box. A red arrow points from the 'debug' node in the palette to the 'msg.payload' node in the flow. The flow contains a 'sakura-websocket' node connected to a 'msg.payload' node. The right sidebar shows the 'debug' node's configuration, including its type and ID, and a description of its functionality.

Node-RED

Deploy

filter nodes

Flow 1

input

- inject
- catch
- status
- link
- mqtt
- http
- websocket
- tcp
- udp
- serial

output

- debug

sakura-websocket

msg.payload

+

info

debug

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.

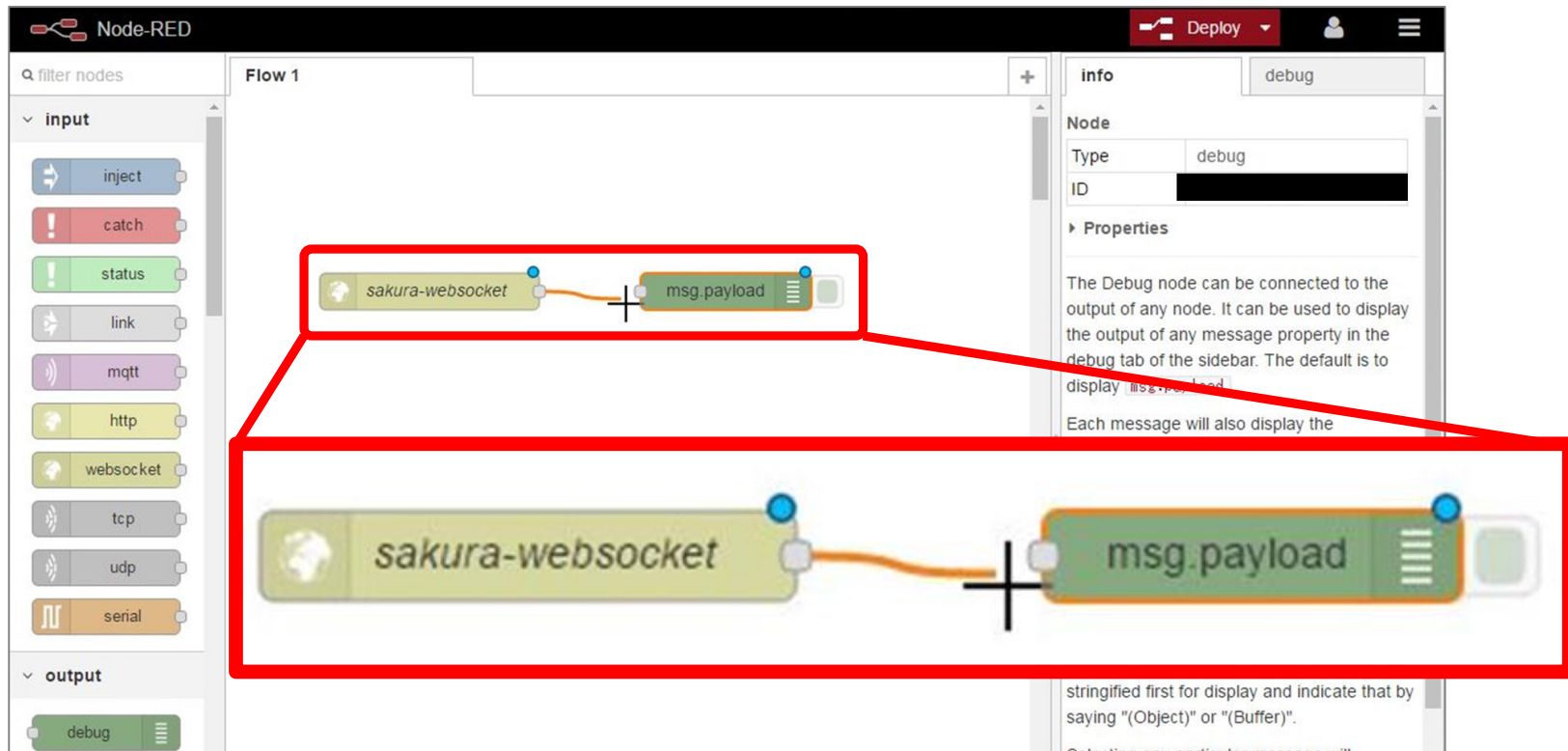
The sidebar can be accessed under the options drop-down in the top right corner.

