Armadillo-IoT ゲートウェイ スタンダードモデル 開発セット スタートアップガイド

AG421-D00Z
AG421-D03Z

Version 2.0.0
2015/06/23

株式会社アットマークテクノ [http://www.atmark-techno.com]

Armadillo サイト [http://armadillo.atmark-techno.com]

Armadillo-IoT ゲートウェイスタンダードモデル 開発セット スタートアップガイド

株式会社アットマークテクノ

札幌本社

〒 060-0035 札幌市中央区北 5 条東 2 丁目 AFT ビル
TEL 011-207-6550 FAX 011-207-6570

横浜営業所

〒 221-0835 横浜市神奈川区鶴屋町 3 丁目 30-4 明治安田生命横浜西口ビル 7F
TEL 045-548-5651 FAX 050-3737-4597

Version 2.0.0
2015/06/23

目次

図目次

表目次

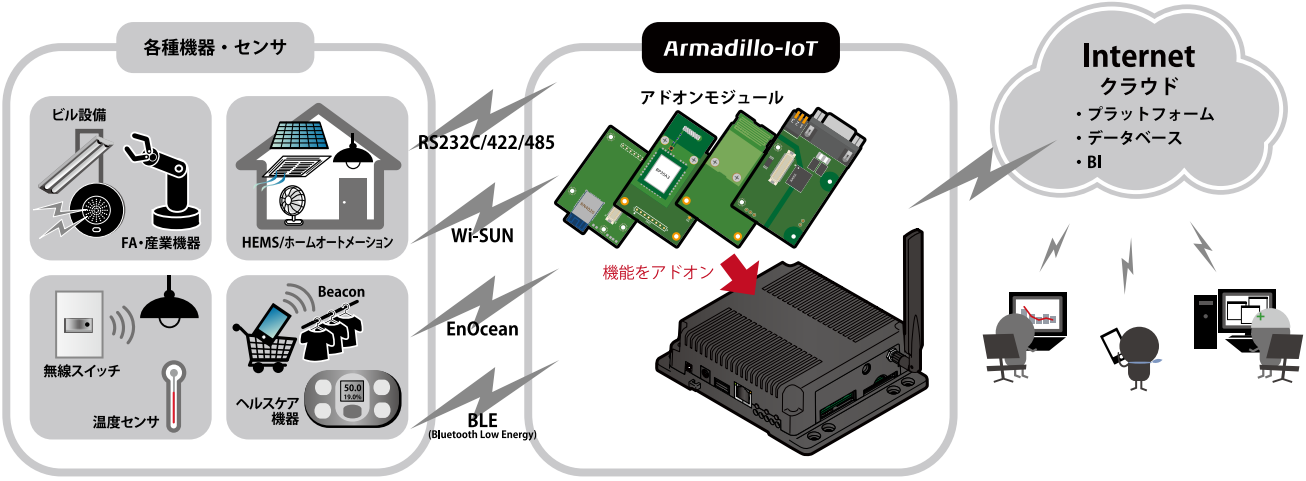
1. はじめに

このたびは Armadillo-IoT ゲートウェイ スタンダードモデル G2 開発セットをご利用いただき、ありがとうございます。

Armadillo-IoT ゲートウェイ スタンダードモデル G2(以下、Armadillo-IoT)は、各種センサとネットワークとの接続を中継する IoT 向けゲートウェイの開発プラットフォームです。ハードウェアやソフトウェアをカスタマイズして、オリジナルのゲートウェイを素早く、簡単に開発することができます。

Armadillo-IoT は、センサ接続用インターフェースとして、RS232C/422/485、接点入出力など一般的なセンサ接続に広く使われるインターフェースの他、EnOcean や Wi-SUN、Bluetooth Low Energy など新しい省電力無線通信規格にも対応しています。これらの機能は専用の「アドオンモジュール」を付け替えることで、用途に応じて柔軟に構成できます。アドオンモジュールのインターフェース仕様は公開されているので、必要に応じてオリジナルのアドオンモジュールを開発することもできます。また、WAN(Wide Area Network)用インターフェースとして、LAN、無線 LAN(IEEE 802.11b/g/n)の他、モバイル通信(3G)も利用可能です。

Armadillo-IoT は標準 OS として Linux がプリインストールされているため、オープンソースソフトウェアを含む多くのソフトウェア資産を活用し、自由にオリジナルのアプリケーションを開発することができます。開発言語としては、C/C++言語だけでなく、Java や Ruby などをサポートしています。さらに MQTT クライアントなど、クラウドサービスと親和性の高いソフトウェアスタックが用意され、ソフトウェア面でも開発の自由度と開発しやすさの両立を図っています。



本書は、センサから取得したデータをクラウド上の Web アプリケーションに送信するサンプルプログラムを通じて Armadillo-IoT の操作方法を理解し、IoT/M2M を簡単に体感できるように構成されています。

Armadillo-IoT のさらに進んだ使い方を知りたい場合は、Armadillo サイトの Howto・FAQ [http://armadillo.atmark-techno.com/armadillo-iot/techinfo]を参照してください。様々な活用方法が掲載されています。また、Armadillo-IoT の詳細な仕様を知りたい場合は「Armadillo-IoT ゲートウェイ スタンダードモデル 製品マニュアル」を参照してください。アットマークテクノ ユーザーズサイト [https://users.atmark-techno.com/]では、製品の使い方や開発方法についてユーザー同士で情報交換できるフォーラムや、開発者ブログを運営しています。

ユーザーズサイトで購入製品登録を行うと、製品をご購入いただいたユーザーに限定して公開している限定コンテンツにアクセスできるようになります。主な限定コンテンツには、下記のものがあります。

- リカバリ用ユーザーランドイメージ(工場出荷時と同等のもの)
- アドオンモジュール回路図
- 各種信頼性試験データ・納入仕様書等製造関連情報

限定コンテンツを取得するには、「6. ユーザー登録」を参照してください。「6. ユーザー登録」で利用する ATDE の起動方法については、製品マニュアルを参照してください。

以降、本書では他の Armadillo ブランド製品にも共通する記述については、製品名を Armadillo と表記します。

1.1. 本書および関連ファイルのバージョンについて

本書を含めた関連マニュアル、ソースファイルやイメージファイルなどの関連ファイルは最新版を使用することをおすすめいたします。本書を読み始める前に、Armadillo サイトで最新版の情報をご確認ください。

Armadillo サイト - Armadillo-IoT ゲートウェイ スタンダードモデル ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-iot/downloads

1.2. 対象となる読者

- ハードウェアの動作確認をされる方
- ソフトウェアの基本的な使用方法の確認をされる方

上記以外の方でも、本書を有効に利用していただけたら幸いです。

1.3. 本書の構成

本書では、Armadillo-IoT の基本的な操作方法を理解し、サンプルアプリケーションを通して IoT/M2M に触れ、オリジナルのアプリケーション開発を始めるために必要となる情報を説明しています。

