Making IoT device with Ruby

HASUMI Hitoshi

Monstar Lab, Matsue office

Nov. 2, 2018 (day 2)

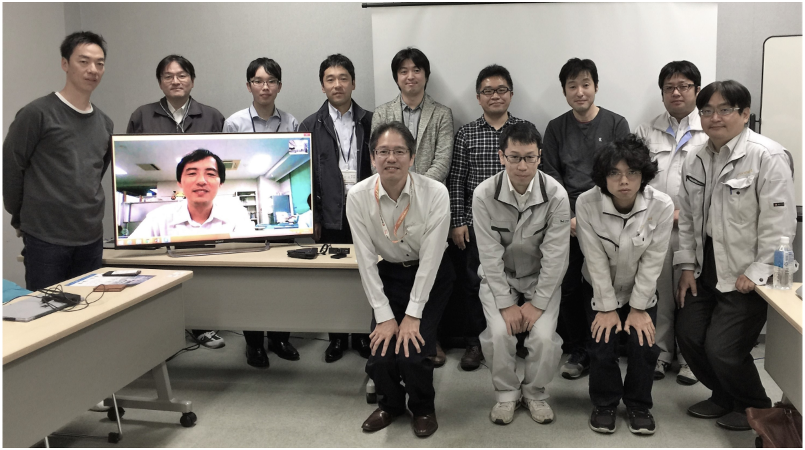
RubyWorld Conference 2018

Kunibiki Messe, Shimane

Information

Information

- mrubycKaigi#1 on Oct. 31, 2018 (the day before yesterday)



Information

Information

- Polish Ruby users group sent me an invitation
  - they want me to talk about **mruby/c**
  - I'm going to have several talks and workshops in May 2019

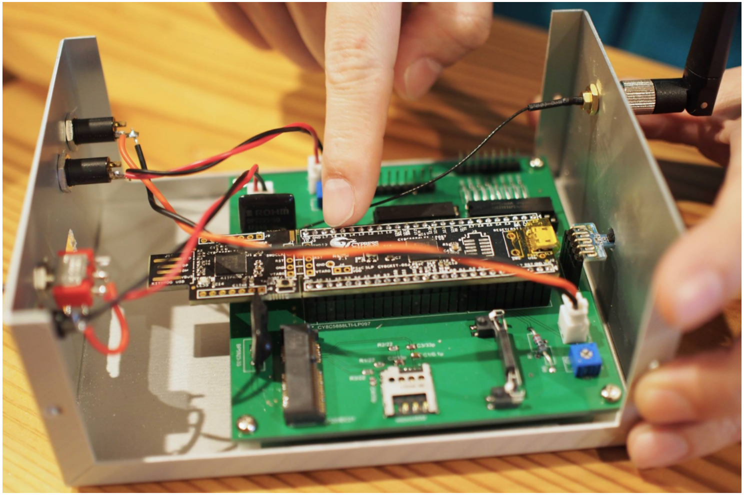
Information

- Polish Ruby users group sent me an invitation
  - they want me to talk about **mruby/c**
  - I'm going to have several talks and workshops in May 2019

RubyWorld!!!

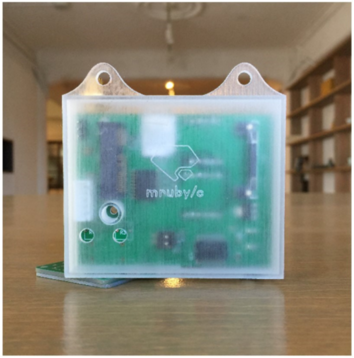
Making IoT device with Ruby

- PoC product for a Sake brewery, 旭日酒造



Making IoT device with Ruby

- prototype for mass production
  - you can see the device at Monstar-lab's booth downstairs today (Nov. 1-2)



Making IoT device with Ruby

- what does **`with Ruby`** mean?
  - not only CRuby
  - not only mruby/c, too

Making IoT device with Ruby

- what does **`with Ruby`** mean?
  - not only CRuby
  - not only mruby/c, too
  - but also **RubyWorld**

About me

About me

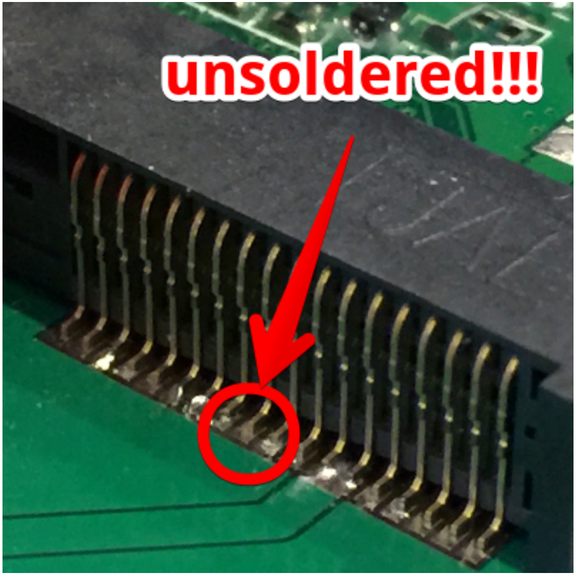
About me

- Monstar Lab / Matsue office (now hiring!)

