ボクにもわかる

IchigoJam BASIC で作る IoT システム

～LTE／Wi-Fi／LoRaWAN 対応～

伝統的 BASIC 言語で小規模 IoT システムを低コストで開発

教育向けコンピュータ IchigoJam を使って，家庭やホビー，実験，小規模事業用の IoT システムを作成します.

国野 亘

https://bokunimo.net/15/

ダウンロードしていただき，ありがとうございました.

はじめに

本書では，IchigoJam に LTE 通信機能，Wi-Fi 通信機能，LoRaWAN 通信機能を追加し，IoT システムを構築する方法について説明します。

第 0 章では，教育用コンピュータ IchigoJam を使ったシステムの特長や役割，各通信方式の違いなどを解説します.

第 1 章では，モバイル LTE 通信に対応した sakura.io モジュールを使った IoT 機器と，クラウドとの連携方法について，第 2 章では，WiFi 通信に対応した MixJuice を使った，Wi-Fi 通信方法，第 3 章では，LoRaWAN を使った小規模システムの製作例について紹介します.

家庭用やホビー用，実験用システム用，小規模事業用の IoT システム構築の参考になれば，幸いです.

ボクにもわかる IchigoJam BASIC で作る IoT システム
～ LTE／Wi-Fi／LoRaWAN 対応 ～

〈国野 亘〉

目次

0章 伝統的 BASIC 言語で小規模 IoT システムを開発する

商店街で八百屋と文具屋を営む店主が，閑散期に一人で二店舗を営業する場合の支援システムを考えてみましょう．

例えば，各店舗へ来店を知らせる IoT 押しボタンを設置すれば，各店舗の来店状況を把握できるようになります．しかし，押しボタンだと，店主に遠慮して押してくれないかもしれません．

人感センサを組み合わせれば自動通知が可能になります．さらに，音声合成スピーカを組み合わせて，「ただいま文具店に居ます．御用の際は押しボタンを押してください」といったメッセージをお客に伝えることもできるようになり，徐々に店番システムへと発展させることも可能です．

本稿では，こういった独自の小規模 IoT システムを構築する方法について説明します．店主に限らず，家庭用やホビー用，実験システムの試作といった様々な小規模システムに活用できるでしょう．

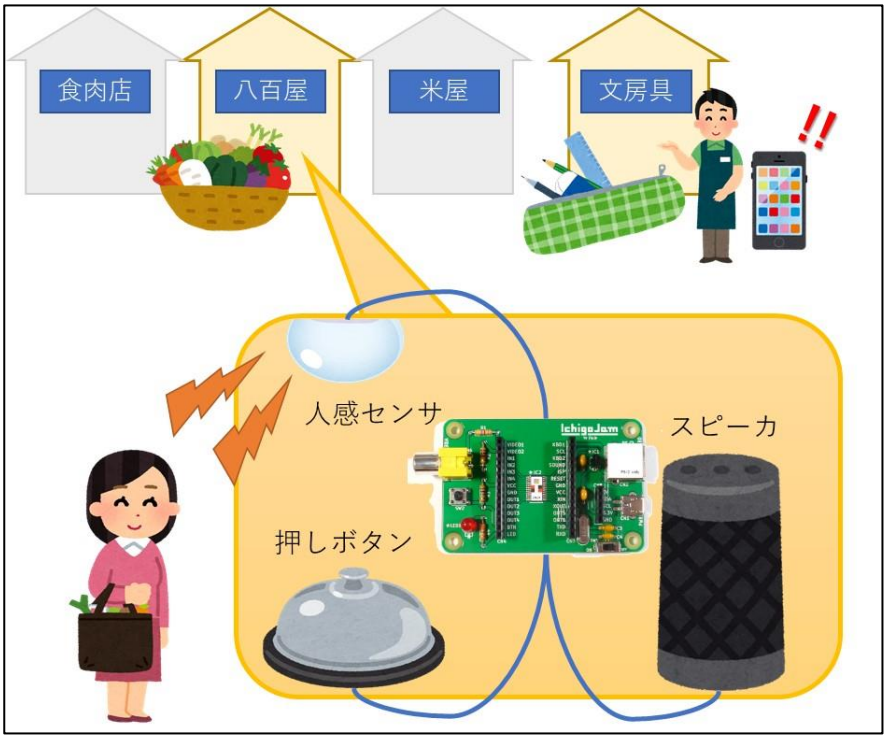

図 0-1
IoT システムの一例
八百屋と文具店を営む店主（架空）が，閑散期に一人で二店舗を営業するために開発した IoT システムの一例（第 1 章で紹介）

教育用に開発された技術を実用的に活用する

IoT 技術の応用範囲は，その実装形態が多種多様で，使い方に応じた専用システムが数多く登場すると期待されています．例えば，押しボタン情報やセンサ情報をクラウドへ送信することで，従来は手作業で収集していた情報収集の自動化が可能になります．IoT デバイスは，24 時間 365 日，あちらこちらの場所で働き続けることができるので，様々な分野で変革をもたらされると言われています．

すでに大手 IT 企業が実用化している Wi-Fi 搭載 IoT 押しボタンの一例を写真 0-1 に示します．この IoT 押しボタンを押すことにより，あらかじめ設定しておいた飲食品や消耗品などの商品の注文を行うことが出来ます．近年，こういった大手企業による IoT への投資が目立ってきていますが，同じような開発手法で小規模システムの開発を行うと，1 台あたりの開発コストが高くなってしまいます．

そこで本稿では，教育用やホビー用として開発された技術を利用し，低コストの小規模 IoT システムの実現方法を紹介します．モジュール化された最新技術を簡単に実現することが出来，開発コストの削減が出来るからです．

具体的には，パソコンなしですぐに IoT デバイスの開発が可能な jig.jp 社（B Inc.社）の IchigoJam を使って，IoT 押しボタンや，IoT 人感センサ，IoT ディスプレイ端末，IoT 音声アナウンス端末，IoT 照度センサ，IoT 温湿度計，IoT バーコード・リーダといった IoT デバイスを製作します．また，スマートフォンや他のクラウドサービスと簡単に連携するための IFTTT や Node-RED についても紹介します．

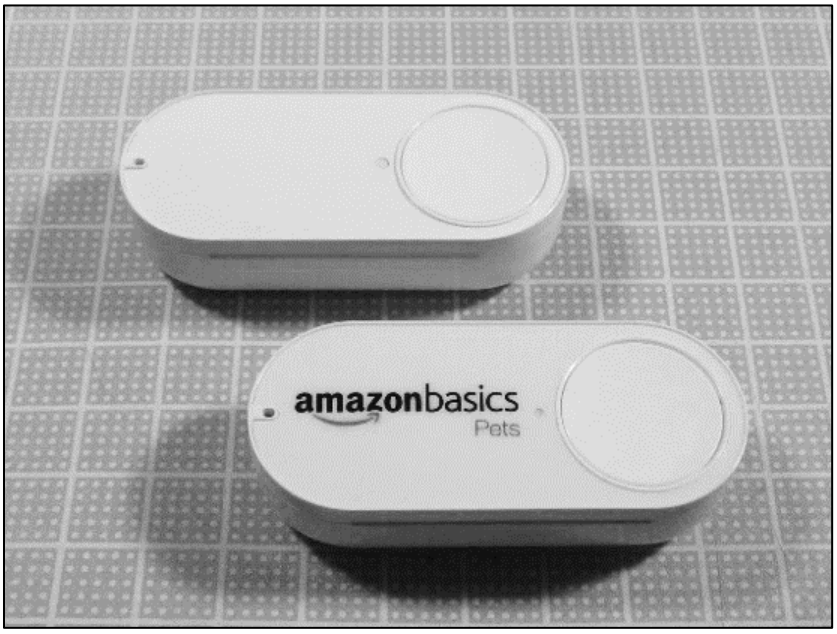

写真 0-1 IoT 押しボタンの一例
ボタンを押すと，予め設定しておいた商品が注文できる IoT 押しボタンの一例．本稿 1 章～3 章で通信方式の異なる IoT 押しボタンを製作する

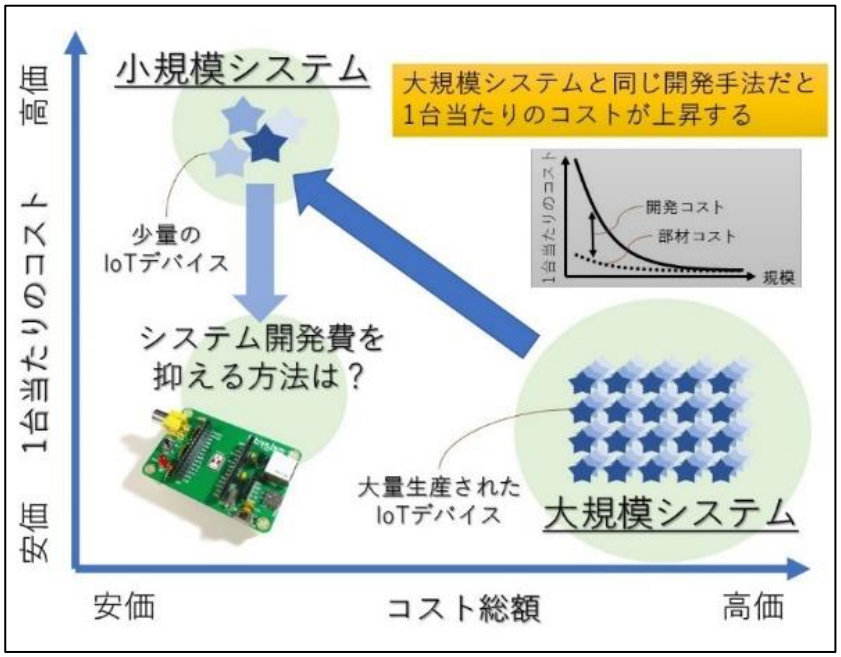

図 0-2 大規模システムと同じ開発手法で小規模システムを開発した場合のコスト比較
小規模システムでは 1 台当たりの開発コストが高額となるので，教育用に開発されたマイコン・ボードで IoT システムの開発費を抑える

IoT 向けに利用可能なマイコン・ボードの比較と役割

教育用に開発されたシングル・ボード・コンピュータやマイコン・ボードとしては，すでに Raspberry Pi や Arduino が世界中に広まっており，これらを使った IoT の実用例も，多く紹介されています．

Arduino UNO は，省電力で低価格な 8 ビットの汎用マイコンを搭載したマイコン・ボードです．組み込み用マイコンのプログラミング学習用に開発されました，単体でプログラムの開発が行えるシン

グル・ボード・コンピュータとは異なり，Windows などが動作する通常のパソコンでプログラムを作成し，コンパイルしてからマイコン・ボードに転送する必要があります．また，純正の Arduino UNO は，マイコン周辺回路の消費電力が大きく，乾電池による長時間駆動を行うには，基板を自作するか，Arduino Pro シリーズなどのマイコン・ボードが必要です．

Raspberry Pi は，通常のパソコン並みの機能を 1 枚の基板に集約したシングル・ボード・コンピュータです．ワープロや表計算，プレゼン資料作成といった標準的なパソコンとして利用できるほか，映像や画像，音声といったデータを扱う処理や，AI 処理を学習する用途にも使われています．消費電力は通常のパソコンよりは低くいですが，Arduino UNO よりも高く，乾電池での駆動には向きません．IoT 用としては IoT サーバ, IoT ゲートウェイといった処理能力を要する目的で使用すると良いでしょう．

表 0-1　IoT で実用利用可能な教育用に開発されたマイコン・ボードの一例

ハードウェア仕様項目	IchigoJam S/T/U	Arduino UNO	Raspberry Pi 3
CPU	NXP LPC1114FDH28	Microchip ATmega328P	Broadcom BCM2837B0
コア	ARM Cortex-M0	AVR RISC MCU	ARM Cortex-A53 (4 コア)
ビット数	32 ビット	8 ビット	64 ビット
動作クロック	48 MHz	16 MHz	1.4 GHz
IO 電圧	3.3 V	5 V	3.3 V
GPIO（デジタル入出力）	最大 12 ch (LED,BTN 含む)	最大 16ch	最大 28 ch
ADC（アナログ入力）	最大 6 ch	最大 6ch	−
UART インタフェース	1 ch	1 ch	(1 ch)
I2C インタフェース	1 ch	1 ch	2 ch
USB インタフェース	−	○ USB デバイス	◎ USB ホスト
ビデオ出力	1 系統	−	1 系統 (HDMI)
キーボード入力	1 系統	−	USB 接続
マウス入力	−	−	USB 接続
電源電圧	DC 5V	DC 5V または 7〜12V	DC 5V
消費電力（実測）	0.09 W	0.26 W	1.5 W
節電動作時（実測）	0.002 W	0.2 W	—
節電待機時（実測）	0.0001 W	0.2 W	0.4 W
参考価格（税別）	1,500〜2,000 円	2,723 円	4,815 円
総合的な特長（筆者主観）	モニタとキーボードを接続することでシングル・ボード・コンピュータとしても使用できる．また，実用的な省電力動作も可能．一品ものに有利．	組み込み向け開発言語をサポートしているので，マイコンとしての機能をフル活用できる．量産製品に向けた試作や実証用に向いている．	教育用ボードとしては最もパワフルな部類で，一昔前のパソコンやサーバ並みの処理が可能．IoT サーバや IoT ゲートウェイの試作や実験用に利用できる．
写真			

- 1500 円から買えるシングル・ボード・コンピュータ IchigoJam

IchigoJam は，2014 年 4 月 1 日に福野泰介氏（jig.jp 創業者・現取締役会長）がテスト販売を開始したシングル・ボード・コンピュータです．基板とパーツのキットが 1,500 円（税別），パーツ実装済みのマイコン・ボードが 2000 円（税別）と安価に入手することが出来，ビデオ入力端子付きのテレビと，PS2 キーボード，AC アダプタを接続するだけで，プログラムの作成や実行が出来ます．

IoT デバイスは，クラウドや IoT サーバ，他の IoT デバイスなどとの通信を行うので，実際に使用する形態で，動作確認しながらプログラムの調整を行うことも多く，作成・修正したプログラムを即実行できるシングル・ボード・コンピュータは， システム開発効率の観点で大きな利点です．試作，実証，運用の各ステップでのプログラムの修正や調整も容易なので，小規模システムの開発期間や開発コストの削減が期待出来るでしょう．

- IchigoJam BASIC による省エネ運転機能搭載

IoT デバイスの中で最も数多く普及すると期待されている IoT センサは，測定したい場所に設置する必要があります．しかし，必ずしも容易に電源を供給できる場所とは限らず，乾電池や環境発電（エナジー・ハーベスト）が必要となる場合が多いでしょう．また，機器内部の発熱がセンサに影響することも多く，省電力化による発熱の抑制も重要です．

IchigoJam は，省電力な組み込み向け 32 ビット ARM コア Cortex-M0 を搭載しています．Raspberry Pi 3 の消費電力が約 1.5W であるのに対し，IchigoJam は約 0.09W と約 17 分の一の低消費電力です．さらに，IchigoJam BASIC が標準でサポートしている省電力機能を使うと，Raspberry Pi 3 の約 1000 分の 1 となる 0.001〜0.002W 程度で動作させることも可能です．乾電池の持ち時間が 1000 倍になると，例えば Raspberry Pi 3 で 3 時間しか動作できない乾電池であっても，IchigoJam では 3〜4 か月の動作を見込めます（通信モジュールの消費電力を除く）．

IoT デバイス向けプログラミング言語の比較

BASIC は，1980 年頃の 8bit パソコンの多くに標準搭載され，市場の拡大とともに成長を遂げた初心者用プログラミング言語です．技術者はもちろんのこと，ホビー用や小規模なコンピュータ・システムで広く利用されました．

しかし，16bit パソコンが普及しはじめると，画像や映像などの贅沢なコンテンツと，完成された汎用の市販オフィスソフトが広まり，1990 年頃から衰退しはじめました．他の言語に比べ，処理速度が遅かった点，取り扱える情報量が少なかった点，ソフトウェアのライブラリ化やモジュール化に劣る点で，次々に増加するマルチメディア機能に追従できなかったことなどが衰退の原因でしょう．

IoT デバイスでも，こういった高機能化が必要な分野もありますが，そうでない分野も多く存在します．そういった高機能が求められない分野での開発に，IchigoJam BASIC を利用すれば，小規模な専用 IoT システムを低コストで開発することができます．

- 大規模ソフト開発環境が（必ずしも）必要ではない IoT

IoT デバイスは，少ない情報処理能力，小さなプログラムで実現できることが多くあります．例えば，IoT 押しボタンや IoT 照度センサを実現するための IoT デバイス用プログラムは，①ボタンやセンサの状態を取得し，②取得結果を通信モジュールへ送信する処理を行います．①は IchigoJam BASIC がサポートしている BTN コマンドや IN コマンド，ANA コマンドなどで簡単に取得することが出来ます．②の通信機能は近年の技術進歩により，通信モジュール内に通信プロトコルスタックが実装されるようになり，通信モジュールごとに決められた処理コマンドを送るだけです．

このように，IoT デバイス内の全処理を一言で説明できる程度の少ない処理であれば，最新のプログラミング言語でなくても，伝統的な BASIC 言語で実現することが出来るのです．

- 初心者向け BASIC 言語で実用 IoT デバイス開発

そもそも，BASIC は初心者向けパソコン用言語として開発され，また IchigoJam BASIC はプログラミング教育用向けに教育現場で活用されています．このため，ソフトウェアの専門技術者でない人であっても，実用レベルのシステムが組める特長があります．BASIC が流行した 1980 年に 20 代だった現 60 代前後の方々による多品種・小規模・ソフトウェアの開発体制も考えられるでしょう．

- マイコン用開発言語 IchigoJam BASIC

IchigoJam BASIC は，2014 年に教育現場での活用が始まり，その後，2015 年 3 月に秋月電子通商での取り扱いが開始されたのをきっかけに，ホビー向け電子工作用としての活用も拡大しました．また，開発者の福野泰介氏は，日々，新たな活用方法を考案しつづけており，実際の教育現場や電子工作ユーザからの意見をもとに，様々なハードウェアやシステムと連携するための機能拡張を図っています．このため，組み込みマイコン用の開発用言語としての機能も充実してきており，基本的な IoT デバイスを簡単に実現することが可能になりました．

表 0-2　IoT デバイス開発用プログラミング言語の比較表

プログラミング言語	IchigoJam BASIC	Arduino 言語	C 言語
実行形式	インタープリタ型 （実機上で即実行可能）	コンパイラ型 （要 Arduino IDE）	コンパイラ型 （要開発環境）
言語上の制約	整数型（16bit）	汎用言語を拡張	汎用プログラミング言語
組み込みマイコン対応	◎ 標準対応	◎ 標準対応	○ 拡張により対応可能
対応マイコン数	△ 少ない （LPC 1114FN28 等）	○ やや多い （ATMega328P, ESP32 等）	◎ とても多い （各種組み込みマイコン）
汎用ライブラリ数	× あまり無い	◎ 豊富	◎ 豊富
専用ドライバ数	× あまり無い	◎ 豊富	△ 少ない
大規模プログラミング	× 非対応（1KB まで）	△ 可能だが向いていない	○ 可能
小規模プログラミング	◎ 向いている	◎ 向いている	○ 可能
試作・開発時のコスト	◎ 安価	◎ 安価	△～× 開発環境による
低消費電力動作	◎ 専用命令で容易に可能	○ 可能	△～○ マイコンによる
プログラム初心者	◎ 教育向け言語	◎ 教育向け言語	△ 技術者向け
開発者の世代傾向	10 代・40 代～60 代	20 代～40 代	30 代～50 代
総合的な特長（筆者主観）	プログラミング初心者が小規模な専用 IoT デバイスを安価に構築するのに向いている．	専門技術者並みのシステムを，最小限度のプログラミングの知識で実現可能．大規模システムの試作や実証実験用に利用可能．	プログラミングの知識やノウハウが必要であり，専門の技術者向け．大規模システムの商用量産機で使用される．

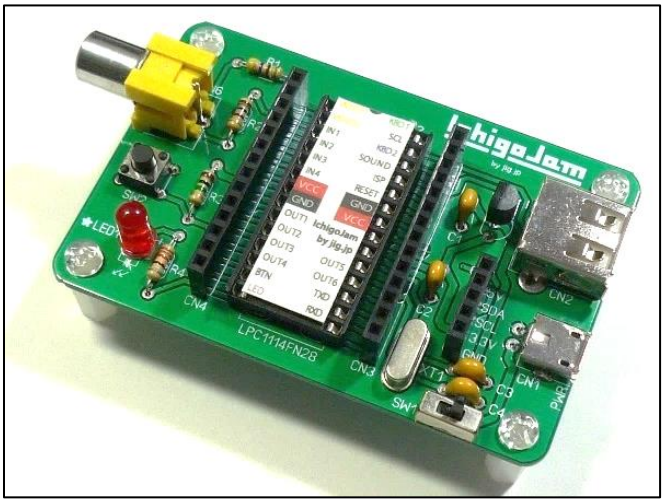

写真 0-2 BASIC を搭載した
マイコン・ボード IchigoJam T
ビデオ出力端子と，PS2 キーボード接続用 USB 端子，電源入力専用 Micro USB 端子を搭載した，IchigoJam BASIC 言語による
シングル・ボード・コンピュータ

IchigoJam マイコン・ボードの種類と違い

2018 年 8 月現在，販売されている純正 IchigoJam マイコン・ボードは，表 0-3 の 4 種類です．

初代 IchigoJam は拡張ボード用の CN3 と CN4 端子の間隔が IchigoJam S/T/U と異なるほか，テレビとの相性が悪いことも多いので，本書では IchigoJam S/T/U を推奨します．IchigoJam U の CN5 には，3.3V の電源や I²C インタフェース信号が接続されていません．IchigoJam U で CN5 を使用する場合は，SDA 信号を CN4 の IN3 端子へ，SCL 信号を CN3 の EX1 端子へ接続する必要があります．

IchigoJam BASIC のファームウェア（現在のパソコンの OS に相当）のバージョンは，生産時に最新のものが書き込まれます．PCN（https://pcn.club/products/）から購入すれば，最新バージョンが得られます．また，機器やソフトを用意して自分で更新することや．PCN が提供する IchigoJam バージョンアップサービス（有償）を利用して更新することも出来ます．

なお，第 1 章で使用する IchigoSoda は，IchigoJam S や T と互換性のあるマイコン・ボードです．

表 0-3　IchigoJam マイコン・ボードの種類と違い

マイコン・ボード		IchigoJam S	IchigoJam T	IchigoJam U	(初代) IchigoJam
マイコン	型番	LPC1114FDH28	LPC1114FDH28	LPC1114FN28	LPC1114FN28
	パッケージ	28 ピン TSSOP	28 ピン TSSOP	28 ピン DIP	28 ピン DIP
CN3 CN4 間隔		1000 mil	1000 mil	1000 mil	850 mil
PS2 キーボード端子		USB Type A	USB Type A	ミニ DIN 6 ピン	ミニ DIN 6 ピン
水晶クロック		12 MHz	12 MHz	12 MHz	なし(マイコン内蔵)
CN5 電源・I²C 端子		電源+I²C	電源+I²C	5V 電源のみ	なし
電源 IC	型番	MicrOne ME6209A	MicrOne ME6209A	SII S812C33A	SII S812C33A
	最大電流	250 mA	250 mA	50 mA	50 mA
	待機電流	3 μA	3 μA	1 μA	1 μA
過電流保護	5 V	なし	なし	200 mA	200 mA
	3.3 V	250 mA	250 mA	200 mA	200 mA
発売時期		2018 年 7 月	2016 年 7 月	2015 年 7 月	2014 年 7 月
出荷時ファームウェア		Ver. 1.2.3 以上	Ver. 1.2.1 以上	Ver. 1.0.1 以上	Ver. 0.8.2 以上
販売(2018年8月現在)		PCN	Aitendo (互換機)	秋月電子通商	–
写真					

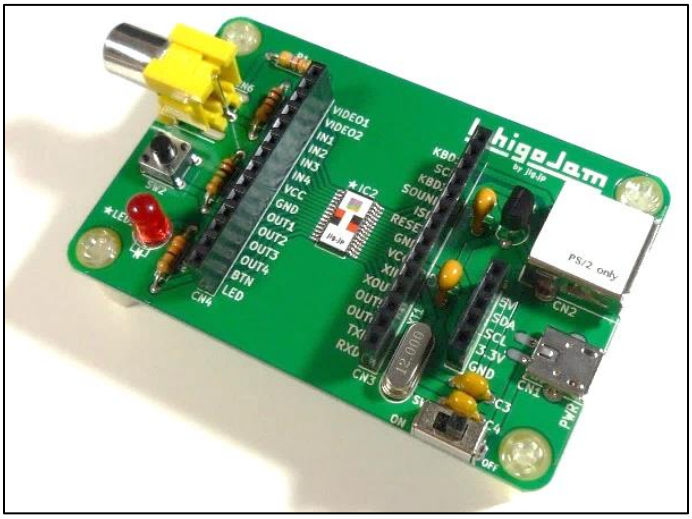

写真 0-3 BASIC を搭載した
マイコン・ボード IchigoJam S

IchigoJam T に比べて，ハンダ付け用のランドの大きさや，電源スイッチの強度が改善された IchigoJam マイコン・ボード

IoT 通信方式

通信方式の選択は，IoT システムを構築する際の実現性，利便性，コストに大きく影響します．本稿で用いる LTE 方式，Wi-Fi 方式，LoRaWAN 方式の特長の違いを表 0-4 に示します．

実現性については，実際に使用する場所で通信できることが必須要件です．通信エリアが広いほど，通信可能な可能な範囲が広まり，実現性や利便性に大きく寄与します．また，LoRaWAN については，基地局が少ないことや，実通信の通信速度が 20bps 程度にまで制限される場合があることに，注意が必要です．以下，本稿で使用する通信方式ごとの特長について，説明します．

表 0-4　本稿で使用する IoT 通信方式の違い

章番号･通信方式	第 1 章 LTE	第 2 章 Wi-Fi	第 3 章 LoRaWAN
通信エリア	◎ 広い	△ 主に家屋・施設内	△ 広いが基地局が少ない
通信速度	○ 早い（50Mbps）	◎ 早い（150Mbps）	× とても遅い（50kbps）
通信料金	△ 他方式に比べて高い	◎ 無料（ネット接続料別）	○ 普及すれば安価に
機器コスト	△ 他方式に比べて高い	◎ 安価	○ 普及すれば安価に
サービスの一例	sakura.io （月額 60 円）	お手持ちの Wi-Fi 環境	SORACOM LoRaWAN （Harvest 日額 5 円）
本書の製作例	IoT 押しボタン IoT 人感センサ IoT 照度センサ IoT ディスプレイ端末 IoT 音声アナウンス端末	IoT 押しボタン IoT 人感センサ IoT 照度センサ IoT 温湿度センサ	IoT 押しボタン IoT 人感センサ IoT 液晶ディスプレイ端末 IoT バーコード・リーダ

IoT 通信方式(1) LTE

スマートフォンなどで利用されている LTE 方式は，カバーエリアが広く，屋内はもちろん屋外での移動中の通信が可能なモバイル通信方式です．大手通信キャリアが提供するサービスは高価ですが，MVNO による格安 SIM や，IoT 専用サービスも登場するなど，低価格化が進んでいます．

そこで第 1 章では，さくらインターネット社が提供する sakura.io モジュールを使用し，LTE 対応の IoT 押しボタン，IoT 人感センサ，IoT 照度センサ，OLED（有機 EL）ディスプレイを搭載した IoT ディスプレイ端末，IoT 音声アナウンス端末を作成します．

写真 0-4 の IchigoSoda は，sakura.io モジュールを接続することが可能な IchigoJam 互換のマイコン・ボードです．IchigoSoda の裏面に sakura.io モジュールを取り付けるだけで，手っ取り早く試作することができます．他の IchigoJam に sakura.io モジュールを接続するには，コネクタの変換や信号レベルの変換，電源回路などが必要です．

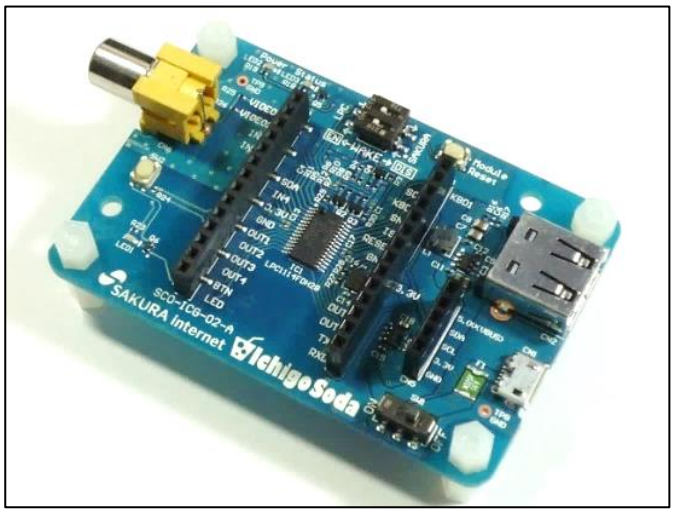

写真 0-4 sakura.io モジュールを搭載可能なマイコン・ボード IchigoSoda

IchigoJam BASIC を搭載し，モバイル回線 LTE 方式に対応した sakura.io モジュールを基板裏面に取り付けることができる

IoT 通信方式(2) Wi-Fi

Wi-Fi は，屋内などの無線 LAN 方式として，最も普及している通信方式です．インターネット回線のある家庭内や事業所内に IoT デバイスを設置する場合，新たな通信料がかからない点が最大の利点です．また，通信機器も比較的安価です．

第 2 章では，PCN 社が販売する Wi-Fi ネットワーク拡張ボード MixJuice を IchigoJam S/T/U へ接続し，Wi-Fi 対応 IoT 押しボタン，IoT 人感センサ，IoT 照度センサ，IoT 温湿度センサを製作します．各 IoT センサの設置場所には，AC アダプタが利用できない場合も多いので，IchigoJam と MixJuice のディープ・スリープ機能を使い，乾電池で長期間動作が可能な省エネ運転対応 IoT センサの製作方法についても紹介します．

