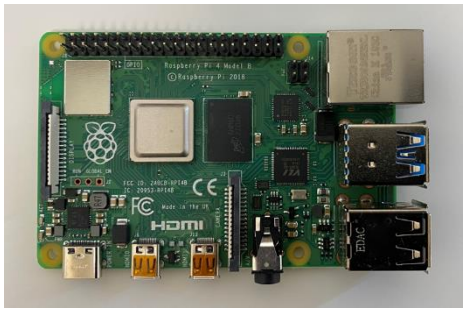
14. 組み込み（１）

Raspberry Pi 4 Model B セットについて

ラズベリーパイは、シングルボートコンピュータの一種で、元々は教育用に開発されていました。最近では、IoT に代表されるエッジコンピューティングのデバイスとして注目されています。ラズベリーパイの OS は、Linux（Debian）ベースの OS である Paspberry Pi OS（以前は Raspbian と呼ばれていた）が採用されています。
ラズベリーパイにはスターターセットがあり、価格帯は 1 万円～2 万円になります。モニタなどの周辺機器は別売りです。Raspberry Pi 4 は、2019 年 9 月 4 日に電波法認証を取得済みです。ラズベリーパイは前述したようにシングルボードコンピュータになるので、基本的には通常の PC を Linux 系 OS で動かすのと同じ操作方法になります。

セット内容の一例（手元のセットと異なる場合もあります）

ラズベリーパイ 4B（4GB RAM）

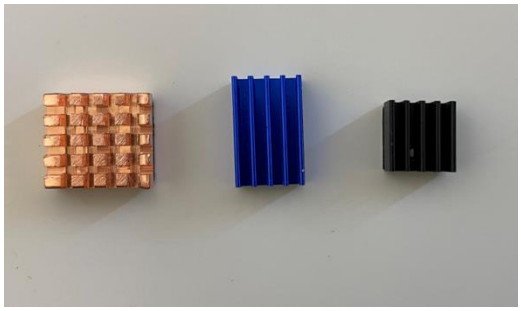


ラズベリーパイ 4B 用ケース

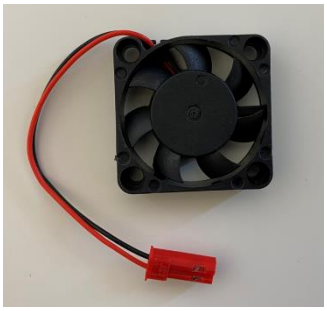
microSD カード（32GB）

カードリーダー

放熱対策のためのヒートシンク



ファン

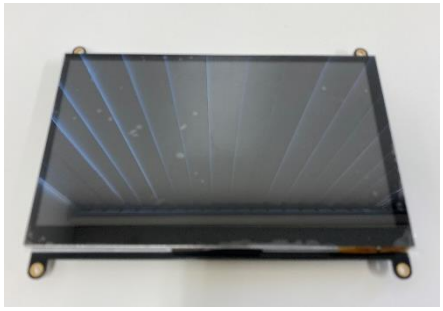


5.1V3.0A USB Type-C 電源アダプタ
（スイッチ付き電源ケーブル）

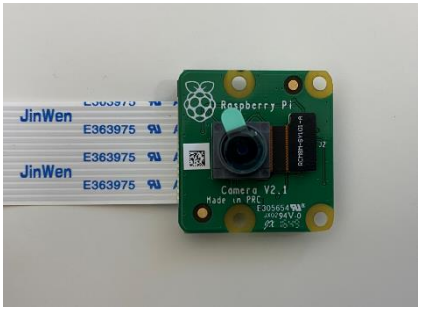
MicroHDMI-to-HDMI ケーブル

その他（LAN ケーブル、プラスドライバなど）

セット以外
モニタ（HDMI）



Camera Module V2

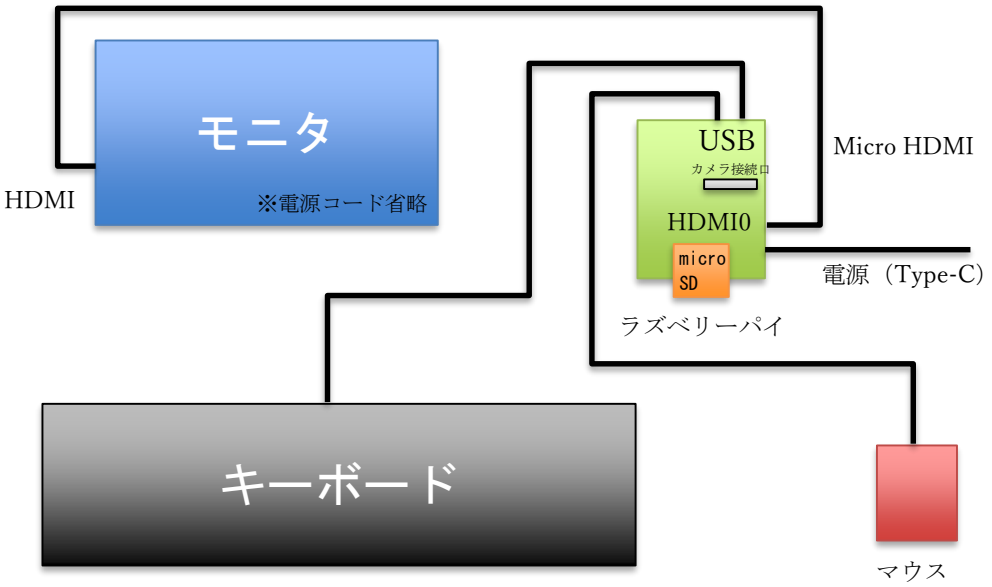


その他（キーボード、マウスなど）

組み立て

組み立て方の詳細は各セットのマニュアルや公式 Web サイトを参考にしましょう。

配線（HDMI 接続の場合）



・ヒートシンクを取り付けましょう。ヒートシンクが個別に分かれたタイプと、大きな 1 つにまとまったタイプなどがあります。

・モニタは HDMI で接続するタイプと、ディスプレイリボンケーブルで接続するタイプなどがあります。

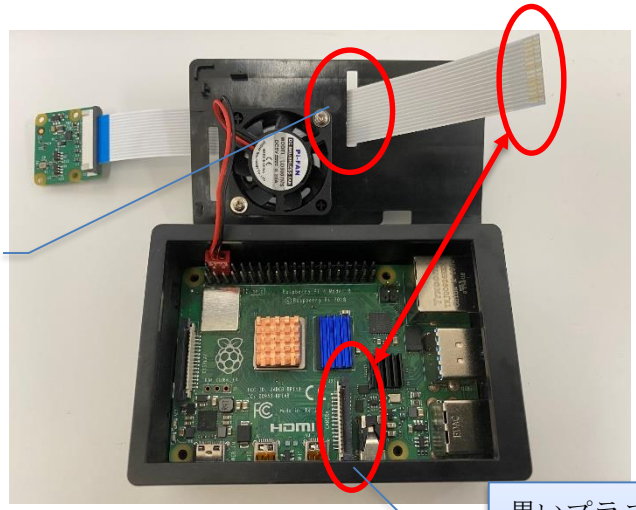
・モニタの電源供給にも、電源コードから供給するタイプと、ラズベリーパイ本体から供給するタイプなどがあります。

・ファンがあるセットとないセットがあります。

外観

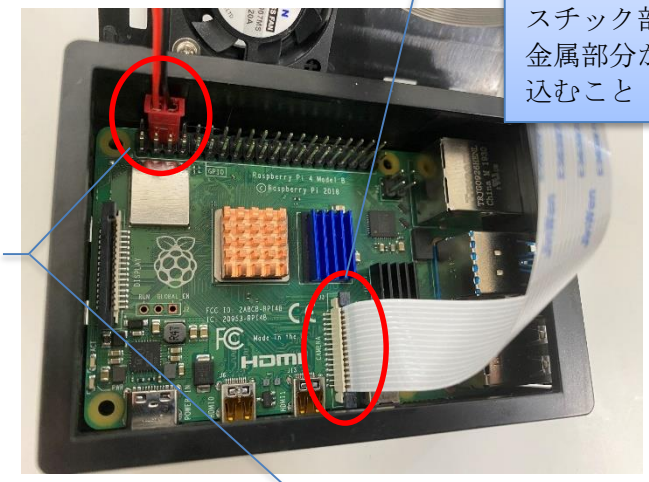
組み立て時には、Micro HDMI, 電源 5V 3A Type-C などのケーブルの種類に注意しましょう。また、映像出力は HDMI0 側（電源側のコネクタ）に接続しましょう。

