Vol.2018-CDS-22 No.7
2018/6/1

組込み機器向け言語mruby/cにより動作する小規模農業支援のための無線センサネットワークの構築について

神崎 映光[1]　北野 数馬[2]　吉崎 翼[2]

概要：本稿では，軽量 Ruby (mruby) の一実装であり，組込み機器向け開発言語として開発が進められている mruby/c によって動作する無線センサネットワークとして，特に我が国における農家の大半を占める小規模農家の支援を目的としたシステムの構築について述べる．具体的には，mruby/c を用いて実装したセンサノードを用いたセンサネットワークシステムの試作について述べた後，センサノード特有の機能を提供するための mruby/c のライブラリの設計および実装について述べる．

On Implementation of Wireless Sensor Networks with a Programming Language for Embedded Devices, mruby/c, for Supporting Small-scale Agriculture

Akimitsu Kanzaki[1]　Kazuma Kitano[2]　Tsubasa Yoshizaki[2]

1. はじめに

近年，我が国における農業の就業人口は減少傾向にあり，後継者不足による農業従事者の高齢化が問題となっている [7]．その原因の 1 つとして，多くの農作業が熟練農業従事者による経験や勘に大きく依存しており，技術の引継ぎが非常に難しいことがあげられる．また，耕作放棄地の増加による生産農業所得の低下も問題となっている．これらの問題を解決するためには，生産性の向上，農作業の効率化や省力化，経験や勘に依存してきた作業やノウハウのデータ化などが必要となる．特に我が国においては，農家の 80%程度を小規模農家が占めており，小規模農家の支援は，我が国の農業において，安定供給や多面的機能の維持において重要な役割をもつ．

農作業の効率化や省力化，およびノウハウのデータ化を実現するシステムとして，無線センサネットワークが注目されている [2], [10]．無線センサネットワークは，無線通信機能を備えたセンサ（センサノード）を複数配置し，各センサノードが取得したセンサデータを無線通信により収集する．このようなシステムを圃場に構築することで，圃場の環境や農作物に関する情報を収集し，データとして蓄積できる．

農業向けに無線センサネットワークを利活用する研究開発は，近年になって盛んに行われており，実際にサービス展開されているのもある [3], [4]．しかし，これらの既存システムおよびサービスは，主に大規模農家向けに開発されており，高機能なセンサノードを設置するためコストが高く，またシステム運用のコストも高いため，小規模農家がこれらを導入するのは非常に困難である．小規模農家の農作業を支援するためには，設置や運用にかかるコストを抑えられる安価かつ簡易な無線センサネットワークの実現が求められる．

我々は，小規模農家においても運用可能な無線センサネットワークの実現を目指し，安価なデバイスを用いたセンサノードの試作を進めている．このセンサノードは，安価なマイクロコントローラ，センサモジュール，および無線通信デバイスによって構成されており，これにより無線

[1] 島根大学学術研究院理工学系
Institute of Science and Engineering, Academic Assembly, Shimane University, Matsue, Shimane 690–8504, Japan
[2] 島根大学総合理工学部
Interdisciplinary Faculty of Science and Engineering, Shimane University, Matsue, Shimane 690–8504, Japan

センサネットワークの構築にかかるコストの低減を実現している．

ここで，無線センサネットワークでは，広範囲に多数のセンサノードを配置するため，個々のセンサノードにおける計算能力や記憶容量が非常に小さくなることが一般的である．また，上述した試作ノードにおいても，設置にかかるコストを低減するため，計算能力や記憶容量を抑えた構成とすることが望ましい．このような背景のもと，センサノード等の組込み機器を対象とした開発言語として mruby/c の研究開発が進められている [9]．mruby/c は，オブジェクト指向スクリプト言語である Ruby の特徴を引き継ぎ，複雑なアルゴリズムを容易に記述できる特徴を持っている．そのため，現在もなお組込み機器向け開発言語の主流である C 言語と比較して 5 倍程度という高い開発生産性をもつ．また，実行プログラムのサイズを小さく抑え，メモリ消費量が非常に小さいという特徴がある．以上の特徴により，mruby/c はセンサノードとの親和性が非常に高いものと考えられる．

