mruby/c: Running on Less Than 64KB RAM Microcontroller

November 18-20, 2019

HASUMI Hitoshi @hasumikin

Me

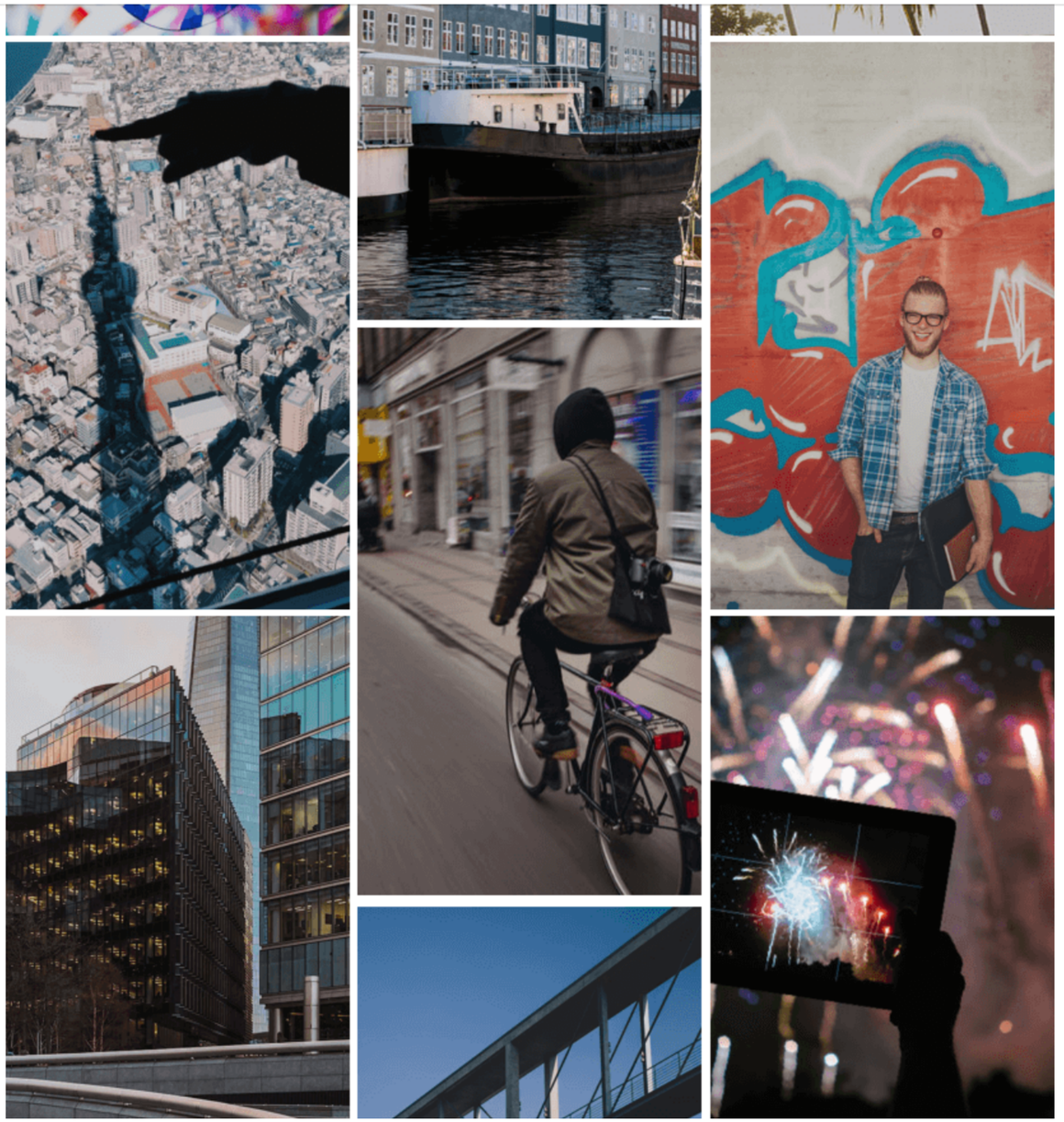
- HASUMI Hitoshi
  @hasumikin
- Live in Matsue city,
  a holy place of Ruby
- Sake
  Soba
  Coffee

Me

With offices in 26 cities around the world, we know your market

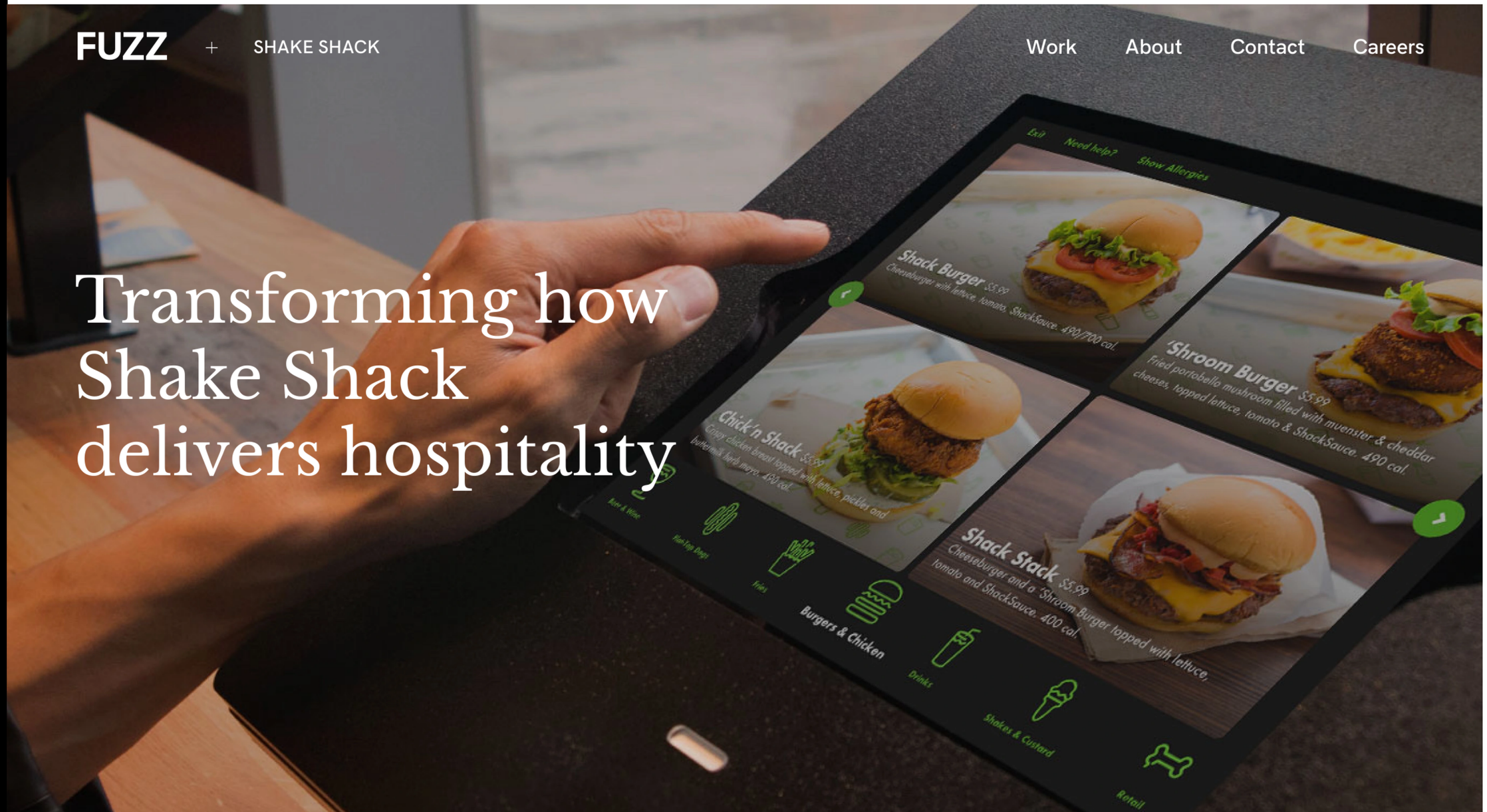
Tokyo, Osaka, Matsue, Chengdu, Qingdao, Shanghai, Beijing, Hanoi, Danang, Singapore, Dhaka, Cebu, Manila, Dubai, Copenhagen, Arhus, London, Manchester, Amsterdam, Prague, Bangkok, Berlin, New York, Boulder