以下に主な項目を挙げます。

- サンプルアプリケーションの概要
- Armadillo-IoT と周辺機器との接続方法
- 初期設定方法
- サンプルアプリケーションの使用方法
- サンプルアプリケーションのカスタマイズ方法

1.4. 表記について

1.4.1. フォント

本書では以下のような意味でフォントを使いわけています。

表 1.1 使用しているフォント

フォント例	説明
本文中のフォント	本文
[PC ~]$ ls	プロンプトとユーザ入力文字列
text	編集する文字列や出力される文字列。またはコメント

1.4.2. コマンド入力例

本書に記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「/」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「~」で表わします。

表 1.2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC 上の root ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[armadillo /]#	Armadillo 上の root ユーザで実行
[armadillo /]$	Armadillo 上の一般ユーザで実行
hermit>	Armadillo 上の保守モードで実行

コマンド中で、変更の可能性のあるものや、環境により異なるものに関しては以下のように表記します。適時読み替えて入力してください。

表 1.3 コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号

1.4.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

1.5. 謝辞

Armadillo で使用しているソフトウェアの多くは Free Software / Open Source Software で構成されています。Free Software / Open Source Software は世界中の多くの開発者の成果によってなりたっています。この場を借りて感謝の意を表します。

2. 注意事項

2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。

- ご使用の前に必ず製品マニュアルおよび関連資料をお読みになり、使用上の注意を守って正しく安全にお使いください。
- マニュアルに記載されていない操作・拡張などを行う場合は、弊社 Web サイトに掲載されている資料やその他技術情報を十分に理解した上で、お客様自身の責任で安全にお使いください。
- 水・湿気・ほこり・油煙等の多い場所に設置しないでください。火災、故障、感電などの原因になる場合があります。
- 本製品に搭載されている部品の一部は、発熱により高温になる場合があります。周囲温度や取扱いによってはやけどの原因となる恐れがあります。本体の電源が入っている間、または電源切断後本体の温度が下がるまでの間は、基板上の電子部品、及びその周辺部分には触れないでください。
- 本製品を使用して、お客様の仕様による機器・システムを開発される場合は、製品マニュアルおよび関連資料、弊社 Web サイトで提供している技術情報のほか、関連するデバイスのデータシート等を熟読し、十分に理解した上で設計・開発を行ってください。また、信頼性および安全性を確保・維持するため、事前に十分な試験を実施してください。
- 本製品は、機能・精度において極めて高い信頼性・安全性が必要とされる用途(医療機器、交通関連機器、燃焼制御、安全装置等)での使用を意図しておりません。これらの設備や機器またはシステム等に使用された場合において、人身事故、火災、損害等が発生した場合、当社はいかなる責任も負いかねます。
- 本製品には、一般電子機器用(OA 機器・通信機器・計測機器・工作機械等)に製造された半導体部品を使用しています。外来ノイズやサージ等により誤作動や故障が発生する可能性があります。万一誤作動または故障などが発生した場合に備え、生命・身体・財産等が侵害されることのないよう、装置としての安全設計(リミットスイッチやヒューズ・ブレーカー等の保護回路の設置、装置の多重化等)に万全を期し、信頼性および安全性維持のための十分な措置を講じた上でお使いください。
- 無線 LAN 機能を搭載した製品は、心臓ペースメーカーや補聴器などの医療機器、火災報知器や自動ドアなどの自動制御器、電子レンジ、高度な電子機器やテレビ・ラジオに近接する場所、移動体識別用の構

内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

2.2. 取扱い上の注意事項

本製品に恒久的なダメージをあたえないよう、取扱い時には以下のような点にご注意ください。

破損しやすい箇所	BtoB コネクタは破損しやすい部品になっています。無理に力を加えて破損することのないよう十分注意してください。
本製品の改造	本製品に改造[1]を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設[2]を行う場合は、作業前に必ず動作確認を行ってください。
電源投入時のコネクタ着脱	本製品や周辺回路に電源が入っている状態で、活線挿抜対応インターフェース(LAN、SD/SDIO、USB)以外へのコネクタやカードの着脱は、絶対に行わないでください。
静電気	本製品には CMOS デバイスを使用しており、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。
ラッチアップ	電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等により、使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには、保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。
衝撃	落下や衝撃などの強い振動を与えないでください。

2.3. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて	本製品の標準出荷状態でプリインストールされている Linux 対応ソフトウェアは、個別に明示されている（書面、電子データでの通知、口頭での通知を含む）場合を除き、オープンソースとしてソースコードが提供されています。再配布等の権利については、各ソースコードに記載のライセンス形態にしたがって、お客様の責任において行使してください。また、本製品に含まれるソフトウェア（付属のドキュメント等も含む）は、現状有姿（AS IS）にて提供します。お客様ご自身の責任において、使用用途・目的の適合について事前に十分な検討と試験を実施した上でお使いください。アットマークテクノは、当該ソフトウェアが特定の目的に適合すること、ソフトウェアの信頼性および正確性、ソフトウェアを含む本製品の使用による結果について、お客様に対し何らの保証も行いません。 パートナー等の協力により Armadillo ブランド製品向けに提供されているミドルウェア、その他各種ソフトウェアソリューションは、ソフトウェア毎にライセンスが規定されています。再頒布権等については、各ソフトウェ

[1] コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2] コネクタを増設する際にはマスキングを行い、周囲の部品に半田くず、半田ボール等付着しないよう十分にご注意ください。

アに付属する readme ファイル等をご参照ください。その他のバンドルソフトウェアについては、各提供元にお問い合わせください。

本製品の標準出荷状態でプリインストールされている以下のソフトウェアは、オープンソースソフトウェアではありません。

- Oracle Java SE Embedded 8
- ボード情報取得ツール(get_board_info)

2.4. 書込み禁止領域について

EEPROM および i.MX257 内蔵エレクトリカルヒューズ(e-Fuse)のデータは、本製品に含まれるソフトウェアで使用しています。正常に動作しなくなる可能性があるため、書込みを行わないでください。また、意図的に書込みを行った場合は保証対象外となります。

2.5. 保証について

本製品の本体基板は、製品に添付もしくは弊社 Web サイトに記載している「製品保証規定」に従い、ご購入から 1 年間の交換保証を行っています。添付品およびソフトウェアは保証対象外となりますのでご注意ください。

製品保証規定 http://www.atmark-techno.com/support/warranty-policy

2.6. 輸出について

- 当社製品は、原則として日本国内での使用を想定して開発・製造されています。
- 海外の法令および規則への適合については当社はなんらの保証を行うものではありません。
- 当社製品を輸出するときは、輸出者の責任において、日本国および関係する諸外国の輸出関連法令に従い、必要な手続を行っていただきますようお願いいたします。
- 日本国およびその他関係諸国による制裁または通商停止を受けている国家、組織、法人または個人に対し、当社製品を輸出、販売等することはできません。
- 当社製品および関連技術は、大量破壊兵器の開発等の軍事目的、その他国内外の法令により製造・使用・販売・調達が禁止されている機器には使用することができません。