カメラモジュールの接続

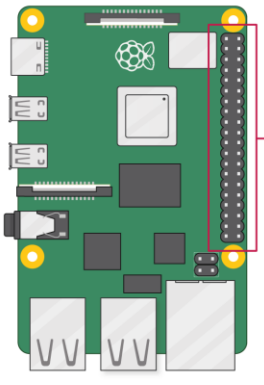


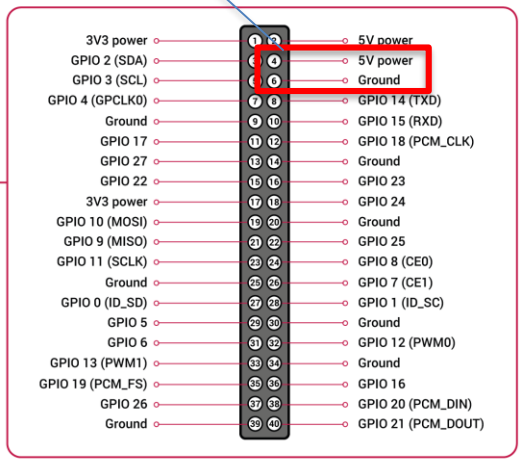
ケースを使う場合はカメラのコードを先に通しておかないといけないこともある

黒いプラスチック部分を上にスライドさせて開いた後にカメラの接続先を差し込み、黒いプラスチック部分を閉じる
金属部分が接触する向きで差し込むこと



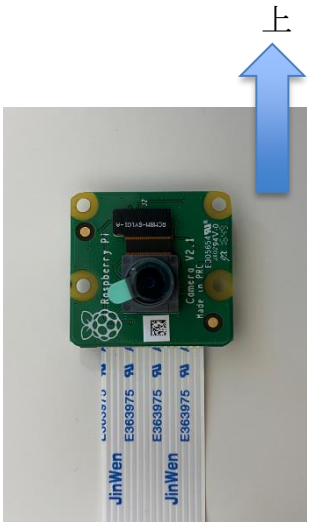
5V と Ground のピンにファンの電源コードを差し込む（赤－5V、黒－Ground）



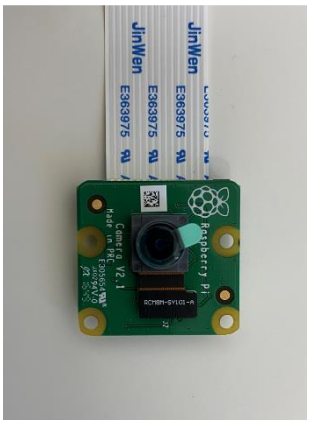


Raspberry Pi Foundation
https://www.raspberrypi.org/documentation/usage/gpio/

カメラモジュールは下の図の状態で向きが上下左右となります。



下のように利用する場合はプログラム内で画像を反転させましょう。



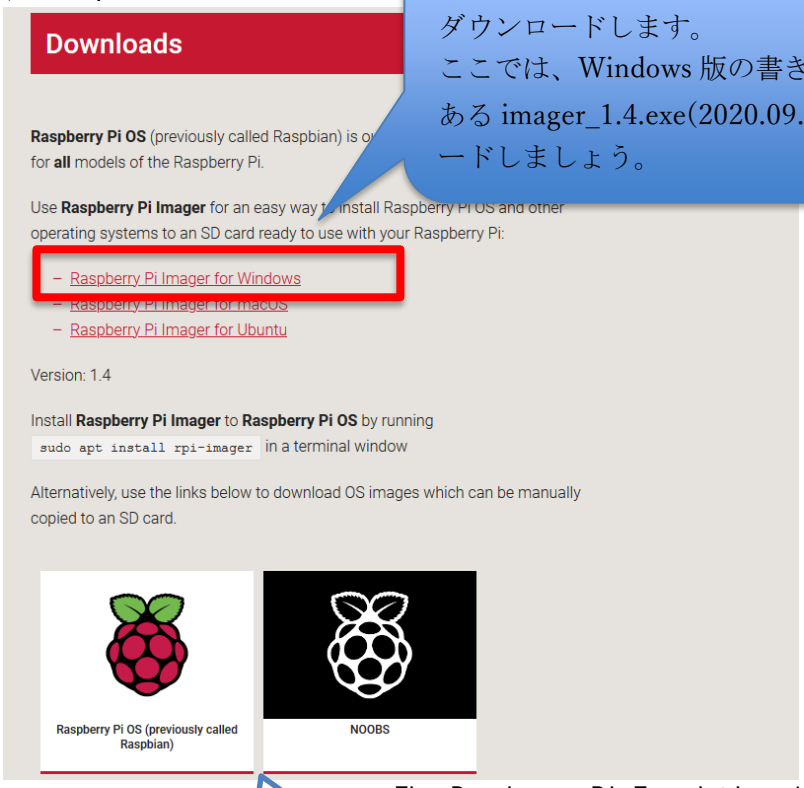
Raspberry Pi OS のインストール

ラズベリーパイで利用する OS（Raspberry Pi OS）を microSD カードに書き込んで準備しましょう。microSD カードへの書き込みにはラズベリーパイとは別で PC を利用します。Raspberry Pi OS の準備方法には、①専用の書き込みツール（Raspberry Pi Imager）を利用する、②一般の書き込みソフトを使ってイメージファイルを SD カードに書き込む、③ネットワークインストーラ（NOOBS）を microSD カードに書き込んでラズベリーパイの起動時にインストールする、④インストール済みの microSD カードを利用する、などがあります。ここでは The Raspberry Pi Foundation が提供している①のイメージファイルを使って書き込む方法を実施していきます。

まず、以下のサイトよりイメージファイルと microSD カードへイメージファイルを書き込むツール（Raspberry Pi Imager）の 2 つをダウンロードします。

