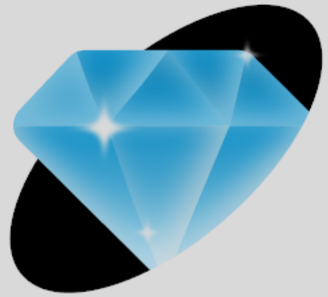
A Beginner's Complete Guide to Microcontroller Programming with Ruby

hasumikin

Euruko 2023

Vilnius, Lithuania

21st - 23rd September 2023



Part 1

Preparation

Part 2

Getting Started with Microcontroller

Part 3

Exploring PicoRuby Further

Part 4

PicoRuby Under the Hood

EURUKO

self.inspect

- Hitoshi HASUMI
- hasumikin (GitHub ,ex-Twitter, Bluesky and Threads)
- Creator of PicoRuby and PRK Firmware
- Committer of CRuby's IRB and Reline
- First prize of Fukuoka Ruby Award (2020 and 2022 ✌)
- A final nominee of Ruby Prize 2021

EURUKO

Part 1

Preparation

EURUKO

Setup (minimal)

- Raspberry Pi Pico
  - Or other RP2040-based controller
- USB cable
- Terminal emulator on laptop

EURUKO

Raspberry Pi Pico

- Raspberry Pi Pico: Microcontroller board
  - MCU: RP2040
    - Cortex-Mzero+ (dual)
    - 264 KB RAM
    - 2 MB flash ROM
  - It generally runs without an OS (bere metal)
- ref) Raspberry Pi: Single-board computer
  - It generally needs an OS like Raspberry Pi OS or Windows for Arm

Terminal emulator

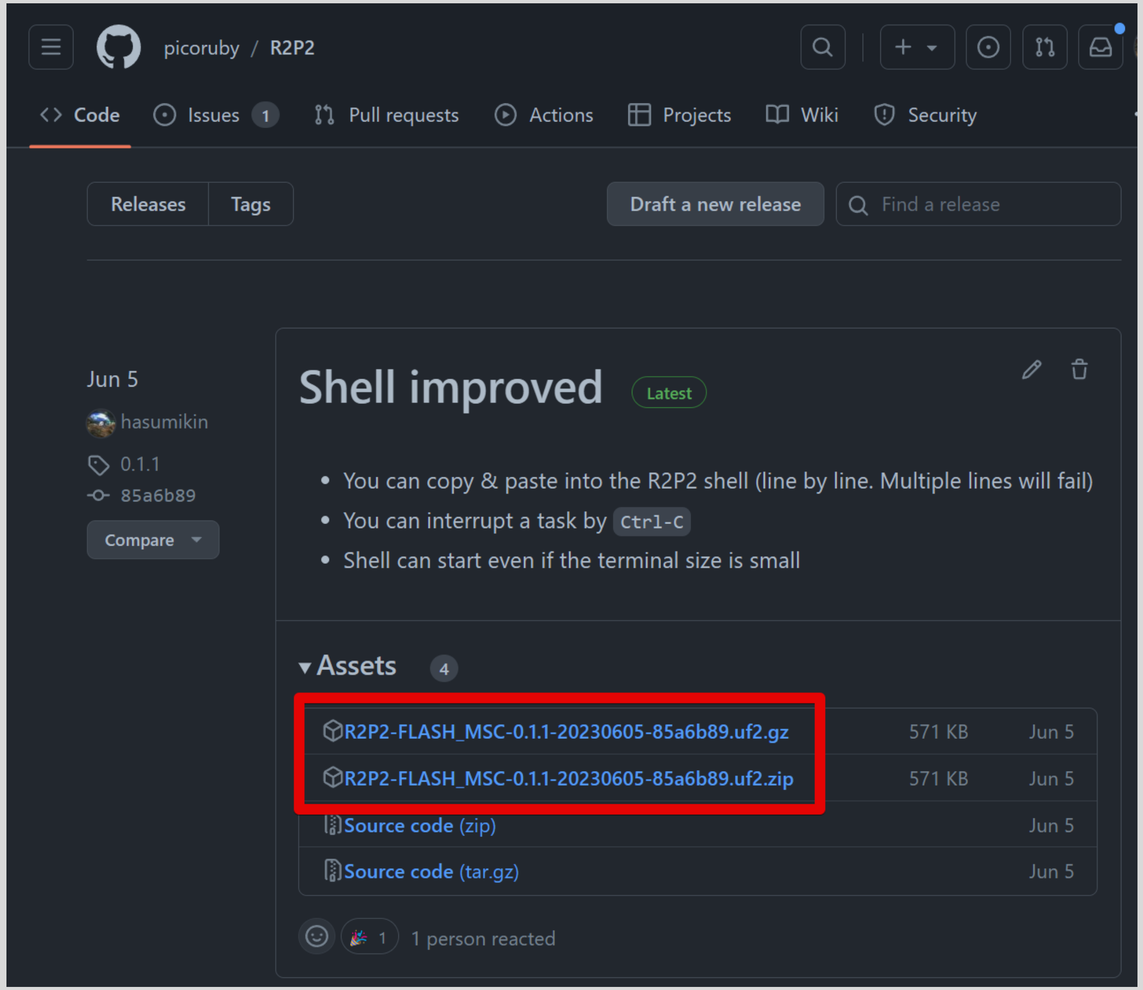
- Linux -> GTKTerm👍
- Windows -> Tera Term🤦
- macOS -> PuTTY (I'm not sure 🤞)
- Traditional CUI/TUI tools may have CR/LF trouble
  - cu
  - screen
  - minicom

EURUKO

Let's begin 1/4

- Download the latest `R2P2-*.uf2` from GitHub

https://github.com/picoruby/R2P2/releases



EURUKO

Let's begin 1/4

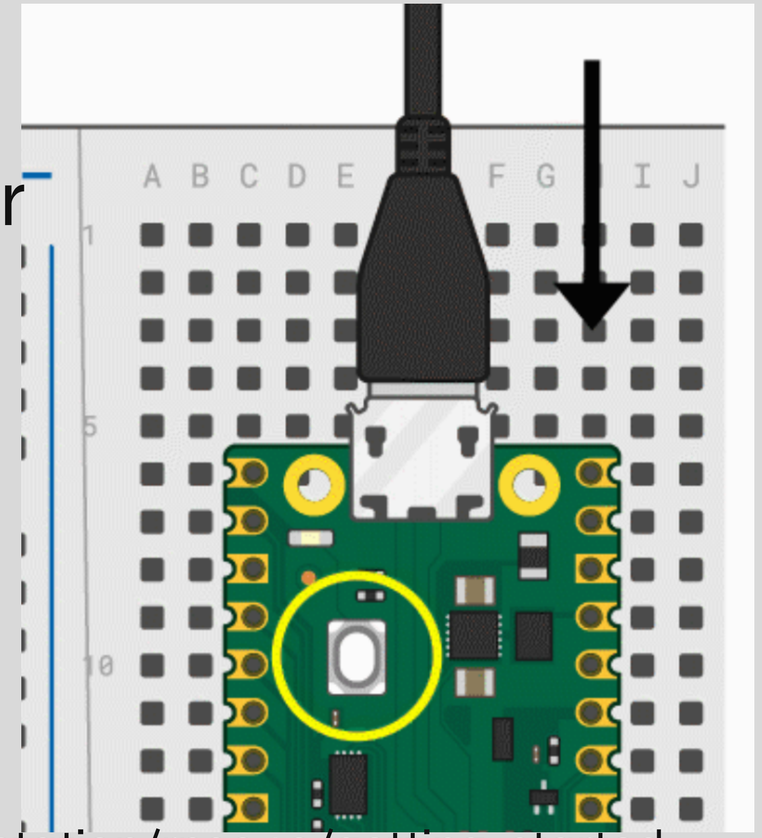
BTW, R2P2 stands for

Ruby on Raspberry Pi Pico

EURUKO

Let's begin 2/4

- Connect Pi Pico and PC while pressing the BOOTSEL button
  - You'll find "RPI-RP2" drive in file manager

