社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8 (36)
2008 / 3 / 28

多様なセンサを考慮した組込機器用ミドルウェアの開発

川原 貴裕† 松浦 知史† 洞井 晋一† 藤川 和利† 砂原 秀樹†

†奈良先端科学技術大学院大学情報科学研究科 〒631-0192 奈良県生駒市高山町 8916-5

E-mail: †takahiro-k@is.naist.jp

あらまし 近年，大規模センサデータ共有基盤が構築され運用されている．このような情報基盤では，目的の異なる個人や団体が設置するセンサのデータを収集することで，高密度データの管理・運用が可能となる．このときユーザや開発者がデータ収集・利用を容易に行うためには，共有するデータの形式が統一されている必要がある．そのため，センサゲートウェイには各センサの通信手順やデータ形式といった仕様の差異を隠蔽するためミドルウェアが実装される．ここで現状の問題点として高いミドルウェア開発コストおよび更新コストがある．これらミドルウェアの問題はデータの共有を望む設置者や運用者にとって負担となり，共有するデータ量の増加を妨げる要因となっている．

本研究では XML で記述したプラグインを読み込むことにより，多様なセンサに対応するミドルウェアを開発した．本ミドルウェアはセンサ毎に異なる仕様を抽象化し，任意の形式のデータを出力する．本ミドルウェアを小型 CPU ボード Armadillo-210 に実装し，有効性を検証した．その結果作業工程から特に高コストなソースコードの記述を削減出来ることを確認した．これにより，大規模センサデータ共有基盤において，データの収集量の増加を妨げていた要因である高いミドルウェア開発コストおよび更新コストを大幅に軽減した．

キーワード 大規模センサネットワーク，組込み機器，ミドルウェア，XML

A Development of Middleware for Various kinds of Embedded Sensors

Takahiro KAWAHARA† Satoshi MATSUURA† Shinichi DOI†

Kazutoshi FUJIKAWA† and Hideki SUNAHARA†

†Nara Institute of Science and Technology 8916-5 Takayama, Ikoma, Nara, 631-0192, Japan

E-mail: †takahiro-k@is.naist.jp

Abstract In these days, large-scale information infrastructures sharing sensing data are constructed and operated. These infrastructures manage high dense sensing data installed by various individuals and organizations. Sensing data need to be standardized in order to use these data easily. For this reason, middleware having an ability to abstract the specification (e.g. communication procedure, message form) of sensor should be implemented as the sensor gateways. However, the cost of implementing and updating the middleware is high. These high costs prevent the increase of the amount of the shared data. In this paper, we propose the new middleware. It abstracts specifications of various kinds of sensors by referring the XML plug-in. By using our middleware, the cost of implementing and updating the middleware is significantly reduced.

Keyword Large Scale Sensor Network, Embedded Sensors, Middleware, XML

1. はじめに

近年，センシングデバイス（以下センサ）の小型化，低価格化が進みセンサが偏在するようになってきている．これに伴い大規模なセンサデータ共有基盤が構築され，高密度なセンサデータの利用が期待される．例えば，Live E!プロジェクト[1]ではインターネットを通じてセンサの取得した情報を広く共有可能な基盤の構築している．これを用いてセンサから提供される気温や湿度，二酸化炭素濃度などの「環境情報」を共有し，公共サービス・教育材料・ビジネスへの応用を目指している．

このような情報基盤(図 1 参照)では，各個人や団体がセンサを予算や用途に応じて設置し，管理する．情報基盤はそれらセンサからデータを収集することで，高密度データの共有を目指している．このときユーザや開発者が容易にデータを収集・利用するためには，共有するデータの形式が統一されている必要がある．しかし，各センサは通信手順やデータの形式などの仕

様が製造業者に依存するため異なる．そこで，センサゲートウェイには以下の三つの機能を有するミドルウェアを実装する．

(1) センサと通信し，出力メッセージを受信する機能
(2) 受信したメッセージ内からデータを抽出する機能
(3) 抽出したデータを任意の形式へ変換する機能

ミドルウェアが実装されるセンサゲートウェイは組み込み機器が用いられる．センサゲートウェイの要件として小型，安価であるのに加え，センサやインターネットと接続するI/Oポートが求められるためである．

