

# 組み込みの課題を乗り越えて:

企業ヒアリングで見えてきた現実と  
可能性を解き放つ

mruby, mruby/cの新しいAPIリファレンス

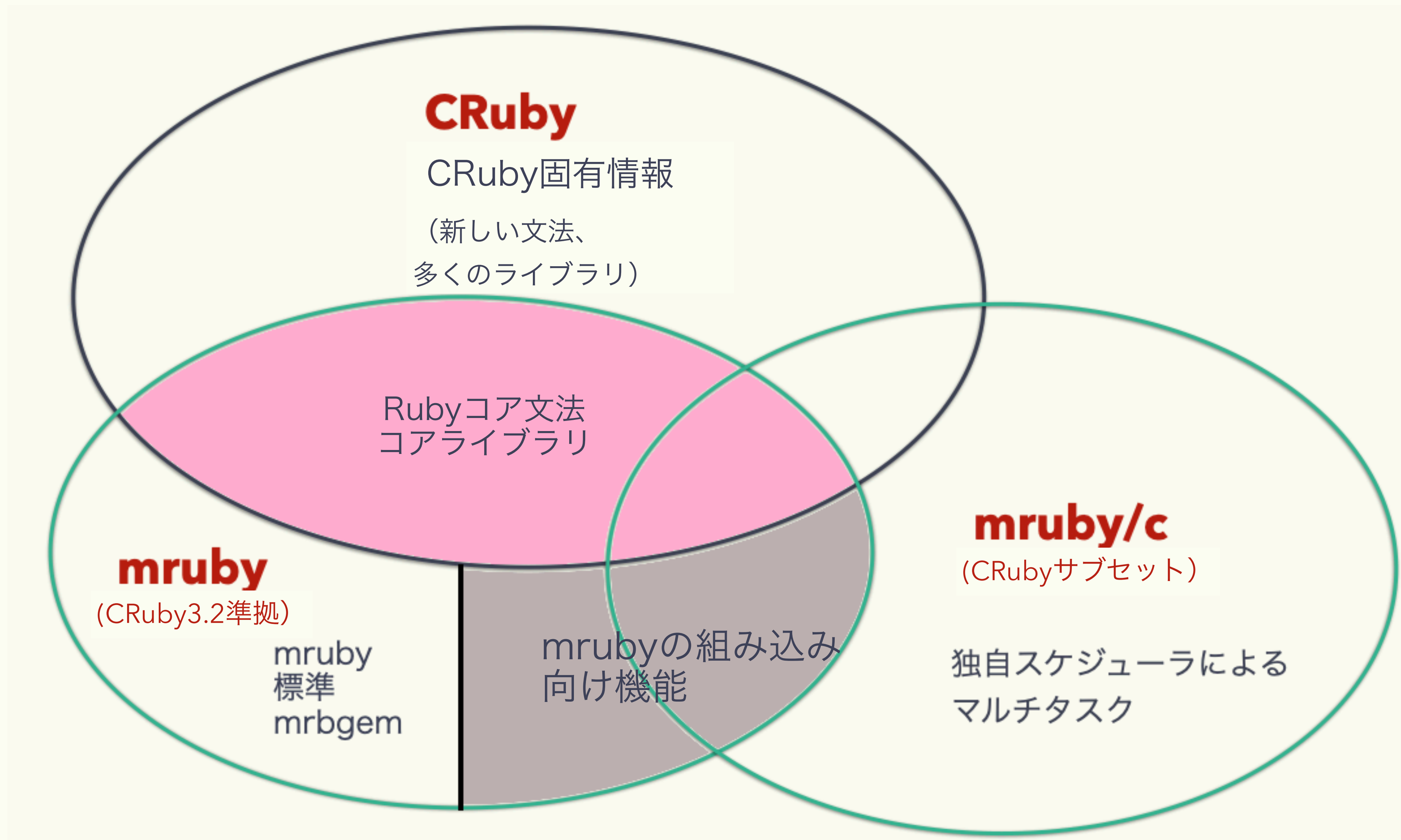


軽量Rubyフォーラム  
石井 宏昌

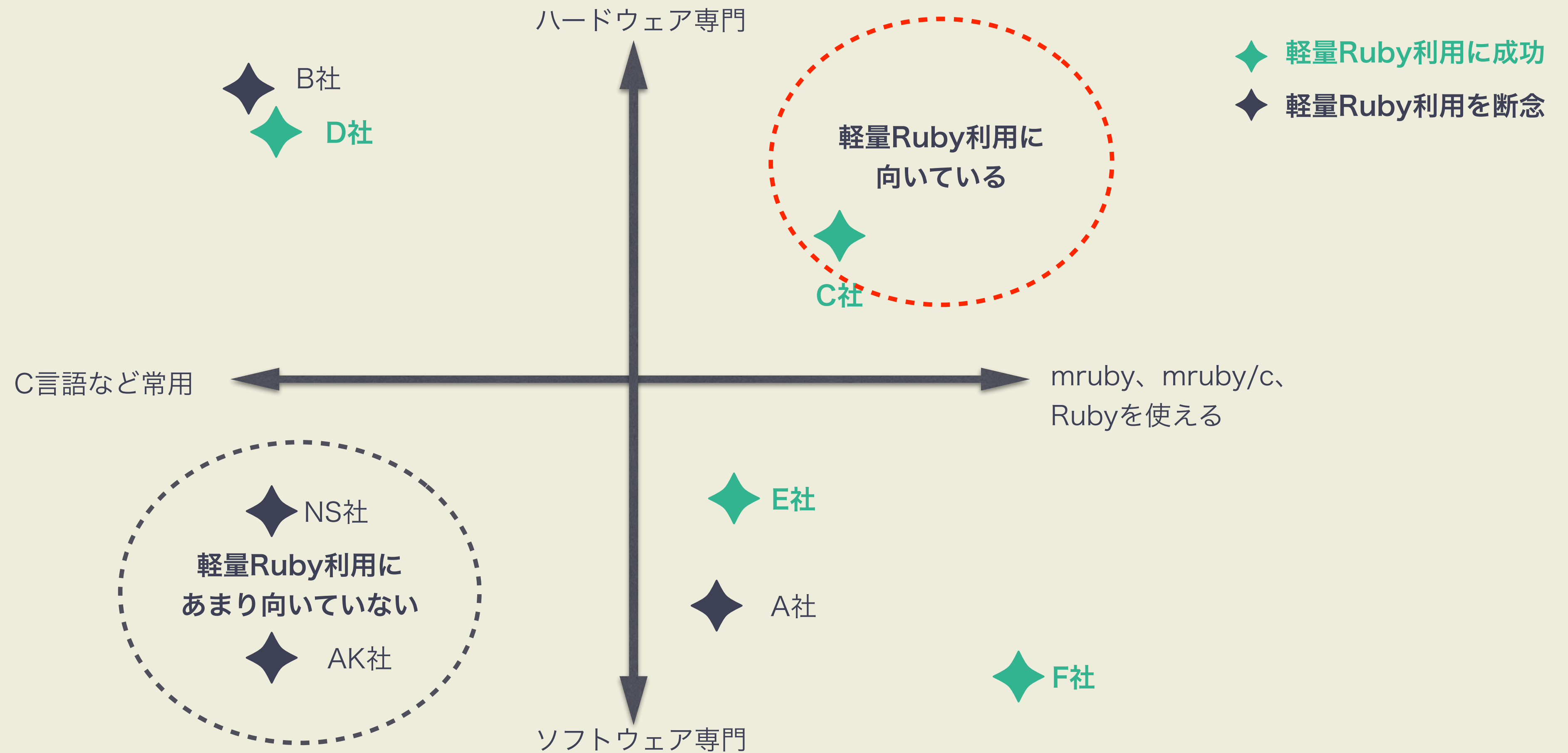


ITOC しまねソフト研究開発センター  
東 裕人

# Ruby Everywhere



# 使ってどうだったのか聞いてみた





# ・ A社 ソフトウェア会社のIoT挑戦

- ・ C言語からJava, Swift, Kotlin, Rubyなど幅広く対応
- ・ 社長はRubyistでRubyの良さを知っていた。  
そこで自社のアイデアの製品化に際しmrubyを使用しようと企画。
- ・ ハードウェアに関しては知見が無く外部委託。
  - ✓ 基板開発をハードウェア会社に委託
  - ✓ 当初ハード会社も軽量Ruby使用に同意
  - ✓ 進めていくうちに徐々にハード会社から不満が

- ・ 結局押し切られC言語で製品化

あちゃー



## ・ B社 機械製造会社のAI挑戦

- ・ 自社製品にAIを組み込み自動判別機能を追加しようと企画。
- ・ PRJ責任者がAIやmrubyなどの新しい取り組みに積極的。
- ✓ CやPLCでの開発専門だったためアプリ開発をmrubyで依頼。
- ✓ 依頼先もC専門 当初mrubyとRubyの区別がついていない状態。
- ✓ 一から進めていくにはハードルが高すぎた。

