

軽量 Ruby (mruby/c) でマイコンプログラミング

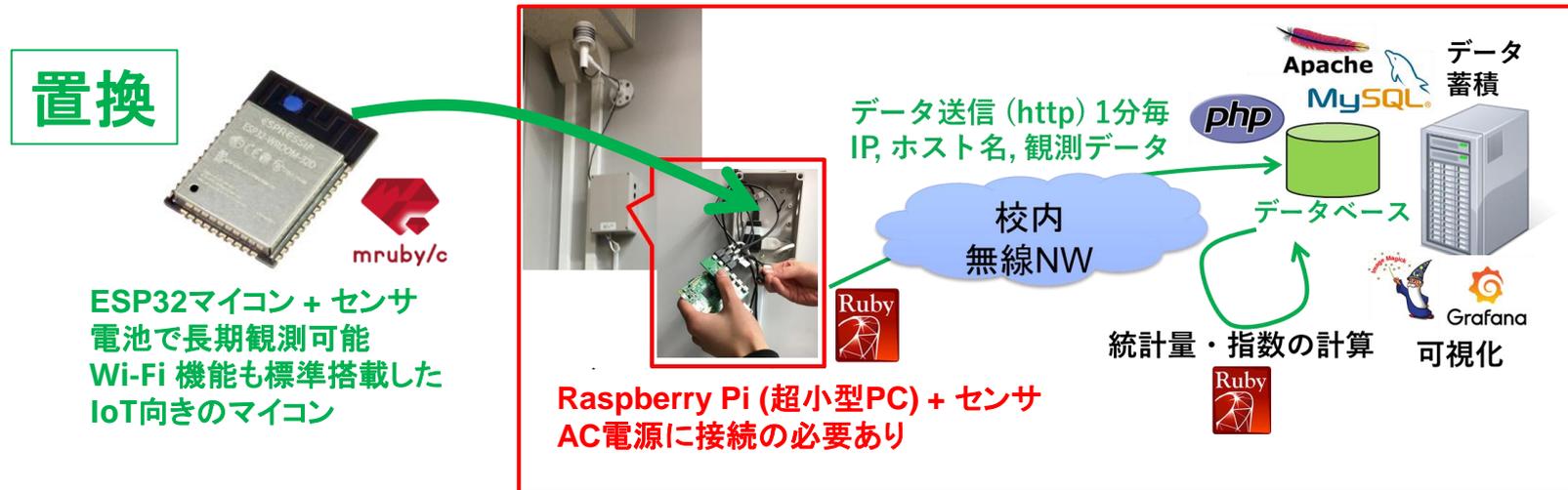
松江工業高等専門学校 情報工学科

杉山 耕一郎

2023年03月20日

杉山の関わり

- mruby/c に着目したきっかけ
 - コミュニティからの声掛け
 - マイコンを用いた IoT システム・IoT 演習の模索 (2018年末)
 - Raspberry Pi (超小型PC) での長期観測には AC 電源が必要



mruby/c チュートリアル の 衝撃 (2018年度末)



ITOC とは 支援事業 セミナー・イベント レポート お問い合わせ

ESP32×mruby/c IoTハンズオン

2019/03/16

トップ / 支援事業 / 先端技術支援 / mruby/c / チュートリアル / ESP32×mruby/c IoTハンズオン / ハンズオン - 3

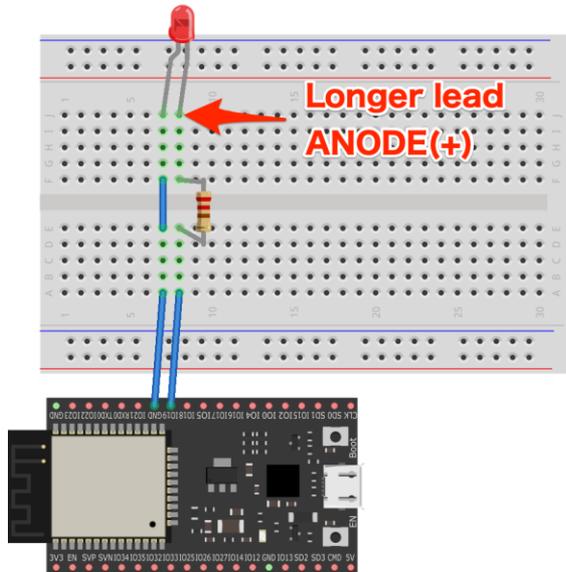
ハンズオン - 3

Lチカ(発光ダイオード点滅)

