

Rubyボードハンズオン テキスト

2016/3/19修正版

Wakayama. rb
山本三七男(たろサ)

目次

ハード仕様	3
ピンマップ	4
基本ソフト仕様	5
ハンズオンで使用するRubyの構文	7
仮想COMポートドライバインストール	9
Rubyボードの接続	11
Rubicの使い方	13

ハンズオン

1. LEDチカチカ	29
2. Hello World! とLEDチカチカ	30
3. スイッチ	31
4. ブザー	33
5. ブザーとスイッチ	35
6. 光センサとAD変換	37
7. 光センサとAD変換2	39
8. 光センサとブザー	41

プログラムの入手方法	43
------------	----

ハード仕様

MCU

32ビットCPU RX63N(100ピン)

96MHz

FlashROM : 1Mバイト

RAM : 128Kバイト

データ用Flash : 32Kバイト

ボード機能

USBファンクション端子 (micro-B)

LED 1個

I/Oピン 20ピン

シリアル 3個(+1個可能)

SPI 1個

A/D 4個

RTC

I2C、PWM、Servoは自由割当てです。

リセットボタン

電 源

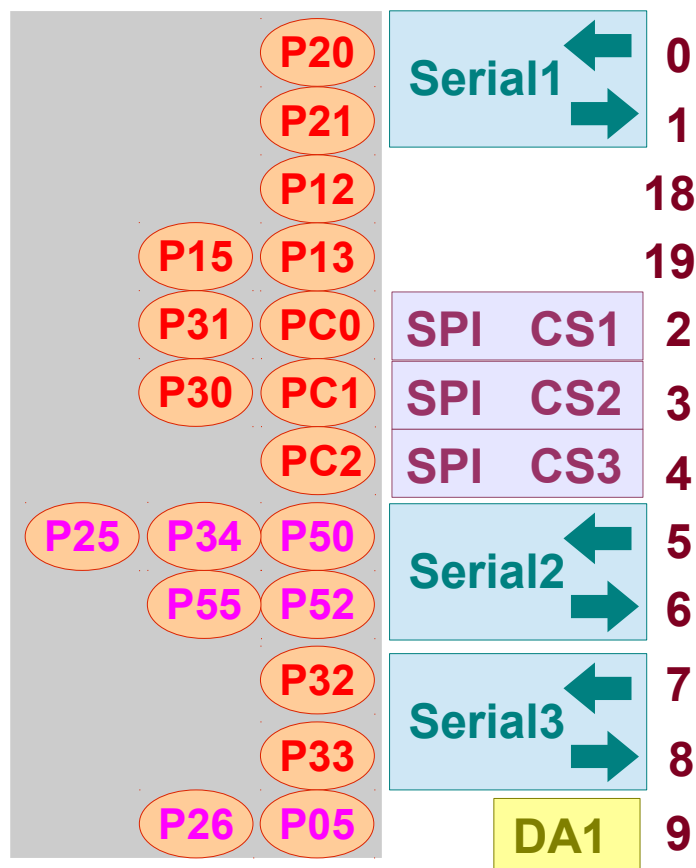
5V (USBバスパワー)

サイズ

50×18mm

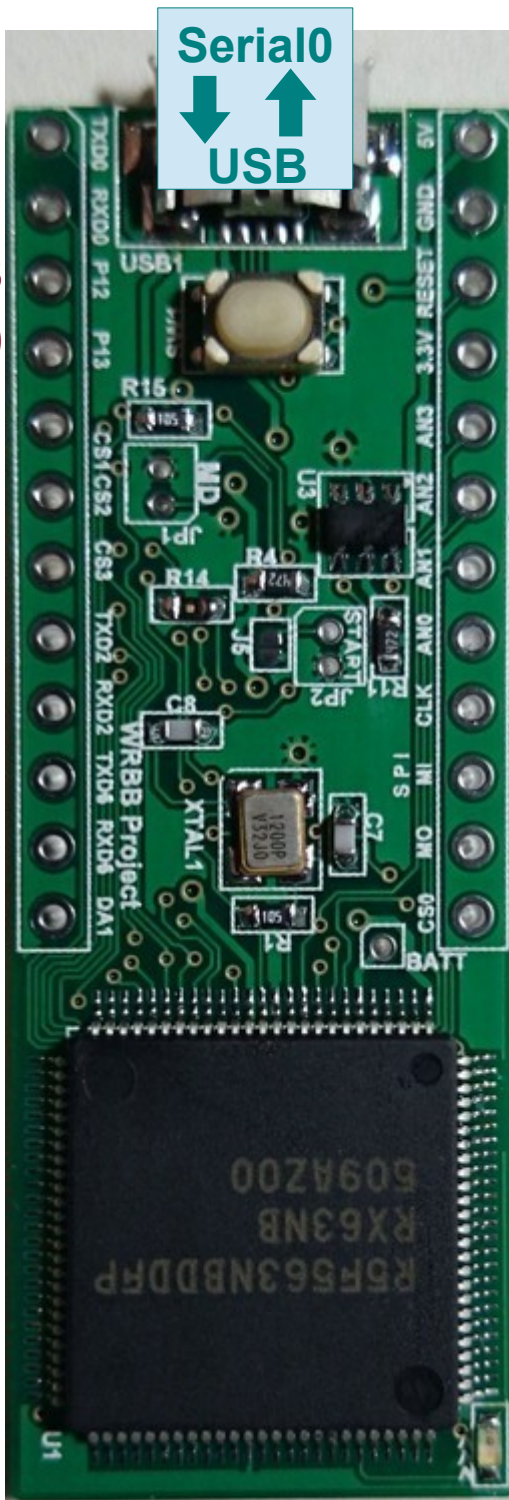


RX63Nピン番号



赤文字ピン番は
5Vトレラント

WakayamaRBボード ピンマップ



Serial0
↓ ↑
USB

5V

GND

◀ RESET

3.3V

17 A3

16 A2

15 A1

14 A0

13 CLK

12 ← SPI

11 →

10 CS0

RX63Nピン番号

P43 PE1

P42 PB5

P41 PB3

P40 P27

PC5

PC7

PC6

PC4

Ver. ARIDA 4

基本ソフト仕様

カーネルクラス

- pinMode(pin, mode)
- digitalRead(pin)
- digitalWrite(pin, value)
- analogRead(number)
- pwm(pin, value)
- pwmHz(value)
- analogDac(value)
- delay(value)
- millis()
- micros()
- led(sw)

システムクラス

- System.exit()
- System.setrun(filename)
- System.version(r)
- System.push(address, buf, length)
- System.pop(address, length)
- System.fileload()
- System.reset()

ファイルクラス

- MemFile.open(number, filename[, mode])
- MemFile.close(number)
- MemFile.read(number)
- MemFile.write(number, buf, len)
- MemFile.seek(number, byte)
- MemFile.copy(src, dst[, mode])

シリアルクラス

- Serial.begin(number, bps)
- Serial.setDefault(number)
- Serial.print(number, string)
- Serial.println(number, string)
- Serial.read(number)
- Serial.write(number, buf, len)
- Serial.available(number)
- Serial.end(number)

I2Cクラス

- I2c.sdasci(sda, scl)
- I2c.write(id, address, data)
- I2c.read(id, addressL[, addressH])
- I2c.begin(id)
- I2c.lwrite(data)
- I2c.end()
- I2c.request(id, count)
- I2c.lread()
- I2c.freq(Hz)

サーボクラス

- Servo.attach(ch, pin[, min, max])
- Servo.write(ch, angle)
- Servo.us(ch, us)
- Servo.read(ch)
- Servo.attached(ch)
- Servo.detach(ch)

基本ソフト仕様

リアルタイムクロッククラス

Rtc.begin()

Rtc.setTime(Array)

Rtc.getTime()

ハンズオンで使用する Rubyの構文

Rubyの構文 いろいろな書き方ができます

【繰り返しループ】

```
for i in 1..10 do
  Serial.print(0, i.to_s) #-> 1~10
end
```

```
10.times do|i|
  Serial.print(0, i.to_s) #-> 0~10
end
```

```
while 条件文 [do]
  処理
end
```

【条件分岐】

```
if 条件文 then
  処理
elsif 条件文 then
  処理
else
  処理
end
```

【条件演算子】 C言語と同じです

a == b	bがaに等しい
a != b	bがaに等しくない
a > b	bよりaが大きい
a >= b	bよりaが大きいかわ等しい
a < b	bよりaが小さい
a <= b	bよりaが小さいかわ等しい

【if修飾子】

```
Serial.print(0, "Hello") if a > 10 #-> 真のとき
```

【unless修飾子】

```
Serial.print(0, "Hello") unless a > 10 #-> 偽のとき
```


USB仮想COMポートドライバ インストール

Windowsユーザのみ、 仮想COMポートドライバのインストールが必要です。

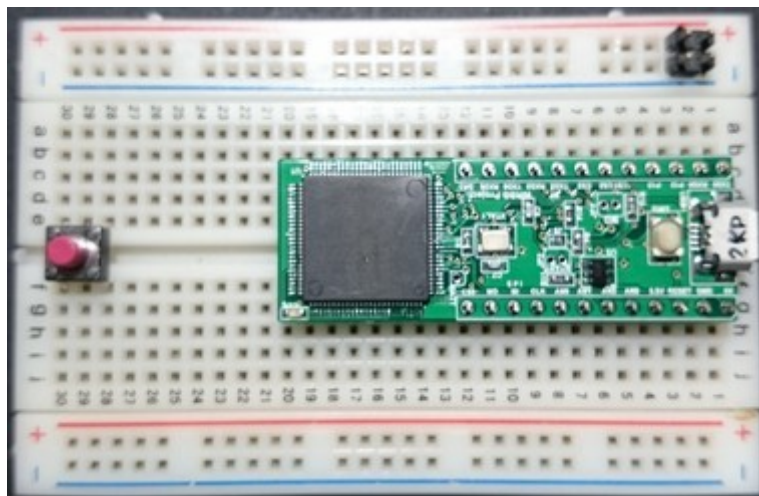
Rubyボードは、特殊電子回路(株)さんの無償版FreeRXduinoライブラリを使用して製作されています。

特殊電子回路(株)さんのFreeRXduinoホームページから
<http://rx.tokudenkairo.co.jp/freesoft.html>

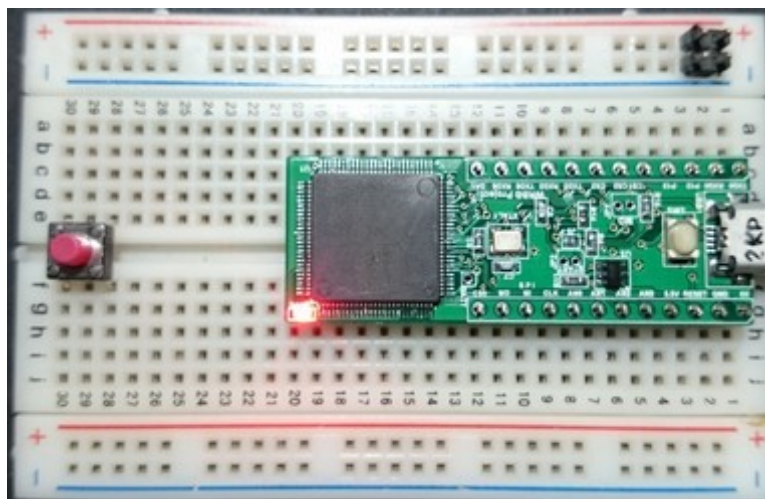
[GR-SAKURA用USB仮想COMポートドライバ](#)をダウンロードしてインストールしてください。

Rubyボードの接続

Rubyボードの接続



①WRBボードを差し込む



ここで、Windowsは仮想COMポートドライバをインストールします。

②USBを接続する。

Rubicの使い方

Rubic
kimu_shu 提供
★★★★★ (0) | [デベロッパー ツール](#) | ユーザー数: 38 人

概要 | レビュー | 関連アイテム

↑ アプリを起動

お使いの端末に対応

Prototyping tool for embedded-boards with Ruby language

Rubic(ルービック)は、Ruby言語を用いた組み込みボードのプロトタイピング環境です。

Rubyスクリプト入力画面でプログラムを書いたら、接続しているボードを選択して[Run]ボタンを押すだけで、プログラムがボード上で走り出します。組み込みボード側のRuby実行エンジンには、mruby 1.1.0を採用しています。

詳しい使い方は
<https://github.com/kimushu/rubic/tree/v0.2.0>をご覧ください。

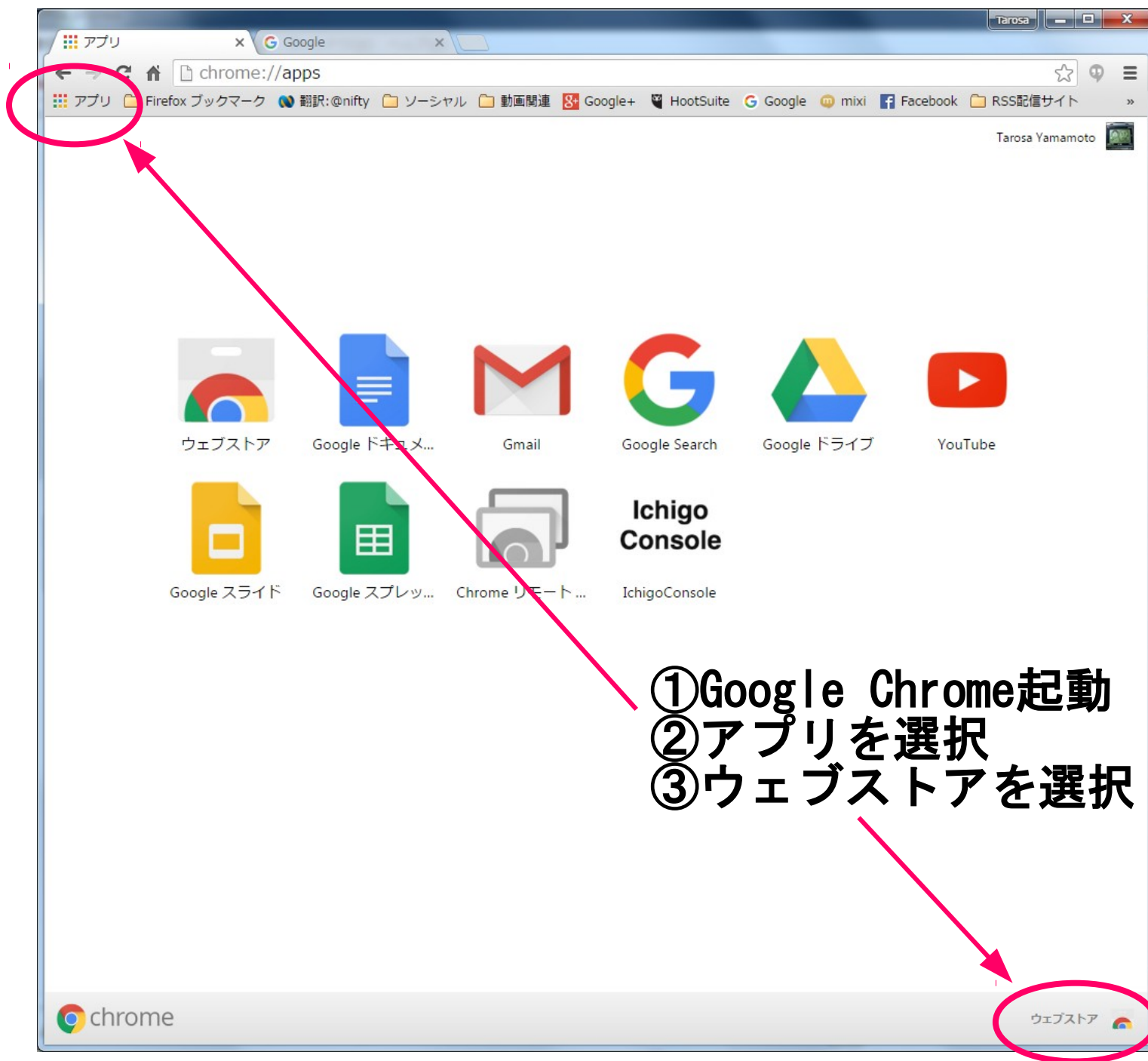
🏠 [ウェブサイト](#)
🚨 [不正行為を報告](#)
バージョン: 0.2.0
更新日: 2015年12月10日
サイズ: 1.27MiB
言語: 日本語

