

Supplement to “Making a MIDI controller device with PicoRuby/R2P2”

2025-05-22

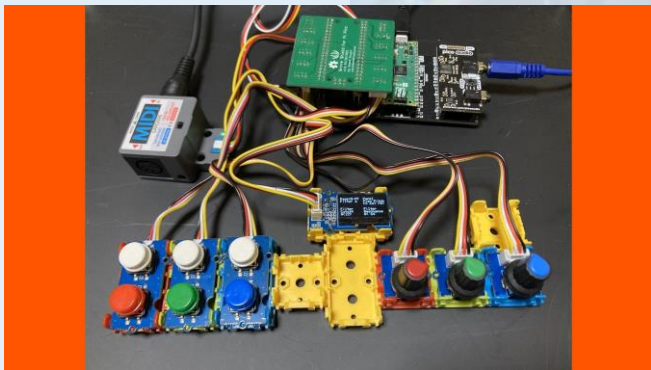
RubyKaigi 2025 事後勉強会 oto

Ryo Ishigaki

- Making a MIDI controller device with PicoRuby/R2P2 (RubyKaigi 2025 LT)
 - <https://speakerdeck.com/risgk/r2p2-rubykaigi-2025-lt>

MIDI Controller PRMC-1 (type-0) Demo

- Rubyを使ったMIDIデバイスを意識：@Kirika_K2 さんの “Do PRK_FIRMWARE Dream of MIDI implementation?” (KeebKaigi 2023)
- R2P2を使うキッカケ：@hachiblog さんの “Drive Your Code: Building an RC Car by Writing Only Ruby” (RubyKaigi 2024 LT)
- @asonas さんの発表やmruby関連発表が、プロポーザル提出の後押しに

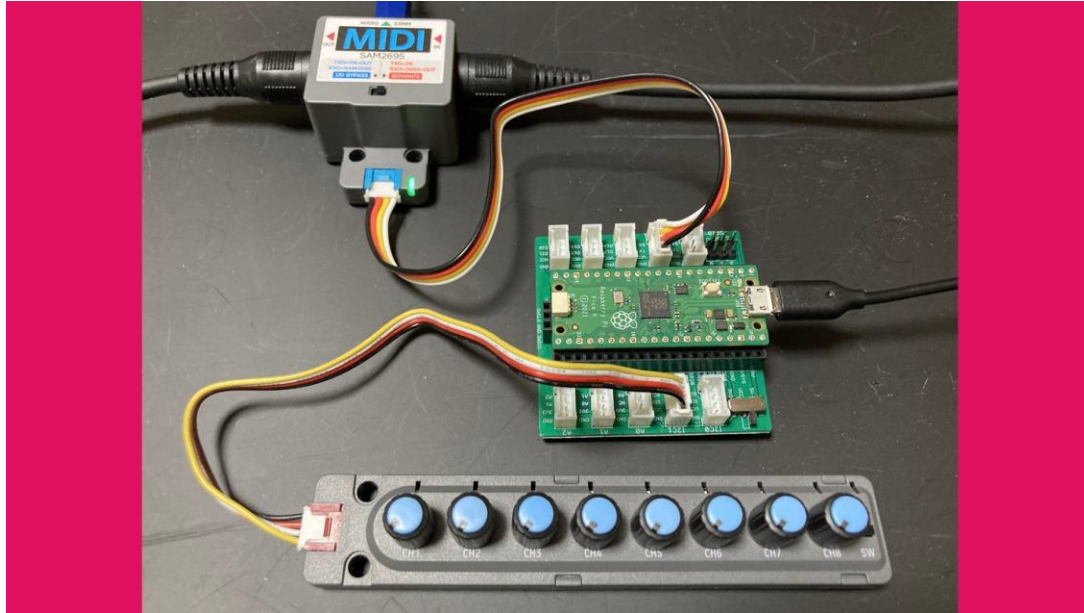


- RubyKaigi 2025のLTでは、実機で音が出ないリスクを考えて、最初に**デモ動画**を流した
- Rubyでのプロトタイピングから始まった**自作シンセサイザー**（シンセ）、の**最新モデル**の音を鳴らしたくなった
- Rubyで書いたMIDIコントローラー**PRMC-1 (type-0)** で、C++で書いた自作シンセ**PRA32-U2 (with Panel)** を演奏

<https://youtu.be/Eq25Ze3kTH4>

Thank you for listening!