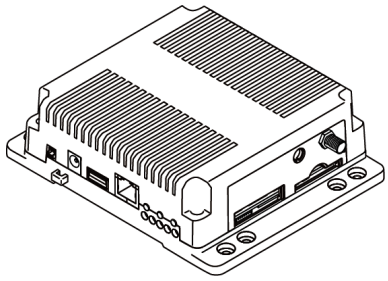
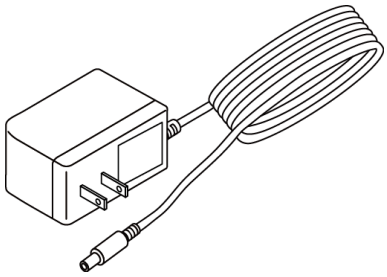
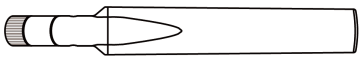
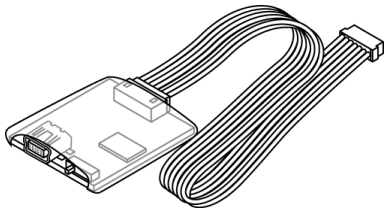
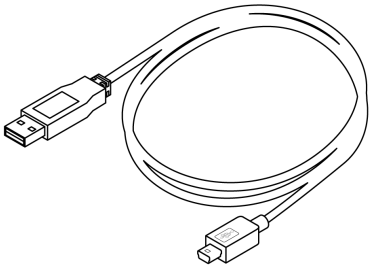
2.7. 商標について

- Armadillo は株式会社アットマークテクノの登録商標です。その他の記載の商品名および会社名は、各社・各団体の商標または登録商標です。™、®マークは省略しています。
- SD、SDHC、SDXC、microSD、microSDHC、microSDXC、SDIO ロゴは SD-3C, LLC の商標です。

3. 作業の前に

3.1. スタンダードモデル開発セット同梱物の確認

お使いになる前に、次の同梱物がすべて揃っていることをご確認ください。 万一、同梱物の不足または部品の破損等がございましたら、ご購入の販売代理店までご連絡ください。

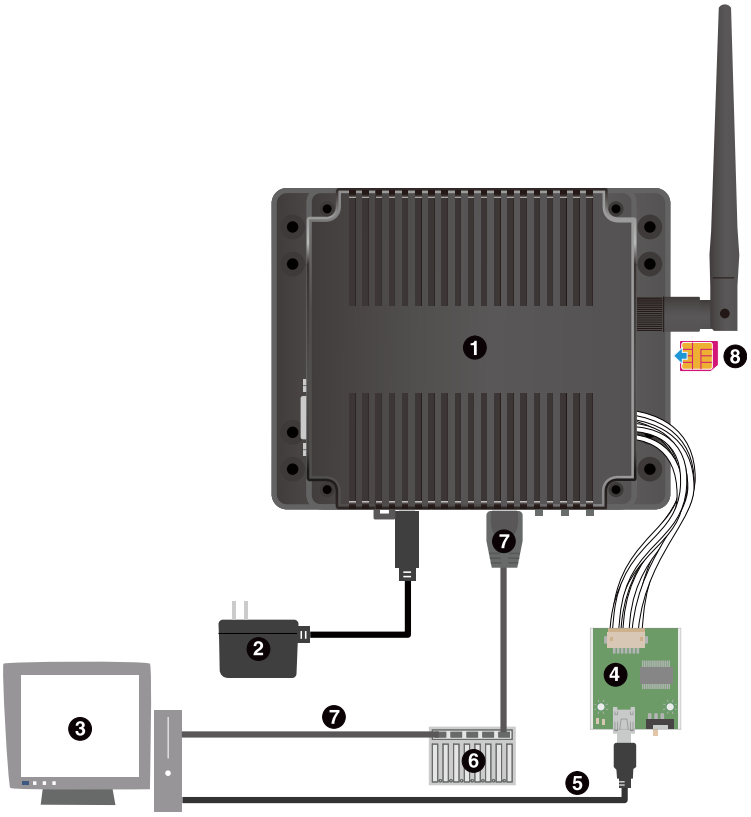
■ Armadillo-Io T ゲートウェイ スタンダードモデル 本体[a]	■ 開発用 DVD-ROM	■ AC アダプタ(12V)
		
■ 3G モジュール用アンテナ	■ USB シリアル変換アダプタ	■ USB ケーブル(A-miniB タイプ)
		

[a]3G モジュール、及び、WLAN モジュール、RS232C アドオンモジュールは内蔵されています

3.2. 接続方法

Armadillo-IoT と周辺装置の接続例を次に示します。Armadillo-IoT の基本的な操作は、シリアルコンソールから行います。作業用 PC とは USB シリアル変換アダプタを介して接続します。また、開発用 PC とのファイル転送やインターネットへの接続には、LAN を介して行います[1]ので、LAN ケーブルを接続してください。

[1]適切な設定を行うことで、WLAN やモバイル通信(3G)を介してインターネットへ接続することもできます。



1. Armadillo-IoT ゲートウェイ スタンダードモデル
2. AC アダプタ(12V)[2]
3. 作業用 PC
4. USB シリアル変換アダプタ[2]
5. USB ケーブル(A-miniB タイプ)[2]
6. LAN HUB
7. LAN ケーブル
8. microSIM カード

図 3.1 Armadillo-IoT ゲートウェイ スタンダードモデルの接続例

3.3. USB シリアル変換アダプタ

USB シリアル変換アダプタは、FTDI 社製 FT232RL を搭載した USB-シリアル変換アダプタです。

作業用 PC のドライバは OS で提供されているものを利用するか、FTDI 社の Web サイトよりダウンロードしてください。

[2]Armadillo-IoT ゲートウェイ スタンダードモデル開発セット付属品

FTDI - Drivers

http://www.ftdichip.com/FTDrivers.htm

3.4. シリアルターミナルソフトウェア

作業用の PC から Armadillo のシリアルコンソールに接続する場合、作業用 PC のシリアル通信ソフトウェア[3]の設定を、次に示す表のように設定してください。また、シリアル通信ソフトウェアの横幅を 80 文字以上にしてください。横幅が 80 文字より小さい場合、コマンド入力中に表示が乱れることがあります。

表 3.1 シリアル通信設定

項目	設定
転送レート	115,200 bps
データ長	8 bit
ストップビット	1 bit
パリティ	なし
フロー制御	なし

[3]Linux や Mac では「minicom」、Windows では「Tera Term Pro」などです。

4. IoT を体験しよう

Armadillo-IoT を使い、データをクラウド上のサーバーにアップロードするサンプルアプリケーションを動かし、IoT の基本部分を体験してみましょう。

4.1. サンプルアプリケーションの概要

サンプルアプリケーションは、Armadillo-IoT 上で動作するクライアントサイドアプリケーションとクラウドプラットフォーム上で動作するサーバーサイド Web アプリケーションの 2 つのアプリケーションから構成されています。サンプルアプリケーションは、Armadillo-IoT(クライアント)からデータをサーバーにアップロードする機能と、サーバーからクライアントに対して操作指示(Push 通知)を行う機能の二つを備えています。