このアプリを使った人は次も使っています:

Postman ★★★★★ (4050)	ペーシランクのステータス ★★★★★ (205)	PR	ページスピード ★★★★★ (60)	SEO Speed Script	Online Alarm Clock ★★★★★ (11)
--------------------------------	------------------------------------	-----------	------------------------------	-------------------------	---

<https://chrome.google.com/webstore/detail/rubic/mgbcgagfggopcpbbfgididddbnhhnhjp>

Rubicのインストール

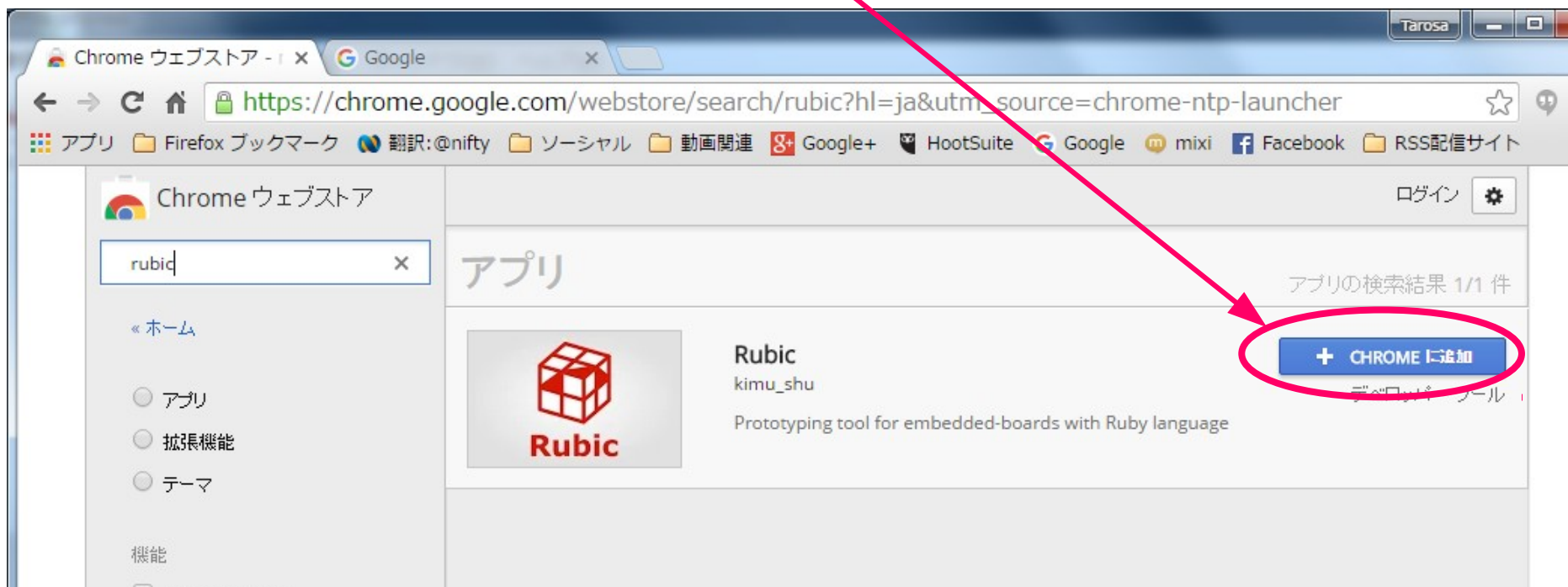


Rubicのインストール



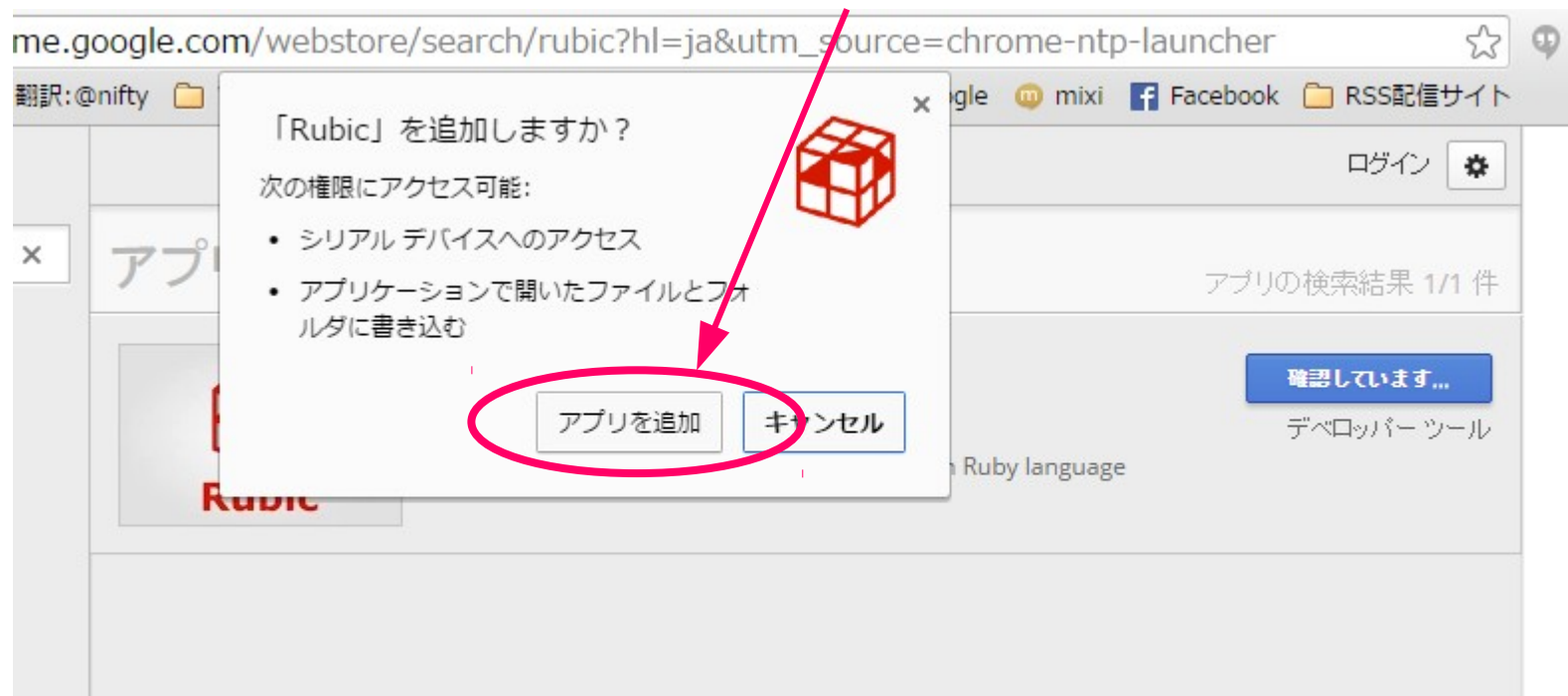
Rubicのインストール

①CHROMEに追加する

