4M-04

データフローを用いた小型デバイス向け IoTアプリケーション開発

辻野 智大†，田中 和明‡

九州工業大学大学院情報工学府†，九州工業大学大学院情報工学研究院‡

1　研究背景

IoT製品を動かすためのIoTアプリケーションは，センサーで入手したデータを処理して通信するといったデータフローがあり，開発の際はこの流れを理解する必要がある．このようなデータフローを視覚的に表現しつつ，直感的な開発が可能になる方法として，GUIを用いたビジュアルプログラミングツールによる開発が挙げられる．コードを書く必要がなくなるため，プログラミング未経験者や初学者でもIoTアプリケーション開発が容易となる．

現在，データフローからIoTアプリケーションを作成できるビジュアルプログラミングツールは，OSが搭載されているようなデバイスでは動作するが，より小型なデバイスでは動作しない．プログラムのコードが自動生成されるビジュアルプログラミングツールも存在するが，生成される言語がCなどであるため，見直しや修正，書き足しなどのコストが高い．より可読性や生産性に優れた言語のアプリケーションを生成できれば，開発効率の向上が期待できる．

本研究では，小型デバイスに対応した開発効率の優れた言語のIoTアプリケーションを，データフローを用いて作成することができる環境の開発を行う．

2　システム構成

2.1　Node-RED

データフローが作成できるビジュアルプログラミングツールとして，IBMが開発したオープンソースであり，Node.js環境で動作するNode-REDを用いた．このツールはノードと呼ばれるデータの処理機能を持ったブロックと，ワイヤーを視覚的に繋げていく．そうすることで，データがノードやワイヤーを通して処理・受け渡しされていくプログラムが作成され，IoTアプリケーションとして実行することができる．

Node-REDにはデータフローをJSONファイルで書き出す機能が備わっており，本研究ではこの機能を利用する．

IoT application development for small devices using data flow
†Chihiro Tsujino・Kyushu Institute of Technology
‡Kazuaki Tanaka・Kyushu Institute of Technology

2.2　Ruby・mruby

本研究により生成されるIoTアプリケーションの言語にはRubyを用いた．Ruby自体は必要メモリ量が大きく，小型デバイスでの使用は適していない．しかし，Rubyの派生言語であるmrubyはRubyの可読性・生産性の高さといった特徴を生かしたまま，IoT向けに軽量化したものである．Rubyをコンパイルして得たバイトコードをVMで実行するため，数百KBのメモリで動作する．また，コード量がC言語などに比べて少なく，可読性・生産性に優れているため，本システムでNode-RED内のノードの機能をコード記述する際や，完成したコードの見直しをするときの効率が向上すると考えられる．

2.3　システム全体の概要

開発したシステム全体の概要を図1に示す．

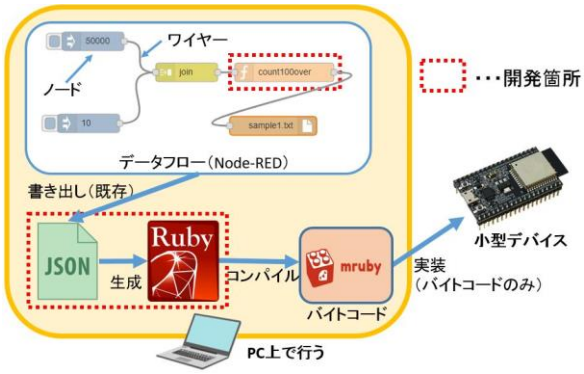

図1：システム全体図

初めにNode-REDを用いてデータフローを作成する．次に，データフローを書き出すことで作成されたJSONファイルからRubyコードを生成する．最後にRubyコードをmrubyのバイトコードにコンパイルして小型デバイスに実装する．小型デバイスにはmrubyのVMとバイトコードだけを実装し，他の作業はビジュアルプログラミングツールが使用できるPC上で行う．