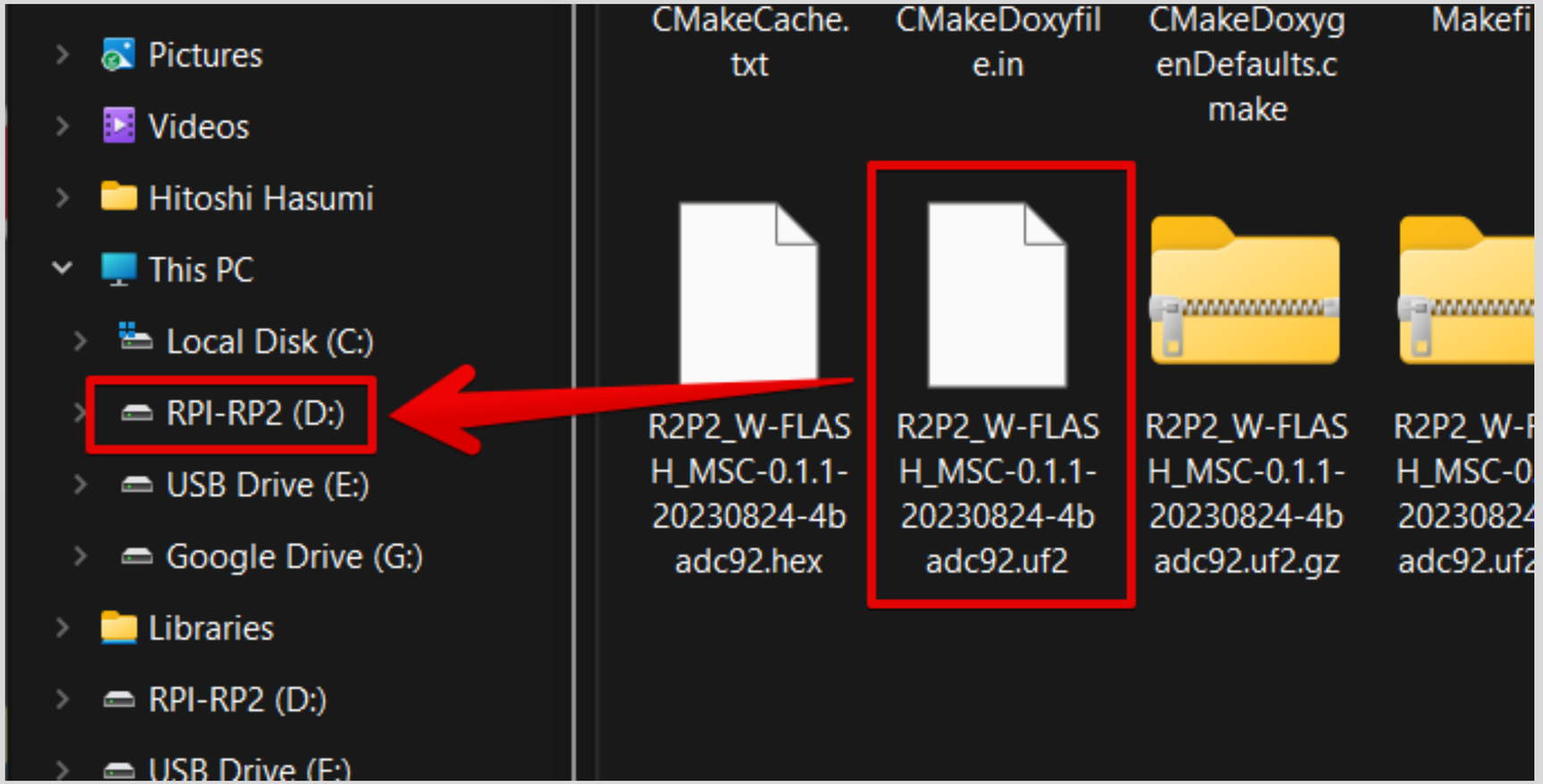


https://www.raspberrypi.org/documentation/rp2040/getting-started

EURUKO

Let's begin 3/4

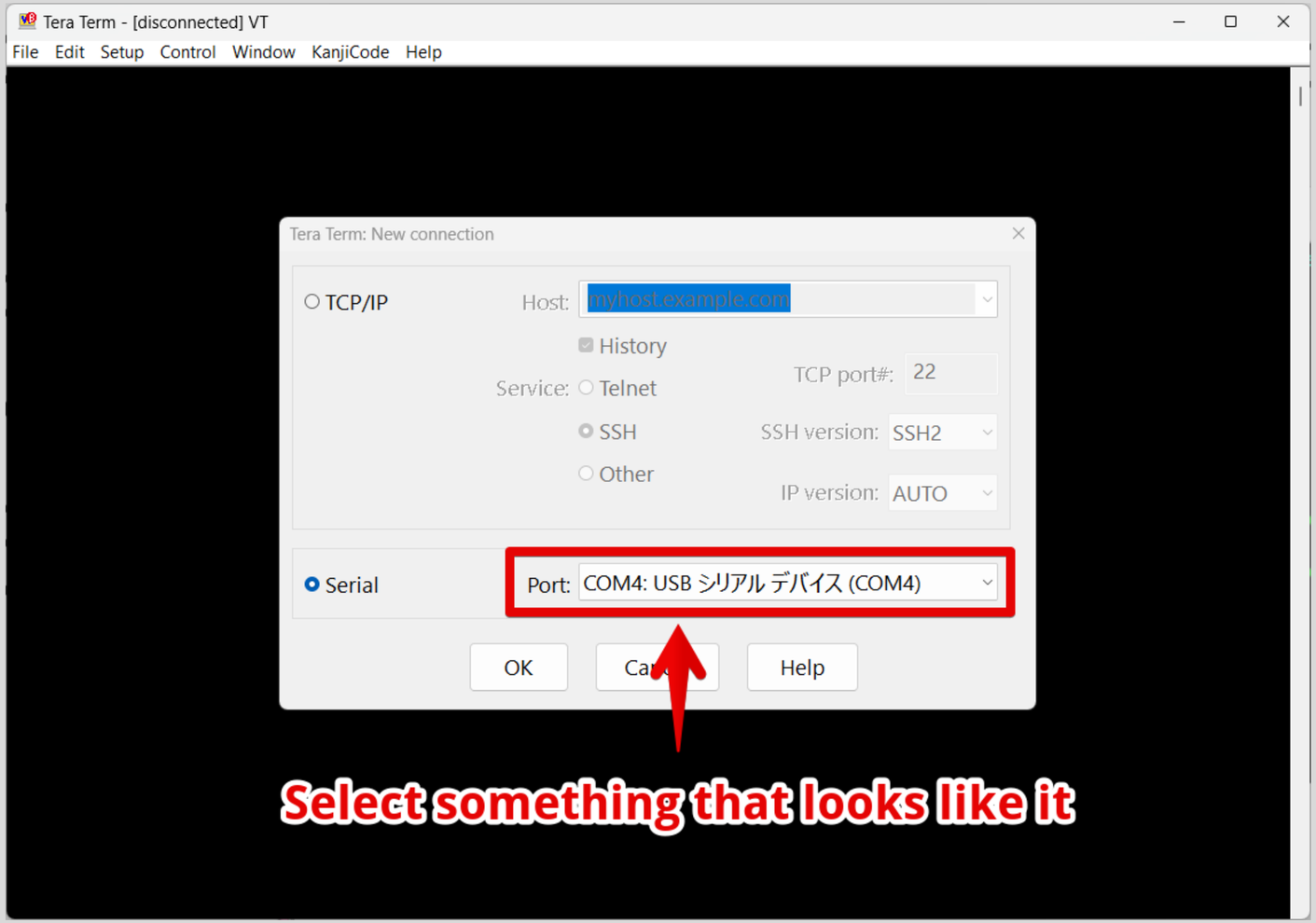
- Drag & drop `R2P2-*.uf2` into RPI-RP2 drive