一方，農業向けの無線センサネットワークでは，対象となる圃場や農作物の特性に応じて収集すべきデータ等が異なるものと考えられる．そのため，必要なデータに応じて利用するセンサデバイスを変更するなど，アプリケーション要求に応じてセンサノードの構成や挙動を個別に規定する必要があり，mruby/c を用いたとしても，他の開発言語と同様，規定すべき処理を言語の構文に従って記述する必要がある．ここで，センサノードの機能として，以下をはじめとして，アプリケーションからの要求等に関わらず共通で実装が必要となる特徴的なものが存在する．

- 温度などの環境情報を取得するセンシング機能．
- データの送受信を行う無線通信機能．
- バッテリの残余電力，消費電力などを管理する機能．

これらの特徴的な機能について，搭載するデバイスなどの差異による影響を受けない統一的なライブラリを提供できれば，センサノードの動作を規定するための開発の煩雑性をさらに緩和できるものと考えられる．

本稿では，小規模農業の支援に適した圃場観測用センサネットワークを対象として試作した，mruby/c によって動作するセンサネットワークシステムについて述べる．また，センサノード特有の機能の記述をさらに容易にすることを目的に実装したライブラリの設計，および試作システムにおけるセンサノードへの実装について述べる．

以降では，2 章で関連研究およびシステムについて述べ，3 章で組込み機器向け開発言語 mruby/c について述べる．4 章で試作したセンサネットワークシステムについて述べ，5 章でセンサノード向けライブラリの設計および実装について述べる．最後に 6 章でまとめと今後の課題について述べる．

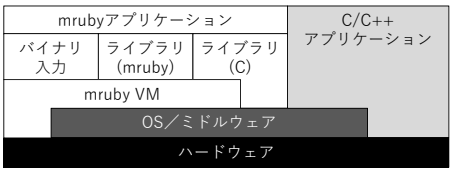

図 1　mruby のソフトウェア構成

2. 関連研究

農業の支援を目的とした無線センサネットワークの試作および構築については，国内外を問わず盛んに研究開発が行われている [1], [8]．これらの研究では，それぞれが独自のデバイスを用いてセンサノードの実装を行い，その動作を規定するソフトウェアについても，C 言語をはじめとした開発言語を用いて，各々のアプリケーションに対応したものを個別に実装している．ここで，これらの異なる研究においても，センサデバイスを用いたセンシング動作や，データ収集のための無線通信は共通で行われるものであり，本稿で述べるライブラリを用いることで，実装がより容易になるものと考えられる．

また，無線センサネットワークを対象とした開発言語や API に関する研究開発も，これまでにいくつか行われている [5], [6]．しかし，本研究で対象とする mruby/c のように，高い開発生産性をもち，かつ省メモリなものは，我々の知る限り存在していない．

3. 組込み機器向け開発言語 mruby/c

mruby/c は，オブジェクト指向プログラミング言語 Ruby の軽量版である mruby[11] の一実装として開発が進められている組込み機器向け開発言語である．Ruby の特徴を引き継ぎ，複雑なアルゴリズムを容易に記述でき，また非常に軽量であるという特徴をもつ．

mruby/c の基礎である mruby のソフトウェア構成を図 1 に示す．図中の mruby アプリケーションは，mruby によって記述されたスクリプトを mruby コンパイラ (mrbc) を用いて変換したバイナリファイルであり，バイナリ入力により mruby VM 上で実行される．また，mruby や C 言語によって記述されたライブラリ (mrbgems) を組み込み，VM 上で動作するアプリケーションを生成することもできる．さらに，C/C++言語で記述されたプログラムに mruby アプリケーションを組み込むことも可能であり，C/C++言語で記述された既存のプログラムとの高い互換性ももつ．

mruby/c の基本的な仕様は mruby と同様であるが，ワンチップマイクロコントローラをはじめとした小型機器を対象に開発されており，またオペレーティングシステムの支援を期待せず，メモリ管理などは自身の機能によって賄うといった特徴をもつ．これらの特徴により，mruby アプリケーション実行時に 400[KB] 程度の必要となるメモリ容