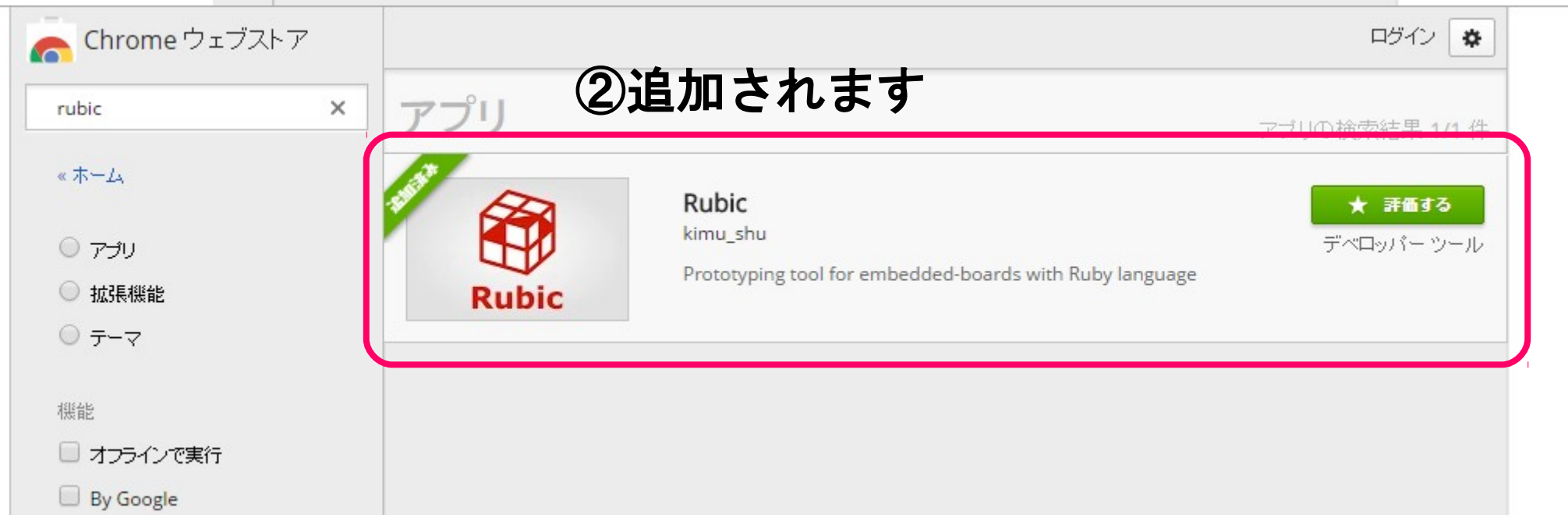


Rubicのインストール

①アプリを追加を選ぶ

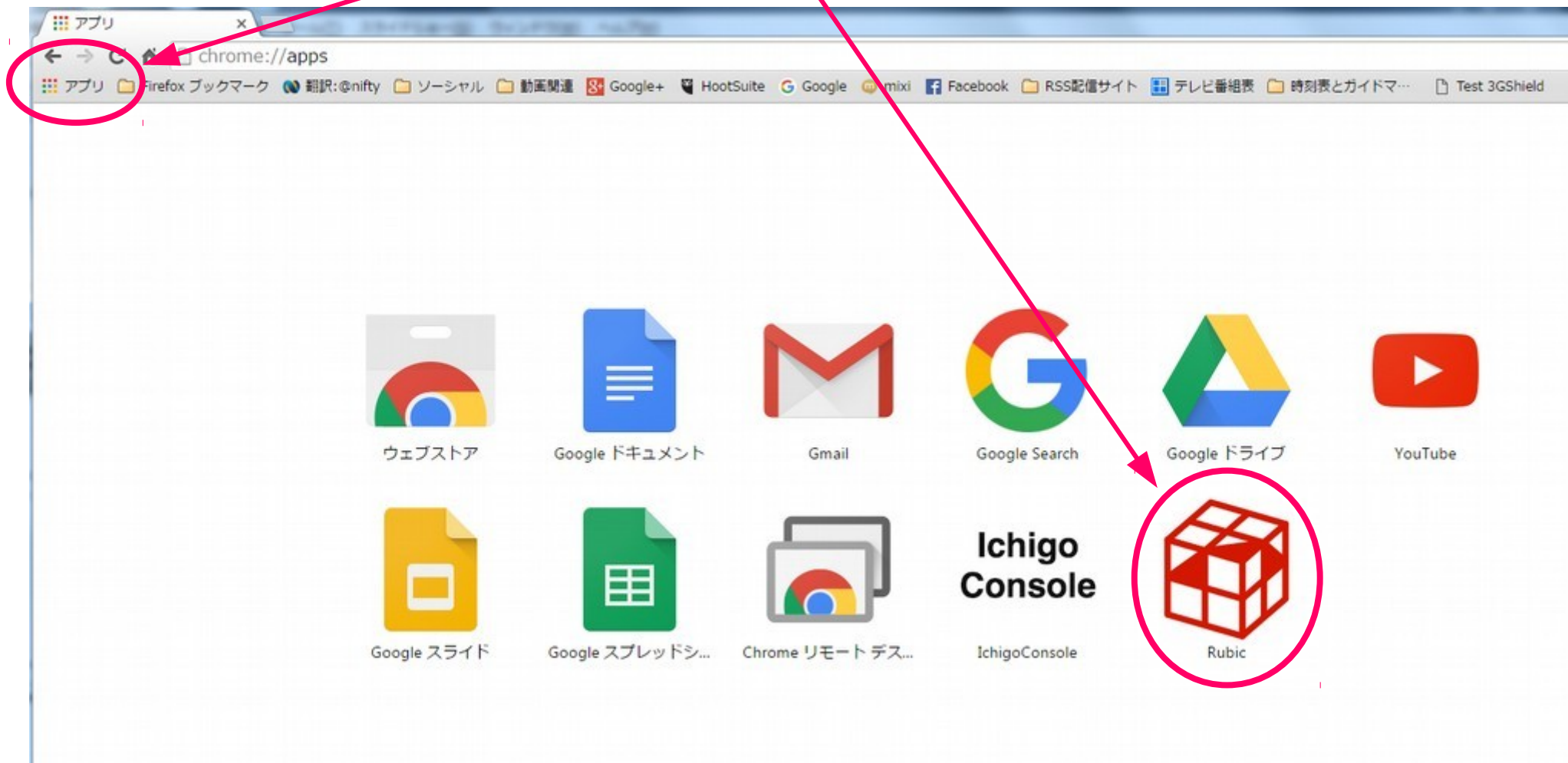


②追加されます

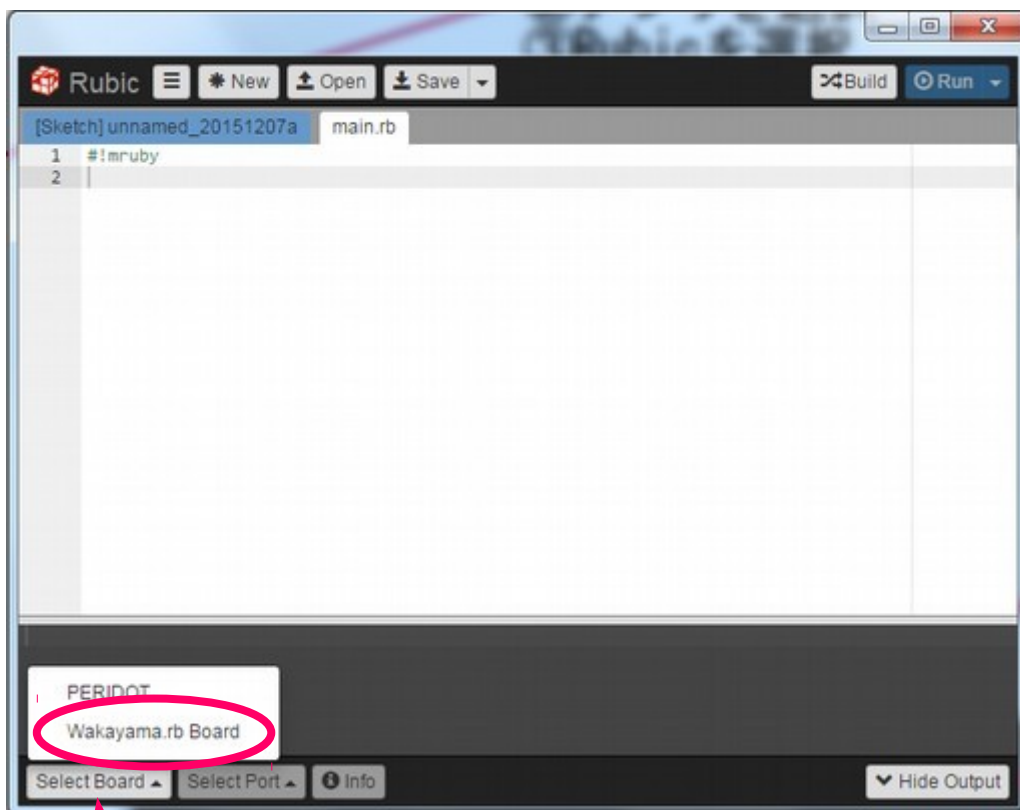


Rubicの起動

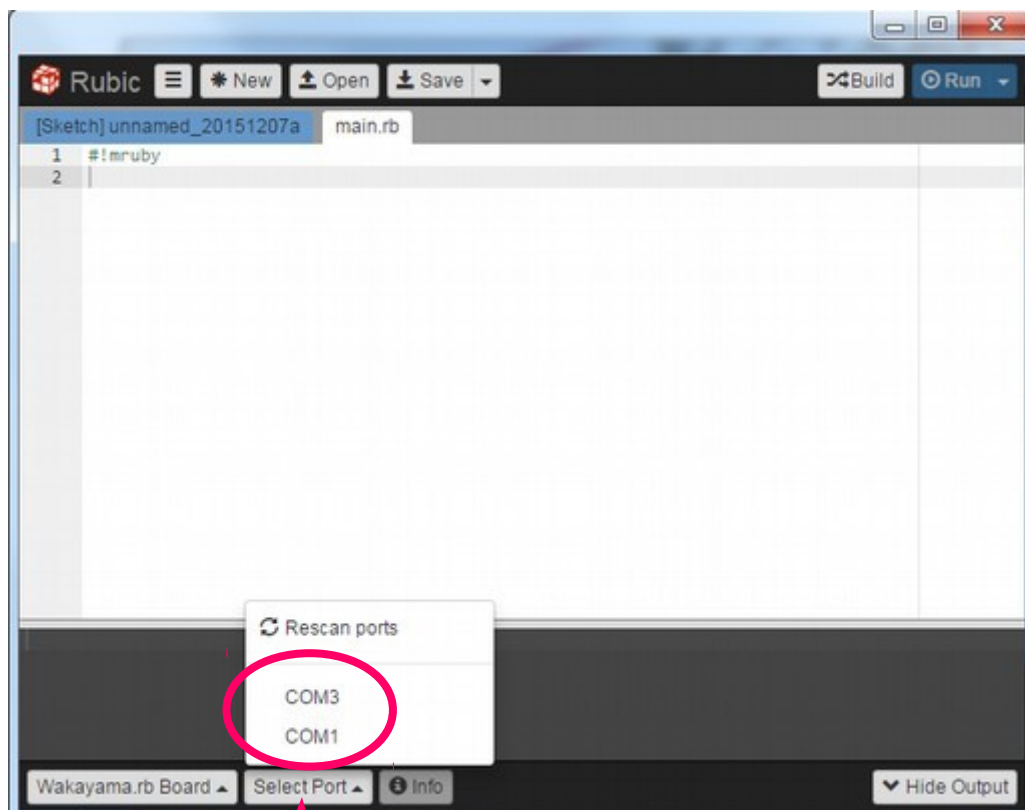
- ①Google Chrome起動
- ②アプリを選択
- ③Rubicを選択



Rubic: ボードとCOMポートの設定

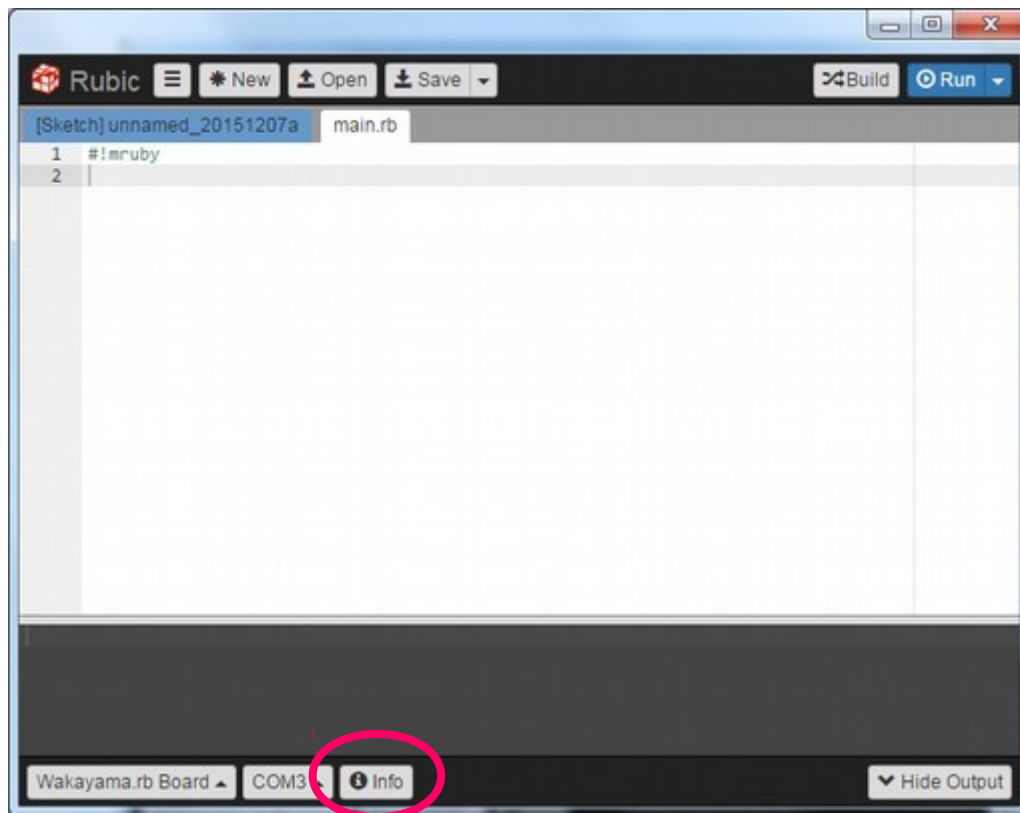


①Wakayam. rb Boardを選択

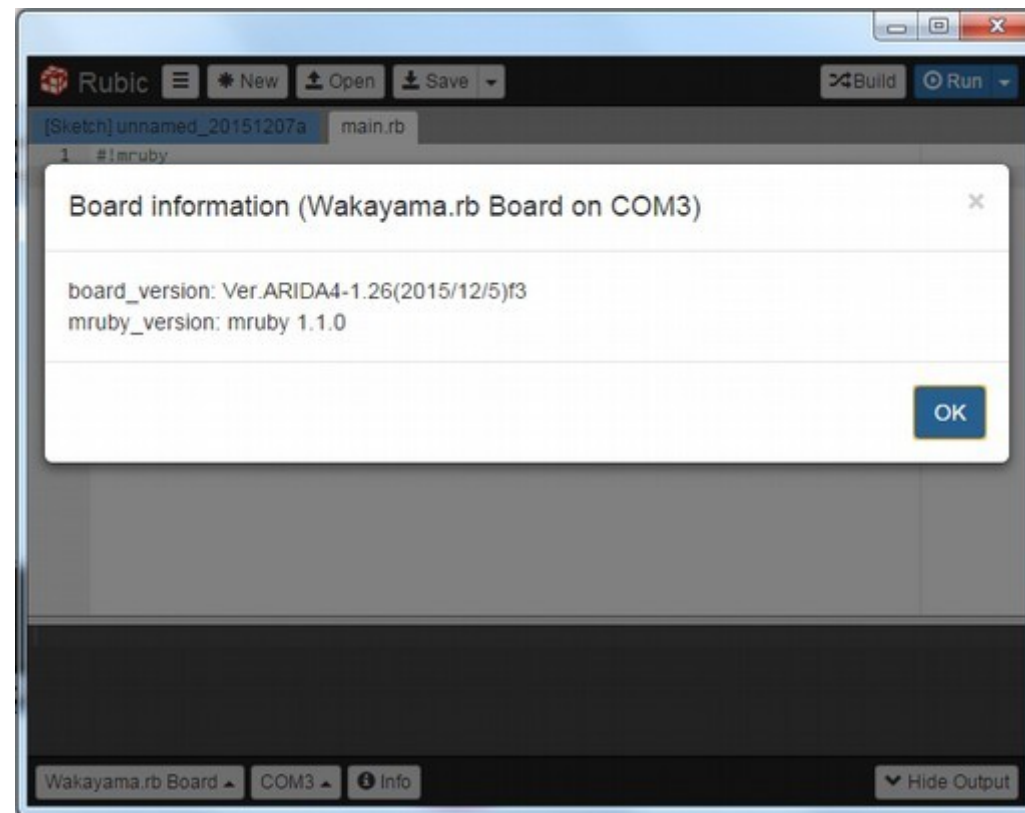


②COMポートを選択

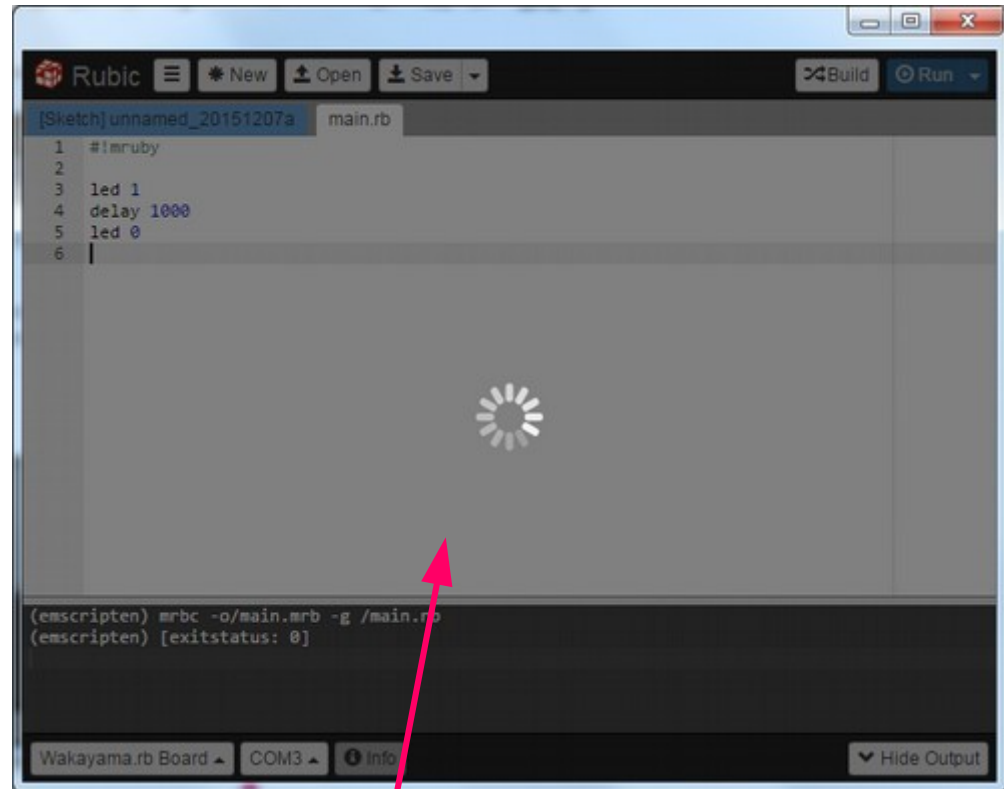
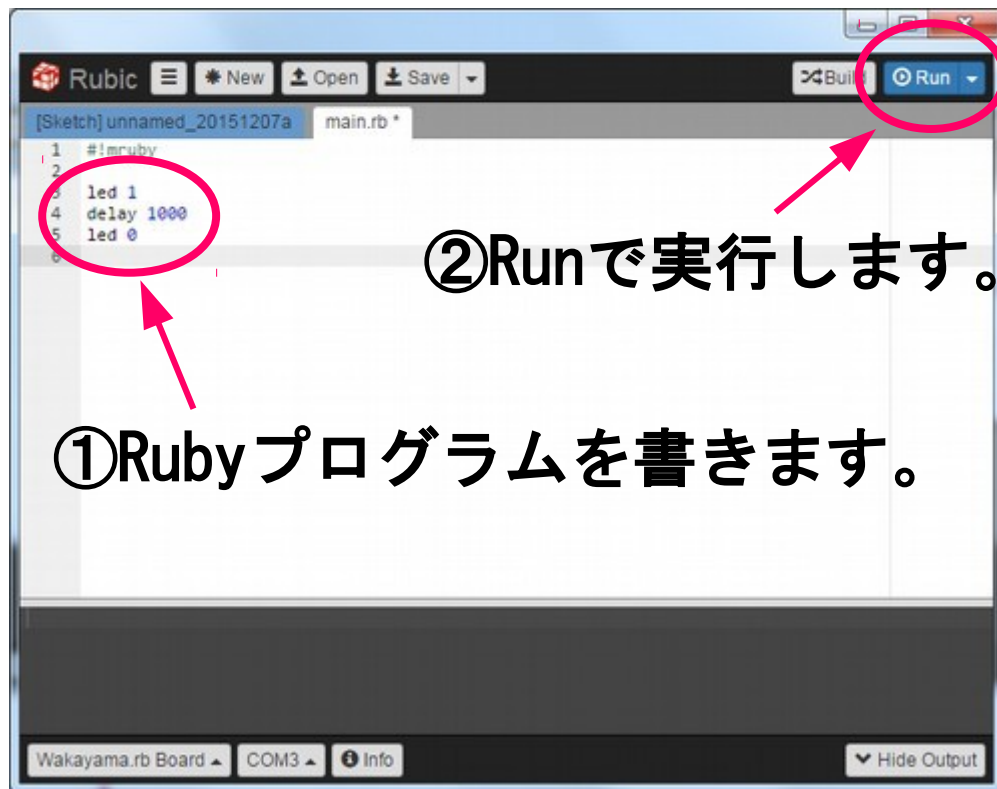
Rubic: インフォメーション表示



