# つっぱり棒 × Raspberry Pi × 3GPI × Nifty Cloud Mobile Backend ワークショップ・ハッカソン

**スケジュール**

６月１１日（土）
11:00 ～ 11:30 挨拶＋趣旨説明
11:30 ～ 12:30 ワークショップ１
12:30 ～ 13:30 昼休み
13:30 ～ 16:00 ワークショップ２
16:00 ～ 16:30 運用テスト・アイデア共有
17:00 撤収

６月１２日（日）
10:00～15:00　黙々と実装
15:00～15:10　Knowledge Connector（成果物のまとめサイト）チュートリアル
15:10～16:00　発表資料作成
16:00～17:00　発表会
17:30　撤収

**参加費　500 円**

**会場　Connectly Lab (リノベる株式会社)**

**配布物**
IoT スターターキット（メカトラックス提供)、SORACOM Air SIM (ソラコム提供)、温度センサーLM75B、ピンヘッダ、ジャンパワイヤ４本、はんだ

**適宜使用可能なもの**
つっぱり棒アタッチメント　（無料）
スイッチサイエンスエイドステーション　（有料）
会場には平安伸銅工業の突っ張り棒が設置されています。

# 内容



主催
　おうちハック同好会　https://www.facebook.com/groups/ouch.hack/

協力
　神奈川工科大学　http://www.kait.jp/index2.php
　平安伸銅工業株式会社　http://www.heianshindo.co.jp/
　メカトラックス株式会社　http://www.mechatrax.com/
　株式会社ソラコム　https://soracom.jp/
　ニフティ株式会社　https://www.nifty.co.jp/
　インターネットアカデミー　https://www.internetacademy.jp/
　LOD チャレンジ実行委員会　http://lodc.jp/

　やまざき　はるき　（スパイスボックス）
　大和田　茂　（ソニーCSL）

## 第一部　メカトラックス社 IoT スターターキットを組み立てる

3GPI を装着したラズパイ

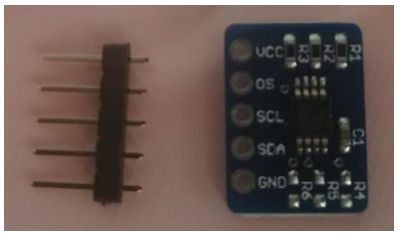
今日の目標は、どこにでも設置できる温度センサーロガーを作ることにあります。インターネットへの接続は 3G 回線を用いることで、場所の自由度が飛躍的に上がりますので、そのためのラズパイ用モジュール（シールド）である 3GPI を用います。3GPI は、設定が非常に楽だということが特徴です。ハードウェアをつなぎさえすれば、ソフトウェアの設定は一切不要です。さらに、ラズパイ最初にして最大の関門、「OS のインストール」という作業をスキップするために、あらかじめ Raspbian が書き込まれた SD カードまでついてくるんです！
今回製造元のメカトラックス社からラズパイ込みの体験キットを、ソラコム社から SORACOM Air SIM をお借りできましたので、最初にこれを組み立てます。装着方法はこちらの記事を参照してください。
http://qiita.com/sgrowd/items/40479e0d1473ebc4a0de

※今回カメラは使いません。また、ラズパイ本体は記事で言及している 2 ではなく最新の 3 ですが、装着方法は変わりません。また、SIM も micro SIM ではなく標準 SIM です。
SIM のつけ方に若干コツがあります。ライブでお教えしますので、壊さないよう注意してつけてください。

## 第二部　378 円の温度センサーLM75B をラズパイにつなぐ

ラズパイに温度センサーをとりつけます。スイッチサイエンスから LM75B という温度センサーが格安に提供されていますのでこれを使います。

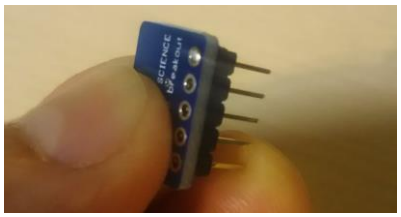

まず、センサーに配線のためのピンヘッダーをとりつけます。これもスイッチサイエンスから「普通のピンヘッダ１０本セット」というのが提供されているので、これを用います。５つだけ切り離して、はんだ付けします。



