きわめる。拓く。創り出す。

長崎総合科学大学

Nagasaki Institute of Applied Science

組み込みシステムの基礎

改定

20211216 ：OSのインストール方法を更新、Python2→3に変更

20230206：インデントについて、関数について、入力について追記

20231212：誤字・リンクの差し替え

20231219：Pi OSセットアップ画面の設定を追加

20240110：細かな内容の変更

20240718：Pi 5に対応

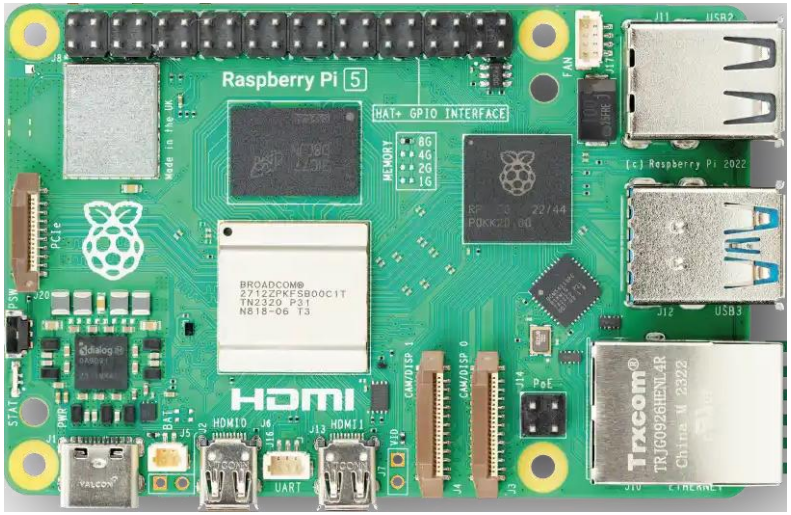
20241107：Pi 5のGPIOのライブラリ変更に合わせた内容の変更

はじめに

- 現在の電気電子機器の制御においては、小型マイクロプロセッサを用いる方式が主流
- 本実験ではRaspberry Pi (ラズベリーパイ)と呼ばれる超小型コンピュータを用いて、その初期設定、プログラミングの手法、これを用いた制御に取り組むことで、組み込みシステムの基礎を身に着ける
- 内容(予定)
  - 第一回目: Raspberry Piのセットアップ, OSのインストール
  - 第二回目: エディタの使い方、プログラムを書いてみる
  - 第三回目: LEDを点灯してみる、スイッチを読み取ってみる
  - 第四回目: 独自の回路を制御してみる
- レポート
  - 第一回目～第三回目は予習・宿題を行ってノートで次回に報告
  - 最終レポートとして、動画を作成し Youtube にアップロードする

Raspberry Pi 5

- Raspberry Pi (以下RP)は、イギリスのRaspberry Pi Foundationが2012年に開発したプログラミング学習・組み込み機器用超小型コンピュータ
- 例えばRP5では2.4GHz Arm Cortex-A76 CPU、4GBのメモリ(8GB版もあり)、EthernetおよびUSBポートを備え、LinuxがなどのOperating Systemを走らせることが可能
- ハードディスクは備えていないが、代わりにMicro SDカードスロットを持つので、これにOSをインストールして使用
- 40 pinのGPIOポートを備え、これを用いて様々な機器の制御実験等を行うことが可能



Raspberry Pi5の外観

「https://www.raspberrypi.com/products/raspberry-pi-5/」より画像の引用

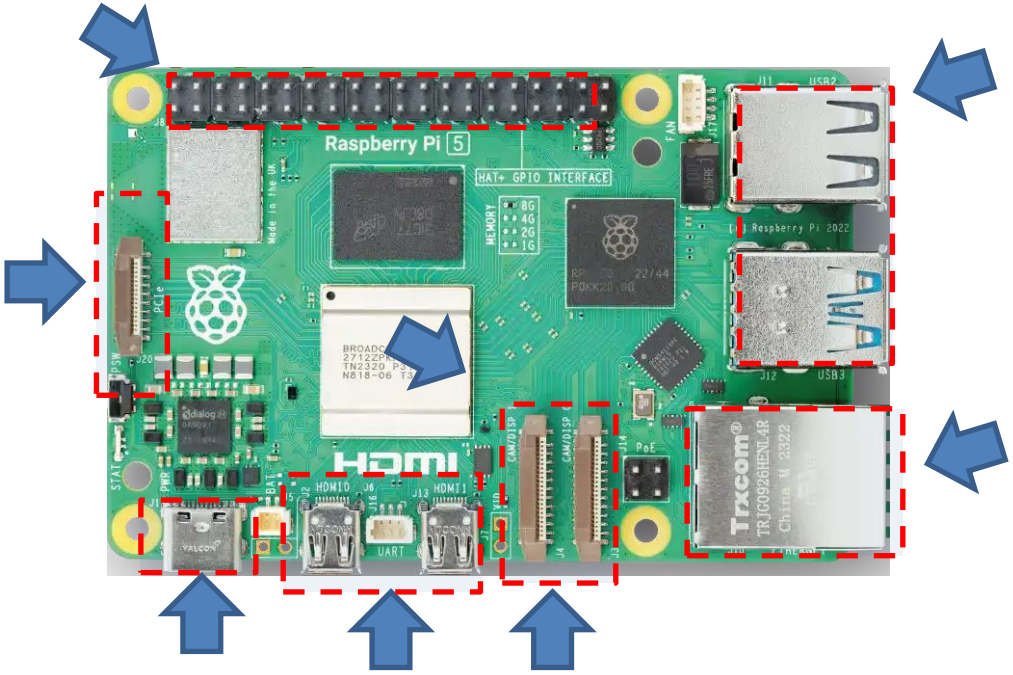
Raspberry Pi5のスペック概要

Specification

- Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptography extensions, 512KB per-core L2 caches and a 2MB shared L3 cache
- VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2
- Dual 4Kp60 HDMI® display output with HDR support
- 4Kp60 HEVC decoder
- LPDDR4X-4267 SDRAM (4GB and 8GB SKUs available at launch)
- Dual-band 802.11ac Wi-Fi®
- Bluetooth 5.0 / Bluetooth Low Energy (BLE)
- microSD card slot, with support for high-speed SDR104 mode
- 2 × USB 3.0 ports, supporting simultaneous 5Gbps operation
- 2 × USB 2.0 ports
- Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT)
- 2 × 4-lane MIPI camera/display transceivers
- PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or other adapter)
- 5V/5A DC power via USB-C, with Power Delivery support
- Raspberry Pi standard 40-pin header
- Real-time clock (RTC), powered from external battery
- Power button

調査事項（課題）

- Operating Systemとは何か
- ARM CPUとは何か
- GPIOとは何か
- モデル名は？PICO、ZEROなど。
- それぞれのコネクタ(下図の矢印部分)が何であるかを調べてくること



OS

- RPの動作には、Operating System (OS)のインストールが必要
- 通常のコンピュータでは、ハードディスク内にファイルシステムを構築し、そこにOSを構成するファイル群をコピーする
- RPでは、このファイルシステムに相当する部分がはじめから一つのイメージファイルとしてインストーラとともに提供されているので、これをSDメモリカードにコピーして、RPに挿入すれば、それだけでOSをインストール可能
- 今回は、Pi OSというOSをインストールしてみる

OSのインストールの準備(1)

- SDカードにOSをインストールする準備をする
  - 中身を空っぽにする必要があるので、フォーマットを行う

- ブラウザ
  https://www.sdcard.org/jp/downloads/formatter_4
  にアクセスする
- ページ下部分にある”SDカードフォーマッター Windows用ダウンロード”をクリックし、ライセンス条項を呼んだあと、”同意します”をクリックする
- “SDFormatterv4.zip” というファイルが出来るので、これを展開する
- Setup.exeというファイルが出来るので、これを実行し、インストールを済ませる

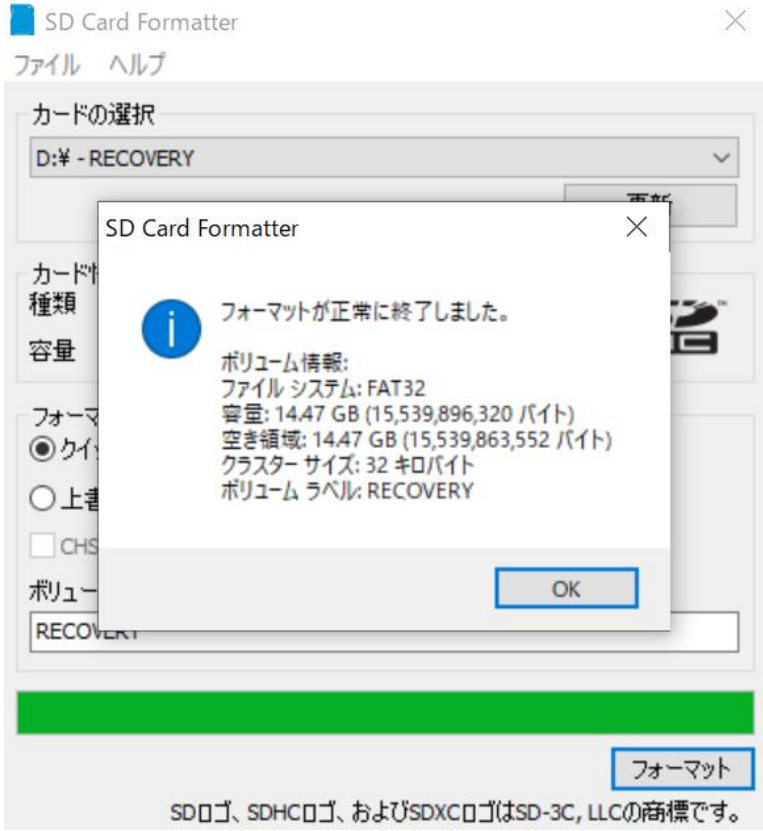
OSのインストールの準備(2)

- MicroSDカードをPCのカードリーダスロットに挿入する
- 自分のパソコンに無ければ、図にあるようなUSB接続のもの(実習用のセットのもの)をパソコンに挿入する

- SDカードはリムーバブルディスクとして認識される

OSのインストールの準備(3)

- SDFormatter を実行する
- Drive の部分に挿入したSDカードのドライブ名が表示されるのを確認
  - この時にSDカード以外のドライブは選択しないこと
- 16GBのSDだと、Sizeが14.4 GBくらいになる
  (1024kB 1000kBと数え方が異なるため、
  容量がすくなくなる)
- フォーマットボタンを押す
- “フォーマットが正常に終了しました”
  と表示されたら成功なので、OKを押し、
  SDFormatterを終了



OSインストール手順(1)

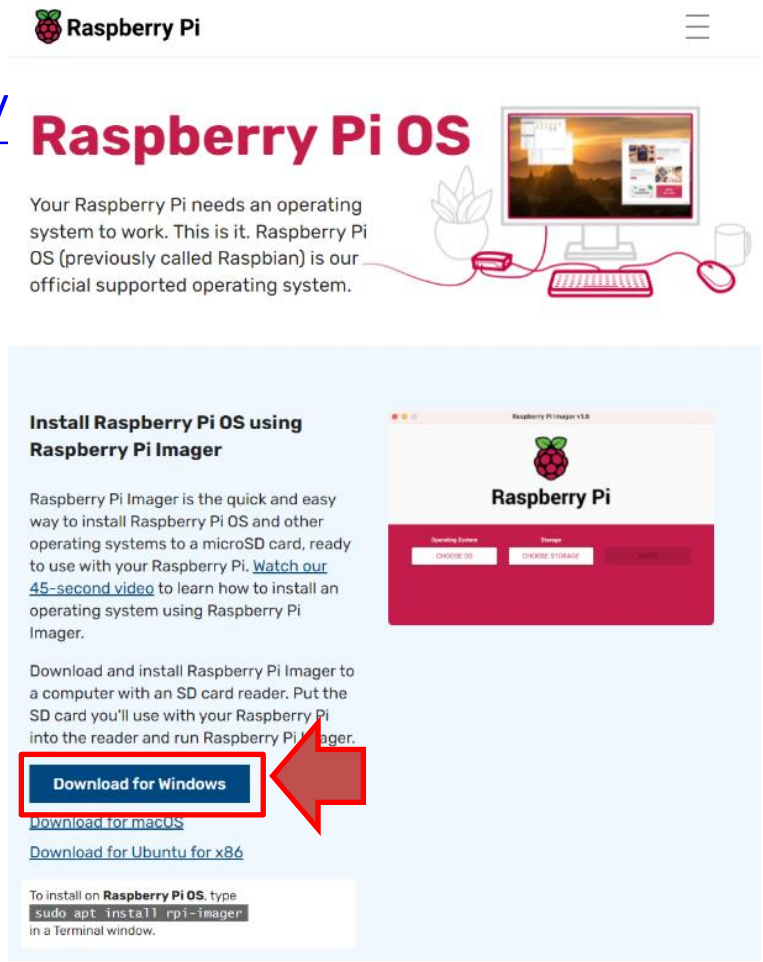
- ブラウザで次のページを開く
  https://www.raspberrypi.com/software/