EURUKO

Let's begin 4/4

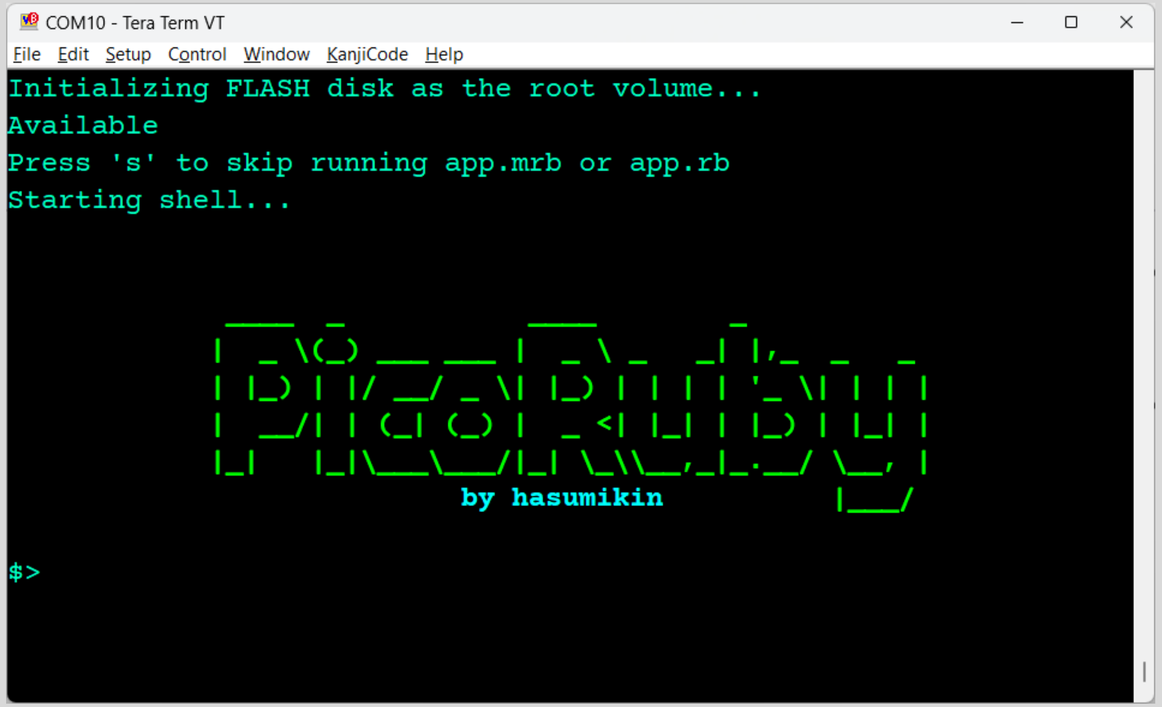
- Open a proper serial port on terminal emulator



EURUKO

R2P2 Shell should start [Demo]

- Unix-like shell running on Raspberry Pi Pico
- You can use some commands like `cd`, `ls`, `mkdir`, and `irb`



EURUKO

PicoIRB [Demo]

- PicoRuby's IRB is running within the R2P2 shell on Raspberry Pi Pico
- Your Ruby snippet is compiled into mruby VM code and executed on the fly
  - It means PicoRuby contains an mruby compiler which can run on a one-chip microcontroller (will be mentioned later)

Part 2

Getting Started with Microcontroller

EURUKO

GPIO (General Purpose Input/Output)

- Fundamental digital I/O
- Variety of uses:
  - Input: Detects on-off state of switch and button
  - Output: Makes a voltage
  - You can even implement a communication protocol by controlling GPIO in milli/micro sec

GPIO --- Blinking LED [Demo]

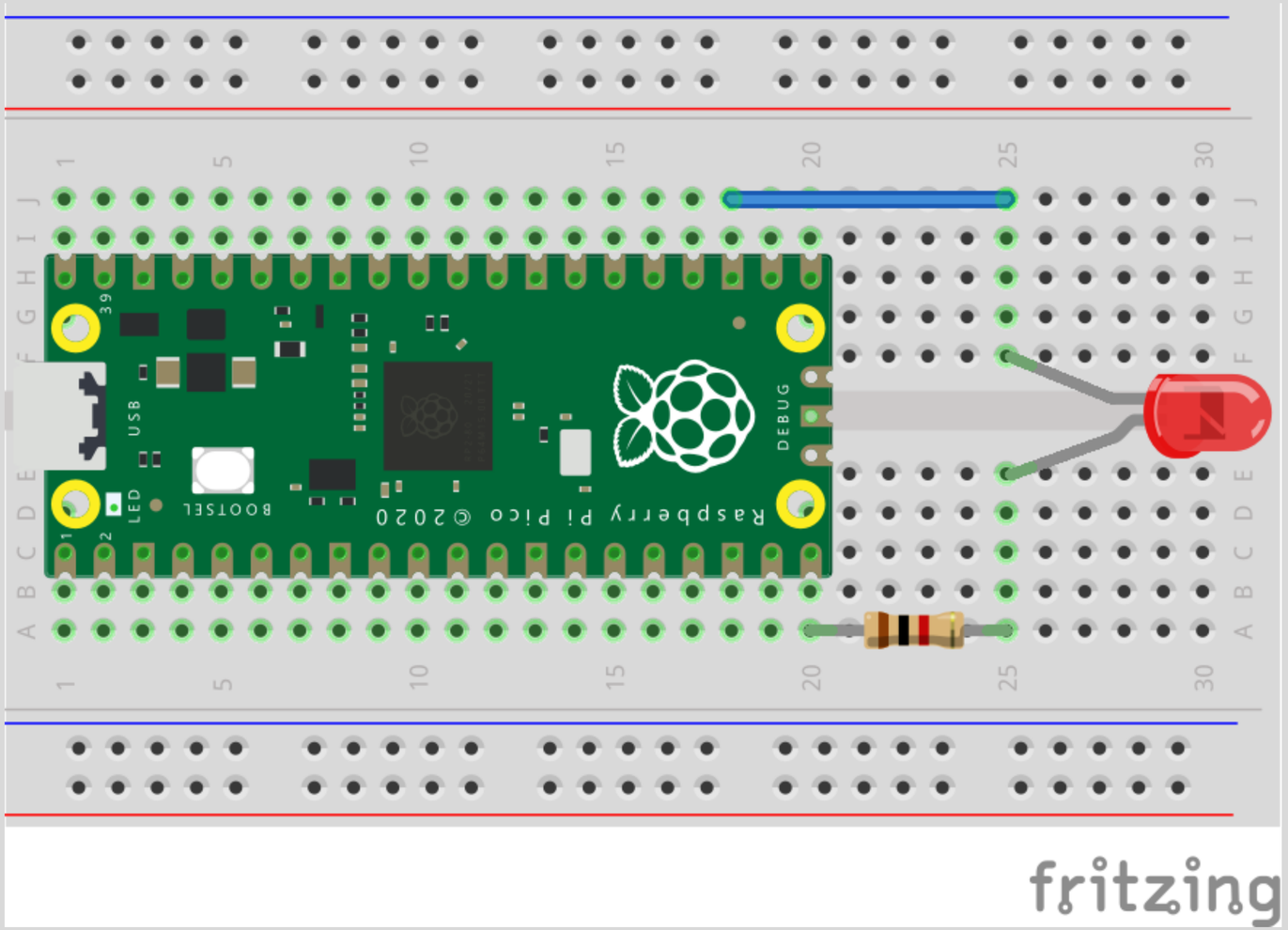
```
irb> led = GPIO.new(25, GPIO::OUT)
irb> 5.times do
irb*   led.write 1
irb*   sleep 1
irb*   led.write 0
irb*   sleep 1
irb* end
```

