特集論文

IoTプラットフォームのアーキテクチャ

田村孝之* 渡邉和樹**
塚本良太**
原田篤史***

Architecture of an IoT Platform

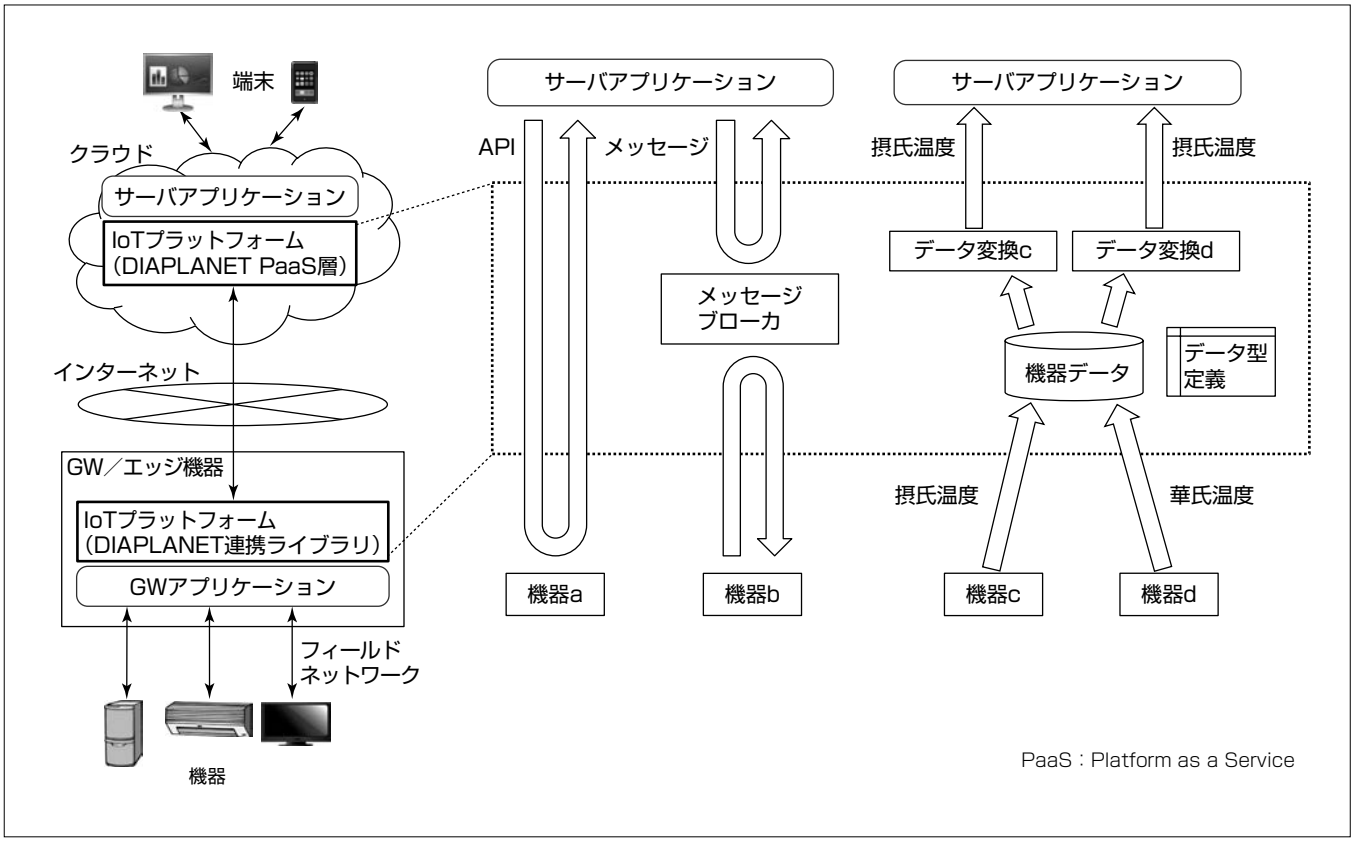
Takayuki Tamura, Ryota Tsukamoto, Atsushi Harada, Kazuki Watanabe