- Download for Windows をクリック

- ダウンロードされた“imager_X.X.X.exe”(Xは数字)をクリックして実行する

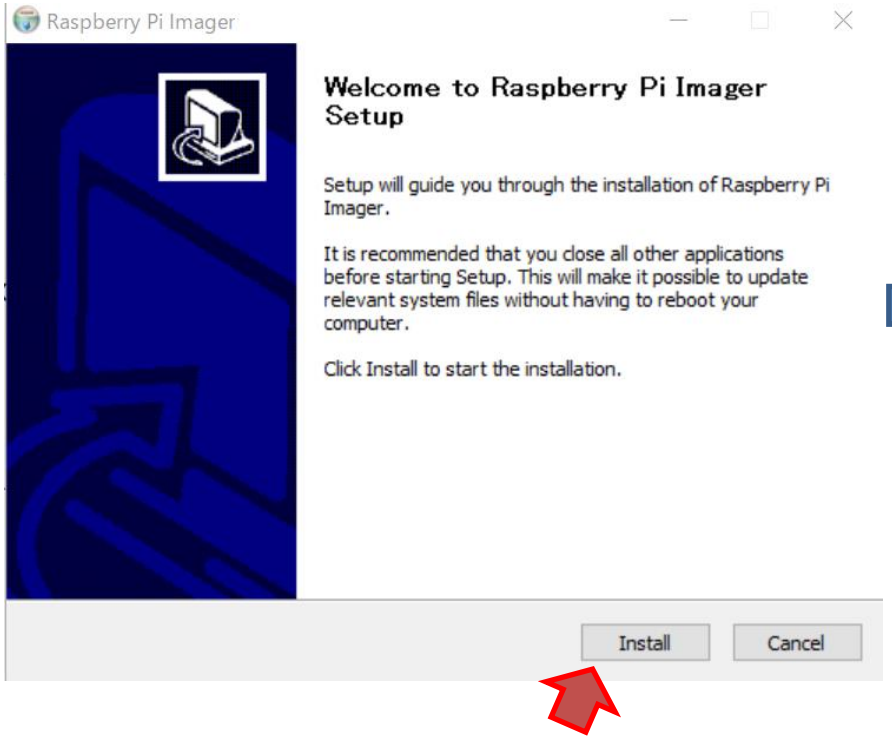
- 管理者権限は はい を押す

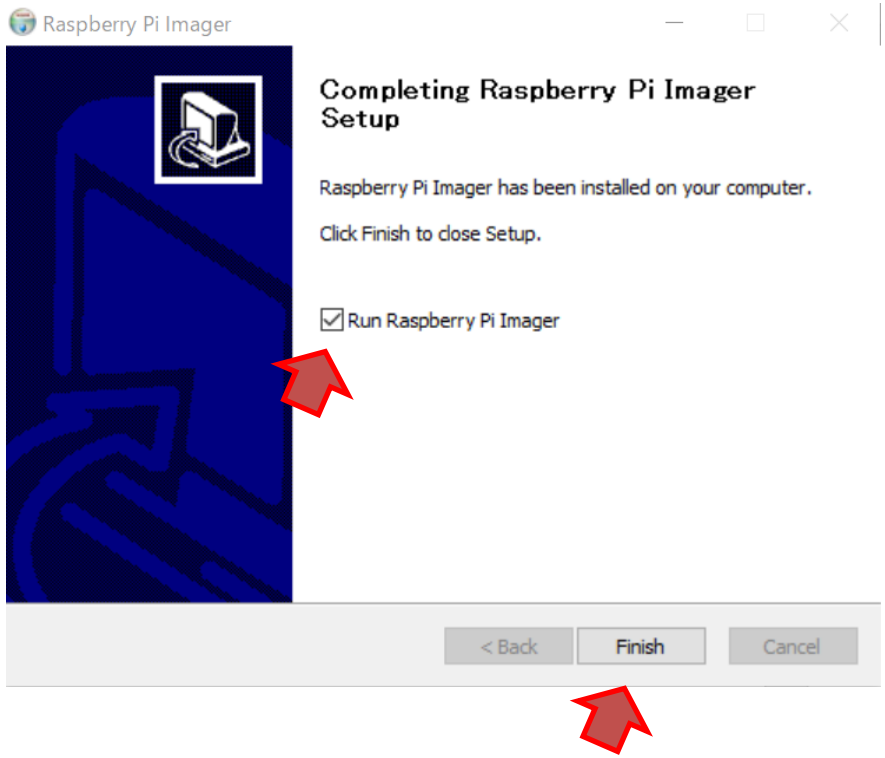
- インストーラーが起動したら次のスライドへ



OSインストール手順(2)

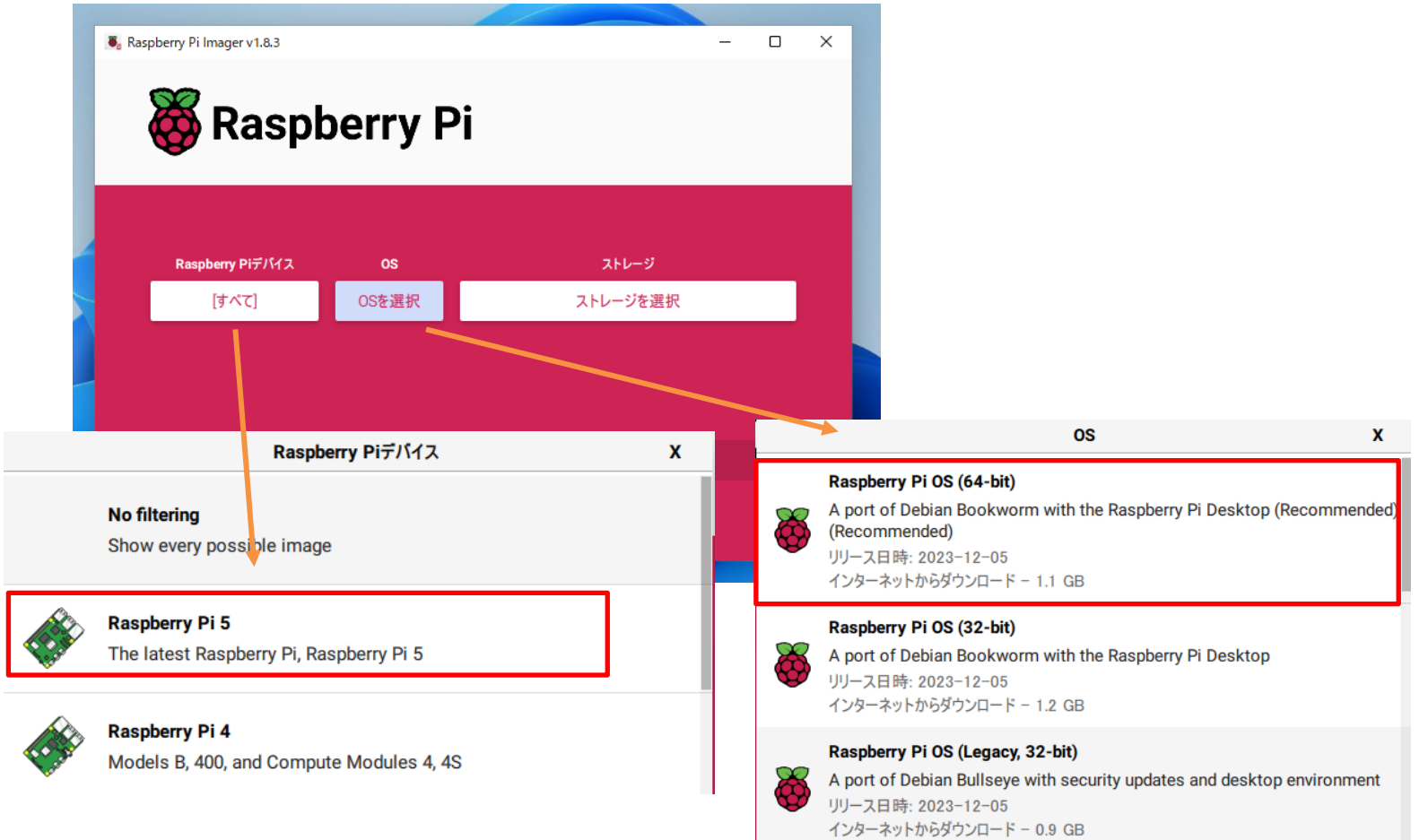
- 下記の画面ではインストールを選択し、Run raspberry Pi Imagerにチェックを入れる





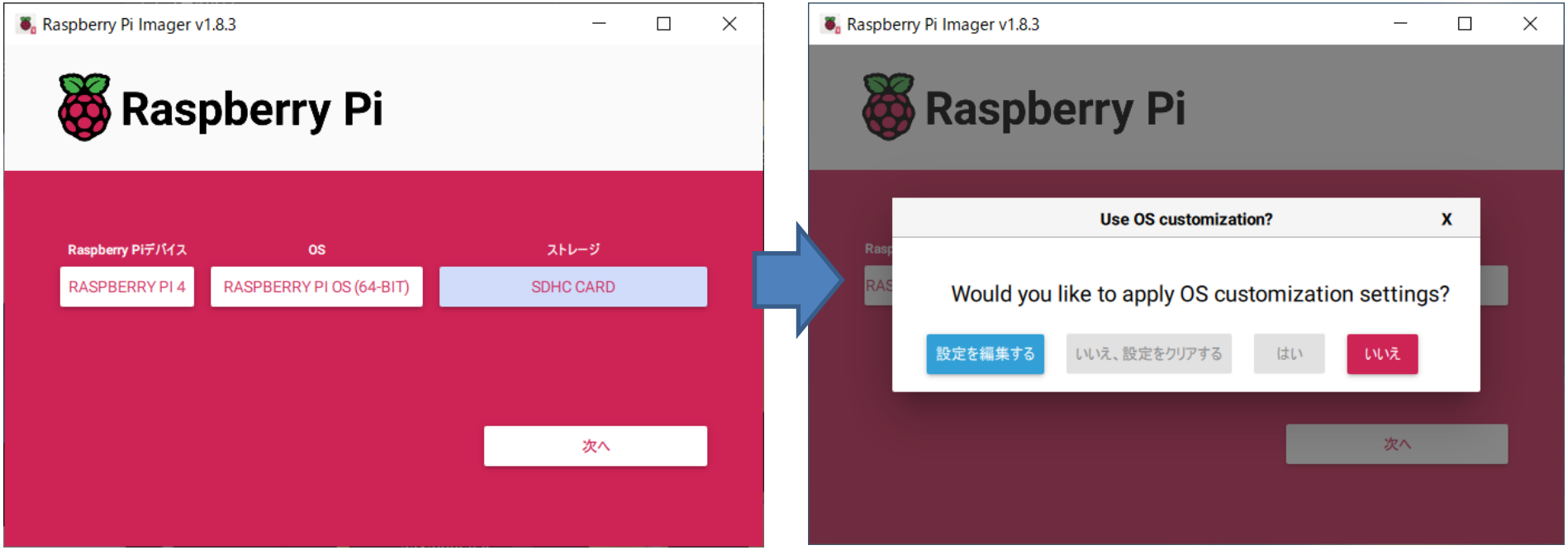
OSインストール手順(3)

- 「Raspberry Piデバイス」：使いたいデバイスを選択する。**5を選択**
- 「OSを選択」： **Raspberry Pi OS (64-bit)を選択**
- 「ストレージを選択」：フォーマットしたSDを選択