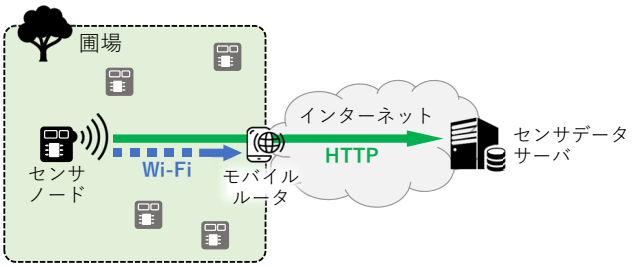

図 2　試作ネットワークの構成

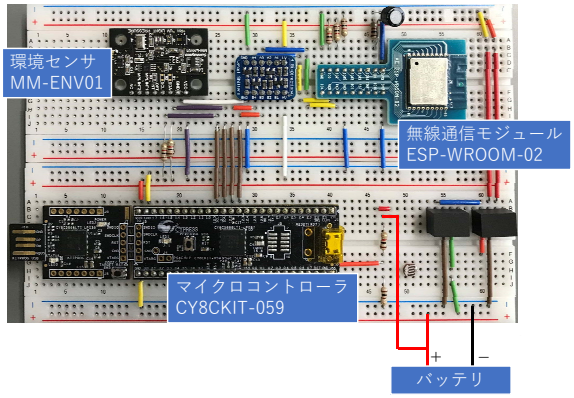

図 3　試作センサノード

量を，mruby/c では 40[KB] 未満にまで抑えている．

4. 試作システム

本章では，本研究において試作したセンサネットワークシステムについて述べる．

試作システムの構成を図 2 に示す．試作システムでは，圃場内にモバイルルータおよび複数のセンサノードを設置し，これらを IEEE802.11g によって接続することで無線センサネットワークを構築する．センサノードが取得した環境情報は，モバイルルータを介してインターネット上のデータサーバにアップロードされる．このとき，データサーバとの接続には HTTP を用い，データサーバは，センサノードから受信したセンサデータを自身のデータベース (MongoDB) に蓄積する．

本システムにおいて設置する試作ノードの概観を図 3 に示す．試作ノードは，mruby/c による動作確認が行われている小型マイクロコントローラであるサイプレス社（Cypress Semiconductor Corporation）製の PSoC 5LP プロトタイプキット CY8CKIT-059 を用い，サンハヤト社製の I^{2}C センサモジュール MM-ENV01 から取得した環境情報を，Espressif Systems 社製の無線通信モジュール ESP-WROOM-02 を介して送信する．各ノードにはノード ID が一意に設定されており，予め設定された周期に基づき，小型マイクロコントローラが一定時間ごとに以下の動作を行う．

(1) I^{2}C 通信により，センサモジュールから環境情報（温度，湿度，照度，気圧）を取得する．また，実証実験において消費電力特性の調査を行う必要性を考慮し，環境情報とあわせてバッテリの電源電圧値も測定する．