Locations > Join our global team >



RubyConf 2019
Nashville

FUZZ



RubyConf 2019
Nashville

Matsue.rb

Chapter 1

Introduction

Terminology

- mruby/c
  - A language implementation I will talk about today
  - I say mrubyc since **/c** [slˈæʃ síː] is hard to pronounce
- Microcontroller
  - Small computer contains CPU, memory and programmable I/O peripherals

Terminology

- RTOS
  - Real-time OS. Usually used for microcontroller
- Task
  - Almost equivalent to `Thread` in Linux. We say `Task` in microcontroller world

Terminology

- 旭日酒造 (Asahi-Shuzo)
  - Shuzo means `Sake brewery`
  - One of the best Japanese Sake brewery
  - FYI, Asahi Breweries (famous for SUPER DRY) has no concern with Asahi-Shuzo
  - Asahi-Shuzo and I make an IoT system using mruby/c

Why microcontroller?

- I don't use single board computer like Raspberry Pi for production environment.
- I use microcontroller, instead
  - It starts immediately right after plugged in
  - End-users, brewery workers in my case, can use it simply like home electical appliance

Why microcontroller?

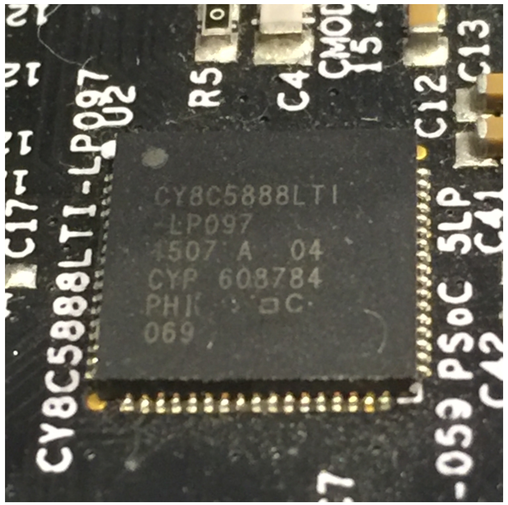
- Microcontroller can run without OS
  - So-called `Bare Metal`
- You can narrow security issue list
  - Many a malware aims at Linux or Windows platform as a target
  - You don't need to consider unnecessary deamon
  - You don't need to do `apt upgrade`
- (RasPi can be bare metal device if you want)

Why microcontroller?

- Low energy
  - Rarely overheated
  - Many choices of power supply
- Mass production
  - You can choose appropriate chipset (number of GPIO, memory size, etc.) for your application
  - Cost advantage for parts supply and subcontractor manufacturing

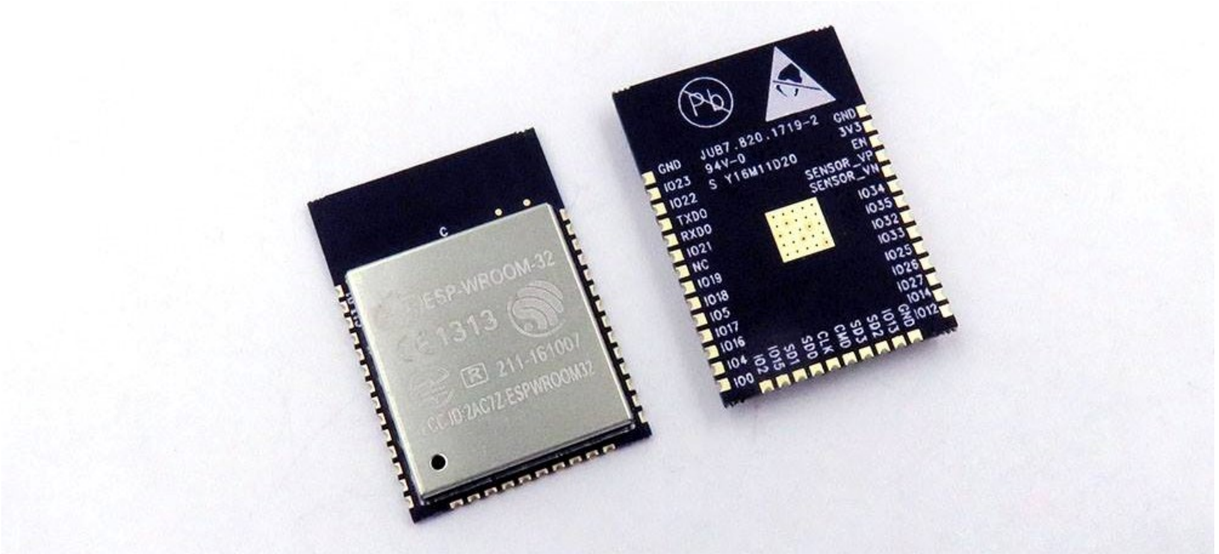
Which microcontroller?

Which microcontroller?



- e.g. CYPRESS PSoC5LP
  - 32 bit Arm Cortex-M3 CPU
  - Flash: 256KB
  - SRAM: 64KB (target size of mruby/c)

Which microcontroller?