MIDI controller PRMC-1 (type-0)



- **MIDIコントローラー**（MIDIコン）は、シンセやMIDI音源（sound module, tone generator）を制御するもので、**自分では音を出さない**
- MIDIユニットは、**MIDIトランスミッター**として使用（レシーバーにもなる）

- PRMC-1 is a **MIDI** Controller using **PicoRuby/R2P2**
 - <https://github.com/risgk/midi-controller-prmc-1>
- Made to control synthesizers in electronic music performances
 - PRMC-1 transmits MIDI messages
- Hardware (no soldering)
 - **Raspberry Pi Pico** • 部品代は合計8,000円くらい
 - Grove Shield for Pi Pico
 - M5Stack Unit 8Angle
 - M5Stack Unit MIDI
- All code is written in **Ruby!**

MIDI (Musical Instrument Digital Interface)

- Examples of MIDI messages
 - **Note On/Off**: Turning on/off a note
 - **Control Change**: Modifying sounds (or adjusting parameters)
 - **Program Change**: Changing tones
 - **Clock**: Synchronizing the tempo of devices (24 clocks per quarter note)
 - **Start/Stop**: Controlling playback
- MIDI receivers and transmitters use **UART** (Universal Asynchronous Receiver/Transmitter) at 31.25 kbaud
- For details, see the specifications
 - MIDI 1.0 Detailed Specification <https://midi.org/midi-1-0-detailed-specification>
 - MIDI1.0規格書 <https://amei.or.jp/midistandardcommittee/MIDIspcj.html>
- 書籍『**MIDIバイブルI**』『**II**』では、規格運用の実際について学べる（『I』だけ読むのもオススメ）
 - 『MIDIバイブルI - MIDI 1.0 規格 基礎編』リットーミュージック（1998年）<https://www.amazon.co.jp/dp/4845602679>
 - 『MIDIバイブルII - MIDI 1.0 規格 実用編』リットーミュージック（1998年）<https://www.amazon.co.jp/dp/4845603039>

Specifications of PRMC-1 (type-0)

- Simple functions
 - 4-step **chord sequencer** for playing chords as arpeggios
 - Adjusting Brightness (Cutoff) and Harmonic Content (Resonance)
 - Unit MIDI has a built-in synthesizer chip (SAM2695), so PRMC-1 can also produce sound directly
- Usage
 - CH1 – CH4 Knob: Root of Step 1 - 4 Chord
 - CH5 Knob: Arpeggio Pattern, 1 - 6
 - CH6 Knob: Brightness
 - CH7 Knob: Harmonic Content
 - CH8 Knob: BPM
 - SW Switch: Start/Stop Sequencer

• MIDIユニットからの音出しは**オマケ機能**

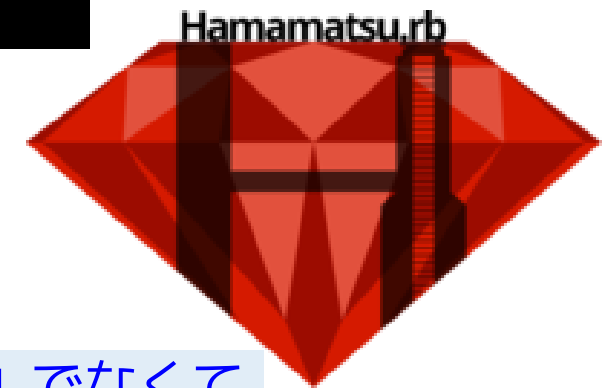
- セットアップ処理で音色を変更し、リリースタイムを長めに調整
- 今のUIでは、細かくパラメーターを編集できない
- PCM音源なので、波形の変化が滑らかでないところもある
- MIDIユニットは、ピアノなどの音も出せる**GM音源**で、**楽器作り**に役立つ
 - 「シンセ」という言葉を「楽器」の意味で使うなら、PRMC-1はシンセでもある
 - 注意：DREAM社の音源チップSAM2695は、**今年で生産終了**予定
- 懇親会では、バッテリーやスピーカーと一緒に首から下げながら音出し

• **アルペジオパターン** (type-0)