(2) 図 4 に示すように，自身に設定されているノード ID，取得した環境情報，および電源電圧値を HTTP GET パラメータに変換する．

(3) 変換した HTTP GET パラメータを含む HTTP リクエストを生成し，無線通信モジュールを介してデータサーバに送信する．

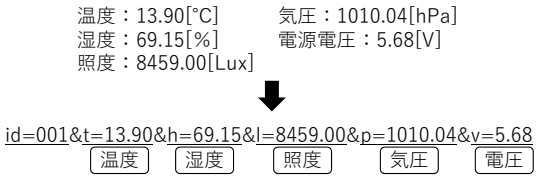

図 4　HTTP GET パラメータへの変換（ノード ID ‘001’ の場合）

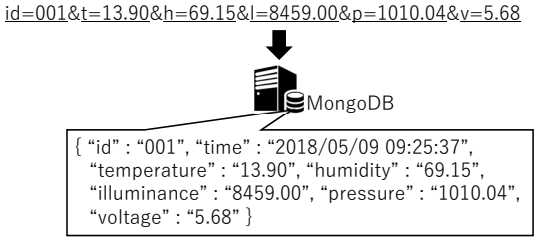

図 5　MongoDB データベースへの格納

センサノードからの HTTP リクエストを受信したデータサーバは，図 5 に示すように，リクエスト内に含まれる HTTP GET パラメータを解析し，リクエスト受信時刻とともに新たなレコード（ドキュメント）として MongoDB データベースに格納する．

5. センシング機能を提供するライブラリ

1 章で述べたとおり，アプリケーション要求に応じてセンサノードの構成や挙動を規定する必要がある状況においては，搭載するデバイスの差異等による影響を受けない統一的なライブラリを提供することが有効であるものと考えられる．本稿では，センサノード特有の機能のうち，センサデバイスから環境情報を取得するセンシング機能に着目し，当該機能を提供するライブラリを設計した．

ここで，センサノードにおけるセンシング動作としては，以下の 3 種が多くのアプリケーションにおいて必要となるものと考えられる．

- ある指定された時刻において単一または複数種別の観測値を 1 回取得する機能
- 予め設定された時間間隔で定期的に上記の観測値取得を行う機能
- 観測値が特定の値を超えた場合や，特定の閾値をまたいで変化した場合にのみ観測値取得を行う機能

本稿では，上記のうち最も単純なものとして，単一種別の

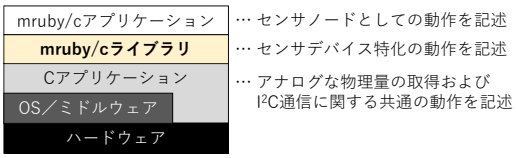

図 6 提案ライブラリを導入したソフトウェア構成

観測値を 1 度だけ取得する機能について，mruby/c を用いたライブラリの設計を行った．以下では，提案するライブラリの設計方針について述べた後，前述した試作ノードへの実装について述べる．

5.1 設計方針

3 章で述べたとおり，mruby/c は，mruby アプリケーションと同様，mruby コンパイラを用いて生成したバイナリファイルを，C/C++言語で記述されたプログラムに組み込んで利用可能である．本稿では，多くの組込み機器において用いられている C 言語で記述されたプログラムの再利用性を考慮し，C 言語のプログラムに組み込んで利用する環境において利用可能なライブラリの提供を目指す．

ここで，センサノードとしての利用が想定されるマイクロコントローラを対象にセンシング機能を実装しようとした場合，センサデバイスとの連携動作等，ハードウェアの動作に直結した箇所については，マイクロコントローラ特化の API を用いる必要があり，本稿での提供を目指す機能すべてを mruby/c で記述することは難しい．一方，想定されるセンサデバイスとしては，マイクロコントローラとの接続仕様においては種類が限定されており，特に以下の 2 種を用いることが大半である．

アナログセンサ 観測対象となる物理量を，抵抗値や電位差等に変換するデバイス．センサデバイスからの出力はアナログな物理量となり，多くの場合，マイクロコントローラにおいて電位差として取得される．

I²C センサ I²C におけるスレーブとして動作し，マスタとなるマイクロコントローラに対し，I²C 通信を用いて観測値を提供するデバイス．センサデバイスからの出力値は，ある範囲内で規定された離散値となる．

上記を考慮して，図 6 に示すように，上記のセンサデバイスを扱うために共通で利用される基本的な動作のみ C 言語を用いて記述し，取得した物理量および観測値の取扱いなど，センサデバイスに特化した動作を mruby/c を用いて記述するよう，機能の切分けを行う．これにより，新たなデバイスに対応する機能についても mruby/c を用いて容易に記述できるようになり，センサノードの動作を規定するプログラムの拡張性を向上させる．

5.2 ライブラリの処理例

ここでは，温度を観測するセンサデバイスを用いる場合を想定し，当該デバイスから温度情報を取得するライブラリとして `getTemperature` メソッドを提供する際の構成および処理について述べる．