- e.g. Espressif ESP-WROOM-32 (ESP32)
  - 32 bit dual core LX6 CPU
  - Flash: 4MB
  - SRAM: 520KB

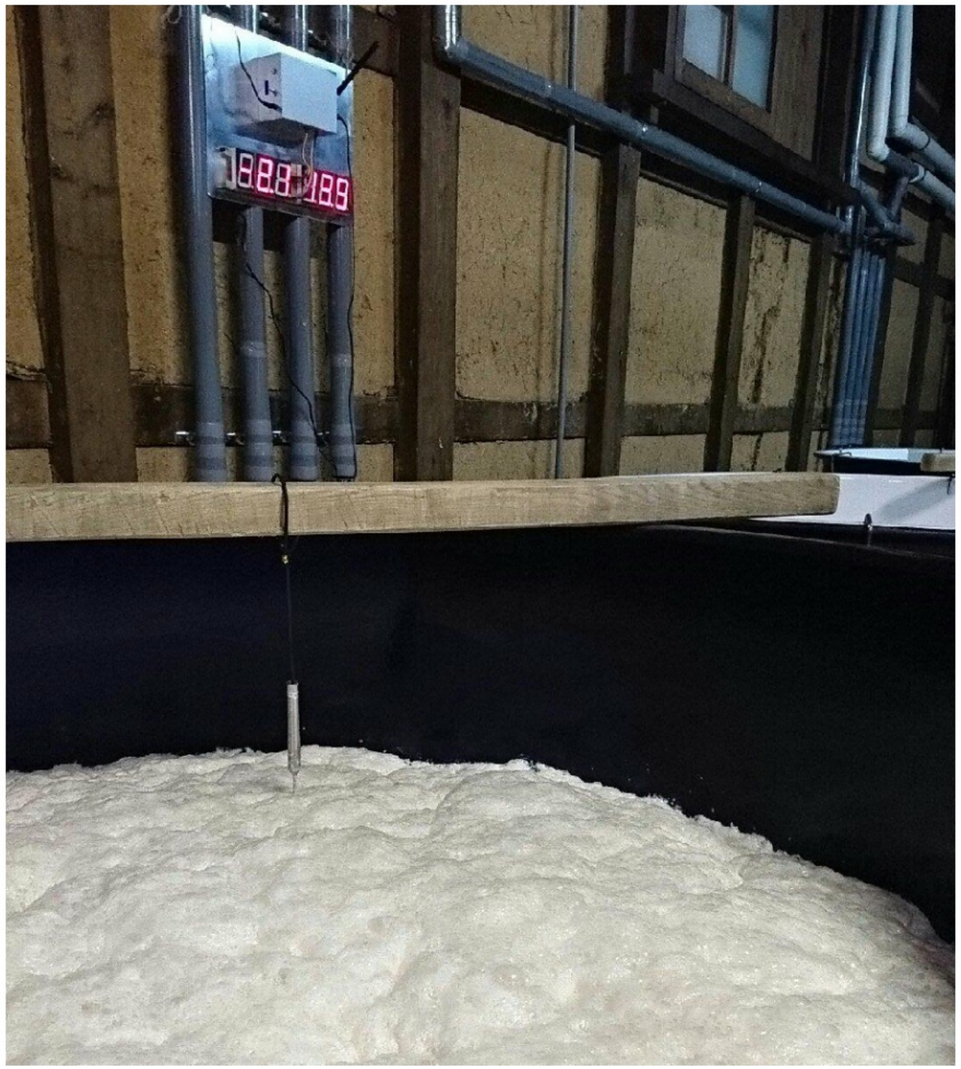
My IoT project

My IoT project

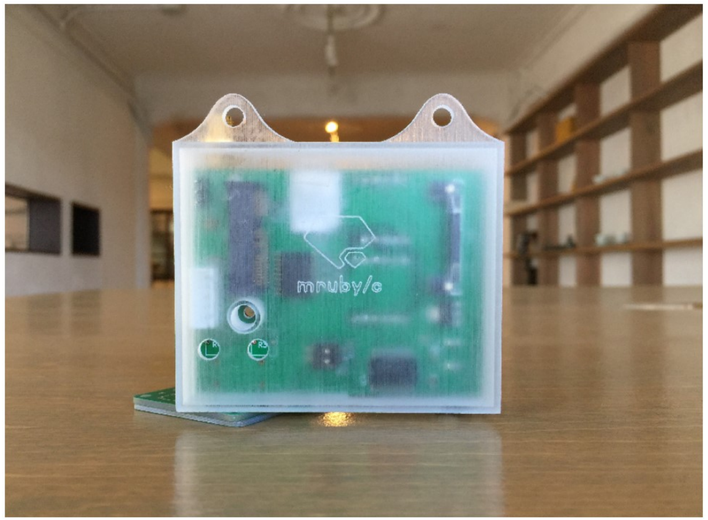
- IoT system for Sake brewing
- PSoC5LP
- Delivered to actual brew work in January 2018
- Devices post temperature of Sake ingredient in brewing, surrounding temperature and humidity to server
- Data is displayed on mobile app

