

データフロープログラミング for mruby/c

九州工業大学 情報工学部
田中 和明

本日の内容

- 自己紹介（+ 大学紹介）
- Ruby, mruby, mruby/c
- mruby/c演習
- データフロープログラミング
- データフロー演習

自己紹介＋大学紹介

スライド資料

Ruby, mruby, mruby/c

プログラム言語Ruby

- スクリプト言語
 - コードが書きやすい、読みやすい
 - 多機能
 - Webサービスなどでよく使われる

例：Rubyのコード

```
a = [1,2,3,4]

a.each do |data|
  puts data * data
end
```

```
score = [6.1, 6.5, 7.3, 5.8, 6.4, 7.6, 6.9]

puts score.sort[1..-2].sum / (score.size - 2)
```

例：Rubyのコード

```
a = [1,2,3,4]

a.each do |data|
  puts data * data
end
```

```
a = [1,2,3,4]

for data in a do
  puts data * data
end
```

```
a = [1,2,3,4]

n = a.size
for i in 0...n do
  puts a[i] * a[i]
end
```


Ruby → mruby

- Rubyを、小さいマイコンで動かしたい
 - 2010年～ 経済産業省のプロジェクト
 - オープンソース

`https://github.com/mruby/mruby`

- 実行時に必要なメモリ
 - Rubyは数MB、OSが必須
 - mrubyは100KB程度

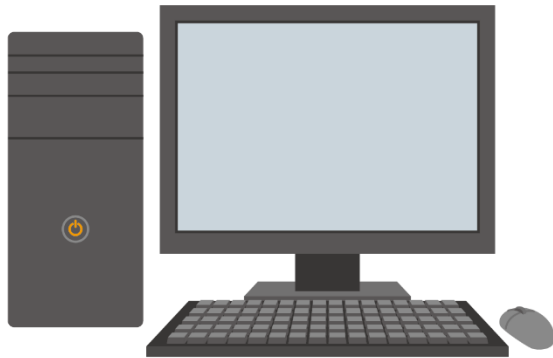
mruby → mruby/c

- Mrubyを、さらに小さくしたい
 - しまねソフト研究開発センターとの共同開発
 - オープンソース

`https://github.com/mruby/mruby`

- 実行時に必要なメモリ
 - mrubyは100KB程度
 - mruby/cは40KB程度

Ruby, mruby, mruby/c



Ruby



mruby



mruby/c

技術的な違い

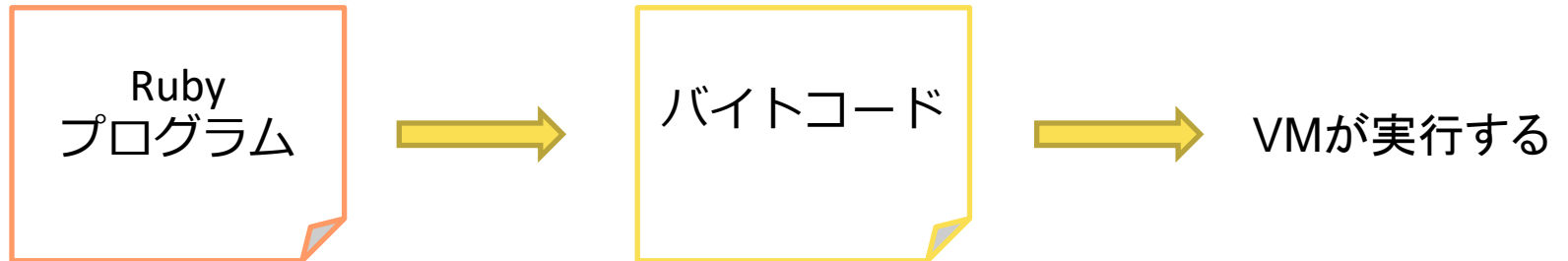
	Ruby	mruby	mruby/c
実行方式	インタプリタ	コンパイラ + VM	コンパイラ + VM
機能	多くのライブラリ	ISO準拠	ISOのサブセット
ライブラリ	gem	mrbgem	(無し)
POSIX機能	必須	一部に必要	不要
実行性能	高い	中程度	低い
動的メモリ割り当て	必要	必要	不要
コンカレント実行	×	×	○

実行方式

Ruby:



mruby, mruby/c:



DEMO

- 実行の方法
- 消費メモリ量

```
/usr/bin/time -f "%M" ruby
```

mruby/c演習



mruby/cを使ってみる

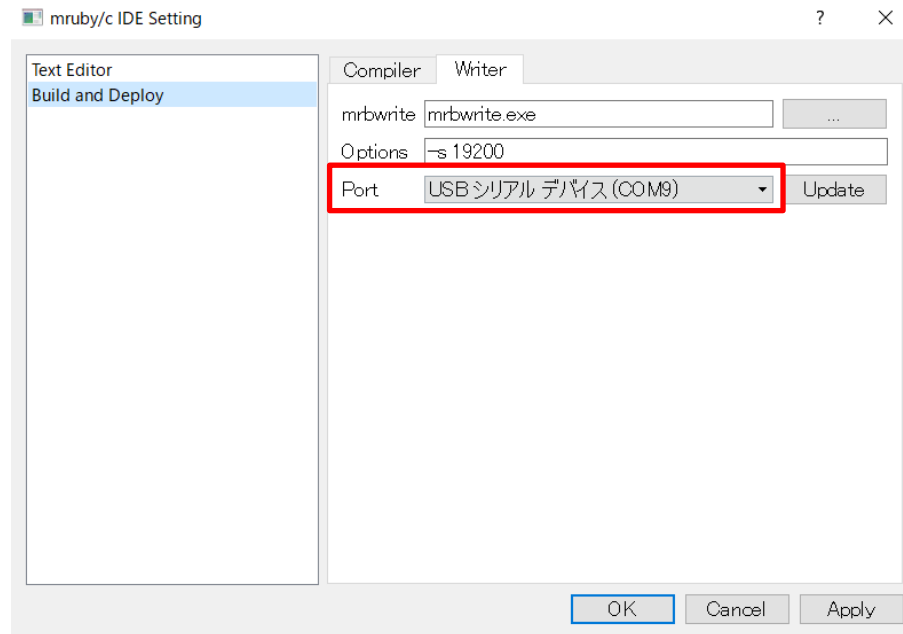
- 「mrubyc_ide」を使う
 - 以下に、mrubyc_ide を置いています

https://github.com/mruby-lab/mrubyc_ide

- ダウンロード後、展開してください
(展開先はデスクトップなど)

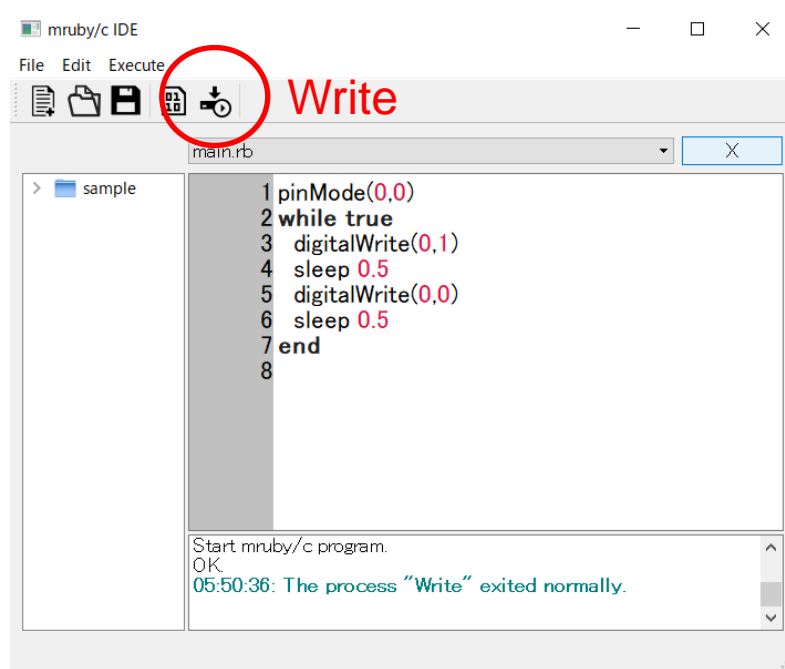
環境設定

- 「mrubyc_ide.exe」を起動
- [File]-[Settings]で Build and Deploy を選択
- 「Writer」タブを選択し、「Port」を設定