- セブンスコード、Up
- セブンスコード、Up & Down
- トライアド、Up
- トライアド、Up & Down
- Root + 4th + 5th、Up
- Root + 4th + 5th、Up & Down

About me

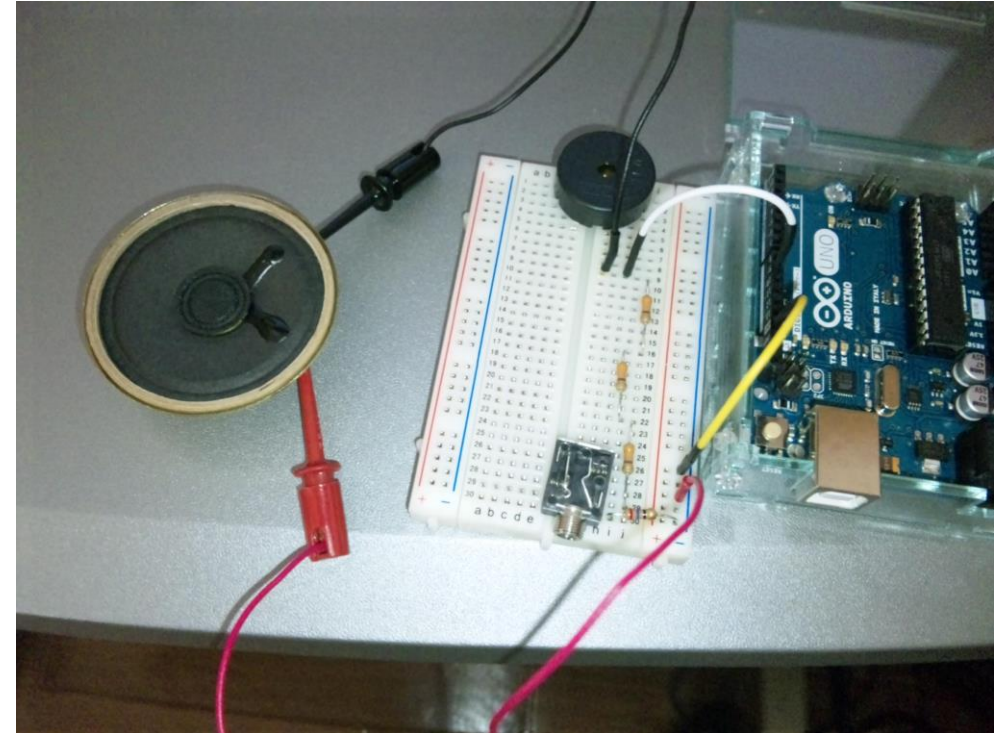
- **Ryo Ishigaki** (石垣 良) @risgk
- Member of **Hamamatsu.rb** (called “Hamamatsu Ruby”)
 - Hamamatsu is the physical birthplace of Ruby
- Embedded software engineer
 - Experience: projector, milling machine, audio product, etc.
- Making synthesizers for microcontrollers using C/C++ as a hobby



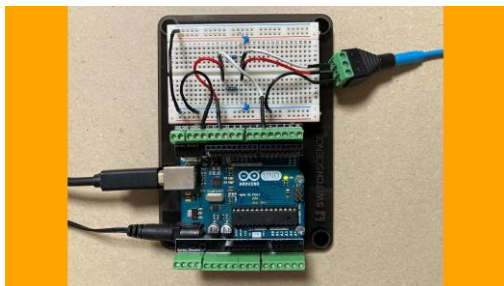
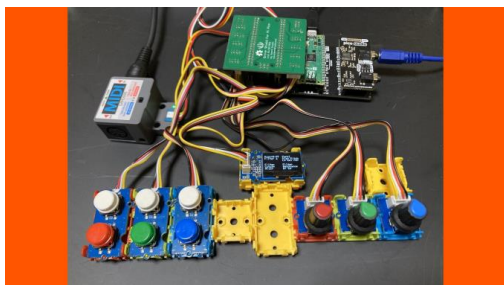
- 正式名称は「ハママツ（ドット）アールビー」でなくて「ハママツルビー」、通称「ハマルビ」

Making synthesizers

- Hamamatsu is famous for its musical instrument industry
- One of the reasons I started making synthesizers is that I moved to Hamamatsu
- About 11 years ago, my first synthesizer **VRA8** was prototyped with Ruby and ported it to **Arduino Uno**
 - Details: Ruby x Arduinoでシンセサイザーを作ってみた (浜松Ruby会議01)
<https://gist.github.com/risgk/0db52ea683530652d933>