https://www.raspberrypi.org/downloads/

ツールのダウンロード

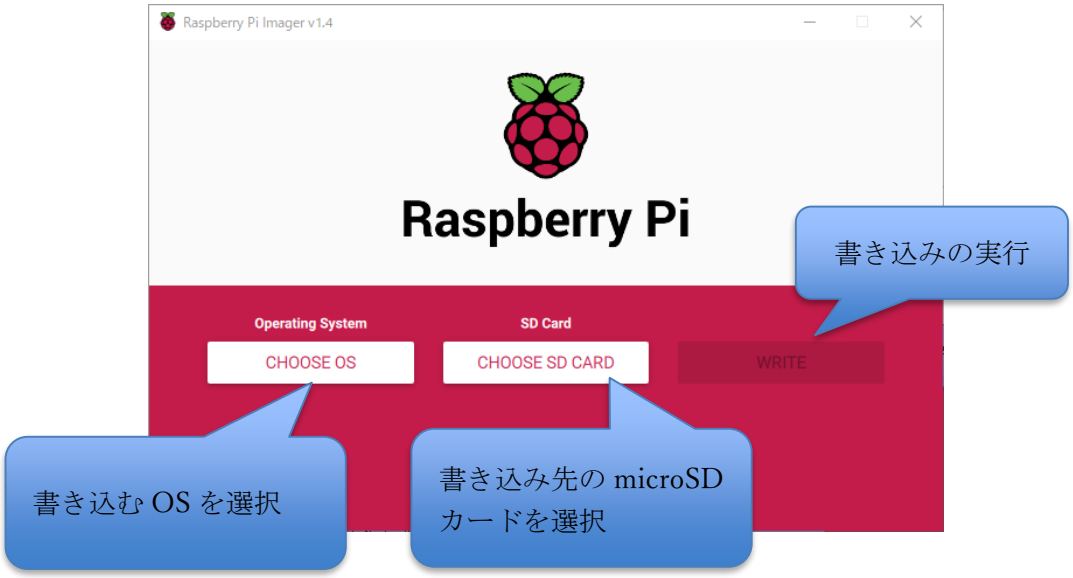


The Raspberry Pi Foundation（2020.09.02）

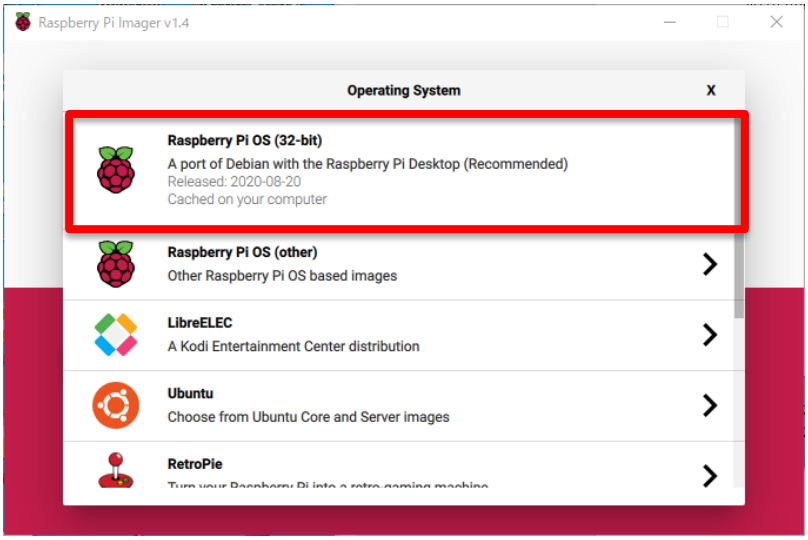
自身の PC 環境に合わせて書き込みツールをダウンロードします。
ここでは、Windows 版の書き込みツールである imager_1.4.exe(2020.09.02)をダウンロードしましょう。

今回は実施しませんが、②③で利用するイメージファイルはここから入手できます。

ダウンロードした imager_1.4.exe を使って書き込みツールを PC にインストールすると、次の図のようなツール（Raspberry Pi Imager）が利用できるようになります。PC に microSD カード差し込んでから、書き込むイメージファイルと差し込んだ microSD カードを選択して WRITE で microSD カードに OS のイメージファイルを書き込みましょう。なお、必要に応じて microSD カードリーダーを利用してください。

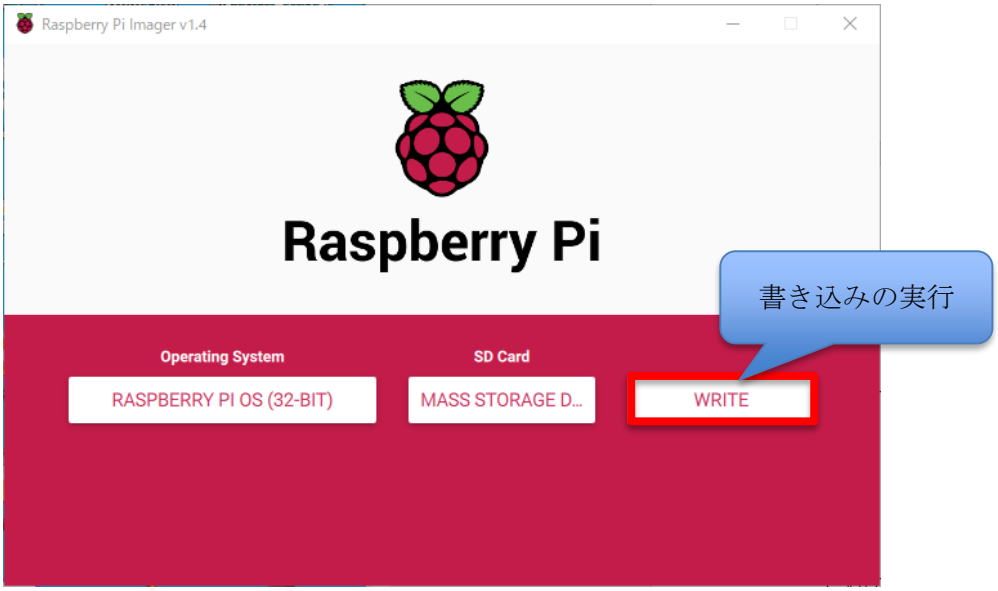


書き込む OS として Raspberry Pi OS (32bit)を選択しましょう。
※2020 年 4 月に 64bit 版の OS がリリースされていますが、ここでは 32bit 版を利用します。

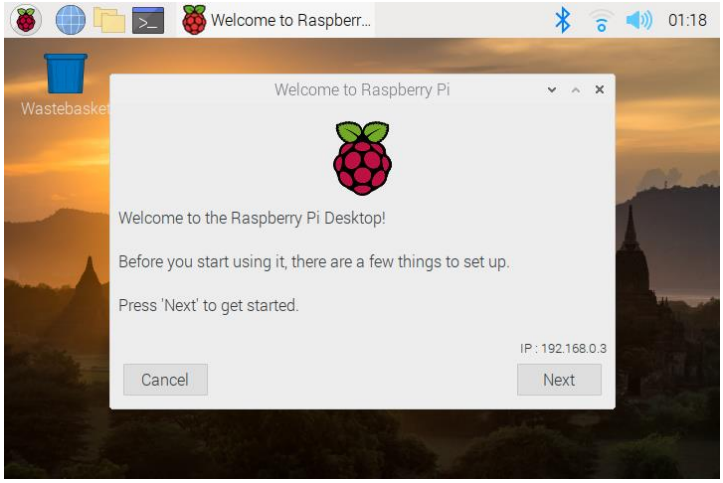


書き込む Raspberry Pi OS の種類に合わせて次のように 4GB・8GB・16GB 以上の SD カードの容量が推奨されています。
・Raspberry Pi OS (32-bit) with desktop and recommended software (microSD カード 16GB 以上推奨)
・Raspberry Pi OS (32-bit) with desktop (microSD カード 8GB 以上推奨)
・Raspberry Pi OS (32-bit) Lite (microSD カード 4GB 以上推奨)
※64GB 以上の microSD カードを使う場合はパーティションを 1 つにして FAT 形式でフォーマットする必要があります。

OS と microSD カードのドライブを選択したら WRITE をクリックして書き込みを開始してください. 数十分で書き込みが終了します。



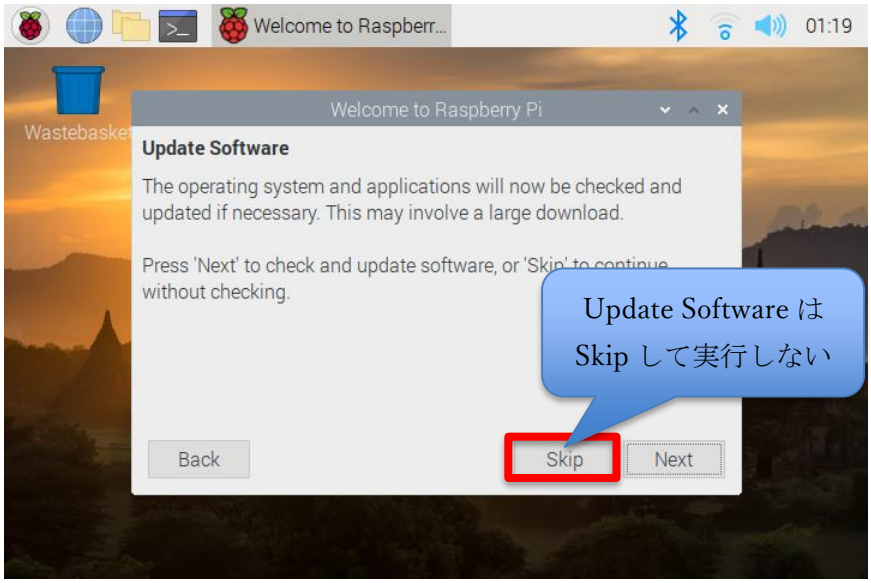
書き込みが終了したら microSD カードを PC からラズベリーパイに差し替えて、ラズベリーパイを起動させましょう。起動後に以下のようなウェインドウが表示された場合は初期設定を行いましょう。



初期設定時や演習時にはネットワーク環境が必要になります。なお、後述しますが、ネットワーク環境や接続デバイス数などによって更新に数時間かかってしまうケースが発生することがあるので、この演習では「ソフトウェアの更新」や「apt upgrade」などのすべてのソフトウェアを更新する作業は行いません。注意してください。