簡単なプログラムを実行

- [File]-[Open] で sample を開く
- Write ボタンをクリックして、マイコンボードの RS ボタンを押す

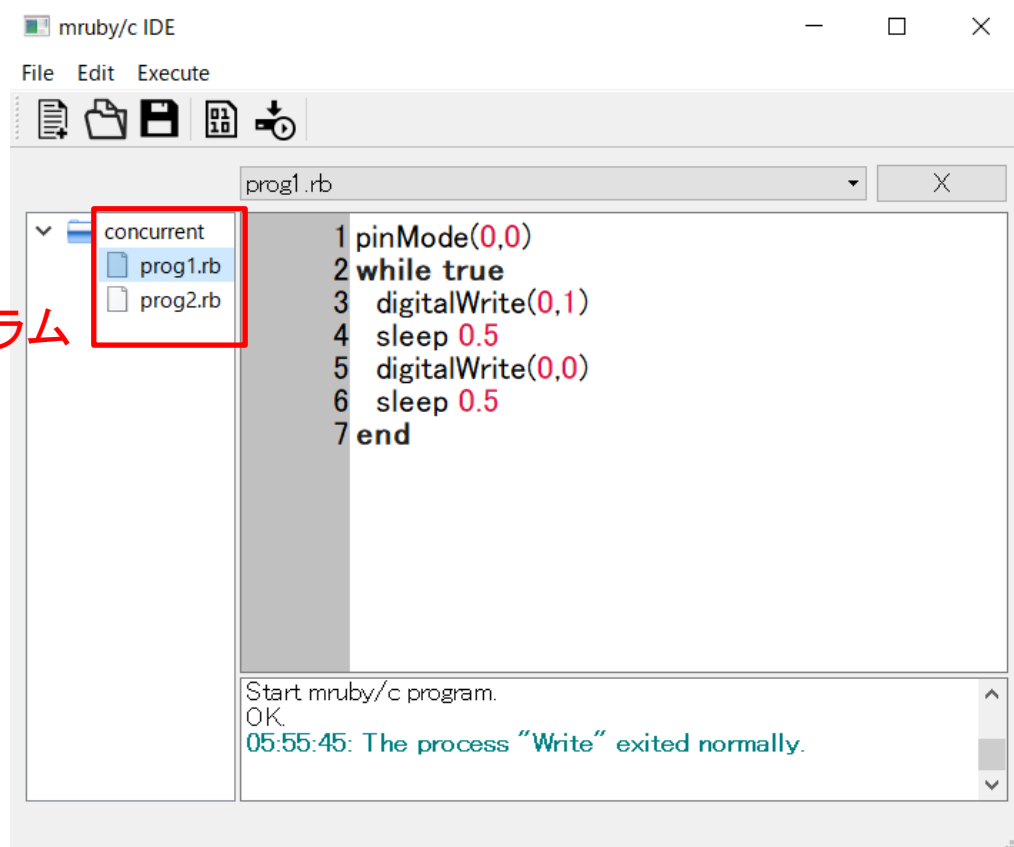


演習

コンカレントプログラム

- [File]-[Open] で concurrent を開く

複数のプログラム



演習

ブレッドボード、部品も使って構いません

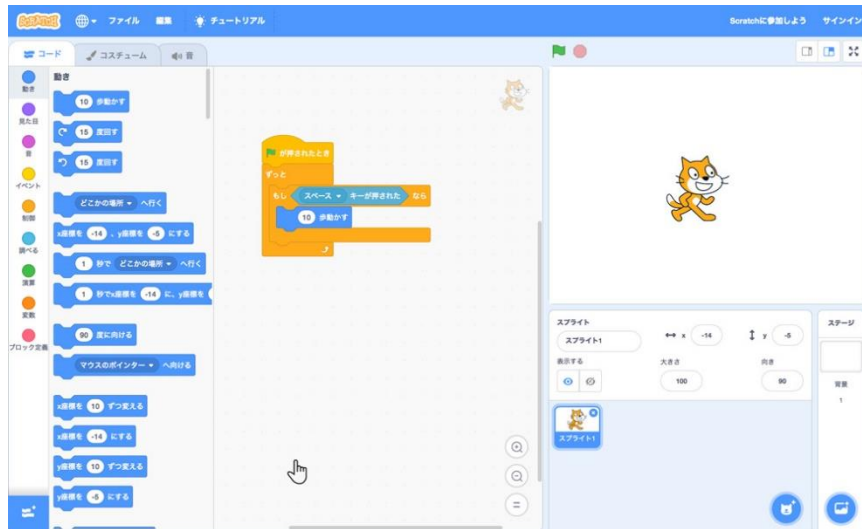
データフロープログラミング

ソフトウェア開発

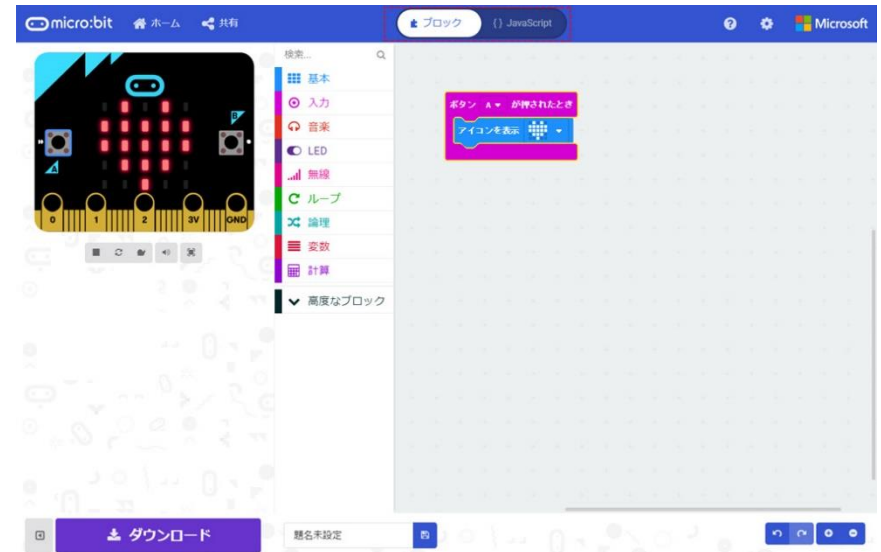
- エディタでプログラムコードを書く
- ビジュアルプログラミング環境を使う
 - プログラミング教育でも使われている

ビジュアルプログラミング ブロックを使うタイプ

