7K-02

小型デバイス向けデータフロー型ビジュアルプログラミング環境構築の開発

尾倉　颯太† 河原　美優‡ 田中　和明* 杉山　耕一朗‡

九州工業大学情報工学部† 九州工業大学情報工学研究員* 松江工業高等専門学校情報工学科‡

1　緒言

今日ではさまざまな場所・場面で、スマートフォンやエアコンなどの多種多様な組込システムが利用されている。しかし、世界的にウェブデザインなどにもちいられる Java・JavaScript や Python などを利用する人口が増えている。そのため比較的 C 言語・C＋＋を用いた開発を行う技術者が少ないにもかかわらず、少子高齢化や会社・工場の人件費コストの削減などによる様々な組込システムに対する需要が高まっている。

また、近年では、内閣府よりサイバー（仮想）空間とフィジカル（現実）空間の融合を目指す科学技術政策 Society5.0 が定められ、IoT や AI といったデータの活用が必要な技術要素として挙げられている。特に、IoT 開発ではデータの流れであるデータフローの理解が必要なことから、データフロー型 VPL に注目した。

本研究では、データフロー型 VPL である Node-RED と C 言語より生産性・可読性が高い Ruby の特徴をもつ mruby/c を用いて、プログラミング初学者でも組込システムを開発できる組込開発環境の構築方法を提案する。

2　アプリケーション構成

2.1　mruby/c

mruby/c[2]は Ruby の特徴である高い生産性と可読性を引き継いだ組込開発環境向けの言語であることからプログラミング初学者が理解しやすい言語となっている。また、Ruby コードをコンパイラにより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、プログラム実行時に少ないメモリ消費量で実行できる。そのため、マイコンを使用した開発を行うことが可能となっている。低消費電力マイコンチップを使用すれば、省電力での実装も実現することが可能である。

2.2　Node-RED

Node-RED[3]は、IBM によって開発されたフローベースドプログラミングツールである。プログラミングの方法は、ノードと呼ばれるそれぞれ機能が与えられたブロックを配置し、それらを線でつなげてフローを作成する。ノード同士でデータを送信・受信し、様々な処理を行う。これは視覚的な表現がされているため、幅広いユーザーにとって利用しやすいツールとなっている。

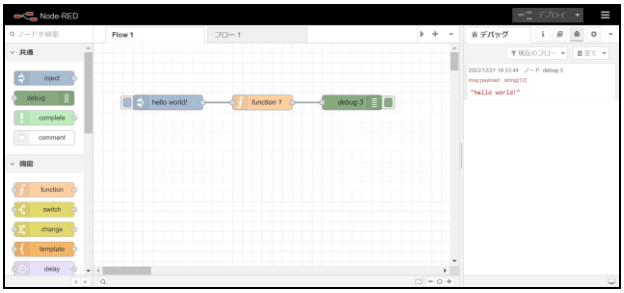

図 1：Node-RED　エディタ画面

2.3　Docker・docker-compose

Docker は Docker 社が提供するコンテナ型仮想化技術である。ゲスト OS を起動せずに、ホスト OS 上に動作している Docker Engine からコンテナと呼ばれるミドルウェアの環境構築がされた実行環境を作成する。その中でアプリケーションを動作させることで軽量に動作させることが可能である。また、Dockerfile と必要ファイルがあれば、複雑なアーキテクチャであっても簡単に共有することが可能である。

3　研究内容

3.1　システム構成

本研究で開発するシステム構成を以下の図 2 に示す。ユーザーは Node-RED 上でノードをつなげ、フローを作成する。作成したフローに保存されている JSON コードを抽出し、Ruby コードへ、Ruby コードからバイトコードへ変換する操作をサーバー上で行う。その後、Web Serial API にてバイトコードを受信し、USB シリアルでパソコンと接続した RBoard にバイトコードを転送し、実行する。

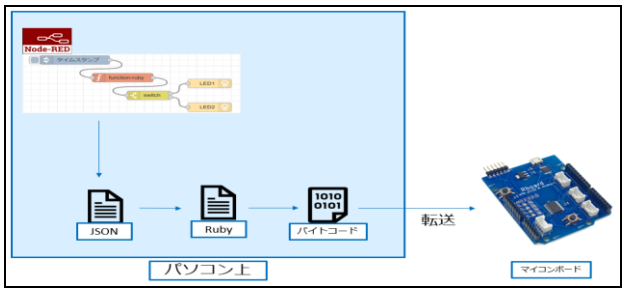

図 2：システム構成

3.2 ノードの開発

Node-RED では、オリジナルのノードを作成することができる。作成したノードと一部のノードの編集ダイアログを図 3 に示す。ノードの外観や編集ダイアログのデザイン、ノードの設定項目などを JavaScript と HTML を用いて作成し、Dockerfile に記述することで反映することが可能である。