要 旨

様々なモノをネットワークに接続し，遠隔監視や遠隔制御等を実現するIoT(Internet of Things)が注目を集めている。三菱電機では，様々な機器に対するIoTアプリケーションの共通プラットフォームとして，スマート制御クラウドサービス“DIAPLANET”を提供している。

機器，ゲートウェイ(GW)／エッジ機器，クラウド，端末等からなるIoTシステムの構築を容易にするため，通信方式やデータ形式等の標準化が進められているが，多数の標準が併存しているのが実状である。そのため，IoTシステムでは，多様な通信方式を用いて機器とクラウドを接続し，多様な形式の機器データを管理することが求められる。また，機器データの分析手法は，実データから得られる知見に基づいて継続的に改良されるため，システム運用開始後もデータ構造の変更等に柔軟に対応できることが望ましい。

DIAPLANETでは，クラウドと機器の通信における通信プロトコルと通信のパターンを隠蔽・変換するAPI(Application Programming Interface)を提供し，サーバアプリケーションの可搬性を高めた。また，機器データの管理では，機器から受け取った生データを標準形式のデータベースに格納しておき，サーバアプリケーションがデータを読み出す際に変換を施すことで，アプリケーションごとのデータベース設計を不要にした。これによって，データ収集開始までのリードタイム短縮と，アプリケーション仕様の継続的な変更を可能にした。今後は，クラウド及びエッジ機器でのデータ分析を支援する機能の充実化を図る。



IoTプラットフォームによる機器の多様性への対応

IoTプラットフォームは，クラウド上のサーバアプリケーションを様々な機器と連携可能にする。機器へのメッセージ送受信では，サーバ型の機器，クライアント型の機器に共通のAPIを提供する。機器データは標準形式のデータベースに格納し，サーバアプリケーションがデータをアクセスする際にデータ形式や解釈方法(例：温度の単位系)の変換を行う。

*情報技術総合研究所(博士(工学)) **同研究所 ***同研究所(情報学博士)

1. まえがき

様々なモノをネットワークに接続し，遠隔監視や遠隔制御等を実現するIoTシステムでは，多様な機器を接続し，多様な機器データを扱うことが求められる。当社ではIoTシステムの開発・運用効率向上のため，機器接続と機器データ管理での多様性を吸収するIoTプラットフォームの開発を行っており，その成果を当社スマート制御クラウドサービスDIAPLANET[1]に適用してきた。

本稿では，IoTシステムにおける多様性と，それに対処するためのIoTプラットフォームの機能について述べる。

2. IoTシステムの多様性

この章ではIoT標準の状況を概観し，IoTアプリケーションの開発では異なる標準の考慮が必要であることを述べる。特にIoTアプリケーションへの影響が大きい差異として，通信におけるメッセージ交換パターン，機器データの形式と解釈，機器の制御命令と制御結果について取り上げる。

2.1 IoT標準の状況

機器，ゲートウェイ(GW)，クラウド，端末等からなるIoTシステムの構築を容易にし，IoTアプリケーションの再利用性を高めるために，様々な標準が提案されている。**図1**は，IoT関連の主な標準やアライアンスのポジショニングを，適用レイヤ(垂直方向)及び適用フェーズ(水平方向)にマッピングしたものである。これらは，ネットワークレイヤの相互接続性に特化したものからアプリケーションレイヤにおけるエコシステム形成を意図したものまで多岐にわたっており，**図1**の全領域をカバーする単一の標準に収束する見込みは低いと考えられる。

例えばネットワークレイヤでは，機器／GWとクラウド間の通信に用いられるプロトコルとしてHTTP，CoAP，MQTTなどがあり，複数プロトコルの混在を許してシステム構成の自由度を高めることが望ましい。そこで，標準に対応しつつ個々の標準の差異を隠蔽するIoTプラットフォームによって，複数の標準に対応したIoTアプリケーションの開発を効率化する必要がある。