Scratch



micro:bit



ビジュアルプログラミング フローを使うタイプ

Node-RED



MESH



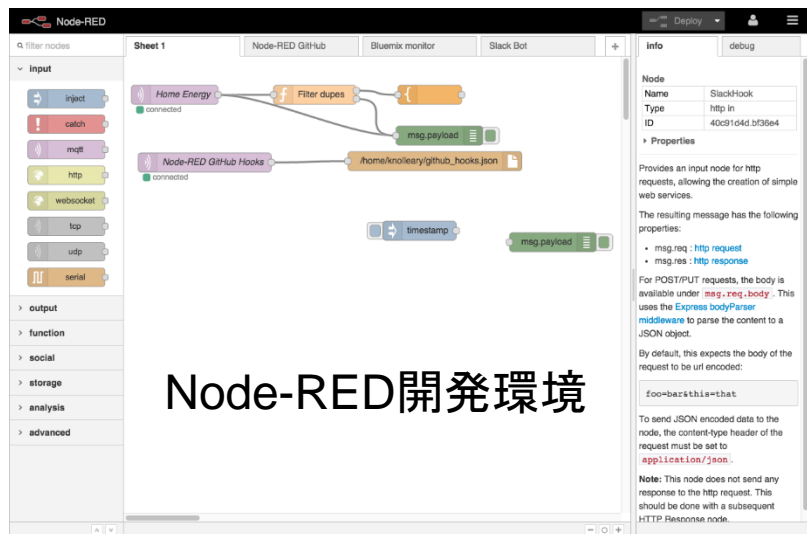
2つのプログラミング環境

- ブロックを使うタイプ
 - プログラムロジックの習得に役立つ
- フローを使うタイプ
 - データフローの理解に役立つ

IoTソフトウェアの開発環境

- IoTは、データの流が重要
 - 例えば、「センサの取得→エッジでの処理→クラウドへ送信」
- マイコンプログラムにも応用できないか？
 - 入力→処理→出力

Node-REDによるソフトウェア開発



Node-RED開発環境

- 各ノードの実行は非同期
- 複数のフローの並行処理

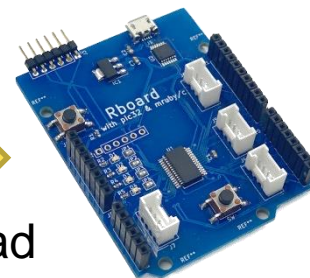


データフロー
(JSON)