本研究では，Node-RED上でRubyコードを記述できるノードの作成と，Node-REDで作成したデータフローの情報が入ったJSONファイルを読み取り，Rubyコードに変換するプログラムを作成した．

3　研究内容

本研究のソフトウェア動作環境は表 1 の通りである．

表 1：　動作環境

PC	windows 10 pro
Node.js	v8.11.3
Node-RED	0.17.4-git
Ruby	2.5.1p57
mruby	2.0.0

3.1　Ruby 記述可能ノードの作成

既存のノードを組み合わせるだけでは実現が難しいプログラムのために，図 2 のようなノードの機能を Ruby で記述できるノードを作成した．ノードの作成には HTML と JavaScript を用いた．

図 2：作成した Ruby 記述可能ノード

3.2　JSON ファイルからコード生成

データフローを書き出して作成した JSON ファイルから，Ruby コードを生成する手順を図 3 に示す．

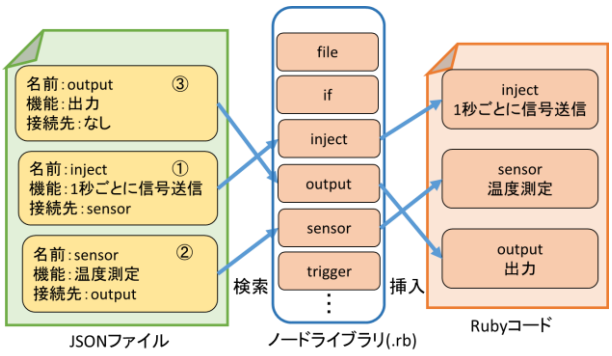

図 3：Ruby コード生成手順

Ruby には JSON 形式のファイルを Ruby で扱うハッシュの形に変換するメソッドがあるため，生成プログラムは Ruby で作成した．JSON ファイルには，ノードをデータフローに配置した順に各ノードの持つ情報が入っているため，ノードが順番通りにコード生成されるようにノードを並び替えた．その後，Ruby コード生成に対応させているノードかどうかを事前に作成したノードライブラリから検索した．コード生成に対応させているノードならば，そのノードの Ruby コードを文字列変数に挿入していった．最後に，すべてのノードの Ruby コード生成が完了したのちに，文字列変数を 1 つのファイルに出力することで，JSON ファイルから Ruby コードを生成した．

4　動作検証

4.1　検証準備

開発したシステムが機能するかを検証するためにフローを作成した(図 4)．このアプリケーションは，1 秒ごとに現在時刻を取得し，26 番ピンの LED に対して 0~49 秒の間は消灯，50~60 秒の間は点灯する．0,1 の出力や，現在時刻の秒数を出力するノードは，3.1 章で作成した Ruby 記述可能ノードに Ruby コードで記述した．3.2 章で作成した Ruby 生成プログラムを用いて，データフローから Ruby コードを生成したのちに，生成されたコードをコンパイルして Raspberry Pi3 で実行した．

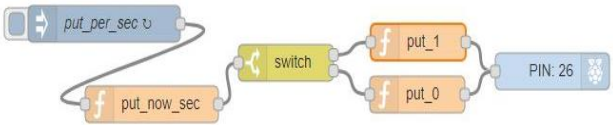

図 4：作成したデータフロー

4.2　検証結果

データフローから自動生成された Ruby プログラムをコンパイルした mruby バイトコードが，Raspberry Pi3 で動作することを確認できた．

5　まとめ

Node-RED のデータフロー上での Ruby 記述可能ノードの作成及び，JSON ファイルから Ruby コードを生成することができた．また，実際に生成された mruby コードが，Raspberry Pi3 で実行できることを確認した．アプリケーション作成時のコード記述量が減少し，データフローにより視覚的，直感的に開発ができる環境の開発ができた．

今後は，多種多様な IoT アプリケーションを作成できるように，まだ対応していない構文やノードをコード生成できるようにする．また，データフロー保存から Ruby コード生成，コンパイルして mruby コードを小型デバイスへの実装を自動化し，より快適に利用できる環境を開発していく．