A “Kaigi” for DIY Keyboard enthusiasts

KeebKaigi

2023

Matsumoto

Pointing Device On The Partner Half

hasumikin

Mar. 10, 2023

Today's mood

- Sharing how a keybosrd firmware creator thinks
- We won't get a conclusion today
- New ideas are welcome!

PRK Firmware - Outline

- QMK inspired firmware
- Written in PicoRuby
- You don't need to build a firmware
- Instead, just drag and drop a `keymap.rb`

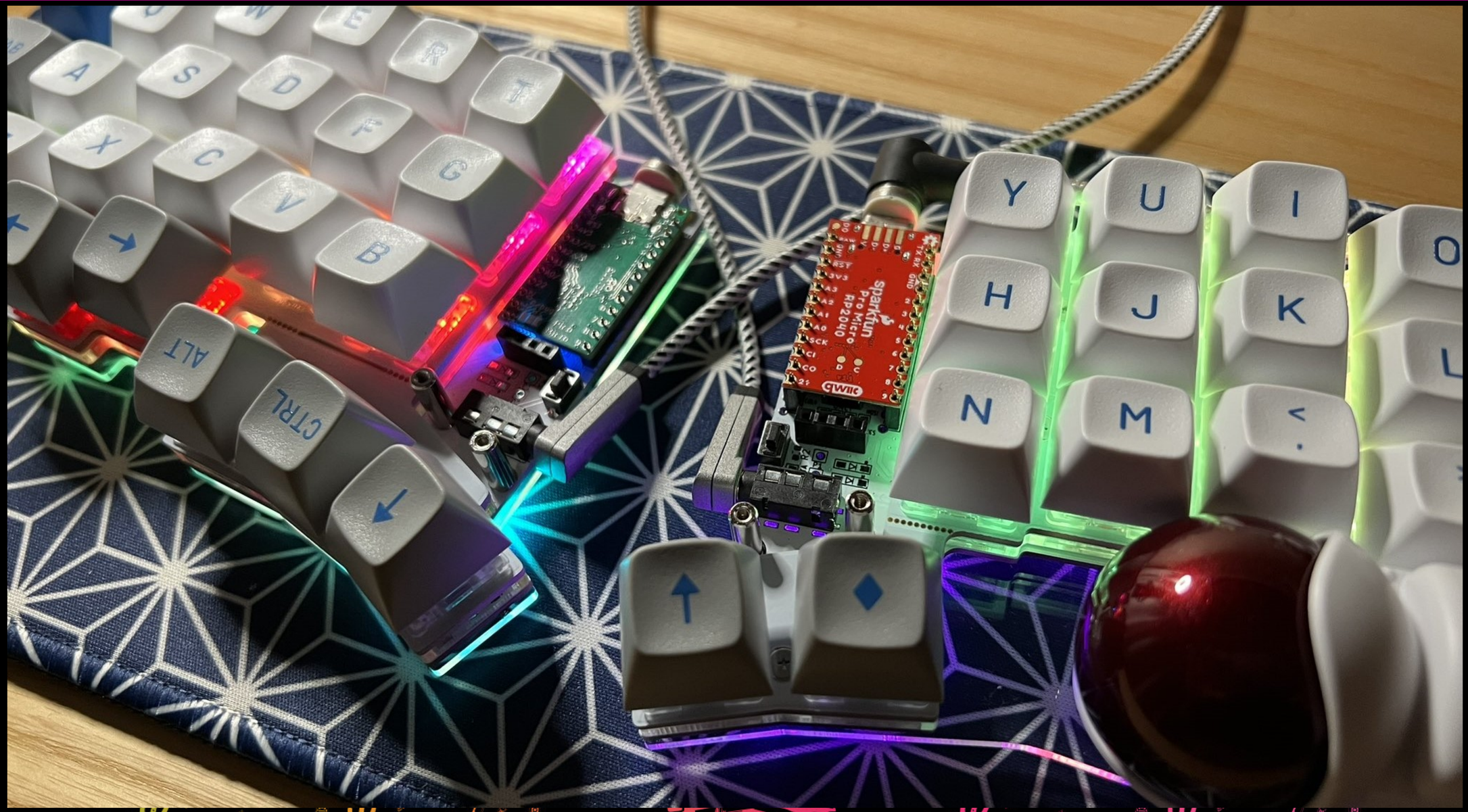
PRK Firmware - Features

- Split-type, Consumer key
- Num/Caps/Scroll lock
- Layer, Debounce, Composite key
- Rotary encoder, Joystick
- RGBLED, Sounder

PRK Firmware - Coming next

- Track ball / Touchpad / Joystick as a **Mouse**
- You can configure a device based on I2C, SPI and ADC
- PicoRuby has implemented general peripheral classes

PRK Firmware - Track ball



PRK Firmware - I2C Track ball

```
require "i2c"
require "mouse"

i2c = I2C.new(unit: :RP2040_I2C0, frequency: 100_000,
              sda_pin: 20, scl_pin: 21)
mouse = Mouse.new(driver: i2c)
mouse.task do |mouse, keyboard|
  left, x, up, y, push = mouse.driver.read(0x0a, 5).bytes
  x = -left if 0 < left
  y = -up if 0 < up
  # LEFT: 0b001, RIGHT: 0b010, MIDDLE: 0b100
  button = (push == 128 ? 0b100 : 0)
  if keyboard.layer == :default
    USB.merge_mouse_report(button, x, y, 0, 0)
  else
    # Works as a scroll wheel when layer is changed
    USB.merge_mouse_report(button, 0, 0, x, -y)
  end
end
kbd.append mouse
```