まず，温度センサからの観測値の読取りは，ハードウェアとの連携動作を行う C 言語のプログラムにおいて提供する．温度センサがアナログセンサである場合は，センサが出力した電圧値を読み取り，その値をそのまま返す機能を提供する．一方，I²C センサである場合は，スレーブであるデバイスとの通信を行うためには，当該デバイスのアドレスや，当該デバイスから特定のデータを読み出すためのレジスタ，および読み出すデータ量を指定する必要がある．これを実現するため，引数としてアドレス，レジスタ，およびデータ量を指定できる関数群を提供する．

mruby/c ライブラリでは，C 言語のプログラムにおいて提供される機能を用いて，温度情報を取得するための `getTemperature` メソッドを提供する．利用する温度センサがアナログセンサの場合は，C 言語側で当該センサからの入力値を得る関数を呼び出し，その値を取得する．一方，I²C センサの場合は，デバイスに応じた引数を指定した上で，C 言語側でデータを取得する関数を呼び出し，その値を取得する．

ここで，これらのセンサから得られる値は摂氏等の単位系で与えられる温度そのものではなく，電圧値や離散的な数値であるため，温度情報として mruby/c アプリケーションに値を提供するためには，C 言語プログラムから取得した値に対する変換等の処理を行う必要がある．ただし，単位系への変換が困難な値を出力するアナログセンサを用いる場合や，変換前の入力値そのものを用いた処理を行う場合があることも考慮し，返り値として単位系への変換を行うか否かを指定できるフラグを引数として提供する．本節の説明で用いている `getTemperature` メソッドにおいては，摂氏に変換するか否かを示すフラグを提供し，フラグが指定されており，かつ摂氏への変換が可能な場合は，変換後の値を返すものとする．

最後に mruby/c アプリケーションでは，mruby/c ライブラリで提供されている `getTemperature` メソッドを，摂氏への変換を行うか否かを示す引数を指定しながら用いることで，所望の形態で温度データを取得できる．

5.3 試作ノードへの実装

4 章で述べた試作ノードに，本章で述べたライブラリを用いたプログラムを実装した．このプログラムでは，電源電圧値をアナログセンサ，センサモジュール MM-ENV01 を I²C センサとして扱い，電源電圧値および環境情報を取得する処理を，提案するライブラリを用いて記述した．以下では，電源電圧値および温度情報を取得するために実装した `getVoltage` メソッドおよび `getTemperature` メソッドを用いた記述について詳述する．

```
static void c_getAnalogData(mrb_vm *vm, mrb_value
  *v) {
  int result = 0;
  analog_sensor_StartConvert();
  if ( analog_sensor_IsEndConversion(
     analog_sensor_RETURN_STATUS)) {
    result = analog_sensor_GetResult16();
  }
  SET_INT_RETURN(result);
}
```

図 7 アナログデータ取得のための関数

```
def getVoltage()
  return getAnalogData()
end
```

図 8 電源電圧値取得のためのメソッド

5.3.1 電源電圧値の取得

C 言語プログラムでは，図 7 に示すように，PSoC 5LP の仕様に従ってアナログデータを取得する処理を記述した c_getAnalogData 関数を定義している．

mruby/c ライブラリでは，図 8 に示すように，mruby/c アプリケーションに提供する getVoltage メソッドを定義している．このメソッドでは，上述した c_getAnalogData 関数にマッピングされている getAnalogData メソッドを呼び出し，取得した値をそのまま返す．

mruby/c アプリケーションでは，mruby/c ライブラリによって提供される getVoltage メソッドを呼び出すことにより，電源電圧値を取得できる．

5.3.2 温度情報の取得

C 言語プログラムでは，図 9 に示すように，PSoC 5LP の仕様に従って I^2C 通信によるデータの読出しおよび書込みを行う処理を記述した関数群を定義している．

