システムソフトウェア開発プラットフォーム
—MakeStack—

1.　背景

Arduinoやmbedといったⅰ安価で初心者でも簡単にプログラミングができるマイクロコントローラや，3Dプリンタの登場により，今までは個人では難しかったハードウェアの開発ができる時代になった．また，Internet of Thingsの振興によりRaspberry PiやIntel Edison，ESP8266といったインターネットに接続する機能を持った安価なコンピュータが活発に利用されている．

しかし，その中で動くソフトウェアの開発は面倒ごとばかりである．開発したサービスを運用するには，リモートアップデート，死活監視，デバイスの管理，ログ収集といった機能が必要となるが，それらを全て自前で実装することは非常にコストが大きい．

resin.io，Isaax，Particleといった，それらの機能を提供するサービスは存在するが，それらは高いCPU処理性能，多くのメモリ容量，高い電力資源を必要とし，巨大なOSであるLinuxが動くコンピュータを前提としていたり，APIが単純な動作を冗長にコーディングしなければいけない仕様であったりと，使い勝手が悪い．

2.　目的

本プロジェクトでは低性能・低消費電力で安価なマイコンに対応し，アイデアを自然な形で短時間にコーディングすることができるAPIを持つ，低コストで高速なハードウェアプロトタイピングを実現するIoT PaaSの構築を目的とする．

3.　開発の内容

本プロジェクトで開発した「MakeStack」のアーキテクチャを図1に示す．

MakeStackには簡単なデバイスセットアップ，サンプルコード駆動開発，コマンド一発デプロイ，Slack Webhookのような手軽な外部連携の4つの特徴がある．

デバイスのセットアップについては，最初にデバイスをPCに接続してコマンド一つでMakeStackハイパーバイザをインストールするだけである．インストール後はサーバから直接更新を受け取るので，デバイスを再度PCと接続する必要はない．

アプリの開発については，プラグインのコードジェネレータによりサンプルコードに加え初期化コードやC++の宣言文が自動生成され，Ruby on Railsのscaffoldingのようにサンプルコードを元にした少ない編集だけで，アプリを短時間で作成することが可能となる．

デプロイについては，コマンドひとつでサーバサイドでのビルド，各デバイスに対する自動アップデートが可能となる．

アプリのサーバサイドとの連携はEndpoint/Store APIを使って簡単に可能となる．Endpoint APIはデバイスから任意の文字列を送る仕組みで，Endpointが呼ばれると