マイコン界におけるLチカは、ソフトウェア界におけるHello Worldのようなものです。LEDを光らせることができれば、あなたも立派なマイコン刑事(デカ)です。

ESP32×mruby/c IoTハンズオン

- ESP32 + mruby/c開発のための第1章 - 導入
- ESP32 + mruby/c開発のための第2章 - macOS
- ESP32 + mruby/c開発のための第3章 -



main/main.c

```
#include "driver/gpio.h"
#include "mrubyc.h"
#include "models/led.h"
#include "loops/master.h"

#define MEMORY_SIZE (1024*40)
static uint8_t memory_pool[MEMORY_SIZE];

static void c_gpio_init_output(mrb_vm *vm, mrb_value *v, int argc) {
    int pin = GET_INT_ARG(1);
    console_printf("init pin %d\n", pin);
    gpio_set_direction(pin, GPIO_MODE_OUTPUT);
}

static void c_gpio_set_level(mrb_vm *vm, mrb_value *v, int argc){
    int pin = GET_INT_ARG(1);
    int level = GET_INT_ARG(2);
    gpio_set_level(pin, level);
}

void app_main(void) {
    mrbc_init(memory_pool, MEMORY_SIZE);

    mrbc_define_method(0, mrbc_class_object, "gpio_init_output", c_gpio_init_output);
    mrbc_define_method(0, mrbc_class_object, "gpio_set_level", c_gpio_set_level);

    mrbc_create_task( led, 0 );
    mrbc_create_task( master, 0 );
    mrbc_run();
}
```

mrblib/loops/master.rb

```
led = Led.new(18)

while true
  led.turn_on
  sleep 1
  led.turn_off
  sleep 1
end
```

mrblib/models/led.rb

```
class Led
  def initialize(pin)
    @pin = pin
    gpio_init_output(@pin)
    turn_off
  end

  def turn_on
    gpio_set_level(@pin, 1)
    puts "turned on"
  end

  def turn_off
    gpio_set_level(@pin, 0)
    puts "turned off"
  end
end
```

C から書くの?!

自分でライブラリを書いてみるか

- Rubyだけ書けば良い状態にしたい
→ mruby/c コミュニティへ参加

C 言語 (現在の主流)

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"

#define BLINK_GPIO 13

void app_main(void) {
    gpio_pad_select_gpio(BLINK_GPIO);
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);

    while(1) {
        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

MicroPython

```
from machine import Pin
from time import sleep

led = Pin(13, Pin.OUT)

while true:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

mruby/c

```
led = Pin.new(13, GPIO::OUT)

while true
  led.on
  sleep 1
  led.off
  sleep 1
end
```

こう書きたい
自分で Pin クラスを作るか

mruby/c ライブラリの作成

- 公式開発環境 (C 言語) を mruby/c でラップ

mrblib/loops/master.rb mrblib/models/led.rb main/main.c

メインプログラム(mruby/c) mruby/c のクラス C言語のプログラム
～ESP32マイコン用の公式
開発環境の関数を利用～

```
led = Led.new(19)

while true
  led.turn_on
  sleep 1
  led.turn_off
  sleep 1
end
```

Ruby のクラス・メソッドを呼ぶ

```
class Led
  def initialize(pin)
    @pin = pin
    gpio_init_output(@pin)
    turn_off
  end

  def turn_on
    gpio_set_level(@pin, 1)
    puts "turned on"
  end

  def turn_off
    gpio_set_level(@pin, 0)
    puts "turned off"
  end
end
```

C 言語の関数を呼ぶ

```
#include "driver/gpio.h"
#include "mrubyc.h"
#include "models/led.h"
#include "loops/master.h"