GPIO25 internally connects to on-board LED through a resistor

EURUKO

GPIO --- Blinking LED by discrete parts

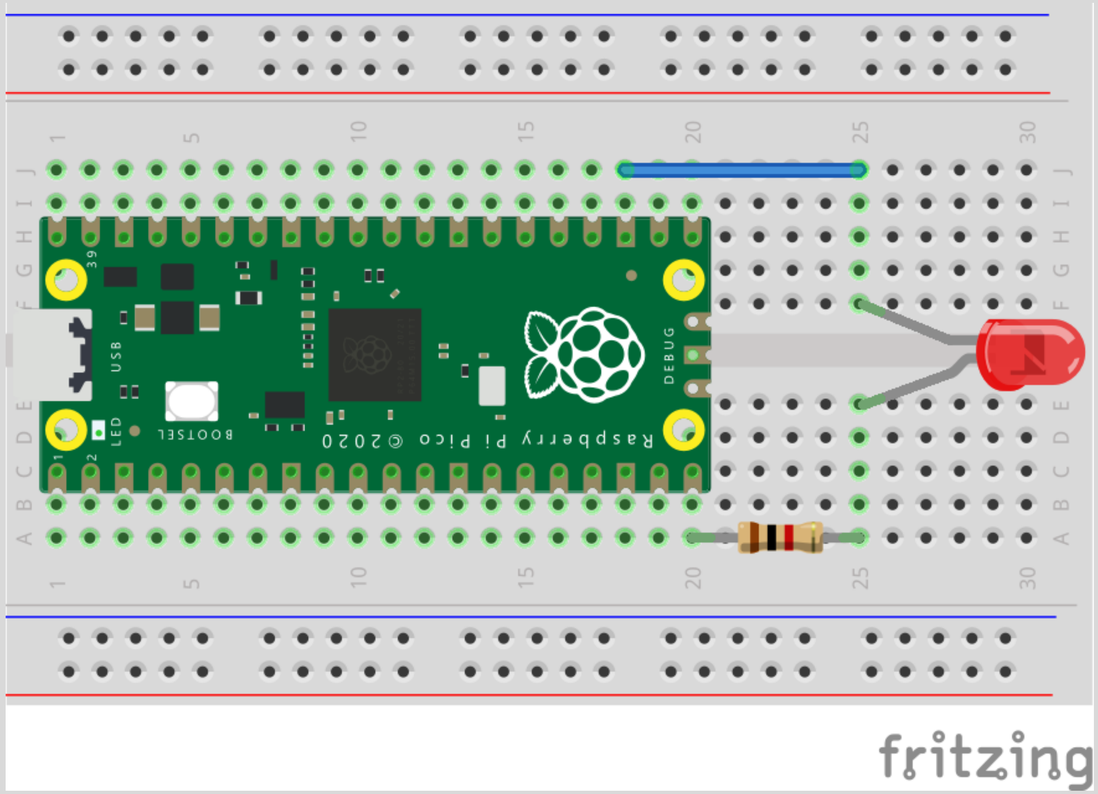
- Parts list:
  - LED (RED)
  - Resistor (1kΩ)



EURUKO

GPIO --- Blinking LED by discrete parts

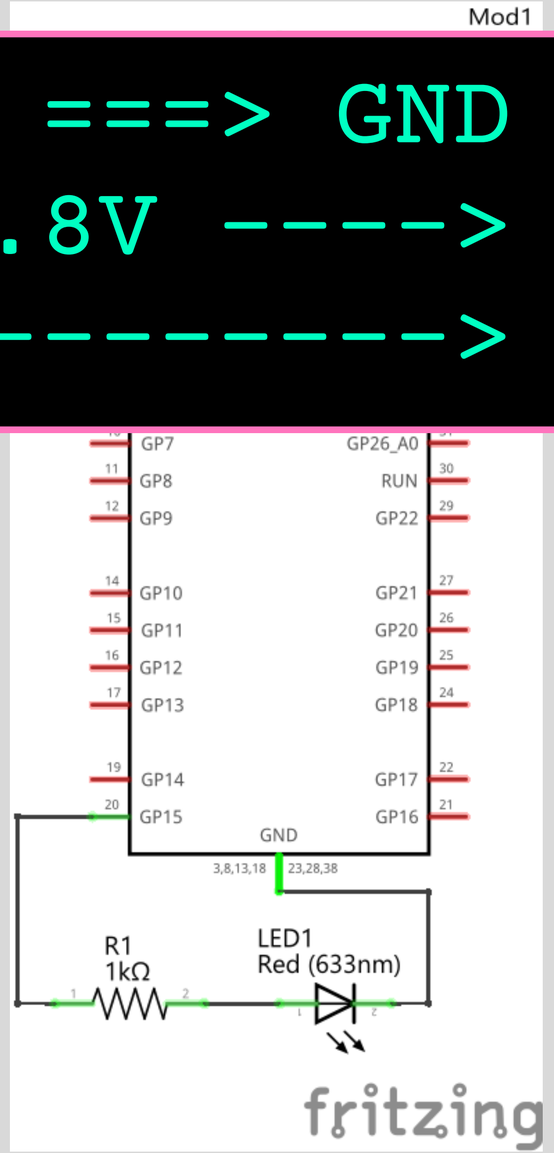
```
irb> pin = GPIO.new(15, GPIO::OUT)
```



EURUKO

GPIO --- Blinking LED by discrete parts

```
GPIO15 ===> 1kΩ ===> LED ===> GND
<----- 1.5V -----><--- 1.8V ---->
<-------------- 3.3V --------------->
```



- RP2040's logic level: 3.3V
- LED voltage drop: 1.8V
  (according to LED's datasheet)
- Current: (3.3V - 1.8V) / 1kΩ = 1.5mA
  (calculated by Ohm's Law)

EURUKO

🏫 Study time: Physics

- Ohm's Law
  - V = I * R ⇔ I = V / R ⇔ R = V / I
- Kirchhoff's Circuit Laws
  - Current law: The algebraic sum of currents in a network of conductors meeting at a point is zero
  - Voltage law: The directed sum of the potential differences (voltages) around any closed loop is zero

ADC (Analog to Digital Converter)

- ADC handles values in-between by converting an analog voltage to a digital value
- e.g. RP2040's ADC has 12 bits depth and accordingly takes a raw value from 0 (0 V) to 4095 (3.3 V)
- Typical usage:
  - Temperature sensor
  - Joystick

EURUKO

ADC --- Temperature [Demo]