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

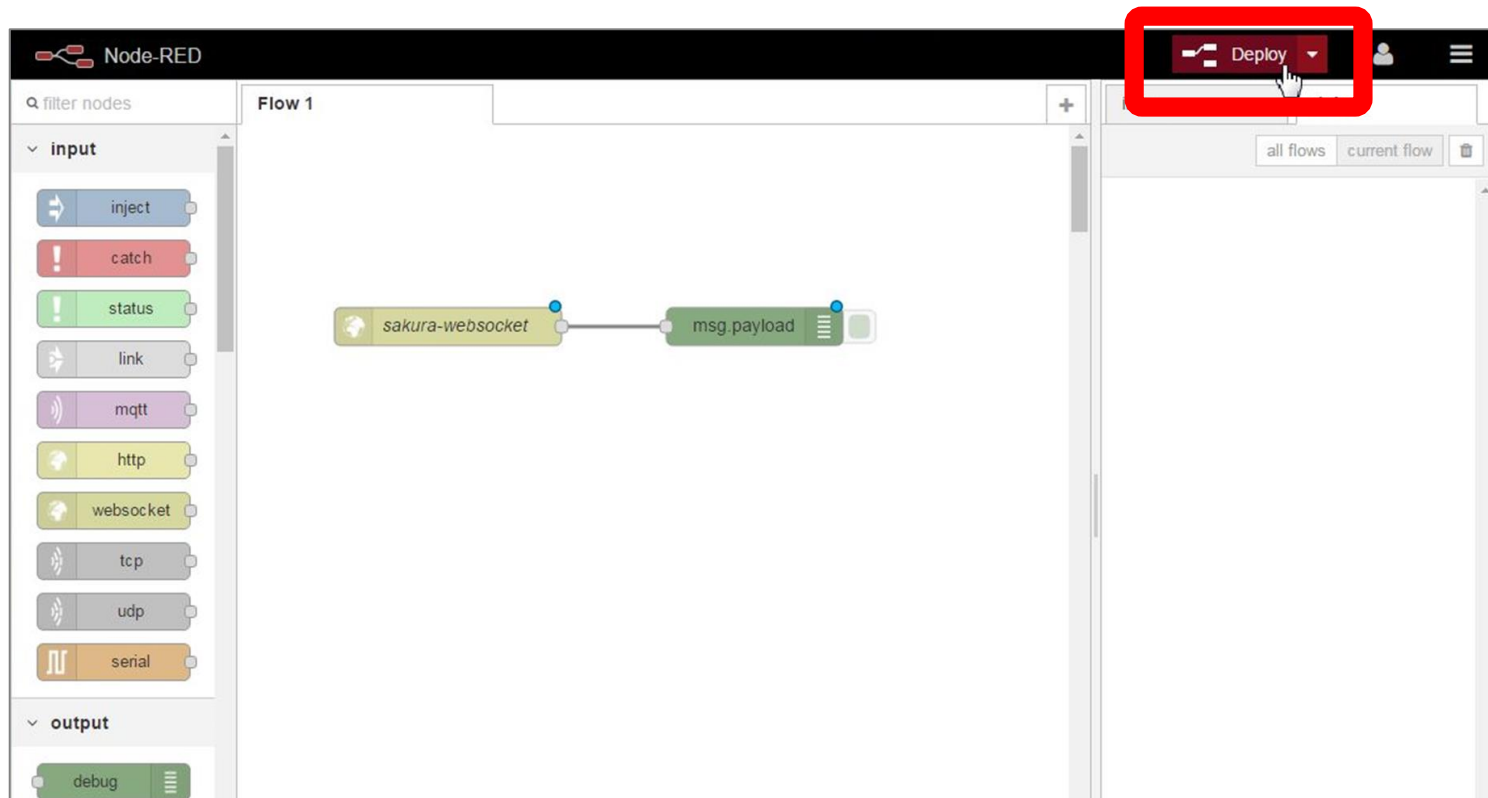
If the payload is an object or buffer it will be stringified first for display and indicate that by saying "(Object)" or "(Buffer)".