infoクリック

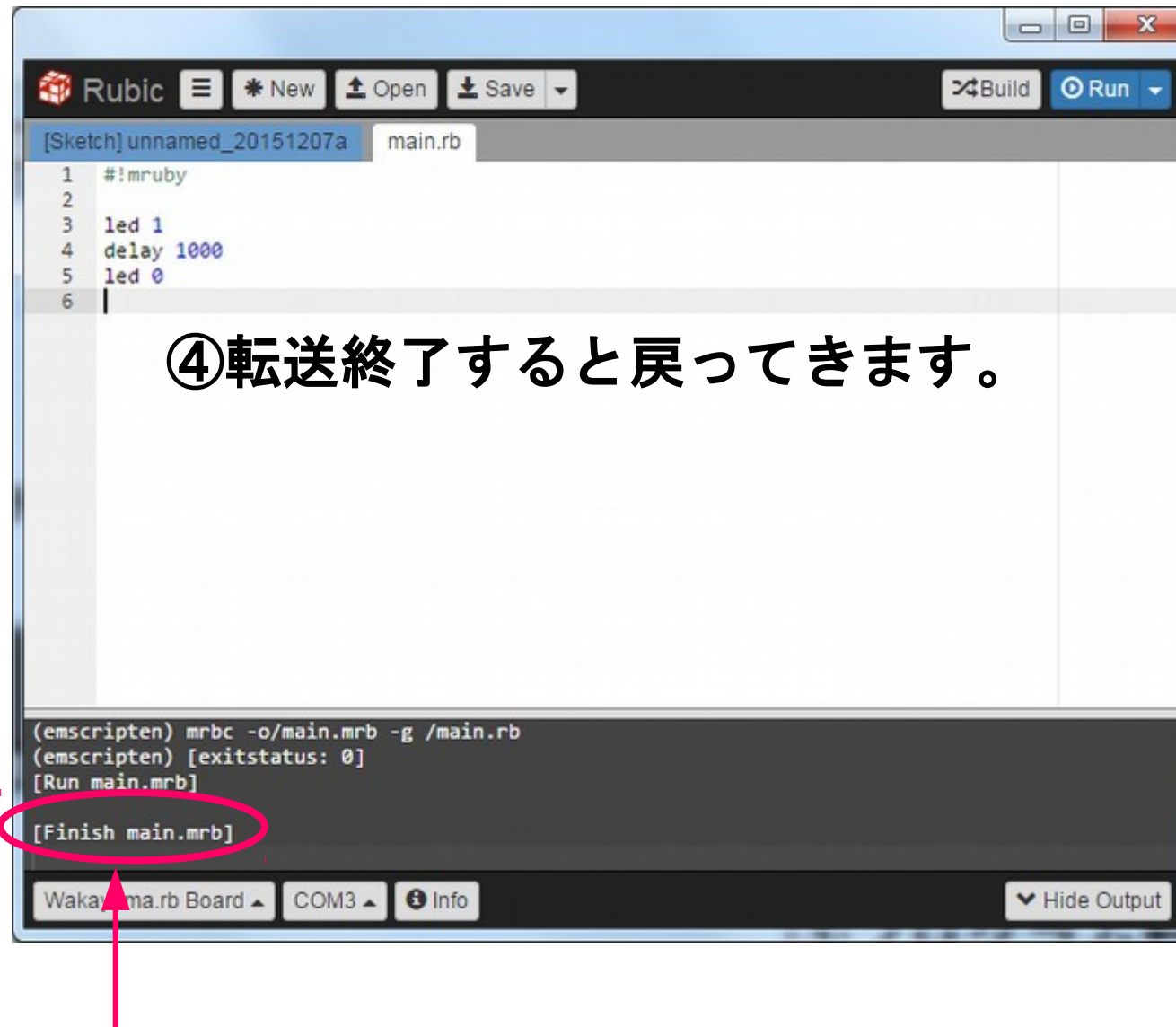


Rubic: プログラム作成と実行



③プログラム転送中です。

Rubic: プログラム作成と実行



⑤プログラムが終了すると「Finish main.mrb」と出ます。

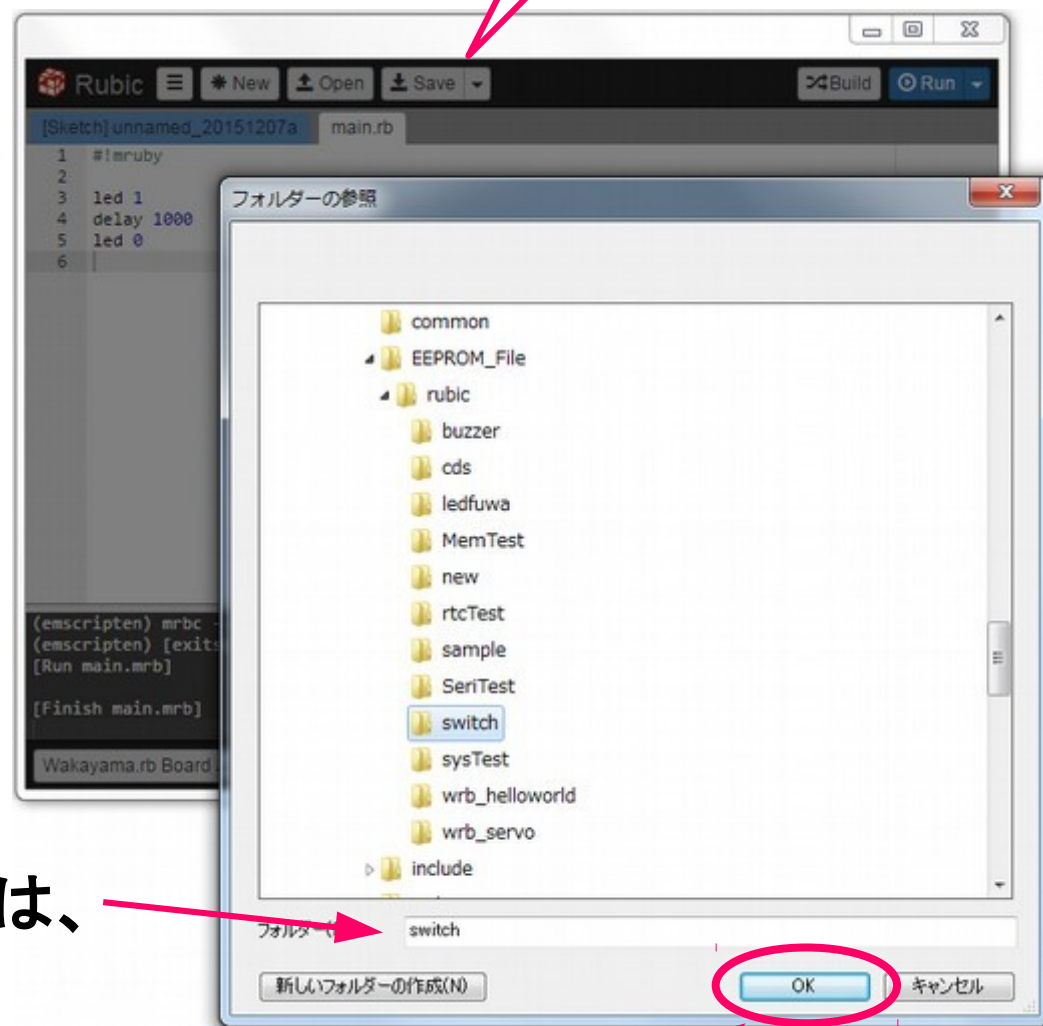
Rubic: プログラムの保存

①Save▼をクリックしてSave as...を選択します。



実行前に必ず保存

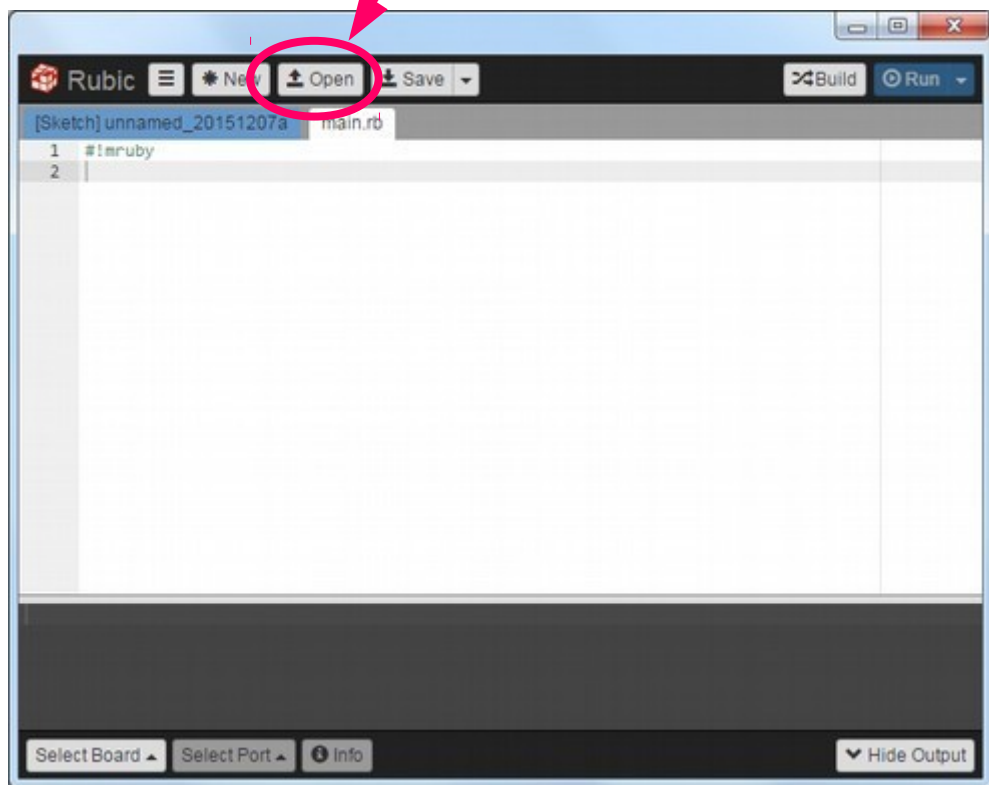
②保存するフォルダを選択、または、新規作成します。



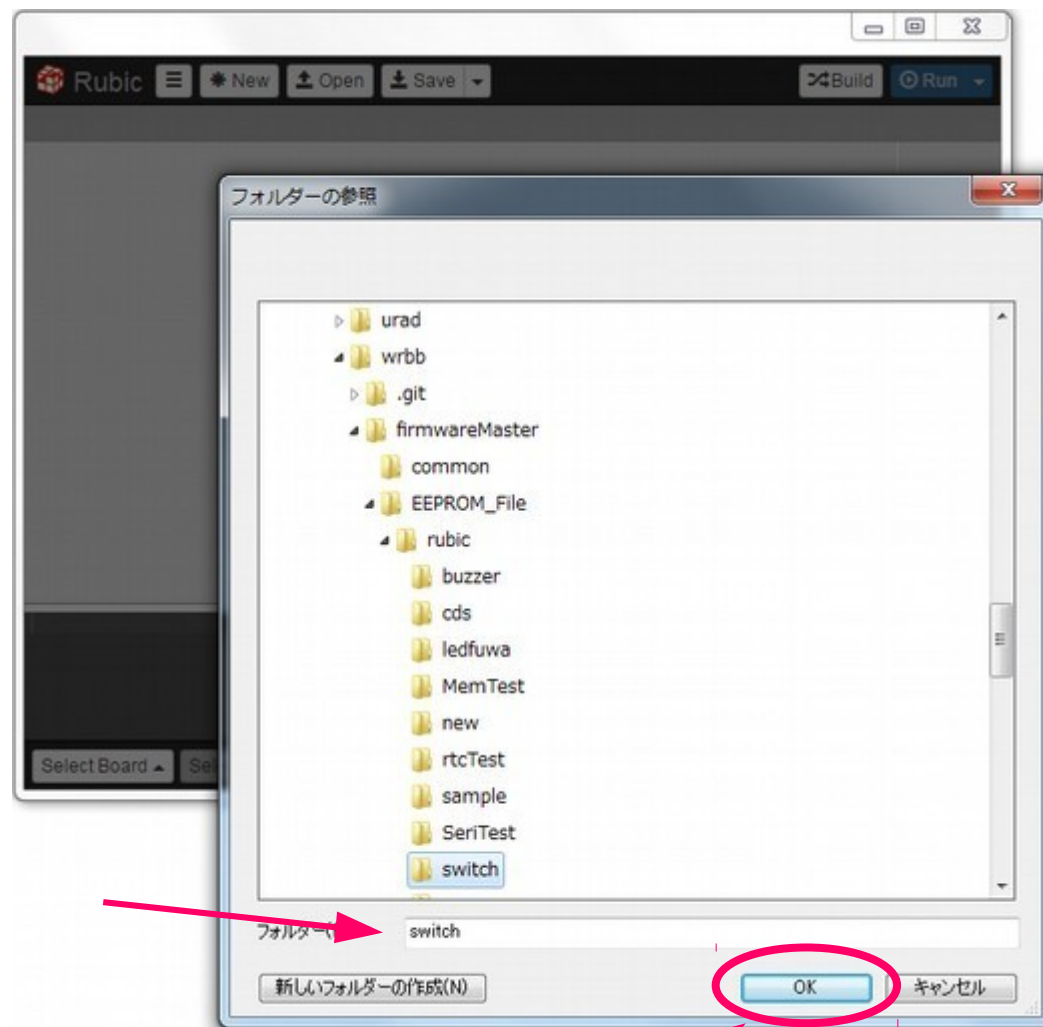
③OKをクリックします。

Rubic: プログラムの読み込み

①Openをクリックしてフォルダー一覧を出します。

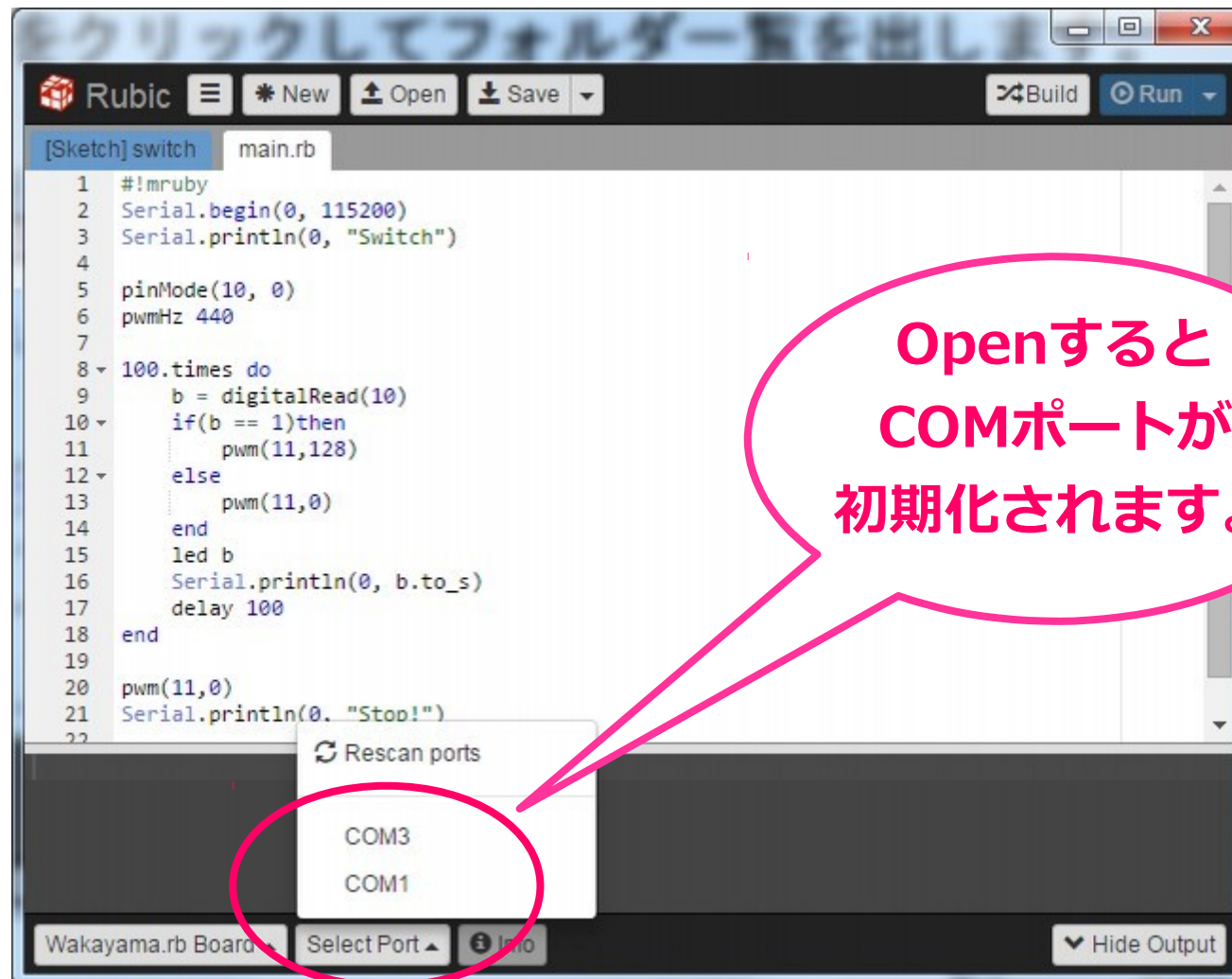


②読み込むフォルダ名を選択。
プログラムはフォルダ単位で、
保存されています。



③OKをクリックします。

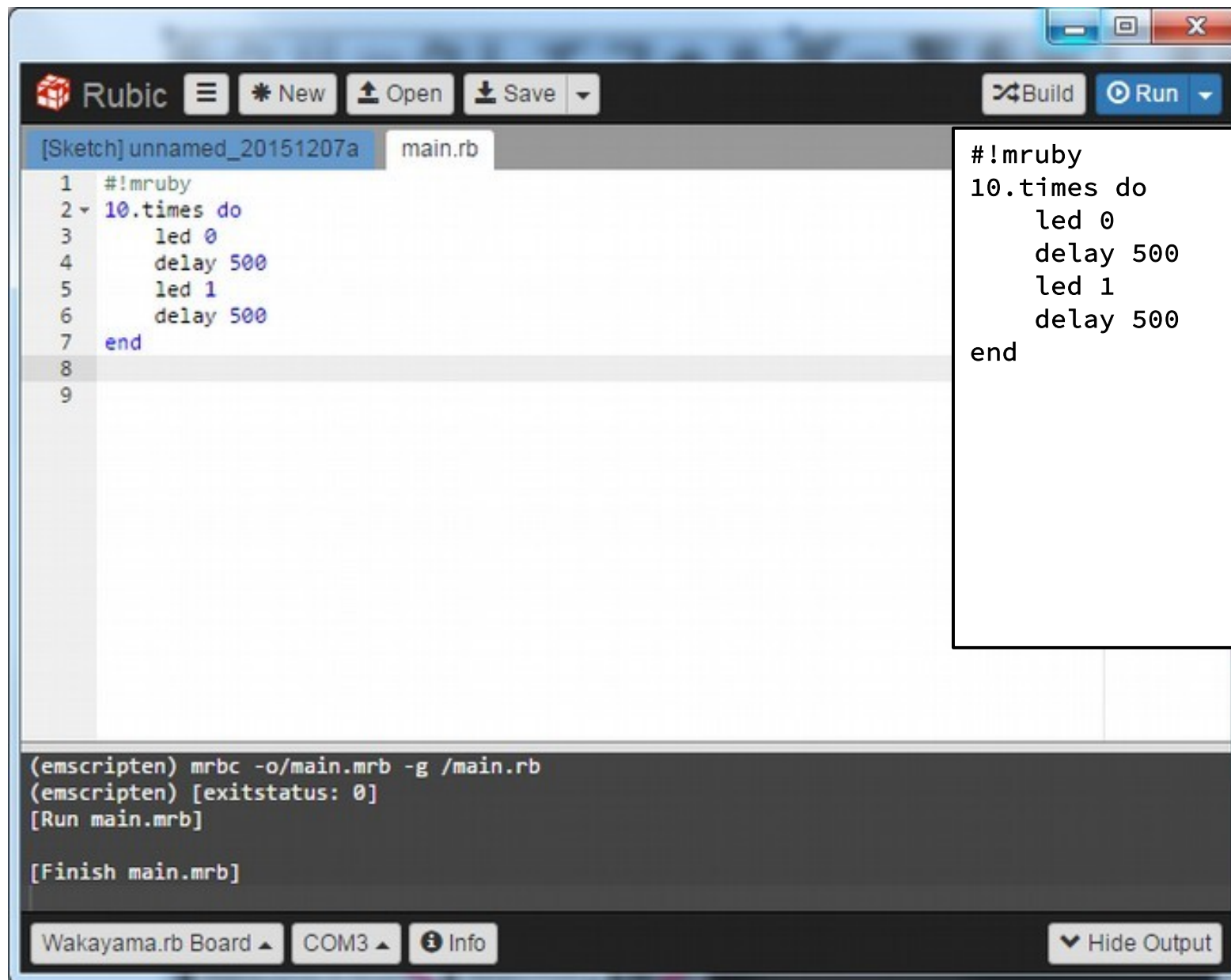
Rubic: プログラムの読み込み



④COMポートを選択します。

ハンズオン

1. LEDチカチカ



The screenshot displays the Rubic IDE interface. The main editor window shows a file named `main.rb` with the following Ruby code:

```
1 #!mruby
2 10.times do
3   led 0
4   delay 500
5   led 1
6   delay 500
7 end
```

The console at the bottom shows the execution process:

```
(emscripten) mrbc -o/main.mrb -g /main.rb
(emscripten) [exitstatus: 0]
[Run main.mrb]