・ **結局mrubyでは作りきれず断念**

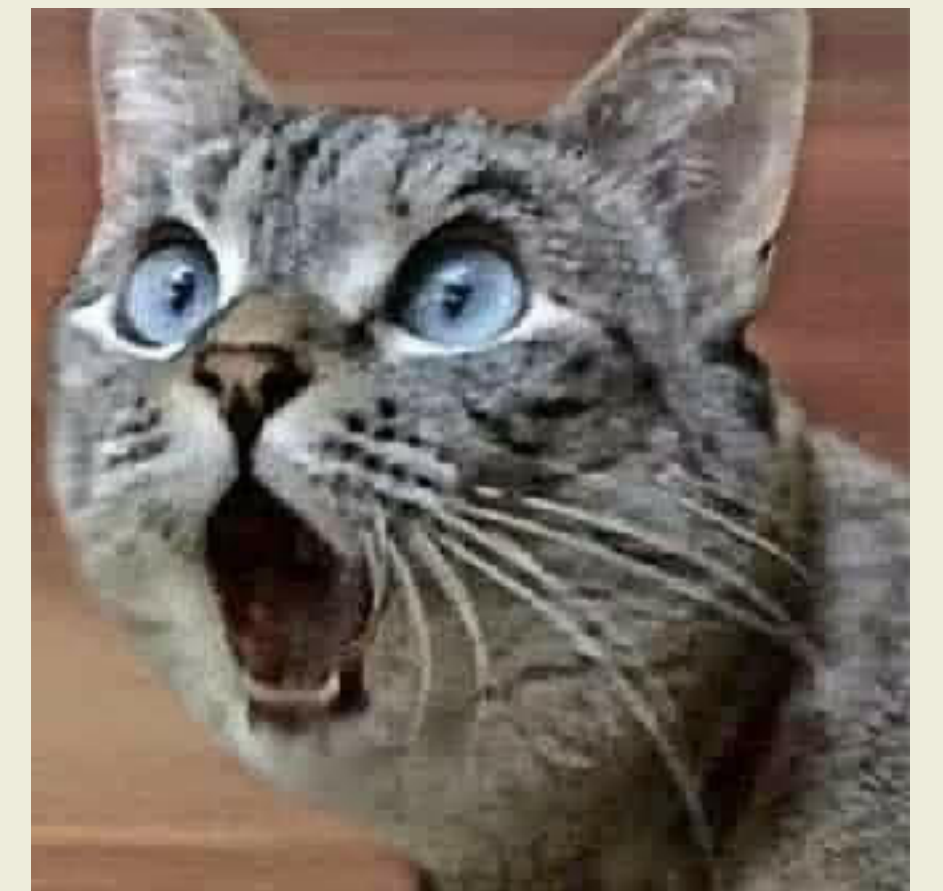


# ・ C社 若手が軽量Rubyで開発・製品化

以前よりサイネージなどの製品を発表。既存マイコンボードを応用したLPWA通信による災害対策システムをベテランC開発者のリーダーとmruby/cを初めて触る若手が開発。

- ✓ 軽量Ruby選択については社内では問題なく許可
- ✓ 基本的通信部分はCで、その他の付加価値部分をmruby/cで開発
- ✓ 適切な外部サポートがあり、“楽しく”開発していた。

・ **短期間（数ヶ月）で製品化実現！**





# ・D社 ハードは外注、ソフトは軽量Rubyで内製

- ・ソフトウェア開発会社 ハードウェアの知識がある上にCやmrubyも経験があった。
- ・センサー・通信部分の基板は連携したハードウェア会社が制作、アプリケーションはmruby, mruby/cで開発した。
- ✓ チップのIDEを使った通信部分の開発は流石に難しく、ハード会社のサポートを受けた。
- ✓ mrubyとmruby/cは機能に応じて使い分けた。
- ✓ GUIからアプリを生成する これはmrubyアプリだから可能

## ・理想的な形での製品化実現



## • 見えてきたこと


- まずはハードウェア知識
- 新しい言語を学ぶコストに見合うメリット

## • うまくいく秘訣

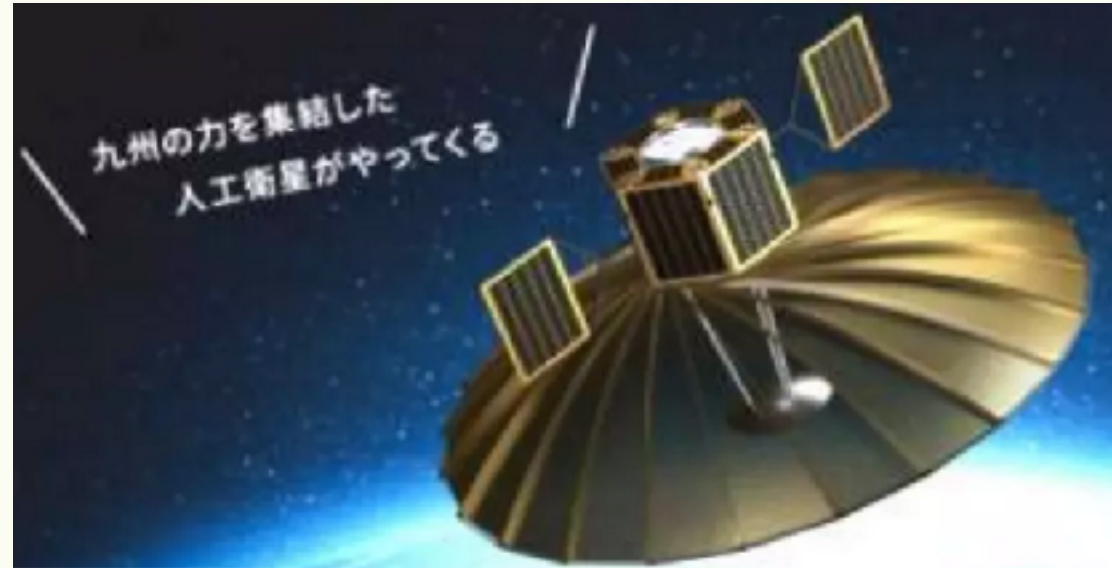
- すべてを軽量Rubyではなく一部からはじめる
- 既存組込み言語の苦手部分は軽量Rubyが得意なところ！