クライアントからサーバーへのデータアップロードの例として、Armadillo-IoT が本体の状態監視用に内蔵している温度センサの値をアップロードします[1]。また、サーバーからの Push 通知の例として、ブラウザに表示されたボタンをクリックすると、その状態(ON または OFF)を Armadillo-IoT に通知します。

クライアントサイドアプリケーションは、Ruby のコンソール(CLI)アプリケーションとして実装されています。実装を簡単にするために、Ruby 用ライブラリの一般的な配布形式である、gem[2]をいくつか使用しています。データのアップロードは REST API(HTTP POST)で行うため、rest-client [https://rubygems.org/gems/rest-client]を使用します。また、サーバーからの Push 通知には Pusher[3]を使用しているため、pusher-client [https://rubygems.org/gems/pusher-client]を使用します。

サーバーサイド Web アプリケーションは、クラウドアプリケーション開発プラットフォームの Heroku 上で動作しています。Node.js で実装されており、クライアントサイドアプリケーションからのデータを受信するとデータベース(PostgreSQL)に格納します。格納したデータは、Web ブラウザからリアルタイムに変化を確認できるようになっています。また、ブラウザ上に表示されたボタンをクリックすると、それがクライアント(Armadillo-IoT)への操作指示として、Push 通知されるようになっています。

[1]GPS データもアップロードしますが、開発セットには GPS 用アンテナが同梱されていないため、ダミーデータを使います。
[2]https://www.ruby-lang.org/ja/libraries/ 参照。
[3]Pusher Ltd.が提供する Websocket 技術を使用したサーバーからクライアントへの Push 通知サービス。

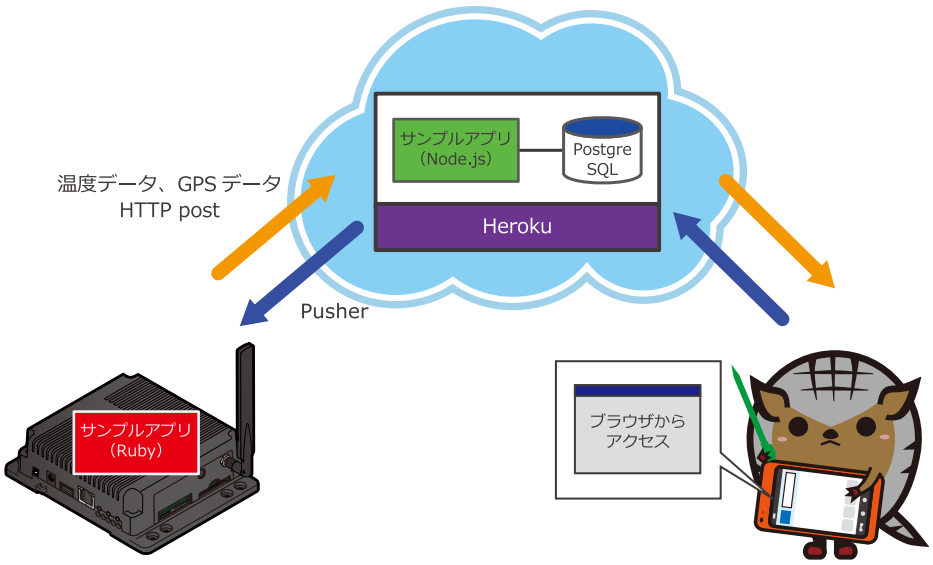


図 4.1 サンプルアプリケーションの構成

REST API

REST(REpresentational State Transfer)は、Web アプリケーションの API に関するアーキテクチャスタイル(設計思想)の一つです。

ネットワーク上のリソースに対し一意の URI を割り当て、リソースに対する操作を HTTP GET(取得)/POST(新規作成)/PUT(更新)/DELETE(削除)で行います。REST の原則に従い API を設計すると、クライアントからのリクエストをステートレスにできるため、サーバー側の実装をスケールアウトしやすいと言われています。サンプルアプリケーションでは、時々刻々と変化する温度センサデータのアップロードは、常にリソース(温度データの時系列情報)の新規作成となるため HTTP POST で行っています。

なお、厳密には REST 原則に従っていない API も REST API と称することが多くなっているため、原則に忠実に従って設計された API のことを、RESTful API ということもあります。

Heroku

Heroku(ヘロク) [https://www.heroku.com/]は、Web アプリケーションの開発、公開が簡単に行えるクラウド上のプラットフォームです。

一昔前の Web アプリケーション開発現場では、まず開発環境に Linux や Apache、MySQL、PHP(LAMP)をインストールして開発を行い、開発がうまくいったら公開用のサーバーを購入するなり、レンタルサーバーを借りるなりして、また色々インストールして、一般公開するためにドメインを購入して、というようにアプリケーションの実装以外に多くの付随する作業が必要でした。

Heroku を使うと、1. Heroku アカウントを作成する、2. アプリケーションを実装する(アプリケーションの実行に必要なサーバーやデータベースの設定は、Heroku のツールが面倒を見てくれるので、開発者は何を使うか指定するだけです)、3. サーバーに「push」する、という単純なステップで、アプリケーションを一般公開できます。また、サーバーで使用する CPU の数やメモリサイズ、データベース容量を動的に変更できるなど、急にアクセスが増えた場合でも、簡単にスケールアウトできるような仕掛けがなされています。

また、Web アプリケーションを開発する際に必要となる様々な機能が「アドオン」として用意されており、それらを活用することで、アプリケーションの本質的な部分に開発リソースを集中することができます。サンプルアプリケーションでは、データベースとして PostgreSQL を、Push 通知を行うために Pusher のアドオンを使用しています。

課金体系としてフリーミアムモデルを採用しており、一定時間までは無料で使用することができます。このことも、利用の敷居を下げる一因になっています。サンプルプログラムも無料枠の範囲内で運用しています。

Node.js

Node.js は、javascript で構成されたサーバーサイドアプリケーションの開発フレームワークです。わざわざサーバーサイドと謳っているのは、Node.js 登場以前は、javascript はクライアント(ここでは Web ブラウザのことを意味します)上で動作するプログラムに使用することが一般的であったためです。

Node.js が登場した背景には、10,000 台のクライアントが一斉に Web アプリケーションにアクセスすると急激に性能が劣化する、いわゆる C10K 問題と呼ばれる課題がありました。これは、それまでの Web アプリケーション開発フレームワークが、クライアントからのリクエストがあるたびに、プロセス(ないしはスレッド)を生成する構成になっていたため、接続が重複すると、サーバーのメモリを使い切ってしまう事が一因です[4]。

Node.js では、非同期 I/O を活用し一つのプロセスで全てのリクエストを処理することで、この課題の解決を試みています。Web アプリケーションのサーバーサイドロジックが単純な場合、サーバーサイドアプリケーションはほとんどの時間、I/O 待ちをしていることになります(CPU bound ではなく、I/O bound な状態)。非同期 I/O を使うと、その間も他の処理ができるため、一つのプロセスで複数のリクエストを捌くことができるようになります。