Making and playing synthesizers

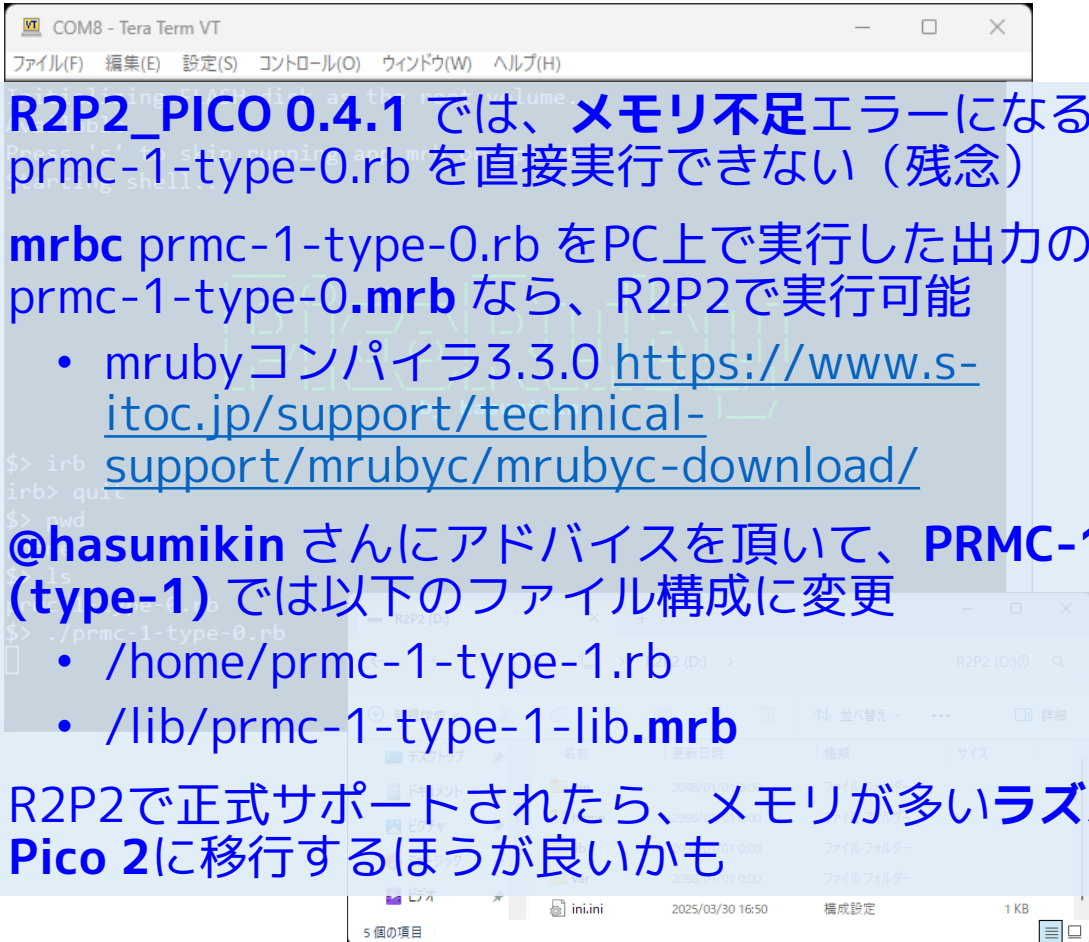


- As **ISGK Instruments**, I make about one synthesizer a year and exhibit it at Maker Faire and other events
 - Latest work: **PRA32-U2** for Raspberry Pi Pico 2
<https://github.com/risgk/digital-synth-pra32-u2>
- I wrote articles on synthesizer making for a magazine and a book
 - Book: ラズベリー・パイPico/Pico W攻略本
<https://interface.cqpub.co.jp/magazine/2023pico/>
- I am challenging electronic music performance using hardware synthesizer, called "machine live"
 - I will be at シンセカイリアル (Synthesizer Meeting Real) on April 28th
https://x.com/Yasushi_K/status/1910670512419738018