ここで，ミドルウェアの問題点として高コストなミドルウェアの開発が挙げられる．高コストである要因は，一つ目にミドルウェアは仕様の異なるセンサ毎に開発しなければならない点にある．二つ目にミドルウェアのセンサゲートウェイへの実装は専用の開発環境が必須であり，開発者にはハードウェアの知識が求められる点にある．また，別の問題点として高コストなミドルウェアの実装がある．ミドルウェアの機能や設定の実装は，センサの設置者が対処する方法と，運用者がネットワーク経由で対処する方法の二つが考えられる．しかし，センサの設置者は実装に求められる専門知識を有しているとは限らず，設置者にとってその作業は困難である．同様に，ネットワーク経由での実装は，悪意のある者によるミドルウェアの書き換えを可能とするため，セキュリティの点で問題がある．そのためミドルウェアの更新は設置者と運用者が連携し対処せねばならず，非効率となっている．これらミドルウェアの問題はデータの共有を望む設置者に運用者とって負担となるため，共有するデータ量の増加を妨げる要因となっている．

本研究では，大規模センサデータ共有基盤におけるデータの収集部における問題点である高いミドルェア開発コストおよび更新コストの軽減を目指す．具体的にはプラグインを読み込むことで，多様なセンサに対応可能なミドルェアを開発する．開発するミドルェアはXMLで記述されたプラグインを読み込むことにより，センサ毎に異なる多様な通信手順やデータ形式を抽象化する．

本論文の2章で開発するミドルウェアの構成や機能について詳細を述べる．3章では開発したミドルウェアの実装に関して記述する．4章では開発したミドルウェアの有効性を検証する．5章では今後の課題示す．最後に6章では本論文のまとめを述べる．

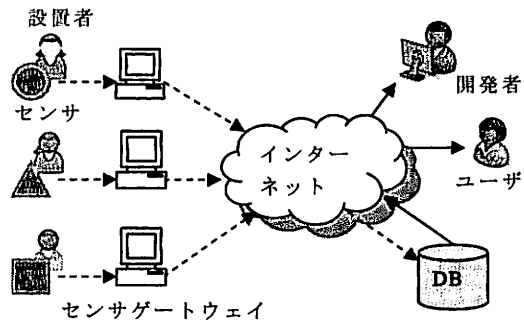

図1　センサデータ共有基盤システム構成

2. 多様なセンサに対応可能なミドルウェアの開発

ミドルウェアはC言語で開発する．ソフトウェアの移植性および組み込み機器の限られたハードウェア資源を考慮したためである．開発するミドルウェアの構成を図2に示す．ミドルウェアは図2のようにセンサに対して専用のプラグインを読み込むことにより以下の三点を抽象化する．

(1) 通信手順
(2) メッセージ形式
(3) データ形式

プラグインには各センサの通信手順やデータ形式等の抽象化すべき項目を記述する．これにより，センサ毎に専用のミドルウェアを開発する手間を省き，開発コストの軽減を図る．また，センサゲートウェイの機能や設定の更新は，読み込むプラグインを変更するのみで可能であり，更新コストの軽減が期待出来る．

本章の2.1節では多様なセンサの仕様を調査し，それにより得た知見を元に抽象化すべきセンサの仕様を明確にする．2.2節ではプラグインの形式，データ構造を定義する．2.3節ではゲートウェイに実装するミドルウェアについて，処理の内容や流れを説明する．

2.1. ミドルウェアの要件

開発するミドルウェアにおいて，抽象化すべきセンサの仕様の差異を明確にするため，多様なセンサの仕様を調査した．代表的な三つのセンサ[2][3][4]について調査結果を表1に示す．なお，調査したセンサの対象はI/Oインタフェースを通じて計算機と接続可能であるものに限っている．