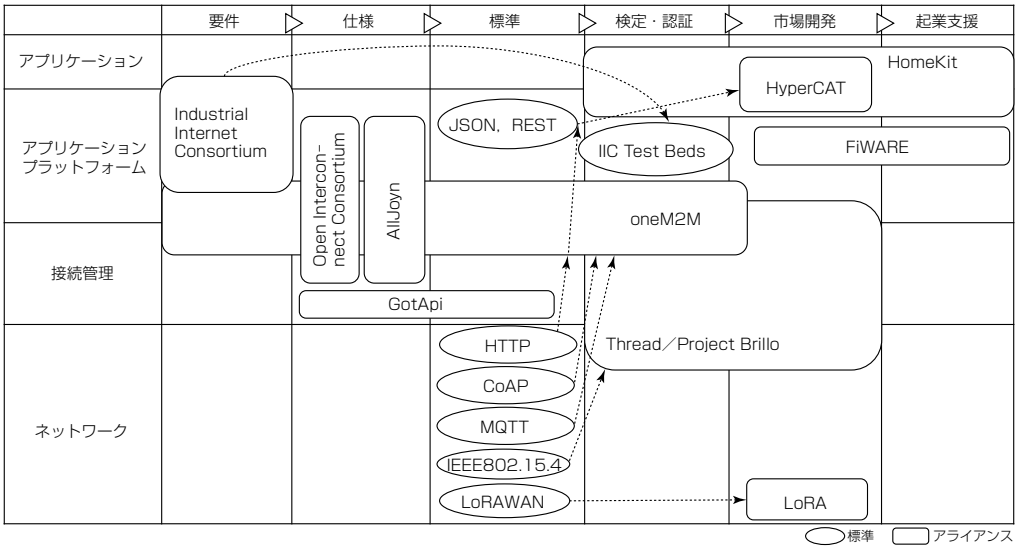


図1．IoT関連標準とアライアンスのポジショニング[2]

2.2 メッセージ交換パターン

IoT標準ごとに大きく異なるものの1つに，機器とアプリケーション間のメッセージ交換パターンがある。**図2**に機器からのデータ収集における主なメッセージ交換パターンを示す。

図2(a)は機器がサーバ機能を持ち，アプリケーションの要求を受け付けて応答するものである。アプリケーションは任意のタイミングで機器に最新の状態を問い合わせ，応答に基づいて画面更新等を行う。工場や設備内のローカルネットワーク上でパソコン等を用いた管理や見える化が可能な機器やアプリケーションはこのパターンに対応している。機器とアプリケーションをインターネット経由で接続する場合は，機器のサーバ機能を第三者に公開することを防ぐため，VPNルータを用いることが多い。しかし，アプリケーション側から工場・設備内ネットワーク上の機器が全てアクセス可能となってしまうため，セキュリティ確保のためには精緻なアクセス制御が必要である。

図2(b)は機器がクライアントとなって自律的に状態を通知するものであり，MQTT等のプロトコルが適用される。アプリケーションは機器が通知する状態データを直接受信するのではなく，一旦データベースに格納されたデータを用いて見える化等を行うことが多い。機器が設置される環境はNATルータを介してインターネット接続されることが多いが，機器をサーバとして公開する必要のないこのパターンはこのような環境と親和性が高い。

図2(c)は両者の中間であり，サーバ機能を持つ機器がアプリケーションからの要求をトリガーとし，指定された周期で持続的に状態データを応答するものである。

異なるメッセージ交換パターンの間でアプリケーションを再利用するには，メッセージ交換パターンを変換する機能の追加が必要になる。例えば，通知型のアプリケーションを要求・応答型の機器と組み合わせるには，応答データをデータベースに格納する別アプリケーションを追加する。一方，要求・応答型のアプリケーションを通知型の機器と組み合わせるに