- **mruby**ファミリーの動きも追いかけてきたが、私に合った「使いどころ」を見つけられなかった
 - リアルタイムな音響合成には、あまり向いていない
- **シンセカイリアル**（次回は6/6）と出会わなかったら、マシンライブを始めず、MIDIコントローラーも作らなかったと思う

PicoRuby and R2P2

- **PicoRuby** is a Ruby implementation for one-chip microcontrollers
 - <https://github.com/picoruby/picoruby>
- PicoRuby's VM is **mruby/c**
- **R2P2** (Ruby Rapid Portable Platform) is a PicoRuby shell for Raspberry Pi Pico and others
 - <https://github.com/picoruby/R2P2>
- Usage of R2P2
 1. Drag and drop a release binary into the RPI-RP2 drive of Pi Pico
 2. Connect the shell through a serial port with a terminal emulator
 3. /home/app.rb is automatically run at startup

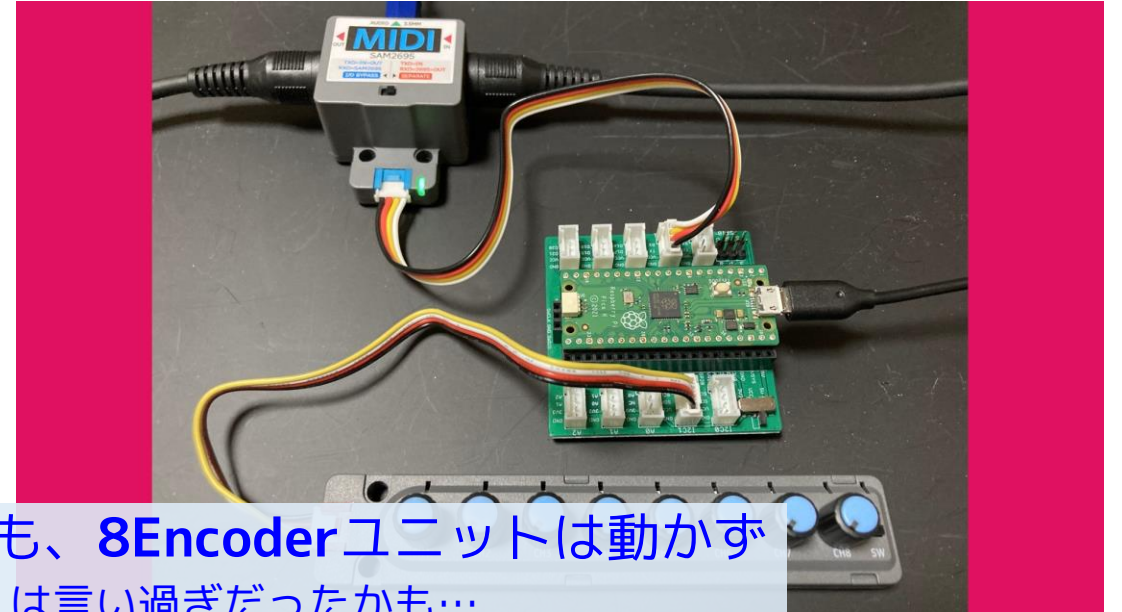


The image shows a terminal window titled 'COM8 - Tera Term VT' with a menu bar (File, Edit, Settings, Control, Window, Help). The terminal output shows the execution of 'prmc-1-type-0.rb' and the prompt 'irb>'. Below the terminal, a file explorer window shows the contents of the 'R2P2 (D:)' drive, listing files like 'prmc-1-type-1.rb' and 'lib/prmc-1-type-1-lib.mrb'.

- **R2P2_PICO 0.4.1** では、**メモリ不足エラー**になるため、`prmc-1-type-0.rb` を直接実行できない（残念）
- **mrbc** `prmc-1-type-0.rb` をPC上で実行した出力の `prmc-1-type-0.mrb` なら、R2P2で実行可能
 - mrubyコンパイラ3.3.0 <https://www.s-itoc.jp/support/technical-support/mruby/mruby-download/>
- **@hasumikin** さんにアドバイスを頂いて、**PRMC-1 (type-1)** では以下のファイル構成に変更
 - /home/prmc-1-type-1.rb
 - /lib/prmc-1-type-1-lib.mrb
- R2P2で正式サポートされたら、メモリが多い**ラズパイ Pico 2**に移行するほうが良いかも