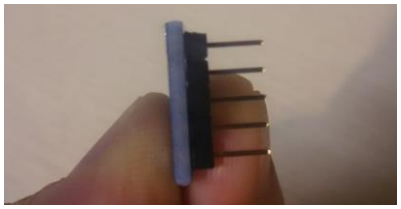

IC が乗っかっている面側（表側と呼びます）にピンヘッダをつけたいのですが、安定が悪いので私は次のようにしてみました。（ベストな方法かどうかはわかりません）

まず、基盤を裏っかえしにして、端っこの穴の上にちょっとだけ半田をつけます。

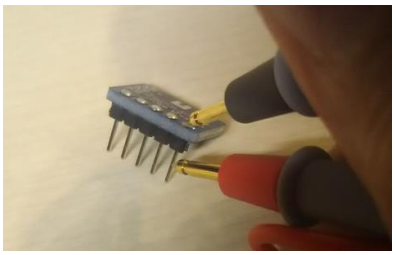
次に、ピンヘッダを表側から差し込み、半田がついていない穴に対応するピンヘッダからを指で軽く押さえつけ、先ほどつけたはんだを溶かして、ピンヘッダを奥まで差し込みます。半田がついているピンを押し込むと熱いので気を付けてください。



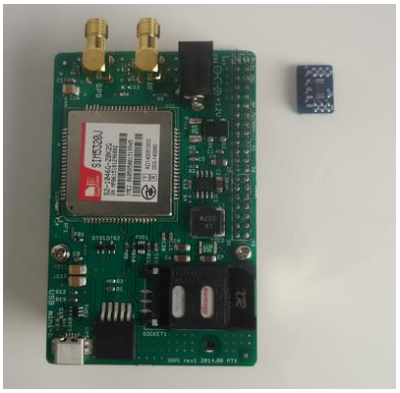
これでピンヘッダは固定されるので、残りの穴も裏側からはんだづけして固定します。ピンヘッダの足は比較的短いので、穴の奥まで半田が流れ込むよう、長めに（２～３秒）熱して十分に半田を溶かしてください。



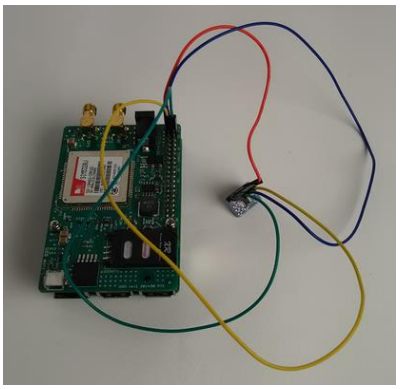
できれば最後にきちんとはんだ付けできたかどうか導通チェックします。
もう半田ごては使わないので、電源を抜いておいてください。



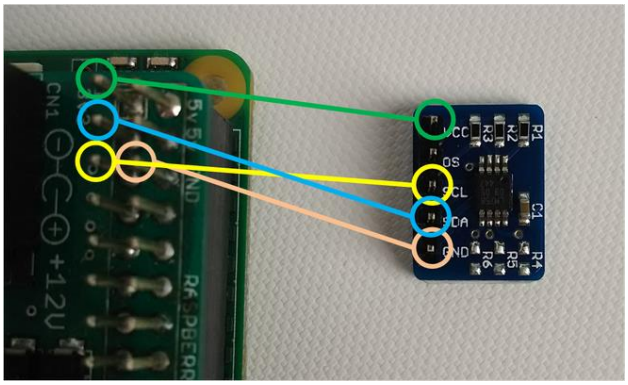
次に、ラズパイと温度センサーを接続します。
右の図のようにラズパイ/3GPI と温度センサーを並べてください。温度センサーは表向きにし、左端にピンヘッダがくる形です。



ラズパイとは I2C (アイスクエアシー)という方式で通信を行うので、対応する端子をつなぎます。それにはジャンパワイヤというものを使います。これもスイッチサイエンスから「普通のジャンパワイヤ (メス～メス)」というのが売っているので、それを使います。



つなぎ方は下図の通りです。つなぐと右図のようになります。




これでハードウェアの設定は終わりです。

## 第三部　温度センサーの値を読む

まず、自分が使うラズパイの IP アドレスを確認します。正直言って、同じネットワークにラズパイが多数接続されているとき、自分の目の前にあるラズパイの IP アドレスがなんであるかを知るのは難しいです。よく使われるのは、適当な IP アドレスにとにかく ssh でログイン（次に説明）してみて、その後にラズパイの LAN ケーブルを抜きます。それでシェルが固まったら当たりです。もう一つ使われるのは、同時に複数のラズパイをネットワークにつなぐのではなく、ひとつづつ増やしていきます。ネットワークをスキャンして、新たに出現した IP アドレスが新たにつないだラズパイです。ネット上のラズパイを探すには、私は Adafruit の Pi Finder というものを使っています。https://github.com/adafruit/Adafruit-Pi-Finder