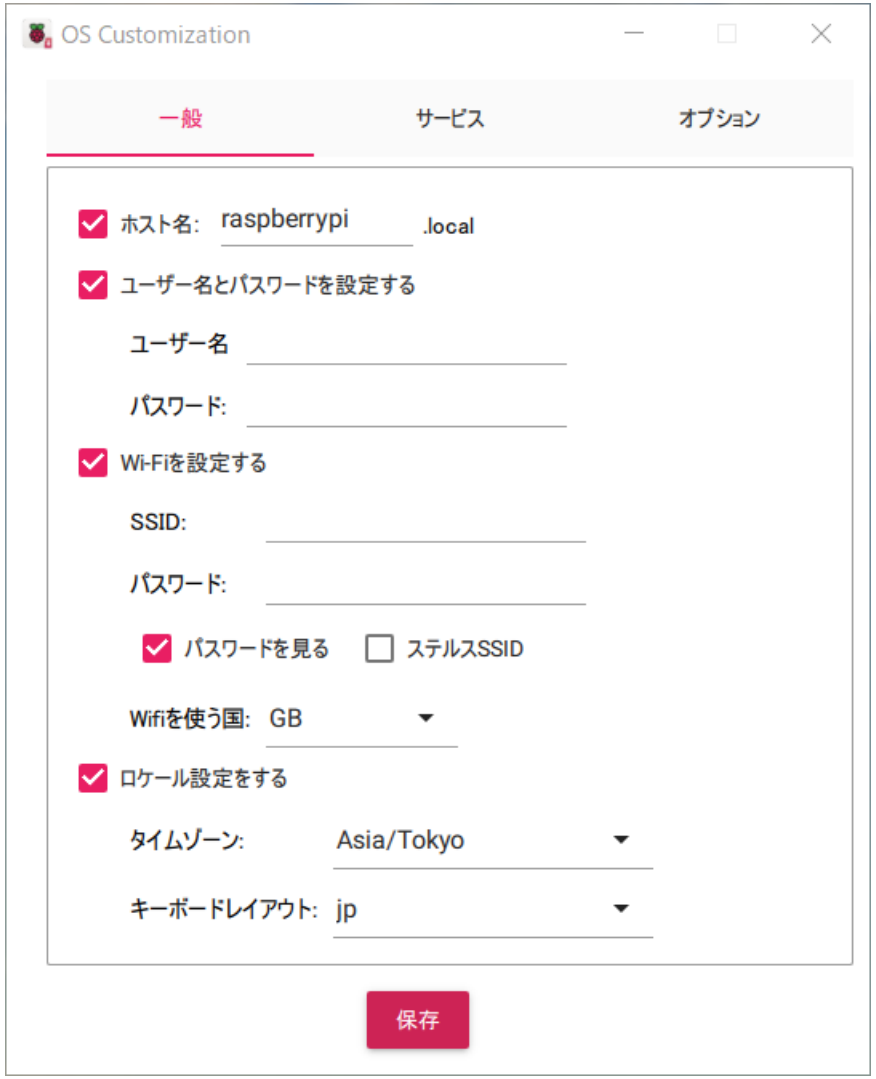
OSインストール手順(4)

- 「次へ」を選択し、ポップアップは「設定を編集する」を選択する。



OSインストール手順(5)

- 設定は下記のように行う。



- ホスト名
  - デフォルトのまま
- ユーザー名・パスワード
  - 忘れないものにする
- Wi-Fiを設定する
  - SSIDは、NIAS-GI
  - パスワードは、掲載の物を使う
- ロケール設定
  - タイムゾーン：Asia/tokyo
  - キーボードレイアウト：jp

※必ずjpにすること。
日本語配列のキーボード
を使用します。

OSインストール手順(6)

- 書き込み状況の進捗が表示される
- だいたい数分～15分程度

書き込み画面
書き込み中は、SDカードリーダーを触らない

OSインストール手順(7)

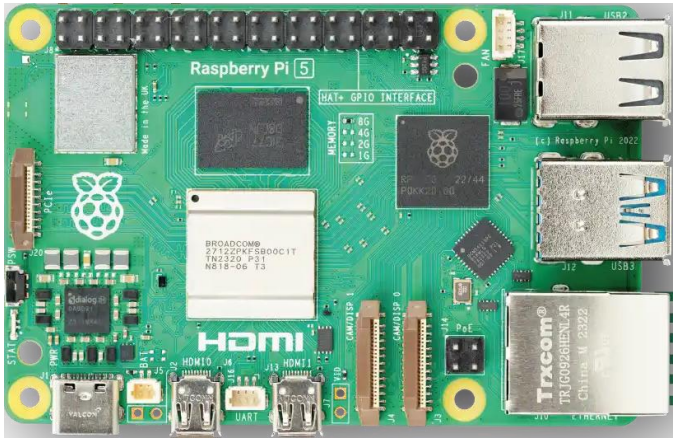
- 下図のように完了したらパソコンからSDカードを外す。

- この時にSDカードをフォーマットしますかと聞かれるが無視する
  →書き込んだデータがフォーマットされてしまいます。なぜか？

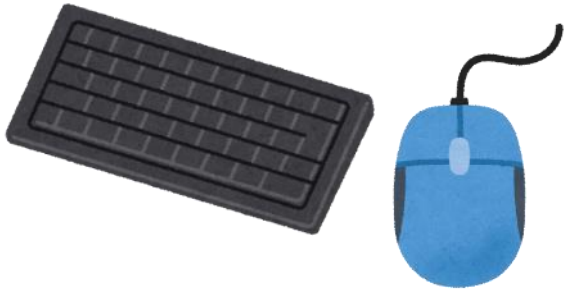
起動の準備

- RPにマウス、キーボード、モニタ、SDカードを接続

OSをインストール
したSD



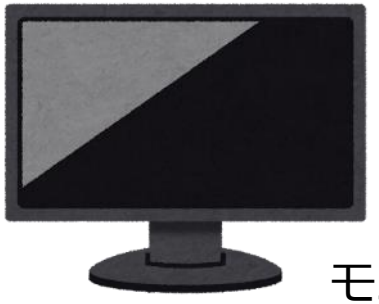
マウス・キーボード



HDMIケーブル

ACアダプタ

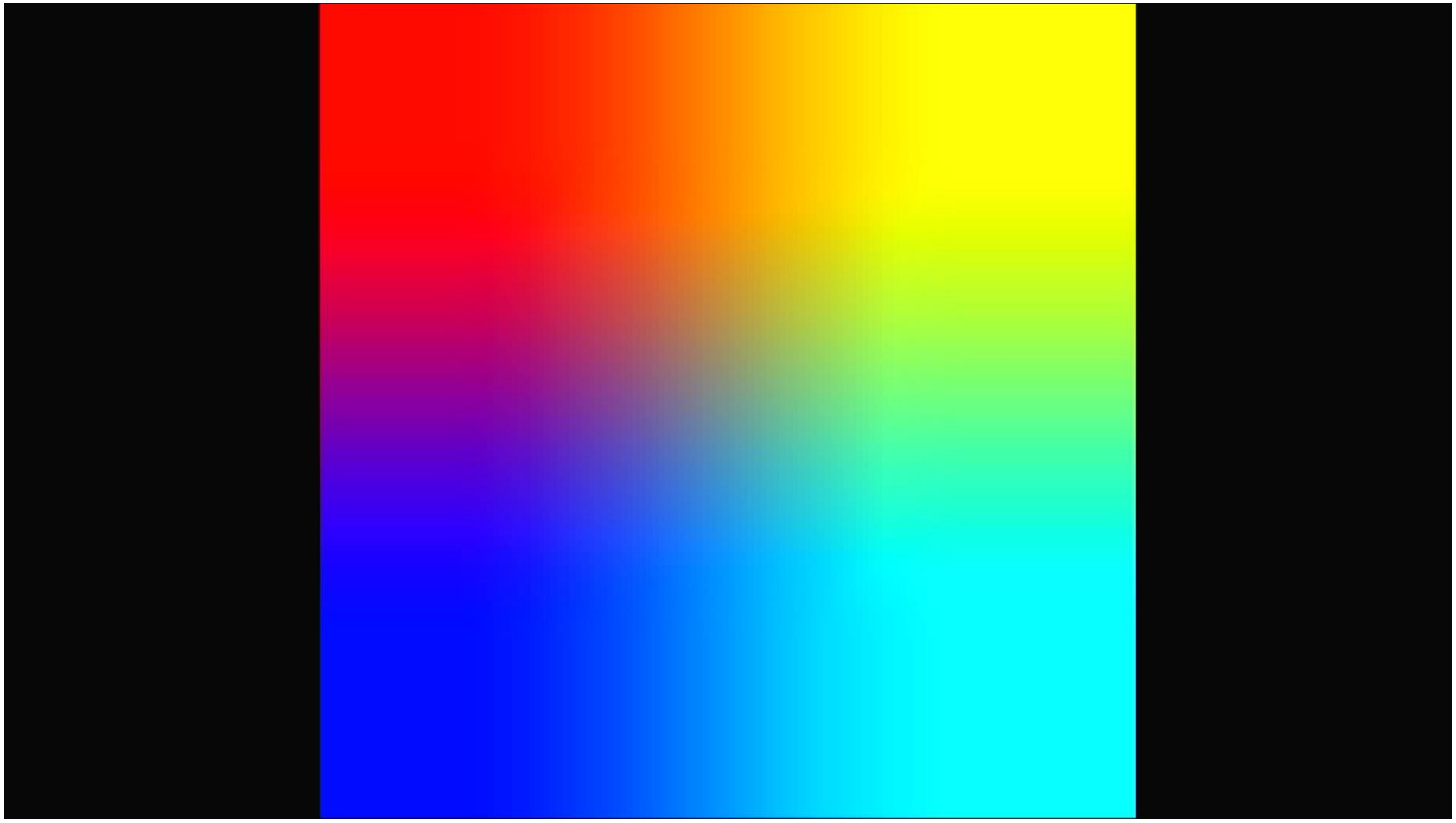
※まだ、コンセントには刺さない



モニタ

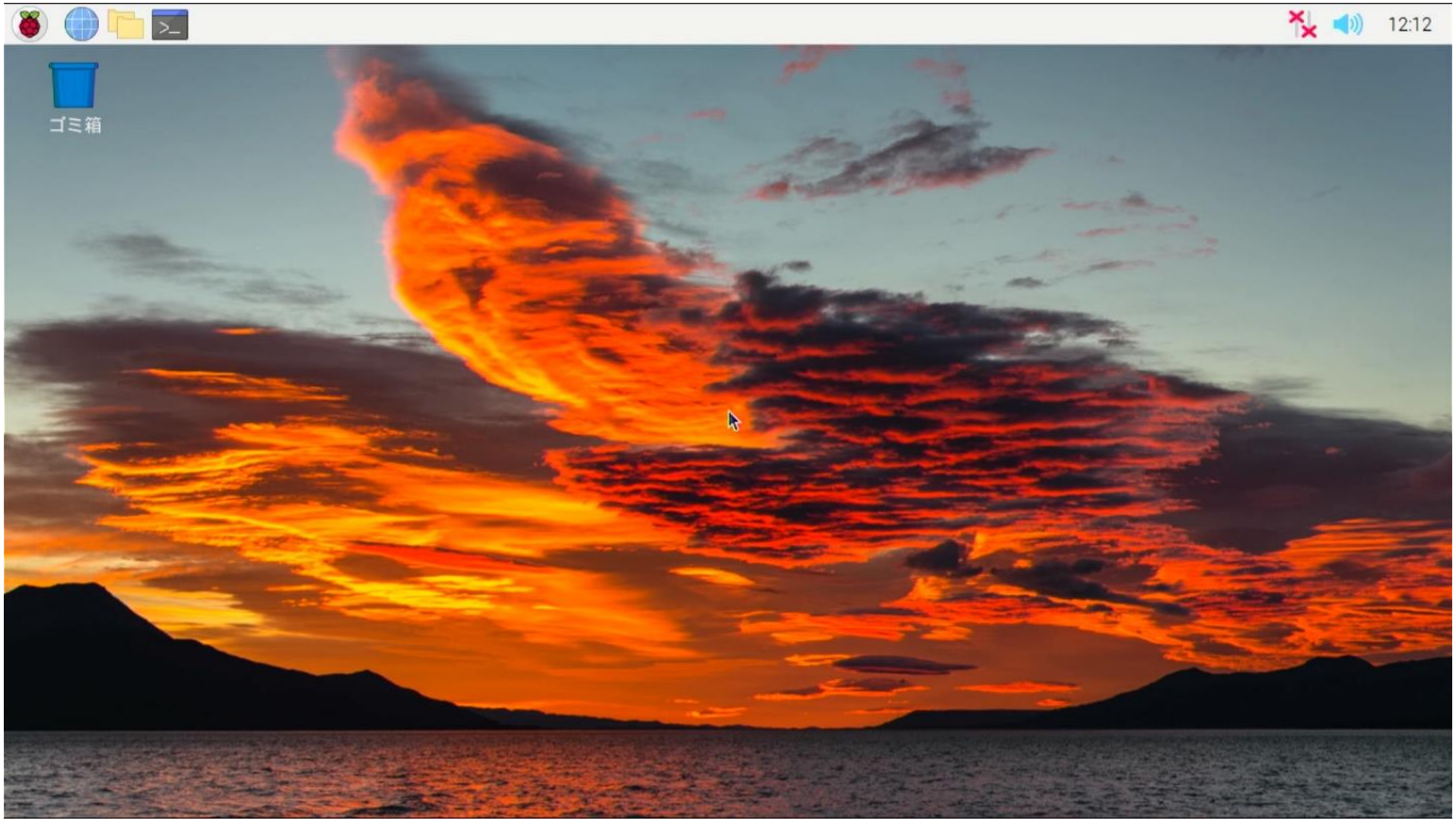
Pi OSの起動

- RPの電源を入れる(ACアダプタをコンセントに差込む)。電源ボタンはない
- 緑色のLED(アクセスランプ)が点滅し、下記の画面が出ると成功。



起動 (少し触ってみる)

- OSが立ち上がると、Microsoft Windowsと似た画面が現れる
  - しかしこれはMicrosoft Windowsとは似て異なるもので、X-Window システムと呼ばれるものである
  - 一通りメニューなどを試してみる