Functional Units of M5Stack

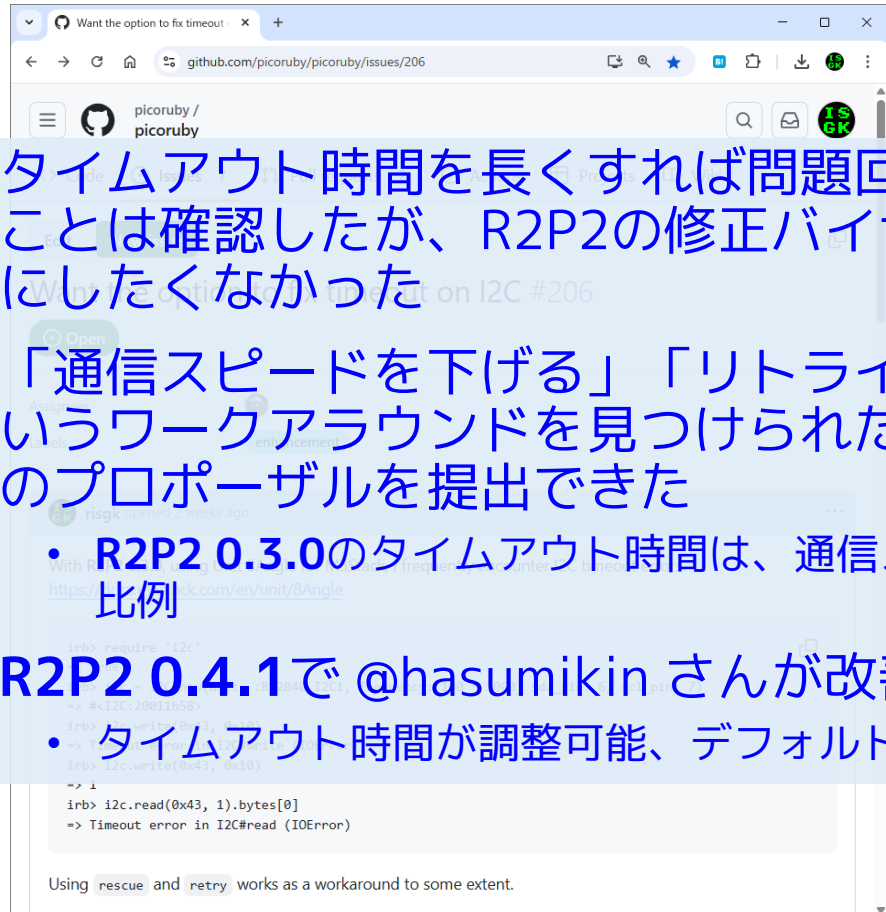
- Functional Units of M5Stack are compatible with **Grove** interface
- So, most of these should work with Raspberry Pi Pico
 - However, as far as I have briefly tested, M5Stack Unit **8Encoder** did not work well for some reason...



- Raspberry Pi Pico **C/C++ SDK**を使用しても、**8Encoder**ユニットは動かず
 - 「ほとんどのユニットはPicoでも動作するはず」は言い過ぎだったかも…
- **PRMC-1 (type-2)** では、**Dual Button**ユニットを追加（約600円）
 - **ByteSwitch**ユニットの動作も問題なさそう（約4000円）
- **@Y_uuu** さんの成果 “Porting PicoRuby to Another Microcontroller: ESP32” を受けて、**M5Stack**とR2P2を使った電子工作が増えていくと思う

Timeout error in I2C

- When accessing M5Stack Unit **8Angle** in R2P2 environment, **I2C** (Inter-Integrated Circuit) timeout errors occurred frequently
- Workaround
 - Use only 1-byte registers
 - Reduce communication speed to **20 kbps** (usually 100 or 400 kbps)
 - If an error occurred, **retry**
- I would like PicoRuby to have an option to adjust the timeout (like Arduino)
 - Want the option to fix timeout on I2C #206
<https://github.com/picoruby/picoruby/issues/206>