# 「ちよこつとRuby」 のすすめ

 ちよこつとRuby

# ・事例からも見える「ちょっとRuby」



## ・QPS研究所

人工衛星の撮影タイミング・スケジュールの頻繁な書き換え部分にmruby



## ・株式会社インターネットイニシアティブ

トラヒックに応じてConfigの書き換えやレシピフレームワークにmruby, 基本通信部分はC言語



## ・CMN株式会社

センサーの変更やデータフォーマットの遠隔からの変更  
にmruby/c, 基本通信部分はC言語

# いま揃うマイコン用のmrbgemとHAL



- ESP32
- ARM Cortex-A RZ/A1H (GR-Peach Renesas)
- nrf52 (Nordic Semiconductor)
- mruby-wiringpi



- ESP32
- PsoC5LP (Infineon Technologies)
- PIC24 (Microchip Technology)
- RX210 (Renesas)





ちよこっとRubyのサンプルなど公開していきます

# mruby, mruby/c の新しいAPIリファレンス

2023/11/10

RubyWorldConference2023

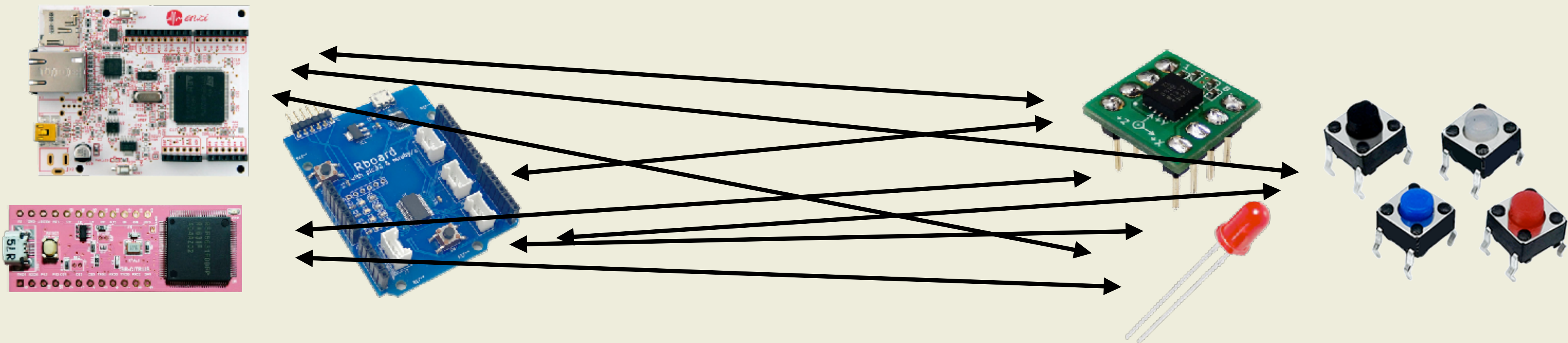
ITOC しまねソフト研究開発センター

専門研究員 東裕人

# 解決したい課題

mruby, mruby/c がだんだん使われるようになってきて

- ・ マイコンに接続するセンサー、LED、スイッチなど
- ・ 同じ物を扱うのでも、実装者によって方法がちまちま





# 共通化をめざす機能

## GPIO

汎用入出力



LED スイッチ

## ADC

Analog Digital Converter



電流センサ ジョイスティック

## PWM

パルス幅変調



サーボモータ 調光器

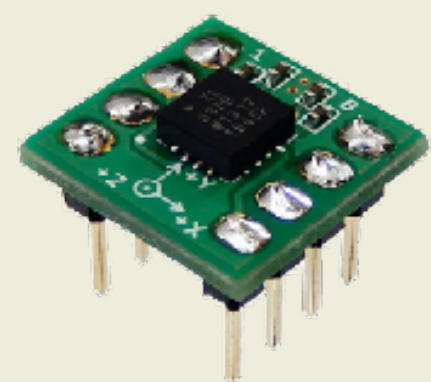
## UART

シリアル通信



通信モジュール シーケンサ(PLC)