調査事項

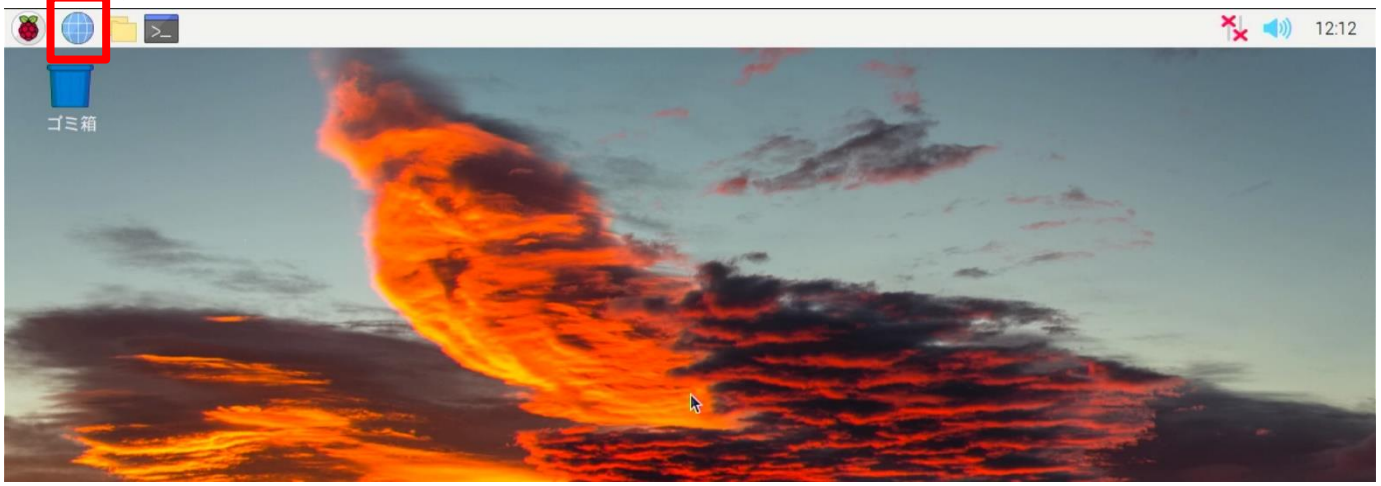
- OSのイメージとは何か
- 通常のPCでは、どのような手順でOSのインストールを行うか
- Pi OSとは、どのようなOSか
- X-Windowシステムは、どのような仕組みで動くのか
- Pi OSの起動画面の操作方法
- シャットダウンとは何か、またその正しい手法とそれをしなかった場合のリスクは何か
- エディタとは何か、どんな種類があるか
- Pythonというプログラミング言語について

ネットワークへの接続

- 接続できない場合
  - 右上のネットワーク インジケータをクリックした際にネットワークが選べるようになっているので、指定されたワイヤレスネットワークを選んで接続する
- ウェブブラウザを起動して、Google検索で長崎総合科学大学を検索し、ネットワークに接続できることを確認

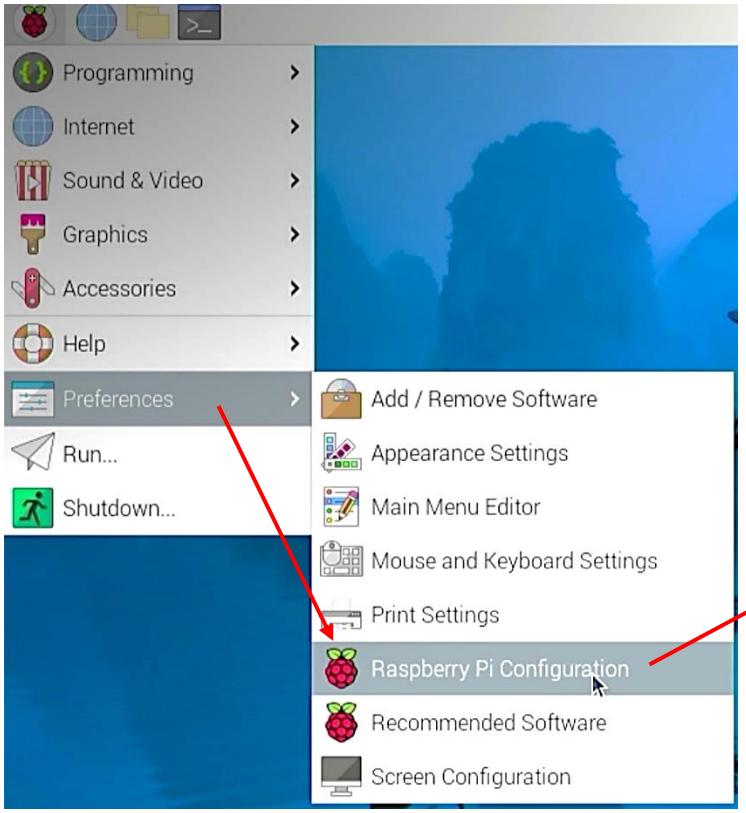
×になっていると、つながらない

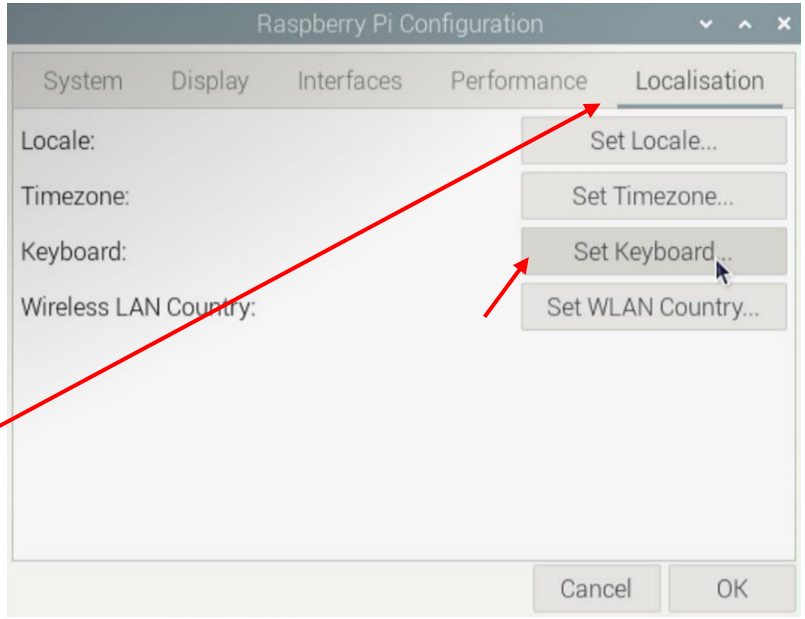
ブラウザ



キーボードの設定

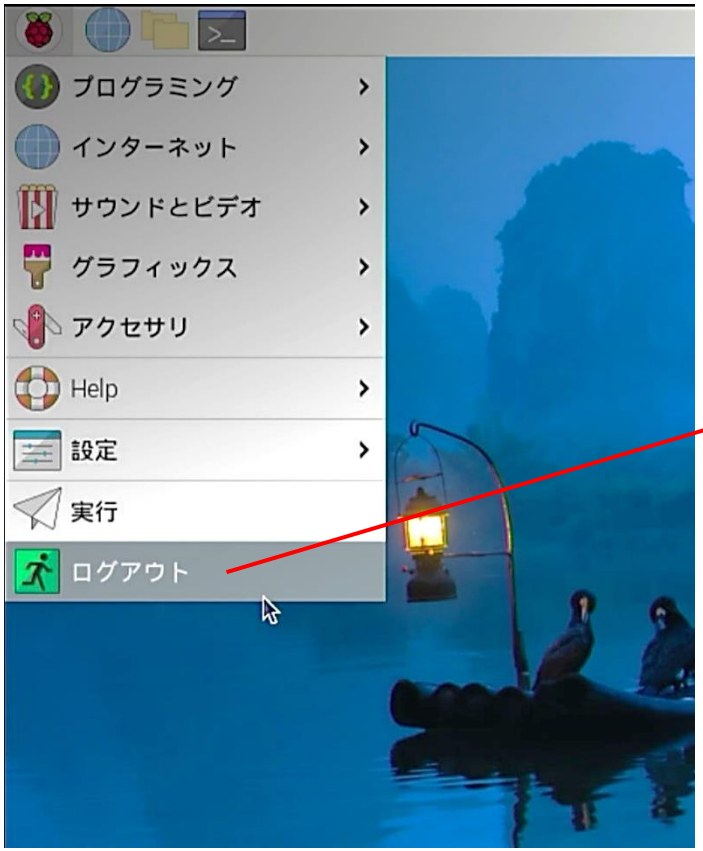
- ラズベリーパイの設定(Preferences)を選択
- キーボードとマウス(Keyboard and Mouse）を選択後、設定する
  - Model : Generic 105-key PC
  - Layout : Japanese
  - Variant : Japanese

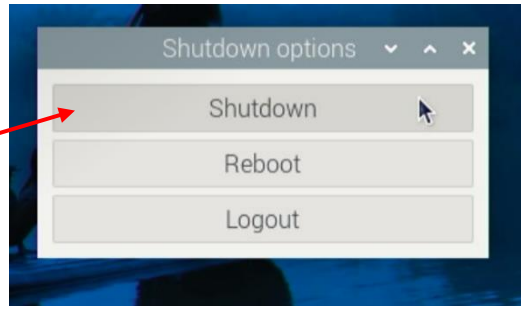




電源を落とす。再起動

- 再起動、電源の落とし方を確かめる。
- コマンドではそれぞれ、reboot、shutdownで行える。
- マウス操作では、  を押し、ログアウトから、選択する。





Linuxの操作

- Linuxにおいては、GUIもあるが基本的な操作はほとんどコマンドラインで行う
- 画面上側にあるアイコンからLXTerminalを開いて、使ってみる
- ウェブで調査をしながら、以下のコマンドについて習得すること
  - `cat, cd, cp, less, ls, mv, pwd, rm, cmp, df, du, mkdir, rmdir, ps, kill, top, sudo`
- 今後、プログラムを書いて機器を制御する方法を学ぶが、プログラムは通常”エディタ”を用いて書く
  - 以下のエディタのどれでもよいので、次回までに習得すること
    - nano, vi, emacs, leafpad (左上メニューからも選べるもの)
- RPには様々なプログラミングで”遊ぶ”ツールが入っている
  - scratch などで遊んでみることがお勧め

User とパーミッション

- LXTerminalにコマンドを打ち込む
  - 自分が誰であるかの確認コマンド: whoami
- root 権限での実行
  - sudo コマンド名
  - チェック: `sudo whoami`
  - どのように変化したかを確かめる。rootとは？
- 基本的に必要なソフトウェアは repository から 、superuser 権限でインストールする
  - 例: `sudo apt install emacs`
  - (RPインストール直後は、事前に sudo apt update と sudo apt upgrade が必要)

プログラミングの前に１

- LXTerminalを開いてコマンドを打ち込む
  - python3 –vと打ち込み返事がある（Python 3.X.X）のを確認する
  - 入っているかの確認です。

- 確認したら下記のどちらかでプログラミングを行う
  - メニューのプログラミングから、「Thonny Python IDE」を開く
    - 打ち込むことでプログラミングができる
  - もしくはnano, vi, emacs, leafpad (左上メニューからも選べるもの)を使用してプログラミングを行う。

プログラミングの前に２

- これまでみなさんが、講義で学んだ言語
  - C言語、C言語の派生(C++、C#)、Arduino IDE(C++ベースの言語)
- この講義で使用する言語
  - Python

- C言語とは、違う言語なので書き方が異なる。

- また、同じラズベリーパイ財団のマイコンボードとして
  - Raspberry Pi PICO、 Raspberry Pi PICO Wなど、
  - 言語：Micro python

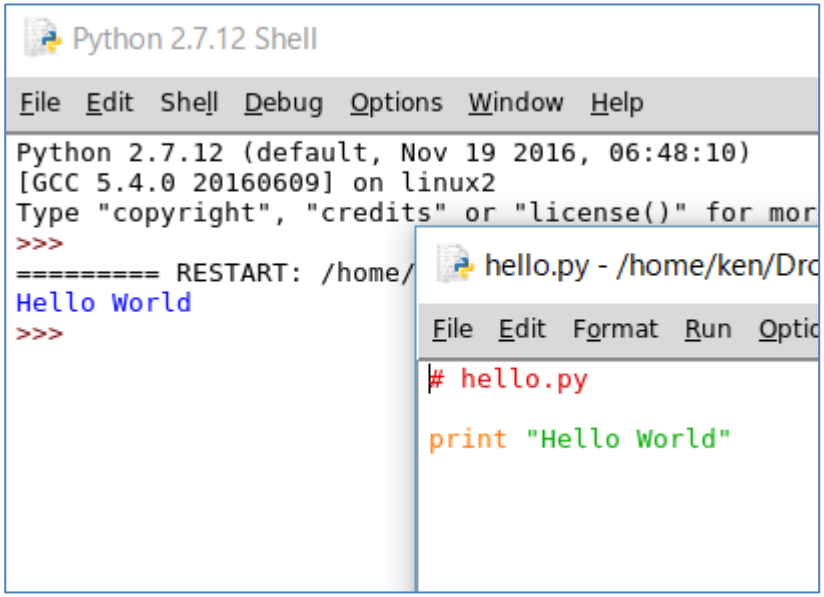
プログラムの作成

- 下のようなプログラムを書いて各自実行してみよう

＃はコメントアウト。C言語での // と同じく、コメント扱いされる

```
# hello.py

print ("Hello, world“)
```

- 手順1
  - エディタで上記を書いてセーブ (拡張子は .py)
  - シェルで次を実行
    - python セーブしたファイル名
- 手順2
  - Python shell である idle から読みこむ
  - idle 上で編集も可能