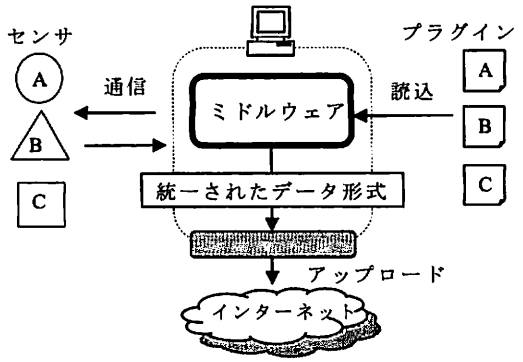

図 2　ミドルウェアの構成

表 1　センサの仕様

センサー名	VantagePro2	WXT510	WM918
I/Oインタフェース	RS232C USB	RS232C RS845など	RS232C
通信手順	対話式	対話式	読み取り式
送信コマンド形式	ASCII	ASCIIなど	-
受信メッセージ形式	16進数	ASCIIなど	2進化10進数
受信メッセージサイズ	固定長	可変長	固定長
センサの制御	可	可	不可
データ項目	温度・湿度 降雨量 気圧 風速・風向 …	温度、湿度 降雨量、降ひょう量 気圧 風速、風向 …	温度、湿度 降雨量 気圧 風速、風向
出力データ(例:温度)	華氏×10	華氏・摂氏	華氏

2.1.1. I/O インタフェース

I/O インタフェースの種類は表 1 に示す RS232C や USB を始め，それ以外にも Ethernet 等の有線通信，赤外線や電波を利用した無線通信がある．各センサの I/O インタフェースの差異を全て抽象化するためには，センサゲートウェイ側に専用の I/O インタフェースやコントローラ IC が搭載されていなければならず，小型，低価格といったセンサゲートウェイの要件を満たせない．しかし，代表的な三つのセンサは全て RS232C での通信が可能である．また，その他のセンサについても仕様を調査した結果，ほぼ全てのセンサは RS232C 通信が可能であった．この理由は，第一に RS232C はシリアル通信方式としては最も普及しているためだと考える．第二にシリアル通信の転送レートは低いが少数の信号線で通信可能で製造コストが低減出来るという特徴が，測定値など少量のデータ転送が出来ればよいというセンサの要件に合致したためだと考える．

本研究では前述した考察から，RS232C による通信が可能なセンサに対象を絞り，仕様の抽象化を行う．

2.1.2. 通信手順，メッセージ形式

通信手順には表 1 に示すように，センサゲートウェイからデータ要求コマンドを送信し，その応答メッセージを受信する対話式と，センサが出力するメッセージをセンサゲートウェイ側では受信のみ行う読み取り式の二つがある．対話式では送信するコマンドやその形式を指定する必要がある．また，双方共にメッセージの受信には，その形式やサイズを指定する必要がある．

続いてセンサとゲートウェイ間でやり取りされるメッセージの形式およびメッセージの受信方法について説明する．送信するコマンドおよび受信するメッセージの形式は，表 1 に示す ASCII 等のテキストフォーマットや 2 進化 10 進数，16 進数といったバイナリフォーマットがある．但し，同一のセンサであっても表 1 に示す VantagePro2 のように送信コマンドと受信メッセージの形式が異なる場合があり，個別の設定が必要である．センサの出力メッセージの受信は，メッセージの始端や終端，サイズを指定することで可能である．固定長メッセージの場合，そのサイズを指定することで受信可能であり，可変長メッセージの場合，メッセージの終端を指定することによって受信可能である

ここで，降雨量を含むメッセージの受信について具体例を挙げながら説明する．対話式のセンサ WXT510 の場合，メッセージ要求コマンドとして ASCII 文字列"0R3"を送信し，それに対する可変長の応答メッセージは同じく ASCII 形式で終端"¥r¥n"を指定する事で受信可能である．一方読み取り式センサである WM918 の場合送信コマンドは必要なく，始端 0xAF とメッセージのサイズを指定することで固定長メッセージを受信可能である．

2.1.3. データ項目，単位

データ項目は表 1 に示す温度，湿度などをはじめそれ以外にも騒音量や加速度など様々なものがある．センサから任意のデータ項目を取得するためには，それぞれがどのメッセージ内に格納されているか，指定し抽出する必要がある．また，センサから取得出来るデータの単位は，センサによって様々である．例えば温度センサの場合，華氏や摂氏の他に，表 1 の VantagePro2 のように華氏×10 の値を出力するものがある．ミドルウェアでは，それら異なるデータ形式を変換し，統一しなければならない．統一する形式はセンサネットワーク運用ポリシよって異なるため任意の形式に変換可能であることが重要である．