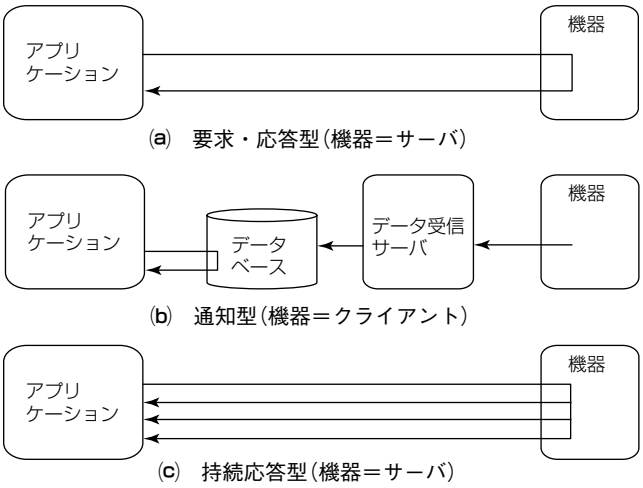


図2．機器データ収集におけるメッセージ交換パターン

は，機器への要求をデータベースへの問合せに変換する処理が必要になる。IoTプラットフォームによってこれらの変換を行うことで，アプリケーションの再利用性を高めることができる。

2.3 データ形式と解釈

標準や機器ごとにデータの表現方法にも差異が生じる。これにはXML(eXtensible Markup Language)やバイナリーなどの，エンコーディング形式の違いも含まれるが，より重大な差異は，値の解釈に関するものである。

例えば温度では摂氏及び華氏のどちらの単位系で表されるかによって同じ物理量に対応する数値が異なってくる。また，電力量では，ある機器は30分ごとの積算値を出力し，別の機器は計測開始以降の積算値を出力するというように，値の定義に含まれるパラメータ(この場合は積算対象の時区間)が異なる場合がある。さらに，センサの生データのように，読み取った値から元の物理量を求めるのに補正処理が必要になる場合もある。

値の解釈に必要となる単位系や補正パラメータ等の情報は，機器・センサの仕様や設置条件としてオフラインで与えられることが多い。したがってIoTアプリケーションがこれらの値を解釈するロジックを内蔵してしまうと，仕様や設置条件が異なる同種の機器への適用が困難になる。そこで，データの値にその解釈方法を示す情報を明示的に付与し，IoTプラットフォームに値の解釈を行うロジックをライブラリとして持たせることで，IoTアプリケーションの汎用性を高めることができる。

2.4 制御命令と制御結果

アプリケーションから機器に命令を送付して制御を行う場合，機器ごとに命令の内容が異なることに加え，制御結果の応答内容が持つ意味にも差異が生じる。すなわち，応答内容が制御命令実行後の機器の状態を反映する場合と，単に制御命令の受領を示すに過ぎない場合である。後者の場合，制御命令がいつ有効になったかを確認するために，