[Finish main.mrb]
```

At the bottom of the IDE, there are buttons for "Wakayama.rb Board", "COM3", "Info", and "Hide Output".

2. Hello World! と LEDチカチカ

The screenshot shows the Rubic IDE interface. The main editor displays a Ruby sketch named 'helloworld' in 'main.rb'. The code is as follows:

```
1  #!mruby
2  Serial.begin(0, 115200)
3  k = 1
4  10.times do |n|
5    led k
6    k = 1 - k
7    Serial.println(0, "#{k.to_s}:Hello World!")
8    delay 500
9  end
10 led 0
11
```

A callout box on the right shows the same code without syntax highlighting:

```
#!/mruby
Serial.begin(0, 115200)
k = 1
10.times do |n|
  led k
  k = 1 - k
  Serial.println(0, "#{k.to_s}:Hello World!")
  delay 500
end
led 0
```

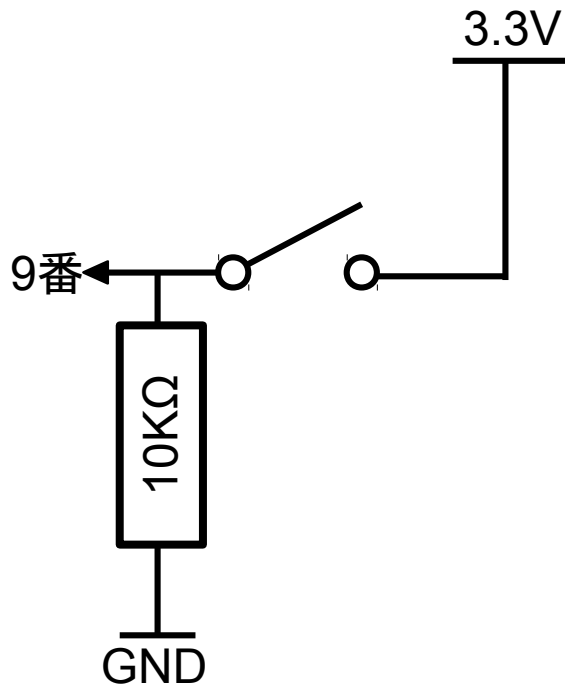
The bottom panel shows the serial output:

```
1:Hello World!
0:Hello World!
1:Hello World!
```

Below the output, it says '[Finish main.mrb]'. At the bottom, there are buttons for 'Wakayama.rb Board', 'COM3', 'Info', and 'Hide Output'.