例えば、API を RESTful にしておくと、HTTP POST(新規作成)リクエストは、クライアントから受けとったデータを(ほとんど何も処理せずに)DB に書き込むだけになり、HTTP GET(取得)は DB からデータを取得してレスポンスを返すだけになるため、I/O bound な状態となります。このようなアプリケーションの場合、Node.js のアプローチは非常にうまくスケー

[4]一つのプロセスあたり約 2MB 必要となる場合、8GB のメモリを積んだサーバーでも同時に捌けるリクエストは 4,000 個が限界という計算になります。

ルします。サンプルアプリケーションはまさにこのような状況なため、Node.js を使って実装しています。

反対に、Node.js では数値計算などが必要で CPU 処理が長くなるアプリケーションには適しません。また、非同期 I/O を活用することで、必然的にイベント駆動型のプログラミングスタイルが強制されるため、複雑なロジックが必要なアプリケーションでは、それに応じてコードも複雑化しがちです。どのようなアプリケーションにも適しているわけではない点には注意が必要です。

4.1.1. サンプルアプリケーションの API

クライアントから https://armadillo-iot-sample.herokuapp.com/api/series という URI に対して、下記の内容で HTTP POST リクエストを発行すると、データが格納されます。

表 4.1 POST 時の QUERY パラメータ

パラメータ名	値	型
uid	UID	character varing(制限付き可変長文字列) 255 文字
latitude	緯度	double precision(8byte 浮動小数点)
longitude	経度	double precision(8byte 浮動小数点)
value	摂氏温度	real(4byte 浮動小数点)
created_at	温度取得時刻	timestamp without time zone(日付と時刻両方、時間帯なし)
token	トークン ID("12345")	文字列

uid は、重複しない値であれば何でも構いません。クライアントサイドサンプルアプリケーションでは、MAC アドレスから「:」を抜いた文字列を使用します。サンプルアプリケーションは、実装の簡単な例を示すことを目的としているため、誰でもデータをアップロードし、閲覧することができます。token は、Armadillo-IoT のことを全く知らない人が、簡単にデータをアップロードできないようするための簡易的な仕掛けとして使用します。

データアップロードは単純な HTTP POST リクエストですので、例えば curl コマンドを使用し、下記のように実行することでも、データをアップロードすることができます[5]。自分でクライアントアプリを実装する際の参考としてください。

```
[armadillo ~]# curl -d
”uid=00110cxxxxxx&latitude=35.681382&longitude=139.766084&value=25.00&created_at=$(date ”+%Y-%m-
%d %H:%M:%S”)&token=12345” http://armadillo-iot-sample.herokuapp.com/api/series
```

図 4.2 curl を利用して POST する例

サーバーからの Push 通知は、下記のパラメータで行われます。

表 4.2 Push 通知パラメータ

パラメータ名	型	値
チャンネル	文字列	UID
イベント	文字列	"button"
データ	JSON 文字列	{"value": "on"} か {"value": "off"}

[5]そのまま実行すると、「00110cxxxxxx」という uid でデータが作成されてしまいます。uid は適当に変えて実行してください。

4.2. 電源投入から起動まで

Armadillo-IoT に電源を投入する前に、USB シリアル変換アダプタのスライドスイッチを次のように外側になるようにしてください。[6]

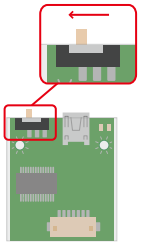


図 4.3 スライドスイッチの設定

Armadillo-IoT に電源を接続すると、シリアル通信ソフトウェアには次のように表示されます。

```
Hermit-At v3.5.0 (armadillo-iotg-std) compiled at 20:38:28, Jun 12 2015
hermit>
```

図 4.4 電源を投入直後のログ

Linux システムを起動するには、次のように "boot"コマンドを実行してください。コマンドを実行するとブートローダーが Linux システムを起動させます。シリアル通信ソフトウェアには Linux の起動ログが表示されます。

```
Hermit-At v3.5.0 (armadillo-iotg-std) compiled at 20:38:28, Jun 12 2015
hermit> boot
Uncompressing  kernel.....................................................done.
Uncompressing ramdisk.....................................................
.....................................................................done.
Booting Linux on physical CPU 0x0
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.14-at1 (atmark@atde5) (gcc version 4.6.3 (Debian4.6.3-14atmark1)
 ) #1 PREEMPT Sat Jun 13 10:48:43 JST 2015
CPU: ARM926EJ-S [41069264] revision 4 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: Armadillo-410
    :
    : (割愛)
    :
atmark-dist v1.41.0 (AtmarkTechno/Armadillo-IoTG-Std)
Linux 3.14-at1 [armv5tejl arch]

armadillo-iotg login:
```

図 4.5 boot コマンドで Linux を起動

[6] スライドスイッチ機能の詳細については、製品マニュアルに記載されています。

Linux システムの起動後に表示される次のメッセージは、エラーメッセージではありません。

```
random: nonblocking pool is initialized
```

このメッセージは、/dev/urandom が内部的に使用するプール領域の初期化完了を示します。

ここでは、root ユーザーでログインします。デフォルトのパスワードは、"root"となっています。

```
atmark-dist v1.41.0 (AtmarkTechno/Armadillo-IoTG-Std)
Linux 3.14-at1 [armv5tejl arch]

armadillo-iotg login: root
Password:
[root@armadillo-iotg (ttymxc1) ~]#
```

図 4.6 root ユーザーでログイン

4.3. 初期設定

クライアントサイドサンプルアプリケーションを動作させるためには、次の項目を設定する必要があります。

- ネットワーク
- 時刻

4.3.1. ネットワークを設定する

Armadillo-IoT は、WAN 用インターフェースとして、LAN、無線 LAN、及びモバイル通信(3G)を備えています。ここでは、LAN を使ってネットワークに接続する設定方法について説明します。無線 LAN、3G の設定方法や、LAN の設定方法の詳細については、製品マニュアルを参照してください。

4.3.1.1. LAN を設定する

Armadillo-IoT では、通常の Linux システムと同様、ネットワークインターフェースの設定は/etc/network/interfaces に記述します。Armadillo-IoT の出荷時の設定では、eth0(LAN のネットワークインターフェース)が自動で up し、DHCP でネットワーク設定を取得するようになっています。

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
iface umts0 inet dhcp
        pre-up 3g-connect
        post-up 3g-monitor start
        pre-down 3g-monitor stop
        post-down 3g-disconnect
```

図 4.7 出荷状態のネットワーク設定

もし、固定 IP に設定したい場合など、ネットワーク設定を変更したい場合、一度ネットワークインターフェースを down してから、設定を変更してください。