#define MEMORY_SIZE (1024*40)

static uint8_t memory_pool[MEMORY_SIZE];

static void c_gpio_init_output(mrb_vm *vm, mrb_value *v, int argc) {
  int pin = GET_INT_ARG(1);
  console_printf("init pin %d\n", pin);
  gpio_set_direction(pin, GPIO_MODE_OUTPUT);
}

static void c_gpio_set_level(mrb_vm *vm, mrb_value *v, int argc){
  int pin = GET_INT_ARG(1);
  int level = GET_INT_ARG(2);
  gpio_set_level(pin, level);
}

void app_main(void) {
  mrbc_init(memory_pool, MEMORY_SIZE);

  mrbc_define_method(0, mrbc_class_object, "gpio_init_output", c_gpio_init_output);
  mrbc_define_method(0, mrbc_class_object, "gpio_set_level", c_gpio_set_level);

  mrbc_create_task( led, 0 );
  mrbc_create_task( master, 0 );
  mrbc_run();
}
```

関数化

Cの関数を Ruby から呼ぶための設定

Ruby のファイル名を教える

mruby/c
C 言語
アセンブラ (機械語)
ハードウェア(マイコン)

ここをライブラリとして自製する

閑話休題：MicroPython

- 公開されているツールの完成度：MicroPython >>>> mruby/c
- センサメーカーは，Arduino や MicroPython のサンプルコードを提供するのが一般的。
- MicroPython を使えばよい？
 - 目標達成のために必要な機能は，どんな言語を使う場合であれ，自分で作らねばならない