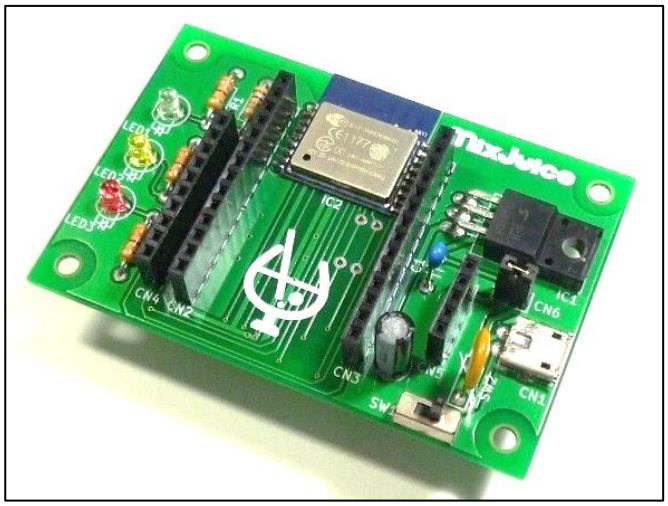

写真 0-5 Wi-Fi ネットワーク拡張ボード MixJuice

IchigoJam S/T/U に重ねるように接続することで，Wi-Fi ネットワーク機能を拡張することができる

IoT 通信方式(3) LoRaWAN

IoT 専用の通信方式として注目されている LoRaWAN 方式についても実験を行ってみました．第 3 章では，SORACOM 社が提供する LoRaWAN へ接続する IoT デバイスの構成方法と，プログラムの製作方法について説明します．また，バーコードで読み取ったコードを送信する IoT バーコード・リーダや，受信したメッセージを I²C 接続 LCD へ表示する IoT 液晶ディスプレイ端末の製作例，複数のシリアル機器の接続方法についても解説します．LoRaWAN に興味が無い人も，ぜひご覧ください．

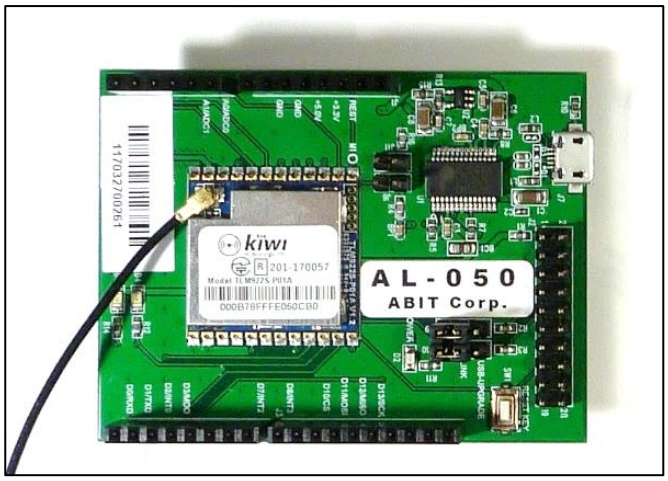

写真 0-6 LoRaWAN 開発ボード AL-050

SORACOM が販売する Arduino 用シールド形状の ABIT 製 LoRaWAN 開発ボード．識別情報が予め設定済みなので，SORACOM のコンソールで，受け取り確認のボタンをクリックするだけで，すぐに LoRaWAN サービスを受けられる

1章 sakura.io モジュールを使った LTE 対応 IoT 店番システムの製作

本章では，IchigoJam BASIC を搭載した IchigoSoda と，LTE 通信が可能な sakkura.io モジュールを用いて，IoT 押しボタンの製作例や IoT 人感センサ，IoT 温度センサの接続例，クラウドサービスの利用例について説明します．純正の IchigoJam S/T/U に sakura.io モジュールを接続して製作することも可能です（但し，コネクタの変換や信号レベルの変換，電源回路などが必要）．

IoT 押しボタンは，IchigoSoda 上のボタン SW2 を操作ときにボタン状態を LTE 送信する IoT デバイスです． IoT 人感センサは，人体などの動きを検出したときに LTE 送信を行います．IoT 温度センサは，10 分間隔で温度データを送信します．さらに省電力機能も使ってみます．

応用例として，閑散期の店舗で音声による接客を行う，IoT 店番システムについても紹介します．

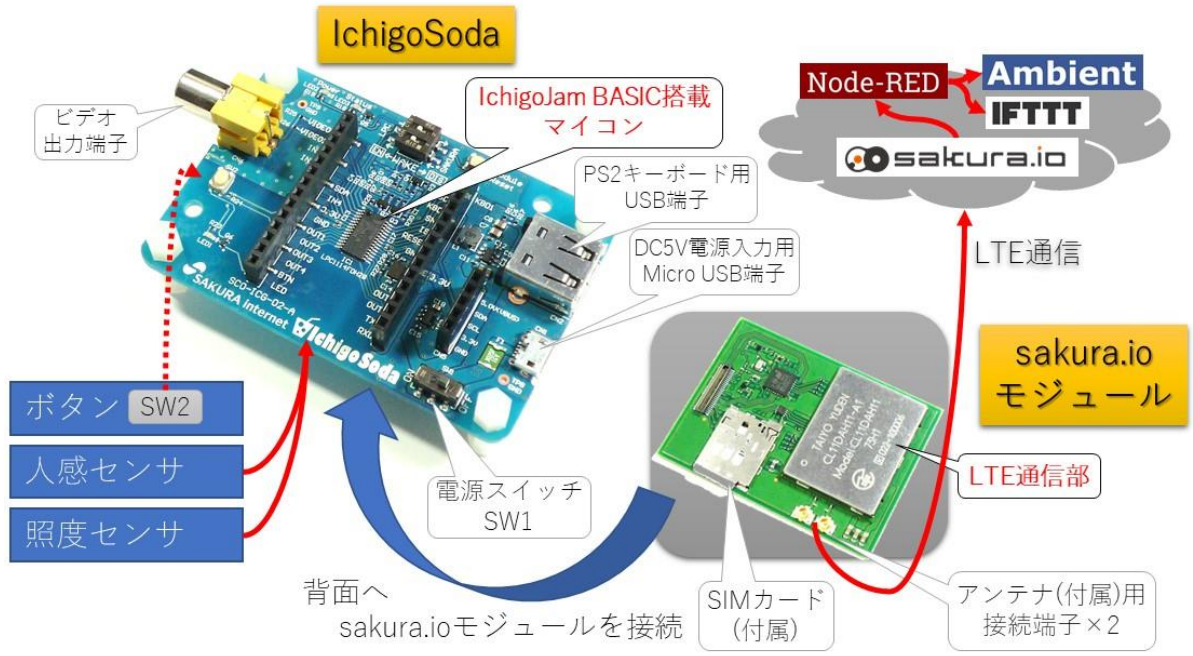

表 1-1　sakura.io を使った IoT デバイス例

通信方式	LTE Category 1
通信エリア	SoftBank 4G LTE サービスエリア内
通信料金	月額 60 円～
機器コスト	13,000 円～
IoT 機能例	ボタン，人感，温度，OLED 表示など

表 1-2　sakura.io による省電力特性の一例

状態	**全体**	**sakura.io**
通常状態	約 75mA	約 60mA
省電力待機時	約 12mA	約 10mA
ディープ･スリープ	約 6mA	約 4mA

5V 入力での実測値

写真 1-1 sakura.io モジュールを使った実験の様子

IchigoSoda の基板裏面に sakura.io モジュールを取り付けて，IchigoJam BASIC で製作したプログラムを使って LTE 送信を行った

月額 60 円の LTE 通信料で利用可能な sakura.io モジュール

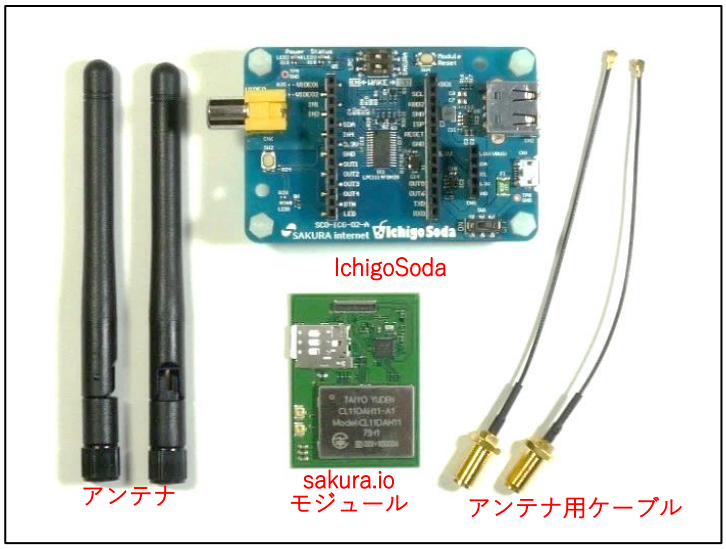

写真 1-2 IchigoSoda および sakura.io モジュール

IoT 押しボタンを製作するために必要な，IchigoSoda（中央・上），sakura.io モジュール（中央・下），付属のアンテナ 2 本（左側）とアンテナ用ケーブル 2 本（右側）の一例

sakura.io モジュールは，ソフトバンクの LTE モバイル回線を使って，さくらインターネット社の IoT クラウドサービスに接続可能な IoT 専用の通信モジュールです．基本サービスの月額は 60 円（税別）と安価なので，モバイル回線を使った IoT デバイスを手軽に運用することが出来るでしょう．

月額には，約 4.5 分間隔の通信に相当する 1 万回分の通信料ポイントが含まれています．IoT 押しボタンや，IoT 人感センサ，IoT 温度センサといった用途であれば，基本サービスの月額だけで十分でしょう．1 万回に満たなかった分の余剰ポイントは，3 か月，有効です．1 万回を超えた場合は，2 万回につき 100 円（税別）の追加料金が課金され，約 1.5 分間の通信で 160 円，約 1 分間隔で 260 円になります．追加料金の余剰ポイントの有効期限は 2 年間あり，月をまたいで使うこともできます．

sakura.io モジュールが手元に届いたら，sakura.io のコントロールパネル（コンソール）へアクセスし，sakura.io モジュールの登録を行います．モバイル回線の契約を伴うので，クレジットカードが必要です．登録方法については，sakura.io のウェブサイトを参照してください．

> sakura.io モジュール登録方法：
> https://sakura.io/docs/pages/guide/tutorial/module-register.html

LTE 対応 IoT 押しボタンのハードウェア構成例

LTE 対応 IoT 押しボタンを製作するための構成例を表 1-3 に示します．IchigoSoda（SCO-ICG-02）と sakura.io モジュールを購入し，IchigoSoda の裏面へ sakura.io モジュールを装着することで，簡単にハードウェアを準備することが出来ます．また，周辺機器として，USB コネクタつき PS2 キーボード，ビデオ入力端子付きテレビ，DC5V 出力の Micro USB 用 AC アダプタが必要です．

表 1-3　sakura.io モジュールを使った IoT デバイスの機器構成例

機器名	販売元	型番	参考価格	備考
IchigoSoda	さくらインターネット	SCO-ICG-02	5,000 円	IchigoJam BASIC 搭載
sakura.io モジュール	さくらインターネット	SCM-LTE-01	8,000 円	アンテナ付属

ハードウェアの組み立てと周辺機器の接続を行い，IchigoSoda 上の電源スイッチ SW1 を ON 側にスライドすると，テレビ画面に「IchigoJam BASIC 1.2.3 by jig.jp」のような文字が表示されます．数字は IchigoJam BASIC のファームウェアのバージョンです．

LTE 対応 IoT 押しボタンのプログラムを入力する

IchigoJam BASIC のコマンドやプログラムは，キーボードから入力し，末尾にリターン・キーを押すことで，入力できます．先頭に数字（行番号）を付与すると，プログラムとしてメモリ内に一時的に保持されます．先頭に数字が無ければコマンドとして即実行されます．なお，IchigoJam BASIC ではアルファベットの大文字と小文字は区別されません．本書で紹介するプログラムでは，読みやすさを考慮し，コマンドを小文字，変数を大文字で表記しましたが，全て大文字で入力しても変わりません．

IchigoJam BASIC を使った IoT 押しボタン用のプログラムをリスト 1-1 に示します．はじめに new コマンドでプログラムを消去してから，リストの左側の BASIC プログラムの内容を入力してください．プログラムは PC や Wi-Fi 拡張ボード MixJuice でダウンロードすることも出来ます．

IchigoSoda の電源を切ると，プログラムが消えるので，入力後に「SAVE 1⏎」を実行して保存しておきます．SAVE コマンドに続く数字は，0～3 の範囲内で指定することができるファイル番号です．ファイルを読み込むときも「LOAD 1⏎」のように末尾にファイル番号を付与します．なお，IchigoJam BASIC IoT 版（コラム参照）については，ファイル番号 0 しか利用できません．

リスト 1-1　LTE 対応 IoT 押しボタン SAKURA IoT TX BTN/PIR

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"SAKURA IoT TX BTN/PIR	1 画面へ「SAKURA IoT TX BTN/PIR」を表示
2 ?"CC BY Wataru Kunino	2 画面へ「CC BY Wataru Kunino」を表示
3 ?"ﾎﾞﾀﾝ ﾆｭｳﾘｮｸ B=BTN	3 画面へ「ﾎﾞﾀﾝ ﾆｭｳﾘｮｸ B=BTN」を表示
100 ?"== ｼﾞｭﾝﾋﾞ ==	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
110 poke #880,#21,#A,1,asc("I")	110 即時送信用の I2C コマンドを準備
120 for I=#884 to #88B	120 変数 I を#884～#88B まで繰り返し
130 　poke I,0	130 変数 I が示すアドレスへ 0 を書き込む
140 next	140 I が#88B 未満時に 1 を加算し 170 行目へ
150 B=btn()	150 変数 B へボタン SW2 の状態を代入
200 ?"== ﾒｲﾝ ==	200 画面へ「== ﾒｲﾝ ==」を表示
210 ?"B=";B	210 画面へ変数 B の内容を表示
220 poke #884,B	220 変数 B の値をアドレス#884 へ書き込む
230 poke #88C,B^#63	230 パリティをアドレス#88C へ書き込む
240 I=i2cw(#4F,#880,13)	240 I2C データ送信の実行．結果を変数 I へ
250 ?"I2C=";I	250 画面へ送信結果を表示
260 if I ?"I2C ERR":end	260 変数 I が 0 以外（エラー）のときに終了
270 for I=0 to 12	270 I が 0 から 12 になるまで繰り返す
280 　? hex$(peek(#880+I));" ";	280 　I2C 送信用のデータを画面へ表示
290 next:?	290 I が 12 未満のとき 1 加算し 270 行目へ
300 ?"== BTN ﾏﾁ ==	300 画面へ「== BTN ﾏﾁ ==」を表示
310 wait 30	310 約 0.5 秒の待ち時間処理
320 if B=btn() cont	320 ボタン SW2 の状態に変化があるまで待機
330 B=!B	330 変数 B の値を反転（0→1，1→0）
340 goto 200	340 行番号 200（メイン）へ戻る

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/11.txt
・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[11]）

LTE 対応 IoT 押しボタンのプログラムの処理内容

リスト 1-1 の行番号 1～3 は，画面へ起動メッセージを表示する処理部です．行番号 1 の「cls」は画面消去命令，その後の「:」は命令の区切りを示し，続く「?」は画面へ文字を表示するための命令です．以下，主な処理内容について説明します．

① 行番号 100 番台は，sakura.io モジュールから LTE 送信を行うための I²C 送信コマンドを準備します．その初期値を，IchigoJam 内のメモリ・アドレス#880 番地から#88B 番地へ書き込みます．

② 行番号 200 番台がメイン処理部です．行番号 220（手順④）では，変数 B に代入された送信データをアドレス#884 番地へ書き込み，次の行番号 230 で，#88C 番地にパリティ（送信データの排他的論理和）を書き込みます．行番号 240（手順⑤）の「i2cw」コマンドで，#880～#88C の計 13 バイトのデータを sakura.io モジュールへ I²C 送信し，その結果を変数 I に代入します．sakura.io モジュールとの I²C 通信に失敗すると I に 1 が代入され，行番号 260 でプログラムを終了します．

③ 行番号 300 番台は，ボタン SW2 の状態変化待ち処理を行います．行番号 320（手順⑥）では，btn 命令でボタン状態に変化が生じるまで留まる処理を行います．その前の行番号 310 の「wait」は 1/60 秒単位で待ち時間処理を行うコマンドです．LTE への送信間隔の確保と，ボタン操作時に値が定まらないチャタリングを防止します．ボタンに変化があった場合は，行番号 200 へ戻ります．

LTE 対応 IoT 押しボタン送信実験・送信結果の確認方法

LTE 対応 IoT 押しボタンが sakura.io プラットフォームへ送信したイベント情報を，PC やスマートフォンなどのブラウザ（Edge や Chrome，Safari）で確認する方法について説明します．

① sakura.io のコントロールパネル (https://secure.sakura.ad.jp/iot/) へアクセスし，図 1-1 のプロジェクト画面を表示します．すでにログインしていた場合は，画面左上の sakura.io のロゴをクリックしてください．本画面では，プロジェクトの[詳細]ボタンをクリックします．

図 1-1 連携サービスの追加① sakura.io のプロジェクト画面で[詳細]をクリック

sakura.io のコントロールパネル（https://secure.sakura.ad.jp/iot/）で，画面左上のロゴをクリックすると，本画面が表示されるので，[詳細]をクリックする．

② 図 1-2 のプロジェクトの詳細画面で[連携サービス]をクリックしてから，画面右方の[サービス追加]をクリックしてください．

図 1-2 連携サービスの追加② sakura.io のプロジェクト詳細画面で[連携サービス]をクリックし[サービス追加]をクリック

連携サービスに websocket を追加するには，図 1-3 の画面で[WebSocket]を選択

③ 図 1-3 の画面で[WebSocket]を選択すると，図 1-2 の連携サービス欄に「websocket」が追加されます．

図 1-3 連携サービスの追加③追加サービスの選択画面で[WebSocket]を選択する

前画面で[サービス追加]を選択したときに表示される追加可能なサービスの一覧．今回は，この中から[WebSocket]を追加する

④ 図 1-2 の連携サービスに追加された「websocket」内の設定アイコン④（青色の歯車）をクリックすると，図 1-4 のサービス連携の編集画面が表示されます．

図 1-4 連携サービスの追加④サービス連携の編集画面で受信データを確認する

図 1-2 の連携サービス内の設定アイコン④（青色の歯車）をクリックすると表示される．IchigoSoda から LTE 送信を実行すると，本画面の下部に受信結果が表示される．12:17:36 に IoT 押しボタン起動時のボタン開放状態の 0 を，12:17:42 に押下状態の 1，約 1 秒後に 0 を受信した．

sakura.io モジュールは自動的に LTE 回線へ接続します．接続されているかどうかは IchigoSoda 基板の右上にある LED 3（Status）の点滅回数で確認出来ます．電源を入れた直後は，点滅回数が 3 回（準備中）ですが，しばらくすると 2 回（ネゴシエーション中），1 回（通信可能状態）へと遷移します．この状態で，IchigoJam BASIC へ「RUN⏎」命令を入力すると，IchigoSoda 上のボタン SW2 の状態が LTE 送信されて，sakura.io の図 1-4 の画面下部に受信結果が表示されます．

LTE 対応 IoT 人感センサで来客を検知

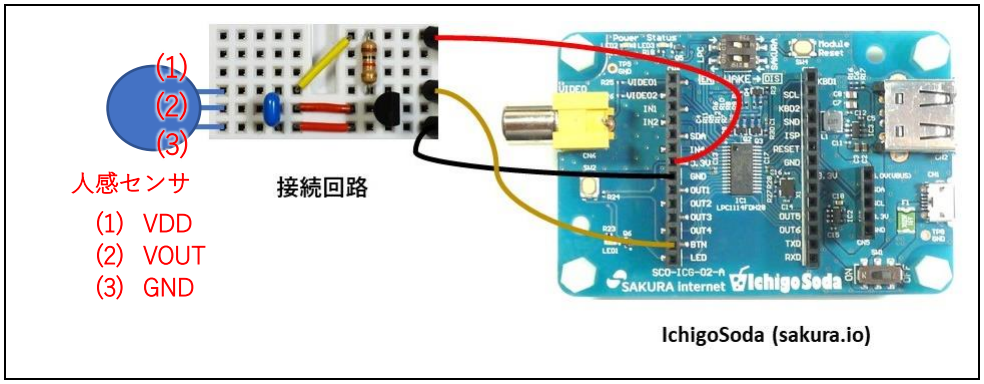

図 1-5 人体の動きなどを検知して LTE 送信を行う IoT 人感センサ

接続回路を経由して人感センサを接続

IoT 押しボタンのハードウェアに人感センサを追加し，人感センサが人の動きなどを検知したときに，LTE 送信を行う IoT 人感センサの製作方法について説明します（図 1-5）．写真 1-3 は，ブレッドボード上に写真 1-4 の人感センサ・モジュール SB412A を実装した IoT 人感センサの製作例です．

人感センサ・モジュール SB412A は，フレネルレンズ，焦電型赤外線センサ，検知 IC を搭載しており，人体などから発生られる赤外線の変化を検出すると，判定結果を VOUT 端子に HIGH/LOW レベルで出力します．写真 1-4 のピン(1)は電源 VDD，ピン(2)は検知出力 VOUT，ピン(3)は GND です．SB412A の特長は，表 1-4 に示すように待機時の電流が標準 1.2μA と低いことと，検知出力（3V）の保持期間を基板上の可変抵抗器で調整できることです．

表 1-4　赤外線センサ・モジュール SB412A

項目	仕様
種別・型番	焦電型赤外線センサ SB412A
電源電圧	DC 3.3V～12V
待機時電流	標準 1.2μA（最大 20μA）
検知時出力	3V
検知出力保持時間	5 秒～60 分(標準 3.5 秒～80 分)
非検知時出力	0V
最大検知距離	3～5m
検知角度	100°

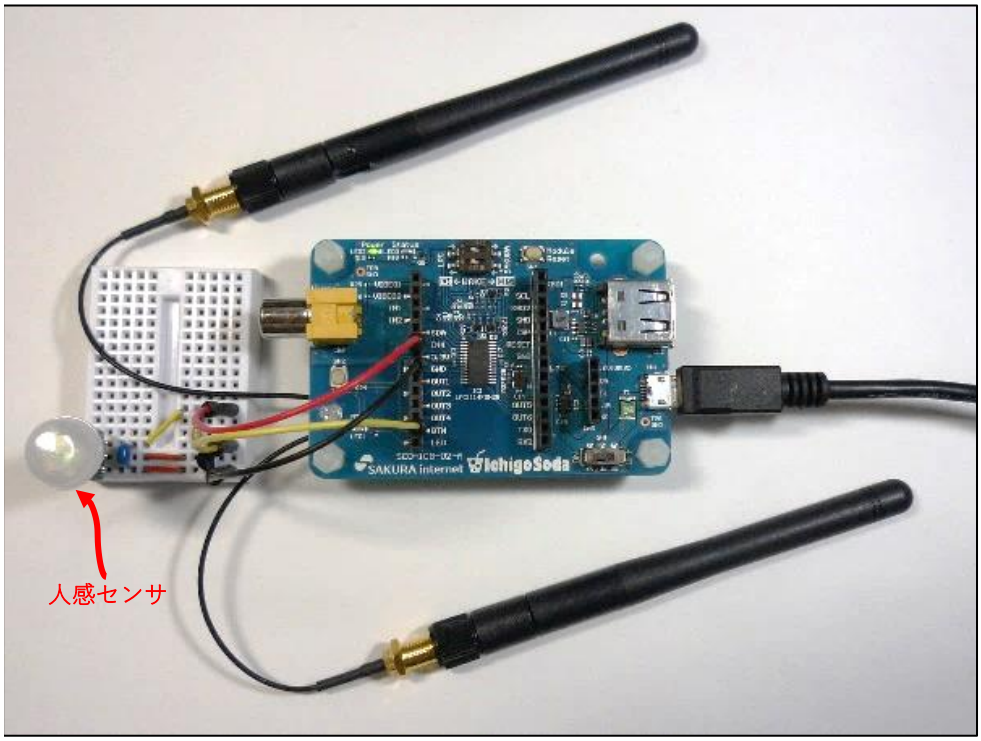

写真 1-3 ハンダ付け不要のブレッドボードで製作した IoT 人感センサ
LTE 方式に対応した sakura.io モジュールを装着した IchigoSoda へ，人感センサを接続した

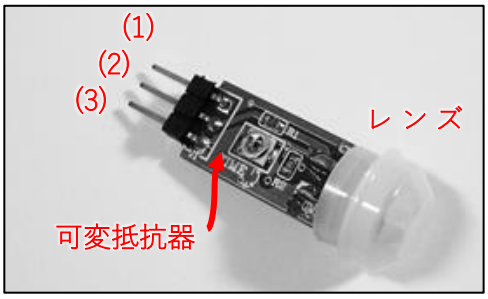

写真 1-4　人感センサ・モジュール SB412A
赤外線の変化から人体などの動きを検知するレンズ付き焦電型赤外線センサ・モジュール．小型で待機時の消費電流が 12μA と小さい

IchigoSoda に人感センサ SB412A を接続するには，図 1-6 のような接続回路を用いて，信号の論理を反転し，BTN 端子へ入力ます．この回路により，IoT 押しボタンのボタン SW2 の操作を，人感センサが行うようになります．なお，ON 抵抗が大きいと IchigoJam のプログラムの自動起動が不安定になるので，MOS FET を使用しました．

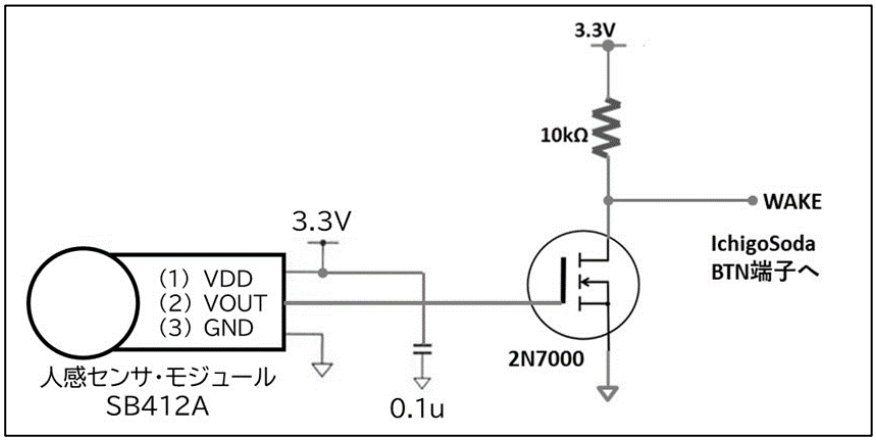

図 1-6 人感センサ接続用回路図
人感センサ・モジュールの出力を，MOS FET で論理反転を行う回路の一例．WAKE 出力を IchigoSoda の BTN 端子に接続する

表 1-5　sakura.io モジュールを使った IoT デバイスの機器構成例

部品名	型番	参考単価	必要数	備考
人感センサ	Nanyang Senba SB412A	500 円	1 個	秋月電子通商などで販売
MOS FET	2N7000	20 円	1 個	ドレイン負荷 1mA 未満で使用する
コンデンサ	村田 0.1uF 250V 5mm ピッチ	15 円	1 個	センサ電源用
抵抗	10 k～100kΩ 1/4W	1 円	1 個	動作確認用（なくても動作する）

人感センサ・モジュール SB412A の検知保持時間は，写真 1-5 のように，最小値（約 3.5 秒）に設定します．センサの動作確認時は，人感センサ SB412A を図 1-6 の接続回路へ接続し，接続回路の WAKE 出力と GND との間の電圧をテスターで測ります．WAKE 出力を IchigoSoda へ接続しなくても，3.3V を人感センサ SB412A と負荷抵抗 10kΩへ供給することで，WAKE 出力の確認が出来ます．身体や指を動かさない状態にしてから約 3.5 秒～10 秒以内に 3.3V が得られ，身体を動かすと 0V が得られるはずです．適切に動作しない場合は，接続回路と検知保持時間の設定を再確認してください．

ハードウェアの製作が完了したら，IoT 押しボタンの製作で保存したリスト 1-1 を「LOAD 1⏎」などのコマンドで読み込み，「RUN⏎」で実行してください．検知するたびに LTE 送信が行われます．なお，検知時間を長く設定して，LTE 送信頻度を下げることで通信料を節約することも出来ます．

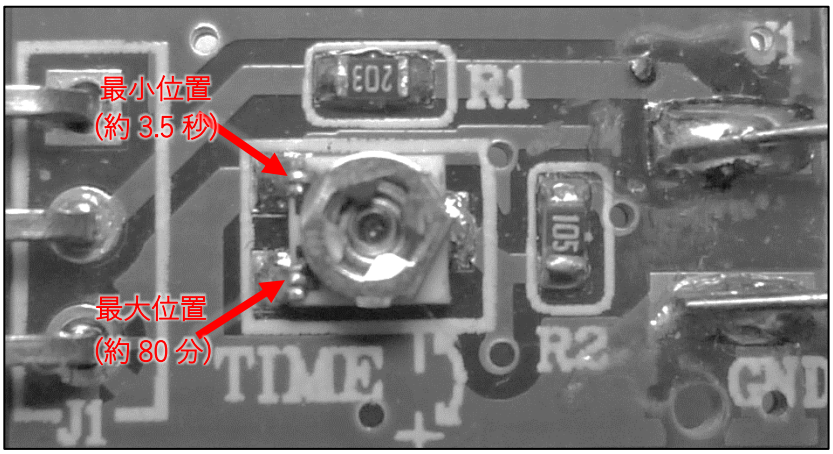

写真 1-5　SB412A の検知時間の設定方法
動作確認時は可変抵抗を最小位置にして検知保持時間を約 3.5 秒に設定する．最大位置では約 80 分になる

LTE 対応 IoT ディスプレイ端末

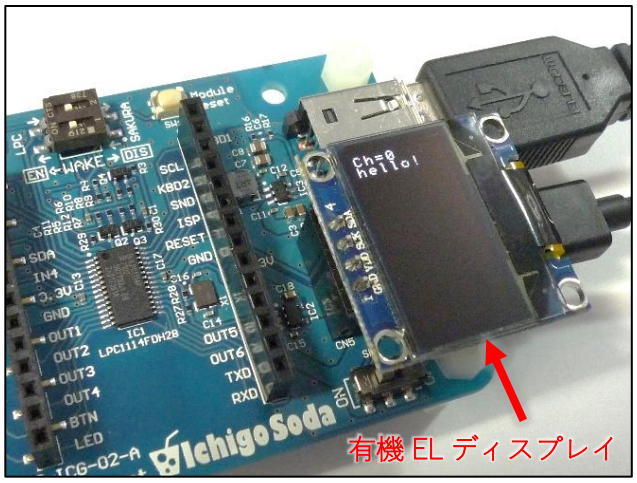

写真 1-6　受信したメッセージを有機 EL ディスプレイへ表示したときの様子

受信用ソフトに，SAKURA IoT RX OLED（http://git.bokunimo.com/MJ/pg08/16.txt）を使用し，sakura.ioモジュールで受信したメッセージを，SSD1306搭載・有機ELディスプレイへ表示した．ピンの並びは，写真の下方からGND，VCC，SCK，SDA.