- タイムアウト時間を長くすれば問題回避できることは確認したが、R2P2の修正バイナリを前提にできなかった
- 「通信スピードを下げる」「リトライする」というワークアラウンドが見つけられたので、LTのプロポーザルを提出できた
 - R2P2 0.3.0のタイムアウト時間は、通信スピードに反比例
- R2P2 0.4.1で @hasumikin さんが改善済み
 - タイムアウト時間が調整可能、デフォルトで100ミリ秒

prmc-1-type-0.rb (about 400 lines)

```
require 'i2c'  
require 'uart'
```

```
# options  
MIDI_CHANNEL = 1  
...  
class M5UnitAngle8  
...  
end
```

```
class MIDI  
...  
end
```

```
class PRMC1Core  
...  
end
```

```
# setup  
...
```

```
# loop  
loop do  
...  
end
```

class M5UnitAngle8

```
def prepare_to_get_analog_input(ch)
  @i2c.write(ANGLE8_I2C_ADDR, ANGLE8_ANALOG_INPUT_8B_REG + ch)
  rescue
    retry
  end

def get_analog_input
  @i2c.read(ANGLE8_I2C_ADDR, 1).bytes[0]
  rescue
    retry
  end
```

- R2P2_PICO 0.4.1用のPRMC-1 (type-1) では、
rescue と retry を削除

class MIDI

```
def send_note_on(note_number, velocity, channel)
  @uart.write((0x90 + channel - 1).chr + note_number.chr + velocity.chr)
end
```

```
def send_note_off(note_number, velocity, channel)
  @uart.write((0x80 + channel - 1).chr + note_number.chr + velocity.chr)
end
```

```
...
```

```
def send_clock
  @uart.write(0xF8.chr)
end
```

class PRMC1Core (1)

```
def change_parameter(key, value)
  case key
...
  when 4
    arpeggio_pattern = (value * (6 - 1) * 2 + 127) / 254 + 1

    case arpeggio_pattern
    when 1
      @arpeggio_intervals_candidate = [1, 3, 5, 7, 1, 3, 5, 7]
      @step_division_candidate = 8
    when 2
      @arpeggio_intervals_candidate = [1, 3, 5, 7, 5, 3, 1, 3]
      @step_division_candidate = 8
```

class PRMC1Core (2)

```
def process_sequencer
  if @playing
    usec = Time.now.usec
  ...
    receive_midi_clock
  ...
end
end
...
def receive_midi_clock
  @midi.send_clock
  ...
  @midi.send_note_on(@playing_note, NOTE_ON_VELOCITY, @midi_channel) if @playing_note != -1
  ...
  @midi.send_note_off(playing_note_old, NOTE_OFF_VELOCITY, @midi_channel) if playing_note_old != -1
```

- 新しいバージョンでは、シーケンサー停止中でもMIDIクロックを送信し続けるように改善

setup

```
led_builtin = GPIO.new(25, GPIO::OUT)
led_builtin.write(1)
i2c1 = I2C.new(unit: :RP2040_I2C1, frequency: 20_000, sda_pin: 6, scl_pin: 7)
angle8 = M5UnitAngle8.new(i2c: i2c1)
uart1 = UART.new(unit: :RP2040_UART1, tx_pin: 4, rx_pin: 5, baudrate: 31_250)
midi = MIDI.new(uart: uart1)
prmc_1_core = PRMC1Core.new(midi: midi, midi_channel: MIDI_CHANNEL)
current_inputs = []
```

- PRMC-1 (type-1) では、
frequency: 100_000 に修正

loop

```
(0..7).each do |ch|  
  prmc_1_core.process_sequencer  
  angle8.prepare_to_get_analog_input(ch)  
  prmc_1_core.process_sequencer  
  analog_input = angle8.get_analog_input  
  
  if current_inputs[ch].nil? ||  
    analog_input > current_inputs[ch] + 1 ||  
    analog_input < current_inputs[ch] - 1  
    current_inputs[ch] = analog_input  
    prmc_1_core.change_parameter(ch, 127 - current_inputs[ch] / 2)  
  end  
end
```

- 8Angleへのアクセスには、1、2ミリ秒かかる
- 発音開始やクロックの送信タイミングがブレないように、**process_sequencer** を随時呼び出し
- カットオフ調整が滑らかでない（約40ミリ秒周期で更新）問題も、呼び出し頻度を上げれば改善