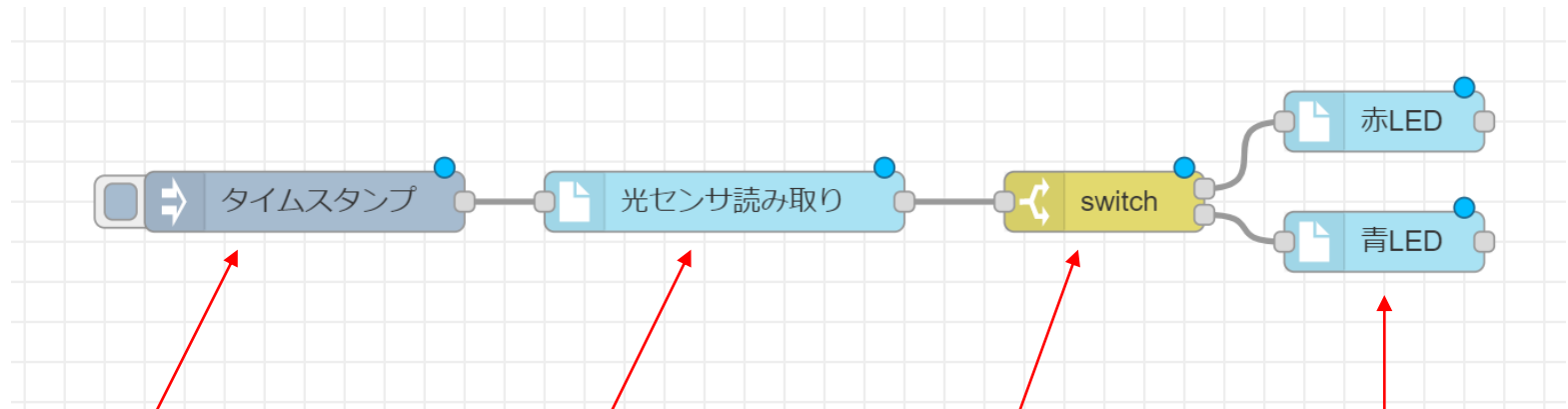
自動生成

mruby/c
プログラム

Download



データフローによる開発例



一定時間ごとに、明るさセンサを取得し、条件によって、LEDを点灯させる



mruby/cのプログラムを自動生成
ワンチップマイコンで実行する

データフロー演習

データフローを使ってみる

- 「Node-RED開発環境」にアクセスする

`https://github.com/mruby-lab/mrubyc_ide`

- `mrubyc_ide` で `sample` を開いておいてください

Node-REDの説明

Node-RED

デプロイ

ノードを検索

フロー 1

yaml

ストレージ

write file

read file

watch

mruby Rboard Nodes

Constant

Button

LED

GPIO - Read

GPIO - Write

I2C

function - ruby

タイムスタンプ

LED

情報

ノードを検索

フロー

フロー 1

サブフロー

グローバル設定ノード

フロー 1

フロー "b9e016343918a554"

ctrl1-[や ctrl1-] で、タブの切り替えができます。

ノード
(ダブルクリックで機能を設定できる)

Rboardのノード


ノードの設定

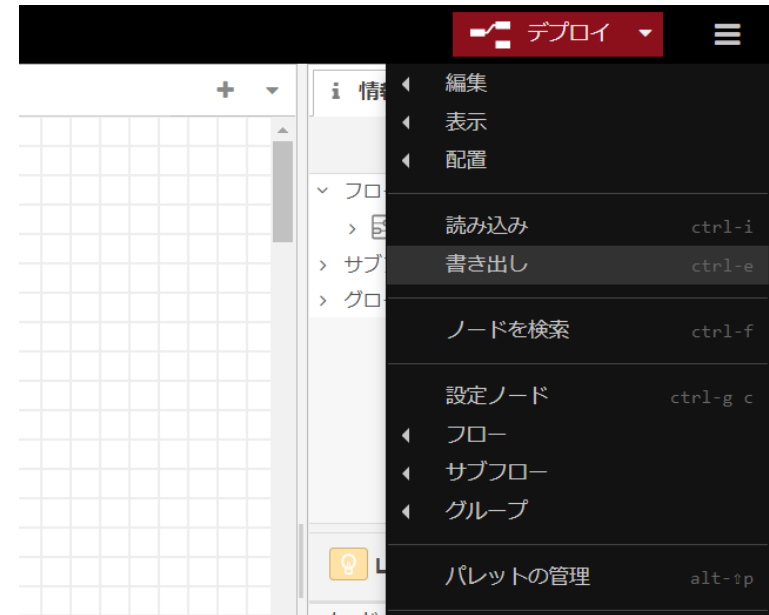
- タイムスタンプ (inject) の設定
 - 「繰り返し」を「指定した時間間隔」
 - 「時間間隔」を「1秒」
- LEDの設定
 - 「制御対象」を「Pinと接続しているLED」
 - 「Pin番号」を「0」
 - 「LEDのON/OFFモード」を「入力に従ってONとOFFを切り替える」