こんどは，sakura.io モジュールが LTE で受信したメッセージを IchigoSoda で表示する IoT ディスプレイの製作方法について説明します．受信機能を使いこなせば，遠隔地にある機器を制御することもできるようになります．

IoT ディスプレイへ表示するメッセージを送信するには，パソコンやスマートフォンのブラウザから，下記の WebSocket 確認ツール Message IoT for sakura.io を使用します．

Message IoT WebSocket 送受信確認ツール：
https://goo.gl/vcSXoo　(http://git.bokunimo.com/IchigoJam/message_iot/)

WebSocket Token の欄には，sakura.io コントロールパネルの「サービス連携の編集」画面に記載されている WebSocket 用の Token を入力します．プロジェクト画面から[詳細]→[連携サービス]を選択し，連携サービス内の「websocket」に表示される[設定]（青色の歯車のアイコン）をクリックすると，表示されます．Token を入力してから，5 秒ほど経過すると，Log の枠内に「connected」が表示されます．表示されなかった場合は，ブラウザをリロードしてやり直してください．

図 1-7 連携サービスの追加④サービス連携の編集画面で，受信データを確認する

WebSocket 確認ツール Message IoT - sakura.io. 全角のひらがなやカタカナを入力すると，半角カナに変換して送信することが出来る．sakura.io モジュールが送信したデータを受信することも出来る

Module ID の欄には，sakura.io モジュールの ID を入力します．Module ID が分からない場合は，sakura.io コントロールパネルのプロジェクト画面で確認してください．

Message の欄に 8 文字までのテキストを入力し，［送信］ボタンをクリックすると，sakura.io へメッセージが送信されます．数値を送信したいときは，Value の欄へ整数 0～32767 を入力してください．

IchigoSoda へリスト 1-2 を入力し，実行すると，待ち受けを開始します．WebSocket 確認ツール Message IoT からテキストメッセージや数値を送信すると，IchigoSoda は受信結果をテレビ画面へ表示します．送信内容と受信内容が一致しているかどうか，確認してください．

また，SSD1306 を搭載した有機 EL ディスプレイへ表示することも出来ます．有機 EL ディスプレイを IchigoSoda の CN5 の，GND，3.3V (VCC)，SCK，SDA の端子へ接続し，SAKURA IoT RX OLED（http://git.bokunimo.com/MJ/pg08/16.txt）をダウンロードして実行したときの様子を，写真 1-6 に示します．秋月電子通商の商品コード P-12031 であれば，IchigoSoda の 5 ピンの拡張コネクタ CN5 へ直結することが出来ます．

リスト 1-2　LTE 対応 IoT ディスプレイ SAKURA IoT RX

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"SAKURA IoT RX	1 画面へ「SAKURA IoT TX BTN/PIR」を表示
2 ?"CC BY Wataru Kunino	2 画面へ「CC BY Wataru Kunino」を表示
3 T=1:?" ｼｭｳｷ=";T;" ﾋﾞｮｳ	3 画面へ「ｼｭｳｷ= 1 ﾋﾞｮｳ」を表示
100 ?"== RX ﾏﾁｳｹ ==	100 画面へ「== RX ﾏﾁｳｹ ==」を表示
110 ?".";:wait T*60	110 待ち受け中に「.」を表示
120 poke #880,#32,0,#32	120 受信確認コマンドを生成
130 I=i2cr(#4F,#880,3,#884,4)	130 受信確認コマンドを実行
140 if I ?:?"I2C ERR"	140 変数 I が 0 以外のときにエラー表示
150 N=!I*peek(#887)	150 変数 N へ受信件数を代入
200 '== RX ==	200 画面へ「== RX ==」を表示
210 if N<=0 goto 110	210 受信件数が 0 以下のときに待ち受けに戻る
220 poke #880,#30,0,#30	220 受信結果の取得コマンドを生成
230 I=i2cr(#4F,#880,3,#884,12)	230 受信結果の取得を実行
240 if I goto 100	240 I2C エラー時に 100 行(待ち受け)に戻る
250 N=N-1	250 受信件数を 1 減算
260 cls:?"Ch=";peek(#886)	260 受信チャンネル番号を表示
270 if peek(#887)=98 goto 300	270 テキスト文字を受信したときに 300 行へ
280 ? peek(#888)+peek(#889)<<8	280 受信した整数値を表示
290 goto 200	290 行番号 200（受信）へ戻る
300 '== String ==	300 テキスト文字の処理部
310 for I=0 to 7	310 8 文字分の繰り返し処理
320 ? chr$(peek(#888+I));	320 受信したテキスト文字（1 文字）を表示
330 next:?	330 繰り返し処理後に改行を出力
340 goto 200	340 行番号 200（受信）へ戻る

①：120 poke #880,#32,0,#32 ～ 150 N=!I*peek(#887)

②：200 '== RX == ～ 290 goto 200

③：300 '== String == ～ 340 goto 200

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/15.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[15]）

有機 EL ディスプレイ対応版は下記からダウンロード可能

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/16.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[16]）

音声で「いらっしゃいませ」．八百屋の店番システムの製作例

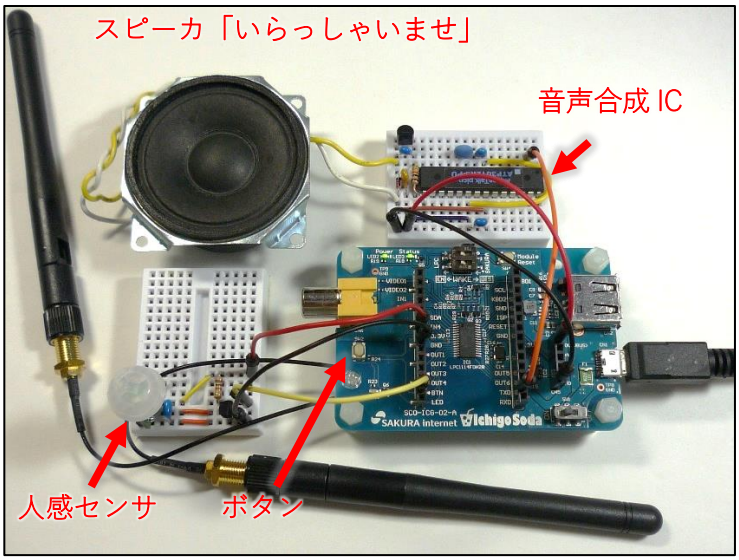

写真 1-7 八百屋の店番システム向け IoT 音声アナウンス端末の製作例
人感センサが人体などの動きを検知すると，スピーカが「いらっしゃいませ」を発声するとともに，LTE通信で来客を通知する．また，スマートフォンからの操作で「少々お待ちください」と発声することも可能

LTE 対応 IoT 人感センサと IoT ディスプレイを応用し，八百屋の店番システム向け IoT 音声アナウンス端末を作成してみました．もちろん，八百屋以外でも利用することも可能です．

人感センサが人体などの動きを検知すると，スピーカから音声「いらっしゃいませ」を再生するとともに，LTE で数値 0 を送信します．このときに，スマートフォンの LINE アプリへ「人感センサが反応しました」と通知することも出来ます（方法は後述）．

また，WebSocket 確認ツール Message IoT の Value の欄に「1」を入力して[送信]ボタンをクリックすると，音声「御用の際はボタンを押してください」を再生し，「2」を入力すると「少々，お待ちください」を再生します．音声再生だけでなく，店舗内の機器をリレーで制御するといった遠隔制御などに応用することも出来るでしょう．

写真 1-7 に試作例，図 1-8 に音声合成 IC 部の回路図例，写真 1-8 に配線例を示します．プログラム SAKURA IoT YaoYa Talk は，PC から 10.txt（http://git.bokunimo.com/MJ/pg08/10.txt），MixJuice のメニュー形式（?”MJ GET git.bokunimo.com/MJ/808.txt”）の[10]でダウンロードすることが出来ます．

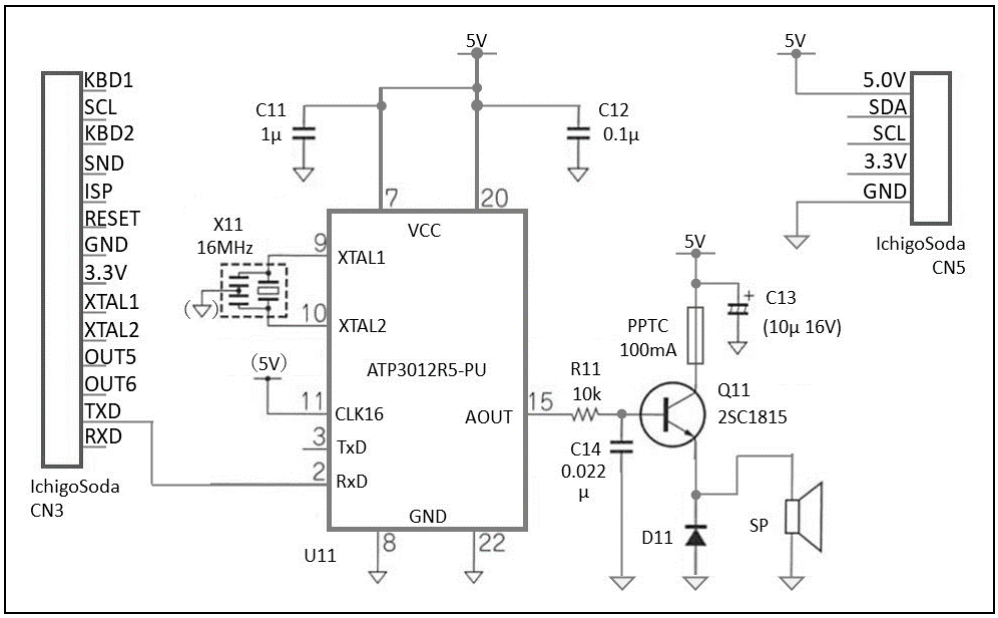

図 1-8 音声合成 IC 部の回路図例
IchigoSoda の UART シリアル出力 TXD ピンに AquesTalk pico を接続し，AquesTalk pico の AOUT に音声アンプを接続．電源 5V と GND は IchigoSoda の CN5 に接続

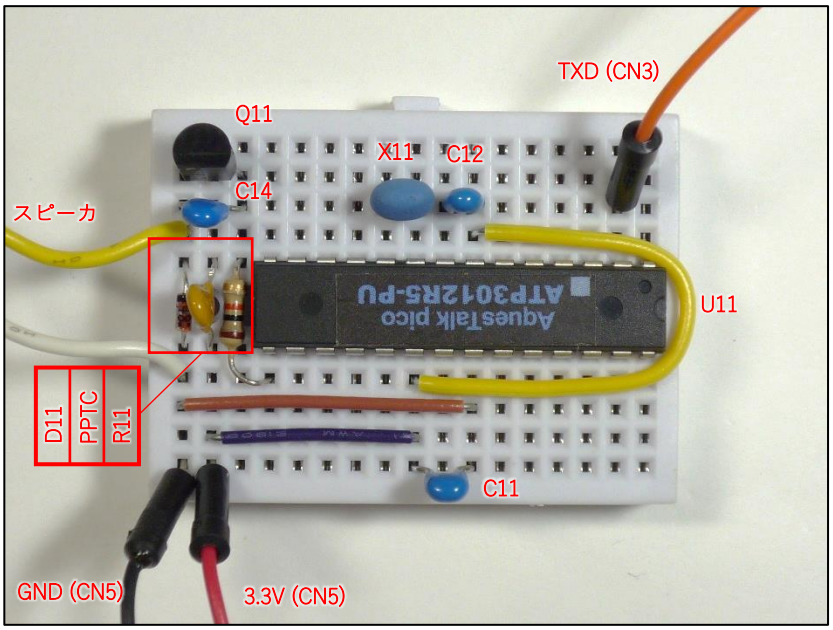

写真 1-8 音声合成 IC 部の配線例

試作・実験用に製作した音声合成IC部の配線例．試作例では，ミニ・ブレッドボードを使用した．セラロックX11の中央のGND端子と，C13を省略した（実力的に動作）

sakura.io モジュールのファームウェア更新方法

以降のプログラムは，sakura.io モジュールのファームウェア Ver 1.4.0 でサポートされた省電力動作モードを使用します．執筆時点で購入したものは，Ver 1.2.1 でしたので，以下の方法でファームウェアの更新を行いました．

ファームウェアの更新は，LTE 網へ接続した状態で実行します．IchigoSoda を起動し，IchigoSoda 上の LED3（Status）の点滅回数が 1 回になるまで待ってから，リスト 1-3 のコマンドを入力してください．

より確実に更新するために，接続状態や更新状況を確認してから更新を行う専用プログラムも準備しました．更新後のバージョンも表示されます．GitHub 内のページ（git.bokunimo.com/MJ/）でファイル名「18.txt」で公開しており，PC もしくは MixJuice でダウンロードすることができます．

リスト 1-3　SAKURA Firmware Updater

BASIC コマンド	解説
POKE#880,#A8,4,#53,#6B,#72,#61,#87:I=I2CW(79,#880,7): POKE#880,#A9,0,#A9:IFIORI2CW(79,#880,3)?"ERR	特殊コマンドアンロック命令とファームウェア更新命令を送信する
BASIC コマンド **・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/19.txt** **・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[19]）** **更新専用プログラム** **・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/18.txt** **・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[18]）**	

省電力動作方法と，省電力 IoT 押しボタン／IoT 人感センサ用プログラム

IchigoSoda と sakura.io モジュールの両方を省電力動作させることで，一例として，通常時 75mA の消費電流を 12mA 程度（約 6 分の 1）に削減することができます.

表 1-6 は，IchigoJam BASIC による IchigoSoda と sakura.io モジュールの省電力動作方法の一覧表です．sakura.io モジュールの消費電力を下げるには，ディープ・スリープ・モードに設定する方法と，省電力動作モードにする方法があります．sakura.io モジュールのディープ・スリープ・モードは，スリープからの復帰に約 1 分の時間を要するので，ここでは省電力動作モードを使用します.

通常時の IchigoSoda 側の消費電力は，全体の 2 割程度ですが，sakura.io の消費電力を下げると，目立つようになります．IchigoJam BASIC には省電力動作させるためのコマンドには，ビデオ出力をオフにするとともにマイコンの動作周波数を下げることが可能な VIDEO コマンド，指定した期間だけスリープに遷移する WAIT コマンド，BTN 端子から復帰可能な SLEEP コマンドの 3 つが用意されています．ここでは SLEEP コマンドを，また後の IoT 照度センサでは WAIT コマンドを使用します.

リスト 1-4 の省電力 IoT 押しボタン・人感センサ用のプログラムの主要な処理を以下に示します.

① 行番号 100 番台の前半の行番号 120～140 は，sakura.io モジュールの省電力設定部です．スリープからの復帰時など，ボタン SW2 が押下状態だった場合は，既に設定済みなので，行番号 150 へ移ります．後半は，LTE 送信するための I²C 送信コマンドの準備を行います.
② 行番号 200 番台は，メイン処理部です．行番号 210 と行番号 240 の gosub 命令は，行番号 300～360 のサブルーチン（手順③）を実行します.
③ 変数 B に代入されたボタン状態を sakura.io モジュールから LTE 送信します.
④ sleep 命令は IchigoJam BASIC を終了する命令です．終了後に IchigoJam BASIC を起動するには，ボタン SW2 を押下します．起動後，「SAVE 0」で保存したプログラムが自動実行されます.

プログラムを「RUN」すると，通常モードでの動作確認が行えます．正しく動作することが確認できたら，行番号 4 の「S=0」を「S=1」に変更し，コマンド「SAVE 0」でプログラムを自動起動用プログラム専用のファイル番号 0 へ保存してください．再度，「RUN」を入力すると，テレビ画面への表示が消え，省電力動作を開始します．停止するときは，キーボードの ESC キーを押したまま，ボタン SW2 を押下するか，一度，IchigoSoda の電源を切ってから入れなおします．なお，キーボードも多くの電力を消費するので，実運用時はキーボードを外しておきましょう．キーボード無しにプログラムを起動するには，ボタン SW2 を押したまま電源を入れます.

表 1-6　IchigoJam BASIC による IchigoSoda と sakura.io モジュールの省電力動作方法の一覧

対象	省電力方法	コマンド例	内容（動作条件）
sakura.io	ディープ・スリープ・モード	OUT 8,0 (設定) OUT 8,1 (解除)	動作を停止して省電力化. (IchigoSoda 上の WAKE スイッチを EN へ)
	省電力モード	POKE #880,#B0,1,1,#B0: ?!I2CW(79,#880,4)	動作させたまま省電力化. (ファームウェア 1.4.0 以降が必要)
IchigoJam	低クロック駆動	VIDEO 0,8 (設定) VIDEO 1(解除)	ビデオ出力を停止し，マイコンの動作クロックを下げて省電力化
	Cyclic Sleep 駆動	WAIT 300,0	指定時間（1/60 秒単位）スリープしてから，自動復帰
	Single Shot 起動	SLEEP	BTN 端子が H レベルでスリープへ遷移. BTN 端子が H→L レベル遷移時に復帰.

リスト 1-4　LTE 対応 IoT 押しボタン SAKURA IoT TX BTN/PIR PS

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"SAKURA IoT TX BTN/PIR PS	1 画面へ「SAKURA IoT TX BTN/PIR PS」を表示
2 ?"CC BY Wataru Kunino	2 画面へ「CC BY Wataru Kunino」を表示
3 ?"ﾎﾞﾀﾝ/ｾﾝｻ ﾆｭｳﾘｮｸ B=BTN	3 画面へ「ﾎﾞﾀﾝ ﾆｭｳﾘｮｸ B=BTN」を表示
4 S=0:?"ｼｮｳﾃﾞﾝﾘｮｸ S=";S	4 省電力動作を行うときは変数 S=1 に変更する
100 ?"== ｼﾞｭﾝﾋﾞ =="	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
110 if btn() goto 150	110 ボタンが押されていた時は行番号 150 へ
120 poke #880,#B0,1,S,S^#B1	120 省電力モード#B0 に変数 S の値を設定
130 ?"I2C_PowSave="; 140 ? S * !i2cw(#4F,#880,4)	130 画面へ「I2C_PowSave=」を表示 140 省電力モード設定を実行
150 poke #880,#21,#A,1,asc("I")	150 即時送信用の I2C コマンドを準備
160 for I=#884 to #88B	160 変数 I を#884～#88B まで繰り返し
170 poke I,0	170 変数 I の番地へ 0 を書き込む
180 next	180 I が#88B 未満時に 1 を加算し 170 行目へ
200 ?"== ﾒｲﾝ =="	200 画面へ「== ﾒｲﾝ ==」を表示
210 B=1:gosub 300	210 サブルーチン③を用いて変数 B=1 を送信
220 wait 30	220 約 0.5 秒の待ち時間処理
230 if btn() cont	230 ボタンが押下状態のときに待機する
240 B=0:gosub 300	240 サブルーチン③を用いて変数 B=0 を送信
250 if S sleep	250 変数 S が 1 のときに IchigoJam を停止
260 wait 30	260 約 0.5 秒の待ち時間処理
270 if !btn() cont	270 ボタンが開放状態のときに待機する
280 goto 200	280 行番号 200 へ戻る
300 ?"== ｿｳｼﾝ =="	300 画面へ「== ｿｳｼﾝ ==」を表示
310 poke #884,B	310 変数 B の値をアドレス#884 へ書き込む
320 poke #88C,B^#63	320 パリティをアドレス#88C へ書き込む
330 ?"I2C_Tx=";B;" ";	330 画面へ「I2C_TX=」と変数 B の値を表示
340 I=i2cw(#4F,#880,13)	340 I2C データ送信の実行．結果を変数 I へ
350 if I ?"ERR" else ?"OK"	350 I2C エラー時に「ERR」を表示する
360 return	360 サブルーチンを終了し，gosub 部へ戻る

①

②

③

④

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/12.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[12]）

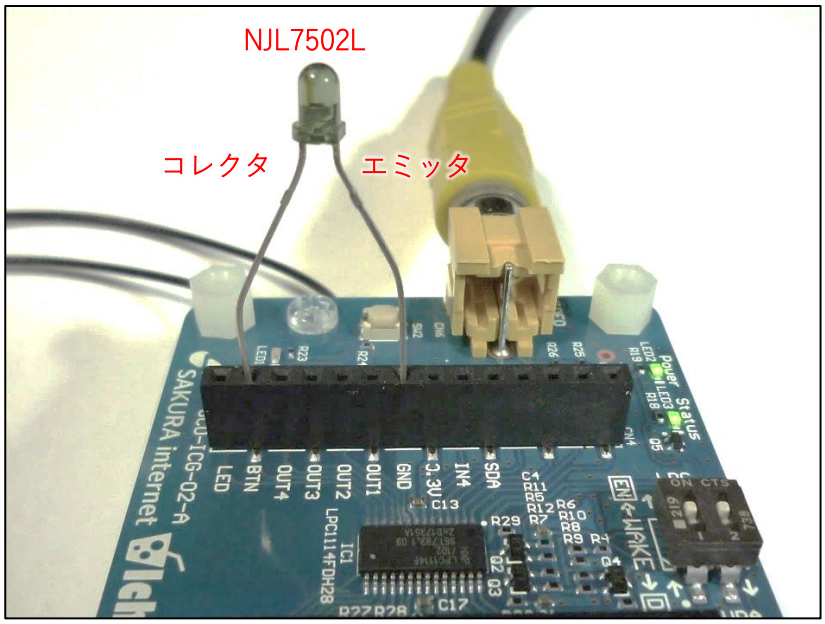

写真 1-9 照度センサの接続
照度センサ NJL7502L の長い方（コレクタ側）のリード線を IchigoSoda の BTN 端子へ接続し，GND 端子へ短い方（エミッタ側）を GND 端子へ接続

こんどは，IoT センサ機器の一例として，LTE 対応 IoT 照度センサを作成します．一定間隔でセンサ情報を送信することで，遠隔地の環境データなどの定期的な履歴データを収集する場合を想定していますが，例えば八百屋であれば，悪天候などで店舗の照度が低くなった時だけ送信するといった使い方に変更することも可能です．

照度センサには写真 1-10 の新日本無線の NJL7502L を使用します．直射日光の入らない室内の照度を概ね得ることが出来ます．より暗い部屋の照度を得たい場合は，高感度な NJL7302L-F3 を使用します．IchigoSoda に取り付けた場合，NJL7502L が約 1000 ルクスまで，NJL7302L-F3 だと約 170 ルクスまでの照度を取得することが出来ます．

写真 1-9 のように，照度センサの長い方（コレクタ側）のリード線を IchigoSoda の BTN 端子へ接続し，短い方（エミッタ側）を GND 端子へ接続すれば完成です．BTN 端子は，IchigoSoda 基板上の 10 k Ωの抵抗 R24 で 3.3V の電源にプルアップされているので，照度センサへ電源を供給しつつ，検出した電圧を BTN 端子へ入力することが出来ます．

IchigoJam BASIC から照度値を取得するには，表 1-7 の取得コマンド例を入力します．NJL7502L の場合，1023 から入力値を減算するだけで，照度値（ルクス）に近い値が得られます．ただし，換算時の誤差や，各部品のばらつき，電圧の変化，温度などの影響を受けます．測定結果を計測器のように利用することは出来ませんが，室内環境の明るさの目安としては十分に利用できることが多いでしょう．

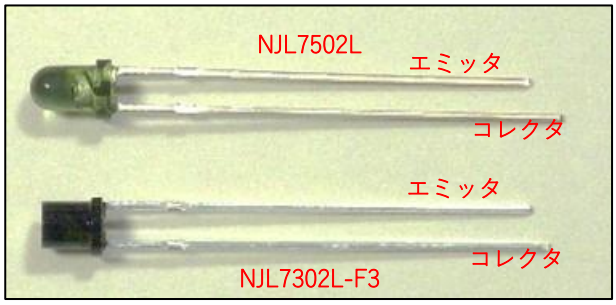

写真 1-10 新日本無線製の照度センサ NJL7502L と NJL7302L-F3
直射日光の入らない室内の照度を概ね得ることが出来る照度センサ NJL7502L と，より感度の高い高感度な NJL7302L-F3

表 1-7　使用する照度センサおよび照度値の取得コマンドの例

照度センサ型番	標準感度	照度換算式	取得コマンド例	取得範囲
NJL7502L	33uA/100lx	= (1023-ana())*0.977	?1023-ana()	0～1023 (lx)
NJL7302L-F3	20uA/10lx	= (1023-ana())/6.2	?170-ana()/6	0～170 (lx)
			?16368-ana()*16	0～1700 (10・lx)

LTE 対応 IoT 照度センサのプログラム

製作した IoT 照度センサのプログラムをリスト 1-5 に，主なプログラムの流れを以下に示します．

① 行番号 100 番台の前半は，sakura.io モジュールの省電力設定部，後半は LTE 送信用コマンドの準備部です．
② 行番号 200 番台は，メイン処理部です．行番号 210 で変数 A に代入された 2 バイトの照度値を，行番号 220 と 230 で変数 B と C へ 1 バイトずつ分割します．また，行番号 240 で照度値が変数 F の照度閾値よりも低い（暗い）ときだけ，gosub 300 で手順③のサブルーチンを実行します．
③ 変数 B と C に代入された照度値を sakura.io モジュールから LTE 送信します．
④ 行番号 4 の変数 T は，秒単位の送信間隔です．ここでは T=90，すなわち 1 分 30 秒が設定されており，月 29,760 回の通信回数が発生します．1 万回以内に収めるには T=270，4 分 30 秒以上に設定すると良いでしょう．
⑤ 行番号 5 の変数 S は，省電力設定です．S=1 に変更すると，約 12mA で動作する省電力動作に切り替わります．
⑥ 行番号 6 の変数 D は，照度センサの種別を示します．D=0 のときは NJL7502 用，D=1 のとき NJL7302L-F3 用です．D=2 のときは NJL7302L-F3 の高感度度特性を活かし，照度の 10 倍値を送信します．例えば，100lx であれば，1000 が送信されます．
⑦ 行番号 7 の変数 F は，照度閾値です．変数 A に代入された照度値が，閾値未満のときだけ送信を行います．初期値は最大の照度値よりも大きな 1024 に設定しています．店舗が暗いときだけ，送信したい場合は，F=200 など，小さな値に設定して下さい．
⑧ 行番号 212～218 は，BTN 端子から得られたアナログ値を照度値へ変換する演算部です．これらの行を削除すれば，入力されたアナログ値のまま送信することも出来ます．また，温度センサなど他のセンサを接続した場合は，この演算部を変更することで，IoT 温度センサなどのプログラムに応用することが出来ます．

ここでは，一定の間隔で照度値を送信するプログラムを紹介しましたが，八百屋の店舗の照度低下の検出用であれば，照度低下の通知を送信後は，しばらくの間，送信を保留しても良いでしょう．照度の推移の記録用途であれば，照度値が大きく変化したときだけ送信を行うといった方法も考えられます．通信料の節約だけでなく，不要な通知やデータ肥大化を防ぐことも出来るでしょう．

このような照度や照度変化の検出閾値や，送信保留時間といったパラメータの調整は，用途や環境によって異なるので，実用的なシステムに仕上げるには，実際に使用する現場での実証実験による調整が必要です．現場でのプログラムの修正や調整が容易な IchigoJam BASIC を使うことで，効率的にシステム開発が行えるでしょう．