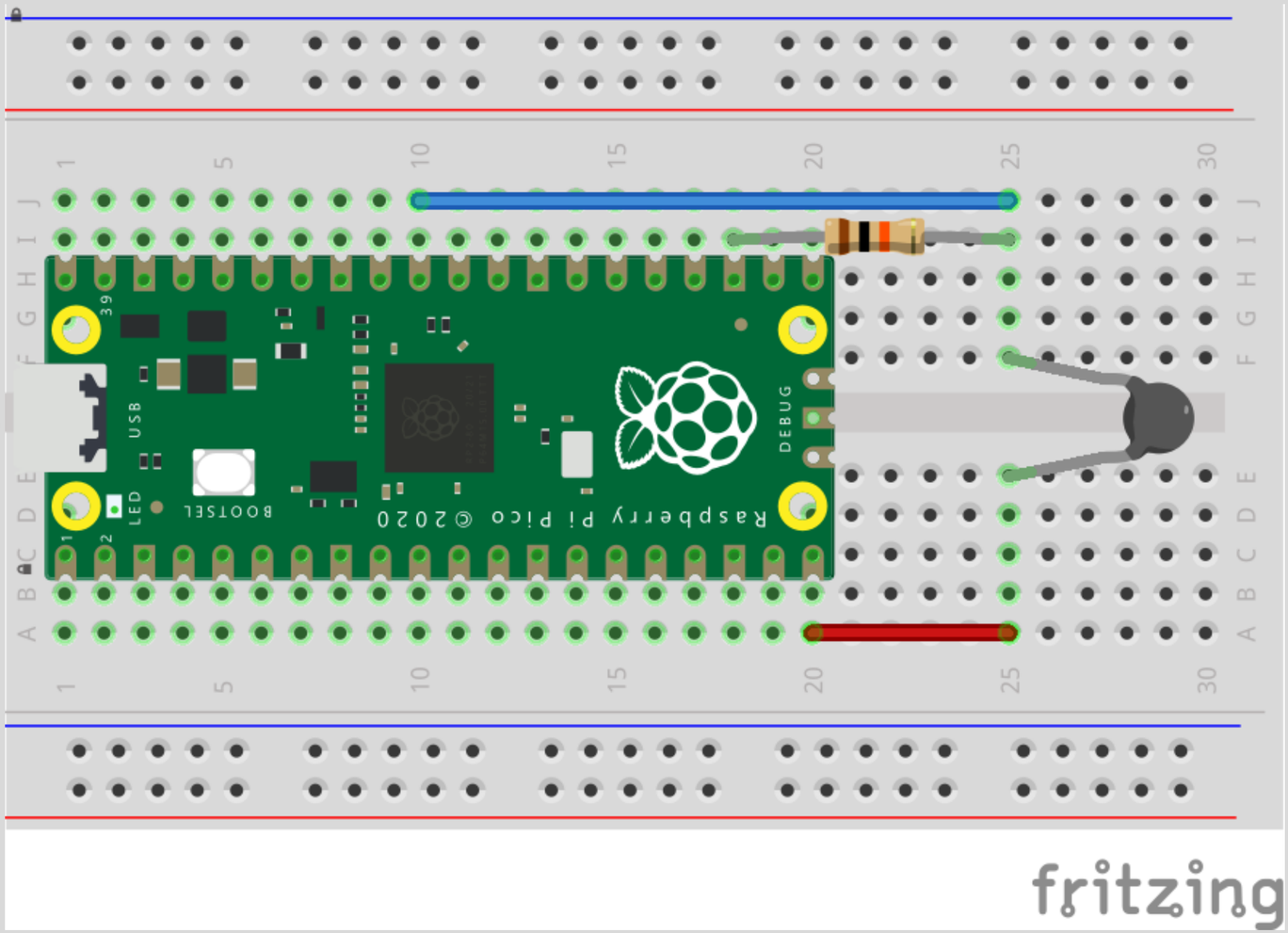
```
irb> require 'adc'
irb> adc = ADC.new(:temperature)
irb> adc.read_raw
irb> while true
irb*   voltage = adc.read_voltage
irb*   puts 27 - (voltage - 0.706) / 0.001721
irb*   sleep 1
irb* end
```

RP2040 has an in-chip temperature sensor that connects to an ADC channel

EURUKO

ADC --- Temperature by discrete parts

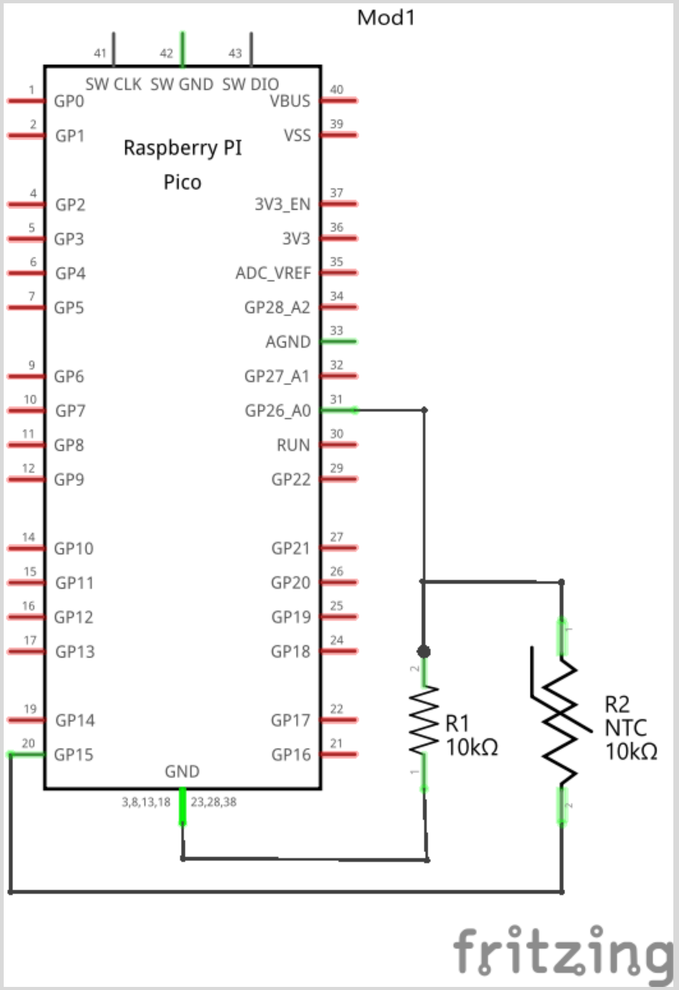
- Parts list:
  - Resistor
    - Rref: 10kΩ
  - Thermistor
    - 10kΩ (at 25 °C = 298.15K)
    - B const: 3950
    - To: 298.15 (kelvin)



EURUKO

ADC --- Temperature by discrete parts

- Parts list:
  - Resistor
    - Rref: 10kΩ
  - Thermistor
    - 10kΩ (at 25 °C = 298.15K)
    - B const: 3950
    - To: 298.15 (kelvin)



EURUKO

ADC --- Temperature by discrete parts

```
irb> require 'adc'
irb> Rref = 10000.0
irb> B = 3950.0
irb> T0 = 298.15
irb> def kelvin_temp(rth)
irb*   temp_inverse = 1 / B * Math.log(rth / Rref) + (1 / T0)
irb*   1 / temp_inverse
irb* end
irb> rth = (3.3 / adc.read_voltage - 1) * Rref
irb> puts "#{kelvin_temp(rth) - 273.15} C"
=> 28.1234 C
```

EURUKO

Part 3

Exploring PicoRuby Further

EURUKO

PicoRuby applications

- R2P2
  - Unix-like shell system written in PicoRuby
  - You may want to say an Operating System in Ruby
- PRK Firmware
  - Keyboard firmware framework for DIY keyboard
  - You can write your keymap and keyboard's behavior with Ruby

EURUKO

R2P2 (again)

- IRB
  - Multiple-line editor
- Built-in commands and executables (all written in Ruby)
  - You can write your own external command

EURUKO

Executables in R2P2

```
# date
puts Time.now.to_s

# mkdir
Dir.mkdir(ARGV[0])
```

EURUKO

Write a Ruby script file [Demo]

```
$> vim hello.rb
```

Edit the file and save it.