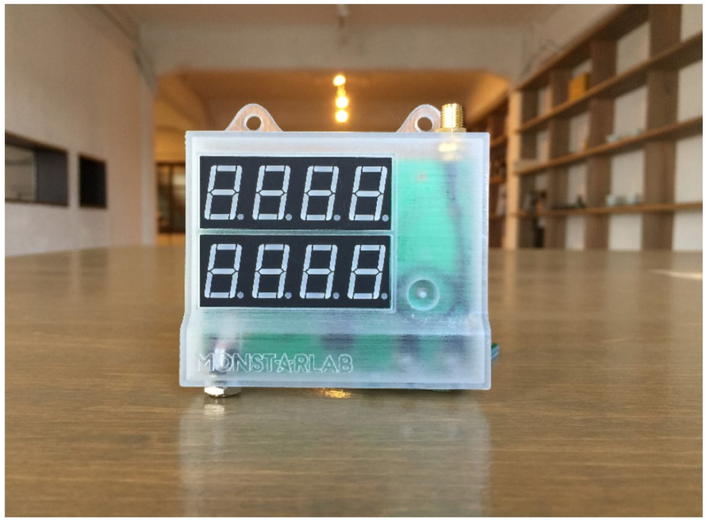
RubyConf 2019
Nashville

My IoT project



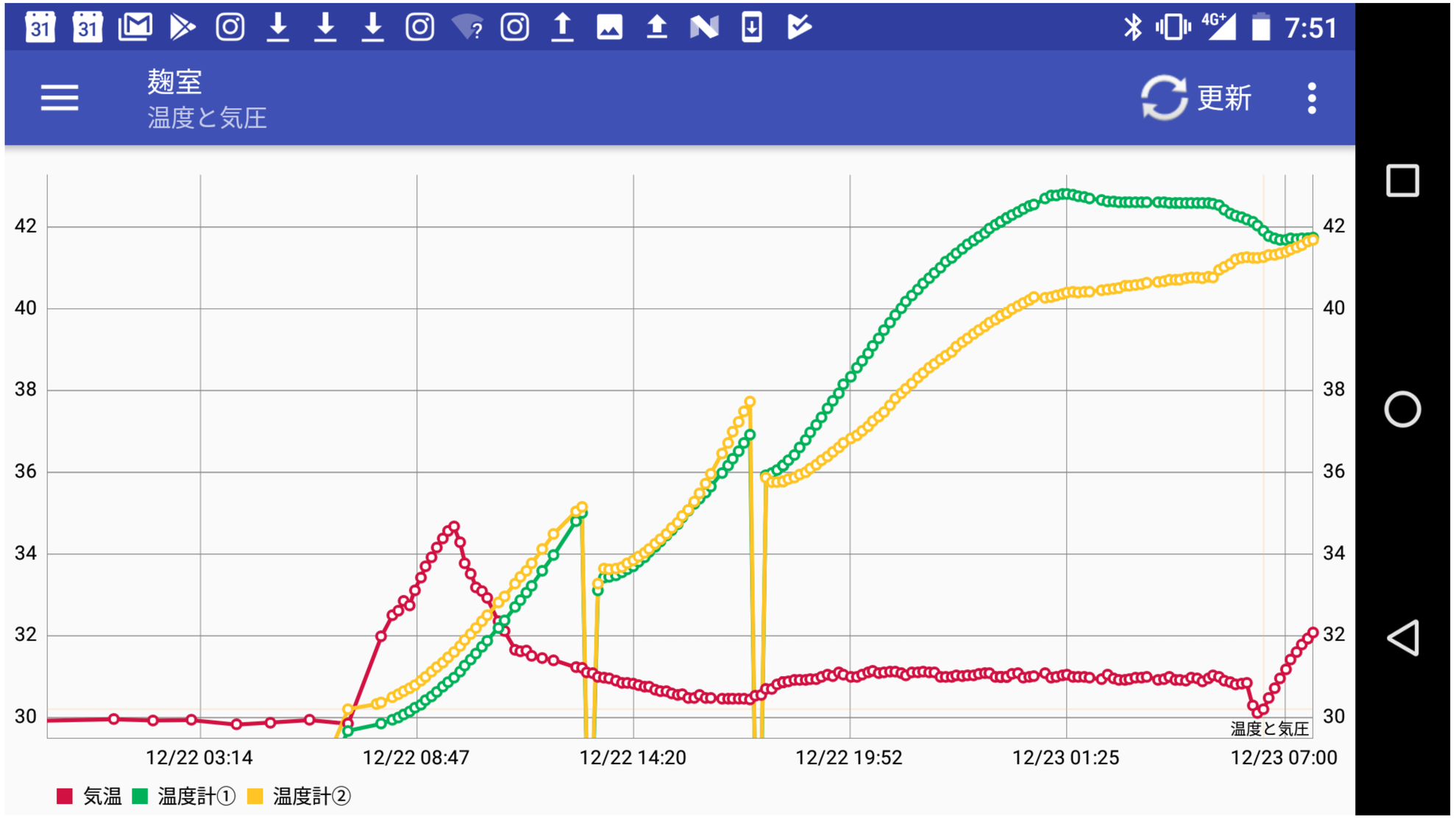
My IoT project





RubyConf 2019
Nashville

My IoT project



IoT in field makes you hurry

- Imagine,
  - You have to go back and forth between dark 5℃ (=41°F) storage cellar and humid 35℃ (=95°F) rice mold 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

RubyConf 2019
Nashville

RubyConf 2019
Nashville

Policies Team

RubyKaigi 2020

Apr 9th (Thu) - 11th (Sat)

Matsumoto Performing Arts Centre (まつもと市民芸術館), Nagano, Japan

Demo

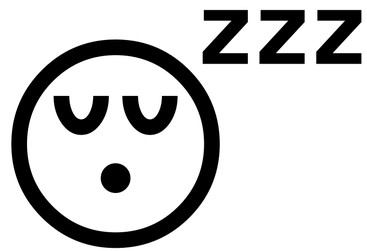
- CO_2 concentration
  - 400ppm : Atmospheric
  - 1000ppm : Your programming speed decreases
  - 1500ppm : FYI, tomatoes 🍅 may grow well
  - > 2000ppm : Sleepiness, headache
  - > 40000ppm : 💀
- (Please look at the screen)

Demo

- My device is taking CO_2 concentration
- CO_2 may increase because of your breathing

Demo

- My device is taking CO_2 concentration
- CO_2 may increase because of your breathing
- I will prove that it is due to CO_2 if you fell asleep while I was speaking



RubyConf 2019
Nashville

Bugaboos in IoT

Bugaboos in IoT 

- Peripheral equipments ...★
- Circuit, wiring and housing
- Printed circuit board = PCB
- Soldering ...★
- Firmware with C, mruby and mruby/c ...★
- Network, TCP/IP, Bluetooth, etc.

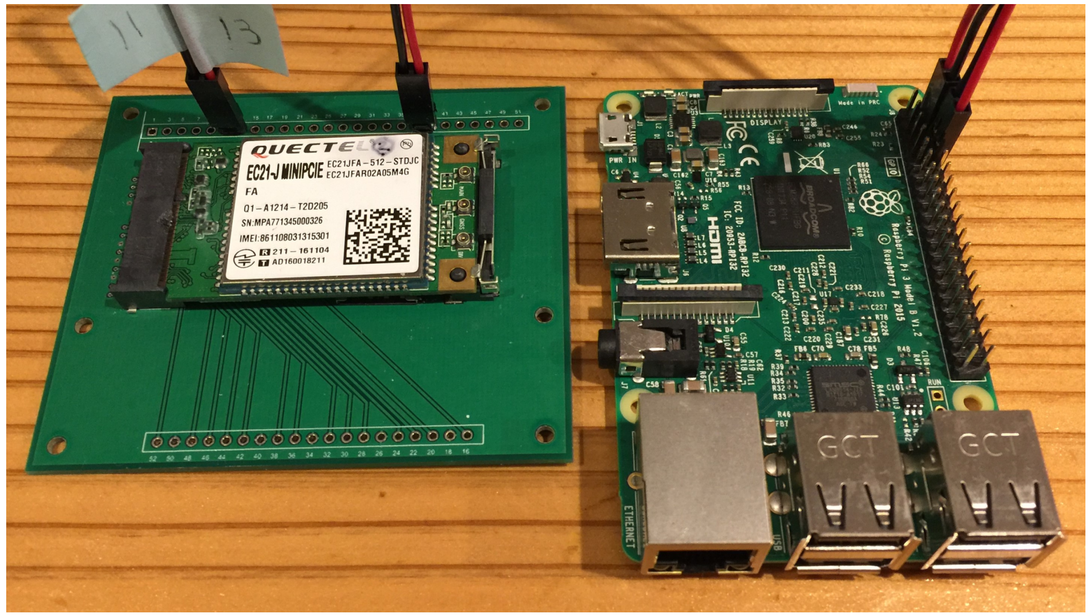
★...I will cover these topics today

Peripheral equipments

Peripheral equipments

- Very important to check the part before writing application code
  - Do equipments like sensor or communication module work as its spec sheets?
  - Whether or not the equipment is broken (sometimes broken by soldering heat 😭)
- Combining parts is unrevertable
  - We don't have a Git for hardware.