 - 開発しやすい・好みな言語を使う方が効率的。
- mruby/c で教材開発する利点・意義
 - 地域に開発者がいる．日本語で相談できる
 - 地域の人が応援してくれる・勝手に宣伝してくれる
 - 学生の卒業論文テーマとして扱いやすい

共同研究の例(2019年度~)

• mruby/c を活用した授業実践

- 杉山ほか：高専の授業にmruby/cを取り入れたプロジェクトの始動と将来展望 (RubyWorldConference 2019)

共同研究の内容

- mruby/c を高専の新規開講授業(2020年度)への組み込み
 - ✓ ESP32 マイコンの利用を想定
 - ✓ 松江高専情報工学科 5 年生「組込システム演習」(2 コマ 180 分 × 16 回)
- ESP32マイコン用mruby/c ライブラリ開発

研究体制

- 松江市 まつえ産業支援センター
- 島根県 しまねソフト研究開発センター (ITOC)
- 株式会社 モンスターラボ (2019年度)
- Rubyプログラミング少年団 (2021年度~) ← 卒業生
- 株式会社 CMCソリューションズ (2021年度~) ← 卒業生

取組実績

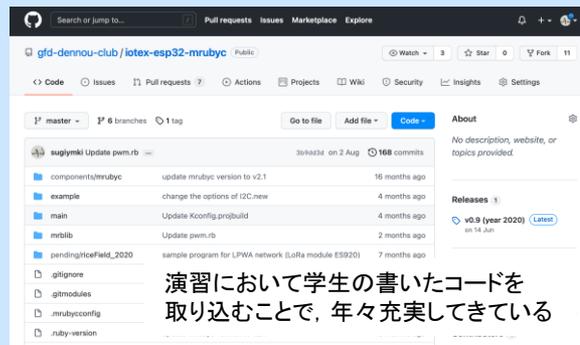
学習ボード



パーツ	Pin など
LED	GPIO13, 12, 14, 27, 26, 25, 33, 32
スイッチ	GPIO34, 35, 18, 19
サーモスタ温度計	GPIO39 (ADC1_CH3)
圧電マイク	GPIO15
ESP (SYSTEM_MAX)	GPIO16 (TX), GPIO17 (RX) (電源 GPIO5)
LCD (AQM0802A-RH-GBW)	I2C (SDA: GPIO21, SCL: GPIO22, アドレス: 0x3E)
RTC (RC-8035SA)	I2C (SDA: GPIO21, SCL: GPIO22, アドレス: 0x32) (電源 GPIO4)
SD card	SPI (MOSI: GPIO23, MISO: GPIO19, SCK: GPIO18, CS: GPIO2)
SHT75	COM1: GPIO25, 26, COM2: GPIO32, 33 [LED と共有]

教科書の整備

