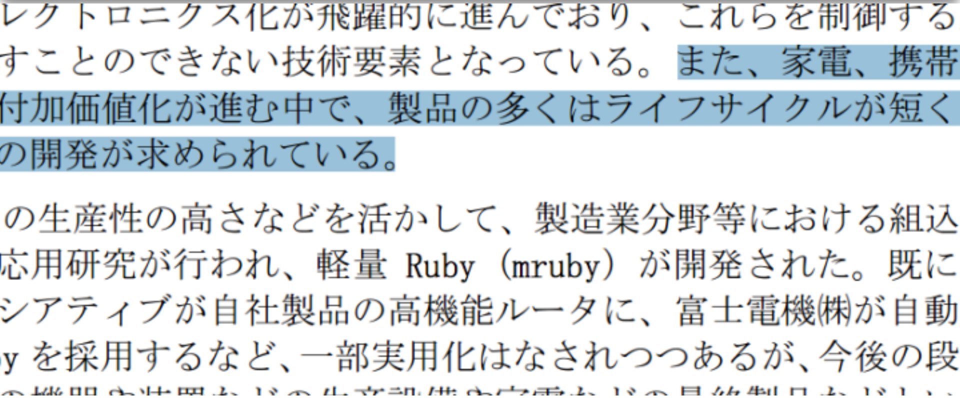
Embedding mruby into C and an actual example

P.S.V.R
（阿里巴巴 － 孝达）

mruby

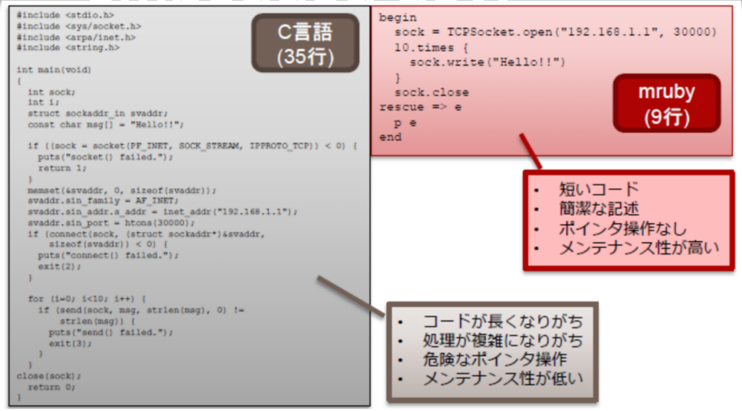
m = **m** atz' + e **m** beddable + **m** inimalistic + **m** odular

why mruby?


レクトロニクス化が飛躍的に進んでおり、これらを制御する
すことのできない技術要素となっている。また、家電、携帯
付加価値化が進む中で、製品の多くはライフサイクルが短く
の開発が求められている。

の生産性の高さなどを活かして、製造業分野等における組込
応用研究が行われ、軽量 Ruby（mruby）が開発された。既に
シアティブが自社製品の高機能ルータに、富士電機㈱が自動
y を採用するなど、一部実用化はなされつつあるが、今後の段

purpose: boost productivities of embedded dev.



C言語
(35行)

```c
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <string.h>

int main(void)
{
  int sock;
  int i;
  struct sockaddr_in svaddr;
  const char msg[] = "Hello!!";

  if ((sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {
    puts("socket() failed.");
    return 1;
  }
  memset(&svaddr, 0, sizeof(svaddr));
  svaddr.sin_family = AF_INET;
  svaddr.sin_addr.s_addr = inet_addr("192.168.1.1");
  svaddr.sin_port = htons(30000);
  if (connect(sock, (struct sockaddr*)&svaddr,
      sizeof(svaddr)) < 0) {
    puts("connect() failed.");
    exit(2);
  }

  for (i=0; i<10; i++) {
    if (send(sock, msg, strlen(msg), 0) !=
        strlen(msg)) {
      puts("send() failed.");
      exit(3);
    }
  }
close(sock);
  return 0;
}
```