リスト 1-5　LTE 対応 IoT 照度センサ SAKURA IoT TX ILM

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"SAKURA IoT TX ILM PS	1 画面へ「SAKURA IoT TX ILM」を表示
2 ?"CC BY Wataru Kunino	2 画面へ「CC BY Wataru Kunino」を表示
3 ?"ｼｮｳﾄﾞ ﾖ ｿｳｼﾝ	3 画面へ「ｼｮｳﾄﾞ ﾖ ｿｳｼﾝ」を表示
4 T=90:?" ｼｭｳｷ=";T;" ﾋﾞｮｳ	4 変数 T へ照度値の取得周期を代入し，表示
5 S=0:?" ｼｮｳﾃﾞﾝﾘｮｸ=";S;" (1=ON)	5 変数 S へ 0 を代入し，変数 S の値を表示
6 D=0:?" ﾃﾞﾊﾞｲｽ=";D;" (0=NJL75)	6 変数 D へデバイス種別を代入
7 F=1024:?" ｼｷｲﾁ=";F	7 LTE 送信するときの照度の判定閾値を代入
100 ?"== ｼﾞｭﾝﾋﾞ =="	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
110 poke #880,#B0,1,S,S^#B1	110 省電力モード#B0 に変数 S の値を設定
120 ?"I2C_PowSave="; 130 ? S * !i2cw(#4F,#880,4)	120 画面へ「I2C_PowSave=」を表示 130 省電力モード設定を実行
140 poke #880,#21,#A,1,asc("I")	140 即時送信用の I2C コマンドを準備
150 for I=#884 to #88B	150 変数 I を#884～#88B まで繰り返し
160 poke I,0	160 変数 I の番地へ 0 を書き込む
170 next	170 I が#88B 未満時に 1 を加算し 170 行目へ
200 ?"== ﾒｲﾝ =="	200 画面へ「== ﾒｲﾝ ==」を表示
210 A=ANA():?" A=";A;	210 変数 A へ BTN の電圧に応じた値を代入
212 A=1023-A	212 照度値へ変換(D=0:NJL7502 用)
214 if D=1 then A=A/6	214 照度値へ変換(D=1:NJL7302-F3 用)
216 if D=2 then A=A*16	216 照度値へ変換(D=2:NJL7302-F3・10 倍値)
218 ?" -> ";A;" lx"	218 照度値を表示
220 B=A%256:?" B=";B	220 変数 B へ変数 A の下位 1 バイトを代入
230 C=A>>8 :?" C=";C	230 変数 C へ変数 A の上位 1 バイトを代入
240 if A<F gosub 300	240 照度が判定閾値よりも暗ければ送信を実行
250 for I=1 to T	250 変数 T の 1/60 回の繰り返し処理
260 led 0:wait 60,!S:led 1	260 1 秒間スリープ
270 next	270 行番号 260 の処理を繰り返す(T/60 秒間)
280 goto 200	280 行番号 200 へ戻る
300 ?"== ｿｳｼﾝ =="	300 画面へ「== ｿｳｼﾝ ==」を表示
310 poke #884,B	310 変数 B の値をアドレス#884 へ書き込む
320 poke #885,C	320 変数 C の値をアドレス#885 へ書き込む
330 poke #88C,B^C^#63	330 パリティをアドレス#88C へ書き込む
340 ?"I2C_Tx=";A;" ";	340 画面へ「I2C_TX=」と変数 A の値を表示
350 I=i2cw(#4F,#880,13)	350 I2C データ送信の実行．結果を変数 I へ
360 if I ?"ERR" else ?"OK"	360 I2C エラー時に「ERR」を表示する
370 return	370 サブルーチンを終了し，gosub 部へ戻る

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/14.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[14]）

1 行リターンで sakura.io の I²C インタフェース実験

IoT システムの開発時や，調整時には，実際に動作するコマンドを実行し，部分的な動作の検証を行うことが良くあります．ここでは，I²C インタフェース用の IchigoJam BASIC コマンドを使って，sakura.io モジュールを制御してみます．

IchigoJam BASIC では，行番号を付与せずにコマンドを入力することで，コマンドを直接実行することが出来ます．LTE 接続状態確認を確認したときの IchigoJam の画面の一例を図 1-9 に示します．本例では，応答値 80 が得られ，LTE 接続状態であることが確認できました．

表 1-8 に sakura.io モジュールに関連したコマンド例を示します．コマンドをプログラム中に記載することも，もちろん可能です．

```
IchigoJam BASIC 1.2.3 by jig.jp
OK
POKE#880,1,0,1:?HEX$(!I2CR(79,#8
00,3,#884,3)*PEEK(#886))⏎
80
OK
```

図 1-9 IchigoJam BASIC のダイレクト・モードで LTE 接続状態を確認したときの様子
I²C インタフェースを使って sakura.io モジュールへ「#01,#00,#01」の 3 バイトを送信し，応答値の 3 バイト目を 16 進数で表示した

表 1-8　IchigoJam BASIC で実行する sakura.io モジュール用ダイレクト・コマンド例

命令の内容	ダイレクト・モードで実行するコマンド例	応答例
LTE 接続状態確認	POKE#880,1,0,1:?HEX$(!I2CR(79,#880,3,#884,3)*PEEK(#886))	80=接続状態 1～3＝エラー
アンテナレベル確認	POKE#880,2,0,2:?!I2CR(79,#880,3,#884,3)*PEEK(#886)	0＝圏外 1＝弱，5＝強
数値データの送信 (1ch，0～255)	A=123: POKE#880,33,10,1,73,A,0,0,0,0,0,0,0,A^99:?!I2CW(79,#880,13)*A&255	0～255 （送信値）
数値データの送信 (2ch，0～255)	A=123: POKE#880,33,10,2,73,A,0,0,0,0,0,0,0,A^96:?!I2CW(79,#880,13)*A&255	0～255 （送信値）
数値データの送信 (3ch，0～255)	A=123: POKE#880,33,10,3,73,A,0,0,0,0,0,0,0,A^97:?!I2CW(79,#880,13)*A&255	0～255 （送信値）
数値データの送信 (1ch，0～65535)	A=12345: POKE#880,33,10,1,73,A,A>>8,0,0,0,0,0,0,A>>8^A^99:?!I2CW(79,#880,13)*A	0～32767 （送信値）
数値データの受信	POKE#880,48,0,48:IF!I2CR(79,#880,3,#884,6)?[68]	-32768～32767 （受信値）
省電力モード設定	POKE#880,#B0,1,1,#B0:?!I2CW(79,#880,4)	1=省電力モード 0＝設定失敗
省電力モード解除	POKE#880,#B0,1,0,#B1:?!I2CW(79,#880,4)	1=通常モード 0＝設定失敗
sakura.io 動作確認	POKE#880,15,4,84,69,83,84,29:?!I2CR(79,#880,7,#888,6)*peek(#888): FORI=2TO5:?CHR$(PEEK(#888+I));:NEXT:?	1=動作状態 0=I²C エラー 2～7＝異常
ファームウェア更新	POKE#880,#A8,4,#53,#6B,#72,#61,#87:I=I2CW(79,#880,7): POKE#880,#A9,0,#A9:IFIORI2CW(79,#880,3)?"ERR	ERR=I²C エラー
ディープ・スリープ・モード設定	OUT8,0 (予めIchigoSoda上のWAKEスイッチをEN側に設定しておく)	なし
ディープ・スリープ・モード解除	OUT8,1 (復帰には約1分の時間を要する)	なし

IFTTT でイベント通知を連携する

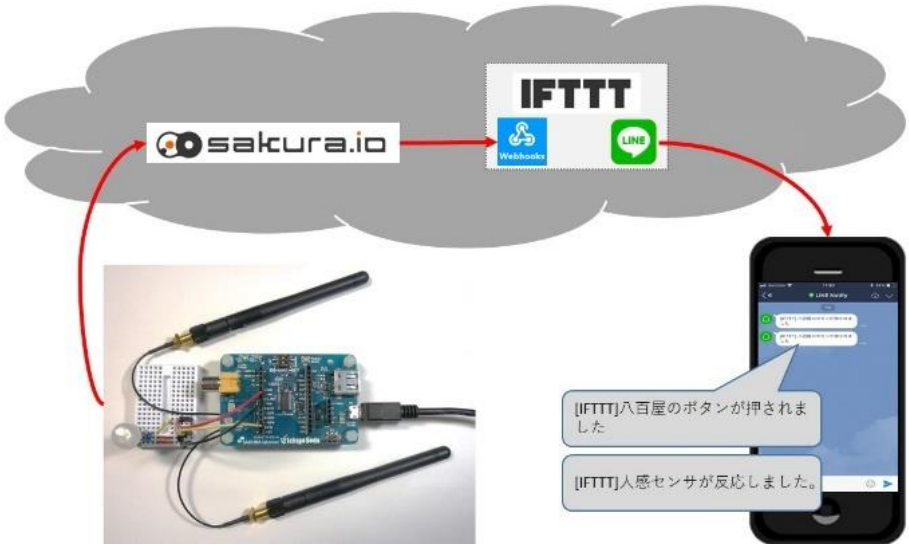

図 1-10 sakura.io からスマートフォンへのイベント通知の連携例（IoT 押しボタン・IoT 人感センサ）
クラウドサービスを利用して，イベント通知をスマートフォンへ連携する様子.

IoT 押しボタンを押すとスマートフォンの LINE アプリへメッセージ「八百屋のボタンが押されました」を送信したり，IoT 人感センサが反応したときに「人感センサが反応しました」と送信したりする方法の一例として，イベント通知を IFTTT（https://ifttt.com/）へ連携する方法を紹介します.

IFTTT は，クラウド連携用のサービスです．図 1-10 のように，sakura.io モジュールが LTE 送信を行うと，sakura.io プラットフォームが IFTTT へイベント情報を送信し，LINE へ通知することができます．ただし，ここでは sakura.io の送信イベントの通知のみを行うので，ボタンを押した時と離した時の 2 回，通知されます．ボタンが押されたときだけ通知したい場合や，照度や温度などのセンサ値データの連携を行うには，Node-RED（次節参照）を使用します.

IFTTT は「if this then that」の略で，「this」に相当するトリガ（入力）に sakura.io からイベントを受信するための Webhooks の設定を，「that」に相当するアクション（出力）に LINE の設定を行います（図 1-11）．ブラウザ（Edge，Chrome，Safari など）で IFTTT へアクセスし，トリガ側のイベント名に「BTN」，アクション側のメッセージに「八百屋のボタンが押されました」を図 1-12 のように設定して下さい．設定後，Webhooks のページ（https://maker.ifttt.com/）へアクセスし，[Documentation]をクリックすると，図 1-13 のような Webhooks 用の Token を含む URL が表示されます．本 URL の{event}に BTN を入力し，[Test It]をクリックすると，LINE アプリへ通知されます.

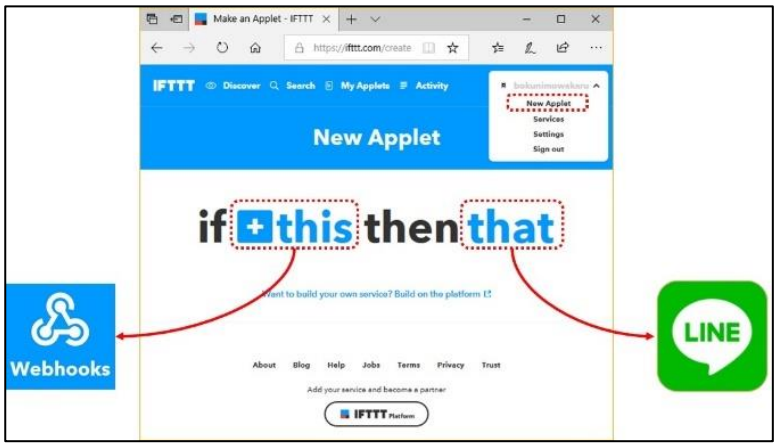

図 1-11 IFTTT によるクラウド連携の仕組み
「if this then that」の this に相当するトリガ入力に Webhooks を，that に相当するアクション出力に LINE を設定することで，Webhooks を LINE へ連携できる．IE などの古いブラウザでは利用できない

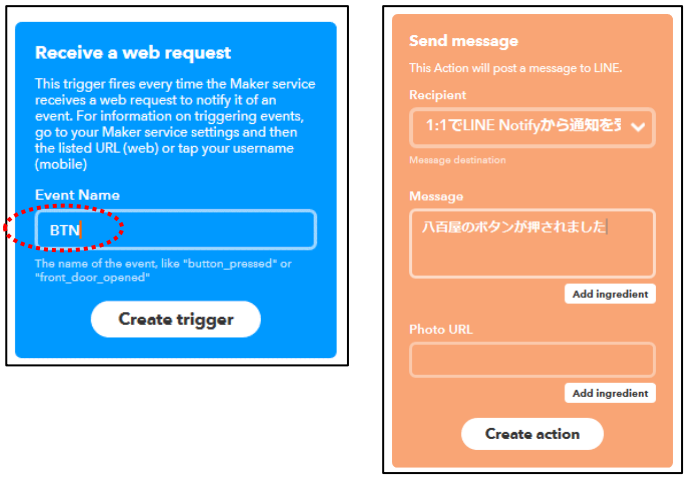

図 1-12 IFTTT へ Webhook と LINE の連携を設定する IFTTT

トリガ入力（左図）に Webhooks の設定を，アクション出力（右図）に LINE の設定を行ったときの一例

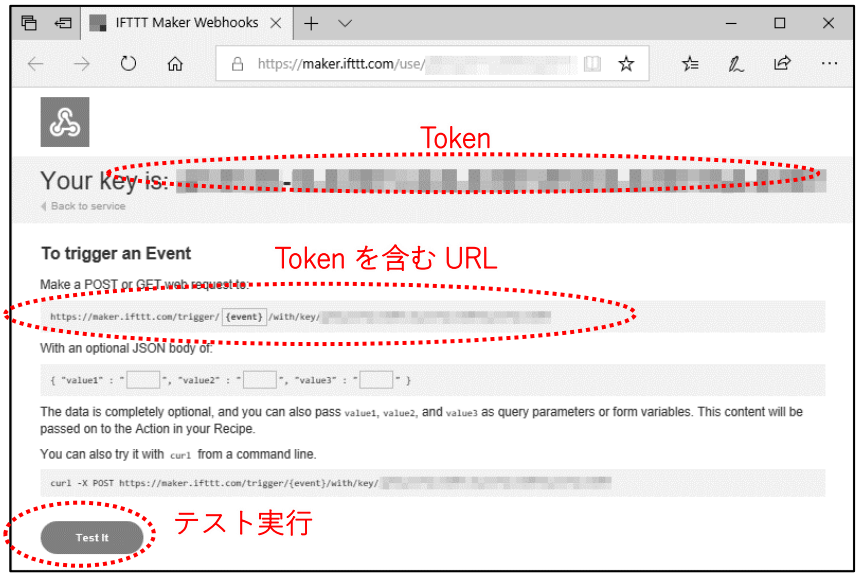

図 1-13 Webhook の Token を含む URL の表示画面

IFTTT の Webhook にアクセス（https://maker.ifttt.com/）し，Token を含む URL を取得する．{event}を BTN に変更し，[Test It]をクリックすれば，テスト実行も可能．

次に，sakura.io プラットフォームへ連携サービス Outgoing Webhook を追加します．初期設定時の WebSocket 追加と同様に sakura.io のコントロールパネルのプロジェクト画面から[詳細]→[連携サービス]→[サービス追加]と進んでください．ここで，「Outgoing Webhook」を選択すると，図 1-14 のような画面が表示されるので，IFTTT の Webhooks 用の Token を含む URL を入力します．

以上の設定が完了したら，製作した IoT 押しボタンもしくは IoT 人感センサから実際に送信して確認してみましょう．なお，IFTTT の that に相当するアクション（出力）には，LINE のほかに E-Mail や Gmail，Facebook，Twitter，google カレンダなどを設定することも出来ます．

図 1-14 sakura.io の連携サービス Outgoing Webhook に IFTTT の URL を設定する

プロジェクト画面から[詳細]→[連携サービス]→[サービス追加]と進み，[Outgoing Webhook]を選択すると表示される画面．Payload URL へ IFTTT の Webhooks の URL を入力する

Node-RED でデータを連携する

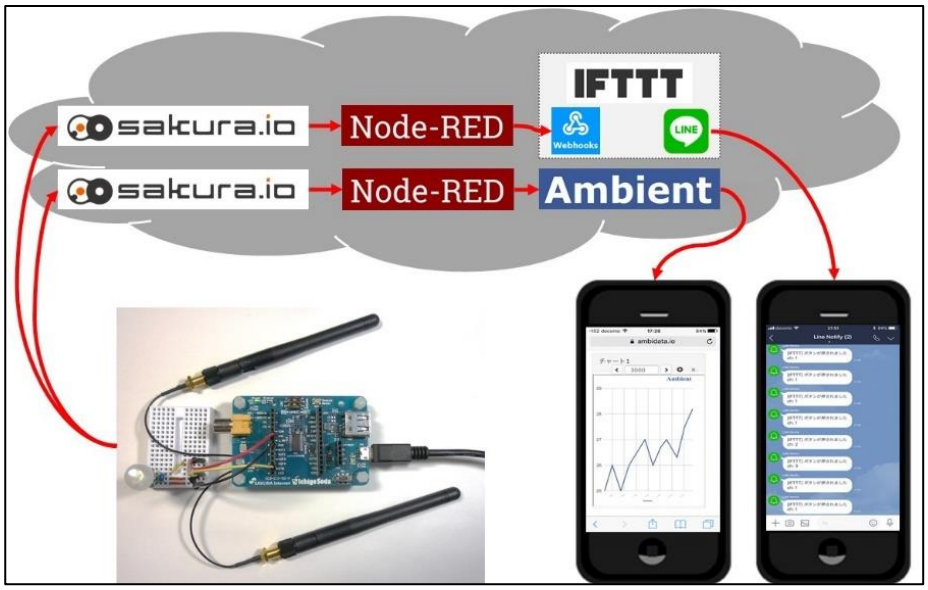

図 1-15 sakura.io からスマートフォンへのデータ連携例(IoT 押しボタン・IoT 人感センサ・IoT 照度センサ)
各種クラウドサービスを利用して，スマートフォンへ通知や測定結果を連携する様子.

次に，ボタン状態に応じた情報やセンサ値データなどを，他のクラウドサービスへ連携する方法について紹介します．IoT 押しボタンや IoT 人感センサでは，ボタンが押されたときや人体の動きを検知したときだけ LINE アプリへ通知します．また，次節の IoT 照度センサでは測定したデータをグラフ化する Ambient（https://ambidata.io/）へ照度値データを連携します．図 1-15 に連携例を示します．

Node-RED は，各種のクラウドサービスの橋渡しをするための入出力とデータ変換を行う，クラウド連携用ソフトウェアです．プログラミングに相当するデータの流れ（フロー）の作成は，ノードと呼ばれる機能モジュールの入出力を線で接続することで行います．IBM がオープンソースで配布しており，IBM Cloud など，Node.js が動作する様々な環境で動作します．

IBM Cloud で動かすには，下記へアクセスしてアカウントを作成後，Node-RED を起動します．

IBM Cloud：
https://console.bluemix.net/

Raspberry Pi で動かすには，「node-red-start &」を入力します．「コマンドが見つかりません」と表示された場合は，「sudo apt-get install nodered」でインストールしてください．起動時に表示される URL（一例：http://192.168.xx.xx:1880/）へ，インターネット・ブラウザでアクセスすると，図 1-16 のような Node-RED の初期画面が表示されます．

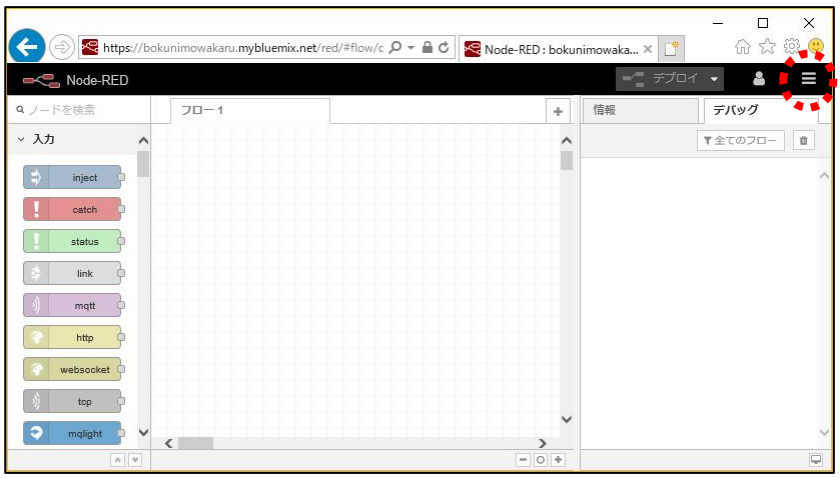

図 1-16 Node-RED（IBM Cloud）の初期画面の一例
アカウントを作成し，Node-RED を起動すると，右上にメニューボタンが表示される

IoT 押しボタン・IoT 人感センサ用のサンプル・フローを Node-RED へ取り込むには，下記から GitHub へアクセスし，ソースリスト右上の[Raw]ボタンをクリックし，キーボードの[Ctrl]＋[A]と，[Ctrl]＋[C]の操作でクリップボードへコピーしてから，Node-RED のメニューから[読込み]→[クリップボード]を選択し，[Ctrl]＋[V]でペーストします.

IoT 押しボタン・IoT 人感センサ用サンプル・フロー（Node-RED 用）：
https://goo.gl/7rpd7j (https://github.com/bokunimowakaru/IchigoJam/tree/master/node-red)
SAKURA_BTN_to_IFTTT.json

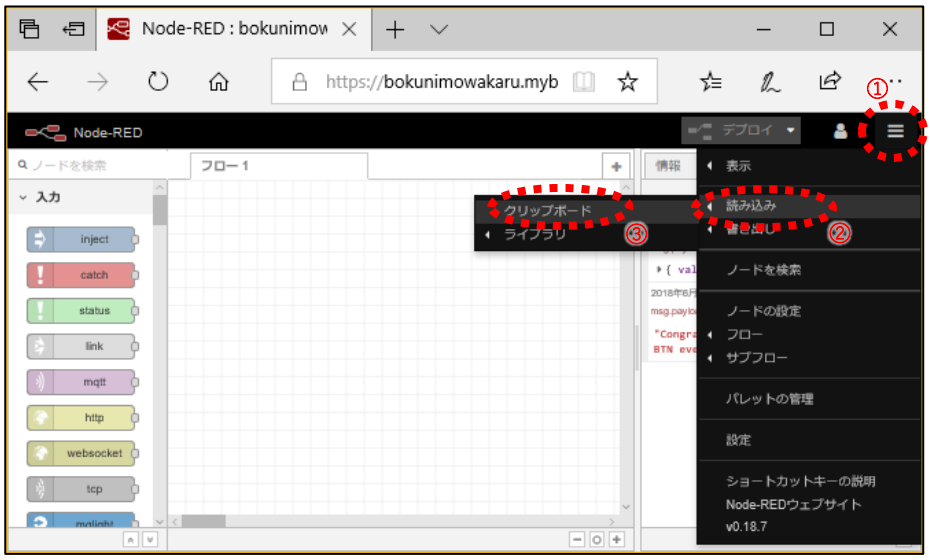

図 1-17 IBM Cloud の Node-RED のメニューからフローを読み込む
メニューボタン（①）→[読込み]（②）→[クリップボード]（③）の順にクリックする

サンプル・フローのノードに sakura.io の設定情報を入力するには，Node-RED 画面上の sakura.io ノードをダブルクリックし，sakura.io の WebSocket の Token を入力してください．また，IFTTT の Webhooks の Token をノード http request へ入力してください.

ノードの設定が完了したら，画面右上の[デプロイ]ボタンをクリックすると，フローが開始され，sakura.io ノードが「connected」に変わります．また，右側のウィンドウで[デバッグ]を選択すると，受信データを IFTTT へ送信し，送信に成功した様子を確認することが出来ます.

なお，画面左側のノードをサンプル・フローへ変更することで，通知する条件を変更したり，センサ値を補正したりすることも出来ます. フローを変更した場合は，再度，[デプロイ]を実行してください.

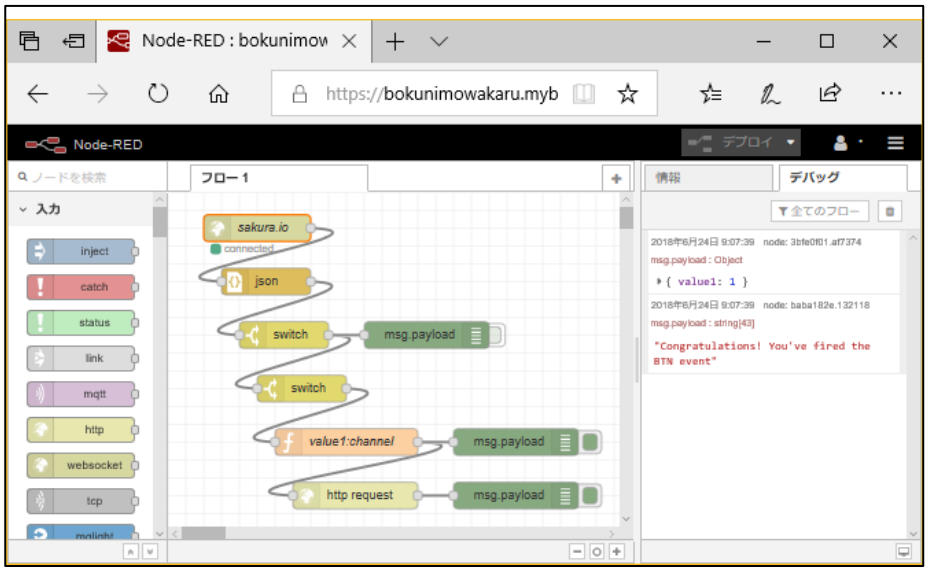

図 1-18 IBM Cloud の Node-RED の実行画面の一例
sakura.io から受けた IoT 押しボタンや IoT 人感センサからの情報を，IFTTT(Webhooks)に連携する処理の実行の様子

IoT センサ値を Ambient へ蓄積＋グラフ化表示

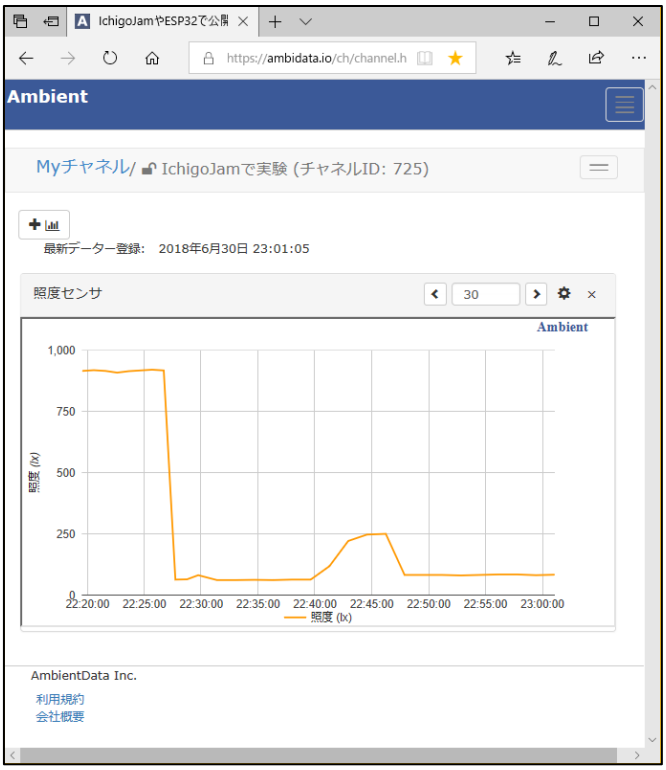

図 1-19　Ambient へ照度値を送信し，グラフ化して表示したときの様子

IchigoSoda が LTE で送信した照度値を，Node-RED を経由して IoT 用クラウドサービス Ambient でグラフ化表示した．データを送信するたびにグラフが自動的に更新される（IE などの古いブラウザでは更新されない）

IoT 照度センサから得られた測定値を，IoT 用クラウドサービス Ambient へ送信することで，照度値の蓄積やグラフ表示を行うことが出来ます．Ambient（https://ambidata.io/）は，アンビエントデータ株式会社が運営する IoT センサ用のクラウドサービスです．PC やスマートフォンのインターネット・ブラウザ（Edge，Chrome，Safari）からアクセスすることが出来ますが，IE など古いブラウザではグラフが更新されないなどの支障があります．

[ユーザ登録(無料)]のボタンからユーザ登録を行い，ログインすると図 1-20 のような My チャネル画面が開くので，[チャネルを作る]を実行してください．

図中の①チャネル ID は，センサ機器 1 台毎に割り当てられた番号です．②ライトキーは，センサ情報をクラウドへアップロードするときに使用する認証キーです．

Node-RED から Ambient へ連携するには，Ambient 専用ノードが便利です．図 1-21 のように，Node-RED 画面の右上のメニューアイコン（3 本の横線）をクリックし，[パレットの管理]を選択すると，図 1-22 のような「ユーザの設定」画面が開きます．続いて，[ノードを追加]タブを選択し，検索窓に「ambient」を入力し，「node-red-contrib-ambient」が表示されたら[ノードを追加]をクリックしてください．追加には，1 分くらいの時間を要することもあります．

IoT 照度センサ用のフロー図は，Node-RED のメニューから[読込み]→[クリップボード]を選択し，下記からクリップボード経由で Node-RED へ取り込んでください．

IoT 照度センサ用サンプル・フロー（Node-RED 用）：
https://goo.gl/7rpd7j
SAKURA_ILM_ANA_to_Ambient.json

IoT 照度センサ用のフロー図では，sakura.io ノードへ sakura.io の Token を，Ambient ノードへ Ambient のチャネル ID と Write Key を入力してください．

Node-RED から Ambient へ連携して，照度値を表示したときの様子を図 1-19 に示します．

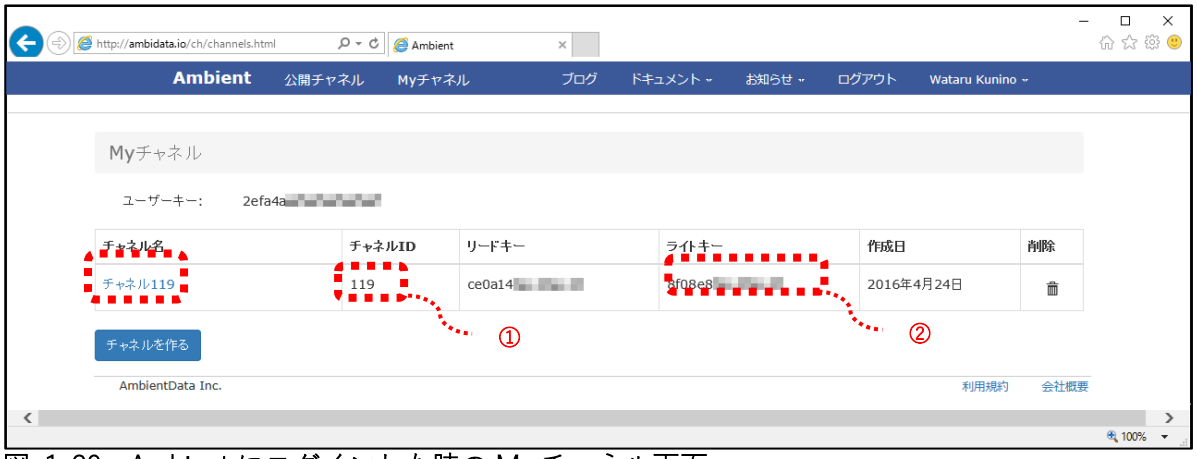

図 1-20　Ambient にログインした時の My チャネル画面

ログイン時に表示される My チャネル画面で，①チャネル ID と②ライトキーを確認する

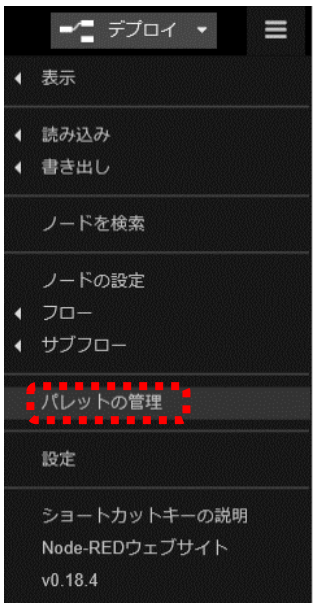

図 1-21　Node-RED のメニューから[パレットの管理]を選択する

図 1-22　[パレットの管理]メニューでタブ[ノードを追加]を選択し，検索窓に「ambient」を入力すると node-red-contrib-ambient が見つかるので，[ノードを追加]ボタンをクリックする