いずれにせよ IP アドレスがわかったら ssh でラズパイにログインします。Windows の場合は putty や ttssh がおすすめです。ユーザー名は pi 、パスワードは raspberry です。
コマンドラインから ssh コマンドで入る場合は　ssh pi@[IP Address] となります。

次に、ラズパイのコマンドラインで必要なものをインストールします。（実行に時間がかかるため、数日前に実施済です）

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y i2c-tools
```

では、温度センサーの値を読みましょう。温度センサーとの通信には I2C (アイスクエアシー)という方式を使いますが、ラズパイでは最初の段階で無効になっているので有効にします。

```
sudo raspi-config
```

出てくるメニューから
「Advanced Options」⇒「A6 I2C」⇒「Yes」
と進み、I2C を有効にします。

一度再起動しましょう。

```
sudo reboot
```

３０秒ほど待ってもう一度ログインし、次のコマンドを打ってみます。

i2cdetect –y 1

以下のような表示になりましたか？

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

48 と表示されていれば、正常にセンサーが認識されています。表示されなければ、センサーの接続が間違っているか、接触不良が考えられるので確認してください。

センサーの値を読み取るコマンドは以下の通りです。

i2cget -y 1 0x48 0x00 w

私の場合は

0x801b

と表示されました。
最初についている 0x は 16 進数という意味です。また、その後の値は２桁ごとのまとまりで小さい順に並んでいます。従って、この値は

１６進数の 1b80

を表しています。1b80 は 10 進数だと 7040 です。このセンサーの場合、256 で割ると正しい摂氏温度になります。7040/256 = 27.5 ですので、27.5℃だとわかります。電卓ソフトな

どを使って確かめてください。

センサーの値を読み取り、摂氏温度で表示するスクリプトを書いてみましょう。コマンドラインで使えるテキストエディタとしては、初めての方には nano がおすすめです。

nano gettemp

でエディタを開き、次の内容のファイルを作ってください。
=======
#!/bin/sh

VAR=`i2cget -y 1 0x48 0x00 w | awk '{print "0x" substr($0,5,2) substr($0,3,2) }'`
echo $((VAR)) | awk '{printf ("%f",$VAR_D/256)}'
=======

nano でファイルをセーブするには[ctrl-x] , y , [Enter]

で gettemp というファイルができるので、このファイルに実行属性をつけます。

chmod u+x gettemp

こうしておいて、

./gettemp

と打ちこむと、現在の気温が小数で表示されるはずです。
先程、手作業で計算した結果と近い値になりましたか？

## 第四部　ニフティクラウド mobile backend を使ってみる

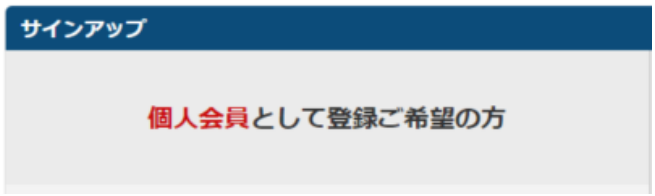
さて、次はクラウドへのアクセスを試してみましょう。それには、NIFTY Cloud mobile backend というサービスを用います。無料で使えます。

まず、Nifty Cloud Mobile Backend のページを開きます。

http://mb.cloud.nifty.com/

右上の「無料登録」を押します。


個人会員として登録をします。




あとは指示に従ってください。

登録が終わったら、ダッシュボード一番左上のメニューから、「＋新しいアプリ」を選び、新規アプリ作成を行います。名前は何でもよいのですが、ここでは RasPiTemp としています。

すると、アプリケーションキーとクライアントキーが表示されるので、ここはとりあえずこのままにしておきます。


ラズパイ上で work/フォルダに移動します。（すでにフォルダが作成されています）

cd work

NIFTY Cloud Mobile Backend のライブラリを落としてきます。

sudo apt-get install npm
npm install ncmb
　(本ワークショップでは実行済みですのでやらないでください)

ここにある、ncmbsample.js というファイルをテキストエディタで開きます。

nano ncmbsample.js

=======
var NCMB = require("ncmb");
var ncmb = new NCMB("YOUR_APPLICATIONKEY","YOUR_CLIENTKEY");

var TestClass = ncmb.DataStore("TestClass");
var testClass = new TestClass();