- コードが長くなりがち
- 処理が複雑になりがち
- 危険なポインタ操作
- メンテナンス性が低い

mruby
(9行)

```ruby
begin
  sock = TCPSocket.open("192.168.1.1", 30000)
  10.times {
    sock.write("Hello!!")
  }
  sock.close
rescue => e
  p e
end
```

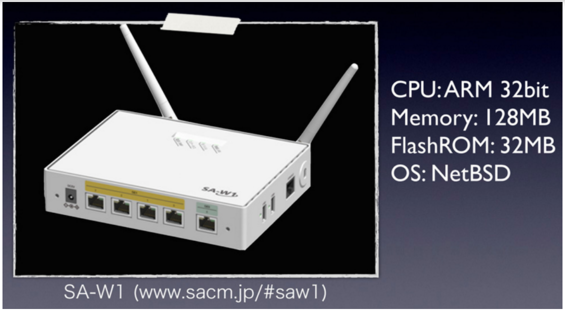
- 短いコード
- 簡潔な記述
- ポインタ操作なし
- メンテナンス性が高い

who is using it?

展示宣传板（从左开始，轻量Ruby，嵌入式离岸外包，培训教育）

- 富士電機（杭州）軟件有限公司

who is using it?


CPU: ARM 32bit
Memory: 128MB
FlashROM: 32MB
OS: NetBSD

SA-W1 (www.sacm.jp/#saw1)

- IIJ’s router products c.f. https://github.com/iij/mruby

who should have been using it?

- could have been more energy-efficient and inexpensive and easy-to-maintain via mruby…

Actually…

- mruby is not only for the EMBEDDED WORLD
- mruby is for the ENTIRE C WORLD
- just knowing C is enough to get you going, you don’t necessarily need embedded devices or EE degrees

who is also using it?

- `mod_mruby`:

https://github.com/matsumoto-r/mod_mruby

- `ngx_mruby`:

https://github.com/matsumoto-r/ngx_mruby

who is also using it?

- `php-mruby`:

https://github.com/chobie/php-mruby

- `ab-mruby`:

https://github.com/matsumoto-r/ab-mruby

who is also using it?

- `go-mruby`:

https://github.com/mitchellh/go-mruby

“反主为客”

- MRI: 主Ruby 客C
- mruby: 主C 客Ruby

“反主为客”

- MRI:

only **1** VM per process, then add features to the VM

- mruby:

N VMs per process (could be one VM per thread, no need to lock), then call VMs from C

how to embed

- open `mrb_state`
- define business-domain ruby-classes and methods in the compile-time
- dynamically invoke ruby-classes and methods in the run-time
- close `mrb_state`

“mrb_state”

- NO GLOBAL VARIABLES, everything are inside “mrb_state”
- refereed to by almost all functions
- [**demo**] see “typedef struct mrb_state”

“mrb_value”

- could store pointer, integer, float, etc
- set: `mrb_fixnum_value`, `mrb_float_value`, …
- get: `mrb_fixnum`, `mrb_float`, …
- [**demo**] see “typedef struct mrb_value” (x2 imple.)

“mrb_funcall”

```
obj = mrb_funcall(mrb, obj, "inspect", 0);
```

- call into instance method of a ruby-object
- e.g.

debian

there is no file I/O

- the underlying machine might not have a file system at all
- it’s a seperate module **mruby-io**, if you want it
- event STD I/O is configurable
- [**demo**] see “#ifdef ENABLE_STDIO”

“mrb_define_module”

```
mrb_ofpsvr = mrb_define_module(mrb, "Ofpsvr");
```

- define business-domain ruby-classes
- e.g.

“mrb_define_module_function”

```
mrb_define_module_function(mrb, mrb_ofpsvr, "uid", ofpsvr_uid, MRB_ARGS_REQ(1));
```

- define business-domain ruby-methods
- e.g.

“mrb_load_string”

```
mrb_load_string(mrb, "puts \"いまは#{Time.now}です。\"");
```

- define business-domain ruby-methods
- e.g.

Exception Handling

```
if (mrb->exc) {
  mrb_value exception = mrb_obj_value(mrb->exc);
  mrb->exc = 0;
}
```

- do not forget to check `mrb-exc` when running unpredictable code!

an actual example

- [**demo**] https://github.com/pmq20/ofpsvr