[Column] IchigoJam BASIC 1.2.4 IoT 版で使用可能な IoT コマンド（2021 年 2 月 7 日・追記）

IchigoJam BASIC のバージョン 1.2.4 IoT 版以降で，sakura.io モジュール専用の IoT コマンドが使えるようになりました．

IOT.OUT コマンドは，0～65535 までの数値を LTE 送信するコマンドです．「IOT.OUT 123」のように入力だけで，送信することが出来ます．LTE 対応 IoT 押しボタンのプログラムは，リスト 1-6 のように簡単に書けるようになります．

IOT.IN()関数は，数値を受信するコマンドです．「? IOT.IN()」を入力すると受信結果が表示されます．受信していないときは 0 が得られるので，受信値が 0 のときは区別がつきません．

これらの IOT コマンドは，プログラムを短くする際に便利なので，各リストの下部に記載したファイル名やメニュー番号に，100 を加算して配布しています．例えば，11.txt であれば，111.txt でダウンロードして下さい．

IchigoSoda に書き込まれている IchigoJam BASIC のバージョンによって，IOT.IN()関数の使い方が異なります．文字列を受信したときの文字データの格納アドレスが，バージョン 1.2.4 IoT 版では #824，バージョン 1.3.1 以降は#114A になりました．また，IOT.IN()関数の戻り値が，バージョン 1.4.1 以降は文字データ格納アドレスとなりました．

表 1-9　IchigoJam BASIC 1.2.4 IoT 版でサポートされる IoT コマンド

命令	使用例	内容
IOT.OUT	IOT.OUT 12345	0～65535 までの数値を sakura.io モジュールで LTE 送信する
IOT.IN	? IOT.IN()	sakura.io モジュールが受信した整数値を取得する
	? STR$(IOT.IN())	sakura.io モジュールが受信した文字列(8 文字まで)を取得する（IchigoJam BASIC 1.4.1 以降）
	IF IOT.IN() POKE #1152,0: ? STR$(#114A)	sakura.io モジュールが受信した文字列(8 文字まで)を取得する（IchigoJam BASIC 1.3.1 以降）
	IF IOT.IN() POKE #82C,0: ? STR$(#824)	sakura.io モジュールが受信した文字列(8 文字まで)を取得する（IchigoJam BASIC 1.2.4 IoT 版のみ）

リスト 1-6　IchigoJam BASIC 1.2.4 IoT 版・LTE 対応 IoT 押しボタン

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"SAKURA IoT TX BTN/PIR	1 画面へ「SAKURA IoT TX BTN/PIR」を表示
2 ?"CC BY Wataru Kunino	2 画面へ「CC BY Wataru Kunino」を表示
3 ?"ﾎﾞﾀﾝ ﾆｭｳﾘｮｸ B=BTN	3 画面へ「ﾎﾞﾀﾝ ﾆｭｳﾘｮｸ B=BTN」を表示
100 ?"== ｼﾞｭﾝﾋﾞ ==	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
150 B=btn()	150 変数 B へボタン SW2 の状態を代入
200 ?"== ﾒｲﾝ ==	200 画面へ「== ﾒｲﾝ ==」を表示
210 ?"B=";B	210 画面へ変数 B の内容を表示
220 IoT.out B	220 画面へ変数 B の内容を LTE 送信
300 ?"== BTN ﾏﾁ ==	300 画面へ「== BTN ﾏﾁ ==」を表示
310 wait 30	310 約 0.5 秒の待ち時間処理
320 if B=btn() cont	320 ボタン SW2 の状態に変化があるまで待機
330 B=!B	330 変数 B の値を反転（0→1，1→0）
340 goto 200	340 行番号 200（メイン）へ戻る
・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/111.txt **・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[111]）**	

製作したLTE対応プログラム

筆者が作成したIchigoSodaおよびsakura.ioモジュール用LTE対応プログラムを表 1-10に示します．パソコンやMixJuiceでダウンロードすることが出来ます．

パソコンでダウンロードする場合は，表の下段に書かれたURLの「●●」の部分にファイル名の数字を入れて，ブラウザでアクセスしてください．文字化けする場合は，ブラウザの文字エンコード設定を[シフト JIS]に設定して下さい．ダウンロードしたプログラムは，USB シリアル変換アダプタ経由でIchigoSodaへ転送することができます．

MixJuiceでダウンロードする場合は表の下段に書かれたIchigoJam BASICコマンドをIchigoJamへ入力してください．

ファイル名100番台はコラムに示したIchigoJam BASIC 1.2.4 IoT版用のプログラムです．IOT.OUTとIOT.INコマンドを使用しました．

表 1-10　製作したLTE対応プログラムの一覧

ファイル名		プログラム名	備考
通常版	IoT版		
11.txt	**111.txt**	SAKURA IoT TX BTN/PIR	LTE対応IoT押しボタン・人感センサ・送信用
12.txt	**112.txt**	SAKURA IoT TX BTN/PIR PS	LTE対応IoT押しボタン・人感センサ・省電力送信用
13.txt	**113.txt**	SAKURA IoT TX ANA PS	LTE対応IoTアナログセンサ用
14.txt	**114.txt**	SAKURA IoT TX ILM PS	LTE対応IoT照度センサ用
15.txt	**115.txt**	SAKURA IoT RX	LTE対応IoTディスプレイ端末用
16.txt	**116.txt**	SAKURA IoT RX OLED	LTE対応IoTディスプレイ端末用・OLED版
17.txt	**117.txt**	SakuraIoRxTxtLCD	LTE対応IoTディスプレイ端末用・LCD版
18.txt	**-**	SAKURA Firmware Updater	sakura.ioモジュールのファームウェア更新用
10.txt	**110.txt**	SAKURA IoT YaoYa Talk	LTE対応IoT音声アナウンス端末用
・PC等でダウンロード：　http://git.bokunimo.com/MJ/pg08/●●.txt ・MixJuiceメニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"			

【参考情報】(2021年2月7日)

本書を執筆した2018年8月時点では，IOT.OUTコマンドとIOT.IN()関数が正式リリースされていなかったため，紙面上のサンプル・プログラムではI2CW関数を使用しています．

動作に変わりはありませんが，必要に応じてIOT.OUT，IOT.IN()を使ったファイル（上表の「IoT版」）をダウンロードしてご利用ください．

改訂については，本書のダウンロード件数などを考慮したうえで実施する予定です．

2章 MixJuice を使って Wi-Fi 搭載＋省エネ運転対応の IoT センサを製作してみよう

本章では，IchigoJam S/T/U へ IchigoJam 用 Wi-Fi ネットワーク拡張ボード MixJuice を接続し，Wi-Fi 対応 IoT 押しボタンや IoT センサを製作する方法について説明します．また，応用例として，単 3 アルカリ乾電池で長時間動作が可能な省電力動作用のプログラムについても紹介します．

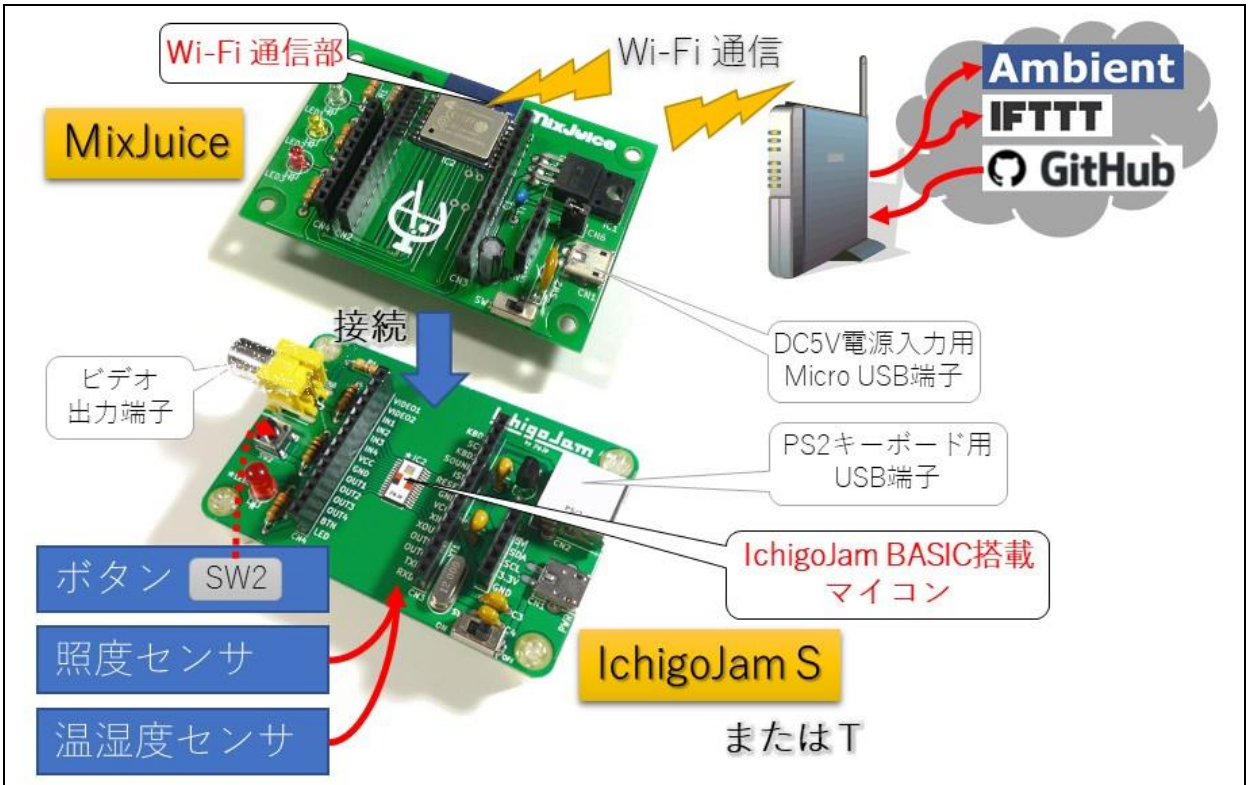

表 2-1　MixJuice を使った IoT デバイス例

通信方式	Wi-Fi (2.4GHz)
通信エリア	住宅内・居室内・倉庫内など
通信料金	無料（ネット接続料別）
機器コスト	4,000 円～
IoT 機能例	ボタン，人感，温度，OLED 表示など

表 2-2　MixJuice の省電力特性の一例

状態	全体	MixJuice
通常状態	94 mA	79 mA
ディープ･スリープ	0.9 mA	0.6 mA

5V 入力での実測値

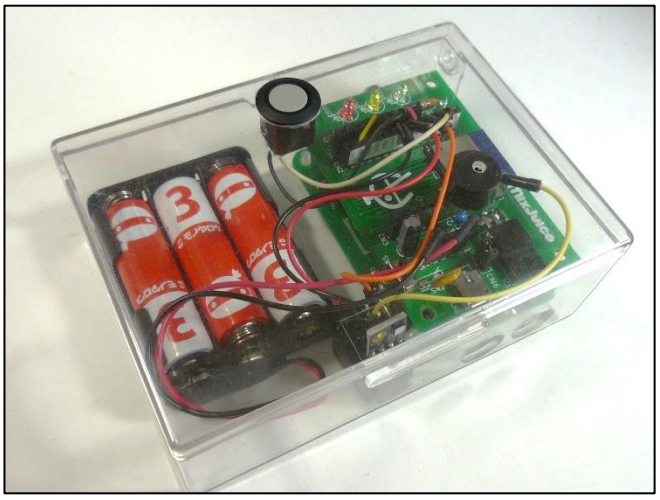

写真 2-1 Wi-Fi ネットワーク拡張ボード MixJuice を使った実験時の様子

IchigoJam T 上に MixJuice を接続し，乾電池で動作する Wi-Fi 対応 IoT 押しボタンや，IoT センサを製作した．ボタンを押すと，LINE へメッセージを送信したり，温度や湿度のグラフをスマートフォンで閲覧したりすることができる．

Wi-Fi 対応 IoT 押しボタン・IoT 照度センサの製作

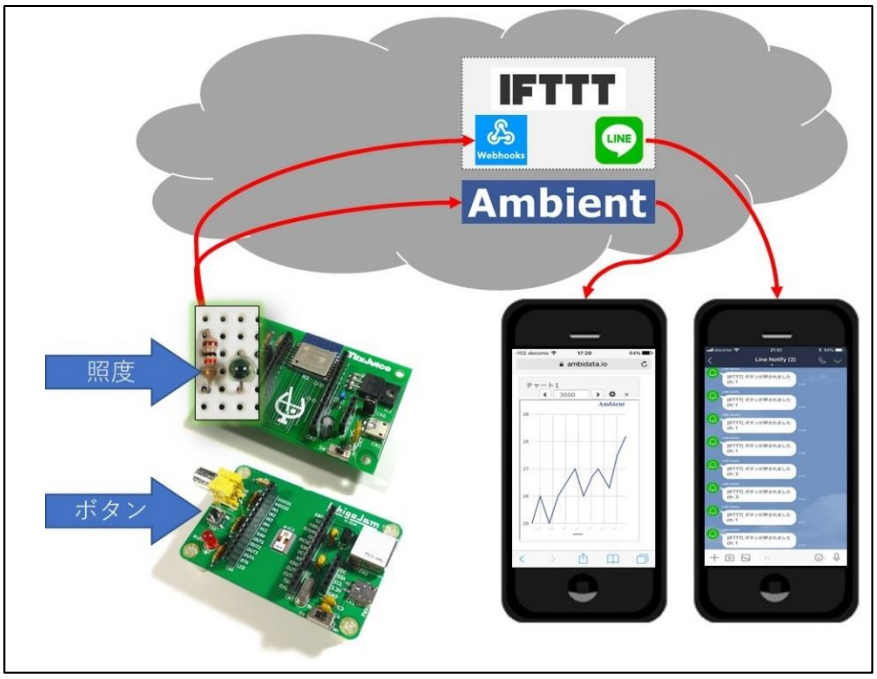

図 2-1　クラウドと連携する IoT 押しボタンと IoT 照度センサ
ボタンを押すと，LINE へメッセージを送信したり，温度や湿度のグラフをスマートフォンで閲覧したりすることが出来る

本章では，乾電池で動作する Wi-Fi 対応 IoT 押しボタンや，IoT センサを製作します．ボタンを押すと，LINE へメッセージを送信したり，温度や湿度のグラフをスマートフォンで閲覧したりすることができるようになります．

Wi-Fi 対応 IoT 押しボタン・IoT 照度センサの機器構成

Wi-Fi 対応 IoT 押しボタンと IoT 照度センサを製作するための構成例を表 2-3 に示します．マイコン・ボード IchigoJam S や，T，U の上に MixJuice が重なるように接続するだけで，簡単にハードウェアを準備することが出来ます．また，周辺機器として DC5V 出力の Micro USB 用 AC アダプタが必要です．開発時には，USB コネクタつき PS2 キーボード，ビデオ入力端子付きテレビも必要です．

表 2-3　MixJuice を使った IoT デバイスの機器構成例

品目	機器名・仕様	開発元	販売元	参考価格	備考
マイコン・ボード	IchigoJam S/T/(U)	jig.jp	マルツエレック，PCN	1,500 円	完成品は 2000 円
Wi-Fi 拡張ボード	MixJuice	NaturalStyle	マルツエレック，PCN	2,500 円	完成品
照度センサ	NJL7502L	新日本無線	マルツエレック，秋月	45 円	
抵抗器	10kΩ 1/4W	汎用品	マルツエレック，秋月	1 円	100 本単位で販売

MixJuice は，PCN 社が販売する Wi-Fi ネットワーク拡張ボードです．ディープ・スリープ機能を搭載しているので，乾電池で長期間動作が可能な省エネ運転対応 IoT センサを製作することが出来ます．

- Wi-Fi 対応 IoT 押しボタン

IchigoJam S/T/U 上のボタン SW2 を押したときに Wi-Fi 送信する IoT 押しボタンを製作します．ボタン SW2 は，MixJuice を接続すると隠れるので，横から指を入れて押します．このとき，部品やリード線などでケガをしないように注意してください．写真 2-1 のように IchigoJam S/T/U の CN4 の BTN 端子へプッシュ・スイッチを接続すると良いでしょう．

- Wi-Fi 対応 IoT 照度センサ

照度値を Wi-Fi 送信することが出来る IoT 照度センサを，写真 2-2 のように照度センサと抵抗を接続して製作します．IchigoJam の VCC 端子を照度センサのコレクタ（リード線の長い方）へ接続し，エミッタ側は，IchigoJam の IN2 端子に接続するとともに，抵抗を経由して GND へも接続してください．

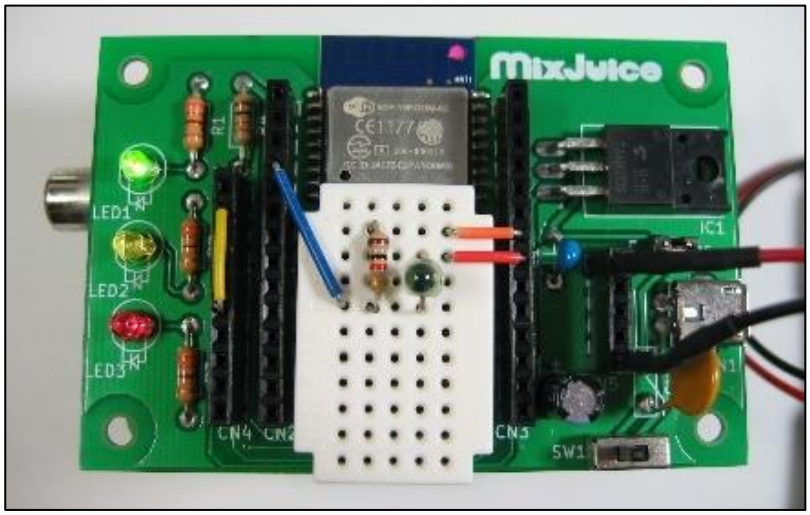

写真 2-2 Wi-Fi ネットワーク拡張ボード MixJuice で製作した IoT 照度センサ

IchigoJam S/T/U の VCC 端子から照度センサのコレクタ（リード線の長い方）へ接続し，エミッタ側は，抵抗を経由して GND へ接続するとともに，IchigoJam の IN2 端子へ入力する．

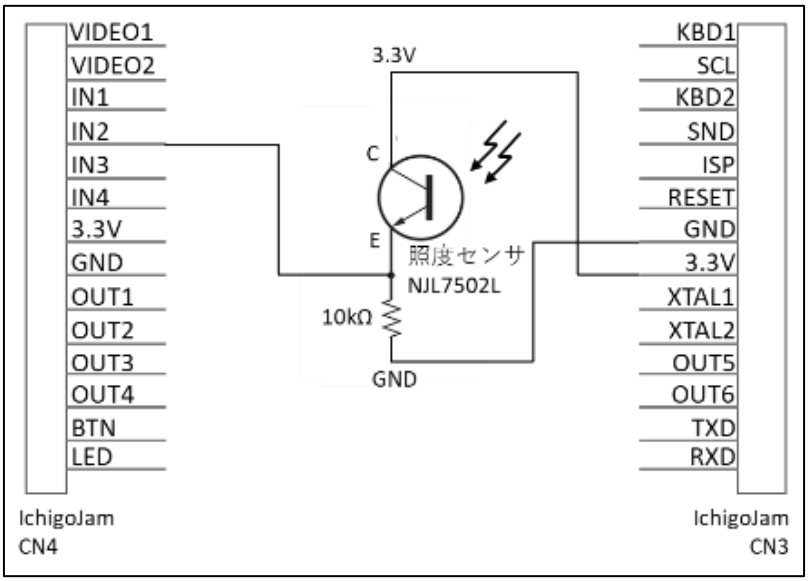

図 2-2　照度センサの接続回路図例

照度センサに光があたると，コレクタからエミッタに電流が流れ，10kΩの抵抗の両端に電圧が生じる．この電圧を IchigoJam マイコンの IN2 端子へ入力する

- IchigoJam と MixJuice に電源を供給する

DC 5V 電源入力用の Micro USB 端子は, IchigoJam S/T と MixJuice の両方にありますが, MixJuice に付属のショート・ピンを MixJuice の CN6 へ取り付けることで，どちらか一方に接続した電源が両方のボードへ供給されます．ただし，IchigoJam U の場合は，Micro USB 端子 CN1 のポリスイッチの影響で電流容量が不足するので，MixJuice 側の Micro USB 端子に電源を供給してください．

以上のハードウェアの組み立てと周辺機器の接続を終えたら，IchigoJam BASIC が起動します．テレビ画面に無意味な文字が表示されることがありますが，Wi-Fi モジュールの起動メッセージが（ビットレートの相違で）文字化けしたものです．キーボードの[F1]キーで消去してください．

MixJuice の Wi-Fi アクセスポイント接続

MixJuice をインターネットへ接続するには，IchigoJam T へ以下のコマンドを入力して Wi-Fi アクセスポイントへ接続します．<SSID>と<PASS>の部分には，お手持ちの無線 LAN アクセスポイントの SSID とパスワード（認証キー）を入力してください．

```
?"MJ　APC　<SSID>　<PASS>"⏎
```

SSID やパスワードの大文字と小文字は区別されます．小文字はシフトキーを押しながら入力してください．SSID にスペース文字が含まれているときは，スペース文字の直前に「¥」記号を付与します．

Wi-Fi へ接続すると，DHCP によって割り当てられた IP アドレスが表示されます．SSID とパスワードは MixJuice 内に記憶されるので，次回からは自動的に Wi-Fi アクセスポイントへ接続します．

筆者が作成したプログラムの一覧を表示するには，以下を IchigoJam へ入力してください．URL の多くは小文字ですが，MJ は大文字です．ファイル名の「808.txt」の数字は「八百屋」を意識しました．

```
?"MJ　GET　git.bokunimo.com/MJ/808.txt"⏎
```

プログラム[21]をダウンロードするには，キーボードから[2][1]と[Enter]を順に入力してください．また，ダウンロードが終わったら，[Enter]キーを押すことで，コマンド待ち状態に戻ります．

```
?"MJ APC <SSID> <PASS>"⏎
MJ APC <SSID> <PASS>
OK
'...............
'WiFi connected: 192.168.1.15
'OK
?"MJ GET git.bokunimo.com/MJ/808.txt"⏎
```

図 2-3 Wi-Fi アクセスポイントに接続する
<SSID>と<PASS>の部分に，お手持ちの無線 LAN アクセスポイントの SSID とパスワード（認証キー）を入力して接続する

```
' IchigoSoda + sakura.io
' [10]YaoYa Talk
' [11]BTN/PIR [12]BTN/PIR PS
' [13]ANA [14]ILM
' [15]RX [16]OLED [17]LCD
' [18]Firm [19] Firmｺﾏﾝﾄﾞ
'
' MixJuice -> Ambient
' [20]TX [21]BTN/PIR [22]B/P PS
' [23]ANA2/ILM PS [24]ANA2/I PS2
' [25]Envｾﾝｻ
'
' MixJuice -> IFTTT
' [26]BTN/PIR [27]BTN/PIR PS
'
' LoRaWAN AL-050
' [31]TX [32]ANA [33]RX [34]TXT
' [30]Barcode [37]LCD
'
' [99]ﾘﾀｰﾝ
inputA:?"MJ GET git.bokunimo.com/MJ/pg08/";A;".txt"
?■
```

図 2-4 筆者作成のプログラム集へ MixJuice でアクセスしたときの様子
ダウンロードしたい番号を入力し，Enter キーを押下するとダウンロード出来る．MixJuice 用のプログラムは 20 番台に収録した．

Wi-Fi テスト送信プログラム

まずは，Wi-Fi のテストを行います．IoT 用クラウドサービス Ambient（https://ambidata.io/）へテスト送信を行い，PC やスマートフォンのブラウザで，送信結果を確認してみましょう．

Wi-Fi テスト送信プログラムをリスト 2-1 に示します．実行する前に，Ambient から取得したチャネル ID とライトキーを，行番号 2 と 3（手順①と②）の「xxx」の部分に入力してください．

図 2-5 は，RUN を 10 回，実行したときのテスト結果の一例です．RUN コマンドを実行するたびに，1 または 0 の値を交互に Ambient へ送信します．

リスト 2-1　Wi-Fi 対応 MixJuice 送信テスト用プログラム・トグル送信

BASIC プログラム	解説
new 1 ?"MixJuice IoT TX　　① 2 C=xxxx 3 K="xxxxxxxxxxxxxxxx" 4 D=1　　② 100 ?"MJ PCT application/json" 110 B=!B 120 ?"MJ POST START ambidata."; 130 ?"io/api/v2/channels/";C; 140 ?"/data" 150 ?"{";chr$(34);"writeKey"; 160 ? chr$(34);":";chr$(34); 170 ? str$(K);chr$(34);","; 180 ? chr$(34);"d";D;chr$(34); 190 ?":";B;"}" 200 ?"MJ POST END" 210 wait 300:clk:end	new=現在のプログラムを消去する 1 画面へ「MixJuice IoT TX」を表示 2 変数 C へ Ambient のチャネル ID を入力 3 変数 K へ Ambient のライトキーを入力 4 変数 D へ Ambient 用データ番号を入力 100 MixJuice の POST データ形式を設定する 110 送信データ用変数 B の値を反転する 120 MixJuice へ HTTP POST を設定 130 Ambient チャネル番号を指定 140 Ambient の URL を MixJuice へ送信 150 JSON の項目に Ambient ライトキー 160 区切り文字を送信 170 変数 K のライトキーを設定 180 変数 D のデータ番号を設定 190 変数 B の送信データを設定し送信 200 HTTP POST データの終了と，HTTP 実行 210 待ち時間処理(5 秒)．受信データの破棄
・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/20.txt **・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[20]）**	

図 2-5 MixJuice から Ambient へ送信テストを行ったときの様子

テスト送信プログラムを実行するたびに，0 または 1 を交互に送信する．

Wi-Fi 対応 IoT 押しボタン/人感センサのプログラム解説

IchigoJam 上のボタン SW2 の操作に合わせて送信を行うプログラムをリスト 2-2 に示します．写真 2-3 のようにボタン SW2 が押しにくいので，写真 2-4 のようにプッシュ・スイッチを SW2 と並列に接続すると使いやすくなります．あるいは，人感センサ・モジュール SB412A を MOS FET で論理反転した信号を BTN 端子へ入力することで，IoT 人感センサを製作することも出来ます．

プログラムは，MixJuice に関する準備部と，送信部，受信部を，行番号 200 番台のメイン部から分離することで，改変しやすいように考慮しました．受信部は，Ambient 以外の他のクラウドサービスへ連携するときや，コマンドやパラメータを変更するときに，送信結果を表示しながらデバッグを行う際に有用となるでしょう．以下に，主要な動作の手順について説明します．

① ～② 変数 C へ Ambient のチャネル ID を，変数 K へライトキーを代入します．
③ 変数 D は Ambient の各チャネル内のデータ番号です．1～8 の整数を代入します．
④ HTTP POST 時のデータ形式を JSON に設定するコマンドを MixJuice へ送信します．
⑤ ボタン状態が代入されている変数 B の内容を表示します．
⑥ 送信用サブルーチン処理⑨を実行します．
⑦ IchigoJam 上のボタン SW2 が変化するまで待機します．
⑧ ボタン状態が変化すると，行番号 200 番台の繰り返しのメインループ処理を再開します．
⑨ 変数 B の値を MixJuice から Ambient へ送信します．
⑩ MixJuice からの応答データをテレビ画面へ表示します．

写真 2-3 MixJuice 基板と接続すると IchigoJam のボタン SW2 が押しにくくなる
IchigoJam のボタン SW2 が MixJuice 基板との間に位置するので，ボタンを押すときに指をケガしないように注意する

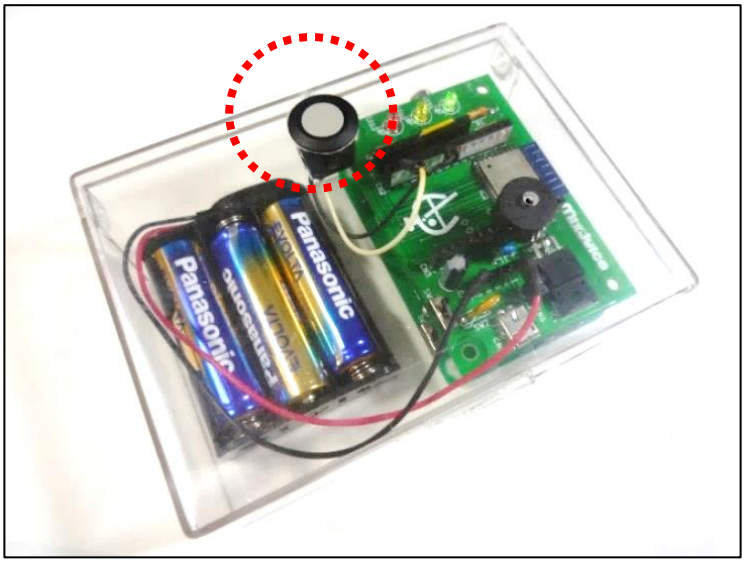

写真 2-4 ボタン SW2 と並列にプッシュ・スイッチを取り付けたときの様子
IchigoJam T 上に MixJuice を接続し，乾電池で動作する Wi-Fi 対応 IoT 押しボタンを製作した．乾電池を長持ちさせる方法は次節で説明する．