- サンプルコードも「教科書」
 - ✓ 教科書作りとしてのライブラリ開発



高専の授業「組込システム演習」

- Arduino, C 言語(ESP-IDF), micropython, mruby/c の各種言語を用いたプログラミング技法の修得
- センサ利用のためのライブラリを mruby/c で書く
 - ✓ C 言語のラッパー作製, Ruby のクラス定義
 - ✓ センサのデータシートを読めるようになる
- Linux でテキストベースのプログラミング



共同研究の例(2019年度~)

● スモウルビー + mruby/c → 初等教育用教材

- 青笹ほか：IoTデバイス用ビジュアルプログラミングツール「SmT」の開発
～地域コミュニティと連携した人材育成の成果報告～ (RubyWorldConference 2020)

共同研究の内容

- プログラミング学習で利用可能なIoT教材の開発
 - ブロックから mruby/c のコード生成
 - クリック一つでマイコンボードにプログラムの書き込み
- 小中学生が IoT を学ぶためのツールとして活用できるかの有効性を検証

研究体制

- 松江市まつえ産業支援センター
- 島根県 しまねソフト研究開発センター (ITOC)
- Rubyプログラミング少年団 ← 卒業生
- 株式会社 島根情報処理センター (2020年度~)
- 株式会社 CMCソリューションズ (2020年度~) ← 卒業生
- サーバートラスト 株式会社 (2021年度~) ← 卒業生

取組実績

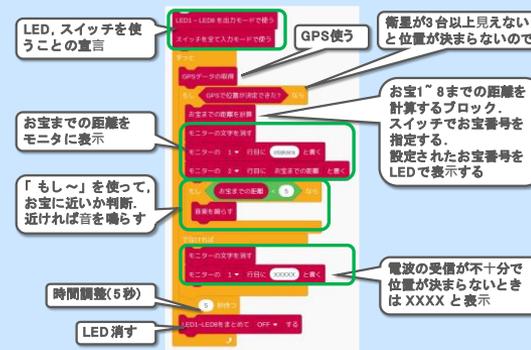
SmT : スモウルビー + 高専製の拡張機能

- ブロックとmruby/cのソースコードとの相互変換機能およびソースコードのコンパイルとRBoardへの書き込み機能を実装



小中学生向け IoT 講座

- 換気センサ (CO₂センサ), 宝探しセンサ (GPS), などのテーマ作成
- IoT教材の有用性の検証



Matz葉がにロボコン

Matz葉がに
ロボコン



軽量 Ruby 言語 mruby/c を用いた IoT プログラミングツールの実証実験フィールドとしての位置づけ。