```
[armadillo ~]# ifdown eth0
[armadillo ~]# vi /etc/network/interfaces
```

図 4.8 ネットワークインターフェースを down し設定を変更する

固定 IP(IP アドレス: 192.0.2.100、デフォルトゲートウェイ: 192.0.2.1、サブネットマスク: 255.255.255.0)に設定する場合の記述例を下記に示します。

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)

auto lo eth0
iface lo inet loopback
iface eth0 inet static
netmask 255.255.255.0
gateway 192.0.2.1
address 192.0.2.100
iface umts0 inet dhcp
        pre-up 3g-connect
        post-up 3g-monitor start
        pre-down 3g-monitor stop
        post-down 3g-disconnect
```

図 4.9 固定 IP の設定例

変更した設定は、**ifup** コマンドで反映されます。

```
[armadillo ~]# ifup eth0
```

図 4.10 ネットワークインターフェースを up し設定変更を反映する

DNS サーバーの指定は/etc/resolv.conf に記述します。DNS サーバーの IP アドレスを 192.0.2.2 に設定する場合の記述例を下記に示します。

```
nameserver 192.0.2.2
```

図 4.11 DNS サーバーの設定例

DHCP を利用している場合には、DHCP サーバーが DNS サーバーを通知する場合があります。この場合、/etc/resolv.conf は自動的に更新されます。

なお、この設定は**再起動すると消えてしまいます** 。これに限らず、Armadillo-IoT ではルートファイルシステムのファイルに加えた変更は再起動すると全て元通りとなります。これは、Armadillo-IoT のユーザーランドが RAMDISK 上にあるためです。これは、再起動すると必ず同じ状態で起動するという意味で組み込み機器としては必要な機能なのですが、毎回起動するたびに設定するのも面倒です。そのため、設定ファイル類だけはフラッシュメモリに永続的に保存できるようになっています。

それを行うためのコマンドが、**flatfsd** です。**flatfsd -s** と**-s** オプションを付けて実行することで、/etc/config ディレクトリにあるファイルを、フラッシュメモリに保存します。実は/etc/network/interfaces や/etc/resolv.conf は、/etc/config 以下のファイルへのシンボリックリンクとなっています。そのため、/etc/network/interfaces などに対して行った変更は、**flatfsd -s** コマンドを実行することで、フラッシュメモリに保存されます。保存されたファイルは起動時に自動で復元されるようになっているため、次回起動時は変更したネットワーク設定が行われるようになります。変更を消したくない場合は、**flatfsd -s** コマンドを実行することを忘れないでください。

4.3.2. 時刻を設定する

サンプルアプリケーションで使用するデータの取得時刻には、クライアント側のシステムタイムを利用します。時刻が大幅にずれている場合には混乱を招く場合があるため、時刻を設定します。

ここでは、NTP クライアントを利用してシステムタイムをタイムサーバー[7]と同期させます。**ntpclient** コマンドで時刻を設定し、**date** コマンドで時刻が設定されたことを確認してください。

```
[armadillo ~]# ntpclient -h ntp.nict.jp -s
[armadillo ~]# date
Tue Jun 2 12:34:57 JST 2015
```

図 4.12 ntpclient を利用した時刻の設定

起動時に自動で時刻設定を行う

時刻設定のように起動するたびに毎回行う操作は、起動時に自動で行うように設定できます。Armadillo-IoT では、/etc/config/rc.local が init スクリプトの最後に実行されるようになっていますので、これを編集し **ntpclient** コマンドを実行するように記述することで、起動時に自動で時刻設定されるようになります。

[7]コマンドライン中で使用しているタイムサーバは一例です。

/etc/config/rc.local ファイルに対する変更も、そのままでは再起動すると消えてしまいますので **flatfsd -s** コマンドでフラッシュメモリに保存するのを忘れないでください。

```
[armadillo ~]# echo "ntpclient -h ntp.nict.jp -s" >> /etc/config/rc.local
[armadillo ~]# flatfsd -s
```

図 4.13 起動時に自動で時刻設定を行う

時刻の保持

Armadillo-IoT では、システムタイムをリアルタイムクロック(カレンダー時計)に保持することができます。リアルタイムクロックのバックアップ用電池を取り付けることで、電源を切っても時刻を保持し続けるようになります。この場合、起動するたびに時刻設定をする必要は無くなります。詳しくは、製品マニュアルを参照してください。

4.4. サンプルアプリケーションの実行

クライアントサイドサンプルアプリケーションを実行すると、デフォルトで設定された URI に対してデータをアップロードし、サーバーからの Push 通知に応じてコマンドを実行します。「表 4.1. POST 時の QUERY パラメータ」で示した QUERY パラメータのうち、「latitude」、「longitude」については、東京周辺を指すようにランダムに生成されます。

クライアントサイドのサンプルアプリケーションと設定ファイルは、それぞれ http://armadillo-iot-sample.herokuapp.com/client/sample.rb と http://armadillo-iot-sample.herokuapp.com/client/config.json から取得します。

```
[armadillo ~]# curl -kO https://armadillo-iot-sample.herokuapp.com/client/sample.rb
[armadillo ~]# curl -kO https://armadillo-iot-sample.herokuapp.com/client/config.json
[armadillo ~]# ruby sample.rb --config config.json
I, [2015-06-03T12:29:56.164923 #1108]  INFO -- : uid: 00110cxxxxxx
I, [2015-06-03T12:29:56.167375 #1108]  INFO -- : top page: http://armadillo-iot-sample.herokuapp.com
I, [2015-06-03T12:29:56.168886 #1108]  INFO -- : personal page: http://armadillo-iot-
sample.herokuapp.com/cockpit?uid=00110cxxxxxx
I, [2015-06-03T12:29:56.170631 #1108]  INFO -- : Searching GPS...
I, [2015-06-03T12:30:01.224029 #1108]  INFO -- : 0: 2015-06-03 12:30:01
I, [2015-06-03T12:30:06.736723 #1108]  INFO -- : 1: 2015-06-03 12:30:06
I, [2015-06-03T12:30:12.169770 #1108]  INFO -- : 2: 2015-06-03 12:30:12
I, [2015-06-03T12:30:17.612970 #1108]  INFO -- : 3: 2015-06-03 12:30:17
I, [2015-06-03T12:30:23.085313 #1108]  INFO -- : 4: 2015-06-03 12:30:23
I, [2015-06-03T12:30:28.519343 #1108]  INFO -- : 5: 2015-06-03 12:30:28
I, [2015-06-03T12:30:33.959218 #1108]  INFO -- : 6: 2015-06-03 12:30:33
I, [2015-06-03T12:30:39.420082 #1108]  INFO -- : 7: 2015-06-03 12:30:39
I, [2015-06-03T12:30:44.860059 #1108]  INFO -- : 8: 2015-06-03 12:30:44
I, [2015-06-03T12:30:50.329457 #1108]  INFO -- : 9: 2015-06-03 12:30:50
```

図 4.14 サンプルアプリケーション実行例

デフォルトでは、5 秒間隔で 10 回データ送信を行います。--interval オプションを指定することで送信間隔を、--times オプションを指定することで送信回数を指定できます。(-1 を指定すると、無制限にデータをアップロードし続けます。)

```
[armadillo ~]# ruby sample.rb --config config.json --interval 1 --times 5
```