Python shell での実行例

調査事項(課題)

- home directory とは何か
- コンパイラとインタープリタの違いは何か、Python はどちらの言語か
- shell とは何か、どのような種類のものがあるか

より複雑なプログラムの開発

- 最初のゴール: LED を点滅させる
- LEDをいきなりつなぐ前に、いくつかのプログラムをしっかり理解しよう

```
# test1.py

npen = 10
hello = ("Hello, world“)

print (npen)
print (hello)
print ("I have", npen, "pens.“)
```

変数の使用例

- print()で()の中の変数が表示される。
- print(“”)で()の中の””で囲まれた文字列が表示される

- test1.pyを実行したときにどのように表示されたかを確認する
- 特にnpenとhelloをよく確認すること

Pythonでのプログラミング１

- ライブラリの読み込み
- この講義で使用するライブラリは下記の２つ。必要な場合は、調べて追加でインポートする。
  - import gpiozero　　→GPIO(入出力ピン)を使うのに必要
  - from time import sleep　→Sleepを使うのに必要

- C言語では#include <stdio.h>や#include “自作したヘッダ.h”のように読み込んでいく

```
#include <stdio.h>

処理1
```

C言語での例

```
import gpiozero
from time import sleep

処理1
```

Pythonでの例

- 読み込むことで、読み込んだライブラリの命令は使用できる。

Pythonでのプログラミング 2

- Pythonでは、インデントをそろえることで同じブロックとみなして処理を行う。そのため、同じ処理のブロックは位置を揃える。

ずれていても、
処理2は実行される

```
if(条件文1){
    処理1;
      処理2;
}else if(条件文2){
    処理3;
}else{
    処理4;
}

処理5;
```

C言語での例

そろえる必要がある

```
if 条件文1:
    処理1
    処理2
else if(条件文2):
    処理3
else:
    処理4


処理5
```

Pythonでの例

- C言語では{}で囲われている範囲が条件式の範囲。Pythonでは先頭をそろえることで1つの処理のブロックとして扱われる
- 例では条件の内容により処理が変わるが、処理5は条件文に含まれてないので、条件にかかわらず実行される

Pythonでのプログラミング3

- コメントアウトについて
  - C言語では「//」「/* */」でコメントアウトした
  - Pythonでは「#」「""" """」でコメントアウトする

```
//一行分コメントアウトする
//処理1;

/*
複数行コメントアウトする
処理2;
処理3;
*/
```

C言語での例

```
#一行分コメントアウトする
#処理1

"""
複数行コメントアウトする
処理2
処理3
"""
```

Pythonでの例

try, except, while

- try,except→try文で囲まれた処理を実行し、例外が起きたときにesceptに書かれた処理を実行する。
- 例2は、通常時はtryの処理を実行し、キーが押されたときに処理3の内容を実行する

```
try:
    処理1
    処理2

except:
    処理3
```

例1

```
try:
    処理1
    処理2

except KeyboardInterrupt:
    処理3
```

例2

- while→条件が満たされている間、処理を繰り返す

```
while True:
    処理1
    処理2

 処理3
```

例3

- 例3はwhileが正の間、実行され続ける。
- whileの処理を抜けたときのみ処理3が実行される

try, except, while

- 次の二つのプログラムの動作の違いを確認すること

```
# test2.py

from time import sleep

try:
    while True:
        print("Hello")
        sleep( 1.0 )
        print("World")
        sleep( 1.0 )

except KeyboardInterrupt:
    print "Interrupted, quitting."
    pass
```

```
# test3.py

from time import sleep

while True:
    print("Hello")
    sleep( 1.0 )
    print("World")
    sleep( 1.0 )

print "Interrupted, quitting."
```

- 何故、左のようなコーディングが必要なのか?
- また、**上記の通りに打つとエラーが出る。エラー文を確認し、エラーがなくなるように修正すること(python2とpython3のコードが混じっている)**

RP の GPIO

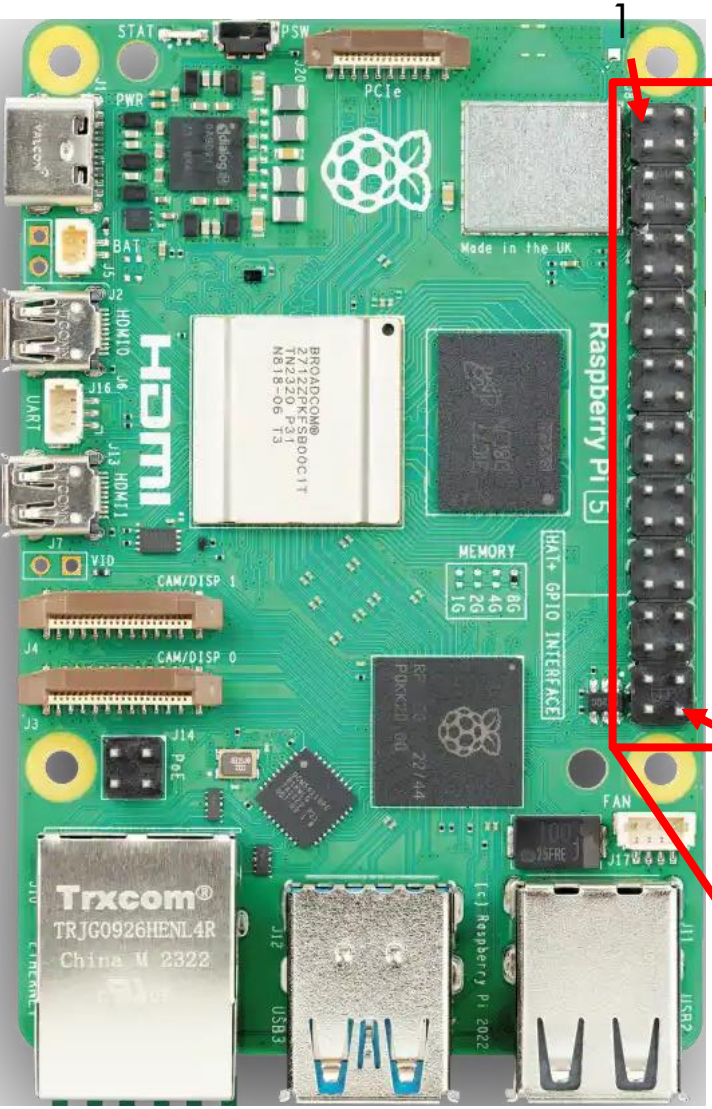
Raspberry Pi2 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

- ピンには役割がある
  - DC Power→電源
  - GPIO→入出力ポート
  - Ground→グランド
  - その他→特定の機能
- GPIO
  - 汎用入出力の略称
    general purpose
    input/output

ユーザーが制御できる入出力

RP の GPIO



Raspberry Pi2 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

GPIO等使用時の注意点

- 流せる電流に制限がある

ピン	最大電流
DC Power 5V [2,4] 合計	電源の最大電流 –900 mA
DC Power 3.3V [1,17]合計	100 mA程度
GPIO 1本あたり:	8 mA
GPIO 合計:	50 mA

- 上記の定格を守らない場合もしくは、ショートなどを引き起こした場合、破損する恐れがある

- GPIO はプログラムで入力(IN)・出力(OUT)を切り替え可能
  - 出力にセットした GPIOピン同士をつなぐと?
  - デフォルトは IN になっている、なぜ?

GPIOを使用する準備

- LXTerminalを開き下記のコマンドを入力していく
  - `sudo apt update`
  - `sudo apt full-upgrade`
  - `sudo apt-get install git-core`
  - `git clone https://github.com/Milliways2/GPIOreadall.git`
- sudo とは何か？
  - whoami コマンドを打ち込んでみる。どのような変化があるか？
    - `whoami`
    - `sudo whoami`
  - ls コマンドを打ち込んでみる。
    - cd と打ち込み、表示されたフォルダに移動する。

- cd はなにをしたのか？
  - ~ $ →~XXX $ に変わったが何があったのか。

GPIOの状態監視

- `cd GPIOreadall` ※最初の1回のみ実行。ディレクトリの移動。
- `python3 GPIOreadall.py` ※状態を知りたいタイミングに、実行する。

状態を監視

cd でディレクトリの移動

```
a@raspberrypi:~/GPIOreadall $ python3 gpioreadall.py
+-----+-----------+------+---+   Pi 5   +---+------+-----------+-----+
| BCM | Name      | Mode | V |  Board   | V | Mode | Name      | BCM |
+-----+-----------+------+---++---+---+---+------+-----------+-----+
|     |      3.3v |      |   |  1 || 2 |   |      | 5v        |     |
|   2 | GPIO2     |      |   |  3 || 4 |   |      | 5v        |     |
|   3 | GPIO3     |      |   |  5 || 6 |   |      | GND       |     |
|   4 | GPIO4     |      |   |  7 ||  8 |   |      | GPIO14    |  14 |
|     |       GND |      |   |  9 || 10 |   |      | GPIO15    |  15 |
|  17 | GPIO17    |      |   | 11 || 12 |   |      | GPIO18    |  18 |
|  27 | GPIO27    |      |   | 13 || 14 |   |      | GND       |     |
|  22 | GPIO22    |      |   | 15 || 16 |   |      | GPIO23    |  23 |
|     |      3.3v |      |   | 17 || 18 |   |      | GPIO24    |  24 |
|  10 | GPIO10    |      |   | 19 || 20 |   |      | GND       |     |
|   9 | GPIO9     |      |   | 21 || 22 |   |      | GPIO25    |  25 |
|  11 | GPIO11    |      |   | 23 || 24 |   |      | GPIO8     |   8 |
|     |       GND |      |   | 25 || 26 |   |      | GPIO7     |   7 |
|   0 | ID_SDA    | IN ^ | 1 | 27 || 28 | 1 | IN ^ | ID_SCL    |   1 |
|   5 | GPIO5     |      |   | 29 || 30 |   |      | GND       |     |
|   6 | GPIO6     |      |   | 31 || 32 |   |      | GPIO12    |  12 |
|  13 | GPIO13    |      |   | 33 || 34 |   |      | GND       |     |
|  19 | GPIO19    |      |   | 35 || 36 |   |      | GPIO16    |  16 |
|  26 | GPIO26    |      |   | 37 || 38 |   |      | GPIO20    |  20 |
|     |       GND |      |   | 39 || 40 |   |      | GPIO21    |  21 |
+-----+-----------+------+---+----++----+---+------+-----------+-----+
| BCM | Name      | Mode | V |  Board   | V | Mode | Name      | BCM |
+----------pinctrl----------+   Pi 5   +--------gpioreadall---------+
a@raspberrypi:~/GPIOreadall $
```

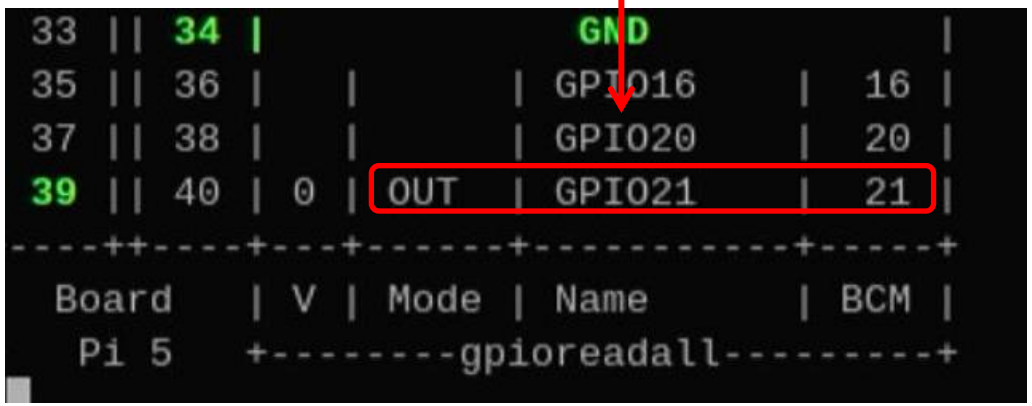
GPIOのモードを変えてみる 出力

- python3 GPIOreadall.py で動作を見ながら idle 上で、下のように実行すると...