2.2. プラグインの定義

前節で考察した要件を満たすようプラグインを定義する．プラグインは XML を用いて記述する．XML を用いることでデータ構造を明確に定義出来る．図 3 にプラグインのデータ構造を示す．プラグインは一つの <SerialPort> 要素と複数の <Unit> 要素から成る．

<SerialPort>内では，ボーレートやストップビット等，シリアル通信のパラメータを指定する．

```
<Plugin sensor="WXT510">
  <SerialPort name="/dev/ttyAM0">
        ……
  </SerialPort>
  <Unit name="get temperature,humidity,pressure">
        <!--図 4 参照-->
  </Unit>
  <Unit name="get rainfall">
        ……
  </Unit>
  <Unit name="get wind_speed,wind_direction">
        ……
  </Unit>
</Plugin>
```

図 3　プラグインのデータ構造

<Unit>要素では，通信手順や受信したメッセージの処理方法等を設定する．詳細を図 4 に示す．

<Unit> 要素は一つの <Interval> 要素および <Interface> 要素と複数の <Parser> 要素から成る．<Interval>要素では，センサとの通信間隔を設定する．値の設定により，数秒ー数十秒の短い間隔を置いて必要な測定値の取得や，数時間ー数日の長い間隔を置いて必要な降雨量のリセット制御，通信開始時に一度のみ必要なセンサの初期設定等に対して，柔軟に対応出来る．<Interface>要素では，センサとセンサゲートウェイ間の通信手順とやり取りされるメッセージの形式を設定する．図 4 の例では<Sender>要素内で，ASCII 形式で温度，湿度，気圧を要求するコマンド"0R2"とコマンドの終端"¥r¥n"を設定している．なお，<Sender>要素はセンサが読み取り式である場合は設定しなくてよい．図 4 中の<Recver>要素では，受信するメッセージは ASCII 文字列形式で始端"0R2"，終端"¥r¥n"であることを意味している．また，<Size>要素を用いることで，固定長データの受信にも対応可能である．

<Parser>要素は，<Extract>要素と<Regularization>要素から成る．<Extract>要素では受信メッセージの中から任意のデータを抽出するためのパラメータを設定する．図 4 の例では ASCII 形式のメッセージ内において，"Ta"から 3 バイトの位置に目的のデータが存在することを意味する．<Regularization>要素で抽出したデータを任意の形式へ変換するためのパラメータを設定する．変換前のデータを lowdata，変換後のデータを Data とすると変換式は　Data = lowdata * <Multiplier> + <Addition>　となる．図 4 では，取得した温度を摂氏から華氏へ変換し，float 型で出力することを意味する．<Parser>要素は複数記述することが可能であり，単一の出力メッセージ内から複数のデータ項目を抽出し，形式を統一することが可能である．図 4 の例では単一メッセージ内にある三つのデータ項目（温度・湿度・気圧）の抽出および正規化を意味している．

```
<Interval>10</Interval>
<Interface type=interactive>
    <Sender format="ASCII">
      <Command>0R2</Command>
      <Terminal>¥r¥n</Terminal>
     </Sender>
    <Recver format="ASCII">
      <Beginning/>
      <Size/>
      <Terminal>¥r¥n</Terminal>
  <Recver>
</Interface>

<Parser item="temperature">
    <Extract format="ASCII">
      <Marker>Ta</Marker>
      <Distance>3</Distance>
    </Extract>
  < Regularization unit="F" format="float">
      <Multiplier>1.8</Multiplier>
      <Addition>32</Addition>
  </Regularization>
</Parser>

<Parser item="humidity">……</Parser>
<Parser item="pressure">……</Parser>
```

図 4　<Unit>要素内のデータ構造

2.3. ミドルウェアの設計

前節で定義したプラグインを読み込み，処理を行うミドルウェアを設計する．ミドルウェアのおおまかなフローチャートを図 5 に示す．以下，図 5 のフローチャートに基づいて説明する．まず load_plugin()ではセンサ毎に記述したプラグインを読み込む．configure_serialport()ではプラグインの< SerialPort >要素で記述された内容を元に，非同期通信ポートを制御する termios 関数群および構造体を用いてシリアルポートの初期設定を行う．