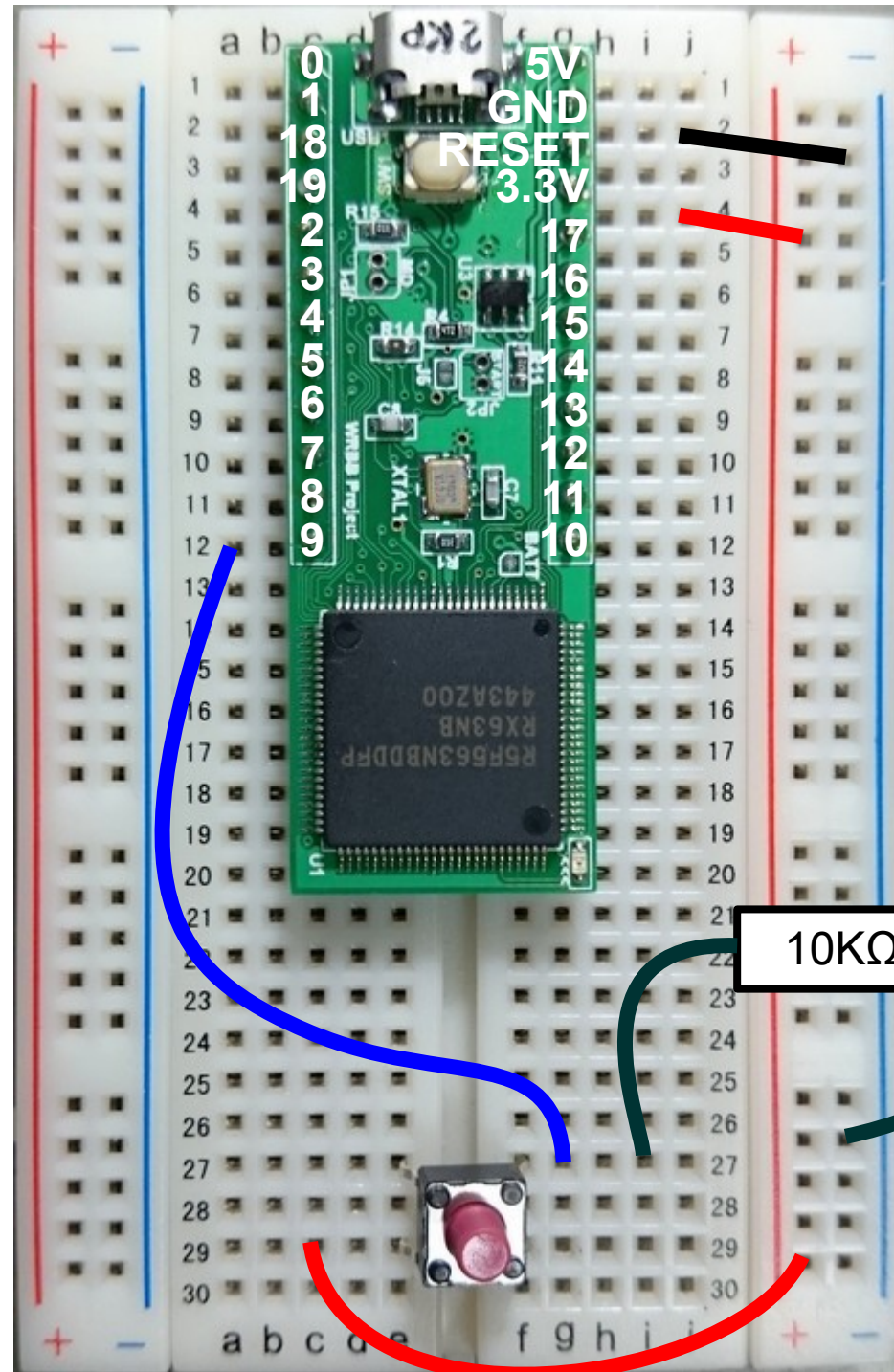
Serial.println が表示されます。

3. スイッチ



9番をg27に

スイッチを、e27,e29
f27,29に



GNDをマイナスに

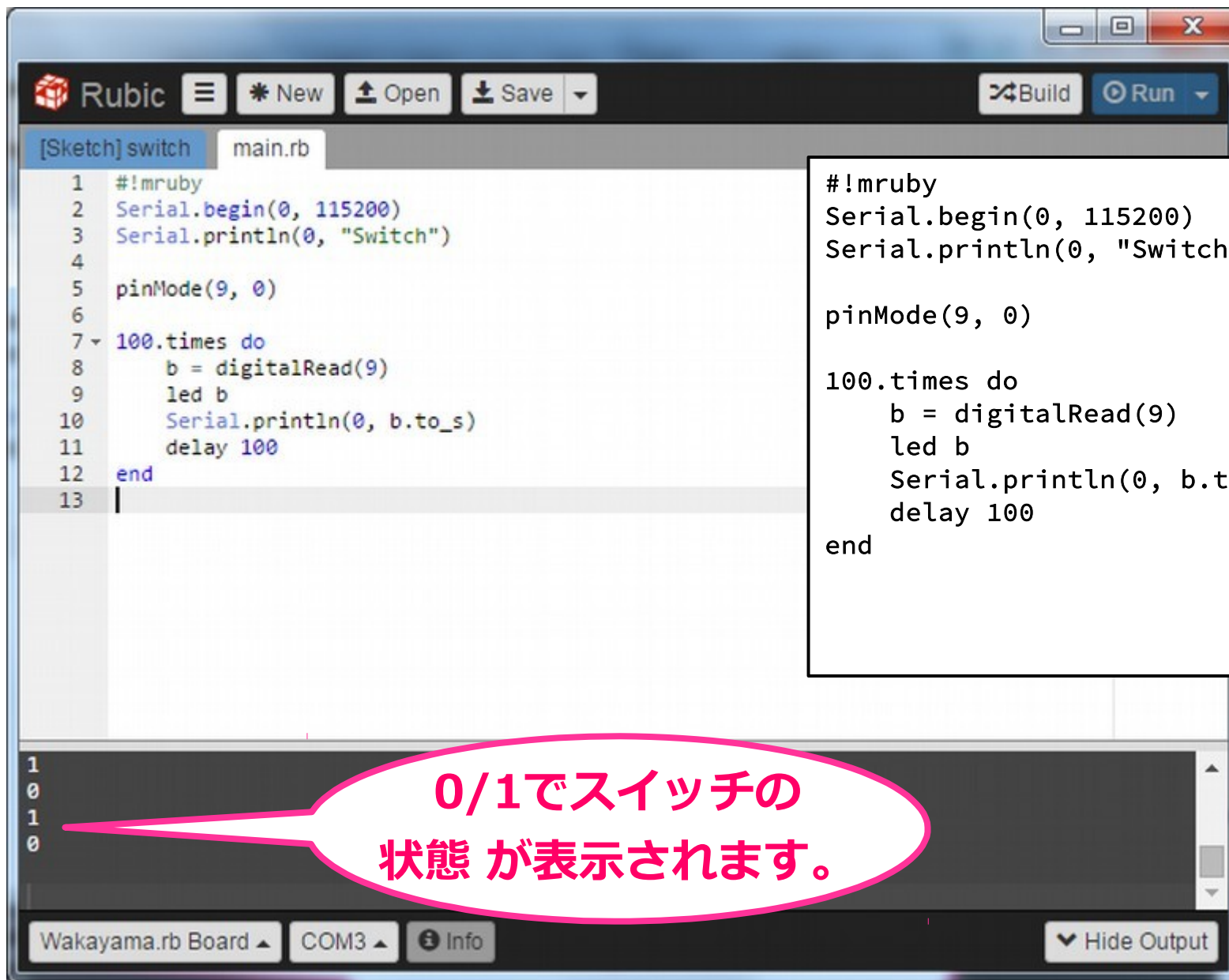
3.3Vをプラスに

10kΩをi27とGNDに

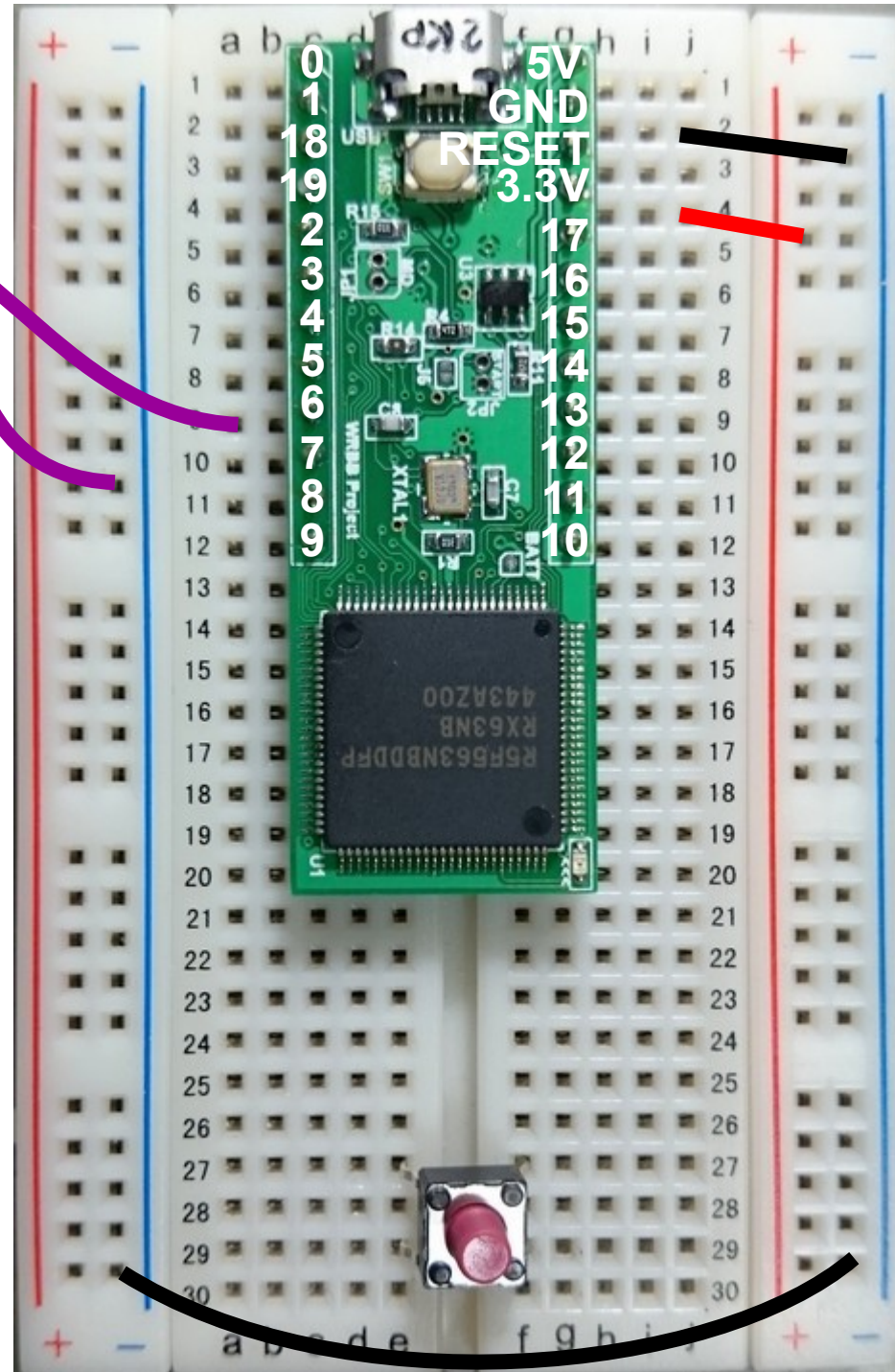
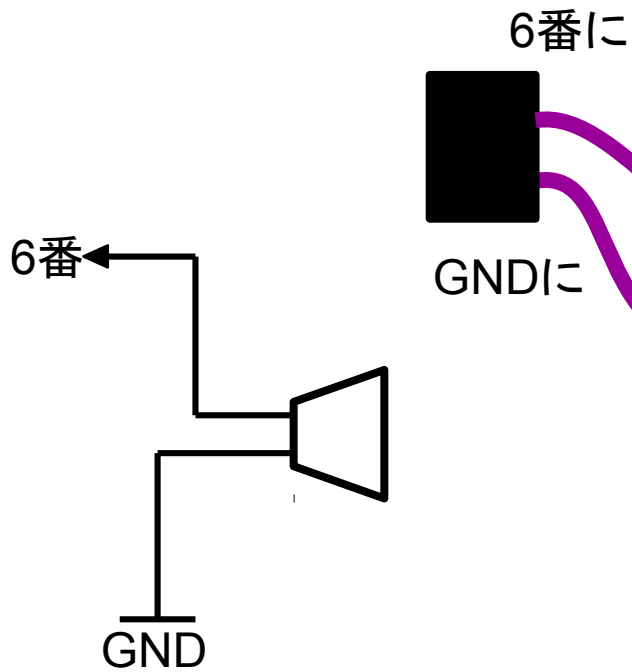
10KΩ