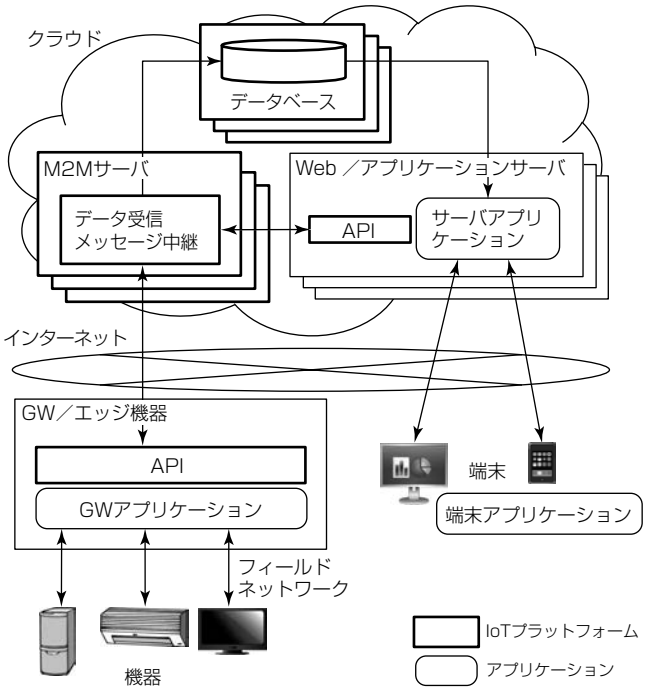


図3．IoTシステムの全体構成例

機器の状態を取得する命令を別途発行する必要がある。

状態問合せの結果についても，ハードウェアの現在の状態を返す機器と，キャッシュされた過去の状態を返す機器が存在する。レガシー機器にハードウェアを外付けしてスマート化した機器では，操作パネルやスイッチ等で本体を直接制御すると外付けハードウェア側ではその内容を検知できない。このような外付けハードウェアが状態をキャッシュする方式を採用していると，機器本体の状態との一貫性を維持できなくなる。

IoTプラットフォームは，制御命令に単なる受領確認で応答する機器に対し，状態が変化するまで問合せを行うことで，各アプリケーションの処理を単純化することができる。

3. IoTプラットフォームの構成

この章では**2章**で示したIoTシステムの多様性に対応するためのIoTプラットフォームの構成について述べる。はじめにIoTシステムの全体構成におけるIoTプラットフォームの位置付けを示し，機器とのメッセージ交換パターンの差異を吸収する方式，及び機器データの解釈の差異に対応するデータベース設計について述べる。

3.1 全体構成

図3にIoTシステムの全体構成例を示す[1]。IoTシステムは機器，GW／エッジ機器，クラウド，及び端末からなり，IoTプラットフォームはクラウド上のサーバアプリケーション及びGW／エッジ機器上のGWアプリケーションに対してAPIを提供する。GW／エッジ機器の機能を機器に組み込み，機器が直接クラウドと通信する構成も取り得る。

M2M(Machine to Machine)サーバは，サーバアプリケーションが機器データを利用するための基本機能として，クラウド上でGW／エッジ機器からデータを受信し，データベースに格納する機能を提供する(**図2**(b)のデータ受信サーバ)。また，M2Mサーバは，サーバアプリケーションから機器に対して要求を送信するための基本機能として，クライアント型のGW／エッジ機器とWeb／アプリケーションサーバの双方向通信を可能にするメッセージ中継(ブローカ)機能を提供する。

データベースはKVS(Key-Value Store)であり，アプリケーションの要件に応じて関係データベース(Relational DataBase：RDB)を併用する。データベースでは，任意の機器データをデータ型名とともに記録することで，データ変換等の処理の詳細を柔軟に変更可能とする。また，KVSはキー値に基づく分散管理が容易であり，サーバを追加してスケールアウトすることでデータ量の増大に対応することができる。

接続するGW／エッジ機器数や端末数の増加に対しては，M2MサーバやWeb／アプリケーションサーバの追加によるスケールアウトで対応する。IoTプラットフォームのクラウド側構成要素は仮想サーバイメージで提供され，クラウド管理機能によってインスタンス数を制御することで最適なシステム構成を維持する。スケールアウト構成とすることで，GW／エッジ機器と接続するM2Mサーバと，端末と接続するWeb／アプリケーションサーバがアクセスの都度異なる可能性が生じるが，両者を中継するメッセージ中継機能を通信相手のアドレスを直接指定しないPub／Sub通信に基づいて実現することでこれに対応している。

端末とWeb／アプリケーションサーバの接続部分，及びGW／エッジ機器とM2Mサーバの接続部分では，それぞれユーザー認証及び機器認証を行う。GW／エッジ機器に対しては，クラウドへの初期登録機能，クラウドからの証明書配布・証明書更新機能が必要になる。

次にIoTプラットフォームの主要な構成要素について述べる。

3.2 メッセージ通信API

3.1節で述べたIoTプラットフォームでは，多様な機器に対するアプリケーションの独立性を高めるため，異なる通信プロトコルやメッセージ交換パターンに対し，共通のAPIを提供する。**図4**にメッセージ通信APIの外部インタフェースの1つと内部動作例を示す。