→ 多くの方の協力を得て「ご当地ロボコン」へ。2023/01/15 のプレ大会には 23 チーム参加



NHK



山陰中央新報



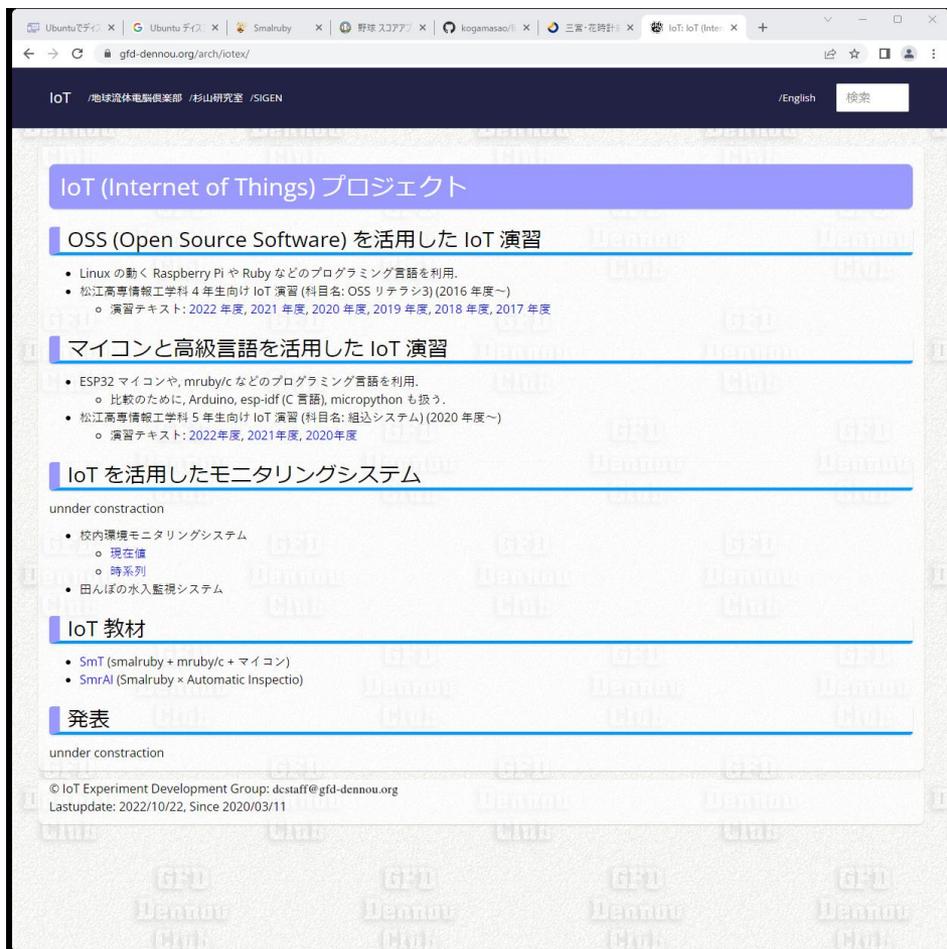
朝日新聞



- ・主催：松江工業高等専門学校
- ・共催：しまねOSS協議会、松江市
- ・後援：島根県
- ・協力：福井県子どもプログラミング協議会
JAXA 宇宙科学研究所 Planet-C プロジェクト
かにロボ連盟：越前がにロボコン

開発している教科書・ライブラリ

- 電脳倶楽部の Web, GitHub にたくさん置いてます



The screenshot shows the website <https://www.gfd-dennou.org/arch/iotex/>. The page is titled "IoT (Internet of Things) プロジェクト" and lists several projects and resources:

- OSS (Open Source Software) を活用した IoT 演習**
 - Linux の動く Raspberry Pi や Ruby などのプログラミング言語を利用。
 - 松江高専情報工学科 4 年生向け IoT 演習 (科目名: OSS リテラシ3) (2016 年度~)
 - 演習テキスト: 2022 年度, 2021 年度, 2020 年度, 2019 年度, 2018 年度, 2017 年度
- マイコンと高級言語を活用した IoT 演習**
 - ESP32 マイコンや, mruby/c などのプログラミング言語を利用。
 - 比較のために, Arduino, esp-idf (C 言語), micropython も扱う。
 - 松江高専情報工学科 5 年生向け IoT 演習 (科目名: 組込システム) (2020 年度~)
 - 演習テキスト: 2022 年度, 2021 年度, 2020 年度
- IoT を活用したモニタリングシステム**

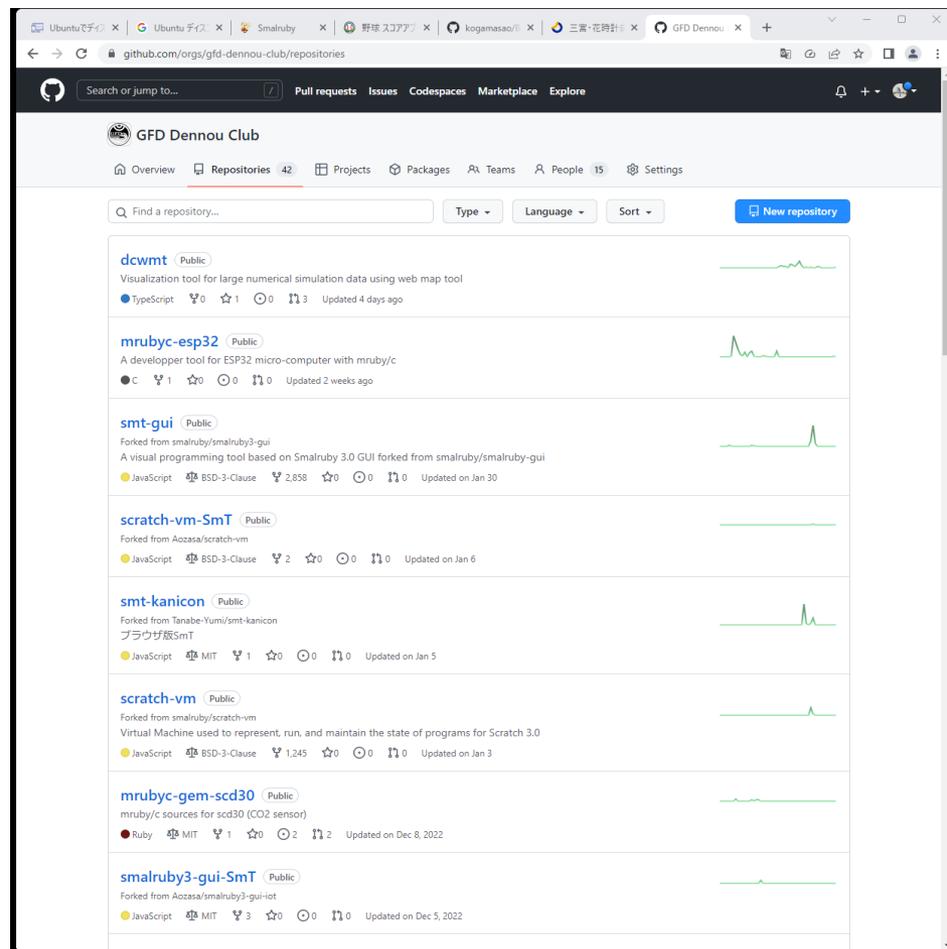
under construction

 - 校内環境モニタリングシステム
 - 現在値
 - 時系列
 - 田んぼの水入監視システム
- IoT 教材**