About me

- HASUMI Hitoshi（羽角 均）@hasumikin
- finished master's degree in architecture department
    - majored in the history of Italian architecture
- became a programmer at 35 years old
- neither a computer specialist nor an electricity expert

Technology stack of IoT



Technology stack of IoT (1/2)

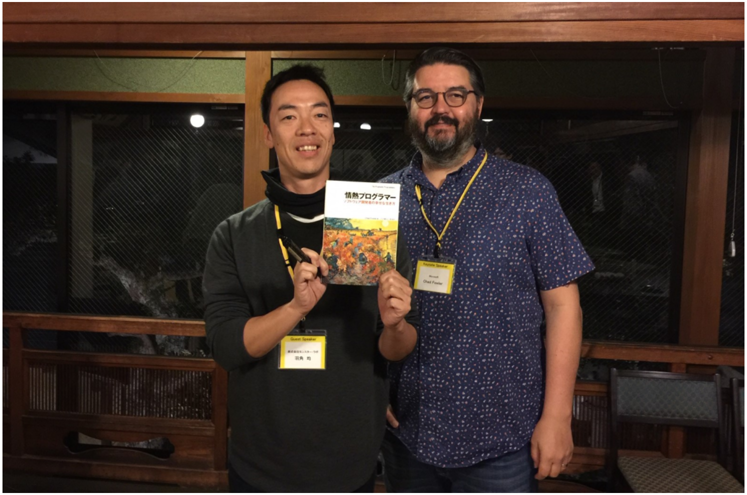
- TCP/IP
- cloud service
- RDB and KVS
- server programming
- mobile programming
- security
- test

Technology stack of IoT (2/2)

- high school physics electricity and transistor
- microcontroller and peripherals like UART, I2C, ADC, etc.
- circuit and PCB artwork
- soldering and wiring
- 3D CAD for housing
- suppliers
- firmware programming

Understanding the business

- Sake brewing process and Sake itself



Sake itself

23BY
加水火入
12 %
生酛純米
旭日
じゅうじあさひ
改良雄町にごり火入
26by
純米吟醸酒
旭日
じゅうじあさひ
五百万石
25BY
生酛純米
旭日
じゅうじあさひ
山廃仕込
純米酒
旭日

The most thing you should know is

The most thing you should know is

Ruby on Rails

Ruby on Rails

- tells you what a good API is
- tells you what a reinventing the wheel is
- tells you what an ecosystem is
- tells you what a web service is

Technology stack of IoT (1/2) again

- ✓TCP/IP
- ✓cloud service
- ✓RDB and KVS
- ✓server programming
- ✓mobile programming
- ✓security
- ✓test

Ruby on Rails

- gives you time on digging into technologies other than the server app
- gives you wings

Technology stack of IoT (2/2) again

- ✓high school physics electricity and transistor
- ✓microcontroller and peripheral interfaces like UART, I2C, ADC, etc.
- ✓circuit and PCB artwork
- ✓soldering and wiring
- ✓3D CAD
- ✓suppliers
- ✓firmware programming

Microcontroller

- I use microcontrollers instead of single board computers like Raspberry Pi

Microcontroller, upside

- starts immediately right after plugged in
  - end users, brewery workers in my case, can use it simply
- you can narrow security issue list
  - many a malware aims at linux or windows platform as a target
  - you don't need to consider unnecessary deamon
  - neither need to do `apt upgrade` nor `yum update`

Microcontroller, upside

- low energy
  - rarely overheated
  - many choices of power supply
- mass production
  - you can choose appropriate chipset (number of GPIOs, memory size, etc.) for your application
  - cost advantage for parts supply and subcontractor manufacturing

Microcontroller, downside

- less resource
  - CPU, memory
- hard to be soldered

Sake IoT project

Sake IoT project

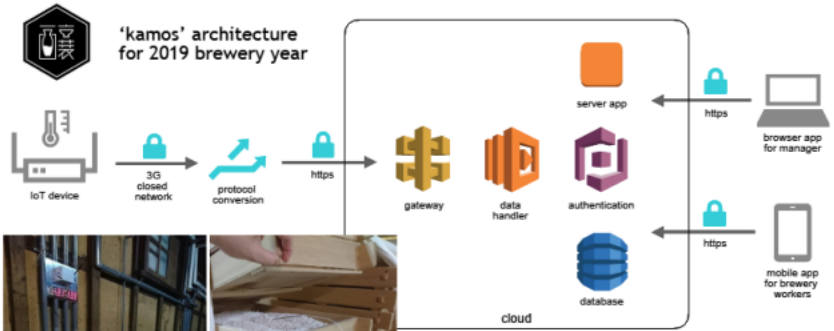
- IoT system for Asahi-shuzo（旭日酒造）
- delivered to actual brew work in January 2018
- devices post temperature of Sake materials in brewing, surrounding temperature and humidity to the cloud
- then, those data are displayed on the smartphone app
- the firmware written in **mruby/c**

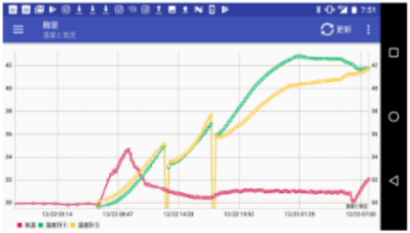
what does mruby**/c** mean?

what does mruby/c mean?