ネットワーク環境のメモ

初期設定では、国、言語、タイムゾーン、パスワード、Wi-Fi（or 有線ケーブル）、画面設定（画面に黒い枠がある場合は「This screen shows a black border around the desktop」のチェックボックスにチェックを入れる）、ソフトウェアの更新などを行います。前述したようにソフトウェアの更新は行いません。次の図ような「Update Software」の画面が開いたら［Skip］で作業をスキップしてください。



初期設定後はラズベリーパイを再起動させます。なお、後で行うリモート接続やデータベースの操作ではパスワードが必要になります。特に、リモート接続の場合、他の人のデバイスに接続してしないよう、パスワードが同じにならないように設定しましょう。

モニタに依存しますが、画面の解像度が変更できる場合があります。左上プログラムメニュー］→［設定］→［Screen Configuration］から Screen Layout Editor を起動して、［Configure］→［Screens］→［HDMI1］→［解像度］から解像度を選択し、［Configuration］→［適用］で設定することができます。5 インチモニタなどでは 800×600 程度を選んでおきましょう。画面が小さすぎる場合は、後から述べるリモート接続で接続して操作しましょう。

ラズベリーパイに microSD カードを差し込んで起動したときに画面が映らない場合があります。そのときは、microSD カード内の/boot/config.txt 内の次の設定をコメントアウトして有効化すると映る場合があります。ファイルを編集する場合は、microSD カードをカードリーダーで PC に読み取ってテキストエディタでファイルを修正してください。

```
#hdmi_safe=1    →    hdmi_safe=1
```

バックアップとリストア

microSD カードがもう 1 つあれば、Raspberry Pi 4 に標準で搭載されている「SD Card Copier」を使って Raspberry Pi 4 で利用中の microSD を丸ごとコピーしてバックアップすることができます。このときには、microSD カード用のカードリーダーが 1 つ必要になります。

SD Card Copier を使ったバックアップ

まず、microSD カードリーダーとバックアップ用の microSD カード（既存の microSD カードと同じものを推奨）を用意します。microSD カードリーダーにバックアップ用の microSD カードを差し込みます。メニューの［アクセサリ］→［SD Card Copier］からソフトを起動して、バックアップ用 SD カードにコピーします。

SD Card Copier を使ったリストア

バックアップ用 microSD カードに差し替えて起動します。

また、microSD カードの読み書きツールを使うことでバックアップとリストアを行うことも可能です. 例えば、Windows 環境の場合、microSD カードの読み書きツールである「Win32 Disk Imager」を使うことで microSD カード全体をイメージファイルとして PC に保存してバックアップをとることができます。

Win32 Disk Imager
https://ja.osdn.net/projects/sfnet_win32diskimager/

Win32 Disk Imager でのバックアップ

Win32 Disk Imager を起動し、保存したい microSD カード内の先頭ドライブと保存する img ファイルの PC の場所と名前（要「.img」拡張子）を設定し「Read」ボタンを押すとバックアップが行えます。「Read Successful.」というメッセージが出ればバックアップ完了となります。

Win32 Disk Imager でのリストア

リストア用 microSD カードを FAT32 形式でフォーマットしておきます。Win32 Disk Imager を起動し、復元用の microSD カードの先頭ドライブとバックアップした img ファイルの 2 つを選択し「Write」ボタンを押すとリストアが行えます。「Write Successful.」というメッセージが出ればリストア完了となります。

TensorFlow を使ったラズベリーパイ上での物体認証

ここでは、まず以前の内容（10/2）で既に実施している（１）TensorFlow を使った物体認証をラズベリーパイ上で実現してみましょう。その後に、モバイルデバイスや IoT デバイスのための Deep Learning フレームワークである（２）TensorFlow Lite を使ったラズベリーパイ上での物体認証を実現（11/6）してみましょう。

（１）TensorFlow を使った物体認証
ここでは、10/02 に実施した「CNN と物体検出手法」の演習で実施したリアルタイム検出を、ラズベリーパイ上で再現することを目的としています。実施する演習の流れは以下のような流れです。

1. 対象物を撮影する
2. 背景画像を撮影する

（ラズベリーパイ）

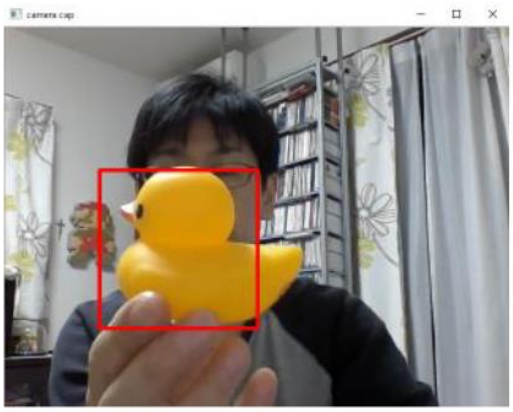
3. 対象物画像を水増しして 300 枚用意する
4. 背景画像を分割して 625 枚用意する
5. 対象物画像と背景画像を csv ファイルに記述する
6. 用意したデータで学習を行う

（Google Colab）

7. 学習されたデータを用いて、リアルタイムで判定させる

（ラズベリーパイ）

完成図は以下になります（過去のスライドより転載）。



ここでは上記に以下のラズベリーパイの演習を追加しています。各自の進み具合に合わせて実施しましょう。操作になれていない人は、無理に以下はやらなくてもよいです。今回の目的はラズベリーパイをセットアップし、Linux 系 OS 上で TensorFlow を実行できるようにすることです。上記の 1-7 を確実にやりましょう。

8. ラズベリーパイにデータベースを準備する
9. 判定結果をデータベースに蓄積させる
10. 蓄積した結果の集計を E-mail で任意のメールアドレスに送信する

ここでは 10/02 のプログラムや手順を一部改変して実現しています。元となるプログラムと改変後のプログラムは以下になります。

元となるプログラム（10/02）	改変後のプログラム
08_Augmentation_Learning.ipynb	14-1a_Augmentation_Learning.ipynb
detection_roi.py	14-1b_detection_roi.py

▼ 改変後のプログラムの概要

14-1a_Augmentation_Learning.ipynb

ラズベリーパイで読み込むことができる学習モデルを作成できるように改変したプログラムです。具体的には、Google Colab の TensorFlow のバージョンをラズベリーパイにインストールするバージョンと同じにするために TensorFlow 2.2.0 にダウングレードさせています。

異なるバージョンの TensorFlow をすでに使っていた場合は、ダウングレードした後にランタイムをリスタートしてください。新しいノートブックを開いたときは、標準のバージョンに戻っているので、必要があれば再度ダウングレードしてください。また、ランタイムの接続が切れて時間がたった場合も、標準のバージョンに戻っていることがあるので、再度ダウングレードから実施する必要があります。

14-1b_detection_roi.py

学習モデルの名前（detection_weight_tf220.h5）を修正済みのプログラムです。

最後に、この演習（１）で利用するプログラムを以下の表にまとめます。

演習（１）で利用するプログラム	
14-1a_Augmentation_Learning.ipynb	Colab 用プログラム。詳細は上記
14-1b_detection_roi.py	ラズベリーパイ用プログラム。詳細は上記
14-1c_database_select.py	データベース検索のテスト用
14-1d_database_insert.py	データベースデータ挿入のテスト用
14-1e_detecRoi_wDB.py	判定結果をデータベースに蓄積する改変版
14-1f_sendResult.py	蓄積した判定結果の集計を E-mail で送信

それでは先に述べた流れに従って進めていきましょう。

1. 対象物を撮影する

まず初期設定を行いましょう。

①初期設定
インターネットに接続した後、ターミナルを起動します。

画面上部のアイコン（LXTerminal）
または
［左上プログラムメニュー］→［アクセサリ］→［LXTerminal］から起動

ターミナル上で、次の apt コマンドを使って Raspberry Pi OS にパッケージやライブラリをインストールします。※以下、インターネットに接続している必要があります。以下のコマンドで apt のリポジトリ情報（更新パッケージの場所の情報）を更新しましょう。なお、時間の関係上、すべてのソフトウェアをアップグレード（upgrade）する作業は省略します（ネット環境とデバイス数によっては数時間かかるケースが発生してしまいます）。

```
sudo apt update
```