サーバアプリケーションがGW／エッジ機器にメッセージを送信する際は，送信先のGW／エッジ機器ID(“devA”)とメッセージ内容(“msg”)を指定してAPIを呼び出す(Request-Response)。API実装の内部ではMQTT等のPub／Sub型通信を用い，M2Mサーバのメッセージ中継機能であるPub／Subブローカに対してこれらの情報をPublish(出版)する。この際，応答メッセージを受信するための論理的なアドレスであるトピックを動的に生成し(“resTopic”)，メッセージ内容に付与する。

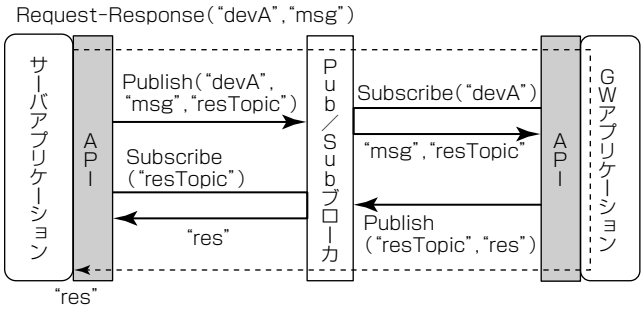


図4．メッセージ通信APIと内部動作

GWアプリケーションは自身のGW／エッジ機器IDをトピックとしてAPI経由でPub／SubブローカにSubscribe(購読)要求を発行しておき，このIDに対してメッセージが発行されると瞬時にその内容と応答用トピックを受け取る。GWアプリケーションは受信したメッセージに従って機器の制御や状態取得を行い，応答メッセージ(“res”)を生成すると，APIを介して応答トピックに対して応答メッセージをPublishする。

サーバアプリケーションはメッセージのPublishと並行して，応答トピックに対するSubscribeを行い，GW／エッジ機器から応答メッセージがPublishされると瞬時にそれを受信する。

このように，APIはGW／エッジ機器側がクライアントとなる通信プロトコルを用いて，要求・応答型のメッセージ交換パターンを提供しており，GW／エッジ機器側をサーバと想定して実装されたアプリケーションの移植を容易にする。他の通信プロトコルを用いる場合でも，実装詳細は隠蔽可能であり，APIを共通に保つことができる。

GW／エッジ機器の台数増加に対しては，Pub／Subブローカを複数用意し，負荷分散を行うことで対応する。同時に，Pub／Subブローカの障害への耐性を得るため，複数ブローカへの同報を行う。こうしたクラスタリング機能をPub／Subブローカ自体が備えていることもあるが，API実装内部で実現することでその他のブローカも利用可能としている。これらによって，共通APIを維持したままスケールアウトを実現できる。

アプリケーションの移植性を高めるだけではなく，アプリケーション自体の再利用が求められる場合もある。このような場合は，サーバやGW／エッジ機器で通信エージェントソフトウェアを動作させ，通信プロトコル及びメッセージ交換パターンの変換を行う。**図5**はGW／エッジ機器上の通信エージェントによって，Pub／Sub型の通信プロトコルで受信したサーバアプリケーションからのメッセージを，HTTP等の要求・応答型の通信に変換して既存のGWアプリケーションに受け渡す例である。

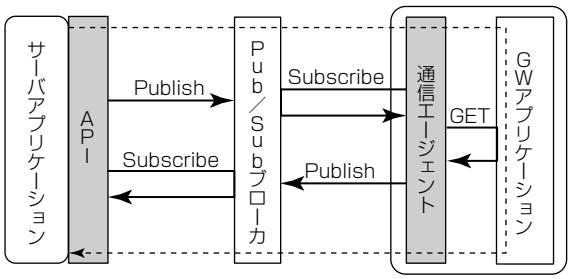


図5．通信エージェントによる通信方式の変換

同様に，クラウド側に通信エージェントを置き，要求・応答型の通信をPub／Sub型に変換することもできる。HTTPを用いるサーバアプリケーションに対しては，通信エージェントはプロキシサーバとして振る舞う。