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。

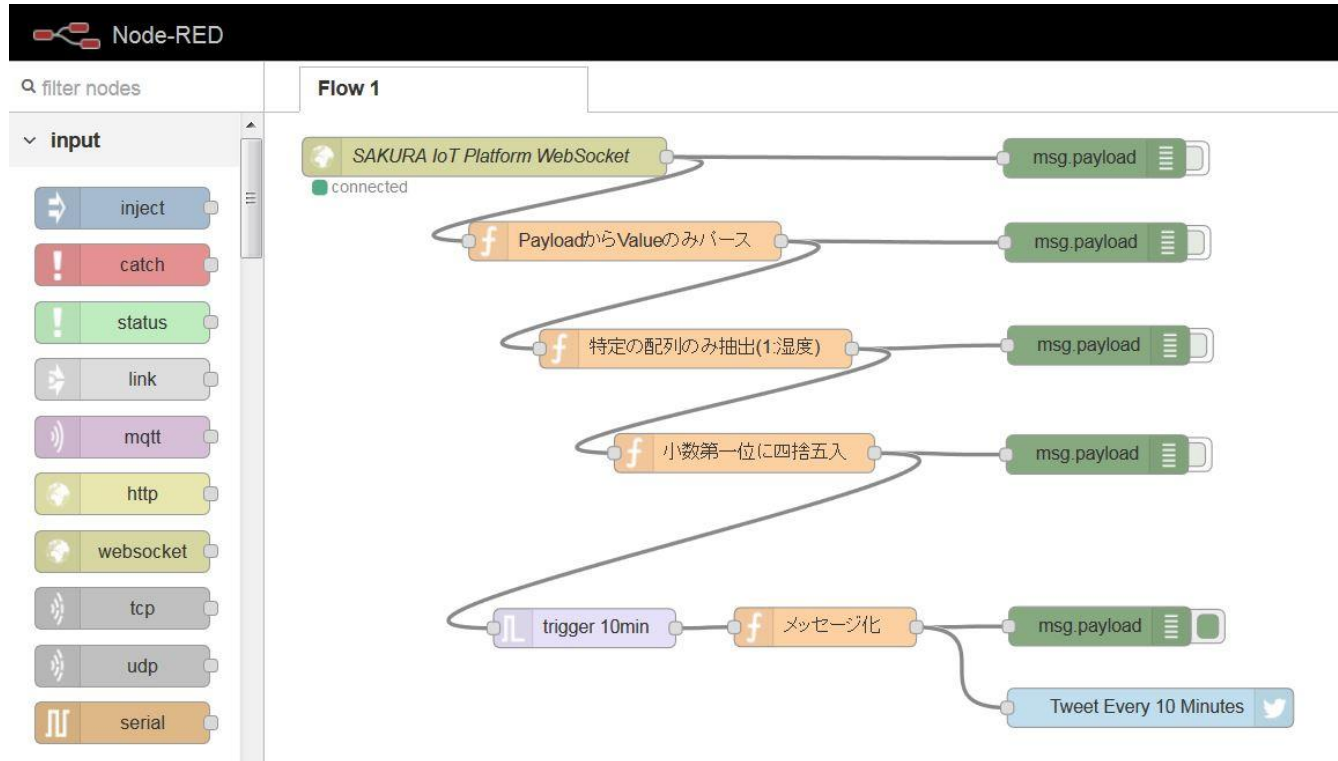


フローに問題がない場合、Websocketノード下部に「connected」と表示され、コンソールのdebug内にプラットフォームから取得したJSONデータを確認できます。Debugノード右端の緑マークをクリックするとdebugへの表示が停止されます。