Linux のターミナルでは以下の様なコマンドをよく使います。

コマンド	説明	使用例
cd	ディレクトリ移動	cd　移動先ディレクトリ名 cd ..（上位パスのディレクトリに移動）
ls	現在のディレクトリ内のディレクトリやフォルダを表示する	(中身を見てみたいディレクトリに cd で移動して) ls ls　見たいディレクトリ名(相対パスで指定)
mv	ファイルの移動	mv　移動させたいファイル名　移動先
mv	ファイル名の変更	mv　元ファイル名　新しいファイル名
cp	ファイルのコピー	cp　コピー元ファイル名　コピー先ファイル名
mkdir	ディレクトリの作成	mkdir　作成するディレクトリ名
rmdir	空のディレクトリの削除	rmdir　削除する空のディレクトリ名
rm	ファイルの削除	rm　削除するファイル名
cat	ファイルの中身の表示	cat　ファイル名
more	ファイルの中身の表示(スペースキーでページ移動)	less　ファイル名
man	マニュアルを表示する	man　コマンド名

mousepad	テキストエディタの起動	mousepad ファイル名
sudo	管理者権限の付与(要パスワード)	sudo コマンド名
apt	パッケージ管理システム	apt install パッケージ名
python3	python 3.x の起動	python3 python3 ファイル名
pip3	python3.x のパッケージ管理システム	pip3 install パッケージ名

apt と pip3 の大まかな違いですが、PC のシステム全体でパッケージを管理するのが apt です。一方、python3. x のパッケージを管理するのが pip3 になります。

②リモート接続の設定
リモートでラズベリーパイを操作したいときは、RDP（Remote Desktop Protocol）や VNC（Virtual Network Computing）、X Windows System、SSH などを利用することができます。VNC や SSH を利用する場合は、［メニューボタン］→［Raspberry Pi の設定］で VNC や SSH を有効にしてラズベリーパイを再起動しましょう。なお、ここでは RDP での接続方法について見ていきます。

まずラズベリーパイに RDP を以下のコマンドでインストールしましょう。

```
sudo apt install xrdp
```

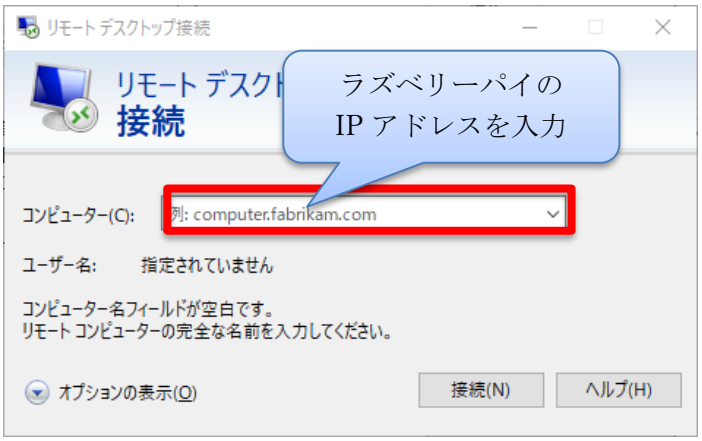
RDP を起動するコマンドは以下になります。

```
sudo service xrdp restart
```

次に、リモートデスクトップクライアントからラズベリーパイの IP アドレスを入力して、ラズベリーパイの ID とパスワードを入力すればリモートで作業ができます。Windows の場合は標準でリモートデスクトップクライアントがインストールされています。Windows のスタートメニュー［Windows アクセサリ］→［リモートデスクトップ接続］からリモートデスクトップ接続を起動してください。Mac OS では Microsoft Remote Desktop などのクライアントを利用しましょう。なお、ラズベリーパイの IP アドレスは以下のコマンドから確認できます。

```
ifconfig
```

なお、ラズベリーパイのユーザ ID は pi になります。パスワードが未設定の場合、RDP でリモート接続できません。［左上プログラムメニュー］→［設定］→［Raspberry Pi の設定］から設定ウィンドウを開き、［システムの設定］タブにある［パスワードの変更］からパスワードを設定してください。



もし RDP を終了するには以下のコマンドを利用しましょう。終了しない場合はコマンドを入力しなくてよいです。

```
sudo service xrdp stop
```

RDP を利用した場合、クリップボードを共有することで、互いの画面でコピーとペーストが利用できます。例えば、Windows 上のあるファイルをコピーして、ラズベリーパイのデスクトップなどにペーストすることができます（その逆もできます）。ドラッグ＆ドロップではない点に注意しましょう。また、フォルダはコピーすることができません。複数のファイルをコピー＆ペーストする場合は、ファイルを複数選択してコピー＆ペーストするか、ZIP 形式に圧縮するなどしてまとめてコピー＆ペーストするようにしましょう。

ラズベリーパイを再起動したときは RDP の起動コマンドを入力しなおす必要があります。ここでは、起動処理を行う systemd を使って xrdp を自動起動させましょう。まず、テキストエディタを使って以下のようなファイルを myxrdp.service という名前で作成します（今回は準備済みです）。

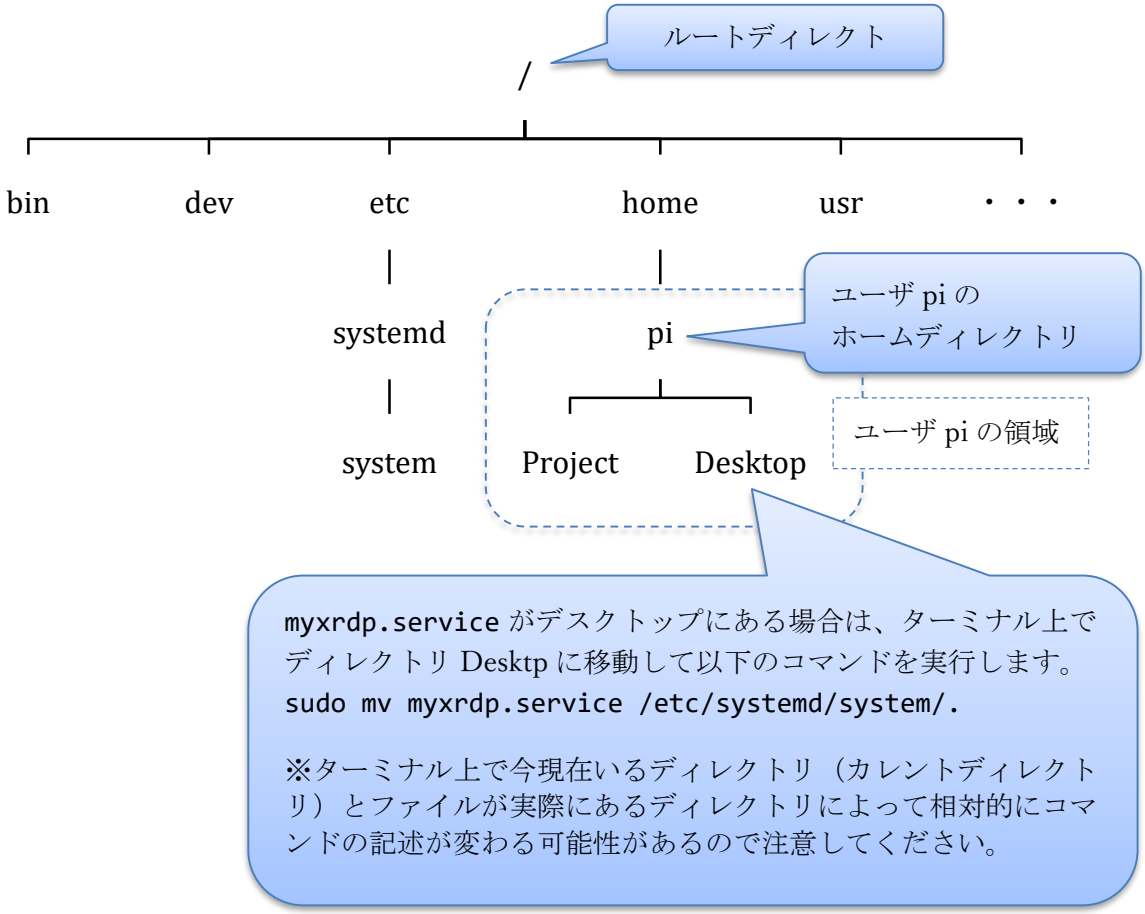
myxrdp.service

```
[Unit]
Description=myXRDP
After=syslog.target

[Service]
Type=simple
WorkingDirectory=/usr/share/xrdp
ExecStart=/usr/sbin/xrdp
```

```
TimeoutStopSec=5
StandardOutput=null

[Install]
WantedBy = multi-user.target
```