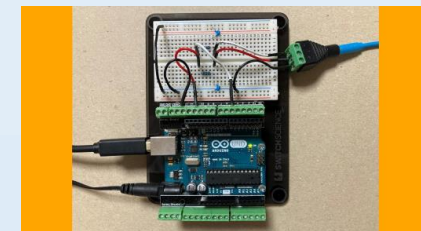
After this

- ネタ1：MIDI受信に対応、MIDIライブラリの整備（API定義？ USB MIDI対応？）
- ネタ2：DJ/VJアプリを自分だけのMIDIコンで制御？（DJは詳しくありませんが）

- I will continue to evolve **PRMC-1**
- **Ruby** code is easily modified, so **PRMC-1 (type-N)** may be made for each performance

- ネタ3：Cのコードで音響合成するシンセの**Picogem**（またはmrbgem）を作る
 - **CircuitPython**（**MicroPython**のフォーク）には、**synthio**というコアモジュールが含まれる
 - エイリアスノイズ（折り返し雑音）対策は充実してなさそう…
 - <https://docs.circuitpython.org/en/latest/shared-bindings/synthio/>
 - Rubyベースのライブコーディング環境**Sonic Pi**は、オーディオプログラミング言語**SuperCollider**のサーバーを使用
 - <https://sonic-pi.net/> <https://supercollider.github.io/>
 - ライブラリとして「いい感じ」の仕様にするには、**Web Audio API**が参考になるかも
 - Web Audio API 1.1 <https://www.w3.org/TR/webaudio-1.1/>
 - **AMY**というマイコン用シンセライブラリはMicroPythonで制御できる
 - <https://github.com/shorepine/amy>

- ネタ4：Rubyのコードで音響合成するシンセをマイコンで動かす
 - ラズパイPico 2とmruby/c（またはmruby）なら、そこそこ行ける？
 - 私のシンセの代表作**VRA8-U**は、Arduino Uno（マイコンは**8ビット**、16MHz）で動作
 - 藤本健のDigital Audio Laboratory 第956回 <https://av.watch.impress.co.jp/docs/series/dal/1442328.html>



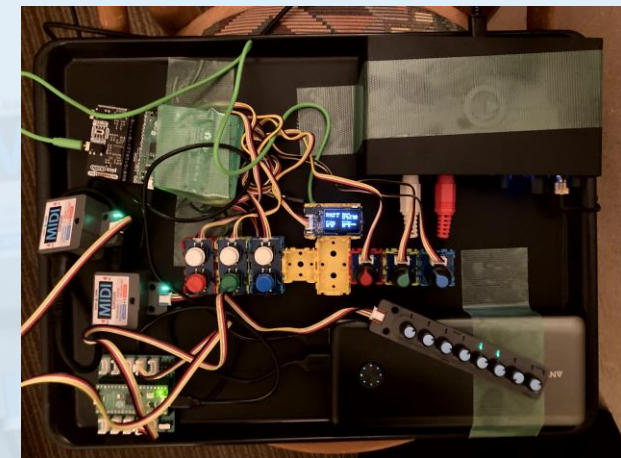
Would you like to create your own musical device?

Enjoy electronics!

- 宣伝1 : 6/1(日) Synth Maker Meetup #4 浜松で開催
 - <https://synthmeetup4.peatix.com/>
- 宣伝2 : 7/5(土) Hamamatsu Micro Maker Faire 2025
 - <https://makezine.jp/event/makerfaire/hmmf2025/>

Finally... Live performance!

- RubyKaigi 2025のLTでは、この時点で残り15秒くらい…
- 再生開始したが、なぜか実機の音が出なかった
 - PCのオーディオ設定不足を疑ったが、PRMC-1の不具合だった可能性も
 - LT前の転換が長引いていたので、音チェックは省略していた
- LT終了後、運営の @a_matsuda さんから「せっかくなので、蛍の光のようにDay 2帰りの音楽を演奏してください」と機会を頂き、実機の音を鳴らしました！ ありがとうございました！
 - LTでも、電子ピアノ（ベル）の音を鳴らす予定でした



- 本日のデモ：MIDIコンPRMC-1 (type-2) を使って、自作シンセPRA32-U2 (with Panel) と、リズム担当のヤマハSEQTRAKを同期演奏

- PRMC-1 (type-2) の主な変更点
 - アルペジオパターンの追加
 - 2～7個目の音符のオン／オフ設定（レゾナンス調整は削除）
 - Dual Buttonでトランスポーズ