 - Smt (smalruby + mruby/c + マイコン)
 - SmrAI (Smalruby × Automatic Inspectio)
- 発表**

under construction

© IoT Experiment Development Group: dcstaff@gfd-dennou.org
Lastupdate: 2022/10/22. Since 2020/03/11

<https://www.gfd-dennou.org/arch/iotex/>

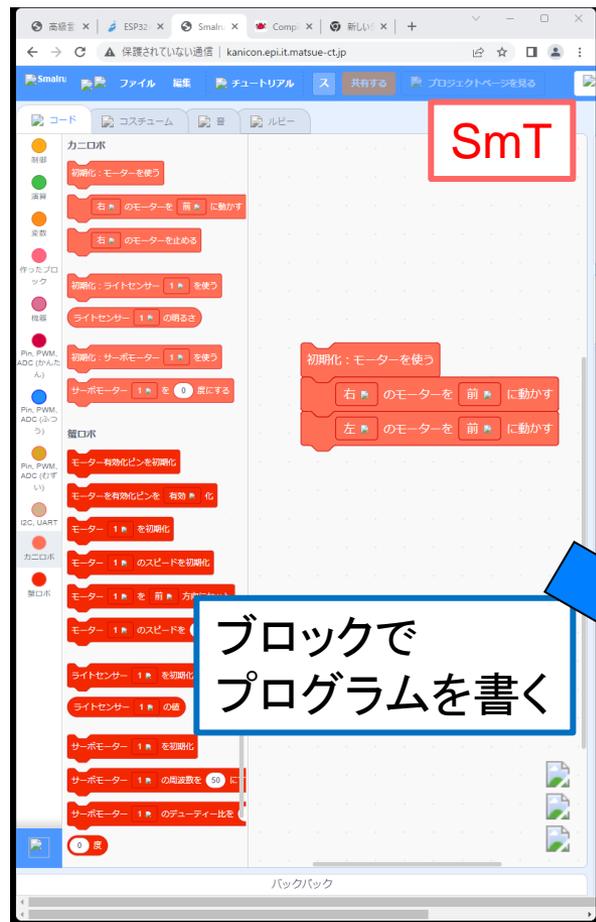


The screenshot shows the GitHub repository page for <https://github.com/gfd-dennou-club/repositories>. The page displays a list of repositories:

- dcwmt** (Public) - Visualization tool for large numerical simulation data using web map tool. Updated 4 days ago.
- mruby-esp32** (Public) - A developer tool for ESP32 micro-computer with mruby/c. Updated 2 weeks ago.
- smt-gui** (Public) - Forked from smalruby/smalruby3-gui. A visual programming tool based on Smalruby 3.0 GUI forked from smalruby/smalruby-gui. Updated on Jan 30.
- scratch-vm-SmT** (Public) - Forked from Aozasa/scratch-vm. Updated on Jan 6.
- smt-kanicon** (Public) - Forked from Tanabe-Yumi/smt-kanicon. ブラウザ版SmT. Updated on Jan 5.