このファイルを /etc/systemd/system/以下に移動して読み込みます。移動は、以下の mv コマンドで移動させましょう（マウス操作だと権限の問題で移動できません）。mv コマンドは、ターミナル上でファイルのある場所から実施する必要があります。
例えば、myxrdp.service をデスクトップにコピーしている場合は、cd でホームディレクトリに移動し、cd Desktop を実行して Desktop ディレクトリに移動したり、
cd /home/pi/Desktop で Desktop ディレクトリに絶対パスで直接移動して、以下のコマンドを実行する必要があります。

```
sudo mv myxrdp.service /etc/systemd/system/.
```



そして次に、以下のコマンドでファイルを読み込みます。

```
sudo systemctl daemon-reload
```

自動起動の設定が以下になります。これでラズベリーパイを再起動したときにラズベリーパイ上で自動的に RDP が起動し、リモート接続サービスを待ち受けている状態になります。あとは Windows 側でリモートデスクトップ接続を実施すれば遠隔で操作ができます。

```
sudo systemctl enable myxrdp.service
```

以下のコマンドでサービスの一覧が確認できます。myxrdp.service がリストにあれば登録されている状態です。なお、閲覧モードはスペースキーで次のページに移動し、終了する時はキーボードの q を押してください。

```
systemctl list-unit-files --type service
```

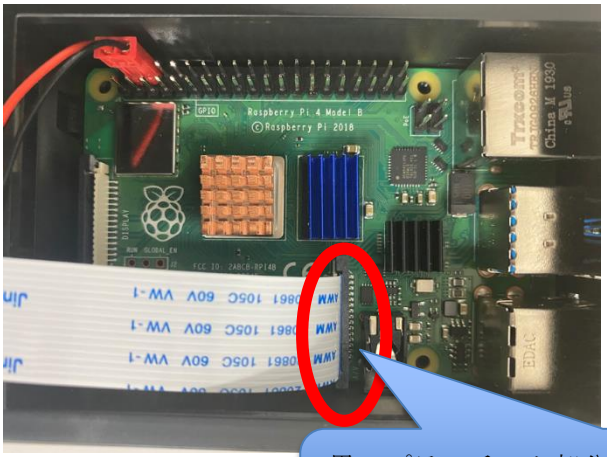
なお、自動起動をやめたり、個別にサービスを起動したり、停止したりするコマンドは以下になりますが、ここでは入力はしません。

自動起動の停止
```
sudo systemctl disable myxrdp.service
```

起動
```
sudo systemctl start myxrdp.service
```

停止
```
sudo systemctl stop myxrdp.service
```

③Camera Module V2 の接続
カメラモジュールをラズベリーパイに接続していない場合はラズベリーパイをシャットダウンして接続しましょう。すでに接続している場合は、このページはスキップして次のページから進めてください。ケースを利用する場合は以下の様になります。また、カメラとラズベリーパイの接続口では、黒いプラスチック部分を上にスライドさせて開いた後に、カメラの接続先を差し込み黒いプラスチック部分を閉じて固定してください。



黒いプラスチック部分を上にスライドさせて開いた後にカメラの接続先を差し込み黒いプラスチック部分を閉じる

④Camera Module の有効化

まず、［左上プログラムメニュー］→［設定］→［Raspberry Pi の設定］をクリックして設定ウィンドウを開き、［インターフェイス］タブのカメラの有効にチェックを入れて［OK］を押して再起動します。

（Camera Module V2 の場合）次に、起動後にターミナルで

```
vcgencmd get_camera
```

と入力して

```
supported=1 detected=1
```

とターミナルに表示されればカメラが有効になっています。もし、上記のようになっていない場合、カメラの接続をもう一度やり直したり、ラズベリーパイを再起動したりしてください。

※ラズベリーパイとカメラモジュールを接続しているリボンケーブルがむき出しの状態なので、引っ張ったときや当たったときの衝撃で認証が外れることがたまにあります。そういった場合はカメラモジュールを１回外して付け直したり、ラズベリーパイを再起動することで正しく認識し直すことが多いので試してみてください。

※USB 接続ウェブカメラの場合
上記のコマンドを入力しても「supported=1 detected=1」とは表示されません。その代わりに以下のコマンドを入力して USB 接続機器を確認してください。USB で接続されているデバイスがリスト表示されるのでウェブカメラが認識されているか確認してください。

```
lsusb
```

USB 接続のウェブカメラが認識されていればカメラアプリなどをインストールして利用できるか確認してみましょう。例えば、guvcview などがあります。

```
sudo apt install guvcview
```

⑤Camera Module V2 の写真撮影機能の確認
写真を撮影する前に以下のコマンドでホームディレクトリ（/home/pi）に移動しておきましょう。

```
cd
```

ターミナルでホームディレクトリがカレントディレクトリになっている状態から以下のコマンドを入力すると、5 秒後に自動写真撮影が行われます。正しく動作していた場合、「/home/pi」のディレクトリ（ホームディレクトリ）に pict.jpg という名前で撮影した写真データが保存されています。※ルートディレクトリ（/）や bin、home などのホームディレクトリの外はシステム管理者の領域です。ホームディレクトリの外では書き込み権限がないので書き込みエラーとなるので注意しましょう。もしホームディレクトリに保存するのをコマンド上で指示したい場合は raspistill -o ~/pict.jpg のようなコマンドの書き方になります（~はホームディレクトリを指し示す記号）。

```
raspistill -o pict.jpg
```

ここでカメラの上下を確認しましょう。上下が逆の場合は、-hf -vf オプションをつけることで反転させて撮影することができます。

```
raspistill -o pict.jpg -hf -vf
```

これでようやく「1. 対象物を撮影する」ができるようになりました。以前の演習と同様に、リアルタイムで検出させる物体を各自で撮影してください。撮影した写真は任意の名前で保存しましょう（例えば、pict.jpg）。保存した画像は Windows 側に移し（例えばリモートデスクトップ接続でコピー＆ペースト）、前回同様に Windows PC 上で XnView を使用して対象物だけを切り抜いて、切り抜いた画像を target.jpg という名前で保存しましょう。

XnView でのおおよその作業の流れは以下のようになります。
1. XnView がまだインストールされていない場合は、オフィシャルサイト（https://www.xnview.com/）やダウンロードサイト（窓の杜など）からインストールしましょう。
2. ［編集］→［選択範囲縦横率設定］→［1:1 (1.00)］を選択しましょう。これで選択ツールを正方形に固定することができます。
3. 対象物を囲むように選択したのち、［編集］→［選択範囲トリミング］を選択し、画像を切り抜きましょう。
4. 切り抜いたのち、［画像］→［リサイズ］でサイズ変更画面を開き、縦横 128 ピクセルを指定してサイズ変更しましょう。
5. 最後に target.jpg という名前で切り抜いた画像を保存しましょう。

2. 背景画像を撮影する

前回同様、自分が映りこまないようにして背景画像を撮影します。撮影した写真の名前を other.jpg としましょう。

3. 対象物画像を水増しして 300 枚用意する
4. 背景画像を分割して 625 枚用意する
5. 対象物画像と背景画像を csv ファイルに記述する
6. 用意したデータで学習を行う

前回同様、Google Colab で実施しましょう。
Google Colab で 14-1a_Augmentation_Learning.ipynb を開きましょう。プログラムの説明はプログラム内のコメントを参照してください。Google Colab のファイル操作を行う左ウィンドウ側で以下の 4 つのファイルをアップロードしましょう。

```
・target.jpg                    # 認証する対象が映っている 128×128 の画像
・other.jpg                     # 背景画像（大きさは任意）
・target_or_other_train.csv     # 学習（訓練用）ファイルパスとラベル
・target_or_other_test.csv      # 検証用ファイルパスとラベル
```

