2K-08

小型デバイス向けのデータフロー型プログラミング環境の構築

村上 旭人† 田中 和明‡

九州工業大学情報工学部† 九州工業大学情報工学研究院‡

1 研究背景

教育機関ではプログラミング科目が必修化されている。プログラミング科目の目的は、プログラミングロジックの理解が主である。そのため、教材としてブロック型のビジュアルプログラミング言語がよく用いられている。近年、内閣府より科学技術政策 Society5.0 が定められ、IoT や AI といったデータの活用が必要技術要素として挙げられている。特に IoT ではデータの流れ(データフロー)の理解が重要である。そのため、プログラミングロジックの理解と、データの活用の理解は異なる観点での教育方法が必要であると考える。そこで本研究ではデータフローの理解と IoT 開発の手法としてデータフロー型プログラミング環境の活用を提案する。

2 目的

本研究では、マイコンボードを利用した IoT 開発に対し、デバイス間の情報の流れを直感的に理解できるプログラミング環境の構築を目的とする。

3 開発環境

3．1 mruby/c

mruby/c は Ruby の特徴を引き継いだ組込み開発向け言語である。Ruby コードを mruby コンパイラによりコンパイルしてバイトコードに変換し、VM (バーチャルマシン)で実行することでメモリ消費量を削減している。これにより、組み込み分野でも Ruby コードによる記述で開発が行えるようになった。

本研究では Ruby または mruby/c を用いている。理由は、Node-RED 上でテキスト言語を記述するノードに対し、Ruby コードが扱える事、生成されるプログラムを見直す際の可読性の高さがあり、プログラミング初学者に理解されやすいプログラム言語であるため教育用途で非常に適していると考えた。

2．1 Node-RED

Node-RED は、IBM により開発されたデータフロー型ビジュアルプログラミング開発ツールである。プログラミング方法は、ノードと呼ばれる各機能がまとめられたブロックを配置していき、これらを線で繋ぎ合わせてフローを作成し、ノード同士でデータのやり取りを行わせることで様々な処理を行わせることができる。また、HTML と Javascript を用いてノードの自作も可能である。本研究ではこれらの特徴を持つ Node-RED を活用していき、プログラミング環境の構築を目指す。

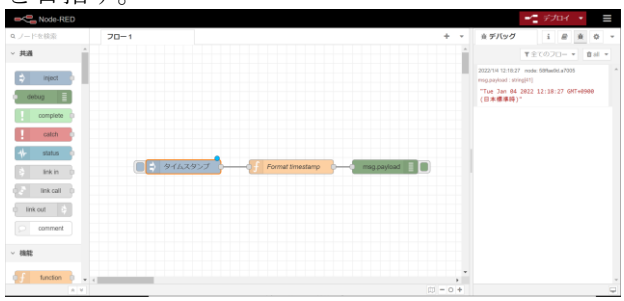

Fig. 1 Node-RED エディタ画面

4 研究内容

4．1 システム構成

本研究で開発するシステム構成を Fig.2 に示す。手順としては、まずユーザーは Node-RED 上でマイコンボード用ノードを使ってフローを作成していく。作成したフローは JSON ファイルで保存されているため、その JSON ファイルを抽出する。その後、Ruby コード生成器プログラムを実行させ、抽出した JSON ファイルを基に、マイコンボードが動作する Ruby コードを生成する。最後に mruby コンパイラにより mruby/c バイトコードに変換させ、マイコンボードに実装する。

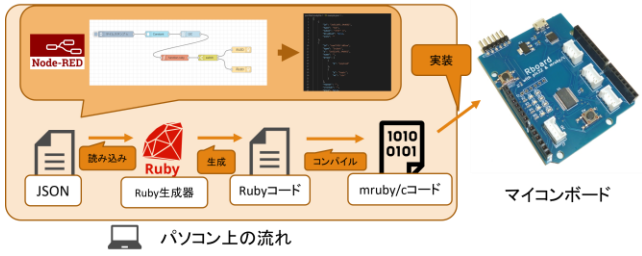

Fig. 2 フロー作成からマイコンへの実装の流れ

4．2 マイコンボード用ノードの自作

マイコンボードの制御に関して必要なノードを開発した。開発したノードと一部のノードの

Building a dataflow programming environment for small device

†Asato Murakami・Kyushu Institute of Technology

‡Kazuaki Tanaka・Kyushu Institute of Technology

編集ダイアログを Fig. 3 に記載する。ノードの外観や編集ダイアログのデザインや設定項目の情報管理などを HTML と Javascript を用いて作成した。