図 4.15 オプションの指定例(1 秒間隔で 5 回送信)

その他のオプションについては、help を参照してください。

```
[armadillo ~]# ruby sample.rb --help
Usage: iotg_sample [options]
    -p, --config=path                Config file path (default: config.json)
    -d, --debug                      Enable debug message
    -t, --times=TIMES                Number of repeat times, -1 means infinite loop (default: 10)
    -i, --interval=INTERVAL          Interval between sending a message (default: 5[sec])
    -c, --command=COMMAND           Command which execute push notification (default: echo $value)
```

図 4.16 サンプルアプリケーションの help

4.5. Web ブラウザでデータを確認

アップロードしたデータが反映されているか、Web ブラウザで確認してみましょう。確認用 URL は、クライアントサイドサンプルアプリケーション実行時のログに表示されています。

```
[armadillo ~]# ruby sample.rb --config config.json
I, [2015-06-03T12:29:56.164923 #1108]  INFO -- : uid: 00110cxxxxxx
I, [2015-06-03T12:29:56.167375 #1108]  INFO -- : top page: http://armadillo-iot-sample.herokuapp.com
I, [2015-06-03T12:29:56.168886 #1108]  INFO -- : personal page: http://armadillo-iot-
sample.herokuapp.com/cockpit?uid=00110cxxxxxx
```

図 4.17 サンプルアプリケーション実行例(確認用 URL)

「top page」と表示されている行の URL (http://armadillo-iot-sample.herokuapp.com) にアクセスすると、地図アプリケーションが表示されます。Basic 認証が掛かっていますので、ユーザー名「username」、パスワード「password」でログインしてください。

図 4.18 サンプルアプリケーション トップページ

データが存在する場所には、ピンが立っています。ピンをクリックすると、UID が表示されますので、自分の UID と一致していたら「こちら」をクリックし、パーソナルページに移動してください。もし、自分の UID を持つピンが見当たらなかったら、地図を広域表示にして探してください。詳細ページはクライアントサイドサンプルアプリケーションの実行ログの「personal page」と表示されている行の URL から直接アクセスすることもできます。

図 4.19 サンプルアプリケーション ピンの確認

ピンが見つからない

地図上のピンは、最新のデータがある位置を示しています。後から他の人がアップロードしたデータがたまたま同じ位置になってしまった場合、上書きされてしまいます。その場合は、再度クライアントサイドアプリケーションを実行して別の位置にピンを立てるか、パーソナルページに直接アクセスしてください。

パーソナルページには、現在の温度データを表示するメーター(左上)と、Armadillo-IoT への操作指示を行うボタン(右上)、時系列を表示するグラフ領域(下)があります。

図 4.20 サンプルアプリケーション パーソナルページ(上側)

図 4.21 サンプルアプリケーション パーソナルページ(下側)

クライアントから温度データがアップロードされると、メーターの表示がリアルタイムに変化することを確認してください。また、グラフ領域上の日付ボックスでデータを表示したい日を選択すると、その日の温度変化が時系列グラフとして表示されます。

また、ボタンをクリックすると、クライアントに Push 通知を行います。クライアントサイドサンプルアプリケーションのデフォルト設定では、シリアル通信ソフトウェアに"on"または"off"と表示するだけです。--command で Push 通知を受けたときに実行するコマンドを指定できます。value 変数に"on"または"off"が格納されてコマンドが実行されます。ボタンの状態に応じて LED[8]を点灯/消灯する場合の実行例を下記に示します。

```
[armadillo ~]# ruby sample.rb --config config.json --command 'if [ $value = "on" ]; then echo 1 > \
/sys/class/leds/led3/brightness; else echo 0 > /sys/class/leds/led3/brightness; fi'
```

図 4.22 サンプルアプリケーション実行例(LED 制御)

[8]LED についての詳細な情報は、製品マニュアルに記載されています。

5. サンプルアプリケーションのカスタマイズ

前章では、サンプルアプリケーションの動作や使用方法について説明しました。サンプルアプリケーションの動作を確認し、あるいは改造してみることで Armadillo-IoT の開発手順を把握でき、オリジナルのシステムを開発する際に効率的に行えるようになるでしょう。本章では、サンプルアプリケーションのカスタマイズに関して、有用なドキュメントなどを紹介します。

5.1. クライアントサイドアプリケーションのカスタマイズ

5.1.1. 起動時に自動実行する

/etc/config/rc.local に起動時に実行したいコマンドを記述することで、プログラムを自動実行できます。クライアントサイドサンプルアプリケーションを自動実行し、データを送信し続けるには、sample.rb と config.json を/etc/config/ディレクトリにコピーし、/etc/config/rc.local の最後に下記のようにコマンドを追記してください。

```
ntpclient -h ntp.nict.jp -s
cd /etc/config
ruby sample.rb --config config.json --times -1 &
```

図 5.1 起動時に自動実行する

実行し続けるプログラムを起動する場合、コマンドの最後に & をつけて、バックグラウンド実行することを忘れないでください。また、設定を保存するために、再起動前に **flatfsd -s** コマンドを実行する必要があります。

5.1.2. 他のセンサからのデータを扱う

クライアントサイドサンプルアプリケーションでは、Armadillo-IoT に内蔵されている温度センサの値を取得して、アップロードするよう実装されています。温度センサの値は、sysfs という疑似(Pseudo)ファイルシステムを読み出すことで取得できるように、ドライバが実装されています(class LM75 の実装参照)。この他、GPIO や LED の操作も sysfs を経由して行うことができますので、Ruby スクリプトからも容易に扱えるようになっています。

EnOcean、Wi-SUN、Bluetooth Low Energy は、全てシリアルポート[1]を通じて通信することができます。Ruby では、serialport [https://rubygems.org/gems/serialport] gem を使うことで、シリアルポートを扱うことができるようになります。serialport gem は Armadillo-IoT の出荷時イメージに含まれています。

5.1.3. 他の言語で実装する

Armadillo-IoT の出荷用イメージには、Ruby の他に、Java Runtime や Lua が含まれています。クライアントサイドサンプルアプリケーションと同等の機能を持つプログラムを、これらの言語を使って実装することもできるでしょう。

[1]Linux では tty デバイスとして扱います。

データのアップロードは、単純な REST API(HTTP POST)になっていますので、比較的簡単に実装できると思われます。しかし、Push 通知を受け取るには、Pusher のライブラリが必要となります。Java 用 Pusher クライアントにはいくつかの実装があります。http://pusher.com/docs/client_libraries#android-java を参照してください。

5.1.4. ROM 化する

/etc/config ディレクトリは大きな容量がありませんので、設定ファイルや小さなプログラム程度しか保存することはできません。多くのライブラリを追加したい場合や、製品化時の出荷イメージを作る場合には、ROM イメージを作成する必要があります。