リスト 2-2　Wi-Fi 対応 IoT 押しボタン/IoT 人感センサ MixJuice IoT BTN/PIR

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"MixJuice IoT BTN/PIR	1 画面へ「MixJuice IoT BTN/PIR」を表示
2 uart 0:?"CC BY ｸﾆﾉ ﾜﾀﾙ ①	2 シリアル送信を OFF に設定. 権利表示
3 C=xxxx:?" Ambient ﾁｬﾈﾙ ID=";C	3 変数 C へ Ambient のチャネル ID を入力
4 K="xxxxxxxxxxxxxxxx":'ﾗｲﾄ ｷｰ ②	4 変数 K へ Ambient のライトキーを入力
5 D=1:?" ﾃﾞｰﾀ=d";D ③	5 変数 D へ Ambient 用データ番号を入力
100 ?"== ｼﾞｭﾝﾋﾞ =="	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
110 uart 1:clk	110 シリアル送信を ON に設定. 受信データ破棄
120 ?"MJ PCT application/json" ④	120 MixJuice の POST データ形式を設定する
130 gosub 700	130 受信データ表示サブルーチン(700 行)へ
140 B=btn()	140 ボタン状態を変数 B へ代入
200 ?"== ﾒｲﾝ =="	200 画面へ「== ﾒｲﾝ ==」を表示
210 ?"B=";B ⑤	210 変数 B(ボタン状態)を表示
220 gosub 500	220 Wi-Fi 送信用のサブルーチン(500 行)へ
230 wait 200 ⑥	230 連続送信を避けるための待ち時間(3.3 秒)
240 ?"== BTN ﾏﾁ =="	240 画面へ「== BTN ﾏﾁ ==」を表示
250 if B=btn() cont ⑦	250 ボタン状態の変化待ち
260 B=!B	260 現在のボタン状態を変数 B へ代入
270 goto 200 ⑧	270 行番号 200 へ戻る
500 '== TX ==	500 画面へ「== TX ==」を表示
510 uart 1:clk	510 シリアル送信を ON に設定. 受信データ破棄
520 ?"MJ POST START ambidata.";	520 MixJuice へ HTTP POST を設定
530 ?"io/api/v2/channels/";C;	530 Ambient チャネル番号を指定
540 ?"/data"	540 Ambient の URL を MixJuice へ送信
550 gosub 700:uart 1	550 受信サブルーチン(700 行)へ
560 ?"{";chr$(34);"writeKey"; ⑨	560 JSON の項目に Ambient ライトキー
570 ? chr$(34);":";chr$(34);	570 区切り文字を送信
580 ? str$(K);chr$(34);",;	580 変数 K のライトキーを設定
590 ? chr$(34);"d";D;chr$(34);	590 変数 D のデータ番号を設定
600 ?":";B;"}"	600 変数 B のボタン状態を設定し送信
610 ?"MJ POST END"	610 HTTP POST データの終了と, HTTP 実行
620 gosub 700	620 受信サブルーチン(700 行)へ
630 return	630 サブルーチンを終了し, gosub 部へ戻る
700 '== RX ==	700 受信部(RX)を示すコメント
710 uart 0:clt	710 シリアル送信を OFF に. タイマー初期化
720 I=inkey()	720 シリアルデータを受信
730 if I clt:? chr$(I); ⑩	730 データがあれば, 表示する
740 if tick()<40 goto 720	740 データ待ち時間 0.67 秒以下なら 720 行へ
750 return	750 サブルーチンを終了し, gosub 部へ戻る

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/21.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　(メニュー[21])

IFTTT 対応版は下記からダウンロード可能

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/26.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　(メニュー[26])

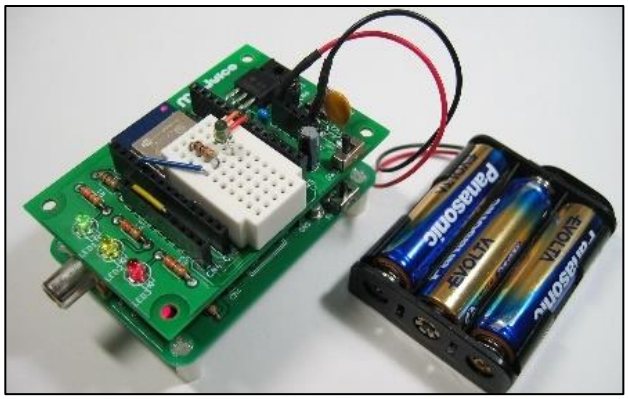

写真 2-5 省エネ運転機能を用いることで，乾電池による長期間駆動が可能になる．
単 3 アルカリ乾電池を使って約 3 か月の連続動作が可能な Wi-Fi 対応 IoT 照度センサ

Wi-Fi 対応 IoT デバイスの乾電池による長期間駆動を実現する省エネ運転方法について説明します．例えば，単 3 アルカリ乾電池で 3 か月の動作を確保するには，平均の消費電流を 1mA 以下にする必要があります．前章の IchigoSoda と sakura.io モジュールとの組み合わせた IoT 機器では，通信やマイコンを停止するディープ・スリープ・モードであっても 6mA の消費電流を必要とするので，0.5 か月程度で乾電池が来ててしまいます．一方，IchigoJam と MixJuice との組み合わせた IoT 機器では，ディープ・スリープ・モード時に 1mA 以下で待機することが出来るので，通信頻度を控えることで，乾電池による長期間駆動を実現することが出来ます．

IchigoJam には，表 2-4 に示す 3 つの低消費電力駆動方法があります．表中の「低クロック駆動」は，テレビ出力の停止と，マイコンのクロック周波数を下げることで低消費電力化を行う方法です．プログラムの最初に設定コマンドを追加するだけで手軽に使用することが出来ます．元に戻すには，[F8]キーまたは「VIDEO 1」を入力します．低消費電力動作時にプログラムを動かし続けることが可能ですが，その一方で，消費電力が 1/3 程度にしかなりません．

次の「Cyclic Sleep 駆動」は，プログラムの途中で指定した時間だけ IchigoJam をスリープさせる方式で，主に温度センサなどの読み値を定期的に送信するときに使用します．スリープ中の消費電力を 1/50 程度に抑えることが出来ますが，プログラムが一時停止します．プログラムの実行位置，変数の内容，GPIO の出力値は保持されており，スリープ時間が経過すると，中断した位置から実行を再開します．

最後の「Single Shot 起動」は，プログラムの末尾に SLEEP コマンドを追加し，プログラム終了とともに IchigoJam をスリープさせる方式です．スリープ中は，最も低消費電力な状態となりますが，プログラムの実行位置や変数の内容が消失し，GPIO の出力値や入力インピーダンスが変化します．ボタンが押されたときや人感センサが反応したときに BTN 端子を Low レベルに設定することで，ファイル番号 0 のプログラムを自動的にロードし，プログラムの先頭行から実行します．

表 2-4　IchigoJam BASIC による IchigoJam と MixJuice の省エネ運転方法の一覧

対象	省電力方法	コマンド例	内容（動作条件）
MixJuice	ディープ・スリープ・モード（0.6mA）	?"MJ SLEEP 50" ?"MJ SLEEP 0"	MixJuice をスリープ状態に設定します．引数は再開させるまでの時間（秒単位）です．
IchigoJam	低クロック駆動（約 5mA）	VIDEO 0,8 (設定) VIDEO 1(解除)	ビデオ出力を停止し，マイコンの動作クロックを下げて省電力化（約 5mA）
	Cyclic Sleep 駆動（約 0.3mA）	WAIT 300,0	指定時間（1/60 秒単位）スリープしてから，自動復帰
	Single Shot 起動（約 20μA）	SLEEP	BTN 端子が H レベルでスリープへ遷移． BTN 端子が H→L レベル遷移時に復帰．

省エネ運転機能①Single Shot 起動

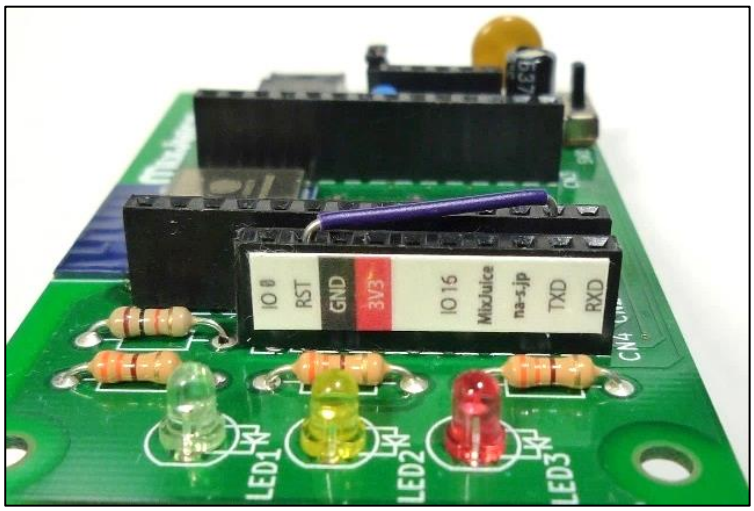

写真 2-6 Single Shot 起動方式の省電力動作では，IchigoJam の BTN 端子を MixJuice の RST 端子へ接続

IchigoJam の BTN 端子を, MixJuice の RST 端子へ接続することで，ボタン SW2 押下時や人感センサの検知時に MixJuice を起動する

まずは，最も低待機電力にすることが出来る Single Shot 起動方式を使った Wi-Fi 対応 IoT 押しボタンのハードウェアとプログラムについて説明します.

スリープを実行する手順は簡単です. MixJuice を MJ SLEEP コマンドでスリープに遷移させてから，IchigoJam BASIC の SLEEP コマンドを実行することで，待機状態に遷移させることが出来ます.

一方, スリープから復帰するときは, BTN 端子と MixJuice の RST 端子を操作する必要があります. 今回は，写真 2-6 のように，MixJuice の CN4 の RST 端子を，IchigoJam の CN4 の BTN 端子へ接続することで，BTN 押下時または，人感センサの検知時に MixJuice を起動するようにしました.

スリープ時の IchigoJam の待機電流は 20μA 程度，MixJuice が 0.6mA 程度となり，アルカリ乾電池による長期間動作が可能になります．なお，MixJuice の待機電流の多くがレギュレータ（IC1）の消費電流なので，レギュレータの交換によって，より長期間動作を可能にすることも出来るでしょう.

Single Shot 起動方式に対応したプログラムをリスト 2-3 に示します．ダウンロードする場合は，前回と同じ方法，もしくは他のプログラムをロードした状態で「GOTO 999」を実行し，ダウンロード・メニューが表示されてから，[2][2][Enter]の順にキーを押下してメニュー「22」を選択します.

省電力動作を行うには，行番号 4 の S=0 を S=1 に変更し，「SAVE 0」でファイル番号 0 へ保存してから「RUN」コマンドを実行します．省電力動作中にプログラムを停止するには，[ESC]キーを押しながら，ボタン SW2 を押します.

Single Shot 起動方式の省電力動作に関わるプログラムの流れは，以下の通りです.

① 変数 C に Ambient のチャネル ID を，変数 K にライトキーを，変数 D にデータ番号を代入します.
② S=0 のときに通常モードで動作し，S=1 に書き換えると Single Shot 起動方式による省電力動作を行います.
③ MixJuice が Wi-Fi ネットワークに接続されているかどうかを確認するコマンドです.
④ Wi-Fi ネットワークに接続していない場合，行番号 110 に戻り，接続するまで待機します.
⑤ ボタン状態を Ambient へ Wi-Fi 送信します.
⑥ （S=1 のときに,）MixJuice をディープ・スリープ・モードに設定します.
⑦ （S=1 のときに,）IchigoJam をスリープします．復帰するとき自動的にファイル番号 0 のプログラムの先頭から実行するので，行番号 260～280 が実行されることはありません.

リスト 2-3　Wi-Fi 対応 IoT 押しボタン・省電力版 MixJuice IoT BTN/PIR PS

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"MixJuice IoT BTN/PIR PS	1 画面へ「MixJuice IoT BTN/PIR PS」を表示
2 uart 0,2:?"CC BY ｸﾆﾉ ﾜﾀﾙ	2 シリアル送信を OFF に設定. 権利表示
3 C=xxxx:?" Ambient ﾁｬﾈﾙ ID=";C	3 変数 C へ Ambient のチャネル ID を入力
4 K="xxxxxxxxxxxxxxxx":'ﾗｲﾄ ｷｰ	4 変数 K へ Ambient のライトキーを入力
5 D=1:?" ﾃﾞｰﾀ=d";D	5 変数 D へ Ambient 用データ番号を入力
6 S=0:?" ﾊﾟﾜｰ S=";S;"(1=ON)	6 省電力動作を行うときは変数 S=1 に変更する
100 ?"== ｼﾞｭﾝﾋﾞ ==	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
110 uart 1,2:clk	110 シリアル送信を ON に. 受信データを破棄
120 if inkey() cont	120 シリアル受信が無くなるまで待機する
130 ?"MJ APS":wait 40	130 MixJuice のネットワーク接続を確認する
140 if inkey()<>49 goto 110	140 ネットワーク未接続のとき 110 行へ
150 clk	150 受信データを破棄
160 ?"MJ PCT application/json"	160 MixJuice の POST データ形式を設定する
170 gosub 700	160 受信データ表示サブルーチン(700 行)へ
200 ?"== ﾒｲﾝ ==	200 画面へ「== ﾒｲﾝ ==」を表示
210 B=1:gosub 500	210 ボタン押下状態 B=1 を Wi-Fi 送信する
220 wait 200	220 連続送信を避けるための待ち時間(3.3 秒)
230 if btn() cont	230 ボタンが開放されるまで待機する
240 B=0:gosub 500	240 ボタン開放状態 B=0 を Wi-Fi 送信する
250 if S goto 300	250 S=1 のときスリープ(300 行)へ
260 wait 200	260 連続送信を避けるための待ち時間(3.3 秒)
270 if !btn() cont	270 ボタンが押下されるまで待機する
280 goto 100	280 行番号 100 へ戻る
300 ?"== ｽﾘｰﾌﾟ ==	300 画面へ「== ｽﾘｰﾌﾟ ==」を表示
310 uart 1:?"MJ SLEEP 0"	310 MixJuice をスリープ状態に設定する
320 wait 40,0	320 MixJuice のスリープ待ち
330 sleep	330 IchigoJam をスリープ状態に設定する
500 ?"== TX ==	500 画面へ「== TX ==」を表示
～ リスト 2-2 と同一につき省略 ～	～ リスト 2-2 と同一につき省略 ～
700 '== RX ==	700 受信部(RX)を示すコメント
～ リスト 2-2 と同一につき省略 ～	～ リスト 2-2 と同一につき省略 ～

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/22.txt
・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　(メニュー[22])

省エネ運転機能②Cyclic Sleep 駆動

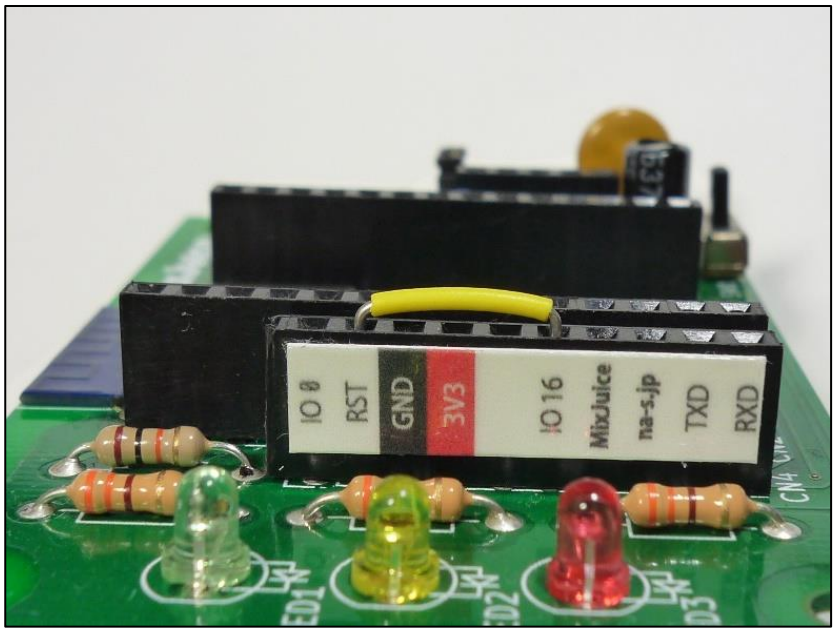

写真 2-7 MixJuice をスリープから自動復帰させるためのジャンパー接続の様子
MixJuice の CN4 の IO16 端子のウェイクアップ出力信号を，RST 端子へ入力することで，指定した経過時間が過ぎると自動起動することが出来る.

こんどは，Cyclic Sleep 駆動方式による IoT 照度センサの省電力動作について説明します．Cyclic Sleep 駆動方式は，10 分間隔，30 分間隔といった送信頻度で，定期的にセンサ情報を得たい場合に適した省電力化方法です．ここでは，IchigoJam S/T/U の IN2 端子に入力した新日本無線の照度センサ NJL7502L の照度値を，約 1 分ごとに送信します．送信間隔は，リスト 2-4 の行番号 6 の T=60 を 600 に変更すると 10 分間隔に，3600 にすれば 60 分間隔に変更することが出来ます.

MixJuice にはスリープからの復帰用のタイマーが内蔵されており，指定した時間が経過すると，ウェイクアップ信号を IO 16 から出力します．写真 2-7 のように，MixJuice の IO 16 の出力を，MixJuice のリセット入力 RST 端子へ接続することで，MixJuice の自動起動が行えます.

同様に，IchigoJam にもスリープからの復帰用のタイマーが内蔵されており，プログラムの中断後，指定した時間が経過してから，プログラムを再開することが出来ます.

MixJuice と IchigoJam の各タイマー機能を使用し，MixJuice は約 60 秒に 1 回，IchigoJam は約 10 秒に 1 回の頻度で起動し，IchigoJam が起動したときに MixJuice が起動していなければ，すぐにスリープに戻ります．MixJuice が起動していた時は，送信処理を行います．以下にプログラム中の主要な処理手順①～⑥について説明します.

① IchigoJam の IN2 端子に入力された電圧に応じたアナログ値を変数 B へ代入します.
② 変数 B の値を Wi-Fi で Ambient へ送信します.
③ MixJuice をディープ・スリープ・モードに設定します．変数 T はスリープから復帰するまでの時間（秒）です.
④ IchigoJam を 10 秒間，スリープ状態に設定します.
⑤ スリープ復帰後に MixJuice の Wi-Fi 接続状態を確認します. MixJuice がスリープ中のときや，Wi-Fi への接続が完了していないときは，手順④に戻り，スリープと接続確認を繰り返します.
⑥ Wi-Fi への接続が完了すると，手順①に戻り，IN2 端子のアナログ値を Wi-Fi 送信します.

リスト 2-4　Wi-Fi 対応 IoT 照度センサ・省電力版 MixJuice IoT ANA2 PS

BASIC プログラム	解説
new	new=現在のプログラムを消去する
1 cls:?"MixJuice IoT ANA2 PS	1 画面へ「MixJuice IoT ANA2 PS」を表示
2 uart 0:?"CC BY ｸﾆﾉ ﾜﾀﾙ	2 シリアル送信を OFF に設定．権利表示
3 C=xxxx:?" Ambient ﾁｬﾈﾙ ID=";C	3 変数 C へ Ambient のチャネル ID を入力
4 K="xxxxxxxxxxxxxxxx":'ﾗｲﾄ ｷｰ	4 変数 K へ Ambient のライトキーを入力
5 D=1:?" ﾃﾞｰﾀ=d";D	5 変数 D へ Ambient 用データ番号を入力
6 T=60:?" ｼｭｳｷ=";T;"ﾋﾞｮｳ	6 送信間隔 60 秒を変数 T へ代入する．
7 S=0:?" ﾊﾟﾜｰ S=";S;"(1=ON)	7 省電力動作を行うときは変数 S=1 に変更する
100 ?"== ｼﾞｭﾝﾋﾞ =="	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示
110 uart 1:clk	110 シリアル送信を ON に．受信データを破棄
120 ?"MJ PCT application/json"	120 MixJuice の POST データ形式を設定する
130 gosub 700	130 受信データ表示サブルーチン(700 行)へ
200 ?"== ﾒｲﾝ =="	200 画面へ「== ﾒｲﾝ ==」を表示
210 B=ANA(2):?" ANA2=";B ← ①	210 IN2 のアナログ値を取得する
220 gosub 500 ← ②	220 Wi-Fi 送信サブルーチン(500 行)へ
230 if !S wait T*60:goto 200	230 S=0 のときの待ち時間処理
300 ?"== ｽﾘｰﾌﾟ =="	300 画面へ「== SLEEP ==」を表示
310 uart 1,2:?"MJ SLEEP ";T ← ③	240 MixJuice をスリープ状態に設定する
320 led 0:wait 600,0:led 1 ← ④	310 IchigoJam を 10 秒間スリープ状態にする
330 if inkey() cont	320 受信データを空読みする
340 ?"MJ APS":wait 40	330 MixJuice のネットワーク接続を確認する
350 if inkey()<>49 goto 320 ← ⑤	340 ネットワーク未接続のとき 310 行へ
360 goto 100 ← ⑥	350 行番号 100 へ戻り，処理を繰り返す
500 ?"== TX =="	500 画面へ「== TX ==」を表示
～ リスト 2-2 と同一につき省略 ～	～ リスト 2-2 と同一につき省略 ～
700 '== RX ==	700 受信部(RX)を示すコメント
～ リスト 2-2 と同一につき省略 ～	～ リスト 2-2 と同一につき省略 ～

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/23.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　(メニュー[23])

[Column] 実用運転するときの注意点

電子工作用に販売されている回路基板をケースに入れることで，金属などの接触によるショートを防止することが出来ます．しかし，電子工作用に販売されている基板やケースの多くは，市販の製品に比べて火災などの事故の発生リスクが高いことを考慮する必要があります．製作時は，主に以下の点に注意してください．

- 回路がショートしたときに備えて，電源にヒューズを挿入する．
- AC アダプタを使用する場合は PSE マークのある市販品を使用する．
- ポリカーボネートなどの燃えにくいケースを使用する．
- 燃えやすい部材（プラスチックなど）に対して，ポリイミドテープで耐熱処理を施す．
- ジャンパ・ワイヤやピンヘッダの接続部，ブレッドボード上の部品，ユニバーサル基板のワイヤー配線などを接着剤で固定する．
- ケース内に金属片やハンダ片，水滴などが残留・混入しないようにする．
- 発熱が生じる場合や，12V 以上の電圧，1A 以上の電流が生じる部分は，市販品を使用する．

省エネ運転機能③MixJuice から IchigoJam を起動

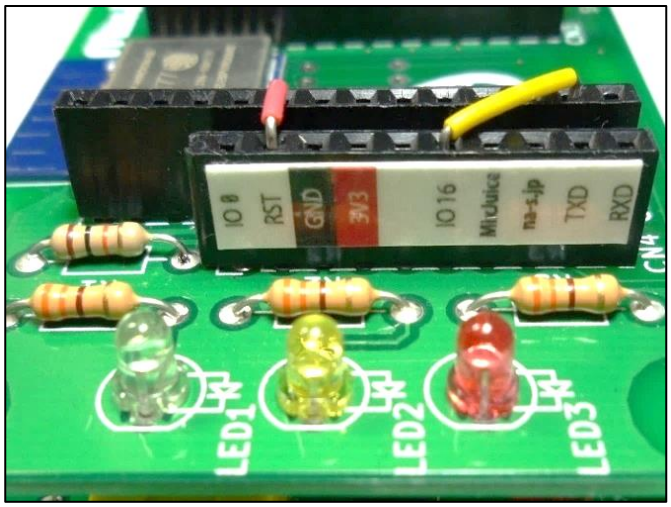

写真 2-8 MixJuice のタイマー出力 IO 16 端子を IchigoJam の BTN 端子へ接続し，IchigoJam の IN4 端子を MixJuice の RST 端子へ接続する

MixJuice の CN4 の IO16 端子のウェイクアップ出力信号を使って，IchigoJam を起動する．起動した IchigoJam は，IN4 端子からのパルス出力によって，MixJuice を起動する

最後の省エネ運転方法は，MixJuice のタイマーを使って，IchigoJam を起動する方法です．ダウンロードページのメニュー[24]は IoT 照度センサ用，[25]は温湿度センサ用です．温度センサには写真 2-9 および表 2-5 に示す Silicon Labs 製 Si7021 搭載モジュールを使用します．温湿度センサ・モジュールの VIN 端子を IchigoJam S または T の CN5 の 3.3V 端子へ，GND，SCL，SDA をそれぞれ同じ信号名の端子へ接続してください．プルアップ抵抗はモジュールに搭載されているので不要です．なお，IchigoJam U の場合は，CN4 の IN2 が SDA，SCL は CN3 にあります．

プログラムはリスト 2-3 と似ていますが，IchigoJam が起動したときに行番号 110 で MixJuice を起動するための起動用のパルス信号を出力する点と，I2C インタフェースを使って温度と湿度を取得し，これら 2 値のデータを Wi-Fi 送信する点などに違いがあります．MixJuice 起動用のパルス信号は，IchigoJam の IN4 端子（OUT 11）から MixJuice の RST 端子へ入力されます．温湿度は行番号 200 番台で取得します．図 2-6 のように温度と湿度の 2 値を一つのグラフ中に表示することも出来ます．

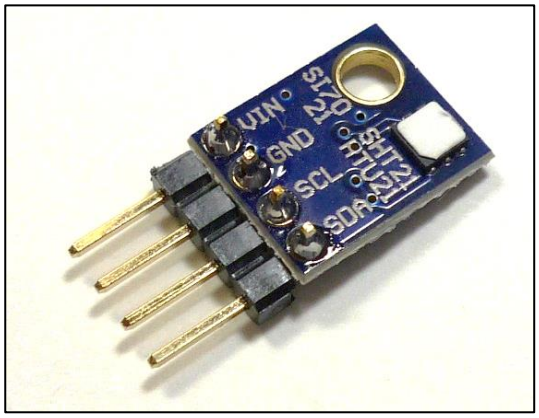

写真 2-9 Silicon Labs 製の温湿度センサ・モジュール Si7021

温度センサと湿度センサを内蔵したセンサ・モジュール．高精度であるにもかかわらず，400 円程度で国内販売されており， $2 程度で中国から個人輸入できるなど，コストパフォーマンスに優れている．

表 2-5　温湿度センサ・モジュール Si7021

項目	仕様
種別・型番	温湿度センサ Si7021
電源電圧	DC 1.9V～3.6V
消費電流	150μA
待機時電流	60 nA
温度センサ精度	最大±0.4°C（-10～85°C）
湿度センサ精度	最大±3%（0～80%）±4.5%（100%）
インタフェース	I^2C インタフェース

リスト 2-5　Wi-Fi 対応 IoT 温湿度センサ MixJuice IoT Env センサ

BASIC プログラム

```
new
1 cls:?"IoT Env ｾﾝｻ
2 C=xxxx
3 K="xxxxxxxxxxxxxxxx"
4 T=60:S=0
```

初期設定

省電力(S=1 で有効)

送信間隔(秒)

```
100 'STUP
110 out 11,0:out 11,1
120 uart 1,2:clk
130 if inkey() cont
140 ?"MJ APS":wait 40
150 if inkey()<>49 goto 120
160 clk
170 ?"MJ PCT application/json"
180 gosub 700
```

準備

```
200 'I2C
210 let[0],#3ae6,#f3,#f5
220 ?i2cw(64,#802,1)::wait 2
230 ?i2cr(64,#806,2);
240 A=([3]>>8+[3]<<8)/37-474
250 ?i2cw(64,#804,1)::wait 2
260 ?i2cr(64,#806,2)
270 B=([3]>>9+[3]<<7)/26-65
```

温湿度の取得

```
300 'TX
310 uart 1:clk
320 ?"MJ POST START ambidata.";
330 ?"io/api/v2/channels/";C;
340 ?"/data"
350 gosub 700:uart 1
360 ?"{";chr$(34);"writeKey";
370 ? chr$(34);":";chr$(34);
380 ? str$(K);chr$(34);",";
390 ? chr$(34);"d1";chr$(34);
400 ?":";A/10;".";A%10;",";
410 ? chr$(34);"d2";chr$(34);
420 ?":";B/10;".";B%10;"}"
430 ?"MJ POST END"
440 gosub 700
450 if !S wait T*60:goto 200
```

送信部

```
500 'SLP
510 uart 1:?"MJ SLEEP ";T
520 wait 40,0
530 sleep
```

省電力制御部

```
700 'RX
710 uart 0:clt
720 I=inkey()
730 if I clt:? chr$(I);
740 if tick()<40 goto 720
750 return
```

受信部

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/25.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[25]）

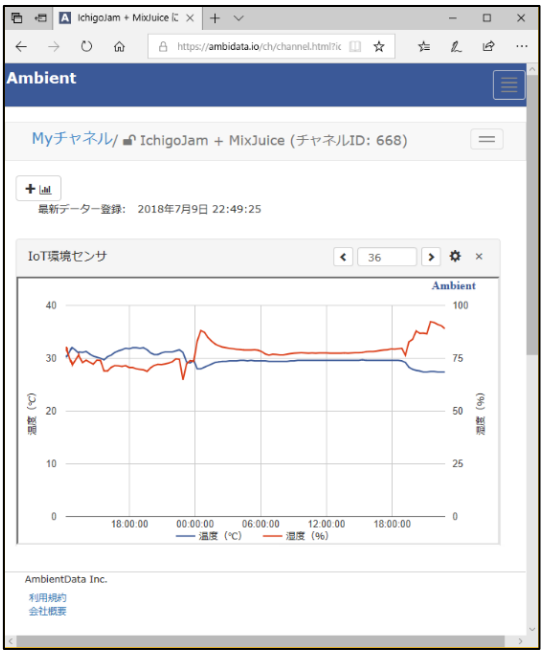

図 2-6 Wi-Fi 対応 IoT 温湿度センサから Ambient へ温度＋湿度データを送信したときの様子

Ambient の一つのグラフに温度と湿度の 2 値を表示してみた．エアコンで温度を下げると相対湿度（%）が上昇する様子が分かる．

製作した Wi-Fi 対応プログラム

筆者が作成した各 Wi-Fi 対応プログラムを表 2-6 に示します. 前述の通り, MixJuice を使えば, 直接, ダウンロードすることが出来ます. パソコンでダウンロードして, USB シリアル変換アダプタ経由で IchigoJam へ転送することも可能です.