An issue of split-type

- Pointing device works on only "Anchor"
  - Anchor: The half that connects to USB
  - Partner: The opposite

An issue of split-type

- Pointing device works on only "Anchor"
  - Anchor: The half that connects to USB
  - Partner: The opposite

I know you want to place balls on both sides

UART - Communication protocol

```
#   Simplex: One wire (TRS)
    Anchor RX <---- TX Partner

#   Full duplex: Two wires (TRRS)
    Anchor TX ----> RX Partner
    Anchor RX <---- TX Partner

#   Half duplex: One wire (TRS)
Anchor RX/TX <===> TX/RX Partner
```

Bit-banging half duplex

- Manually manipulating GPIO to simulate a UART communication protocol
- Manually toggling the TX and RX to send and receive data on **one wire**
- MCU may breaks if both sides become TX at the same time

PRK Fimeware - Bit-banging


```
Anchor(TX) ---> Partner(RX)
   Stop-bit      : 11111111
   Start-bit     : 0
   Data          : 87654321
   Stop-bit      : 11
   Extra-bit     : 0
-----------------------------------------------------
 Demilitarized zone (both are RX)
   DMZ           : 0000
-----------------------------------------------------
 Anchor(RX) <--- Partner(TX)
   Idle-bit      : 0000
   Stop-bit      : 1
   Start-bit     : 0
   Data1,2,3     : 87654321 87654321 87654321
   Stop-bit      : 1
-----------------------------------------------------
Anchor(TX) ---> Partner(RX)
   Idling        : 0
```

PRK Fimeware - Bit-banging

- In a bit-bang cycle,
  - Anhcor sends 8 bits then receives 24 bits
  - Partner receives 8 bits then sends 24 bits

What can 8-bit represent?

```
# One key switch
  87654321
  ^^^      row number (0 to 7)
     ^^^^^ col number (0 to 31)

# Four rotary encoders
  87654321
  ^^  ^^
    ^^  ^^ 2-bit each (-1,0,1)
```

What can 8-bit represent?


```
# One coarsened axis data of an analog stick
  87654321 (0 to 255. Originally 12 bits: 0 to 4095)

# One axis and one button of a pointing device
  87654321
   ^^^^^^^ Signed 7 bit (-64 to 63)
  ^        Button push
```

How many bits are required?

- A key switch: 8 bits
- A rotary encoder: 2 bits
- A joystick (X, Y): 16 bits (ideally 24 bits)
- A pointing device (X, Y, 2 buttons): 16 bits
  - If more buttons or multiple gestures: 17+ bits

What if a track ball on partner?

- We'd need 40 bits (5 bytes) data width
  - 24 bits (3 bytes) for three key switches
  - 16 bits (2 bytes) for a track ball

24 bits width would be short


```
Anchor(TX) ---> Partner(RX)
   Stop-bit      : 11111111
   Start-bit     : 0
   Data          : 87654321
   Stop-bit      : 11
   Extra-bit     : 0
-------------------------------------------------------
 Demilitarized zone (both are RX)
   DMZ           : 0000
-------------------------------------------------------
 Anchor(RX) <--- Partner(TX)
   Idle-bit      : 0000
   Stop-bit      : 1
   Start-bit     : 0
   Data1,2,3     : 87654321 87654321 87654321
   Stop-bit      : 1
-------------------------------------------------------
Anchor(TX) ---> Partner(RX)
   Idling        : 0
```

Plan A: Expand the data width

```
Anchor(TX) ---> Partner(RX)
   Stop-bit      : 11111111
   Start-bit     : 0
   Data          : 87654321
   Stop-bit      : 11
   Extra-bit     : 0
-------------------------------------------------------------
 Demilitarized zone (both are RX)
   DMZ           : 0000
-------------------------------------------------------------
 Anchor(RX) <--- Partner(TX)
   Idle-bit      : 0000
   Stop-bit      : 1
   Start-bit     : 0
   Data1,2,3,4,5: 87654321 87654321 87654321 87654321 87654321
   Stop-bit      : 1                                  👆👆👆👆 👆👆👆👆
-------------------------------------------------------------
Anchor(TX) ---> Partner(RX)
   Idling        : 0
```

Plan A: Expand the data width

- Plan A: Easy to implement but,
  - What if one needs 6 bytes?
  - Joystick and rotary encoders and,..... 🔥

Plan B: Multiple comm threads

- Divide threads into:
  - Thread 1: 24 bits for three key switches
  - Thread 2: 24 bits for user difined device
  - Each input sustains for two thread cycles
- Three or more threads?🤔

Plan B: Multiple comm threads

- Plan A: Easy to implement but,
  - What if one needs 6 bytes?
  - Joystick and rotary encoders and,..... 🔥
- Plan B: Harder to implement 🐛
- Both: Poorer performance?

Plan C: ??????????????????????

If you have an idea,

reach out to

(twitter | github).com/

hasumikin

KEGAKAIGI
2023

Thank you!!!!q