※状態を知りたいタイミングに、実行する。

```
# gpiotest.py

from gpiozero import LED

led = LED(21)

#書きたいプログラムを書く
```

変化があらわれる
(コマンドを実行し、確認する)



import 文は、
おまじないのようなもの

出力レベルを変えてみる

- 例として、21ピンを変えてみる。

  led = LED(21)

  21番ピンにledと名前を付け、21番ピンを出力に設定

- 21番ピンをハイにする:

  led.on()

- 21番ピンをローにする:

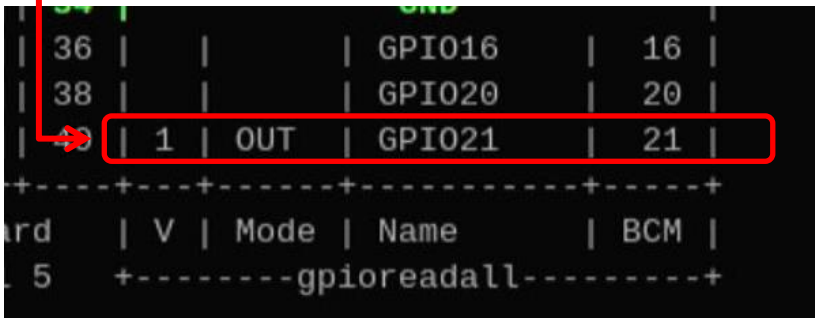
  led.off()

- 各自実行して確認すること

実行例

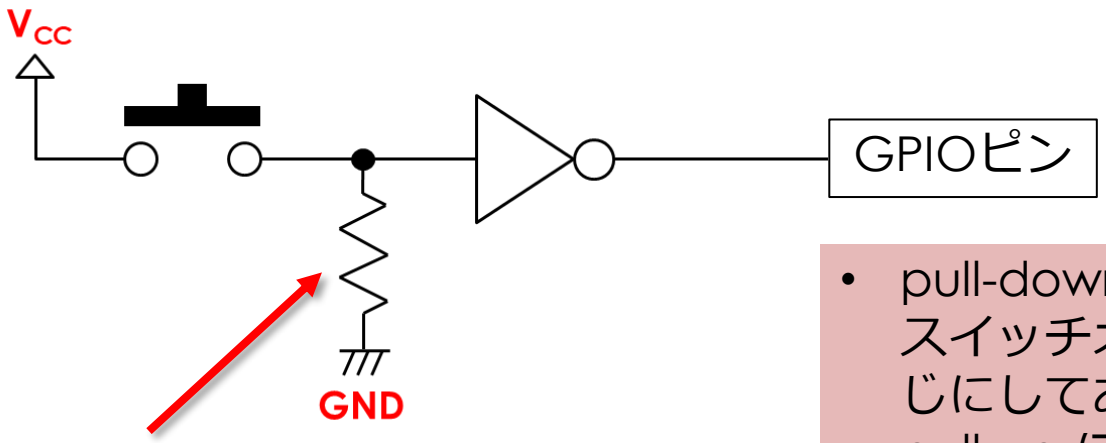
```
# gpiotest.py

from gpiozero import LED

led = LED(21)
led.on()
```

目的のチャンネルの出力が1になれば成功



入力を試す スイッチ入力の例1

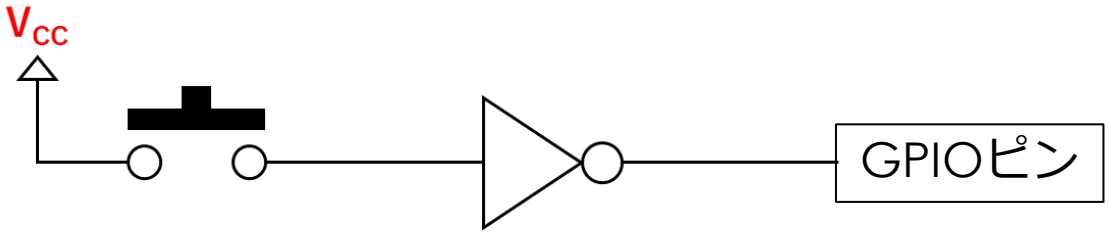
- 図のように回路を作成する。Vccは、3.3 Vにすること。



1～10 kΩ程度の抵抗でGNDに繋ぐ

- pull-down (プルダウン)抵抗で、スイッチオフ時はGNDと電位を同じにしてあげる
- pull-up (プルアップ)もある

- なぜ、下記のような回路はだめなのか(なぜ抵抗がいるのか)を復習する



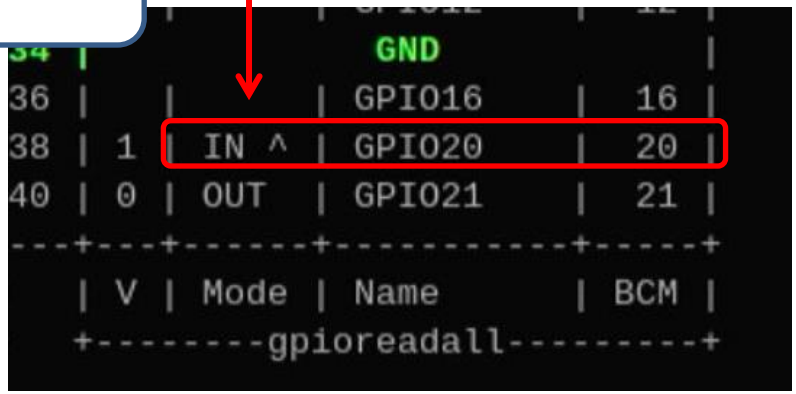
入力を試す スイッチ入力の例2

- python3 GPIOreadall.py で動作を見ながら idle 上で、下のように実行すると...

※状態を知りたいタイミングに、実行する。

```
# gpiotest.py

from zerogpio import Button

button = Button(20)

#書きたいプログラムを書く
```

変化があらわれる
(コマンドを実行し、確認する)



import 文は、
おまじないのようなもの

入力を試す スイッチ入力の例3

- 特定のピンの状態を読み込む(値は1,0でかえってくる)

```
button(ピン番号)
```

- ピンモード入力でプルアップとして使う

```
button(ピン番号, pull_up=True, pin_factory=factory)
```

- ピンモード入力でプルダウンとして使う

```
button(ピン番号, pull_down=True, pin_factory=factory)
```

→プルアップとプルダウンで入力が反転していることを確認すること

- 各自実行して確認すること

入力を試す スイッチ入力の例4

- ボタンを押す、離すの検出
  - 押したことを検知　※押されるまでループして、何もしない

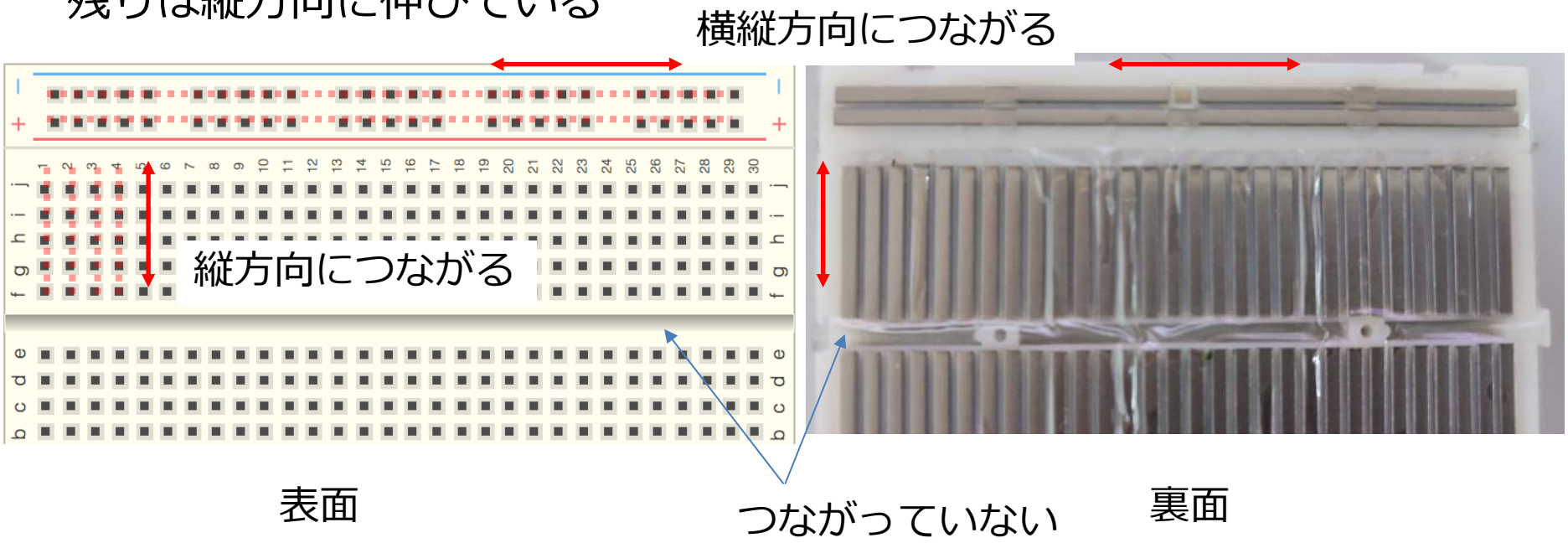
    ```
    button.wait_for_press()
    ```
  - 離したことを検知

    ```
    button.when_released()
    ```

- 各自実行して確認すること

課題1の準備

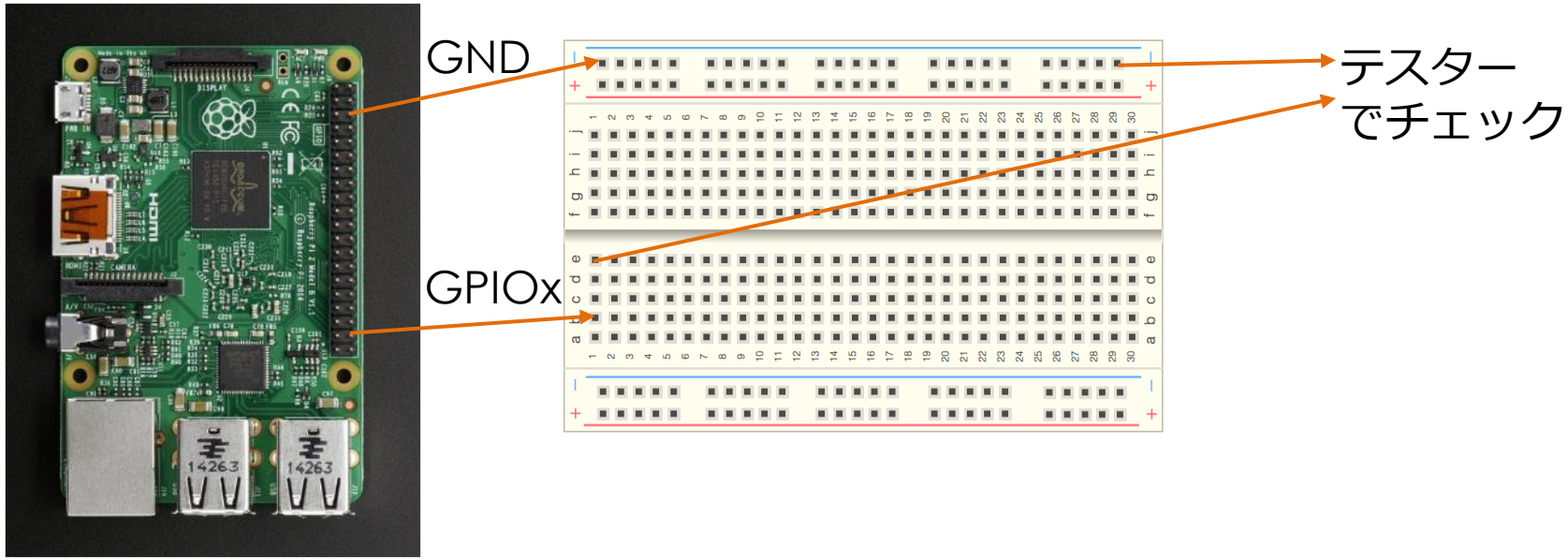
- 課題1ではブレッドボードを使用する
- ブレッドボードは裏面で写真のように＋とーの列は横方向に、
  残りは縦方向に伸びている



- 真ん中は、分離している

課題1

- これまでの実験の知識を活用して、以下を行う
- 特定のピンのレベルを変化させて電圧を確認する
  - テスターとブレッドボードを使用する
  - RP の電源を切った状態でブレッドボードに GND と必要なピンを引き出し、テスターの端子を接続
  - RPが動作中に回路の変更をしたりテスターの端子を動かしたりしないこと！！



gpiozeroに関する各種ドキュメント

- 下記のドキュメントを参照しながらプログラムをしていく

- ピン番号に関するドキュメント

https://gpiozero.readthedocs.io/en/stable/recipes.html#pin-numbering

- 入力に関するドキュメント

https://gpiozero.readthedocs.io/en/stable/api_input.html

- 出力に関するドキュメント

https://gpiozero.readthedocs.io/en/stable/api_output.html

課題2の準備

- 課題2でもブレッドボードを使用する
- ブレッドボードの使い方は、課題2を参照すること。
  - 抵抗値を計算する。
  - 条件は下記で設定する
    - ラズパイのGPIOは何Ｖか
    - LEDに1～10 mA程度流れる
    - 使用するLEDの色を決める
      - 赤色、緑色
      - 青色
      - 以上の色のV_fは？？？
- 計算をして、了解をもらってからLEDと抵抗を受け取る。