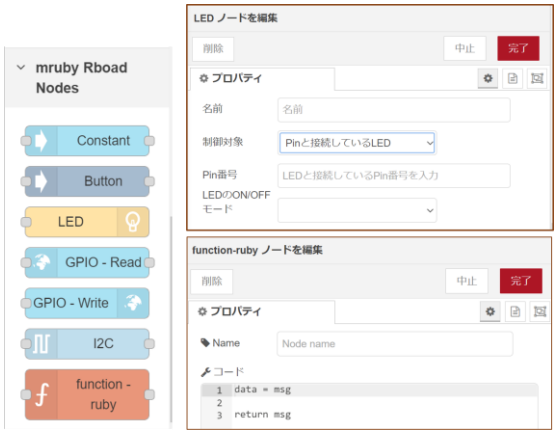

Fig. 3 (左)自作ノード一覧 (右)編集ダイアログ

4．3 マイコンボード用 Ruby コードの生成

Node-RED で抽出した JSON ファイルには、配置されたノードの種類、ノード同士の接続関係、ノードの識別 ID や設定情報といったノードのプロパティ情報が記録されている。Ruby コード生成器はこの JSON ファイルに基づき、Ruby コードの生成を行う。Ruby コード生成器の処理手順を Fig.4 のフローチャートに示す。

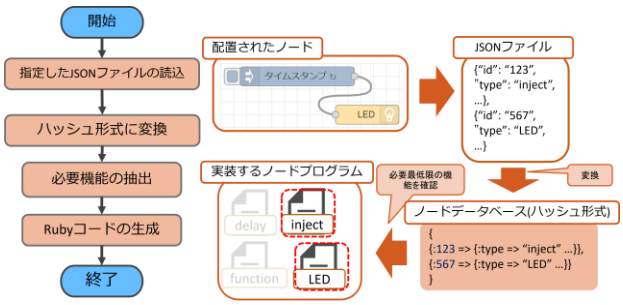

Fig. 4 Ruby コード生成手順

まず、JSON ファイルを読み込み、Ruby で扱えるように Fig.4 右図のようにハッシュ形式のデータに変換する。以降はこのノードデータベースを用いて、Ruby コードを生成していくことになる。必要機能の抽出作業では、ノードデータベースからノードのタイプを抽出し、抽出されたタイプに応じてノードプログラムを選択する。ノードプログラムとは、ノードのタイプごとに機能をまとめたプログラムであり、事前に用意されている。最後に、ノードデータベースやノード間のデータ制御機能といった基本機能と、選択されたノードプログラムをマイコンボード用 Ruby コードに記述する。生成される Ruby コードの構成を Fig.4 に示す。

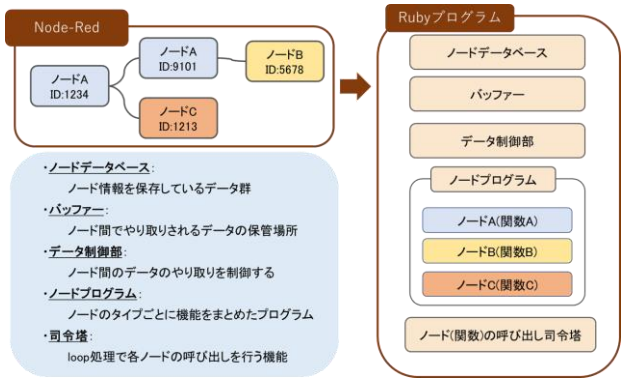

Fig. 5 生成される Ruby コードの構成

5 動作確認

温湿度センサ(SHT31)を用いて、温度による LED 制御を行うプログラムを作成し、RBoard による動作確認を行った。Node-RED で作成したプログラムを Fig.6 に示す。このプログラムは摂氏温度が 24 度以上であれば赤色 LED が点灯し、24 度以下であれば青色 LED を点灯する。

inject ノードは Constant ノードを通して 1 秒ごとにデータを送信する。受信した I2C ノードは温湿度センサにバイトコードのコマンドを送信し、センサから温度値を受け取る。しかし、受け取った温度値はバイトコードであるため摂氏温度への変換処理を function-ruby ノードによる自由記述で実装する。その後、温度値を switch ノードに送信し判定を行い、LED 制御を行う。

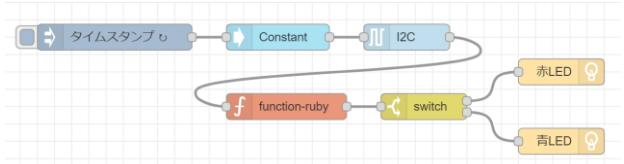

Fig. 6 作成したフロー 温度センサによる LED 制御

6 動作結果

Fig.6 のフローを基に生成された Ruby コードをコンパイルし、バイトコードを RBoard に送信した結果、想定通りの動作を行った。

7 まとめ

データフロー型ビジュアルプログラミング言語を用いたプログラミング環境の構築に成功した。これにより、データフローを意識しながらプログラム開発が行えるようになった。現在、本システムで動作できるマイコンボードが RBoard のみ対応している。他マイコンボードでも動作できるよう、Ruby コード生成ルーチンの切り替え機能の実装について検討していく。