- compact
- concurrent
- capability

Sake IoT project





So many factors to be troubled in IoT

- circuit design, soldering, wiring, peripheral equipments, network...
- hard to find why the application doesn't work well
- in addition to above, I introduced a new layer of mruby/c
- one year ago, mruby/c was yet young, had bugs and insufficiency
    - (now it is enough good)

So many factors to be troubled in IoT

then, was mruby/c bad?

So many factors to be troubled in IoT

then, was mruby/c bad? - NO

- IoT at work makes you hurry, imagine
    - you have to go alternately to dark 10℃ storage cellar and humid 35℃ manufacturing room
    - brewery workers run around
    - you have to amend your firmware with your small laptop in 10 minutes
    - you will thank Ruby's descriptiveness and agility

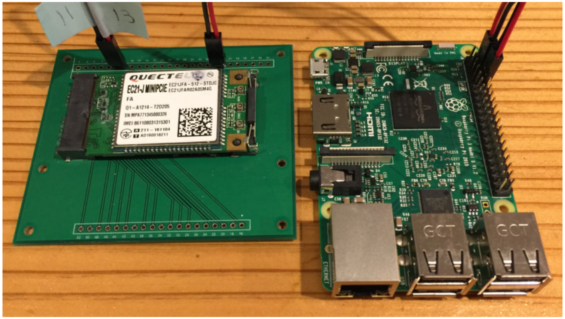
Does IoT at work make you hurry?

Does IoT at work make you hurry?

Pre-prototyping

- preparation is the most important thing
- you have to confirm if a part works as same as the datasheet
    - sometimes it is different
- you can prepare with Ruby

Pre-prototyping



- Raspberry Pi & CRuby are great for pre-prototyping
  - use breadboard or make PCB for test like this photo

Pre-prototyping

ex) CRuby for serial communication test

```ruby
# notice this is CRuby for RasPi
require 'rubyserial'
require 'timeout'
sp = Serial.new '/dev/serial0', BAUDRATE, 8 # match with your instrument
loop do
  puts '[command]'
  command = gets
  sp.write command.sub("\n", "\r") # replace LF if needed
  sleep 0.1
  result = ''
  begin
    Timeout.timeout(10) do
      loop do
        line = sp.read(128)
        break if line == '' && result != ''
        result << line
        sleep 0.1
      puts '=> ' + result
  rescue Timeout::Error
    puts 'timeout!'
ennnnd
```

Pre-prototyping

ex) CRuby for serial communication test

```
$ serial_communication_test.rb
[command]
AT                # command
=> OK             # response
[command]
AT+CIMI           # command
=> 123456789012 # response
[command]
AT+XXX            # command
=> error          # response
```

Pre-prototyping

- then, you can copy and paste CRuby snippet to mruby/c source

Firmware programming with mruby/c

Firmware programming with mruby/c

- Ruby power
  - string operations
  - encapsulation (object oriented)

Firmware programming with mruby/c

```ruby
# ex) string operations
#
# concatenation
parameter = 'name=' + name + '&age=' + age.to_s
# => name=hasumikin&age=43

# substitution
'what_a_wonderful_world'.tr('_', '-')
# => what-a-wonderful-world
```

Firmware programming with mruby/c

```
# ex) encapsulation (object oriented)
class LoggerBase
  def info(line)
    write(:info, line)
ennd
class LoggerBLE < LoggerBase
  def initialize(*args)
    @ble = BluetoothLowEnergy.bind_characteristic(args[0])
  end
  def write(log_level, line)
    @ble.notify(line)
ennd
class LoggerFlashROM < LoggerBase
  def initialize(*args)
    @rom_io = RomFileStream.open('/log.txt', 'w')
  end
  def write(log_level, line)
    @rom_io.write_ln(line)
ennd
logger = LoggerBLE.new(:log)  /* or */  logger = LoggerFlashROM.new
logger.info('this is log')
```

Firmware programming with mruby/c

- you must write both mruby and C
  - C for microcontroller I/O
  - mruby for business logic
- mruby/c seems like a thin wrapper for C
  - two sides of the same coin:
    - you have to write C that directly communicate with peripherals
    - you can fall back to C anytime you get stuck

Find more information on

- rubykaigi.org/2018/presentations/hasumon.html
- shimane.monstar-lab.com/hasumin
- follow twitter.com/mrubyc_jp
  - ITOC and I are planning to make workshops of mruby/c

Conclusion

Conclusion

- Thank Ruby
  - from pre-prototyping to production

Conclusion

- Thank Ruby
  - from pre-prototyping to production
- Thank Rails
  - full of really important things

Conclusion

- Thank Ruby
    - from pre-prototyping to production
- Thank Rails
    - full of really important things
- Thank Sake

Thank you all!