学習が終わったら、作成された重みファイル（detection_weight_tf220.h5)をダウンロードしましょう。

前述したプログラム群やインストール用のデータファイル（.whl ファイル）、作成した重みファイル（detection_weight_tf220.h5）は、次のページで説明しているラズベリーパイのホームディレクトリに作成する Project ディレクトリ(/home/pi/Project)に RDP でコピー＆ペーストして移動しておきましょう。

```
・14-1b_detection_roi.py                          ┐
・14-1c_database_select.py                        │
・14-1d_database_insert.py                        │
・14-1e_detecRoi_wDB.py                           ├ 準備済みのファイル
・14-1f_sendResult.py                             │
・scipy-1.4.1-cp37-cp37m-linux_armv7l.whl         │
・tensorflow-2.2.0-cp37-cp37m-linux_armv7l.whl    ┘

・detection_weight_tf220.h5                       ┐
                                                  ├ 各自が作成したモデル
                                                  ┘
```

7. 学習されたデータを用いて、リアルタイムで判定させる

ここではまずラズベリーパイ上で TensorFlow（2.2.0）と OpenCV（4.1.0.25）が動くように準備していきましょう。

まずプログラムなどを保存するディレクトリ Project を作ってから作業を進めます。以下、LXTerminal 上で操作を行いましょう。

まず各自のホームディレクトリ(/home/pi)に移動しましょう。

```
cd
```

ホームディレクトリに移動したらディレクトリ Project を作成します。

```
mkdir Project
```

ls コマンドで作成できたか確認しましょう。

```
ls
```

その後で、Project ディレクトリ内に移動しましょう。

```
cd Project
```

①仮想環境のインストール
pip によるパッケージの導入状態をプロジェクトごとに独立させるために仮想環境 venv をインストールしましょう。venv は python3.3 以降に python の標準機能となっているものです。仮想環境を作成するコマンドは以下になります。後で仮想環境名を TF として実行しましょう。ここではコマンドの確認のみです。

```
python3 -m venv [仮想環境名]
```

まず現在の環境を確認しておきましょう。例えば以下のコマンド（-V は大文字の V です）で python のバージョンを確認しておいてください。おそらく python2.x 系が表示されるはずです。

```
python -V
```

それでは仮想環境名に TF を指定してコマンドを実行しましょう。

```
python3 -m venv TF
```

仮想環境下のコマンドや情報を保存する仮想環境名のディレクトリ（ここでは TF）が作成されます。正しく作成されたか ls コマンドで確認しておいてください。

それでは以下のコマンドで仮想環境 TF を有効にします。なお、以下のコマンドは Project ディレクトリ直下から相対パスで activate コマンドを読み込んでいます。カレントディレクトリ（現在の場所）が変わるとコマンドの相対パスも変わってしまうので注意しましょう。例えば、TF 内に移動している場合は「source bin/activate」になりますし、TF/bin 内に移動している場合は「source activate」になります。

```
source TF/bin/activate
```

仮想環境が有効になると、コマンドプロンプトの前のカッコ内に仮想環境名が表示されます。ここでは例えば以下の様に表示されるはずです。再起動したときに activate するのを忘れないように注意しましょう。

```
(TF) pi@raspberry:~ $
```

それでは、以下のコマンド（-V は大文字の V です）で、python のバージョンを確認してみましょう。python のバージョンが 3.x 系に切り替わっているはずです。

```
python -V
```

以上で TensorFlow と OpenCV をインストールする準備が整いました。なお、仮想環境を終了するときは以下のコマンドで終了します。試しに終了してみた人は、もう一度上記の activate コマンドを実行して仮想環境 TF を有効にしてください。

```
deactivate
```

以下の演習は venv の仮想環境 TF 内で実施していきます。

①TensorFlow（2.2.0）のインストール
まず、TensorFlow と関連するライブラリをインストールしましょう。なお、sudo apt update を未実施の場合、インストールできないので注意しましょう。apt の-y オプションは、インストールするかどうかの確認[Yes/no]をあらかじめ Yes と選んでおくオプションになります。操作になれていない人は 1 つずつインストールしていきましょう。

・scipy 用の fortran コンパイラ

```
sudo apt -y install gfortran
```

・h5py 用ライブラリ、DNS 用 C ライブラリ、ベクトル計算用 C++ライブラリ

```
sudo apt -y install libhdf5-dev libc-ares-dev libeigen3-dev
```

・線形代数などの数値計算用ライブラリ

```
sudo apt -y install libatlas-base-dev libopenblas-dev libblas-dev
```

・数値計算用ライブラリ、cython

```
sudo apt -y install liblapack-dev cython
```

以下は pip3 コマンドになるので注意してください。pybind11 の 11 は数字の「イチイチ」です。

・C++11 用ライブラリ

```
pip3 install pybind11
```

・バイナリデータファイルフォーマット HDF 用ライブラリ

```
pip3 install h5py
```

pip3 の--upgrade オプションはアップグレードを指示します。

・セットアップツール

```
pip3 install --upgrade setuptools
```

あらかじめダウンロードしておいた scipy-1.4.1-cp37-cp37m-linux_armv7l.whl と tensorflow-2.2.0-cp37-cp37m-linux_armv7l.whl を使って scipy（1.4.1）と TensorFlow（2.2.0）をインストールしましょう。以下のコマンドはターミナル上で scipy-1.4.1-cp37-cp37m-linux_armv7l.whl と tensorflow-2.2.0-cp37-cp37m-linux_armv7l.whl のファイルがある場所でそれぞれ実行しましょう。

・数値解析ソフトウェア

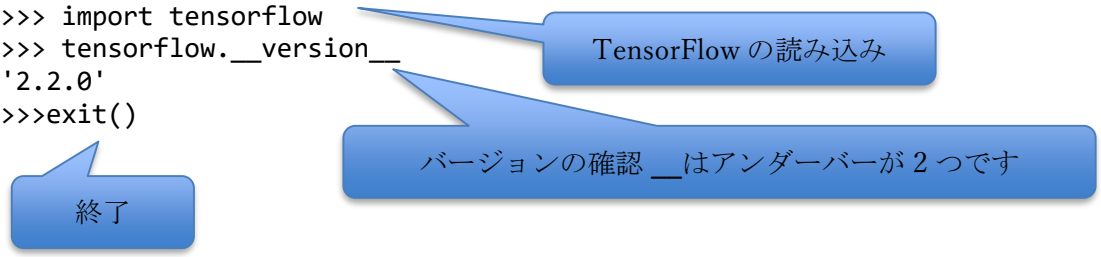
```
pip3 install scipy-1.4.1-cp37-cp37m-linux_armv7l.whl
```

・機械学習 tensorflow

```
pip3 install tensorflow-2.2.0-cp37-cp37m-linux_armv7l.whl
```

TensorFlow が無事にインストールされたら、python3 コマンドで python の対話モードに移行して import とバージョンを確認してみましょう。

```
pi@raspberrypi:~ $ python3
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>> tensorflow.__version__
'2.2.0'
>>>exit()
```



②OpenCV（4.1.0.25）のインストール
次に、OpenCV をインストールします。まず、OpenCV の動作に必要となるライブラリをインストールします。ここも操作になれていない人は 1 つずつインストールしましょう。なお、先ほどインストールしたものには下線を引いています。

・h5py 用ライブラリ

```
sudo apt -y install libhdf5-dev libhdf5-103
```

・GUI アプリケーション用 Qt 関連ライブラリ

```
sudo apt -y libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
```

・数値計算用、JPG 画像処理用ライブラリ

```
sudo apt -y libatlas-base-dev libjasper-dev
```

バージョンによって　libhdf5-XXX　の XXX が変更する可能性があるのでエラーが出た場合はエラーメッセージに注意しましょう。

python 用の OpenCV（バージョン 4.1.0.25）を pip3 でインストールします。

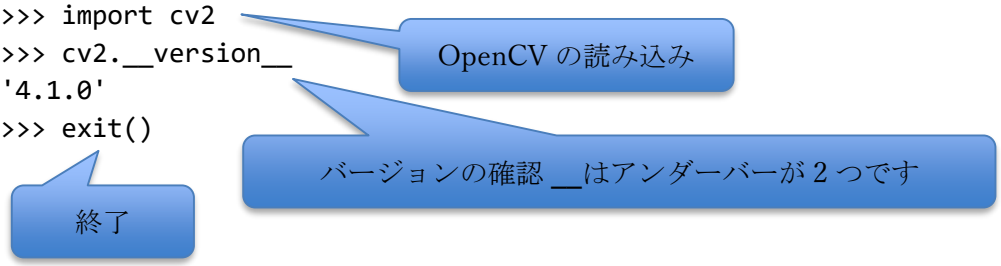
```
pip3 install opencv-python==4.1.0.25
```