- scratch-vm** (Public) - Forked from smalruby/scratch-vm. Virtual Machine used to represent, run, and maintain the state of programs for Scratch 3.0. Updated on Jan 3.
- mruby-gem-scd30** (Public) - mruby/c sources for scd30 (CO2 sensor). Updated on Dec 8, 2022.
- smalruby3-gui-SmT** (Public) - Forked from Aozasa/smalruby3-gui-iot. Updated on Dec 5, 2022.

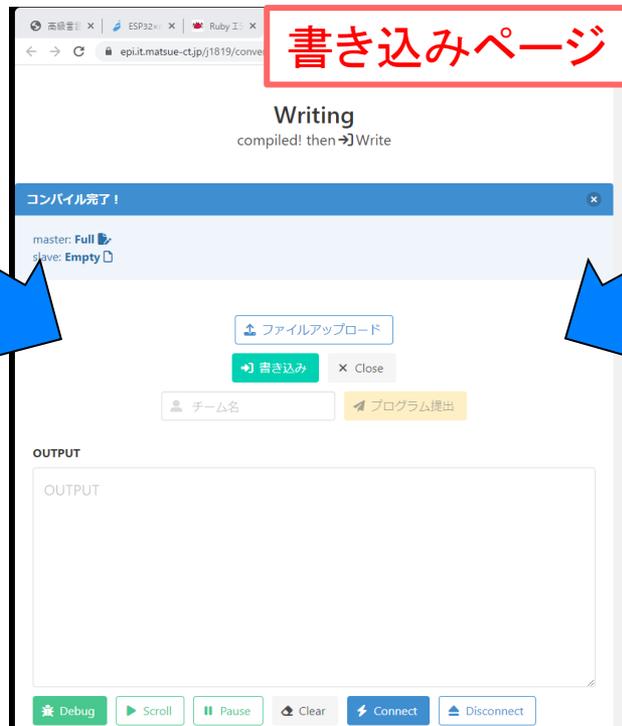
<https://github.com/gfd-dennou-club/>

ブラウザツールの開発も



ブロックで
プログラムを書く

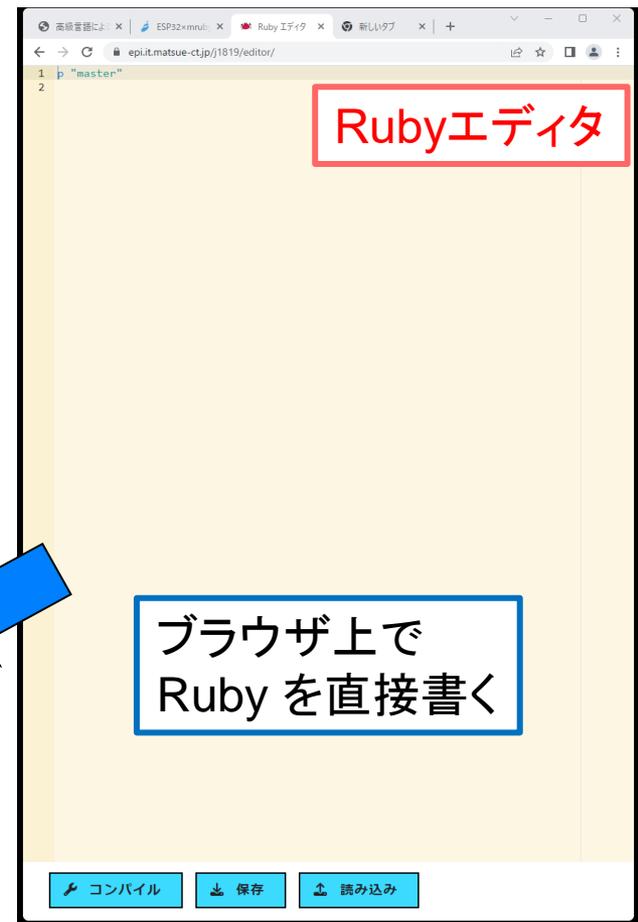
<http://kanicon.epi.it.matsue-ct.jp/>



書き込みページ

ボタンをクリックするだけでマイコン
にプログラムが書き込まれる

<https://www.epi.it.matsue-ct.jp/j1819/convert/upload.php>



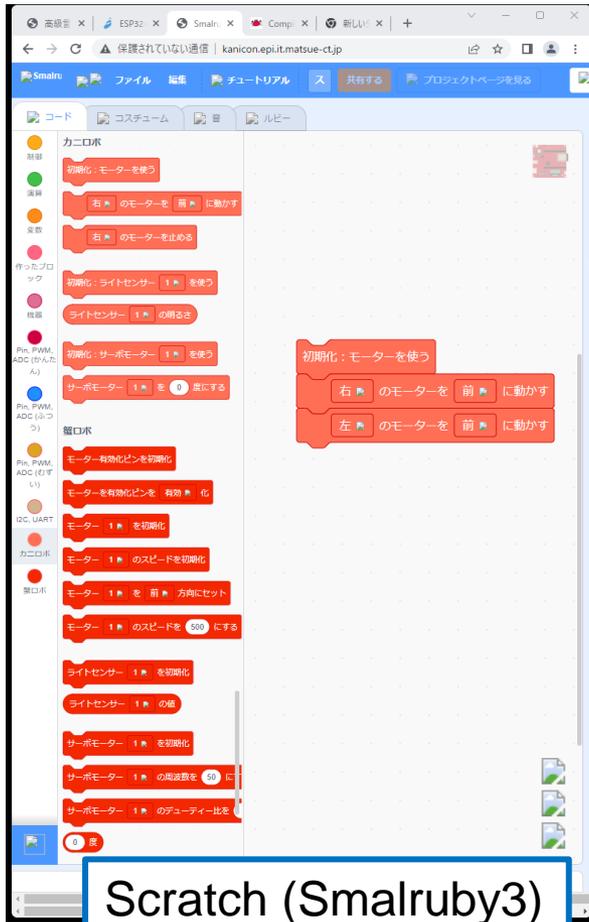
Rubyエディタ

ブラウザ上で
Ruby を直接書く

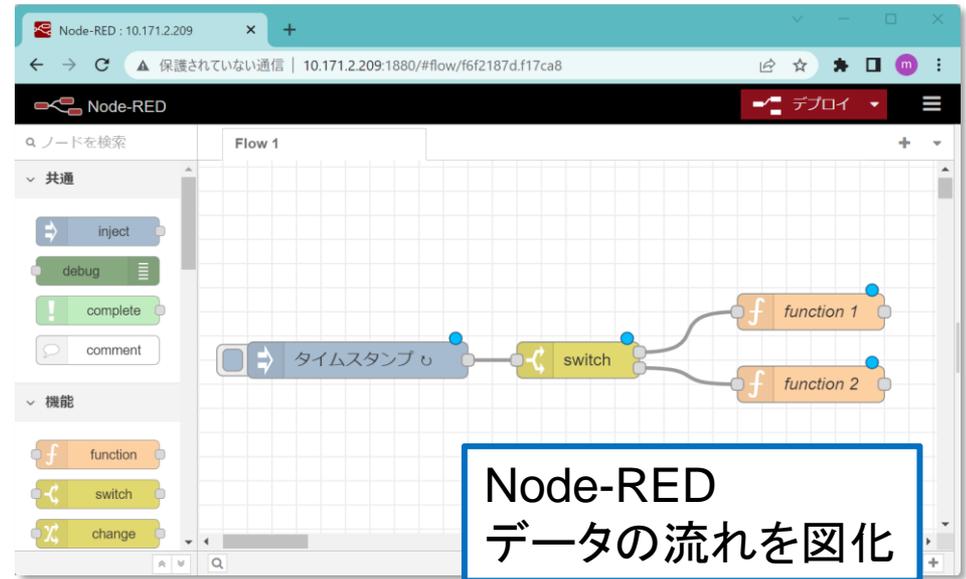
<https://www.epi.it.matsue-ct.jp/j1819/convert/>

ブラウザツールの開発も

- ビジュアルプログラミングツールは2種類



Scratch (Smalruby3)
論理構造を図化



Node-RED
データの流れを図化

まとめ

- mruby/c ライブラリ開発・ツール開発を進めている
 - よければ使ってみてください
 - 一緒に開発しませんか？
- 地球惑星科学的な展開を模索したい
 - アイデア・コメントを頂けると喜びます