## I2C



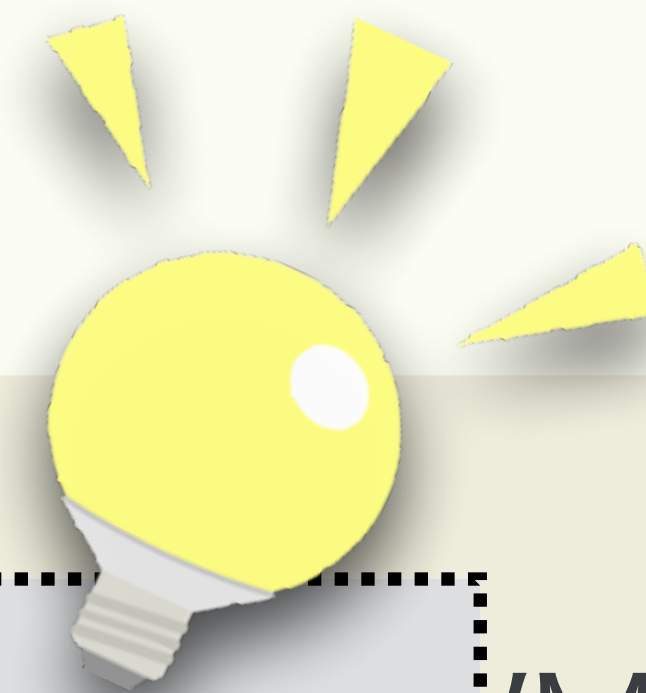
各種センサー

## SPI



表示装置

# 共通化をめざす機能



GPIO

汎用入出力



LED スイッチ

```
led1 = GPIO.new( 1 )  
led1.write( 1 )
```

```
io1 = DigitalIO.new(D1, OUTPUT)  
io1.high
```

UART

シリアル通信

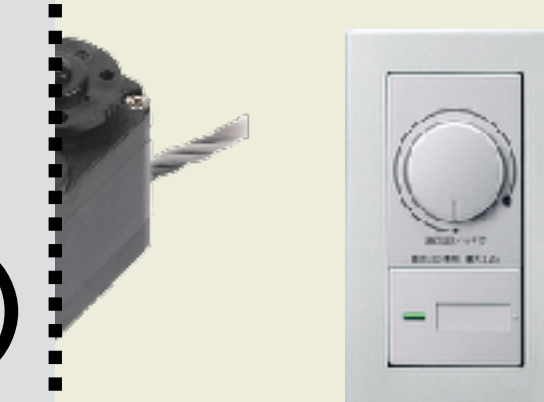


通信モジュール シーケンサ

```
pinMode( pin, 1 )  
digitalWrite( pin, 1 )
```

VM

ス幅変調



ボモータ 調光器

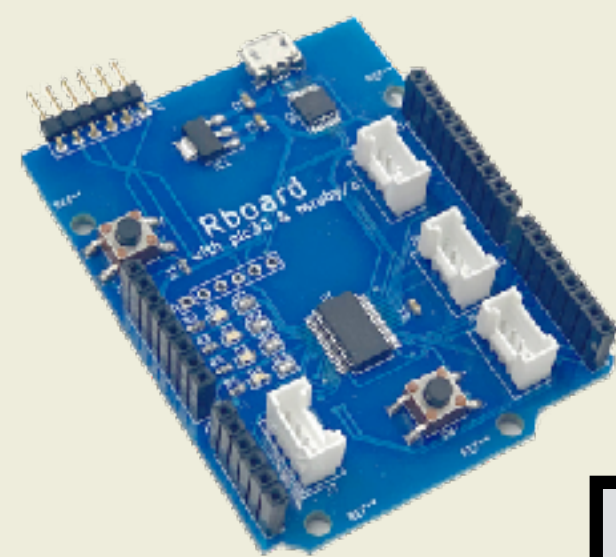
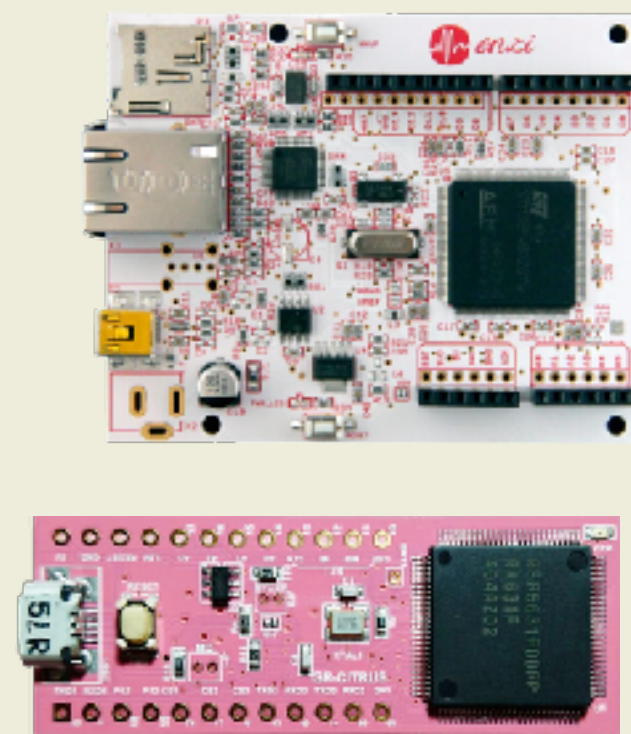