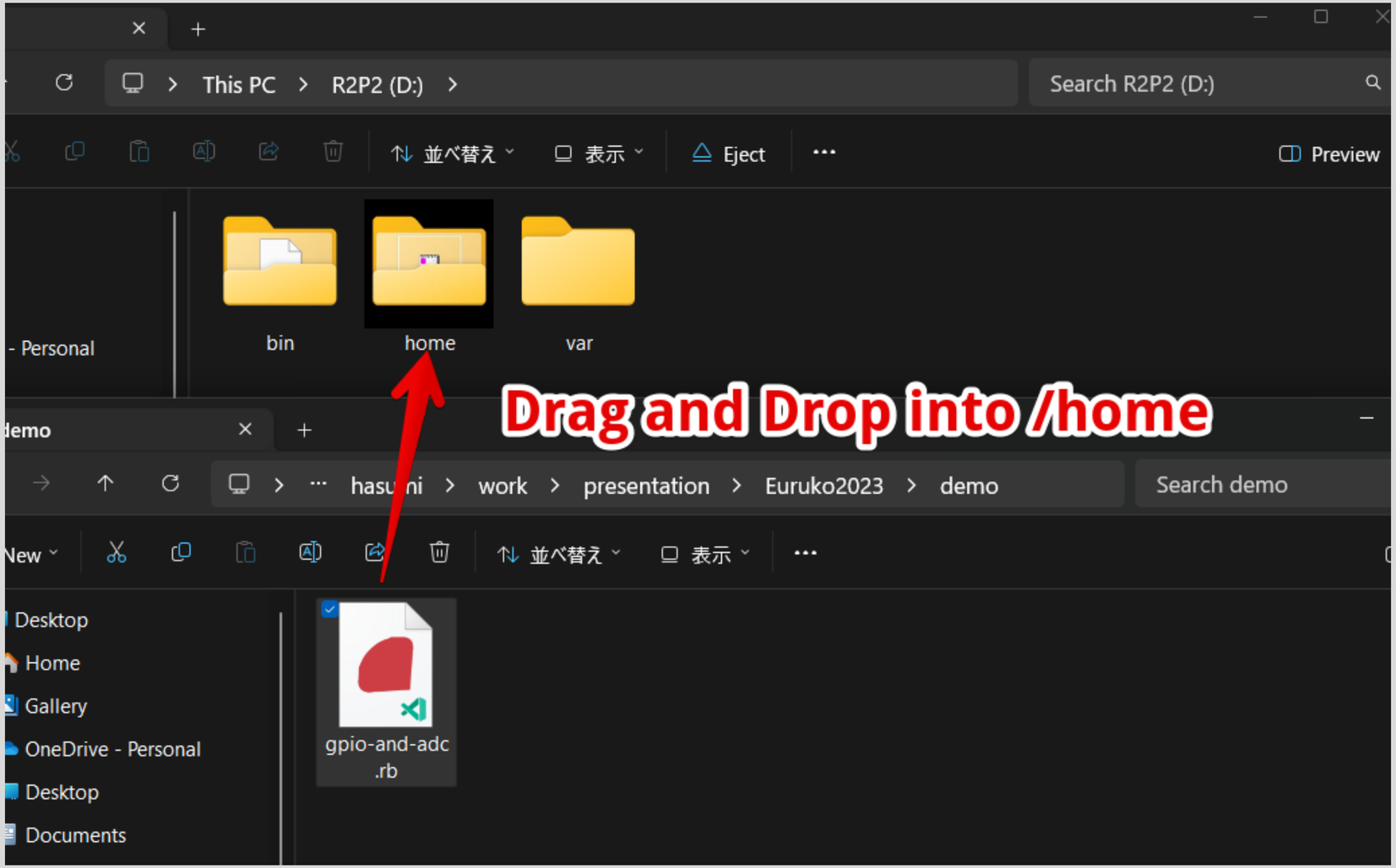
```
puts "Hello World!"
```

Then run it.

```
$> ./hello.rb
```

EURUKO

Or just drag and drop [Demo]



EURUKO

GPIO and ADC work together [Demo]

```
require 'adc'
def calc_temp(volt)
  27 - (volt - 0.706) / 0.001721
end
adc = ADC.new(:temperature)
led = GPIO.new(25, GPIO::OUT)
while true
  temp = calc_temp(adc.read_voltage)
  puts "temp: #{temp} C"
  led.write(30 < temp ? 1 : 0)
  sleep 1
end
```

EURUKO

R2P2 [Demo]

- `/home/app.rb` automatically runs on start up

```
# You can stop by Ctrl-C
led = GPIO.new(25, GPIO::OUT)
while true
  led.write 1
  puts "Hello World!"
  sleep 1
  led.write 0
  sleep 1
end
```

EURUKO

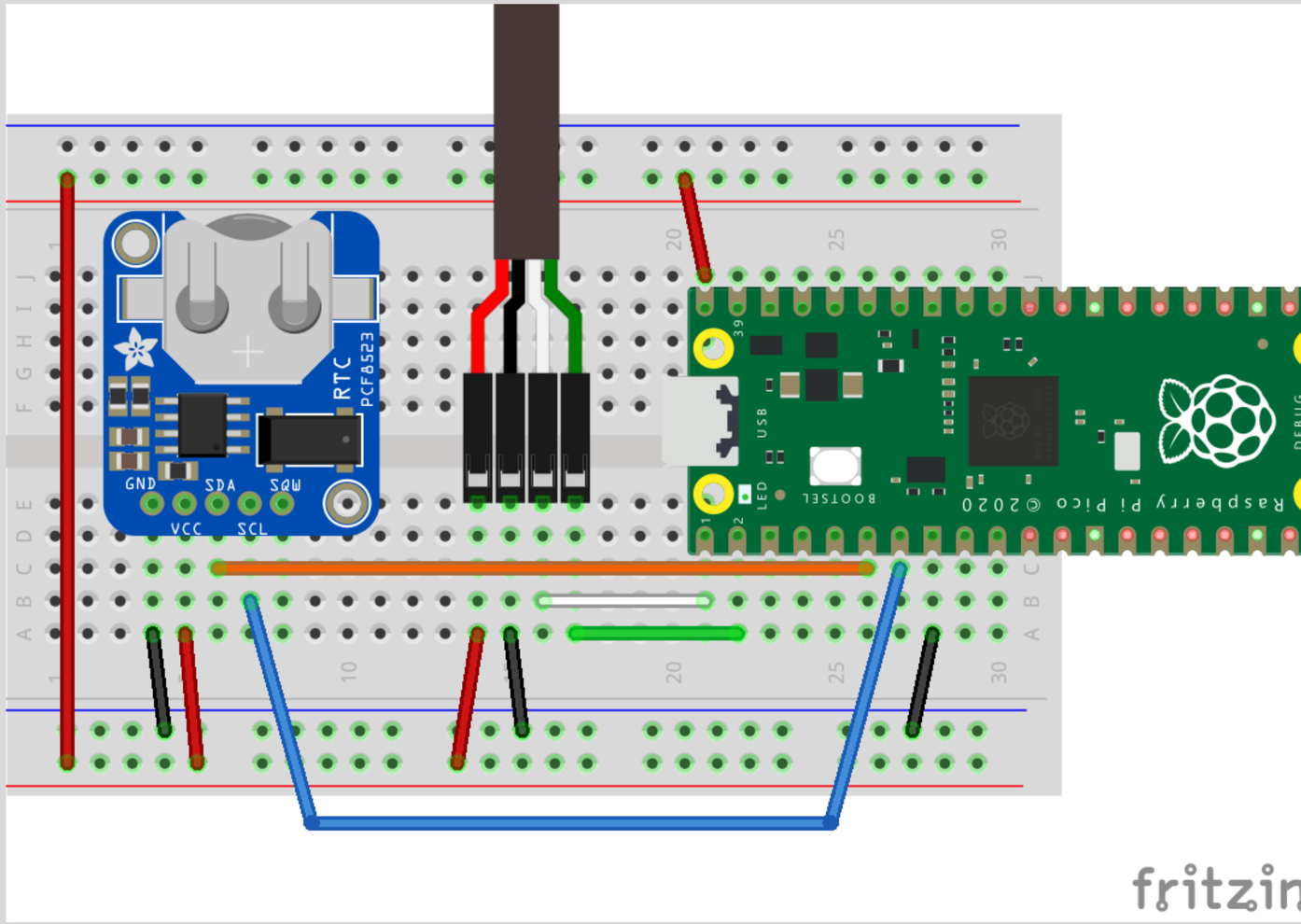
Serial communication protocols

- SPI: High speed, full duplex. e.g. Acceleration sensor, Color display, etc.
- I2C: Low speed, Addressing network with fewer wires. e.g. RTC, Temperature sensor and Charactor display, etc.
- UART: Buffered asyncronous communication. e.g. Terminal emulator, Wireless module like BLE and LTE/5G, etc.