課題2

- HIGH 時の GPIO の出力電圧をもとに、LEDを点灯する回路を設計、製作する
  - LEDの電圧降下、動作(最大)電流、およびRP2 GPIOの最大電流に注意して設計すること
  - 制作開始は、回路図を提出し了解を得てから行うこと

- LEDを定期的に点滅させるプログラムを作成し、動作を確認すること

課題3の準備：関数

- 次のプログラムを作成し、動作させること。
- def, for などの使用方法を理解すること。簡単な解説は次のスライド

```
# func.py

from time import sleep


def printDataNtimes( data, n ):
    for i in range(n):
        print (data)

try:
    while True:
        printDataNtimes( "Hello", 2 )
        sleep( 1.0 )
        printDataNtimes( "World", 2 )
        sleep( 1.0 )

except KeyboardInterrupt:
    print ("Interrupted, quitting.“)
    pass
```

課題3の準備：関数の簡単な解説

```
# func.py

from time import sleep


def printDataNtimes( data, n ):
    for i in range(n):
        print (data)

try:
    while True:
        printDataNtimes( "Hello", 2 )
        sleep( 1.0 )
        printDataNtimes( "World", 2 )
        sleep( 1.0 )

except KeyboardInterrupt:
    print ("Interrupted, quitting.“)
    pass
```

- def→関数を定義している
  - Print(略)times→関数名
  - (data , n)→引数
- for文
  - for X in renge(Y):

  →変数XにＹまでの数字を繰り返し代入する
- 以上を踏まえて、どのように動いているかを実行結果と比較しながら、確認すること。

課題3の準備：素子を調べる

- 配布した7セグメントLEDについて調べる。(型番、データシート、使い方)

- データシート
  - 部品の仕様やスペック、使い方が載っている

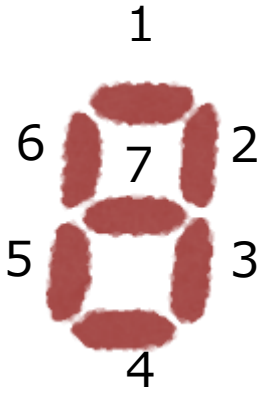
- 7セグメントLEDのデータシートを検索する
  - データシートは製造元のサイトか、販売元のサイトにある
  - Digi-key,RSコンポーネンツ, Mouserなど
  - (国内ショップ：秋月、マルツパーツ、共立、鈴商、若松、仙石)

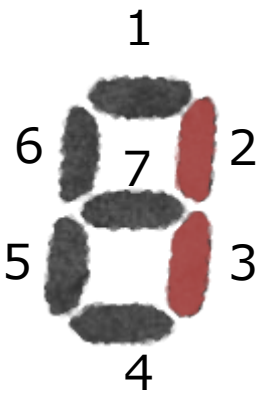
- アノードコモン、カソードコモンについて調べる
  - 調べたら、データシートから7セグメントLEDの仕様を確認する
  - ピン配置も確認する

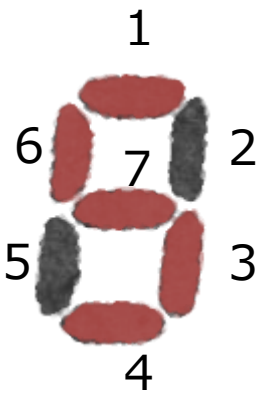
課題3の準備：7セグメントLEDを動かす

- カソードコモンorアノードコモンを意識して、7個のLEDを光らせる
  - 光らせ方は自由、点灯と消灯の動作をすること
  - 完成後、7セグメントLEDにつなぎ変えて、それぞれが光るのを確認

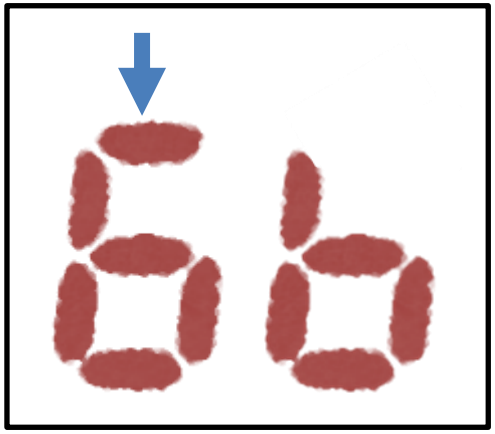
- 数字の0～9までを作り、光らせる







数字の例(６，９)



上につけるかつけないかは、好みの方に合わせる

  - 表現の例として、
    - 数字の8は1～7番がすべて光る
    - 数字の1は2、3が光る
    - 数字の5は、1、3、4、6、7が光る

➡GPIOピン番号に置き換えると…

課題3：関数を作成し、LEDを光らせる

- 以下の仕様の関数を作成せよ
  - 関数名: setLed(n)
  - 入力: n (0-9の整数)
  - 動作7セグメントLEDを与えられた数字のパターンで光らせる
    - 準備2で作ったパターンを光らせる
  - まず回路図を設計し、それに合わせてGPIOを制御する関数を記述

- RPのGPIOにスイッチなどの情報を入力するための回路とプログラムを完成し、動作を確認せよ
  - Outputモードではなく、Inputモードとして使用する方法を各自調査し、実装する→ラズベリーパイ側からLEDを動かすこと
  - なぜ電源投入時のデフォルトは Input モード?

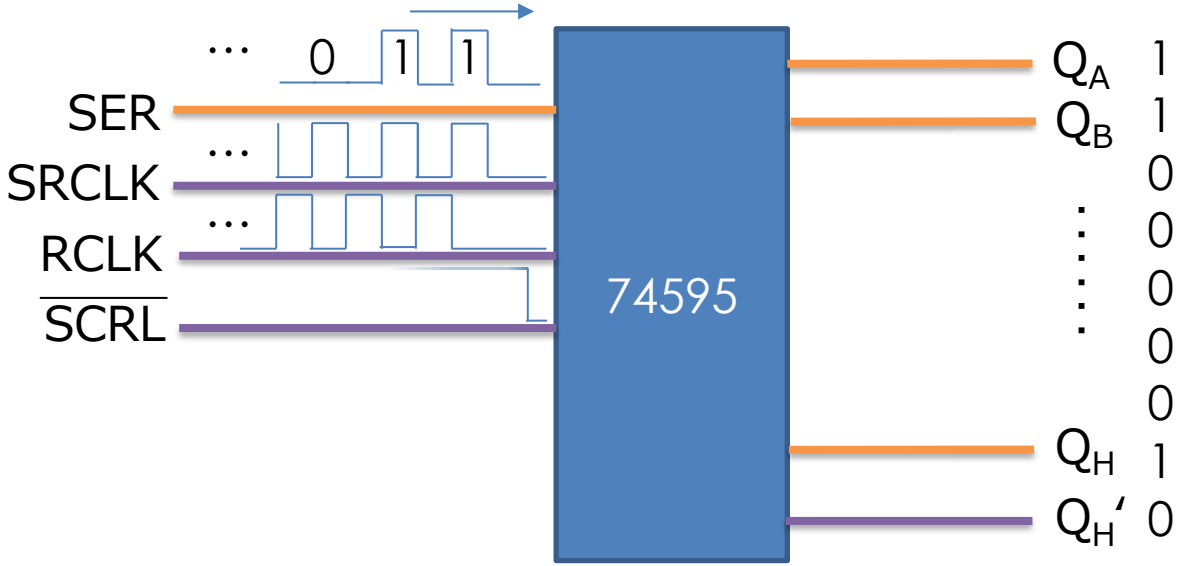
課題4の準備(1)

- 74HC595を用いて、7セグメントを駆動する
  - 74595は、シフトレジスタIC
    - シリアル信号(例として11000101)で送ると、74595のそれぞれのピンから

      Q_A:1、Q_B:1、Q_C:0、Q_D:0、Q_E:0、Q_F:1、Q_G:0、Q_H:1

  - 下記の図は、動作のイメージ

    10100011を送る



課題4の準備(２)

- 74HC595を用いて、7セグメントLEDを駆動する
  - 最初は、1つ駆動を目指し、作成する。
  - Dフリップフロップの復習。
  - 74595はシリアル→パラレル変換IC(シフトレジスタIC)
    - タイミングや使い方はデータシートから読み取る
      - ピンの位置　　→データシートの3ページ目
      - タイミングチャート →データシートの7ページ目
      - 真理値表　　→データシートの12ページ目
- 自分なりに考えて作る
  - 書籍やネットのものを参考にした場合は、そのプログラムについて説明ができ、参考サイトを見ずに配線を行う(完全に理解しておく)
- 来週は作るだけの状態にしておく

課題4

- 74HC595を用いて、7セグメント**LED2個を同時駆動**するための**回路と関数**を設計せよ
  - 参考資料(データシート)
    リンク(データシート)
  - SPI通信の関数は使用禁止

    ※自力で書くこと

データシートQRコード

- 設計した回路を実際に作成・動作させ、その動作状況をまとめた動画を作成せよ
- 個人情報は入れないように注意すること

→リンクを知っている人のみの限定公開モードにするとよいかも？

- 回路図、接続図などを入れ、どのような回路を作成したかをテロップまたは音声によって解説すること
- プログラムの動作原理を解説すること

課題4の動画投稿

- 個人情報は入れないように注意する
  - リンクを知っている人のみの公開モードにすると良いかも？

レポート

- 作成した動画を Youtube などにアップロードすること

- アップロードしたら、 その旨および、動画へのリンクを、メールで報告する
  - メールタイトルは次のようにすること:
    電気電子工学実験I 組み込み基礎 レポート ●班（氏名、氏名、、）
  - 班の全員に CC し、各人の担当箇所を簡潔に記述すること

- 提出先メールアドレスと締切日はクラスルームで告知します。
- 締切日は厳守すること

  →教務によって成績登録が締め切られので確定します

付録

- 過去の資料(参考用)

Raspberry Pi4までの設定とGPIO

※RP4までと、RP5からGPIOの仕様が変わりました。

Raspberry Pi 2

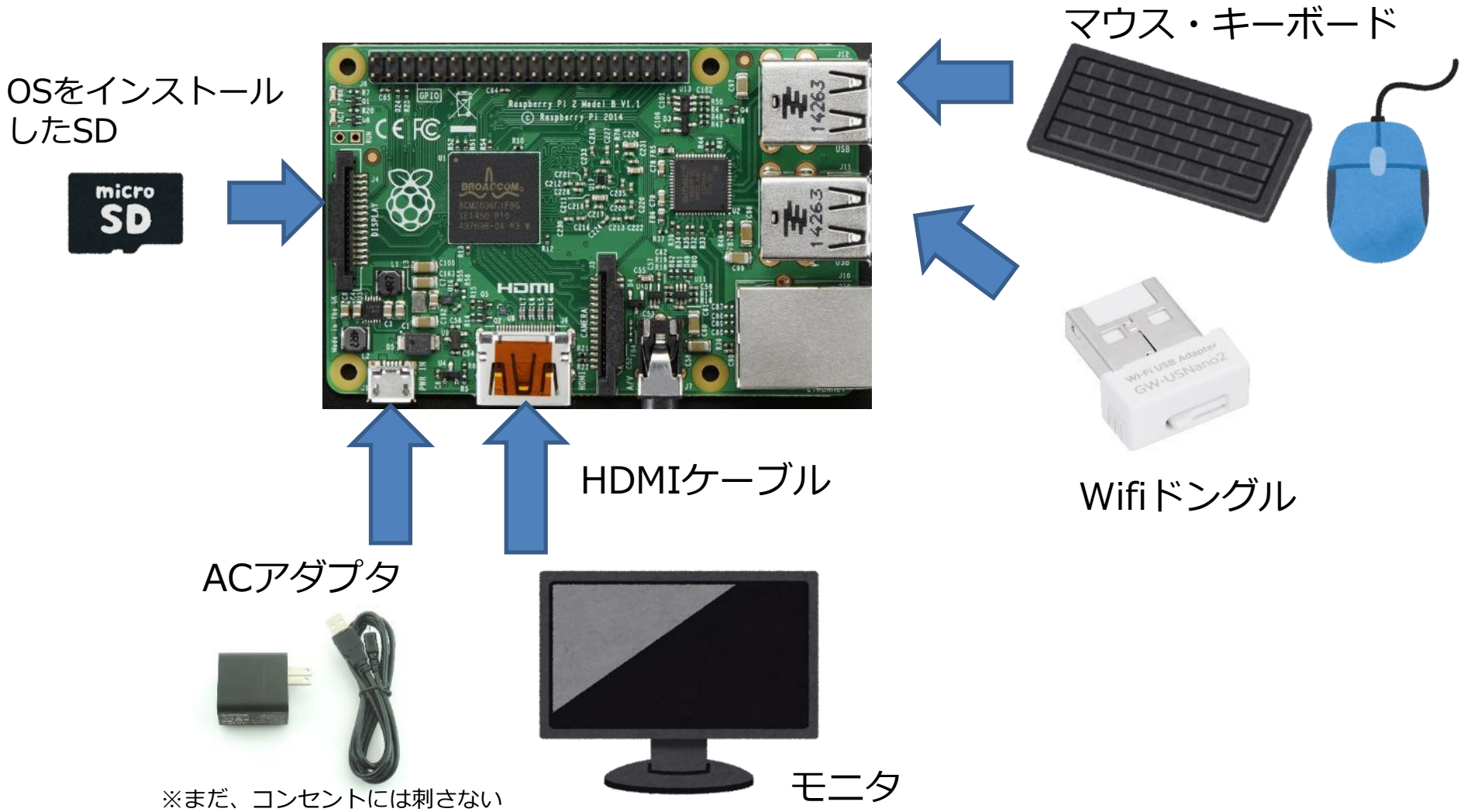
- Raspberry Pi (以下RP)は、イギリスのRaspberry Pi Foundationが2012年に開発したプログラミング学習・組み込み機器用超小型コンピュータ
- 例えばRP2ではCPUにQuad-core ARM Cortex-A7、1 GBのメモリ、EthernetおよびUSB2.0ポートを備え、LinuxがなどのOperating Systemを走らせることが可能
- ハードディスクは備えていないが、代わりにMicro SDカードスロットを持つので、これにOSをインストールして使用
- 40 pinのGPIOポートを備え、これを用いて様々な機器の制御実験等を行うことが可能