The screenshot shows the Node-RED web interface. On the left, the 'input' palette contains various nodes, including 'websocket'. In the center workspace, a flow named 'Flow 1' contains a 'sakura-websocket' node (green) with a 'connected' status indicator below it, followed by a 'msg.payload' node (green). A red box highlights the 'sakura-websocket' node and its 'connected' status. On the right, the 'debug' console is open, showing a list of messages. A red box highlights the 'debug' tab and the first message, which contains a JSON payload. The JSON payload is as follows:

```
2016/8/22 22:23:02 [redacted]
msg.payload : string [235]
{"module":"[redacted]","type":"channels
06-22T13:22:58.040119829Z","payload":
{"channels":
[{"channel":0,"type":"f","value":26.447144},
{"channel":1,"type":"f","value":48.199463},
{"channel":2,"type":"I","value":113}]}}
```

WebSocketでデータ入手 → 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

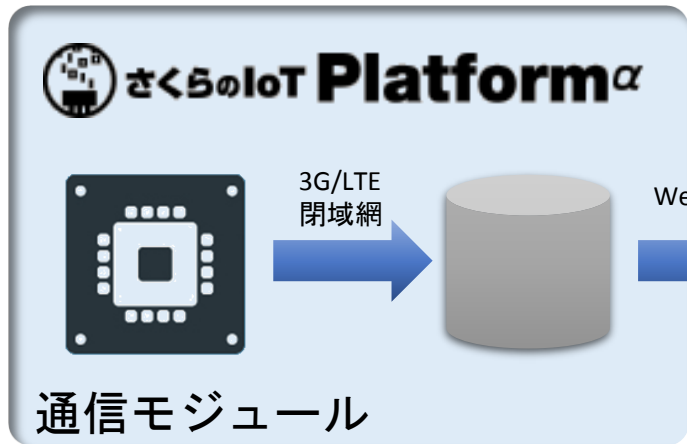
さくらインターネットのハンズオンで湿度情報を取得中！ただ今の現地湿度は56度だよ。β版もよろしくね！ #さくらのIoTPlatform #さくらインターネット



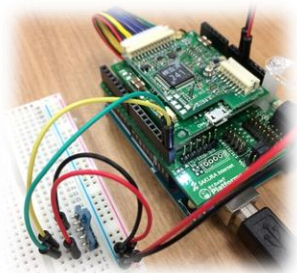
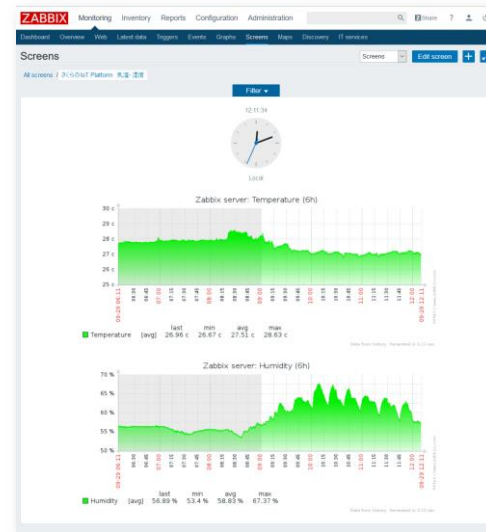
法林浩之(バースト用) @hourin_burst · 11月16日