EURUKO

I2C and UART

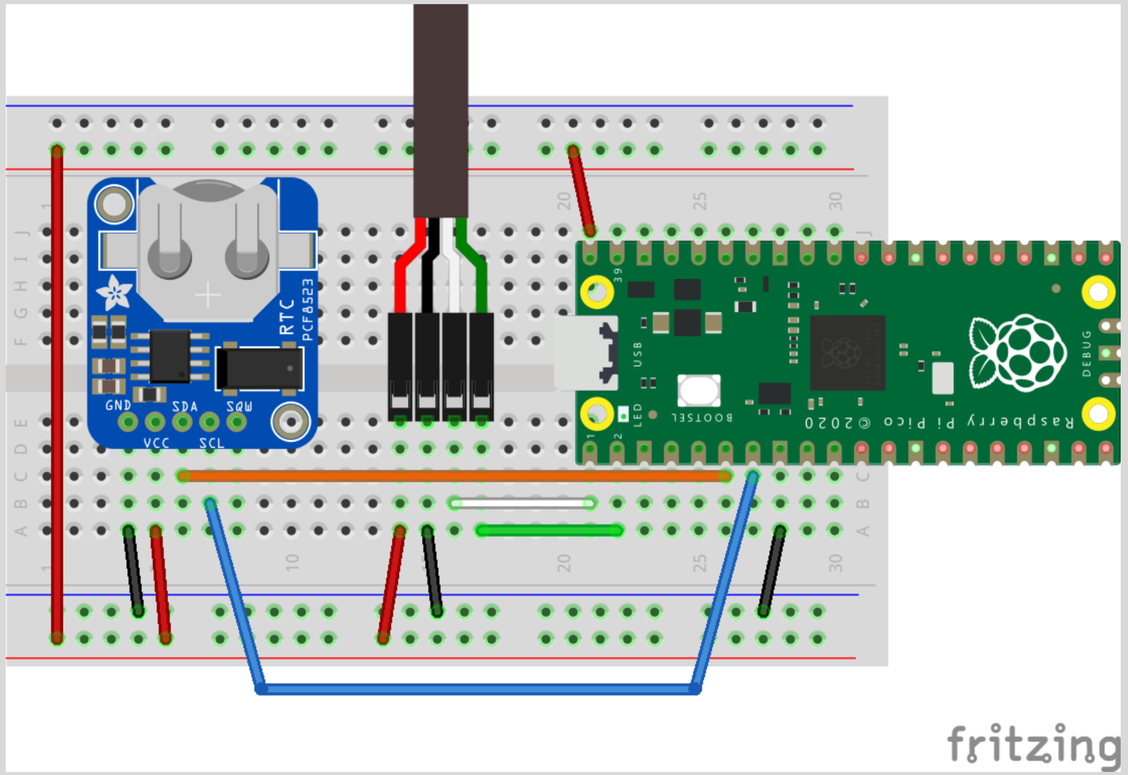
- Parts list:
  - PCF8523 RTC module
  - FTDI USB to TTL Serial Adapter Cable (3.3V)



EURUKO

I2C and UART

- github.com/picoruby/rp2040-peripheral-demo
  - Example of I2C (RTC) and UART (USB serial)
  - Watch the demo video in README.md
  - Also an example of how to build your own app



EURUKO

PRK Firmware - Corne (CRKBD)

EURUKO

PRK Firmware - Meishi2

```
require "consumer_key"
kbd = Keyboard.new
kbd.init_pins(
  [ 6, 7 ],    # row0, row1
  [ 28, 27 ]   # col0, col1
)
kbd.add_layer :default, %i[ RAISE  KC_2  KC_A  KC_4 ]
kbd.add_layer :raise,   %i[ RAISE
                            KC_AUDIO_VOL_UP
                            KC_AUDIO_VOL_DOWN
                            KC_AUDIO_MUTE ]
kbd.define_mode_key :RAISE, [ :KC_SPACE, :raise, 200, 200 ]
kbd.start!
```

EURUKO

Part 4

PicoRuby Under the Hood

EURUKO

mruby and PicoRuby

- mruby
  - General purpose embedded Ruby implementation written by Matz
- PicoRuby
  - Another implementation of murby targeting on one-chip microcontroller (smaller foot print)
  - Based on the mruby's VM code standard

EURUKO

Small foot print

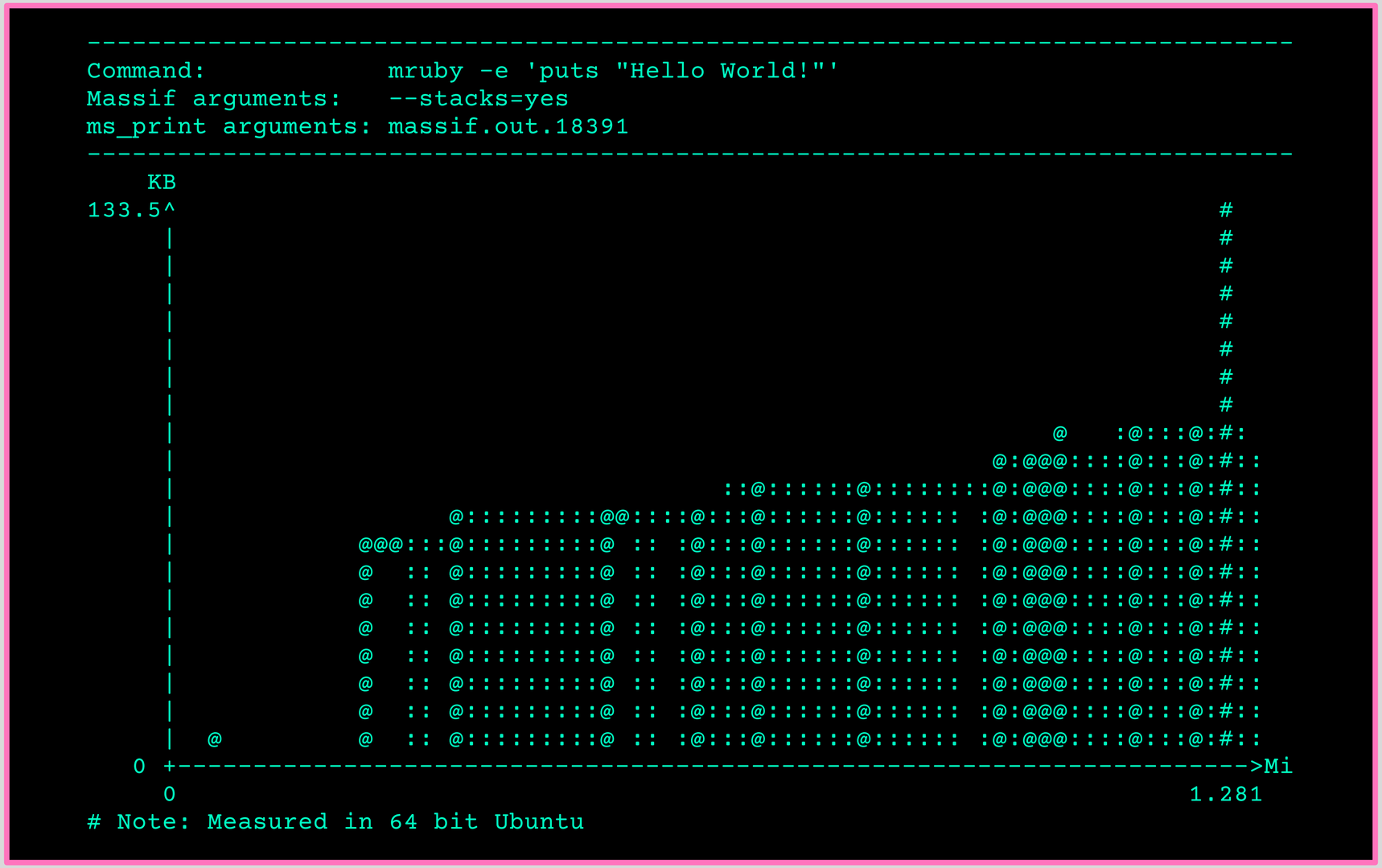
```
$ valgrind                                  \
    --tool=massif                           \
    --stacks=yes                            \
    path/to/(mruby|picoruby)                \
    -e 'puts "Hello World!"'
```