Peripheral equipments



- Raspberry Pi & CRuby are great for pre-prototyping
  - Use breadboard or make PCB for experiment

CRuby and serial port

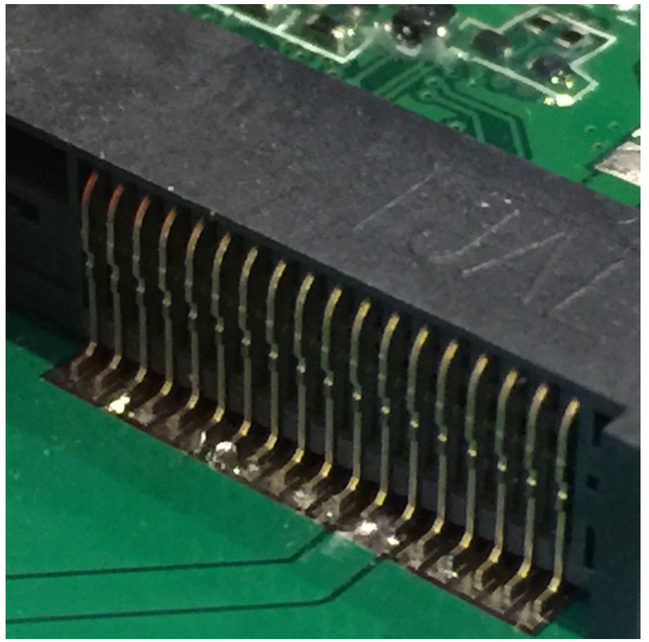
```ruby
# Notice: This is CRuby for RasPi
require "rubyserial"
require "timeout"
sp = Serial.new "/dev/serial0", 9600, 8
loop do
  puts "[command]"
  command = gets
  sp.write command.sub("\n", "\r") # replace LF if needed
  sleep 0.1
  result = ""
  begin
    Timeout.timeout(10) do
      loop do
        line = sp.read(128)
        break if line == "" && result != ""
        result << line
        sleep 0.1
      end
      puts "=> " + result
    end
  rescue Timeout::Error
```

CRuby and serial port

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

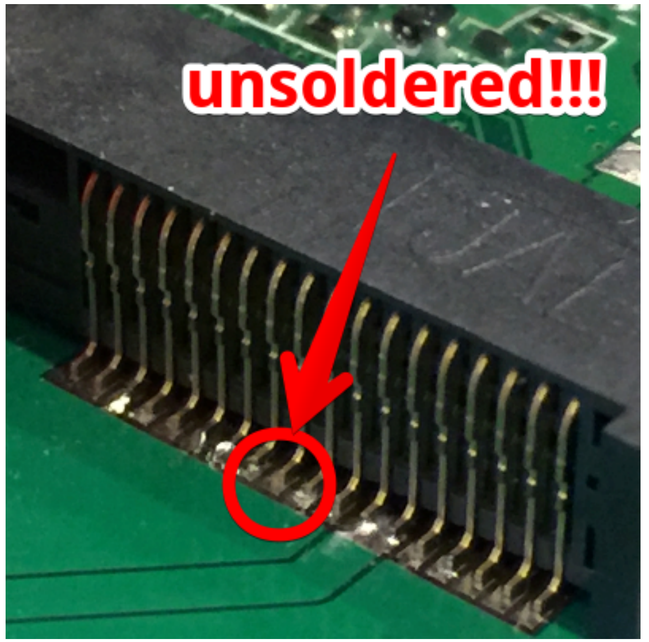
Soldering

Soldering



- It often works even if you leave a pin unsoldered on surface mounting
  - Because the pin touches circuit's plate
  - Then, it will come not to work one day

Soldering



- Discovering this kind of bug is much more difficult than software bug
- My teacher said "All the cause of failure, it is **impatience**"

Chapter 2

mruby/c

What is mruby?

What is mruby?

- github.com/mruby/mruby
- Another implemantation of Ruby for general embedded usage
- Easily combined with system programming like C/C++
- e.g. ngx_mruby is a popular product of mruby
- Good for making command line tool as one-binary executable

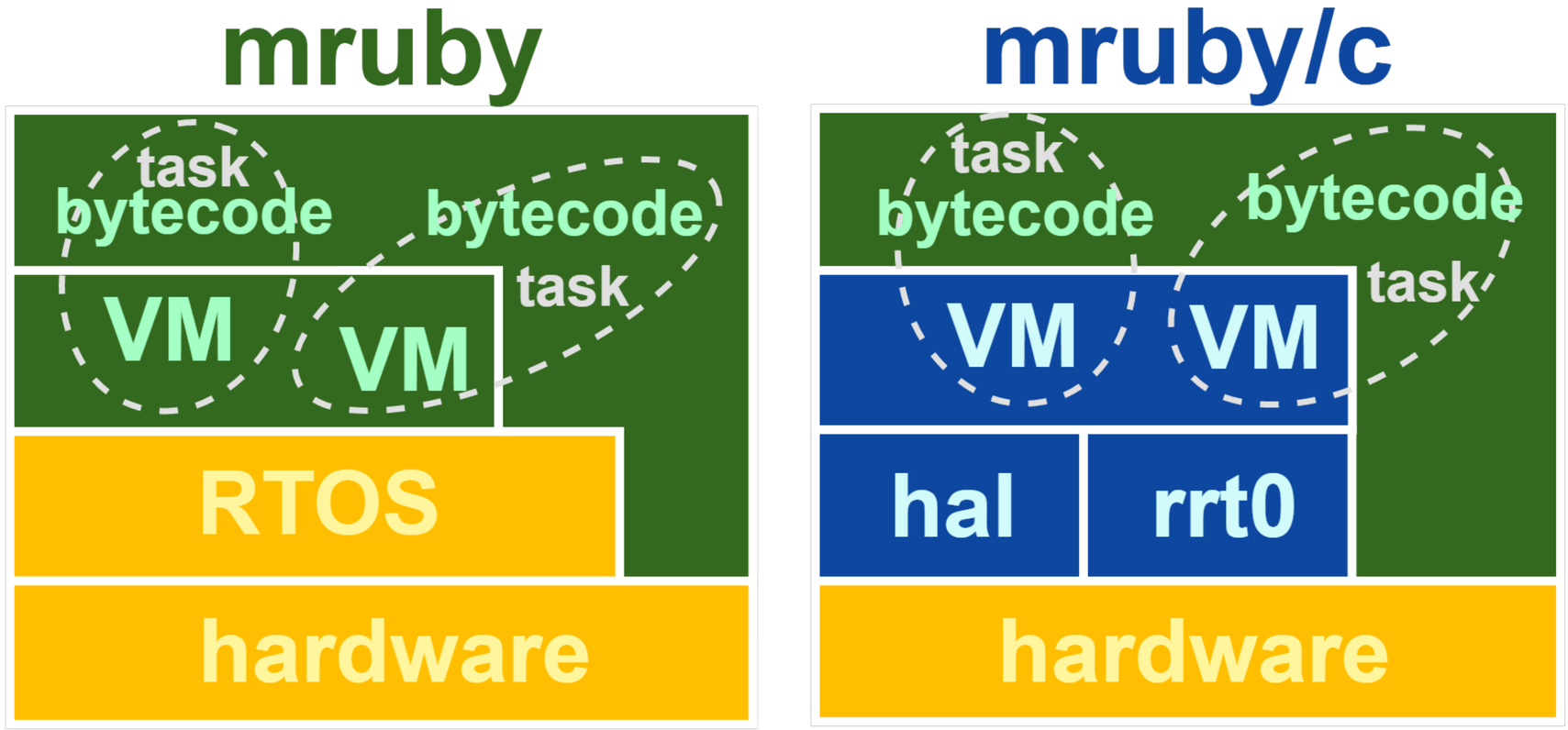
What is mruby/c?