mrubyコードの生成

- Node-REDで作成したプログラムは「ノードの接続情報」
- 「ノードの接続情報」 → 「mruby/cプログラム」の変換が必要

接続情報を取り出す

- 右端の  アイコンで、書き出しを選択
- 「書き出し」で
クリップボードに
コピーされます



「書き出し」の注意点



クリップボードにコピーされる

mruby/cコード生成

- 「mruby/cプログラム生成」に貼り付け
- 「rbファイルを生成する」をクリック

mruby/cプログラム生成

Node-REDで「書き出し」をして、ペーストしてください

ここに貼り付け

JSONデータ・

rbファイルを生成する

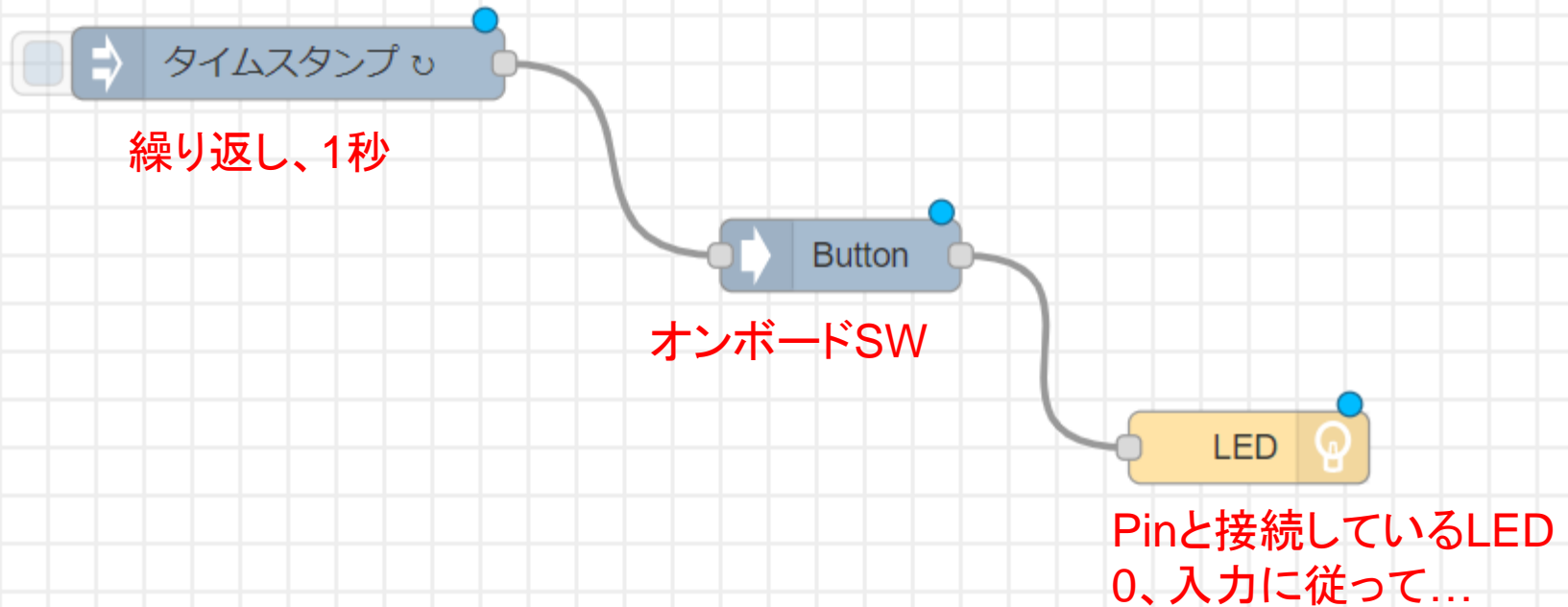
mruby/cコードを生成する

mruby/cコード生成

- 「rbファイルを生成する」をクリック
 - ブラウザに生成された mruby/c プログラムが表示される
 - 全てを選択してコピーする
- mrubyc_ide の sample に貼り付けて実行

演習

プログラム例



プログラム例