省エネ運転機能の「①Single Shot 起動」を使用したプログラムは IoT 押しボタンの省電力版,「②Cyclic Sleep 駆動」は IoT 照度センサの省電力版,「③MixJuice から IchigoJam を起動」は IoT 照度センサと IoT 温湿度センサに使用しました.

表 2-6　製作した Wi-Fi 対応プログラムの一覧

ファイル名	プログラム名	備考（〇数字は省エネ運転機能の番号）
20.txt	MixJuice IoT TX	Wi-Fi 対応 MixJuice 送信テスト用プログラム・トグル送信
29.txt	MixJuice IoT TX LOOP	Wi-Fi 対応 MixJuice 送信テスト用プログラム・繰り返し送信
21.txt	MixJuice IoT BTN/PIR	Wi-Fi 対応 IoT 押しボタン/IoT 人感センサ・Ambient 版
26.txt	MixJuice IoT BTN/PIR	Wi-Fi 対応 IoT 押しボタン/IoT 人感センサ・IFTTT 版
22.txt	MixJuice IoT BTN/PIR PS	Wi-Fi 対応 IoT 押しボタン/IoT 人感センサ・①SS 省電力版
27.txt	MixJuice IoT BTN/PIR PS	Wi-Fi 対応 IoT 押しボタン/IoT 人感センサ・①SS 省電力・IFTTT 版
23.txt	MixJuice IoT ANA2 PS	Wi-Fi 対応 IoT 照度センサ・②Cyclic Sleep 省電力版
24.txt	MixJuice IoT ANA2 PS2	Wi-Fi 対応 IoT 照度センサ・③MJ タイマー省電力版
25.txt	IoT Env ｾﾝｻ PS2	Wi-Fi 対応 IoT 温湿度センサ・③MJ タイマー省電力版
・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/●●.txt ・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"		

本文中では Ambient 版を紹介しましたが，IFTTT 版も似たような処理になります．ただし，送信先の URL が異なる点，HTTP POST ではなく HTTP GET で送信する点などが異なります.

[Column] LINE Notify へ簡単送信（2021 年 2 月 7 日・追加）

IchigoJam から LINE Notify へのメッセージ転送サービスを利用すると，IFTTT を使用せずに LINE アプリへの送信が出来るようになります．下記のクエリ TOKEN の xxx には，LINE Notify 用のトークン（図 2-7 の説明参照）を，クエリ MES にはメッセージを渡すだけです.

```
IchigoJam から LINE にコンニチハを送信する：
?"MJ GETS BOKUNIMO.COM/15/TOLINE/?TOKEN=xxx&MES=ｺﾝﾆﾁﾊ"↵
```

下図は，IchigoJam から「コンニチハ」を送って，LINE アプリで表示したときの様子です.

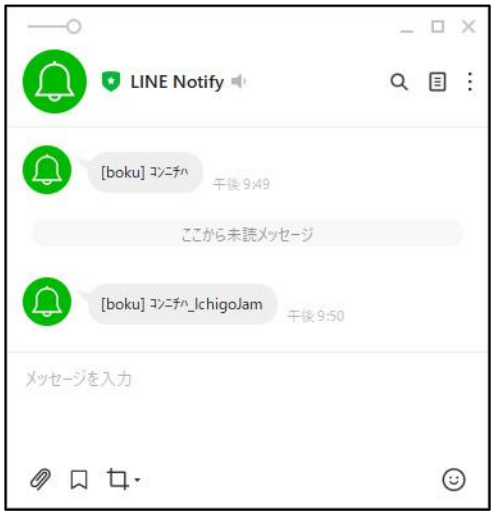

図 2-7 LINE Notiy へのメッセージ転送サービス

下記の方法で取得したトークンを使って LINE Notify へ送信したときの様子

1. https://notify-bot.line.me/ へアクセス
2. 右上のアカウントメニューから「マイページ」を選択
3. アクセストークンの発行で「トークンを発行する」を選択
4. トークン名「boku」（任意）を入力
5. 送信先のトークルームを選択する
   （「1:1 で LINE Notify から通知を受け取る」など）
6. [発行する]ボタンでトークンが発行される
7. TOKEN=xxx の「xxx」の部分にトークンを記入

詳細（筆者ブログ）：https://bokunimo.net/blog/ichigojam/1023/

3章 SORACOM LoRaWAN を使った IoT バーコード・リーダの製作

最終章では，未来の IoT 専用通信方式 LoRaWAN を使った IoT デバイスの実験例を紹介します．従来の LTE 対応 IoT 機器には，端末ごとに回線契約済みの SIM カードが必要でした．Wi-Fi 対応 IoT 機器の場合は，通信距離が短く，屋内でしか使えないことが課題でした．LoRaWAN は，これらの課題を解決した IoT 専用の最新の通信方式で，今後の普及が期待されています．応用の一例として，無人直売所でのバーコードによる商品管理やレシート発行などが考えられるでしょう．

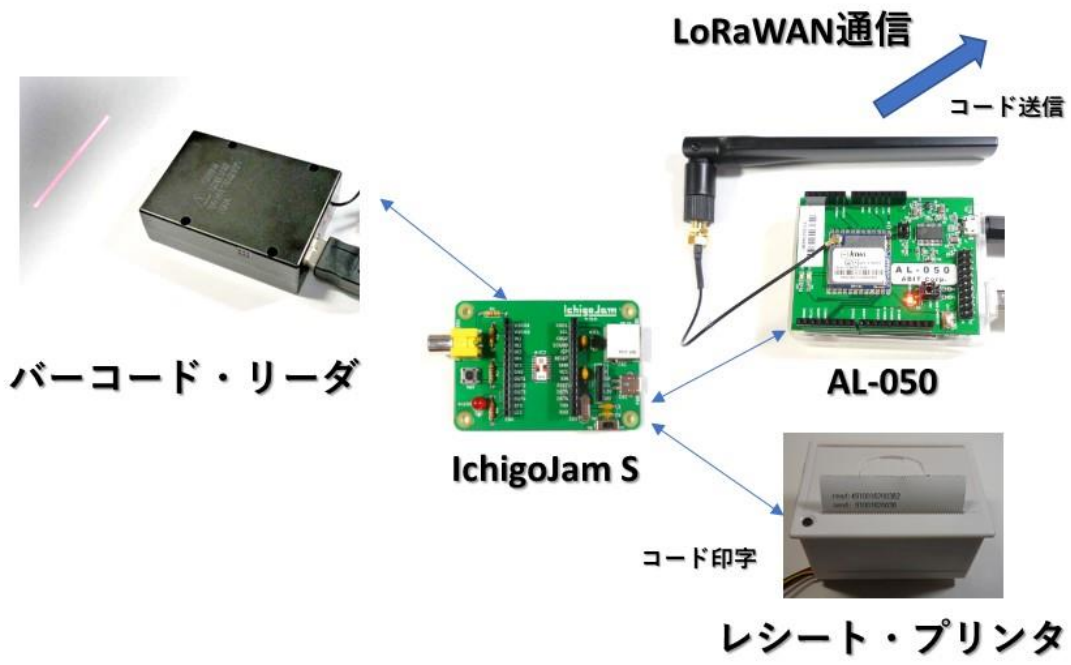

表 3-1　AL-050 を使った IoT デバイス例

通信方式	LoRaWAN 920MHz 帯
通信エリア	SORACOM LoRaWAN サービスエリア
通信料金	Harvest 1 日 5 円 Beam 0.0018 円など
機器コスト	9,500 円～
IoT 機能例	ボタン，LCD，バーコード・リーダなど

表 3-2　AL-050 の省電力特性の一例

状態	**全体**	**AL-050**
通常状態	45 mA	32 mA
スリープ時	1 mA	1 mA

5V 入力での実測値

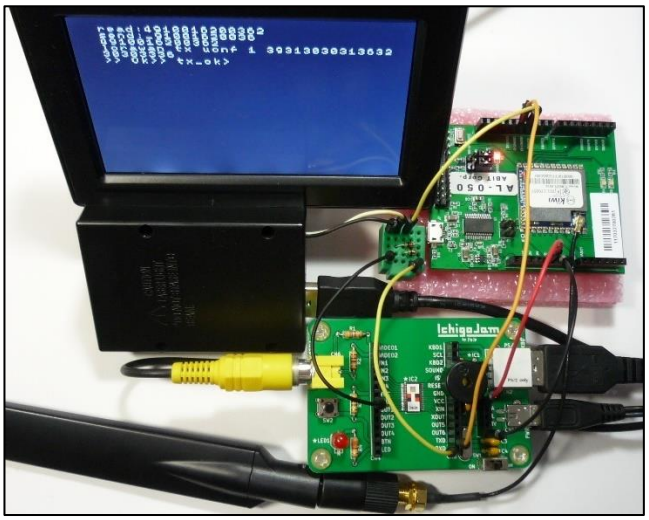

写真 3-1 LoRaWAN 対応 SORACOM AL-050 を使った実験時の様子

バーコード・リーダと SORACOM AL-050 を IchigoJam S へ接続し，読み取った JAN コードなどを LoRaWAN 送信する実験の様子．無人直売所での商品管理やレシート発行などの応用が考えられる

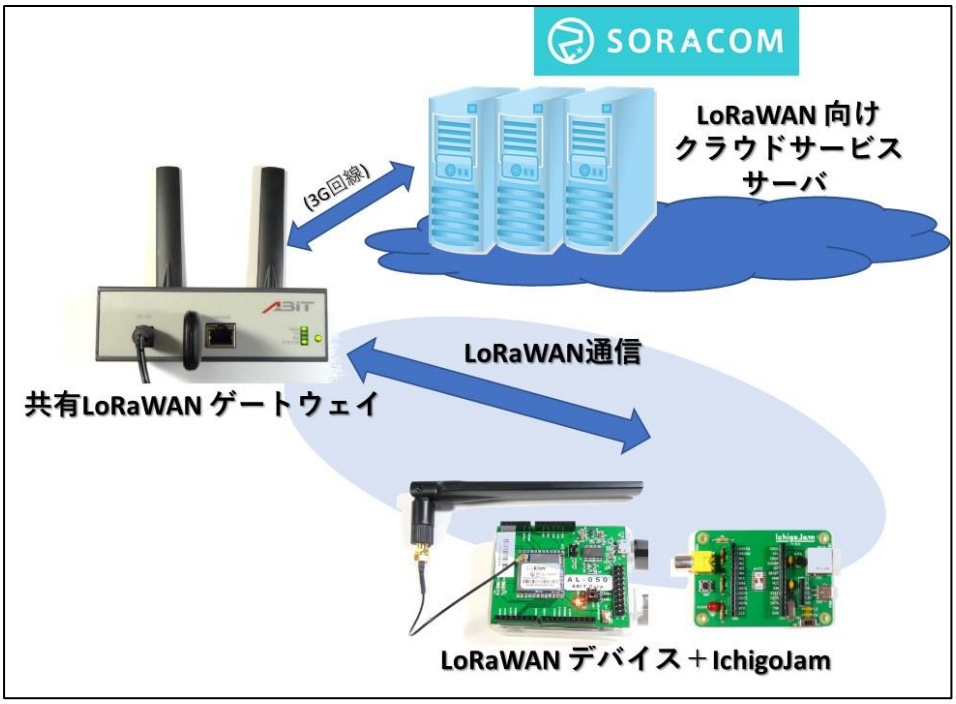

図　3-1 SORACOM LoRaWAN の概略
SORACOM の LoRaWAN デバイスを購入すれば，共有の LoRaWAN ゲートウェイを経由して同社が提供するサーバへ接続できる．

LoRaWAN は，LoRa 変調を用いることで受信感度を高めることが可能な IoT 専用の通信方式です．パケットに付与可能な情報量を 11 バイト（拡散度 SF10 の場合）に制限し，伝送速度を落とすことによって，モバイル通信回線に迫る 144dB ものリンクバジェットを得ることが出来，最大 10km 程度の長距離のワイヤレス通信が可能になります．

SORACOM の LoRaWAN サービスであれば，ユーザ・コンソール（https://console.soracom.io/）から LoRaWAN デバイス製品（写真 3-2）を購入するだけで，共有ゲートウェイや，同社のサーバを利用することができます．ただし，執筆時点では利用可能な共有ゲートウェイが多くありません．必ず同社のウェブページ（https://lora-space.soracom.jp/map/）でカバーエリアを確認してください．共有ゲートウェイを SORACOM からレンタル（有償）して利用することも可能です．なお，SORACOM 以外で購入した LoRaWAN デバイスを，SORACOM の LoRaWAN サービスへ接続することは出来ません．

写真 3-2 SORACOM から販売されている LoRaWAN 製品の一例
本稿で使用する LoRa Arduino 開発シールド AL-050（左）と単体動作が可能な STM32L0 LoRa Discovery Kit（右）が，SORACOM ユーザ・コンソールから購入できる．

LoRaWAN 対応 IoT 押しボタンのハードウェア構成

はじめに，SORACOM の LoRaWAN に対応した IoT 押しボタンのハードウェア構成について説明します．必要な機器は，IchigoJam S/T/U と，SORACOM の AL-050 です．

IchigoJam と AL-050 とは UART シリアル・インタフェースで接続します．AL-050 の MOSI 端子を IchigoJam の RXD 端子へ，MISO 端子を TXD へ接続し，AL-050 の電源（5V と GND）は IchigoJam の CN5 から供給します．本機の MOSI／MISO 端子の名称は PC とマイコンとの接続関係でつけられているので，混乱しやすくなっています．接続を誤らないように注意してください．

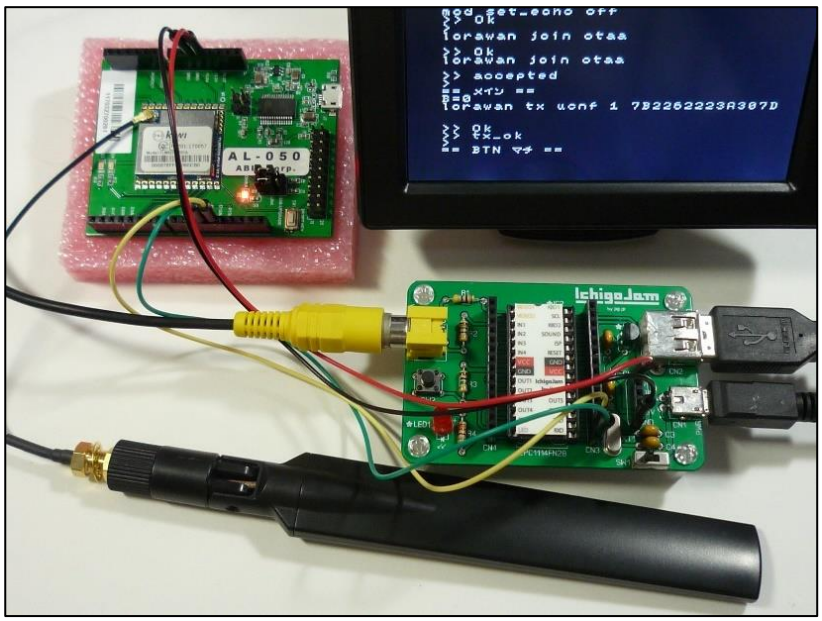

写真 3-3 LoRaWAN 実験の様子
IchigoJam の TXD 端子を LoRaWAN 開発シールド AL-050 の MISO へ，RXD 端子を MOSI へ接続する．MOSI／MISO 端子の名称の関連付けが通常とは異なる点に注意する（開発シールド側がマスタになる）．

SORACOM のサーバ設定

SORACOM LoRaWAN デバイスの購入やサーバ側の設定を行うには，Web ブラウザを使用します．ユーザ・コンソールにログインし，LoRa グループを作成してから，図 3-2 の SORACOM LoRa Space を[利用する]に，SORACOM Harvest 設定を ON に設定し，LoRaWAN デバイスを LoRa グループへ登録してください．詳細な設定方法は，図 3-2 や，SORACOM が提供している LoRaWAN デバイス設定ガイドをご覧ください．

> LoRaWAN デバイス設定ガイド(SORACOM)：
> https://dev.soracom.io/jp/start/lora_uc_device/

リスト 3-1 のプログラムを IchigoJam に入力し，「RUN」で実行すると，ボタン SW2 の状態を SORACOM Harvest へ送信します．ただし，エラーが発生した時などに，IchigoJam から LoRaWAN 開発シールドへシリアル送信されたデータが，LoRaWAN 側でもエラーとなり，相互にエラーを繰り返し続けることがあります．その場合は，IchigoJam の RXD のジャンパ・ワイヤを取り外してください．

SORACOM 側が受信したデータの内容を確認するには，SORACOM ユーザ・コンソールの[LoRa デバイス管理]で対象のデバイスにチェックマークを入れてから，[操作]メニューの[データを確認]を選択してください．図 3-3 のようにデータが蓄積されていることが確認できるでしょう．

なお，SORACOM の LoRaWAN デバイスの月額利用料は発生しませんが，SORACOM のサーバに

ついては利用料が発生します．SORACOM Harvest は 1 日 5 円，受信したデータを他のサーバへ転送する SORACOM Beam は 1 リクエスト 0.0018 円，LoRaWAN デバイスへのダウンリンク通信には 1 リクエスト 0.0054 円などです．回線ごとに固定の月額料金や解約手数料が発生する一般的なモバイル回線と比較して，通信量が少なく，契約・解約件数の多い IoT 機器に適した課金方法となっています．

図 3-2　SORACOM ユーザ・コンソールで LoRa グループの設定を行う

LoRa グループの管理画面で，SORACOM LoRa Space を［利用する］に設定し，SORACOM Harvest 設定を ON に設定する．SORACOM Harvest については利用料（1 日 5 円）が発生するので，実験が終わったら OFF に戻しておく

図 3-3　LoRaWAN デバイスの送信結果を SORACOM Harvest で確認する

IchigoJam から LoRaWAN 送信したボタン状態値（押下＝1，開放＝0）が，SORACOM Harvest に蓄積されたときの様子

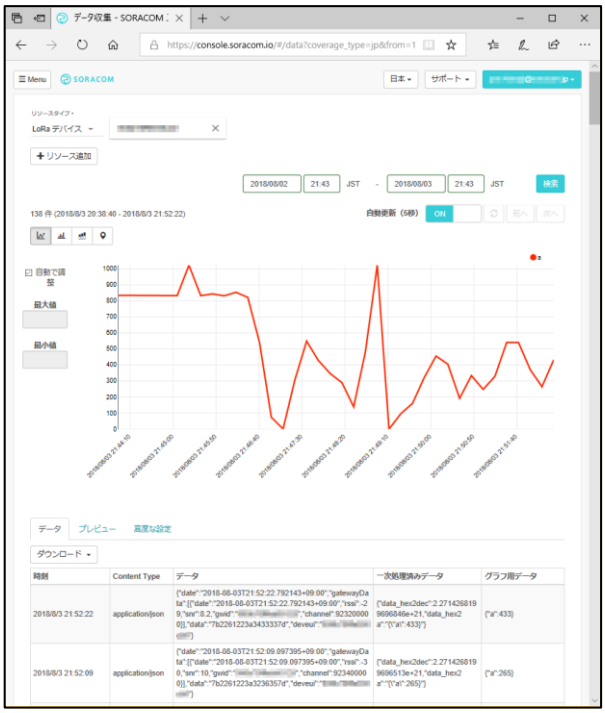

図 3-4　アナログ値の LoRaWAN 送信結果を SORACOM Harvest で確認する
IchigoJam からセンサ値を LoRaWAN 送信し，SORACOM Harvest に蓄積したときの様子

LoRaWAN 対応 IoT 押しボタン/人感センサのプログラム解説

以下に，IchigoJam BASIC で LoRaWAN 送信を行う LoRaWAN 対応 IoT 押しボタン/人感センサのプログラム（リスト 3-1）の主要な処理部①～⑤について説明します．

① IchigoJam のシリアル通信速度を，LoRaWAN 開発シールドの 9600bps に合わせます．
② LoRaWAN 開発シールドの初期設定を行います．行番号 130 では LoRaWAN モジュールの初期化コマンドを送信し，次行で行番号 700 の受信サブルーチンを実行します．初期化に成功したら行番号 150 でシリアルのエコー設定を行います．
③ LoRaWAN ネットワークへの参加を行う処理部です．行番号 220 でネットワーク参加コマンドを送信し，参加するまで繰り返し処理を行います．
④ メイン処理部です．行番号 320 で行番号 500 の送信サブルーチンを実行します．送信後，行番号 340 でボタン SW2 に変化があるまで待機します．変化したときは，行番号 300 に戻り，処理を繰り返します．
⑤ 送信処理部です．変数 B に代入された値を LoRaWAN 送信します．
⑥ 受信処理部です．予め配列変数に 3 文字の待ち受け文字列を代入してから呼び出します．受信結果に，待ち受け文字列が含まれていた場合は，変数 R へ 1 が代入され，含まれていない場合は 0 となります．例えば，処理部③から呼ばれたときは，「_ac」の文字列を待ち受けます．ネットワーク参加に成功すると，LoRaWAN 開発シールドが「_accept」を応答するので，R=1 を代入します．

なお，照度センサなど，アナログ入力値を送信するプログラムは，32.txt または，MixJuice のメニュー番号 32 でダウンロードすることが出来ます．BTN 端子へ 0～3.3V の電圧を入力すると電圧に応じた 0～1023 の整数値を送信します．SORACOM Harvest へ蓄積した様子を図 3-4 に示します．

以上を応用すれば，照度センサや温度センサ，人感センサなど様々な LoRaWAN 対応 IoT センサを製作することが出来るようになるでしょう．

リスト 3-1　LoRaWAN 対応 IoT 押しボタン/IoT 人感センサ LoraWAN AL-050 TX

BASIC プログラム	解説
new 1 cls:?"LoRaWAN AL-050 TX" 2 bps 9600:uart 0 3 ?"CC BY ｸﾆﾉ ﾜﾀﾙ 4 P=1:?" ﾎﾟｰﾄ";P 5 B=btn()	new=現在のプログラムを消去する 1 画面へ「LoraWAN AL-050 TX」を表示 2 シリアルの設定とシリアル出力の無効化 3 権利表示 4 変数 P へ LoRaWAN のポート番号を代入 5 変数 B へ現在のボタン状態を代入
100 ?"== ｼﾞｭﾝﾋﾞ == 110 [0]=32:[1]=79:[2]=107:'Ok 120 uart 3:clk:?:gosub 700 130 ?"mod factory_reset" 140 gosub 700:if !R goto 170 150 ?"mod set_echo off" 160 gosub 700:if R goto 200 170 uart 0:?"ERR" 180 end	100 画面へ「== ｼﾞｭﾝﾋﾞ ==」を表示 110 配列変数へ待ち受け文字" Ok"を代入 120 シリアルへ改行を出力してから受信開始 130 LoRaWAN シールドへ初期化コマンド送信 140 OK の応答が無かったときに 170 行へ 150 LoRaWAN シールドのエコーを OFF に設定 160 OK の応答があれば 200 行へ 170 シリアル出力の無効化と ERR 表示 180 プログラム終了
200 '== Join == 210 [1]=97:[2]=99:'ac 220 ?"lorawan join otaa" 230 gosub 700 240 if !R goto 220 250 uart 0	200 LoRaWAN ネットワーク参加処理 210 配列変数へ待ち受け文字"ac"を代入 220 ネットワーク参加コマンドを送信 230 シリアル受信処理を実行 240 ネットワーク未参加のとき 220 行へ戻る 250 シリアル出力の無効化
300 ?"== ﾒｲﾝ == 310 ?"B=";B 320 gosub 500 330 ?"== BTN ﾏﾁ == 340 if B=btn() cont 350 B=!B 360 goto 300	300 画面へ「== ﾒｲﾝ ==」を表示 310 変数 B の値を表示 320 送信処理(500 行)を実行(変数 B を送信) 330 画面へ「== BTN ﾏﾁ ==」を表示 340 ボタン状態の変化待ち 350 ボタン状態を反転(0⇔1) 360 行番号 300 へ戻って繰り返す
500 '== TX == 510 uart 3:clk 520 [0]=95:[1]=111:[2]=107:'_ok 530 ?"lorawan tx ucnf ";P; 540 ?" 7B22";hex$(asc("b")); 550 ?"223A";hex$(48+B);"7D" 560 gosub 700:uart 0 570 return	500 画面へ「== TX ==」を表示 510 シリアル送信を ON に設定. 受信データ破棄 520 配列変数へ待ち受け文字"_ok"を代入 530 LoRaWAN 送信コマンドとポート番号を送信 540 送信データの変数名 b を送信 550 変数 B の値を送信 560 受信サブルーチン(700 行)へ 570 サブルーチンを終了し, gosub 部へ戻る
700 '== RX == 710 uart 0:clt:R=0 720 I=inkey() 730 if I clt:? chr$(I); 740 if tick()>330 goto 780 750 if R=3 goto 720 760 if I=[R] R=R+1 else R=0 770 goto 720 780 ?:R=R=3:uart 3 790 return	700 受信部(RX)を示すコメント 710 シリアル送信を OFF に. タイマー初期化 720 シリアルデータを受信 730 データがあれば, 表示する 740 データ待ち時間 5.5 秒以上なら 780 行へ 750 待ち受け文字列と合致済なら 720 行へ 760 待ち受け文字との比較を実行 770 行番号 720 へ戻る 780 待ち受け合致済なら R=1 を不一致なら 0 790 サブルーチンを終了し, gosub 部へ戻る

・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/31.txt

・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　(メニュー[31])

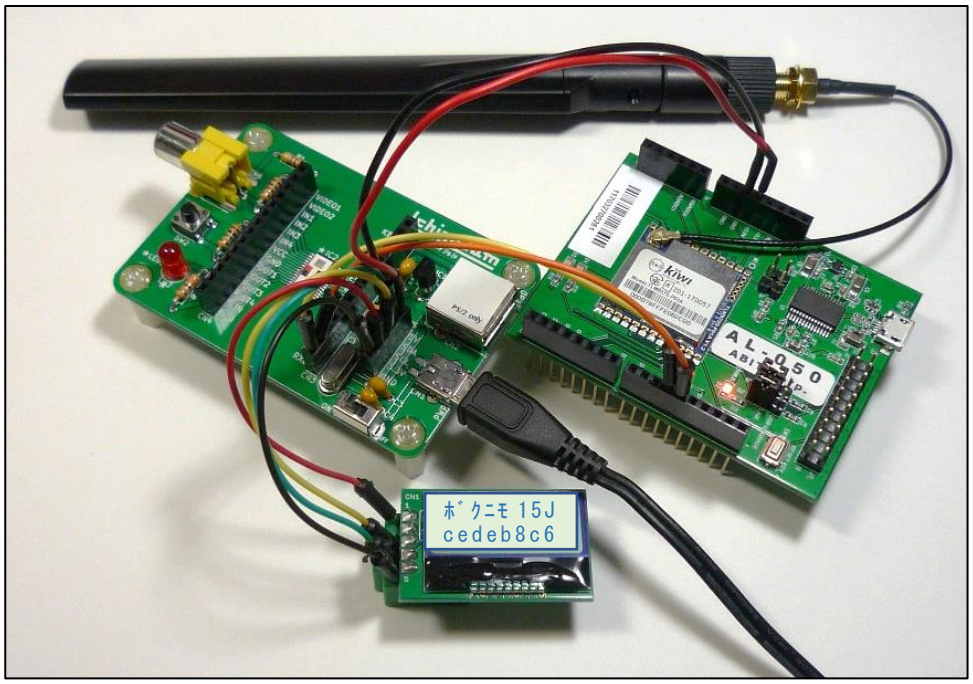

写真 3-4 LoRaWAN 対応 IoT 液晶ディスプレイ端末
LCD モジュール AE-AQM0802（秋月電子通商製）を接続し，LoRaWAN で受信したテキスト文字を表示する端末を製作した

遠隔地にある照明を制御したり，文字を送信したり，音声を出力するためのコード番号を送ったりするには，LoRaWAN の受信機能を使用する必要があります．ここでは，SORACOM のユーザ・コンソールから情報を送信する方法と，LoRaWAN で受信するプログラム，受信結果を LCD（液晶ディスプレイ）へ表示する実験方法を紹介します．

SORACOM から送信を行うには，ユーザ・コンソールを起動し，[LoRa デバイス管理]から対象デバイスを選択し，[操作]メニューから[ダウンリンク通信]を選び，16 進数の送信データと fPort 番号「2」を入力してから[送信]ボタンを押します．

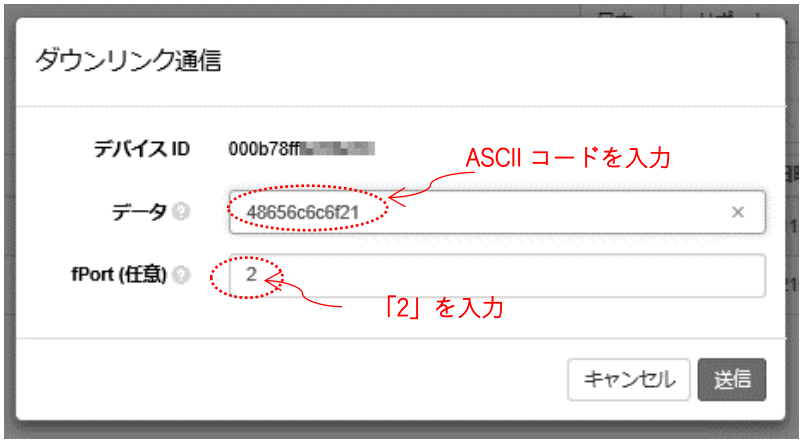

図 3-5 ダウンリンク通信の入力例
LoRaWAN デバイスへ送信したいデータを16進数で入力する．本例では「Hello!」の ASCII コードを入力した．fPort には受信側のポート番号（2）を入力する．