What is mruby/c?

- github.com/mrubyc/mrubyc
- Yet another implementation of mruby
- `/c` symbolizes compact, concurrent and capability
- Especially dedicated to one-chip microcontroller

Bytecode is a common stuff

- They are compiled by `mrbc` (mruby compiler) and each VM execute bytecode



Bytecode?

- A kind of intermediate representation
- mruby bytecode is designed for mruby VM
- mruby VM dynamically interprets the bytecode and processes the application

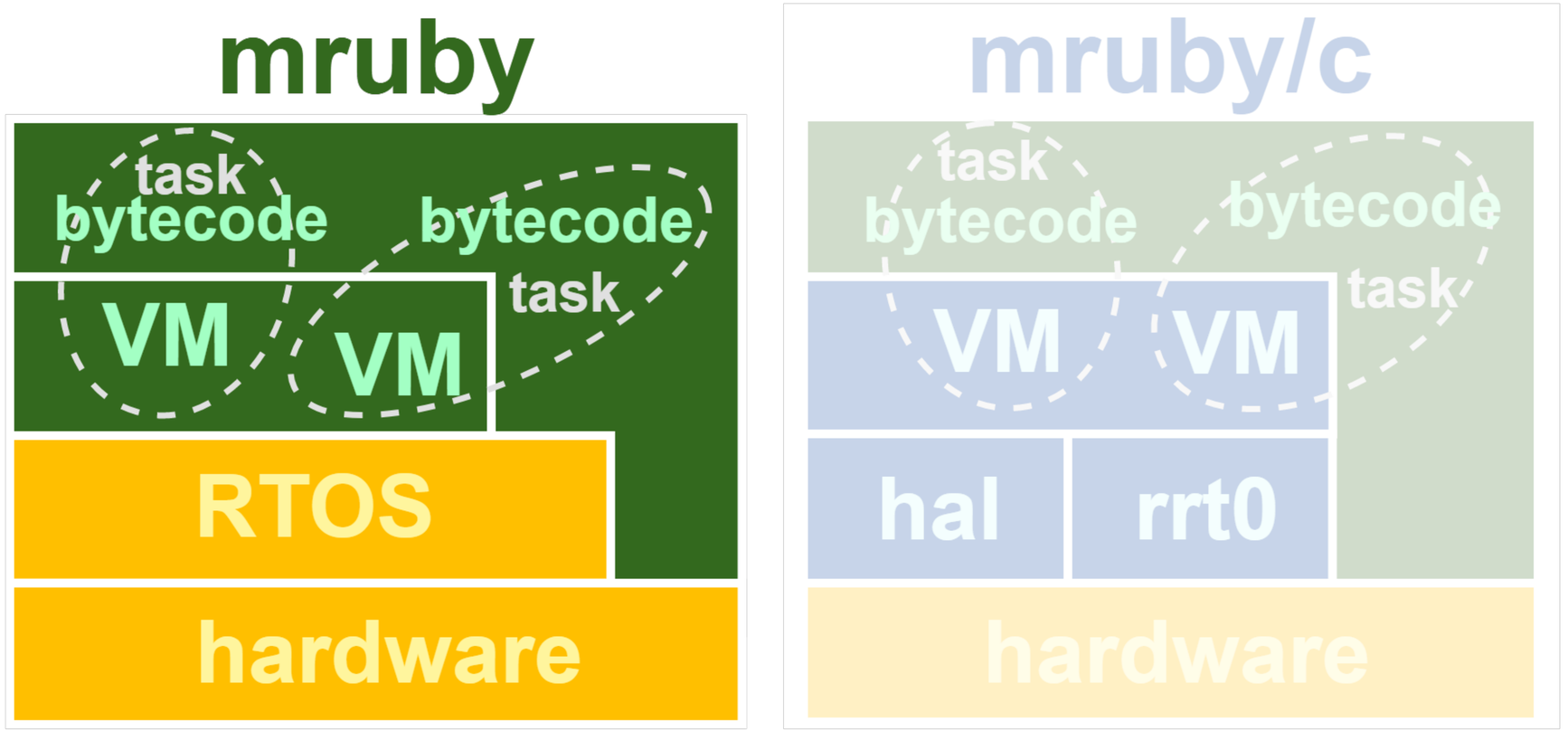
HEX dump of bytecode

Looks like this if you compile
`puts "Hello World!"`

```
5249 5445 3030 3036 9a78 0000 0062 4d41  RITE0006.x...bMA
545a 3030 3030 4952 4550 0000 0044 3030  TZ0000IREP...D00
3032 0000 0060 0001 0004 0000 0000 000c  02...`..........
1001 4f02 002e 0100 0137 0167 0000 0001  ..O......7.g....
0000 0c48 656c 6c6f 2057 6f72 6c64 2100  ...Hello World!.
0000 0100 0470 7574 7300 454e 4400 0000  .....puts.END...
0008                                     ..
```