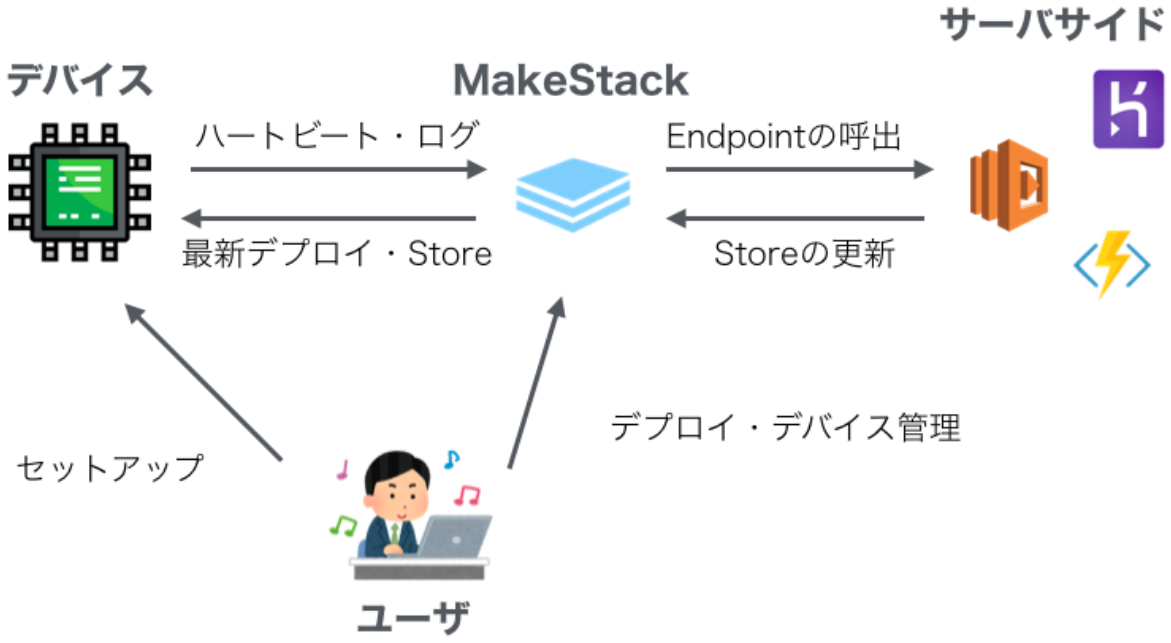

図1: MakeStackのアーキテクチャ

デバイスからのデータがアプリに登録された固有のURLに送られる．デバイスはStore APIを利用してサーバからデータを取得することができる．Storeのデータを変更する処理は，MakeStack ServerのWebhookにJSON形式のデータを送るだけである．このシンプルな仕組みによって，AWS Lambdaといったサーバレスアーキテクチャとの連携が簡単に可能となる．

デバイス上で動作するソフトウェアの構成を図2に示す．

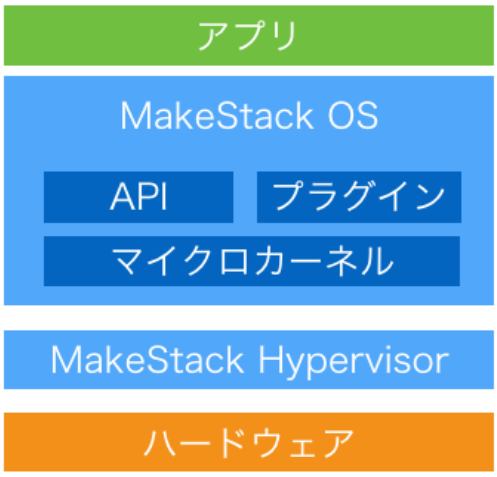

図2: デバイスサイドのソフトウェアスタック

MakeStack HypervisorはデバイスのMakeStack OSが動作するための環境とウォッチドッグタイマを提供する．つまりPCのBIOSやUEFIと同じ立ち位置である．本プロジェクトではESP8266に対応した．

MakeStack OSはMakeStackのAPI群の実装，プラグイン機構，独自の軽量マイクロカーネルReseaをベースにしたオペレーティングシステムである．

4. 従来の技術（または機能）との相違

既存のIoT PaaSは，resin.io，Isaax，C3 IoTなどLinuxが動くことを前提としたものが多い．しかし，今日においてはArduinoのAVRマイコンレベルの制御に加えてサーバサイドとの通信ができれば十分であり，潤沢なハードウェアリソースを用意することは，金銭的なコストと消費電力において無駄が大きい．

本プロジェクトでは，数千円するLinuxボードの代わりに500円程度で購入できるESP8266という低性能・低消費電力マイコンで十分に利用可能なIoT PaaSを実現した．

また，resin.ioのようなIoT PaaSは提供されているAPIがデプロイまでであり，GPIOやI2Cといった制御を行うためのAPIは提供されておらず，アプリ開発においてどういうプログラミング言語・ライブラリをどう使うかについてはユーザ次第となっている．しかし，Arduinoスケッチが一つのファイルで完結できていることから分かるように，アプリ開発においてそれほど複雑な処理を書く必要はあまりないため，好きなプログラミング言語を使えるというのはメリットとしては薄い．その点，ParticleやArduino CREATEはAPIレイヤを提供しているが，単純な処理の実装に対して冗長なコーディングを要求するAPIとなっている．対してMakeStack APIではイベントドリブンでアイデアを自然に記述することを可能にした．

加えて，MakeStackはコーディング量をより少なくするために，コードジェネレータ機構を導入した．サンプルコードに加え初期化コードやC++の宣言文を自動生成することで，Ruby on Railsのscaffoldingのように，サンプルコードを元にした少ない編集だけで，アプリを短時間で作成することを可能にした．

5. 期待される効果

低価格・低消費電力のマイコンに対応しているため，多量のデバイスを乾電池駆動で十分使える「動くもの」を開発できるようになった．

また，コードジェネレータや直感的な開発者ツールにより，ハードウェア開発経験者だけでなくソフトウェアエンジニアもWebアプリを作るように気軽に「動くもの」を作り共有するコミュニティとなることを期待する．

6. 普及（または活用）の見通し

MakeStackはオープンソースソフトウェアとして開発されている．今後，ドキュメントやチュートリアルの整備を行った上で，ESP8266のユーザフォーラムといったコミュニティを通して広めていく．

7. クリエータ名（所属）

怒田 晟也（筑波大学 情報学群 情報科学類）
左野 寛之（会津大学）

（参考）関連URL

Webサイト: http://makestack.org
ソースコード: https://github.com/makestack