表 3-3 ASCII コード（カナ含む）による文字列の例

文字列	ASCII コード（カナ含む）
A	41
ABC	414243
ABCDEFGHIJK	4142434445464748494a4b
abcdefghijk	6162636465666768696a6b
01234567890	30313233343536373839
Hello!	48656c6c6f21
ボクニモ 15J	cedeb8c6d331354A

例えば，図 3-5 のように，送信データに ASCII コード（表 3-3）で「48656c6c6f21」を入力すると，「Hello!」のメッセージを LoRaWAN デバイスへ送ることが出来ます．ダウンリンク通信のポート番号 fPort は初期値の 2 を使用しました．次節のプログラムの変数 P が受信ポート番号です．

LoRaWAN で受信する IchigoJam 用プログラム

リスト 3-2 に LoRaWAN の受信用プログラムを示します．LoRaWAN 通信では，受信だけを行いたい場合でも，はじめに送信を行う必要があるので，リスト 3-1 の送信用のプログラムと似ています．行番号 530 が 540 受信要求です．受信要求したいポート番号を変数 P へ代入し，送信データ（ペイロード）の中身が 0 の空データを送信します．

受信結果は行番号 700 のサブルーチンで表示します．ダウンリンク通信で入力した 16 進数の値が表示されたら受信成功です．行番号 320 で T=5 秒の待ち時間処理を行ってから，受信を繰り返します．行番号 700 の受信サブルーチンで 5.5 秒の処理時間を要するので，実際の周期は約 11 秒です．

本プログラムでは，受信結果の 16 進数値をテレビ画面へ表示するだけですが，受信した値をテキスト文字列として保持するプログラムを 34.txt，MixJuice のメニュー34 へ収録しました．

受信結果を I2C 接続 LCD へ表示するには，IchigoJam の I2C 信号を小型 LCD モジュール AE-AQM0802（秋月電子通商製）へ接続し，37.txt のプログラムを実行します．ユーザ・コンソールからメッセージを ASCII コードで送信すると，LCD にメッセージが表示されます．

リスト 3-2　LoRaWAN 対応 IoT 制御用・受信プログラム LoRaWAN AL-050 RX

BASIC プログラム	解説
new 1 cls:?"LoRaWAN AL-050 RX" 2 bps 9600:uart 0 3 ?"CC BY ｸﾆﾉ ﾜﾀﾙ 4 P=2:?" ﾎﾟｰﾄ";P 5 T=5:?" ｼｭｳｷ=";T;"ﾋﾞｮｳ 100 ～～ リスト 3-1 と同一 ～～ 200 ～～ リスト 3-1 と同一 ～～ 300 ?"== ﾒｲﾝ == 310 gosub 500 320 wait T*60 330 goto 300 500 '== TX == 510 uart 3:clk 520 [0]=114:[1]=120:[2]=32:'rx 530 ?"lorawan tx ucnf ";P; 540 ?" 00" 550 gosub 700 560 uart 0 570 return 700 ～～ リスト 3-1 と同一 ～～	new=現在のプログラムを消去する 1 画面へ「LoraWAN AL-050 TX」を表示 2 シリアルの設定とシリアル出力の無効化 3 権利表示 4 変数 P へ LoRaWAN のポート番号を代入 5 変数 T へ受信確認間隔を代入 100 LoRaWAN 開発シールドの初期設定処理部 200 LoRaWAN ネットワーク参加処理部 300 画面へ「== ﾒｲﾝ ==」を表示 310 送信処理(500 行)を実行 320 T 秒の待ち時間処理 330 行番号 300 へ戻って繰り返す 500 画面へ「== TX ==」を表示 510 シリアル送信を ON に設定．受信データ破棄 520 配列変数へ待ち受け文字"rx "を代入 530 LoRaWAN 送信コマンドとポート番号を送信 540 0 を送信 550 受信サブルーチン(700 行)へ 560 シリアル出力の無効化 570 サブルーチンを終了し，gosub 部へ戻る 700 LoRaWAN 受信部
・PC 等でダウンロード：　https://git.bokunimo.com/MJ/pg08/33.txt **・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"　（メニュー[33]）**	

LoRaWAN 対応バーコード・リーダの製作

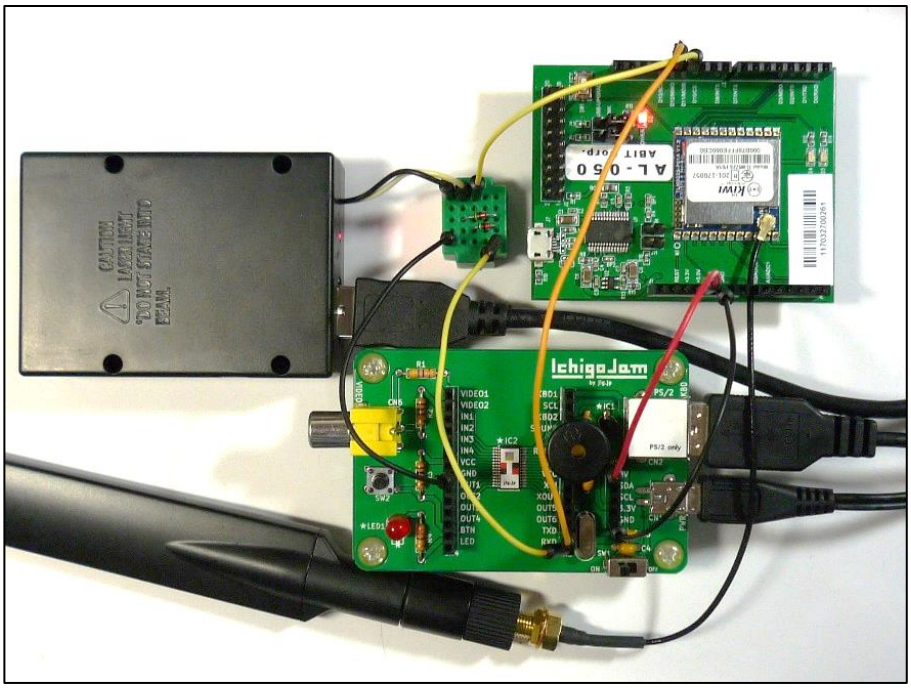

写真 3-5 LoRaWAN 対応バーコード・リーダ の製作例

IchigoJam S に，LoRaWAN 開発シールド AL-050 と，バーコード・リーダ Symcode MJ2090 を接続した．LoRaWAN 開発シールドとバーコード・リーダのシリアル出力はダイオードで合成してから IchigoJam の RXD へ入力した．

バーコード・リーダを IoT 化することで，野菜などの無人直売所での，商品管理や自動レシート発行が出来るようになります．ここでは，写真 3-6 の小型バーコード・リーダ Symcode MJ2090 を IchigoJam へ接続し，読み取ったバーコードを LoRaWAN 送信する機器の製作例を紹介します．なお，レーザ光が目に入ると，失明などの傷害を負う場合があるので，レーザが照射されていないときも絶対に覗き込まないように十分に注意するほか，未成年者を対象にした活用はご遠慮ください．

表 3-4　バーコード・リーダ Symcode MJ2090 の仕様

項目	仕様
種別	可視光レーザ方式バーコードスキャナ
電源電圧	DC 5V (USB)
消費電力（実測）	260mW (レーザ照射中は 245mW)
読取可能距離	40mm～500mm（幅 10～250mm）
インタフェース	USB HID / USB COM

写真 3-6 バーコード・リーダ Symcode MJ2090 の動作時の様子

バーコード・リーダの前面（写真手前）に紙などを近づけると，レーザが照射されバーコードを読み取る．照射されていないときも，絶対に覗き込まないよう，十分に注意する．

バーコード・リーダ Symcode MJ2090 を IchigoJam へ接続するには，本体内の基板からシリアル信号を引き出す必要があります．筐体のビスを外すと写真 3-7 のように RS-232C レベル変換用の 16 ピン IC のランドが見えるので，10 番ピンのシリアル出力 TX の信号と 15 番ピンの GND を引き出してください．写真では 16 番ピンの 5V も引き出していますが，電源については USB 端子から供給しても良いでしょう．配線コードは，写真左側のレーザや拡散ミラーの駆動部などの部位に回り込まないように引き出します．また，動作確認を行うときは，必ずケースの蓋を閉めてください．

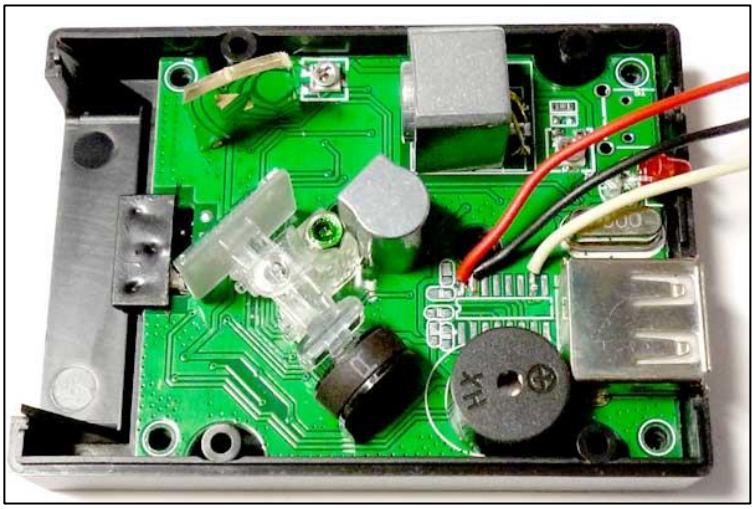

写真 3-7 バーコード・リーダ Symcode MJ2090 からシリアル出力信号を取り出す

バーコード・リーダの基板上の RS-232C 用レベル変換 IC 用ランドから，シリアル出力 TX（10 番ピン），GND（15 番ピン），5V 入力（16 番ピン）を引き出した．電源は USB から供給しても良い．電源を入れる際はケース蓋を必ず取り付ける．

ケースの蓋を閉めたら，電源を入れて，図 3-6 のバーコードを facrory default，RS232C Mode，Baud rate 9600 の順にスキャンし，バーコード・リーダの初期設定を行ってください．スキャン時に「ピポッ」と 2 音が鳴り，設定値は FLASH に保存されます．

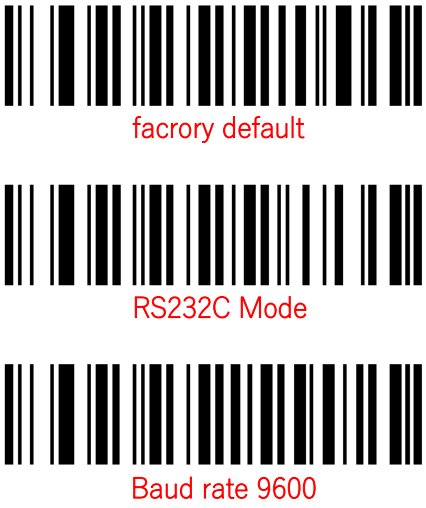

図 3-6 シリアル出力の設定用バーコード

バーコード・リーダ Symcode MJ2090 のインタフェースをシリアル出力へ切り替えるためのバーコード．上から順にスキャンして設定する．

引き出したシリアル出力 TX は，IchigoJam の RXD へ入力しますが，すでに RXD には LoRaWAN 開発シールドが接続されています．これらをダイオードを使って簡易的に合成する回路図を図 3-7 に示します．写真中央の小型のブレッドボード上にシリアルの簡易合成用のダイオードを実装しました．なお，IchigoJam の TXD を LoRaWAN 開発シールドとプリンタで共用するには，抵抗器と OUT 端子を使用します．使用する側の OUT 端子をオープンにし，使用しない側を H レベル出力にすることで，シリアルの切り替えが可能です．

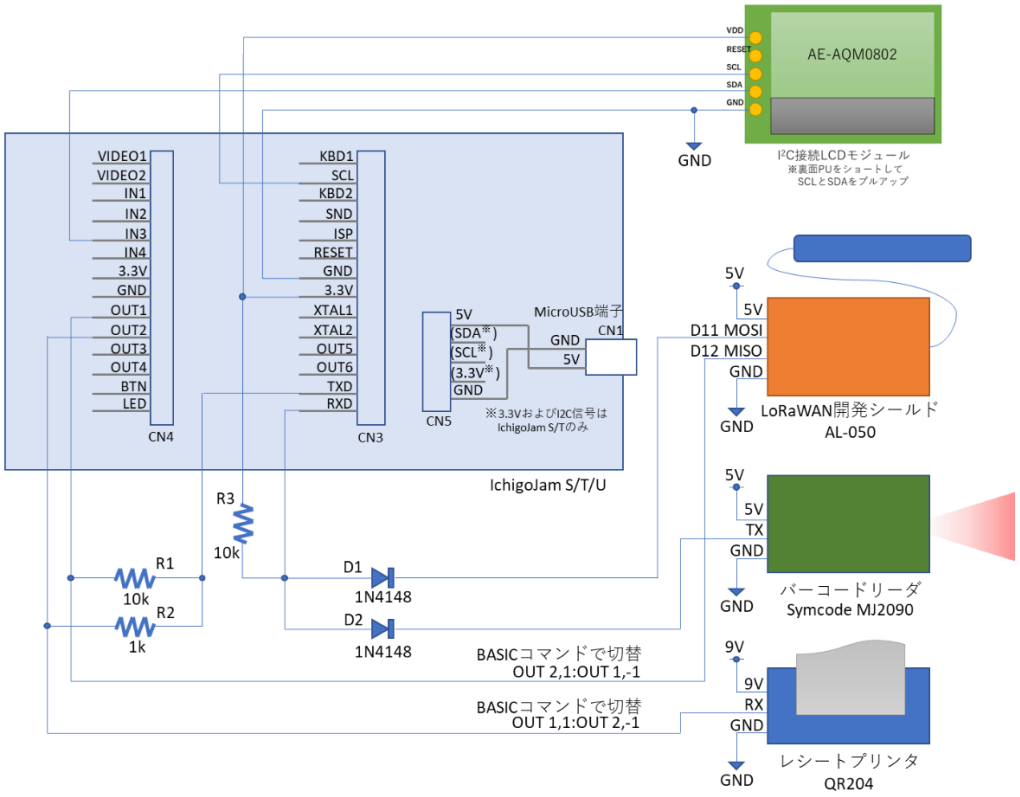

図 3-7 LoRaWAN 対応 IoT 液晶ディスプレイ表示端末／IoT バーコード・リーダの回路図

IchigoJam へ LoRaWAN 開発シールド, LCD モジュール，バーコード・リーダ，レシート・プリンタを接続する場合の回路図．IchigoJam の送信 TXD の接続先は BASIC の OUT コマンドで切り替える．

写真 3-8 バーコード・リーダで読み取ったコードを，レシート・プリンタ QR204 で印刷した

13 桁の JAN コードを読み取り，前後 1 桁を除いた 11 桁のコードを LoRaWAN 送信したときの印字例

プログラムは表 3-5 の一覧のファイル「30.txt」として収録した「LoraWAN Barcode」を使用します．商品管理に良く用いられている JAN コードは 13 桁ですが，LoRaWAN の拡散度 SN10 の制約から，ここでは JAN コードの先頭 1 桁目と末尾の 1 桁を省いた 11 桁を LoRaWAN 送信します．JAN コードの先頭 2 桁は 45 または 49，自主コードは 02 と 04 なので，国内の商品については，これら 4 つのコードを 2 桁目で判断がつきます．また，末尾 1 桁はパリティなので送信不要です．

さらに，バーコードとプリンタを組み合わせた使用例として，バーコードを読むたびにプリンタへコードを印字する「Barcode Reader」をファイル「29.txt」として収録しました．行番号 4 に書かれた 13 桁の待ち受け文字に一致すると，変数 R に 1 を応答し，写真 3-8 のようにコードを印字します．

ここでは，基本的な機器構成とサンプル・プログラムを紹介しましたが，これらの機器やプログラムを実際の用途に合わせて調整することで，より実用的な IoT バーコード・リーダを製作することが出来るでしょう．

[Column] Arduino シールド対応 IchigoJam 用マイコン・ボード

IchigoJam 用コンピュータ電子工作学習キット(販売終了・IF ICH-KIT)に付属するプリント基板を使えば，LoRaWAN 開発シールド AL-050 を簡単に接続することが出来ます．また，スイッチ一つでシールドからシリアル信号を切断し，USB シリアルへ切り替えることも出来るので，パソコンを使った IoT 機器の開発に便利です．スライドスイッチ SW 5 を USB 側にすると，IchigoJam 用マイコンをパソコンに接続することが出来，IJUtilities (https://ijutilities.micutil.com/) などを使えば，パソコンで IchigoJam 用のプログラムを開発することが出来ます．ファームウェア書き換えボタン SW4 は，押すだけで IchigoJam マイコンのファームウェア書き換えモードに設定することが出来ます．液晶ディスプレイや 128 個までのプログラムを保存できる EEPROM も付属します．

プリント基板「Personal Computer」へ LoRaWAN 開発シールド AL-050 を装着し，シールド上の D0 RXD と D11 MOSI，D1 TXD と D 12 MISO をジャンパ・ワイヤで接続し，スライドスイッチ SW 5 を UART IO 側にすれば，本稿のプログラムを動かすことが出来ます．

雑誌「1 行リターンですぐ動く! BASIC I/O コンピュータ IchigoJam 入門」をお買い上げいただいた方であれば，基板 CAD データ（ガーバ・データ）をダウンロードし，プリント基板だけを試作することも出来ます．詳しくは，同誌 P.91 のサポートページをご覧ください．

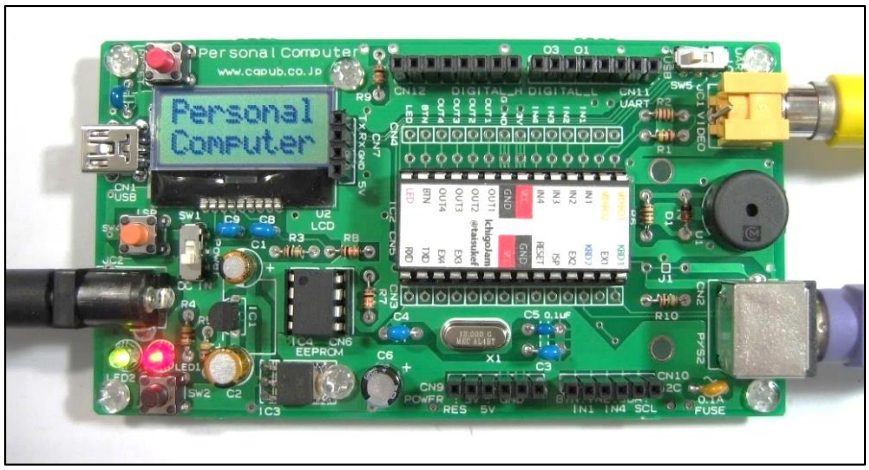

写真 3-9 IF ICH-KIT に含まれている IchigoJam 用プリント基板（販売終了）
LoRaWAN 開発シールド AL-050 を接続可能な拡張端子や USB シリアル切り替えスイッチが実装されている「Personal Computer 基板」

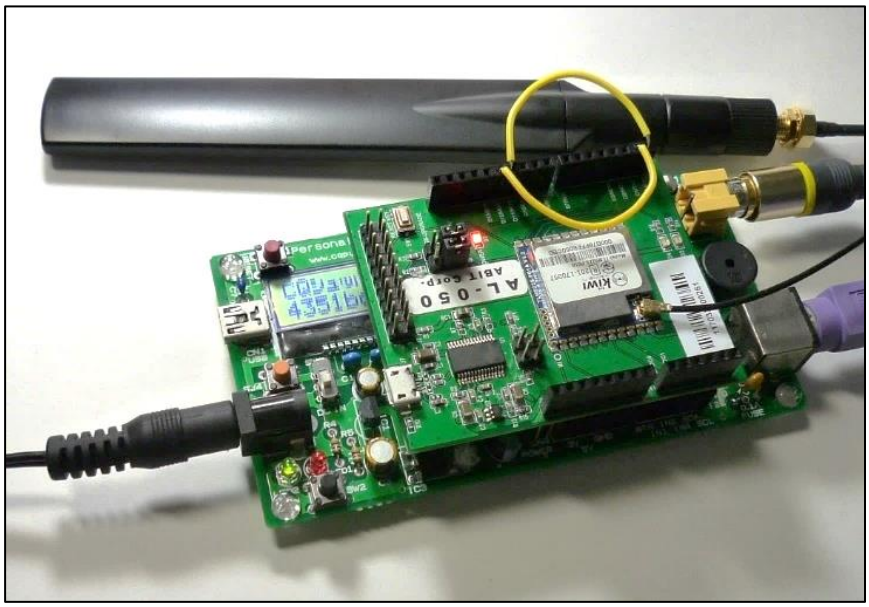

写真 3-10 LoRaWAN 開発シールドを Arduino シールド対応 IchigoJam 用マイコン・ボードに接続した
LoRaWAN 開発シールド AL-050 の D0 RXD と D11 MOSI, D1 TXD と D 12 MISO をジャンパ・ワイヤで接続し，スライドスイッチ SW 5 を UART IO 側にすれば，本稿のプログラムを動かすことが出来る．

[Column] STM32L0 LoRa Discovery Kit による LoRaWAN 接続実験

STM32L0 LoRa Discovery Kit は，村田製作所製の LoRa モジュール CMWX1ZZABZ-091 を搭載した LoRaWAN 評価ボードです．SORACOM が配布するサンプル・プログラム AT_Slave を書き込むことで，LoRaWAN モデムとして使用することが出来ます．AT_Slave を書き込んだ評価ボードを，写真 3-11 のように IchigoJam へ接続し，LoRaWAN 通信を行う実験を行ってみたので紹介します．

SORACOM ユーザ・コンソールから LoRa Discovery Kit を購入し，同封されているダウンロード用の URL からサンプル･プログラムの ZIP ファイルをダウンロードし，サンプル AT_Slave を STM32L0 LoRa Discovery Kit へ書き込みます．サンプルのコンパイルや書き込みには，ARM 社の Keil MDK が必要です．評価ボードに同封されている PSN（ARM 社 Keil 製品インストール用シリアル番号）または，ARM 社が STM32L0/STM32F0 用に無料で配布している PSN を使って，LIC（ライセンス番号）を取得し，アクティベートを実行してください．これらは本格的な製品向けの開発環境です．ダウンロードやインストール，環境設定などに，少なくとも数時間～半日程度の時間を要します．

ハードウェアは，IchigoJam の TXD を LoRaWAN 評価ボードの RX へ，RXD を TX へ接続し，電源 5V と GND をそれぞれ接続して製作します．IchigoJam 用のソフトウェアは，表 3-5 の 38.txt を入力してください．

LoRaWAN 評価ボード上に実装されている金属製のシールドに覆われた部品が村田製作所製 CMWX1ZZABZ-091 です．ほぼマイクロ SD と同じ 12.5×11.5mm のサイズのモジュール内に，ST マイクロエレクトロニクス製の MPU と Semtech 製 LoRa トランシーバ SX1276 を内蔵しています．ここで説明した LoRaWAN モデム機能は，この小さなモジュールだけで動作可能であり，今後，このモジュールを搭載した組み込み機器の登場も期待できます．

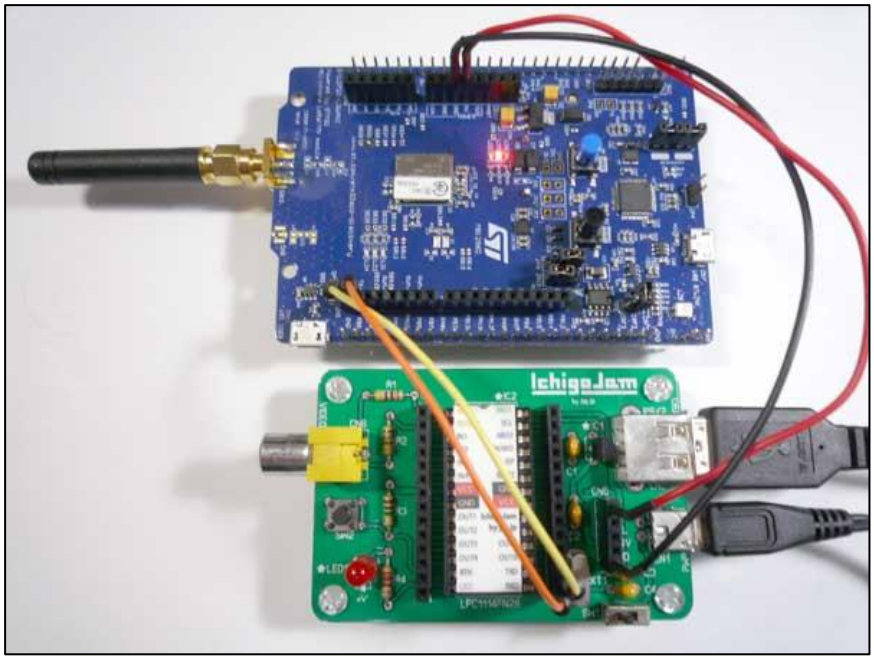

写真 3-11 STM32L0 Discovery Kit を IchigoJam マイコン・ボードに接続したときの様子

IchigoJam の TXD を LoRaWAN 評価ボードの RX へ，RXD を TX へ接続し，電源 5V と GND をそれぞれ接続する．

製作した LoRaWAN 対応プログラム

筆者が作成した各 LoRaWAN 用のプログラムの一覧を表 3-5 に示します. パソコンのブラウザでダウンロードする場合は, 表の下段に書かれた URL 中の「●●」の部分にファイル名の数字を入れてアクセスしてください. 文字化けする場合は, ブラウザの文字エンコード設定を[シフト JIS]に設定して下さい.

MixJuice でダウンロードする場合は表の下段に書かれた IchigoJam BASIC コマンドを IchigoJam へ入力してください.

表 3-5　製作した LoRaWAN 対応プログラムの一覧

ファイル名	プログラム名	備考
31.txt	LoRaWAN AL-050 TX	LoRaWAN 対応 IoT 押しボタン・送信用
32.txt	LoRaWAN AL-050 ANA	LoRaWAN 対応 IoT アナログセンサ・送信用
33.txt	LoRaWAN AL-050 RX	LoRaWAN 対応 IoT 制御・受信用
34.txt	LoRaWAN AL-050 RX TXT	LoRaWAN 対応 IoT 制御・テキスト文字受信用
37.txt	LoRaWAN LCD	LoRaWAN 対応 IoT ディスプレイ
38.txt	LoRaWAN AT Modem	STM LoRa Discovery Kit 用 IoT 押しボタン・送信用
39.txt	Tester Barcode	バーコード読み取りテスト用
30.txt	LoRaWAN Barcode	LoRaWAN 対応 IoT バーコード・リーダ
・PC 等でダウンロード：　http://git.bokunimo.com/MJ/pg08/●●.txt ・MixJuice メニュー形式：　?"MJ GET git.bokunimo.com/MJ/808.txt"		

実運用での注意点

紹介したマイコン・ボードや製作方法は，一般の家庭用の電気製品向けではありません．保護や動作保障用の回路部品が省かれていたり，コネクタ振動などや挿抜によって傷みやすかったりする場合があります．

また，ここではBASIC言語を簡単な手続きプログラム用として使用しました．BASIC言語は，拡張性が低く，またプログラムの構造化が行いにくいので，プログラムが長くなるほど人的ミスによる不具合の発生率が高まります．IchigoJam BASICには，16bit整数型変数を基本としていることや，一つのプログラムの容量が1KBまでと限られているなどの制限もあります．BASIC以外のプログラミング言語に慣れているのであれば，BASICを選択することがリスクとなる場合もあるでしょう．

以上のようにハードウェアやソフトウェアの信頼性が低いと，その対策費用がかさむことに配慮する必要はあるものの，小規模事業における最大のリスクが開発コストであることを考えると，一定の妥協による品質とコストのバランスの取れたシステム設計が重要でしょう．

おわりに

コンピュータによるネットワーク技術は，総合的な処理能力を高めるために，集中処理型から分散処理型に変化し続けており，初期のホスト集中型からネットワーク化，インターネット化，クラウド化と分散処理方向に進んでいます．このように，ネットワーク技術が分散化へ進んでいるなか，実システムのアプリケーションとしては，大規模なシステムが小規模システムを呑みこみ，規模による集中の傾向もみられます．こういった大規模投資によるビジネスの優位性が目立ち，小規模システムの開発投資に二の足を踏むこともあるでしょう．

しかし，分散化によって能力を高めるというネットワーク技術の本質を考えれば，個人向けや実ビジネスにおいても，小規模なシステムこそIoTを活用した新サービスの展開や事業効率化の効果がもたらされ，個人や個々の事業者による総力の発展につながると考えられます．

例えば，小売り事業者はエンドユーザにより近い場所に位置し，一方，IoT技術を支えるデータセンターやクラウド上のサービスプラットフォームは大きな投資規模によって実現される役割分担があると思います．今後，社会全体の処理能力を高める方向に淘汰されたとき，こういった役割分担が明確になり，個人や小規模な事業こそ，大きな発展を遂げる日が来るかもしれません．

本書が，そのような時代に向う流れの中の小さな役割になればという気持ちで，執筆しました．

2018年8月

国野　亘

bokunimo.net/15

履歴（変更内容）

2018 年 05 月 25 日	初稿執筆開始
2018 年 08 月 19 日	初稿執筆完了
2021 年 02 月 07 日	ネット配布用リリース （ダウンロード URL，コラム IOT コマンド，コラム LINE 簡単送信など）
2021 年 02 月 08 日	ダウンロード URL のリンク先の修正漏れ，誤字，体裁の修正

権利情報

by bokunimo.net