装置

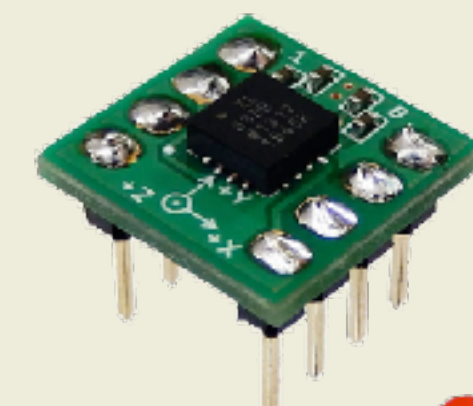


# 解決したい課題

- ⇒ 使用方法を共通化し、同じ方法で扱いたい！
- ⇒ mrubyとmruby/c 共通 I/O クラス検討開始！



共通化



```
led1 = GPIO.new(1,GPIO::OUTPUT)  
led1.write( 1 )
```



# プロセス

- ・ 調査 (rubymgem, mrbgem, MicroPython, Arduino, Qt)
- ・ 提案作成 ([https://github.com/HirohitoHigashi/mruby\\_io\\_class\\_study](https://github.com/HirohitoHigashi/mruby_io_class_study))
- ・ 合意形成 (会議体)
  - ・ mruby開発者
  - ・ CRuby開発者
  - ・ 諸先輩 Rboard開発者(mruby/c)、 wakayama.rb (mruby)...

# 設計思想、方針

- ・ 使い方の大枠を決める
  - ・ クラス名
  - ・ メソッド名とその動作
  - ・ 引数と戻り値
- ・ 正常系の動作はきちんと決める
- ・ 異常系（エラーハンドリング）は、運用してみて今後追加

# 設計思想、方針

- ・ コードの、部分を見るだけで何をしているかだいたい把握できること
- ・ 速度の低下を極力抑えるようにすること
- ・ うまく動いている別な実装（Rubyに限らず）あれば、それを真似るようにすること



# 設計思想、方針

- ・ コードの、部分を見るだけで何をしているかだいたい把握できること
- ・ 速度の低下を極力抑えるようにすること
- ・ うまく動いている別な実装（Rubyに限らず）あれば、それを真似るようにすること

当初は、3つめのこれ↑を最優先にしていた....

# 設計

## 具体化

1. コードの読みやすさのため、必要に応じてキーワード引数を使う。
2. 各クラスで共通したメソッド名を使う。
3. ステートレスにする。

# 設計

1. コードの読みやすさのため、必要に応じてキーワード引数を使う。
  - ・ 実行速度が遅くなりがちだが、引数が多いコンストラクタにおいてはコードの読みやすさを優先してキーワード引数を使う。
  - ・ コンストラクタは、このようなプログラムではたいてい初期化時に1回のみ呼ばれるだけなので、速度的ペナルティーが小さい。
  - ・ 頻繁に呼ばれる事が想定されるメソッドは、引数を単純にしてメソッド名と併せて考えれば意味が想像できるように配慮する。



# 設計

## UARTの例

(コードの読みやすさを優先してキーワード引数を使う)

```
SerialPort.new( "/dev/cuau0", 9600, 8, 1 )
```



(gem serialport より)

```
UART.new( unit:"/dev/cuau0", baudrate:9600, data_bits:8,  
          stop_bits:1 )
```

# 設計

## SPIの例

(頻繁に呼ばれる事が想定されるメソッドは引数を単純にしてメソッド名と併せて考えれば意味が想像できるるように)

```
spi.xfer( txdata: [0x10, 0x00] )
```

(gem spi より)



```
spi.transfer( [0x10, 0x00] )
```

# 設計

## 2. 各クラスで共通したメソッド名を使う

- ・ 読み込みは、**read** (POSIXから)
- ・ 書き込みは、**write** (POSIXから)
- ・ 設定は、**setmode** (MicroPythonのGPIOから)