`massif.out.[pid]` file will be created. Then,

```
$ ms_print massif.out.1234 | less
```

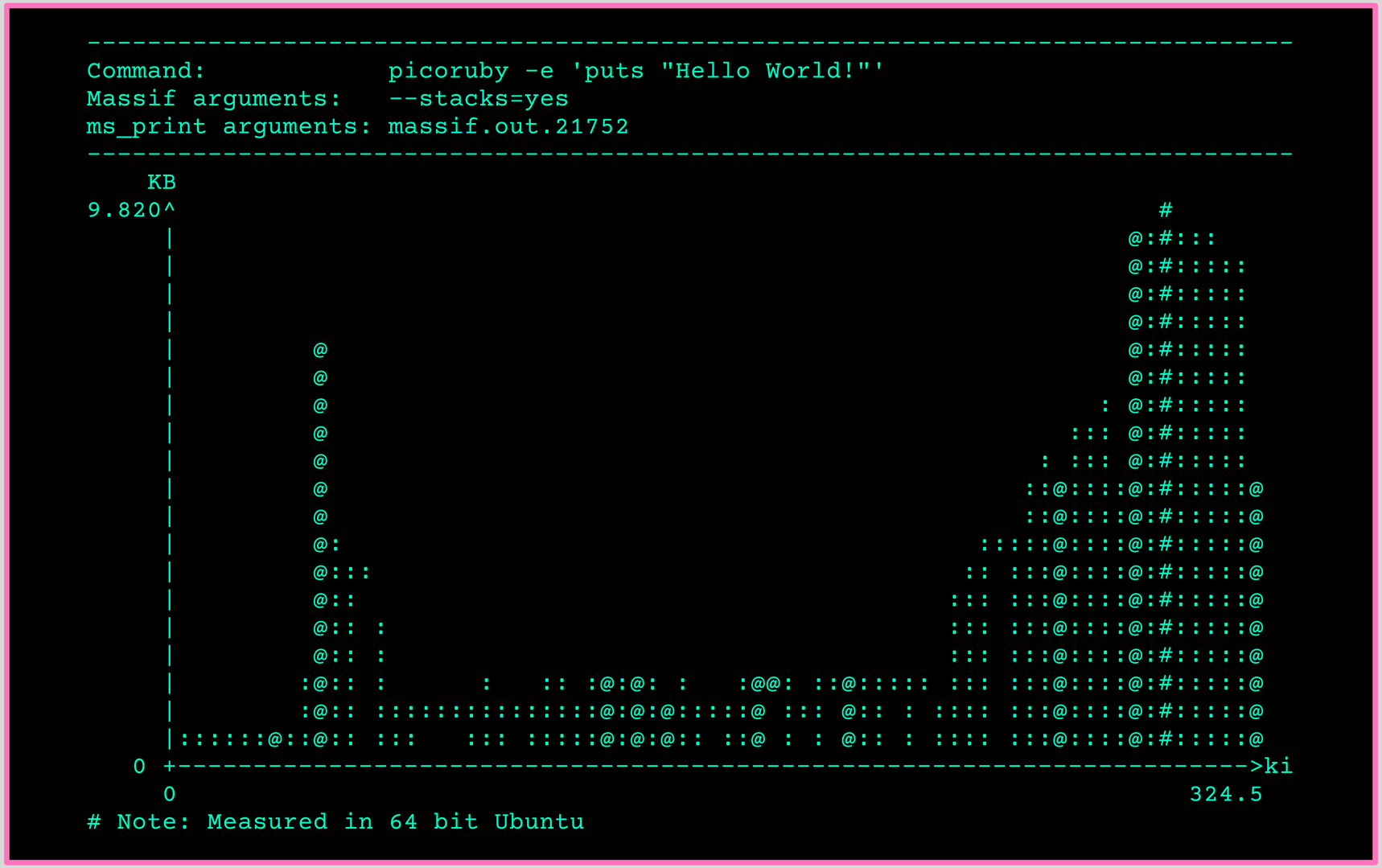
EURUKO

Small foot print



EURUKO

Small foot print



EURUKO

Small foot print

- RAM consumption of `puts "Hello World!"`
  - mruby: 133.5 KB (on 64 bit)
  - PicoRuby: 9.82 KB (on 64 bit)
- RP2040 (32 bit) has 264 KB RAM
  - Only small applications written in mruby should work
  - R2P2 and PRK Firmware should be written in PicoRuby

PicoRuby ecosystem

- Picogems
  - PRK Firmware is also a Picogem
  - Peripheral gems
    - picoruby-gpio
    - picoruby-adc
    - picoruby-i2c
    - picoruby-spi
    - picoruby-uart
  - Peripheral interface guide
    - https://github.com/mruby/microcontroller-peripheral-interface-guide

EURUKO

PicoRuby ecosystem

- Build system forked from mruby
  - You can build your application in a similar way to mruby
  - You can also write your gem and host it on your GitHub
- RP2040 is the only target as of now though,
  - Carefully designed to keep portability

EURUKO

Restriction of PicoRuby

- Minimum built-in classes and methods
- Doesn't support
  - Some syntax like heredoc and numbered parameters
  - Module due to VM implementation
- No strict distinction between instance methods and singleton methods
- Some bugs (because I'm lazy🥺).
  See github.com/picoruby/picoruby/issues

EURUKO

Conclusion

- PicoRuby is a Ruby implementaiton targeting on one-chip microcontroller
- You can develop your microcontroller application step by step using the R2P2 and IRB
- You need only R2P2 to run small applications
- Future work
  - BLE and WiFi with Raspberry Pi Pico W (soon)
  - Porting to microcontrollers other than RP2040 (someday)

EURUKO

RubyKaigi 2024

In Okinawa🏝️ May 15th - 17th

1000+ attendees, tons of tech talks and various parties🍻

https://098free.com/photos/14262/

That's all! Visit repos and stargaze🌟

github.com/picoruby/picoruby

github.com/picoruby/R2P2

github.com/picoruby/prk_firmware

github.com/picoruby/rp2040-peripheral-demo

EURUKO