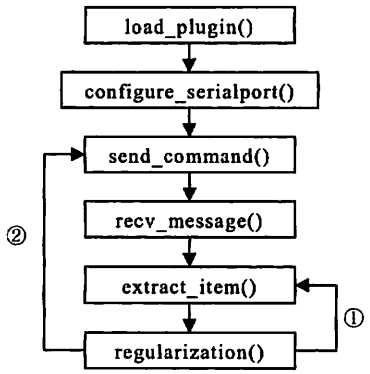

図 5　ミドルウェアのフローチャート

センサの通信手順が対話式の場合，send_command()により<sender>要素で指定した形式でコマンドを送信する．recv_message()ではプラグインで設定した始端や終端，サイズを参照し，メッセージを受信する．受信したメッセージは char 型配列に格納する．格納されたメッセージ内から目的のデータ項目を extract_data()で抽出する．最後に reglarization()でデータをセンサ毎に異なるデータ形式を，任意の単位や形式に統一する．ここで<Parser>要素が複数記述されている場合，その数だけ図中の①ように処理を繰り返す．また，同様に<Unit>要素が複数記述されている場合，その数だけ図中の 2 ように処理を繰り返す．なお，send_command()以下の処理は，<Interval>要素で設定した間隔で実行される．

3. 実装

設計したミドルウェアの有効性を検証するため，実装する．ミドルウェアを実装するゲートウェイとして，Armadillo-210 を採用した．Armadillo-210[5]は Atmark Techno 社が提供する Linux 対応の小型 CPU ボードである．表 2 にそのハードウェア及びソフトウェア仕様を示す．Armadillo-210 は I/O インタフェースとして RS232C および Ethernet を搭載しており，センサゲートウェイとして適している．プラグインは各センサの仕様書や説明書を参照し記述する．

プロセッサ	EP9307
CPUコア	ARM920T
クロック	200MHz
SDRAM	32MB
FLASH	4MB

OS	Linux 2.6
I/O	Ethernet Serial GPIO
開発環境	gcc4.1 libc6

表 2 Armadillo-210 の仕様

4. 評価

Armadillo-210 と表 1 に示す各センサを接続し，ミドルウェアの動作を検証した．結果，各センサから任意のデータ項目を取得し，形式を変換出来ることを確認した．表 3 に対応可能なセンサの仕様を示す．

表 3　対応可能なセンサの仕様

I/Oインタフェース	RS232C
通信手順	対話式、読み取り式
送信コマンド形式	ASCII、NMEA-0183、SDI-12
受信メッセージ形式	ASCII、NMEA-0183、SDI-12 16進数、2進化10進数
受信メッセージサイズ	固定長、可変長
センサの制御	可能
任意のデータ項目の取得	可能
出力データの統一	可能

以下開発したプラグインにより，新たなセンサのデータ共有およびミドルウェアの機能変更に要する作業がどれほど軽減したかを検証する．検証は現状の作業工程と比較して行う．現状の作業工程を図 6 に，開発ミドルウェアを適応した場合の工程を図 7 示す．

現状では，特にソースコードの記述が新たなセンサのデータ共有を妨げる原因となっている．一つ目の理由はソースコードの記述はプログラミング能力に加え，ハードウェアに関する知識が求められるためである．二つ目の理由は，ソースコードの記述法は明確に定義されていないため，再利用性に欠けるためである．プログラミング能力等を持たないセンサの設置者がデータの共有を望む場合，情報基盤の運用者等にミドルウェアの開発を依頼することにより解決できる．しかしこの場合，実装以降は両者間で密な連携が必須であり，非効率である．特にミドルウェアが正常に動作するかは，実際にセンサと通信することでしか判別できないため，デバッグ作業は困難となる．

一方，プラグインを取り入れたミドルウェアの場合，以下の工程を削減出来る．

(1) 開発環境の構築
(2) ソースコードの記述
(3) 実装

ただし，(2)に関してはソースコードの記述の代わりにプラグインの記述を要する．プラグインはソースコードとは異なり，データ構造が明確に定義されているため，仕様書さえあれば誰にでも記述可能である．また，同様の理由から再利用性に優れおり，一部同じ仕様を持つセンサに対しては該当部分を転用可能である．プラグインの改編作業に関しては，前述したようにプラ

グインの記述は容易であるため，設置者が単独で行える．

これらのことから，本研究で開発したミドルウェアは特に高コストであったソースコードの記述を作業工程から削減出来る．また，開発者と設置者が連携して進めていた作業を設置者が単独で行えるため，作業効率が向上する．