# 設計

## 3. ステートレス

- ・ 状態をもたないようにする。
- ・ コールの順番依存性がなくなる。

(MicroPythonの例)

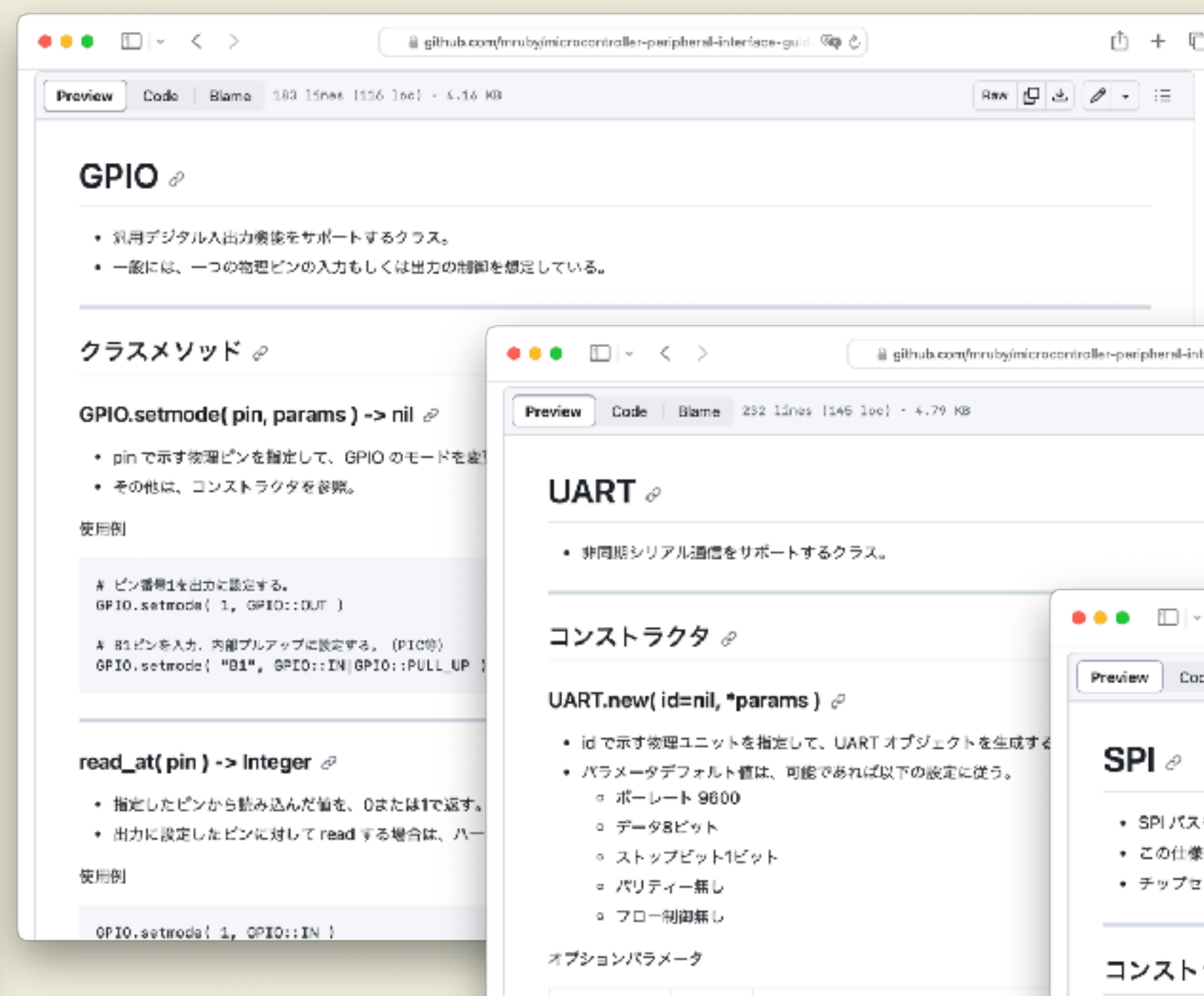
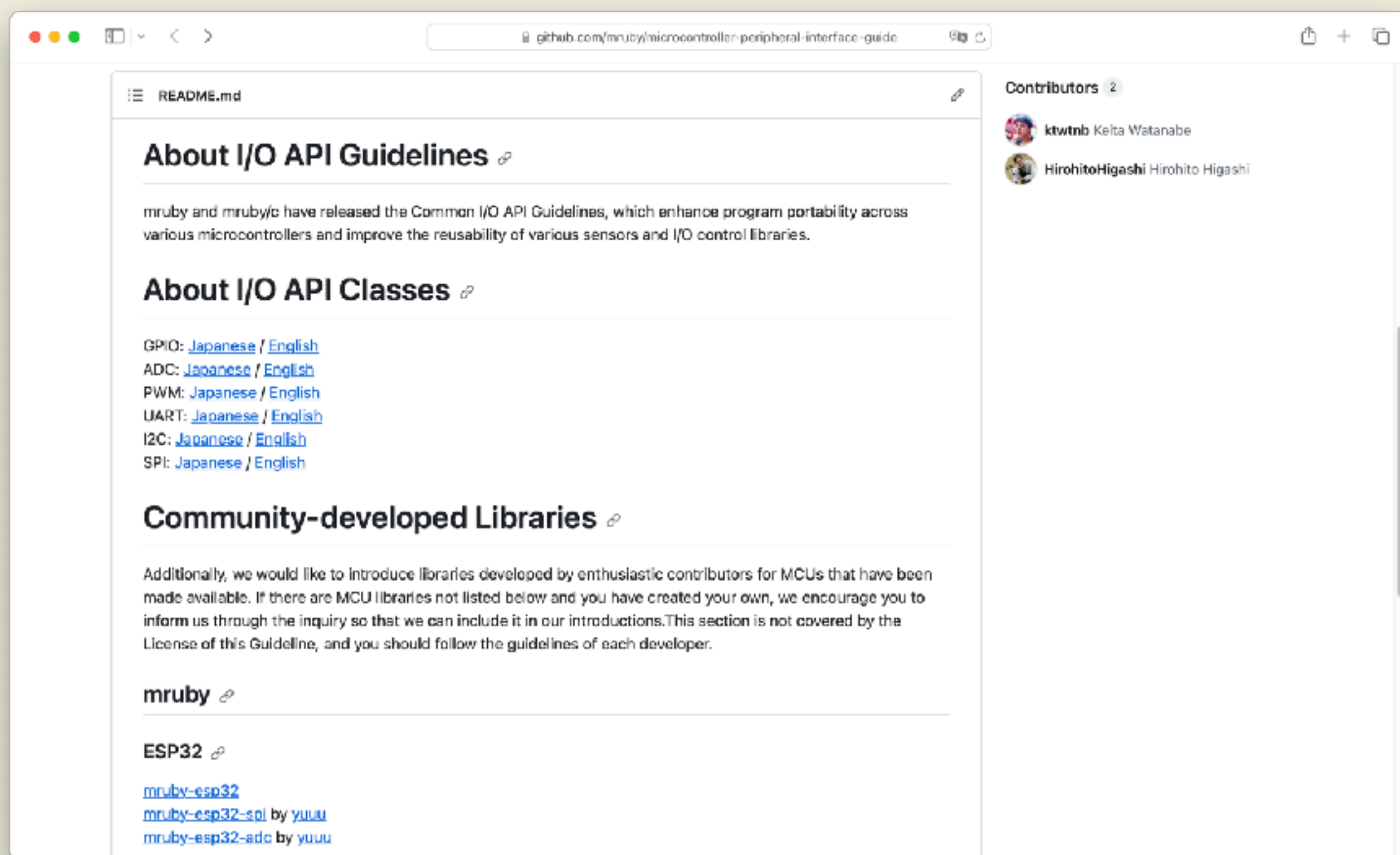
```
i2c.readfrom( i2c_addr, nbytes, stop=True )
```