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 secondary 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 trail: 'すべてのホスト / Zabbix server' followed by a status '有効' and several filters: ZBX, 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) with the value 'Temperature'; 'タイプ' (Type) with a dropdown menu showing 'Zabbix-ラッパー'; 'キー' (Key) with the value 'sakura_iot_temp'; 'データ型' (Data type) with a dropdown menu showing '数値 (浮動小数)'; and '単位' (Unit) with the value '°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://api.sakura.io/ws/v1/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";
```

テスト WebSocket

名前

テスト

Token

[Redacted]

WebSocket

wss://api.sakura.io/ws/v1/

[Redacted]

受信データ

時刻	モジュール	チャンネル	型	値
----	-------	-------	---	---

< 戻る

削除する

+ 保存する

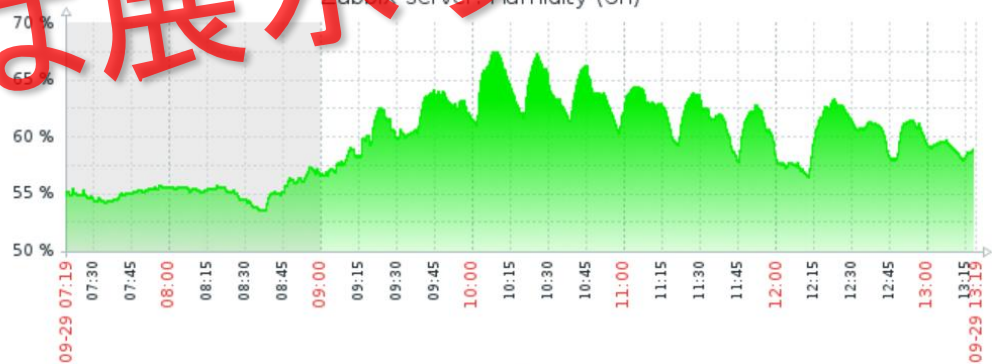


Zabbix server: Temperature (6h)



■ Temperature [avg] 27.89 c last 27.89 c min 26.67 c avg 27.43 c max 30.75 c

Zabbix server: Humidity (6h)



■ Humidity [avg] 58.67 % last 58.67 % min 53.4 % avg 59.58 % max 67.37 %

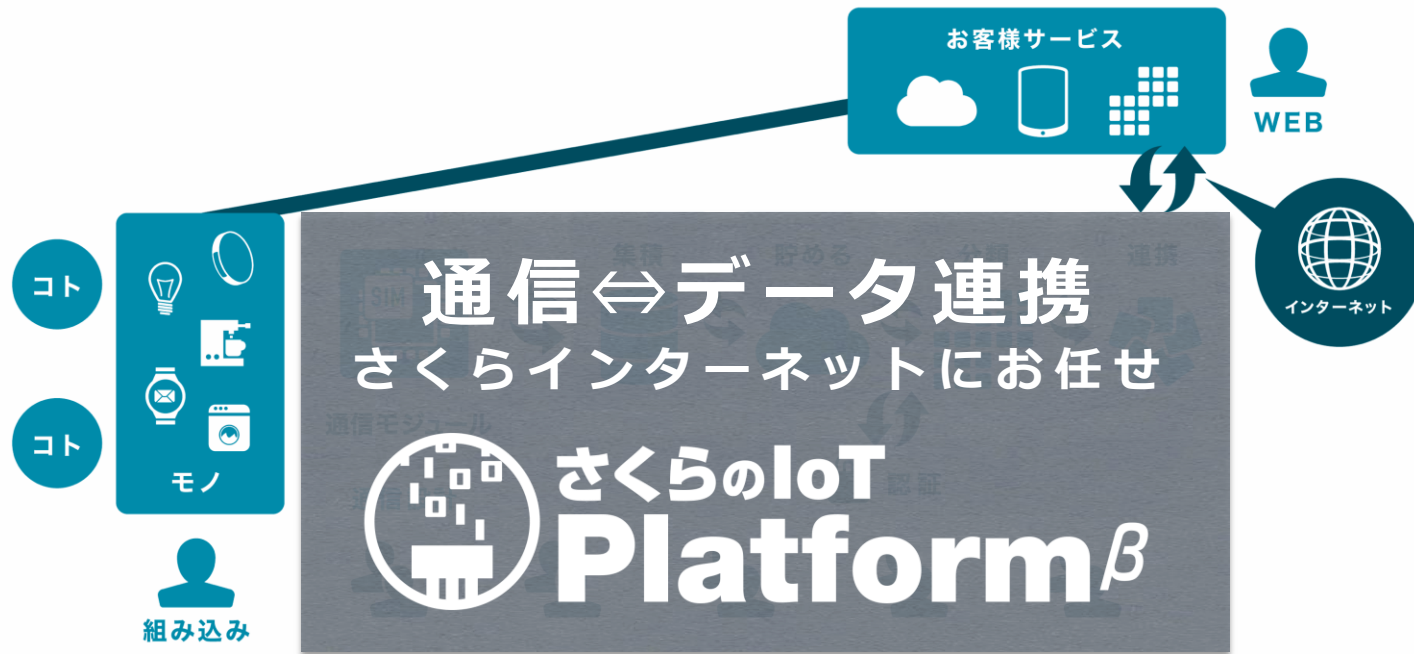
Data from history. Generated in 0.10 sec.

実機は展示ブースで!!!

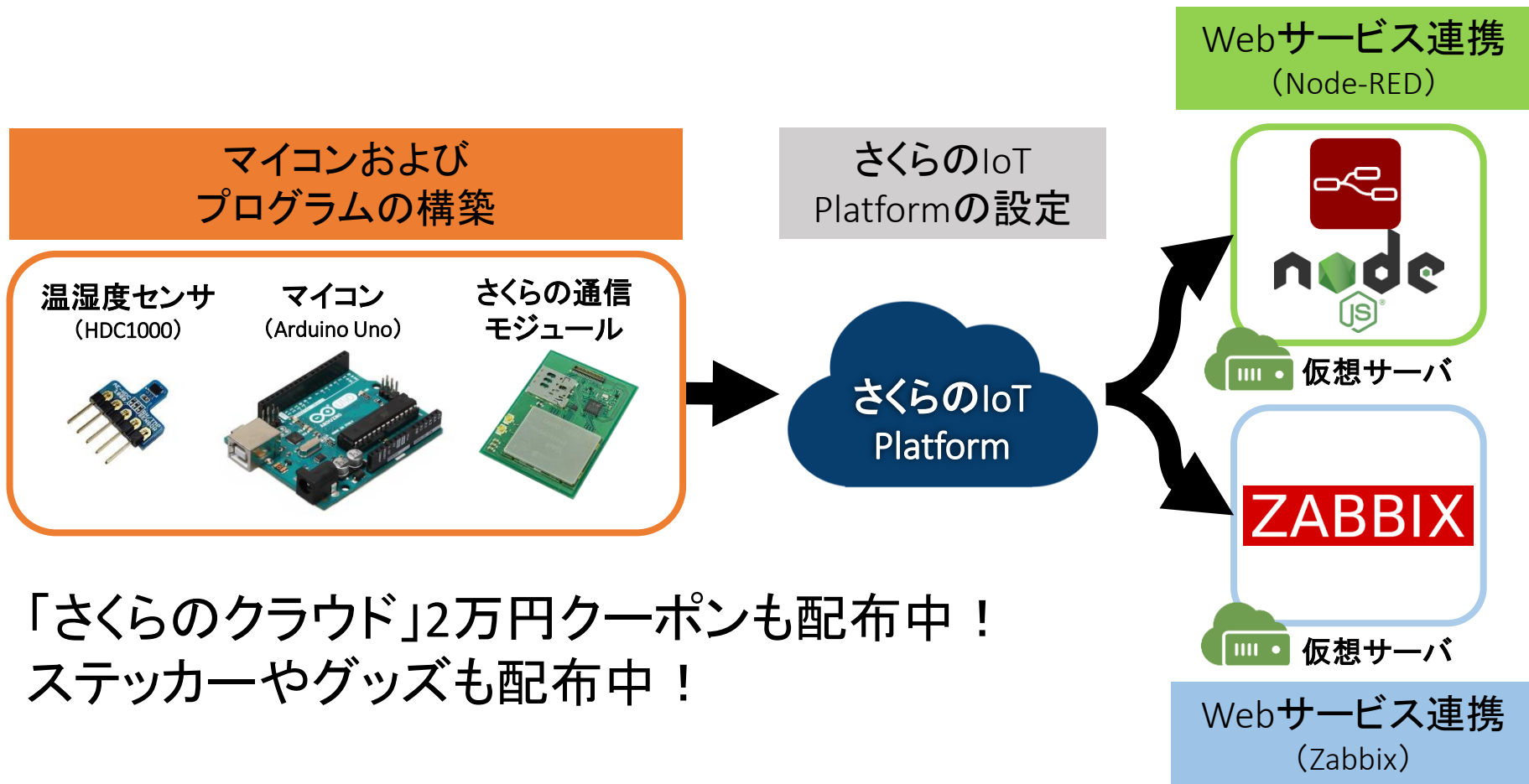
まとめ



- 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



「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！



- さくらのイベントを全国で開催したい！
 - さくらのIoT Platformのハンズオン
 - さくらのクラウド / 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