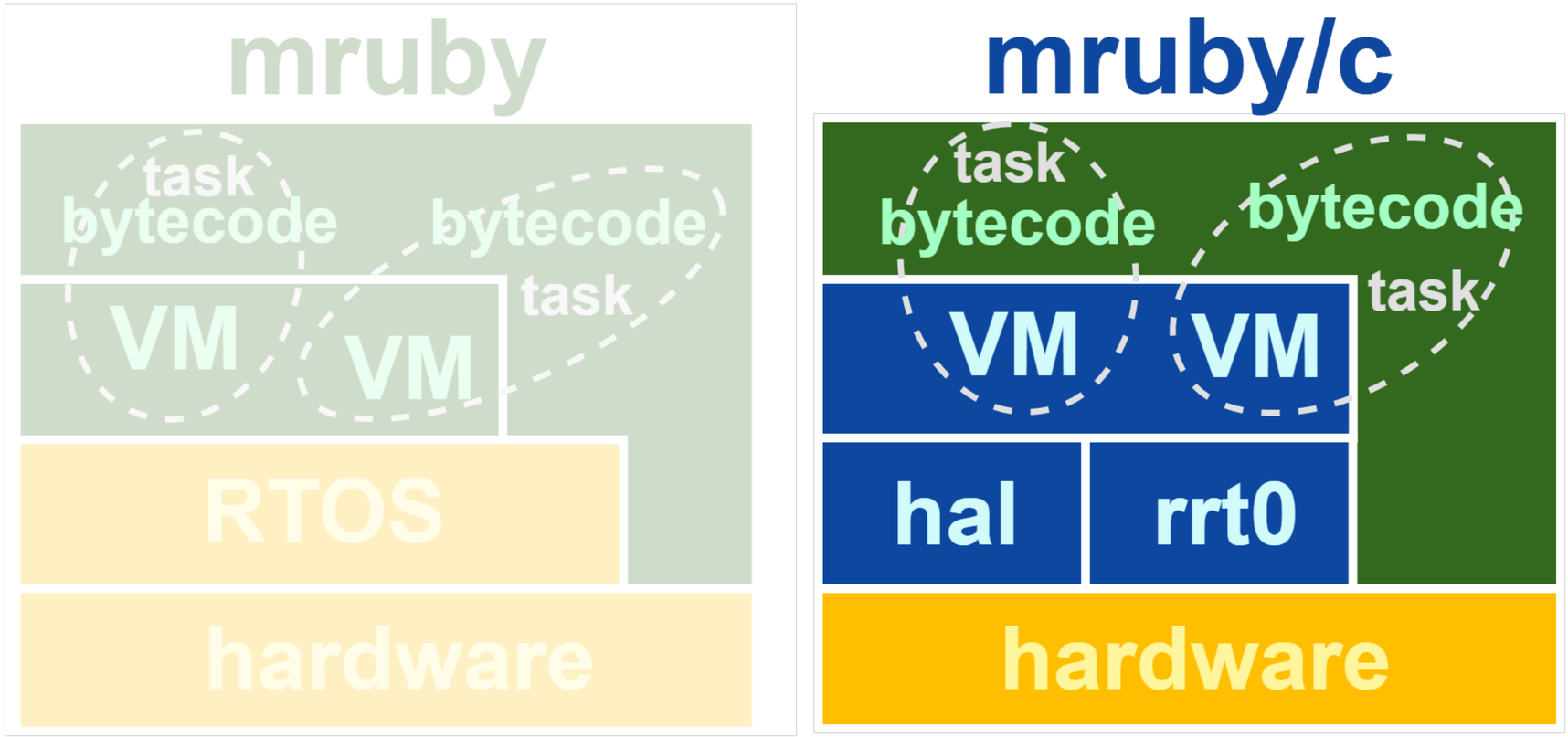
mruby on microcontroller

- RTOS (Real-Time OS) manages mruby VMs in order to realize multi tasking



mruby/c on microcontroller

- mruby/c has its own mechanism to manage multi tasks: **rrt0**



mruby and mruby/c

mruby	mruby/c
v1.0.0 in Jan 2014	v1.0 in Jan 2017
for general embedded software	for one-chip microcontroller
mrbgems	no package manager
RAM < 200KB ...(*)	RAM < 40KB ...(*)

(*)...It depends on the situation

mruby/c's Virtual Machine

- Much smaller than mruby's one
  - That's why mruby/c runs on smaller RAM
- Accordingly, mruby/c has **less** functionality than mruby and CRuby

How **less**?

How **less**? - For example

- mruby/c doesn't have module, hence there is no Kernel module
- Then you must wonder how can you `#puts`?
- In mruby/c, `#puts` is implemented in Object class
- mruby/c doesn't have #send, #eval, and #method_missing, etc.

How **less**? - For example

- The full list of mruby/c's classes
  - Array, FalseClass, Fixnum, Float, Hash, Math, Mutex, NilClass, Numeric, Object, Proc, Range, String, Symbol, TrueClass, VM

Despite the fact,

- No problem in practical usage of microcontroller
- As far as IoT goes, mruby/c is enough Ruby
- We can fully develop firmwares with these features of mruby/c

How does mruby/c work

```
~/project/sample_project
├── main.c
├── mrblib
│   ├── task_1.rb
│   └── task_2.rb
└── src
    ├── task_1.c
    └── task_2.c
```

- task_*.c is compliled bytecode from task_*.rb

How does mruby/c work

```c
/* main.c */
#include "src/task_1.c"
#include "src/task_2.c"
// using 40KB RAM for VM heap in this case
#define MEMORY_SIZE (1024 * 40)
static uint8_t memory_pool[MEMORY_SIZE];
int main(void) {
  mrbc_init(memory_pool, MEMORY_SIZE);
  mrbc_create_task(task_1, 0);
  mrbc_create_task(task_2, 0);
  mrbc_run(); // 2 tasks run concurrently!
  return 0;
}
```

How does mruby/c work

- We can run easily multiple VMs with **concurrency** feature of **rrt0**
- You might be disappointed to know you have to write C
  - Yes, we have to write **main.c**
  - Don't worry, it's almost boilerplate code

Chapter 3

Application code and tools

Application code

github.com/hasumikin/co2_demo

Application code

```
~/project/co2_demo
 ├── main.c
 └── mrblib
      ├── loops
      │    ├── primary.rb
      │    └── secondary.rb
      └── models
           ├── co2.rb
           ├── led.rb
           └── thermistor.rb
```

Application code

```
# loops/primary.rb
$co2 = Co2.new      # Makes it global so that secondary task can use it
$thermistor = $thermistor.new
led = Led.new(19)  # 19 is a pin number which LED connects
while true
  co2 = $co2.concentrate
  if co2 > 2000     # When CO2 reaches fatal level
    5.times do      # Turning LED on and off
      led.turn_on
      sleep 0.1
      led.turn_off
      sleep 0.1
    end
  else              # Safe level
    led.turn_off    # Turns off
    sleep 1
  end
end
```

Application code

How does **Led#trun_on** work?

Application code

```ruby
# 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) # high
  end
  def turn_off
    gpio_set_level(@pin, 0) # low
  end
end
```

Application code

```c
/* a part of main.c */
#include "models/led.c"
static void c_gpio_init_output(mrbc_vm *vm, mrbc_value *v, int argc) {
  int pin = GET_INT_ARG(1);
  // Function of microcontroller's library
  gpio_set_direction(pin, GPIO_MOĐE_OUTPUT);
}
static void c_gpio_set_level(mrbc_vm *vm, mrbc_value *v, int argc){
  int pin = GET_INT_ARG(1);
  int level = GET_INT_ARG(2);
  gpio_set_level(pin, level); // Function of microcontroller's library
}
int main(void){
  ...
  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);
  ...
}
```

Application code

```
/* a part of main.c */
#include "models/co2.c"
static void c_get_co2(struct VM *vm, mrbc_value v[], int argc){
  uint8_t command[] = { // Command to take CO2
    0xFF, 0x01, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79
  };
  uart_write_bytes(uart_num, (const char*)command, 9);
  // ↑Write then ↓Read data
  uint8_t data[10];
  int length = 0;
  length = uart_read_bytes(uart_num, data, length, 10);
  mrbc_value array = mrbc_array_new( vm, 9 ); // mrubyc's variable
  for( int i = 0; i < 9; i++ ) {
    mrbc_value value = mrbc_fixnum_value(data[i]);
    mrbc_array_set( &array, i, &value ); // Adding a value to array
  }
  SET_RETURN(array); // Returning the array object to mruby
}
int main(void){
  ...
  mrbc_define_method(0, mrbc_class_object, "get_co2", c_get_co2);
  ...
}
```

Application code

```
# models/co2.rb
class Co2
  def concentrate
    res = get_co2
    # checks if the sensor works
    if res[0] == 255 && res[1] == 134
      res[2] * 256 + res[3]
    else
      0
    end
  end
end
```

Application code

- Trivial tip to reduce memory usage
  - C function can return String instead of Array
  - `mrbc_array_new` will allocate larger memory than `mrbc_string_new`
  - So, you can use String instead of Array if memory becomes short
  - Remember STRING of C is just an array of char and '\0'