続いてミドルウェアの機能更新として，センサから取得するデータ項目を変更する場合を考える．現状ではわずかな機能変更であっても，ソースコードを改編する必要がある．そのため，図 6 に示す全行程を要し高コストである．一方，プラグインを取り入れたミドルウェアの場合，プラグインの編集は要するが，編集箇所は図 4 に示す<Unit>要素内だけで済む．よって，ミドルウェアの更新コストは格段に軽減する．

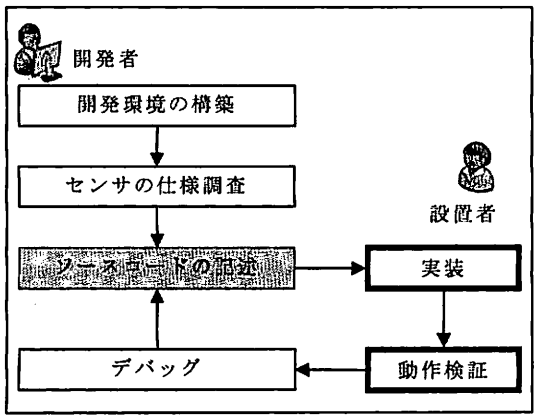

図 6　現状のデータ共有に要する作業工程

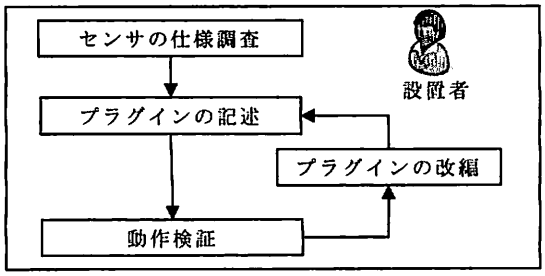

図 7　提案するミドルウェアを適応した作業工程

5. 今後の課題

今後の課題として以下の 3 つに取り組む．

(1) より多くのセンサに対するミドルウェアの評価

本論文では代表的な 3 種類のセンサに対してのみ評価を行った．今後は開発したミドルウェアを外部へ公開し，さらなるセンサに対してその妥当性を検証する必要がある．

(2) ミドルウェアの実基盤への導入

開発したミドルウェアの実基盤への導入に関して，プラグイン管理機構およびユーザインタフェースの構築が必要である．プラグインは再利用可能であるため，ユーザや開発者間の共有によりさらなるコスト削減に繋がる．よって，プラグインを収集・管理・提供する機構を構築する．続いて，ユーザインタフェースはプラグインの取得および生成に対して必要である．前者に対してはセンサゲートウェイに任意のプラグインを取得する CGI を実装する等して対処する．後者は，GUI を用いたプラグイン生成アプリケーションを提供することで解決を図る．

(3) 多様な I/O インタフェースを対象としたミドルウェアの開発

本研究では，現存する通信可能なセンサの多くが RS232C 通信可能であることに着目し，ミドルウェアを開発した．しかし，今後は USB や無線インタフェースを持つセンサも増加すると考えられる．今後は本研究で得た知見を活かし，多様な I/O インタフェースを考慮したミドルウェアを開発する．

6. まとめ

本研究ではプラグインを記述することで多様なセンサからデータを取得し，取得したデータ形式を統一するミドルウェアを開発した．開発したミドルウェアは，大規模センサ共有基盤におけるセンサゲートウェイへの実装を想定している．本ミドルウェアを小型 CPU ボード Armadillo-210 に実装し，その妥当性を検証した．結果，多様なセンサに対して通信手順やメッセージ形式，データ形式を抽象化し，任意の形式のデータを取得可能であることを確認した．本ミドルウェアの開発により，大規模センサデータ共有基盤において，データの収集量の増加を妨げていた要因である高コストなミドルウェア開発および更新を解決出来る．

文　献

[1] Live E! web site “http://www.live-e.org/”
[2] VantagePro2 web site
http://www.davisnet.com/weather/products/vantage2.asp
[3] WXT510 web site
“http://www.vaisala.co.jp/weather/products/weathermultisensor”
[4] WM918 web site
“http://www.oregonscientific.com”
[5] Armadillo-210 web site
“http://www.atmark-techno.com/products/armadillo/a210”