```
testClass.set("message", "Hello, NCMB!");

testClass.save()
        .then(function(){
                console.log("message is saved.");
        })
        .catch(function(err){
                console.log(err.text);
        });
=======
```

三行目の
YOUR_APPLICATIONKEY
YOUR_CLIENTKEY
となっているところを、ブラウザで表示されている値に変更します。

変更したら、ファイルをセーブしてエディタを抜けます。

nano でファイルをセーブするには[ctrl-x] , y , [Enter] です。

ブラウザの方はもうキーをコピーしたので、OK ボタンを押してダッシュボードを表示しておきます。
そして、ラズパイのコマンドラインで、ncmbsample.js を実行します。

```
node ncmbsample.js
```

すると、

```
message is saved.
```
と表示されるはずです。

これでクラウドにデータがアップロードされました。ブラウザのダッシュボードから「データストア」⇒「TestClass」と進むと、message という行に「Hello, NCMB!」という文字列が入っているのがわかると思います。

## 第五部　温度センサー値をクラウドに連続的にアップロードする

さて、次はいよいよセンサーの値をクラウドにアップロードしましょう。そのために、先ほどの ncmbsample.js を少し変更します。まず ncmbsample.js をコピーして temp.js を作りましょう。（すでに temp.js がある場合は実行しないでください）

cp ncmbsample.js temp.js

そして、nano で temp.js を開き、以下のように書き換えます。(YOUR_APPLICATIONKEY と YOUR_CLIENTKEY は適切なものに変更しておいてください。

=======

```
var NCMB = require("ncmb");
var ncmb = new NCMB("YOUR_APPLICATIONKEY","YOUR_CLIENTKEY");

// 外部プログラムを実行する準備
var exec = require('child_process').exec;
var TempLog = ncmb.DataStore("Temperature") ;

// i2cget を外部プログラムとして呼び出します。
exec("/usr/sbin/i2cget -y 1 0x48 0x00 w",function(err,stdout,stderr){
        // 出力を取り出します。
        var hexnum = '0x'+stdout.substring(4,6)+stdout.substring(2,4) ;

        if (err) {console.log(err); return; }
        var temp = parseInt(hexnum)/256.0 ;
        // 計測した温度値が temp に入っています。

        var tempLog = new TempLog() ;
        tempLog
                .set("date",(new Date()))
                .set("temperature",temp)
                .save()
                .then(function(){
                        console.log("message is saved.");
```

```
                })
                .catch(function(err){
                        console.log(err.text);
                });
});
=======
```

これを一回実行してみましょう。

node temp.js

「message is saved.」と表示されましたか？表示されれば、温度データがタイムスタンプと共にクラウドにアップロードされています。ブラウザのダッシュボードから「データストア」の Temperature を表示して確認してみましょう。

では最後に、このプログラムを自動的に定期実行するようにしてみましょう。それには cron というソフトウェアを用います。crontab というコマンドで cron の設定ファイルを開くことができます。

crontab -e

エディタがいくつか選べるので、2 の nano を選んでください。単に Enter を押しても nano になります。
最後の行に