Application code

```
# loops/secondary.rb
http_client = HttpClient.new("http://data.server")
while true
  co2 = $co2.concentrate
  temperature = $thermistor.temperature
  if co2 > 0 # No trouble
    data = "co2=#{co2}&temperature=#{temperature}"
    http_client.post(data)
    sleep 180
  else         # A trouble happens?
    sleep 3    # Or you can retry
  end
end
```

RubyConf 2019
Nashville

Dev tools for mruby/c

Dev tools for mruby/c

- mrubyc-utils
- mrubyc-test
- mrubyc-debugger

RubyConf 2019
Nashville

Dev tools for mruby/c

I am trying to make mruby/c development

Rubyish

Rubyish? - IMHO

- Unix/Linux
- Command line
- No-IDE (as far as possible)

Rubyish? - IMHO

- Unix/Linux
- Command line
- No-IDE (as far as possible)

Taking full advantage of
our Laptop and Ruby World

mrubyc-utils

- github.com/hasumikin/mrubyc-utils
- One-binary tool made with mruby
- Helps to install boilerplate of application
- Utility subcommands like...

mrubyc-utils

```
$ mrubyc-utils classes
- Array
- FalseClass
- Fixnum
- Float
- Hash
- Math
- Mutex
- NilClass
- Numeric
- Object
- Proc
- Range
- String
- Symbol
- TrueClass
- VM
```

mrubyc-utils

```
$ mrubyc-utils methods --class=array
Array
- +                     - inspect
- <<                    - join
- []                    - last
- []=                   - length
- at                    - max
- clear                 - min
- collect               - minmax
- collect!              - new
- count                 - pop
- delete_at             - push
- dup                   - shift
- each                  - size
- each_index            - to_s
- each_with_index       - unshift
- empty?                < Object
- first                   - !
- index                   ...
```

mrubyc-test

- github.com/mrubyc/mrubyc-test
- Unit testing framework
- A RubyGem implemented with CRuby
- Supports stub and mock
- Official test tool of mruby/c dev team

mrubyc-test

- Gathers information from app and test code
- Internally generates stub and mock methods
- Makes all-in-one script: **test.rb**

```
class Sample
  def hoge
    "fuga"
  end
end
```

```
class SampleTest < MrubycTestCase
  def some_case
    obj = Sample.new
    assert_equal obj.hoge, "fuga"
  end
end
```

```
SampleTest#some_case
  assertion : assert_equal
  result    : fuga
Finished
1 examples, 0 failures
```

mrubyc-debugger

- github.com/hasumikin/mrubyc-debugger
- Debugger for mutiple infinite loops

DEMO

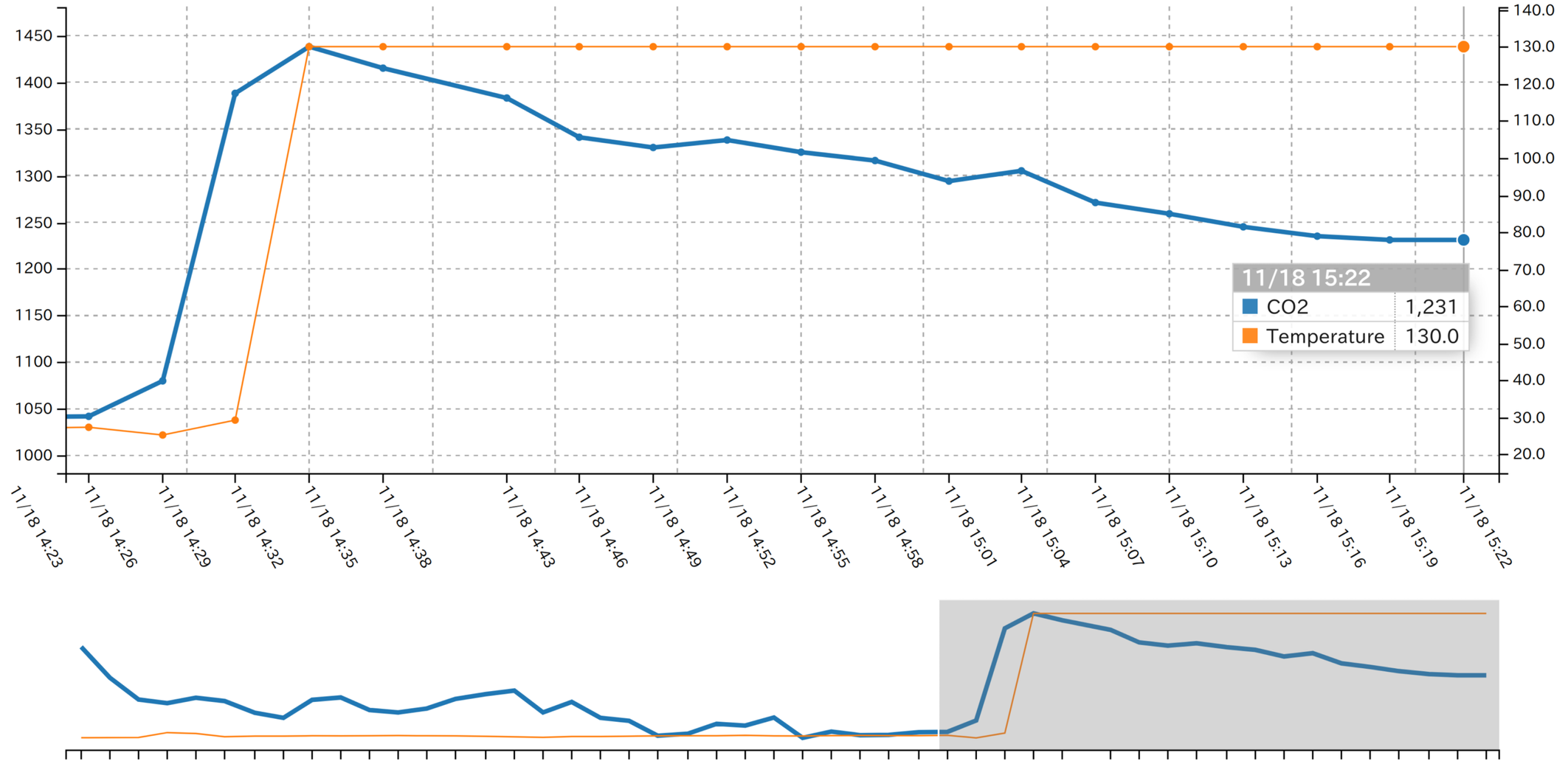
github.com/hasumikin/
mrubyc-debugger

DEMO

How is CO_2 going?

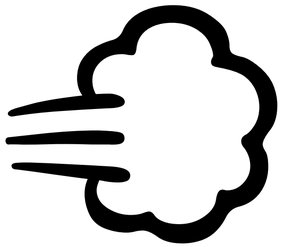
RubyConf 2019
Nashville

DEMO (added after the Conf)



Conclusion

Conclusion

You should refresh air 

RubyConf 2019
Nashville

Thank you!