ROM イメージの作成には、Atmark Dist という Armadillo 専用のディストリビューションを使います。Atmark Dist を使うと、必要最小限のプログラムや設定ファイルだけを含んだ、オリジナルの ROM イメージを作ることができます。詳細は、製品マニュアルを参照してください。

5.2. サーバーサイドサンプルアプリケーションのカスタマイズ

サーバーサイドサンプルアプリケーションのソースコードは、アットマークテクノ公式 GitHub [https://github.com/atmark-techno]で公開しています。

また、Heroku アカウントを持っている方であれば、「Heroku ボタン」を使って簡単に自分用のコピーを作成し、Heroku 上で実行できます。情報が公開されてしまうことに抵抗がある方や、サーバーサイドサンプルを改造したい方は活用してください。具体的な方法は、Armadillo サイトの Howto [http://armadillo.atmark-techno.com/armadillo-iot/techinfo]に掲載しています。

6. ユーザー登録

アットマークテクノ製品をご利用のユーザーに対して、購入者向けの限定公開データの提供や大切なお知らせをお届けするサービスなど、ユーザー登録すると様々なサービスを受けることができます。サービスを受けるためには、「アットマークテクノ ユーザーズサイト」にユーザー登録をする必要があります。

ユーザー登録すると次のようなサービスを受けることができます。

- 製品仕様や部品などの変更通知の閲覧・配信
- 購入者向けの限定公開データのダウンロード
- 該当製品のバージョンアップに伴う優待販売のお知らせ配信
- 該当製品に関する開発セミナーやイベント等のお知らせ配信

詳しくは、「アットマークテクノ ユーザーズサイト」をご覧ください。

アットマークテクノ ユーザーズサイト

https://users.atmark-techno.com/

6.1. 購入製品登録

ユーザー登録完了後に、購入製品登録することで、「購入者向けの限定公開データ[1]」をダウンロードすることができるようになります。

Armadillo-IoT 購入製品登録

https://users.atmark-techno.com/armadillo-iot/register

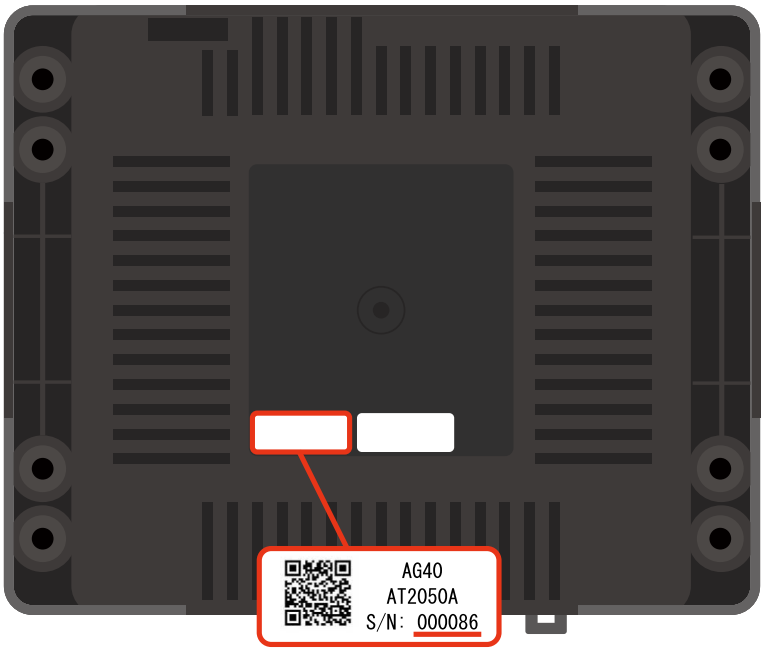
Armadillo-IoT の購入製品登録を行うには、ユーザーズサイトで「シリアル番号」の入力および「正規認証ファイル」のアップロードを行う必要があります

Armadillo-IoT のシリアル番号の確認方法を「6.1.1. シリアル番号を確認する方法」に、Armadillo-IoT から正規認証ファイル(board-info.txt)を取り出す手順を「6.1.2. 正規認証ファイルを取り出す手順」に示します。

6.1.1. シリアル番号を確認する方法

シリアル番号は、ケース貼付シールに記載された 6 桁の数値です。次の例では、シリアル番号が「000086」であることが確認できます。

[1]アドオンモジュールの回路図データなど



シリアル番号を「Armadillo-IoT 購入製品登録」ページの「シリアル番号」欄に入力してください。

6.1.2. 正規認証ファイルを取り出す手順

Armadillo にログインし、コマンドを実行すると正規認証ファイルが生成されます。そのファイルをお使いの Web ブラウザを使ってダウンロードしてください。

1. ATDE で minicom を立ち上げて、Armadillo-IoT に root ユーザーでログインします。デバイスファイル名(/dev/ttyUSB0)は、ご使用の環境により ttyUSB1 や ttyS0、ttyS1 などになる場合があります。Armadillo に接続されているシリアルポートのデバイスファイルを指定してください。

```
atmark@atde5:~$ LANG=C minicom --noinit --wrap --device /dev/ttyUSB0

armadillo-iotg login: root
Password:
[root@armadillo-iotg (ttymxc1) ~]#
```

2. "get-board-info"コマンドを実行して正規認証ファイル(board-info.txt)を作成します。

```
[root@armadillo-iotg (ttymxc1) ~]# get-board-info
[root@armadillo-iotg (ttymxc1) ~]# ls
board-info.txt
[root@armadillo-iotg (ttymxc1) ~]#
```

3. Armadillo 上で動いている WEB サーバーがアクセスできる場所に、正規認証ファイルを移動し、アクセス権限を変更します。

```
[root@armadillo-iotg (ttymxc1) ~]# mv board-info.txt /home/www-data/
[root@armadillo-iotg (ttymxc1) ~]# chmod a+r /home/www-data/board-info.txt
```

4. minicom を終了させ、お使いの Web ブラウザから、Armadillo の URL にアクセスしてください。下記どちらかの指定方法でアクセス可能です。

   - http://armadillo-iotg.local/board-info.txt
   - http://[Armadillo の IP アドレス]/board-info.txt [2]

   取り出した正規認証ファイルを「Armadillo-IoT 購入製品登録」ページの「正規認証ファイル」欄に指定し、アップロードしてください。

[2] Armadillo の IP アドレスが 192.0.2.10 の場合、http://192.0.2.10/board-info.txt となります。

改訂履歴

バージョン	年月日	改訂内容
2.0.0	2014/06/23	・初版発行

Armadillo-IoT ゲートウェイスタンダードモデル 開発セット スタートアップガイド
Version 2.0.0
2015/06/23

株式会社アットマークテクノ

札幌本社
〒 060-0035 札幌市中央区北 5 条東 2 丁目 AFT ビル
TEL 011-207-6550 FAX 011-207-6570

横浜営業所
〒 221-0835 横浜市神奈川区鶴屋町 3 丁目 30-4 明治安田生命横浜西口ビル 7F
TEL 045-548-5651 FAX 050-3737-4597