c29をプラスに

3. スイッチ



4. ブザー

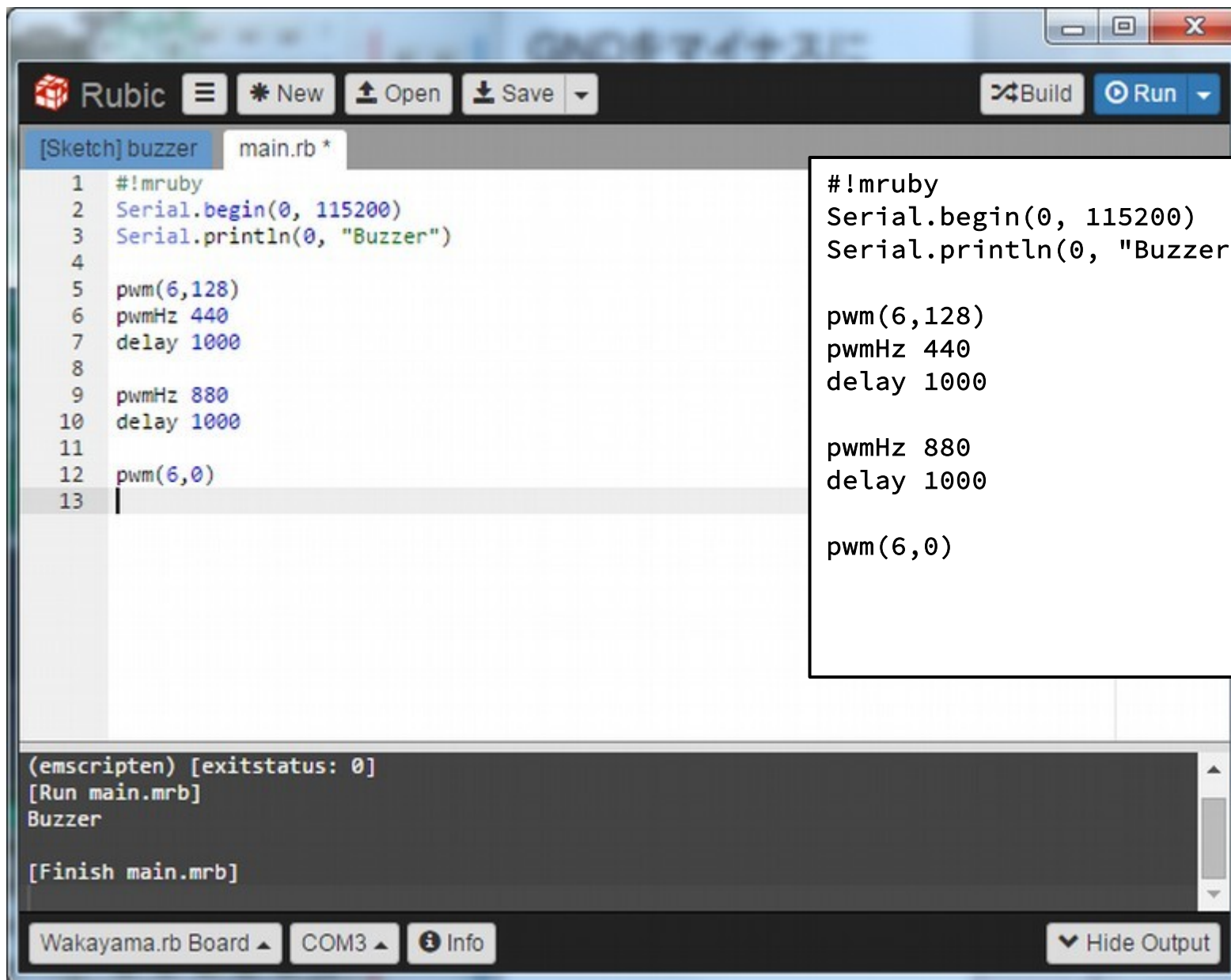


GNDをマイナスに

3.3Vをプラスに

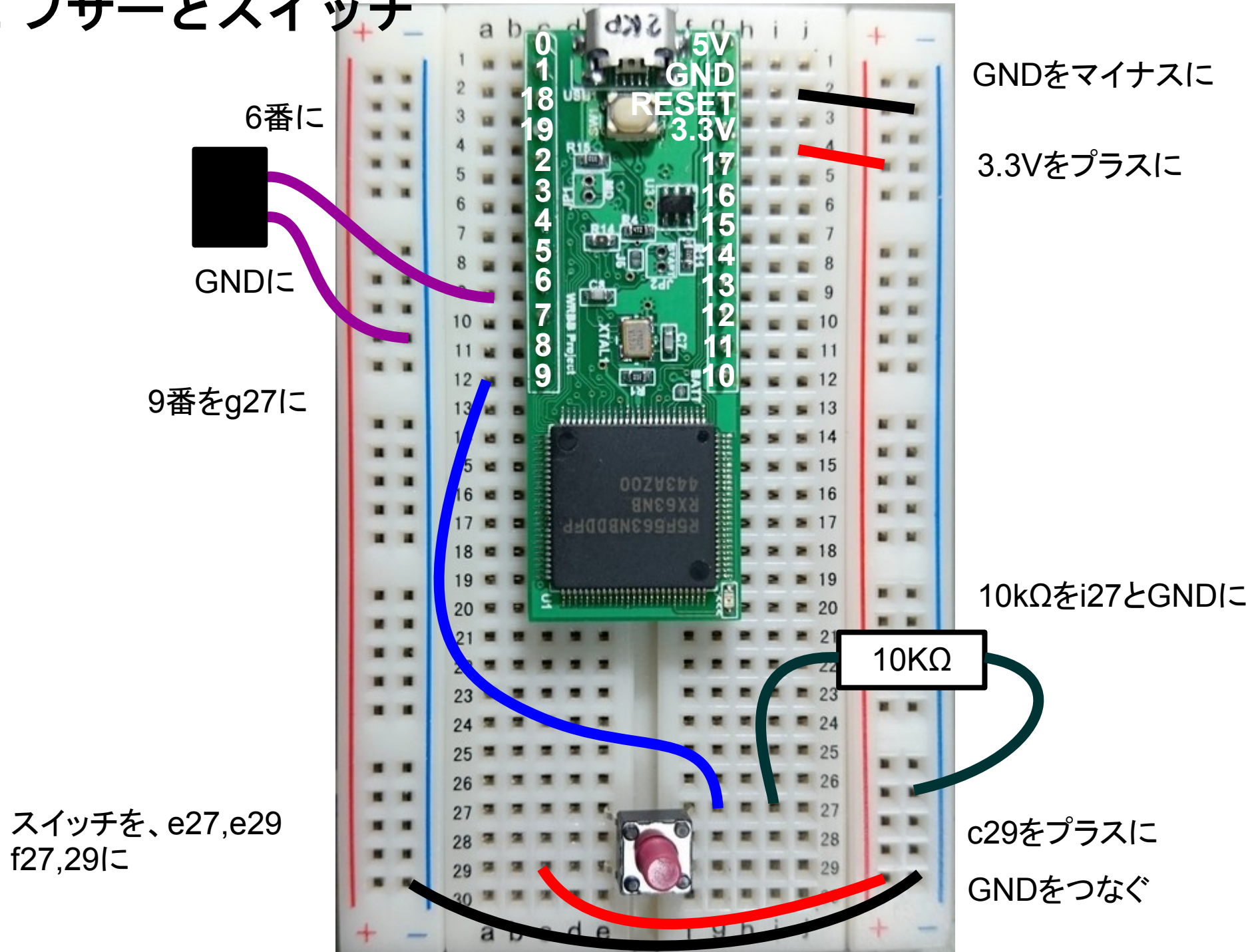
GNDをつなぐ

4. ブザー

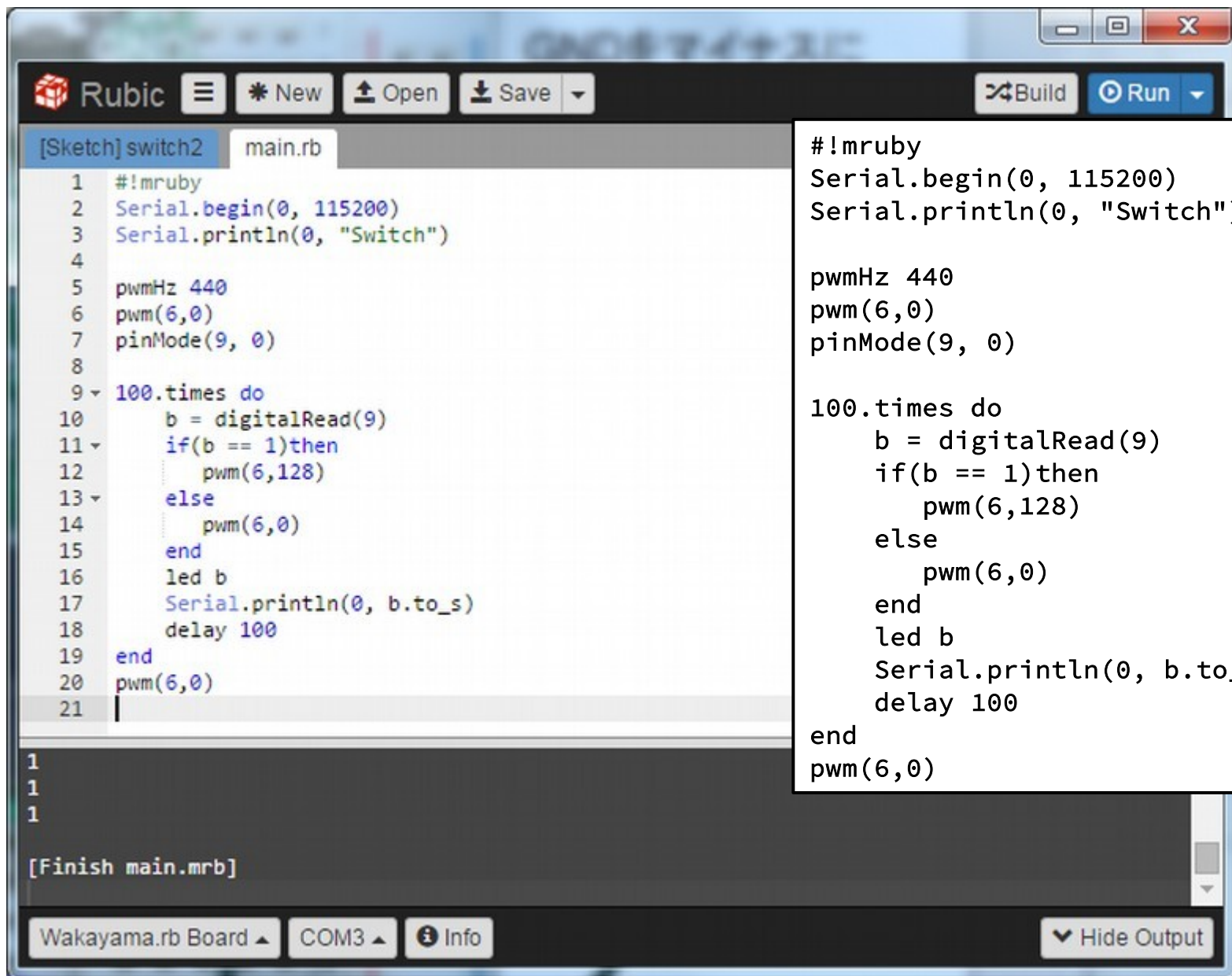


```
[Sketch] buzzer  main.rb *  
1  #!mruby  
2  Serial.begin(0, 115200)  
3  Serial.println(0, "Buzzer")  
4  
5  pwm(6,128)  
6  pwmHz 440  
7  delay 1000  
8  
9  pwmHz 880  
10 delay 1000  
11  
12 pwm(6,0)  
13  
  
(emscripten) [exitstatus: 0]  
[Run main.mrb]  
Buzzer  
  
[Finish main.mrb]  
  
Wakayama.rb Board  COM3  Info  Hide Output
```

5. ブザーとスイッチ



5. ブザーとスイッチ



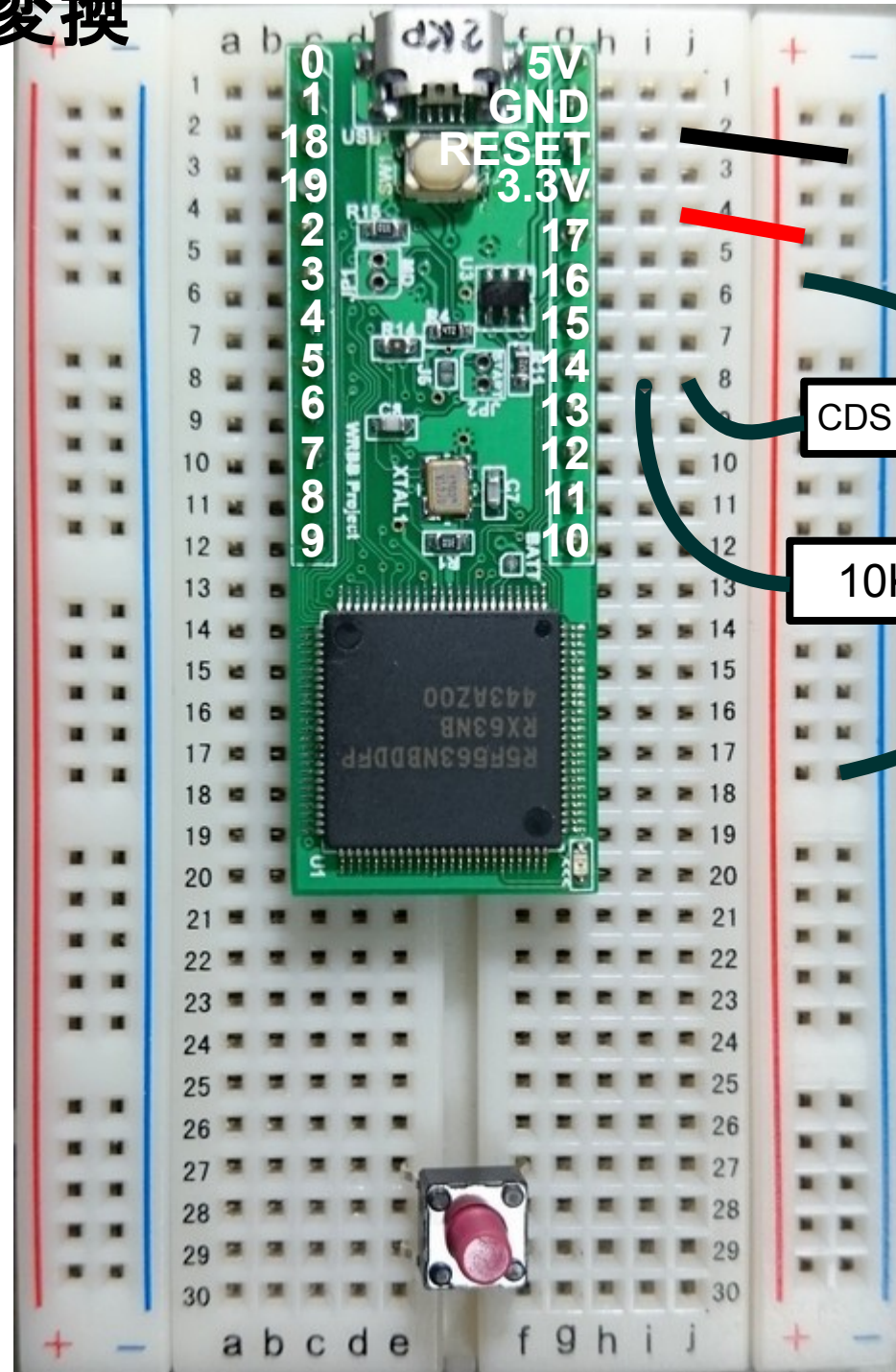
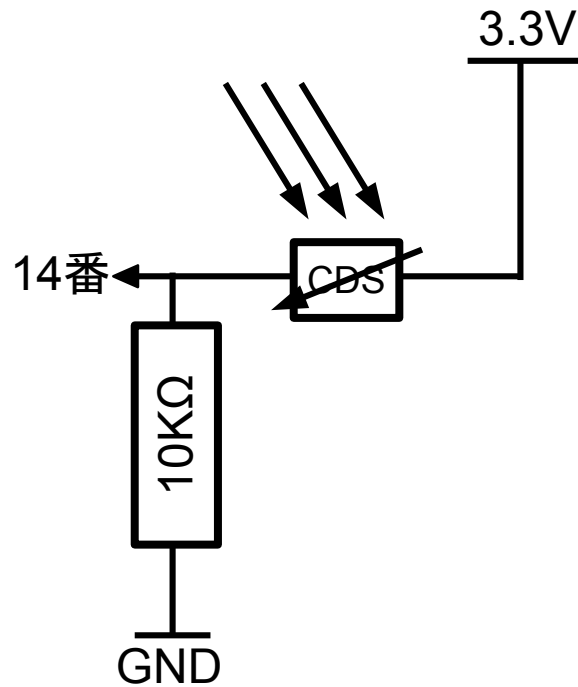
```
[Sketch] switch2  main.rb
1  #!mruby
2  Serial.begin(0, 115200)
3  Serial.println(0, "Switch")
4
5  pwmHz 440
6  pwm(6,0)
7  pinMode(9, 0)
8
9  100.times do
10    b = digitalRead(9)
11    if(b == 1)then
12      pwm(6,128)
13    else
14      pwm(6,0)
15    end
16    led b
17    Serial.println(0, b.to_s)
18    delay 100
19  end
20  pwm(6,0)
21

1
1
1

[Finish main.mrb]

Wakayama.rb Board  COM3  Info  Hide Output
```