mruby/c ライブラリでは，図 10 に示すように，mruby/c アプリケーションに提供する getTemperature メソッドを定義している．なお，試作ノードにおいて使用した温度センサは，アドレスが 0x40 であり，温度データを取得するためには，3 バイトのデータ書込みの後，2 バイトのデータ読出しを行う必要がある．これらを実現するため，上述した関数群にマッピングされているメソッドを，必要なアドレスおよびデータを引数として呼び出し，読み出した値を変数 data に格納している．ここで，I^2C センサから取得する入力値は離散的な数値であり，摂氏としての温度情報を取得するためには，入力値に対する変換を行う必要がある．そのため getTemperature メソッドでは，単位系の変換を行うか否かを指定する引数 conv を提供し，この値が 1 に設定された場合に摂氏に変換した値を返す（図 10 の 6–7 行目）．

mruby/c アプリケーションでは，mruby/c ライブラリによって提供される getTemperature メソッドを呼び出すことにより，温度情報を取得できる．このとき，getTemperature メソッドに与える引数を変更することに

```
/* I2C通信によるデータ読出し */
uint8 I2C_Read(uint8 slaveAddr, uint8 *dataAddr,
  uint8 byte, uint8 mode) {
 uint8 result;

 result = I2C_MasterWriteBuf(slaveAddr, dataAddr,
   1, mode);
 while(result != I2C_MSTR_NO_ERROR);
 while(I2C_MasterStatus()&I2C_MSTAT_XFER_INP);
 result = I2C_MasterClearStatus();

 CyDelay(60);

 result = I2C_MasterReadBuf(slaveAddr, dataAddr,
   byte, mode);
 while(result != I2C_MSTR_NO_ERROR);
 while(I2C_MasterStatus()&I2C_MSTAT_XFER_INP);
 return result;
}

/* 1バイト読出し */
static void c_I2C_Read8(mrb_vm *vm, mrb_value *v)
  {
 uint8 wr_buf = reg;
 I2C_Read(GET_INT_ARG(0), &wr_buf, 1,
   I2C_MODE_COMPLETE_XFER);
 SET_INT_RETURN(wr_buf);
}

/* 2バイト読出し */
uint16 c_I2C_Read16(mrb_vm *vm, mrb_value *v) {
 uint16 result;
 uint8 wr_buf[2] = {reg, 0x00};
 I2C_Read(GET_INT_ARG(0), wr_buf, 2,
   I2C_MODE_COMPLETE_XFER);
 result = wr_buf[0] << 8;
 result += wr_buf[1];
 SET_INT_RETURN(result);
}

...
```

図 9 I^2C 通信のための関数（データ読出しに関する箇所のみ抜粋）

```
def getTemperature(conv)
 I2C_Write24(0x40,0x02,0x00,0x00)
 data = I2C_Read16(0x40)

 if conv==1 then
  temp = ((data*165)/64436)-40
  return temp
 else
  return data
 end
end
```

図 10 温度情報取得のためのメソッド

表 1　プログラム比較

	C 言語	mruby/c
プログラムサイズ	1375[B]	2739[B]
動作記述部のステップ数	28	9

より，I^{2}C センサから得られた入力値および摂氏に変換された温度情報のいずれも利用可能であり，これにより，センサノードの動作をより柔軟に記述できる．

5.4 考察

本章で述べたライブラリにより，使用するセンサデバイスに特化した処理に関する記述が mruby/c ライブラリに集約される．そのため，センサデバイスの変更が生じた場合でも，mruby/c ライブラリを当該デバイスに対応したものに差し替えるのみで，デバイス変更前と同等の動作を行うセンサノードが設置可能となる．また，ライブラリとして提供するメソッドを mruby/c を用いて記述できるため，新たなデバイスに対応するライブラリも容易に作成できる．