```
*/1 * * * * /usr/bin/node /home/pi/work/temp.js
```

を追加します。これで一分に一回、自動的にプログラムが起動されます。最初の*/1 というのが、一分間隔という指定になるので、例えば 30 分間隔でよければ*/30 となります。

ブラウザで Temperature データベースを表示し、たまに右上の更新ボタンを押すと、データが自動的にアップデートされていきますので確認してください。

## 第六部　見える化

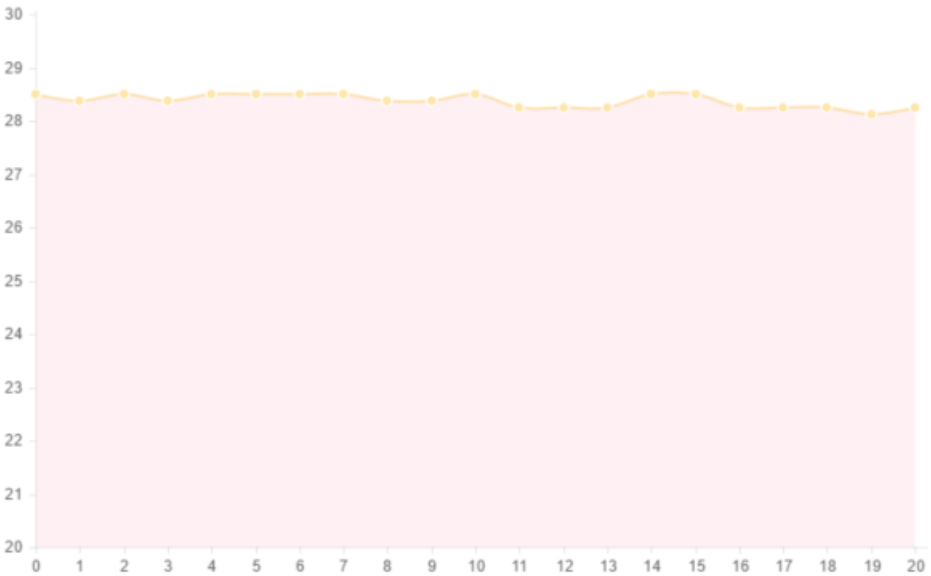
温度データがアップロードされるようになったら、今度はグラフで表示したいですね。ニフティの川原さんが、そのためのプログラムを準備してくださっています。

PC で
https://github.com/hounenhounen/ChartNcmbJs
を開き、緑色の「Clone or download」から「Download ZIP」をクリックして zip ファイルを落とし、適当なところに展開してください。（git に慣れている方は git clone しても結構です）

この index.html 内 24 行目の YOUR_APPLICATIONKEY と YOUR_CLIENTKEY を、自作のアプリのアプリケーションキーとクライアントキーに変更してセーブしてください。

さらに、その時の温度にもよりますが、45 行目 scaleStepWidth を 1.0 に、次の行の scaleStartValue を 20.0 あたりにしておくと、目指す温度が表示領域に入りやすいかもしれません。

こうしておいて、index.html をブラウザで開けば、温度のグラフが表示されます。



## 第七部　設置

ラズパイには 3GPI が装着されているので、イーサネットを外してもデータはアップロードされ続けます（再起動が必要かもしれません）。また cron の設定も完了しているので、電源を入れさえすれば自動的にデータがアップロードされていきます。
※今回の SIM 通信料はメカトラックスさんの厚意でご負担いただいていますのでご留意ください。

つっぱり棒アタッチメントを使って、Connectly　Lab に設置されたつっぱり棒に装着してみましょう！つっぱり棒アタッチメントは大和田が適当に作ったやわなものなので、注意して丁寧に扱わないとすぐに割れます。ご了承を。

## 第八部　ハッカソンと成果物のまとめ　（二日目用）

二日目は、ここまで学んだ成果を変更したり、新たに考えたりして役に立つ作品を作るハッカソンです。特にファシリテーション等は行いませんので、ご自分で作りたいものを各自作ってください。メンバーを募ってチームで開発しても結構です。知識を共有しあいながら、楽しく開発していただければ幸いです。

１６時から成果発表会を行います。特に評価はしませんので温かく盛り上がりましょう。ただ、成果をまとめる場所として、Knowledge Connector というサイトを使っていただきたいと思います。

http://idea.linkdata.org/all


Knowledge Connector はハッカソン成果をまとめ、新たなアイデアを啓発したり、ビジネスマッチングを行うこともできるサイトです。すでに 5/28 に開催したつっぱり棒アイデアソンの成果が多数アップされているのでご覧ください。

http://idea.linkdata.org/ideas?tag=%E3%81%A4%E3%81%A3%E3%81%B1%E3%82%8A%E6%A3%92

開発物の登録の仕方は、ここでアプリ情報を「新規作成」して、タグとして「おうちハック」と「生活デザインコンテスト SEJ2016 in Osaka」の二つをつけ、さらにライセンスとして

「CC-BY」を設定することです。今回は追加で「つっぱり棒」というタグも追加してください。

このサイトへの登録に加え、A2 サイズの pdf で概要をまとめ、

submit2016@lifedesign.tech

までお送りいただくと、「生活デザインコンテスト＃３」というコンテストへの応募をすることもできます。
このコンテストでは、おうちハックを体験できる入門用コンテンツ（ドキュメンテーション）の募集を行っています。内容は下記をご覧のうえ、奮ってご参加いただければ幸いです。締め切りは６月２４日です。

http://lifedesign.tech/