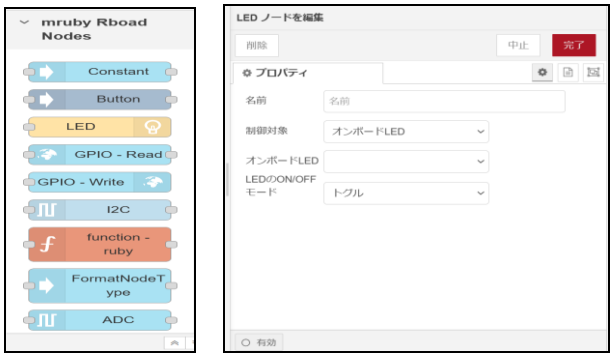

図 3：オリジナルのノード一覧(左)
LED ノードの編集ダイアログ(右)

3.3 フロー作成から実行までの手順

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・識別 ID、ノード同士の接続 ID、フローID、設定した情報などが記載されている。この JSON ファイルを基にバイトコードを生成する。図 4 にフロー作成から実行までのフローチャートを示す。

まず、Node-RED 上で各ノードをつないでフローを作成する。そのフローの JSON コードから Ruby で扱えるようにハッシュ形式のデータに変換する。これ以降はノードのデータベースから必要機能の抽出を行い、抽出したノードのタイプに応じて事前に用意されたノードのプログラムを選択する。そして、ノードのデータベースやノードのプログラムをまとめて Ruby コードとして生成する。生成された Ruby コードをバイトコードへ変換し、シリアルポートへ送信してプログラムを実行する。バイトコードを書き込む際に、生成されたバイトコードを配列形式で読み込み、8 ビット符号なし整数値に変換したものを送信し、実行する。

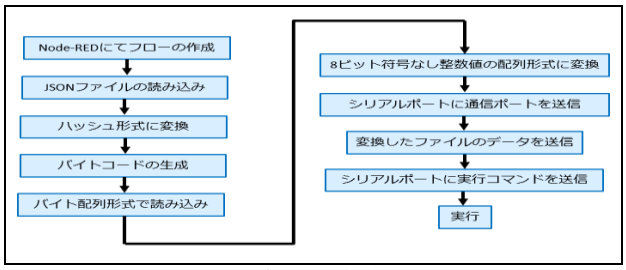

図 4：フロー作成から実行までの手順

4 動作検証

4.1 温度センサによる LED 制御のプログラム作成

温度をアナログ電圧に変換する温度センサを用いた LED 制御を行うフローを作成し、RBoard[4]による動作検証を行う。Node-RED で作成したプログラムを以下の図 5 に示す。このプログラムは摂氏 15 度以上であれば赤色 LED を、15 度未満であれば緑色 LED を点灯させる。使用した温度センサは温度をアナログ電圧として受信するため、自由記述が可能な function-ruby ノードにて、アナログ電圧から温度への変換を行う。その後、switch ノードにて判定を行い、それぞれの LED を点滅させる。

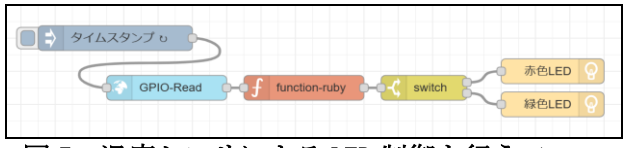

図 5：温度センサによる LED 制御を行うフロー

4.2 動作結果

図 5 に示したフローを基に Ruby コードを生成、コンパイルし、バイトコードを RBoard に転送した結果、想定通りの動作結果を得られた。

5 結言

4 章より、データフロー型 VPL を用いた開発環境の構築が成功した。このことから、これを用いた経験を問わず IoT 開発を行うことが可能だと考える。今後は、まだ実装していないノードを実装し、RBoard 以外のマイコンボードでも利用できるようにアプリケーションの汎用化を目指す。

参考文献

[1]村上旭人・田中和明, 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, 情報処理学会第 84 回全国大会, 2022
[2]mruby/c, しまねソフト研究開発センター URL:https://www.s-itoc.jp/activity/research/mrubyc/
[3]Node-RED, https://nodered.jp/
[4]株式会社 島根情報処理センター, RBoard, https://www.sjcinc.co.jp/service/rboard, 2022/10/18 .