5.5 動作検証

提案ライブラリの有用性を検証するため，試作ノードを動作させるプログラムを実装し，プログラムサイズおよびステップ数について，C 言語のみを用いて実装した場合と比較した．結果を表 1 に示す．結果より，プログラムの記述に mruby/c を用い，また提案ライブラリを用いることにより，C 言語と比較して main 関数内におけるステップ数をおよそ 32%にまで削減でき，センサノードとしてのプログラムを容易に記述できることがわかる．一方，プログラムサイズは C 言語と比較して大きくなるものの依然として十分に小さく，小型マイクロコントローラにおいて動作可能なプログラムが容易に記述できることが確認できる．

6. おわりに

本稿では，小規模農業の支援を目的とした圃場観測用センサネットワークシステムの試作，およびセンサノードの機能に特化した機能を提供するライブラリの設計および実装を行った．特に，センサノードに搭載されたセンサデバイスから環境情報を取得するセンシング機能に着目し，アナログセンサおよび I^{2}C センサからのデータ取得を容易に記述できるライブラリを設計し，試作ノードを用いた動作検証を行った．

現在は，センシング機能として，特定種別の観測値を 1 度だけ取得する機能を実装するに留まっているが，5 章で述べたとおり，無線センサネットワークでは，定期的な観測などもよく利用されることが想定される．今後は，これらのライブラリも提供し，さまざまなアプリケーションに対応したセンシング機能の提供を行う予定である．また，1 章で述べた無線通信機能や省電力化機能についても，設計および実装を行う予定である．さらに，実装したライブラリを mrbgems あるいはこれに類する機能を用いて利用可能とすることにより，ライブラリの再利用性を向上する方法についても検討する予定である．

謝辞　本研究の一部は，日本学術振興会科学研究費補助金・基盤研究 (C)(16K07973)，および東北大学電気通信研究所における共同プロジェクト研究によるものである．ここに記して謝意を表す．

参考文献

[1] S. Abraham and X. Li: A cost-effective wireless sensor network system for indoor air quality monitoring applications, *Procedia Computer Science*, vol.34, pp.165–171 (2014).
[2] M.H. Anisi, G.A. Salaam and A.H. Abdullah: A survey of wireless sensor network approaches and their energy consumption for monitoring farm fields in precision agriculture, *Precision Agriculture*, vol.16, no.2, pp.216–238 (2015).
[3] e-kakashi (online), 入手先 ⟨http://www.e-kakashi.com/⟩ (2018.05.07).
[4] フィールドサーバ::イーラボ・エクスペリエンス - elab experience (online), 入手先 ⟨http://www.elab-experience.com/fieldserver⟩ (2018.05.07).
[5] C.L. Fok, G.C. Roman and C. Lu: Agilla: a mobile agent middleware for self-adaptive wireless sensor networks, *ACM Trans. Autonomous and Adaptive Systems*, vol.4, no.3, pp.382–387 (2009).
[6] D. Gay, P. Levis, R.V. Behren, M. Welsh, E. Brewer and D. Culler: The nesC language: a holistic approach to networked embedded systems, *Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI 2003)*, pp.1–11 (2003).
[7] 平成 24 年度 食料・農業・農村白書（平成 25 年 6 月 11 日公表）: 農林水産省 (online), 入手先 ⟨http://www.maff.go.jp/j/wpaper/w_maff/h24/⟩ (2018.05.07).
[8] 増井崇裕, 松野智明, 安部恵一, 峰野博史, 大須賀隆司, 水野忠則, “高密度無線センサネットワークを利用した農業技術の形式知化に関する検討,” 情報処理学会マルチメディア通信と分散処理ワークショップ (DPSWS 2010) 論文集, pp.93–98 (2010).
[9] mruby/c—島根ソフト研究開発センター (online), 入手先 ⟨http://www.s-itoc.jp/activity/research/mrubyc/⟩ (2018.05.07).
[10] T. Ojha, S. Misra and N.S. Raghuwanshi: Wireless sensor networks for agriculture: the state of the attribute in practice and future challenges, *Computers and Electronics in Agriculture*, vol.118, pp.66–84 (2015).
[11] K. Tanaka, A.D. Nagumanthri and Y. Matsumoto: mruby – Rapid software development for embedded systems, Proc. Int. Conf. Computational Science and Its Applications (ICCSA 2015) (2015).