3.3 データベース

M2Mサーバのデータ受信機能がデータを格納するデータベースはKVSをデフォルトとし，任意形式のデータをタイムスタンプ，センサ等データソースのID，データ型名とともに記録する構造を標準としている。RDBに基づくシステム開発のようにアプリケーションごとに最適なスキーマを事前に定義する開発手順では，機器データ収集開始までのリードタイムが大きくなってしまう。また，収集したデータから得られた知見に基づくアプリケーション改良に伴ってスキーマを変更する際は，蓄積したデータのマイグレーション処理が必要となる。データの形式と解釈に必要な情報をデータ型名として記録しておくことで，任意のデータを単一の構造で扱うことが可能となり，事前のスキーマ定義が不要となる。また，アプリケーションの要件が変わった際には，データ変換等の処理の詳細を柔軟に変更することができる。

自己記述的なデータによってデータの構造と解釈を疎結合にするアプローチは，“Late-bound schema”“Schema-on-Read”などと称され，IoTシステムにおけるビッグデータ管理に適した手法とみなされている[3]。また，こうしたアプローチに基づいて多種多様なデータを生のまま集積したリポジトリは，企業内データにおけるデータウェアハウスと比してデータレイクと呼ばれる[4]。

一方，同一データに繰り返しアクセスし，対話的な応答が求められる可視化アプリケーション等に対しては，データの解釈を読み出し時に決定するこのアプローチでは十分な速度性能が得られない場合がある。そこで，KVSとともにRDBを併用し，定型的なデータ変換の結果をRDBに格納してアプリケーションからアクセスするようにする。RDBへのデータ格納は，KVSからETL(Extract-Transform-Load)ミドルウェアを用いる方法と，M2Mサーバのデータ受信機能のカスタマイズによってKVSへの格納と並行して行う方法を選択できる。ETLを用いる方法は，データ受信に影響を与えずにRDBのスキーマを変更できるため，より一般的である。

データにデータ型名を付随させるアプローチでは，データ型名とデータ処理方法を1対1に対応付ける必要がある。データ型名の衝突や別名の発生を避けるため，命名規則の整備が課題である。ベースとしては，Webやメール等のインターネットアプリケーションで様々なコンテンツ形式の表現に用いられるMIMEタイプが利用できるであろう。データ型名と対応付けてデータ変換のロジックを集積し，ライブラリ化していく。また，機器のファームウェアを更新した際は，機器データの仕様が変更され得る。このような場合，データ型名も合わせて変更することが望ましいが，GW／エッジ機器側で対応できない場合もある。そのため，データ型名自体のバージョン管理を行い，データのタイムスタンプと組み合わせてデータ変換ロジックの選択を行うようにする。機器側の構成管理との連携が重要になると考えられる。

4. むすび

IoTシステムの多様性に対応し，IoTアプリケーションの開発を効率化するためのIoTプラットフォームの機能について述べた。クラウドと機器の間の通信に対しては，使用プロトコルとともにメッセージ交換のパターンを隠蔽・変換するAPIを提供することでアプリケーションの可搬性を高めた。また，機器から受け取ったデータを共通データ構造に一旦格納し，読み出す際にアプリケーションの要件に従った変換を施すことで，多様な機器データの柔軟な管理を可能にした。

今後，IoTプラットフォームでは，データ分析を支援する機能の充実化を進める。特に，クラウドとエッジ機器の連携による効率的かつセキュアなデータ分析の実現について検討していく。

参考文献

(1) 伊藤正裕，ほか：三菱電機スマート制御クラウドサービス“DIAPLANET”，三菱電機技報，**89**，No.8，430～433 (2015)

(2) Fiqueredo，K.：IoT alliances and interoperability (2015)
http://www.more-with-mobile.com/2015/06/iot-alliances-and-interoperability.html

(3) Abadi，D.，et al.：The Beckman Report on Database Research，CACM，**59**，No.2，92～99 (2016)

(4) Dixson，J.：Pentaho，Hadoop，and Data Lakes (2010)
https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/