stop=False とした場合に、ライブラリ側がそのことを覚えておく (=状態) 必要がある。

# 完成



<https://github.com/mruby/microcontroller-peripheral-interface-guide>



# 完成

CRuby版実装も作って  
おきました！

The screenshot shows the RubyGems search results page for the query "mruby- AND updated:[2023-09-25 TO \*)". The page displays four gems, each with its name, version, description, and download count.

Gem Name	Version	Description	Downloads
<b>mruby-linux-i2c</b>	0.9.2	I2C bus driver class library using Linux i2cdev. Compliant with mruby, mruby/c common I...	607
<b>mruby-sysfs-gpio</b>	0.9.3	GPIO class library using Linux sysfs. Compliant with mruby, mruby/c common I/O API guid...	479
<b>mruby-serialport-uart</b>	0.9.2	UART class library using serialport gem. Compliant with mruby, mruby/c common I/O API g...	273
<b>mruby-linux-spi</b>	0.9.2	SPI bus driver class library using Linux spidev. Compliant with mruby, mruby/c common I...	271



**いつもサポートしてくださるコントリビュータ  
の皆さん ありがとうございます！**

**Thank you to all the contributors who  
always support mruby and mruby/c !**

mruby Get Started mrubyを始めてみましょう

mruby/c チュートリアル

フクオカ&しまね  
mruby×IoT パビリオン



EdgeTech+2023 パシフィコ横浜 2023.11.15-17

Rubyフェスタ2024 アクロス福岡 2024.1.13



ご清聴ありがとうございました