Raspberry Pi2の外観

「https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/」より画像の引用

起動の準備(RP2)

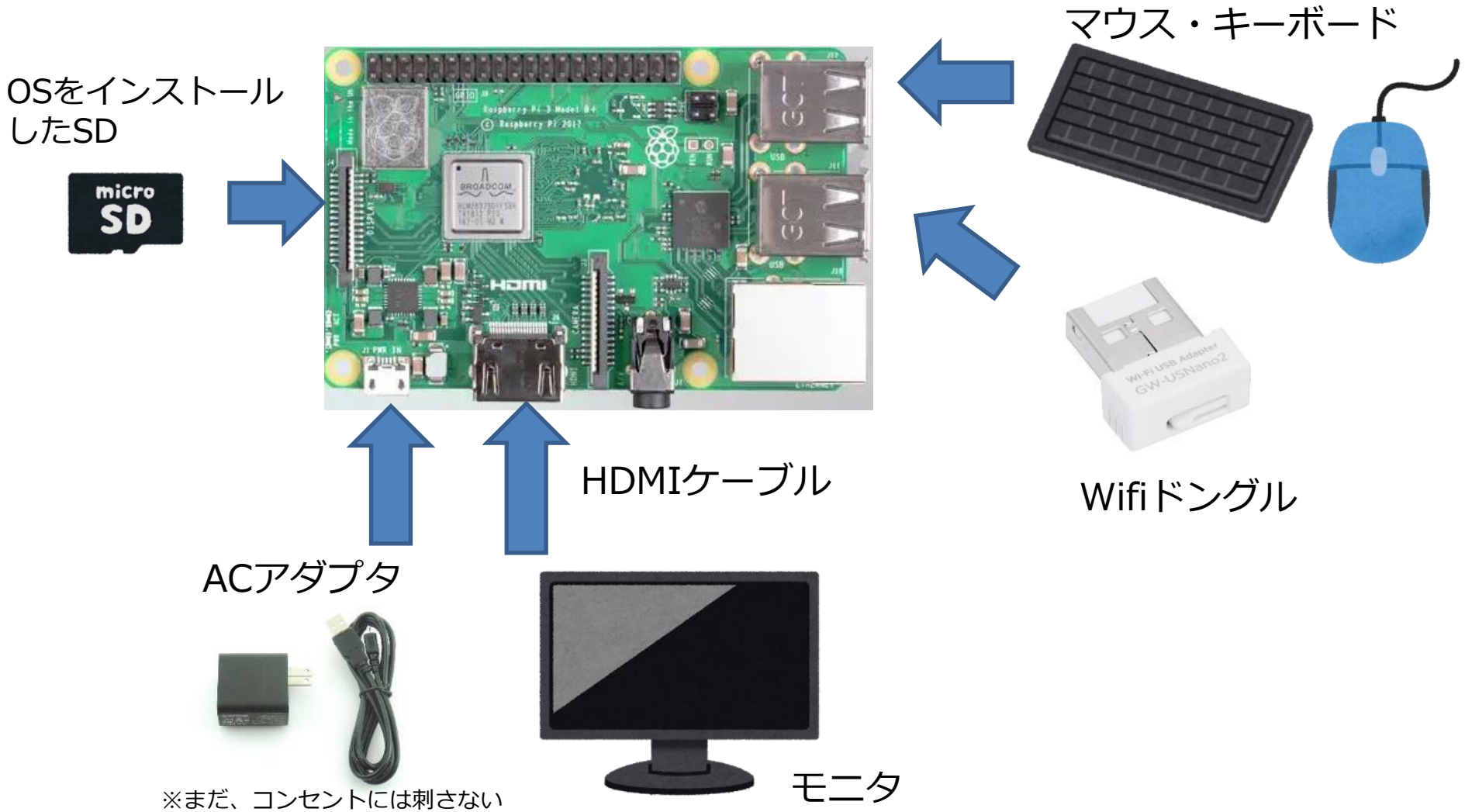
- RPにマウス、キーボード、モニタ、SDカードを接続



※RP3以後は、Wifiモジュールが内蔵されたため、不要

起動の準備(RP3以後)

- RPにマウス、キーボード、モニタ、SDカードを接続



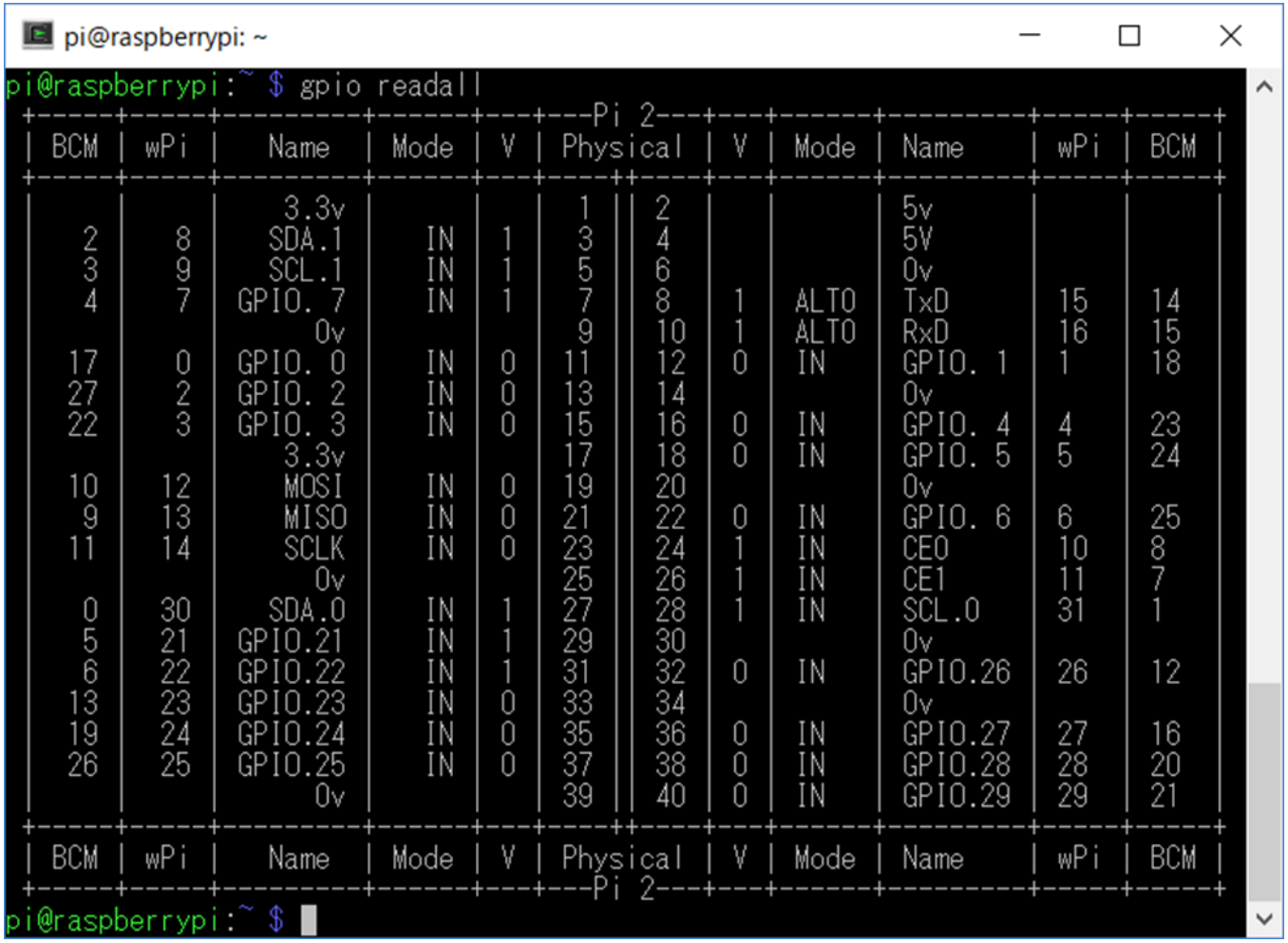
過去の資料(参考用)：Raspberry Pi4まで GPIOを使用する準備

- LXTerminalを開き下記のコマンドを入力していく
  - `sudo apt-get install libi2c-dev`
  - `sudo apt-get install git-core`
  - `git clone git://git.drogon.net/wiringPi`
  - `cd wiringPi`
  - `./build`

- sudo とは何か？
  - whoami コマンドを打ち込んでみる。どのような変化があるか？
    - `whoami`
    - `sudo whoami`
- cd はなにをしたのか？
  - ~ $ →~xxx $ に変わったが何があったのか

過去の資料(参考用)：Raspberry Pi4まで GPIOの状態監視

- `watch -n 1 gpio readall` でチェック可能 watch で監視しておくと便利
- ナンバリングに注意！ 前にあった図はここの BCM ナンバリングに相当

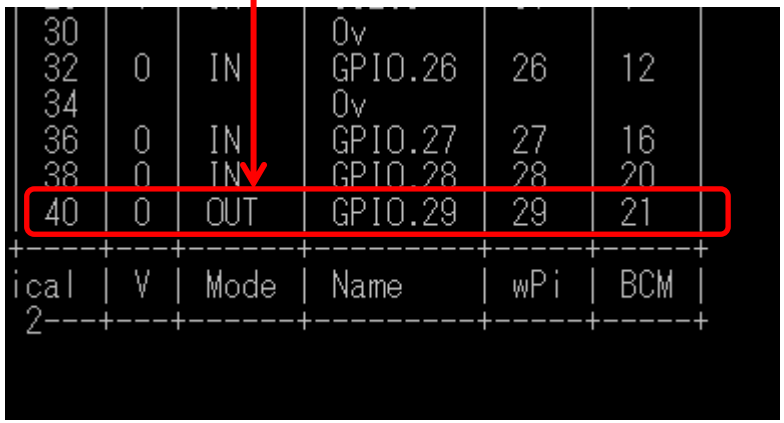


過去の資料(参考用)：Raspberry Pi4まで GPIOのモードを変えてみる 出力

- `watch -n 1 gpio readall` で動作を見ながら idle 上で、下のように実行すると...

```
# gpiotest.py

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(21,GPIO.OUT)

#書きたいプログラムを書く

GPIO.cleanup()
```

この瞬間に変化があらわれる



この cleanup ですべて元に戻る

import 文と GPIO.setmode の読出しは、おまじないのようなもの

過去の資料(参考用)：Raspberry Pi4まで 入力を試す スイッチ入力の例

- watch –n 1 gpio readall で動作を見ながら idle 上で、下のように実行すると...

```
# gpiotest.py

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.IN)

#書きたいプログラムを書く

GPIO.cleanup()
```

出力(OUTPUT)と同じように設定する

このcloseですべて元に戻る

import 文と GPIO.setmode の読出しは、おまじないのようなもの

過去の資料(参考用)：Raspberry Pi4まで 入力を試す スイッチ入力の例3

- ピンモード入力でプルアップとして使う

  GPIO.setup(ピン番号,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

- ピンモード入力でプルダウンとして使う

  GPIO.setup(ピン番号, GPIO.IN, pull_up_down=GPIO.PUD_UP)

→プルアップとプルダウンで入力が反転していることを確認すること

- 特定のピンの状態を読み込む(値は1,0でかえってくる)

  GPIO.input(ピン番号)

- 各自実行して確認すること