6. 光センサとAD変換



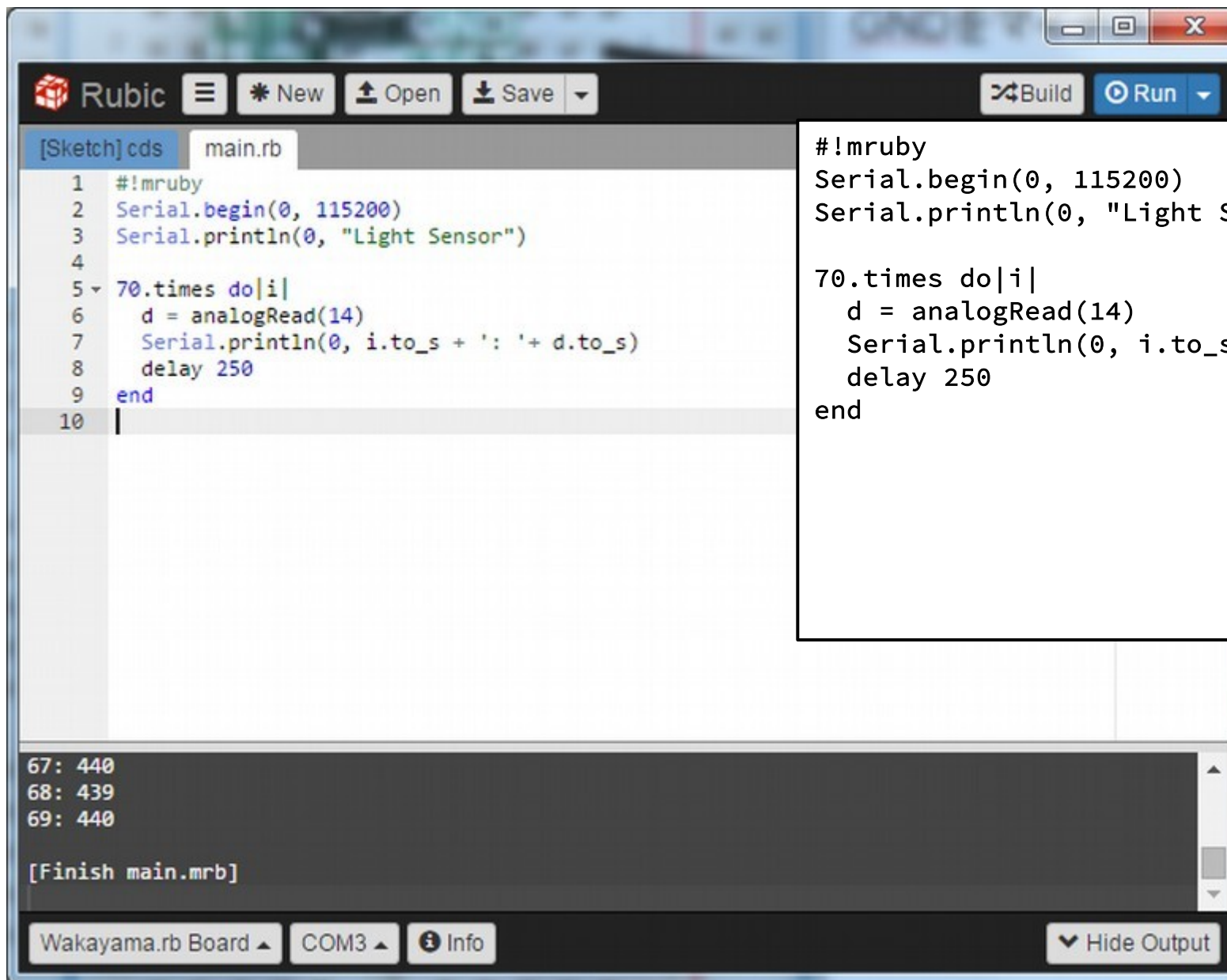
GNDをマイナスに

3.3Vをプラスに

CDSを3.3Vと14番に

10kΩを14番とGNDに

6. 光センサとAD変換

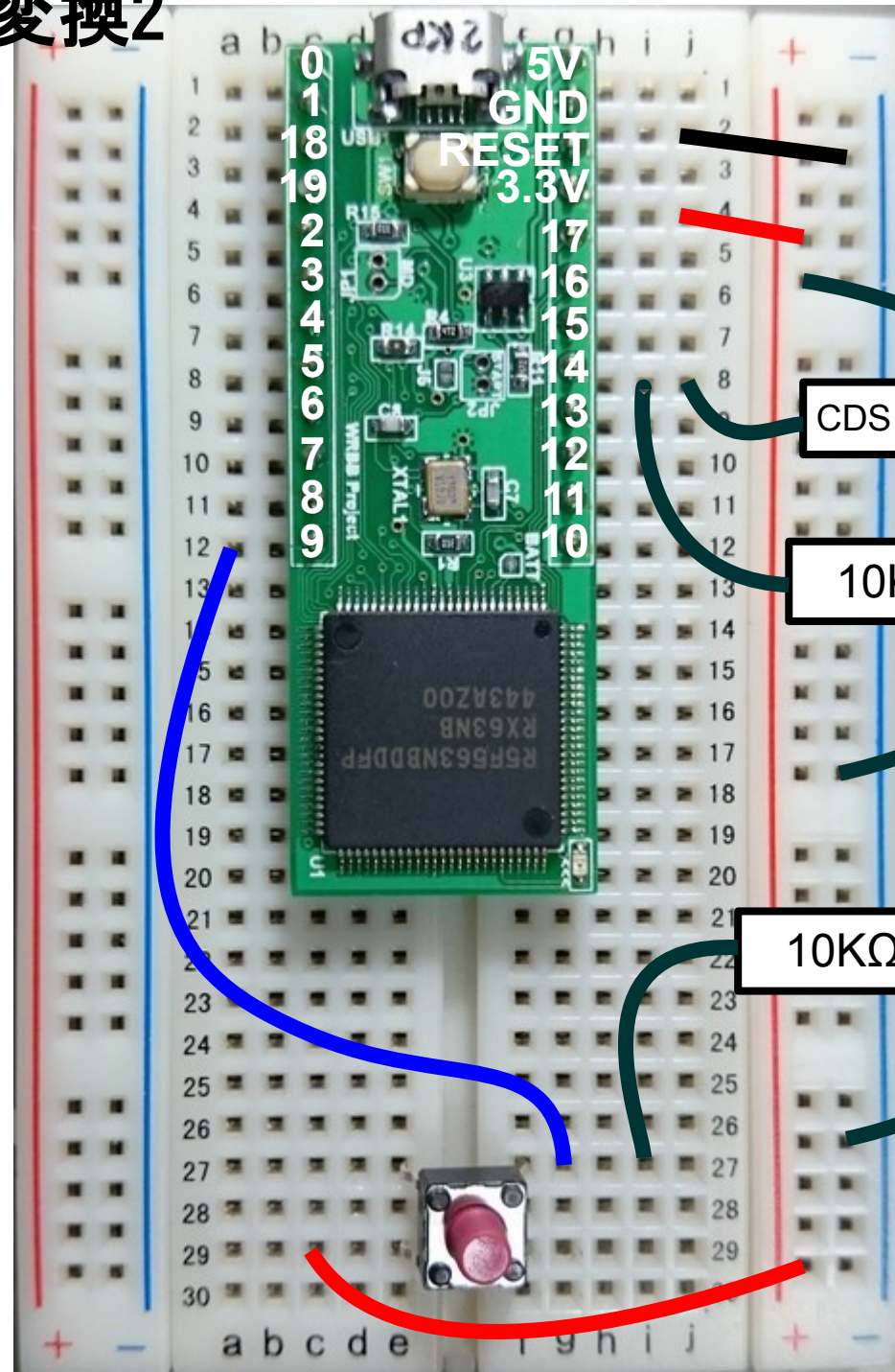


```
[Sketch] cds main.rb
1  #!mruby
2  Serial.begin(0, 115200)
3  Serial.println(0, "Light Sensor")
4
5  70.times do|i|
6    d = analogRead(14)
7    Serial.println(0, i.to_s + ': ' + d.to_s)
8    delay 250
9  end
10 |
```

```
67: 440
68: 439
69: 440

[Finish main.mrb]
```

Wakayama.rb Board COM3 Info Hide Output



3.3Vをプラスに

CDS

10K Ω

10k Ω を14番とGNDに

10KΩ

10kΩをi27とGNDに

c29をプラスに

7. 光センサとAD変換2

The screenshot shows the Rubic IDE interface. The main editor displays a Ruby script in a file named `main.rb`. The script is as follows:

```
1 #!mruby
2 Serial.begin(0, 115200)
3 Serial.println(0, "Light Sensor2")
4
5 pinMode(9, 0)
6
7 while true
8   if(digitalRead(9) == 1)then
9     break
10  end
11
12  d = analogRead(14)
13  Serial.println(0, d.to_s)
14  delay 100
15 end
16
```

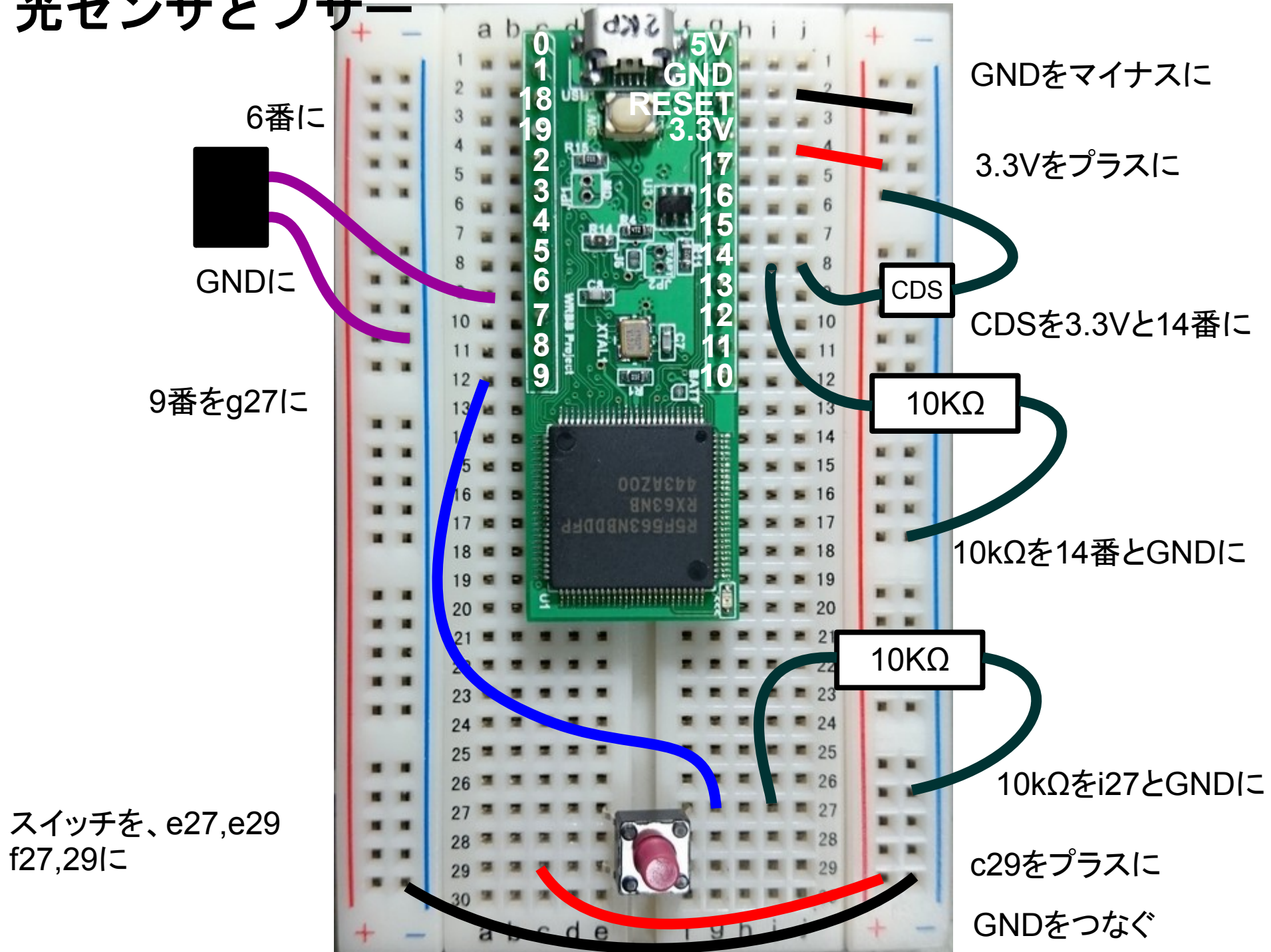
A pink callout bubble points to the `while true` loop, containing the text: **ボタンを押すと whileを抜けます** (When you press the button, you exit the while loop).

Below the editor, the output window shows the following text:


```
417
419
425
[Finish main.mrb]
```

The bottom status bar indicates the board is `Wakayama.rb Board`, the port is `COM3`, and there is an `Info` button. A `Hide Output` button is also present.

8. 光センサとブザー



8. 光センサとブザー



The screenshot shows the Rubic IDE interface. The main editor displays a Ruby script for a light sensor and buzzer. A pink callout bubble points to the line calculating the frequency based on the sensor value.

```
[Sketch] cds_buzzer main.rb
1  #!mruby
2  Serial.begin(0, 115200)
3  Serial.println(0, "Light Sensor and Buzzer")
4  lmax = 400
5  lmin = 170
6  hz = 0
7  pwm(6,128)
8  pwmHz hz
9  pinMode(9, 0)
10 while true
11   if(digitalRead(9) == 1)then
12     break
13   end
14   d = analogRead(14)
15   Serial.println(0, d.to_s)
16
17   hz = (d - lmin)/(lmax-lmin)*(6000-20) + 20
18   pwmHz hz
19   delay 120
20 end
21 pwm(6,0)
22
```

センサ値で周波数が変わります

Wakayama.rb Board ▾ Hide Output

```
#!/mruby
Serial.begin(0, 115200)
Serial.println(0, "Light Sensor and Buzzer")
lmax = 400
lmin = 170
hz = 0
pwm(6,128)
pwmHz hz
pinMode(9, 0)
while true
  if(digitalRead(9) == 1)then
    break
  end
  d = analogRead(14)
  Serial.println(0, d.to_s)

  hz = (d - lmin)/(lmax-lmin)*(6000-20) + 20
  pwmHz hz
  delay 120
end
pwm(6,0)
```

ソースプログラムの入手方法

Wakayama.rbボードの基本プログラムは、githubでオープンソースとして、すべて公開されています。
makeしたプログラムはwrbb.motです。

<https://github.com/wakayamarb/wrbb-v1lib-firm>

今回使用した基板は、WRBB4ファームが動作しているものです。Ver.4.0基板です。

ハンズオンに用いたプログラムは、下記URLにあります。

<https://github.com/wakayamarb/wrbb-v1lib-firm/tree/master/sample/rubic>

Wakayama.rbボードへのwrbb.motファイルの書き込み方法は、下記URLのブログに公開しています。

<http://d.hatena.ne.jp/tarosay/20151123/1448301501>

以上
ありがとうございました。
お疲れさまでした。

毎月勉強会を開いています。是非、Wakayama.rbに参加してください。
<https://wakayamarb.doorkeeper.jp/>