ダウンロードが 100%になる前に終了してしまう場合は、以下の様にタイムアウトのオプションをつけて実行しましょう。

```
pip3 install opencv-python==4.1.0.25 --default-timeout=1000
```

opencv-python をインストールしたら動作するかどうか python3 の対話モードから確認しましょう。

```
$ python3
Python 3.7.3 (default, Feb  21 2020, 18:55:00)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.1.0'
>>> exit()
```



もし、python の対話モードの import cv2 で以下の様なエラーが出た場合は、以下の設定を行いましょう。エラーが出なかった場合は次のページに進みましょう。

```
ImportError: /home/pi/cv2/cv2.cpython-37m-arm-linux-gnueabihf.so:
undefined symbol: __atomic_fetch_add_8
```

ターミナルでパスを通す以下の設定を実行しましょう。

```
LD_PRELOAD=/usr/lib/arm-linux-gnueabihf/libatomic.so.1
```

再起動後も上記の設定を有効にしたい場合はホームディレクトリにある隠し設定ファイルである.bashrc に設定を書き込みます。
テキストエディタで.bashrc を開き次の 1 行を追記します。

```
export LD_PRELOAD=/usr/lib/arm-linux-gnueabihf/libatomic.so.1
```

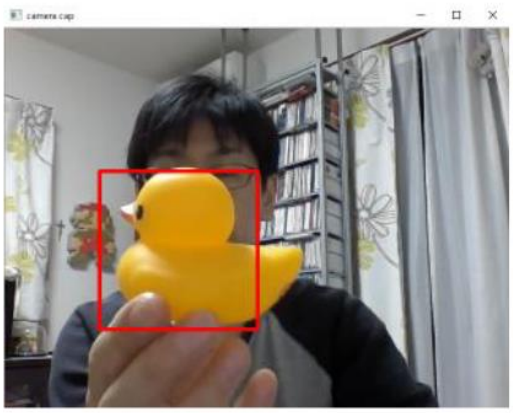
ファイルは、テキストエディタ mousepad で開きましょう。修正した.bashrc は、以下のコマンドで読み込んで反映させます。

```
source .bashrc
```

③14-1b_detection_roi.py によるリアルタイム検出
10/2 に実施したのと同じリアルタイム検出をラズベリーパイ上で実現しましょう。学習済みの重みファイルである detection_weight_tf220.h5 と 14-1b_detection_roi.py を先ほど作成した Project ディレクトリに入れて、14-1b_detection_roi.py を以下のコマンドで実行しましょう。

```
python3 14-1b_detection_roi.py
```

検出されると緑色の枠が赤色の枠に色が変わります。



プログラムを終了するときは ESC キーを押しましょう。

以上で先日の演習をラズベリーパイ上で再現する事ができました。作業が順調に進んだ人はさらに以下を実施していきましょう。

8. ラズベリーパイにデータベースを準備する
9. 判定結果をデータベースに蓄積させる
10. 蓄積した結果の集計を E-mail で任意のメールアドレスに送信する

8. ラズベリーパイにデータベースを準備する
それではラズベリーパイにリレーショナルデータベースである MariaDB を以下のコマンドでインストールしましょう。MariaDB は MySQL から派生した GPL ライセンスのオープンソースウェアになります。そのためか、MariaDB 内のコマンドには mysql という名前がついています。

```
sudo apt install mariadb-server mariadb-client
```

次に、python から MySQL を操作するパッケージの 1 つである PyMySQL を次のコマンドでインストールします。

```
pip3 install PyMySQL
```

まず管理者である root から mysql にログインします。-u はユーザ指定のオプションです。

```
sudo mysql -u root
```

ログインに成功すると、以下のようなプロンプトが表示されます。まだデータベースを指定していない状態（none）です。

```
MariaDB[none]>
```

それでは新しいデータベースを作りましょう。以下の大文字部分が SQL の命令です。大文字の命令は小文字で入力しても大丈夫です。

```
CREATE DATABASE mydb;
```

コマンドが成功すると以下のようなメッセージ文が表示されます。※数字は少し変わります。
Query OK, 1 rows affected (0.001 sec)

次に、SQL 文を使ってユーザ pi を追加します。以下を 1 行ずつ入力して Enter キーで実行してください。パスワードは、最初に作成したパスワードと同じにしておきましょう。

```
CREATE USER 'pi'@'localhost' IDENTIFIED BY 'パスワード';

GRANT ALL PRIVILEGES ON mydb.* TO 'pi'@'localhost';

quit
```

今作成したDB用のユーザpiでMySQLにログインします。-pオプションは、パスワード入力機能を呼び出すオプションです。

```
mysql -u pi -p
```

パスワード認証に成功すると、MySQLにログインした状態になります。それでは先ほど作成したmydbデータベースに切り替えてみましょう。

```
USE mydb;
```

そうするとプロンプト部分が以下のように切り替わって、mydbの作業モードに切り替わります。

```
MariaDB[mydb]>
```

それではmydbの中に新しいテーブルを以下のコマンドで作成します。なお、以下のコマンドは1行なのに注意してください。

```
CREATE TABLE `mytable` (`id` int NOT NULL AUTO_INCREMENT PRIMARY KEY
COMMENT "ID", `result` VARCHAR(100) NOT NULL COMMENT "結果", `accuracy`
VARCHAR(100) NOT NULL COMMENT "確率", `created` datetime DEFAULT NULL
COMMENT "登録日") ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

上のコマンドは、データベースmydb内に、テーブルmytableを作成しています.
mytalbeには以下のカラム（列）が作られます。

id：識別番号。自動的にカウントされる。
result：画像判別の結果を格納する。0だと学習した対象物だと識別し、1だとその他だと識別している。
accuracy：対象物だと予測している確率

テーブルのカラム（列）が正しく作成されているか以下のコマンドで確認しましょう。

```
describe mytable;
```

以下のような各カラムの情報が表示されるので確認しましょう。

```
MariaDB [mydb]> describe mytable;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| result   | varchar(100) | NO   |     | NULL    |                |
| accuracy | varchar(100) | NO   |     | NULL    |                |
| created  | datetime    | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.004 sec)
```

次に、試しに仮のデータを以下のコマンドで挿入してみましょう。

```
INSERT INTO mytable (result, accuracy, created) VALUES ("1", "2", now());
```

データが正しく挿入されているか以下のコマンドで確認しましょう。

```
SELECT * FROM mytable;
```

以下のように id は自動的に番号がつき、仮のデータとして 1 と 2、挿入した日時が表示されるはずです。

```
MariaDB [mydb]> SELECT * FROM mytable;
+----+--------+----------+---------------------+
| id | result | accuracy | created             |
+----+--------+----------+---------------------+
|  1 | 1      | 2        | 2020-10-17 09:37:36 |
+----+--------+----------+---------------------+
1 row in set (0.001 sec)
```

それでは一旦 MariaDB から抜け出して、python のプログラムからデータベースが操作できるか確認してみましょう。

```
quit
```

次にプログラム（14-1c_database_select.py）から検索結果を表示させてみましょう。プログラム内のデータベースのパスワードを各自で登録したパスワードに変更して保存してください。

14-1c_database_select.py の修正箇所

```
#PyMySQL のインポート
import pymysql.cursors


#データベースへの接続
connection = pymysql.connect(host='localhost',
                     user='pi',
                     password='データベースのパスワード',
                     db='mydb',
                     charset='utf8mb4',
                     cursorclass=pymysql.cursors.DictCursor)


（以下省略）
```

データベースのパスワードに書き換えます

修正して保存したプログラムを以下のコマンドを使ってターミナル上で実行すると検索結果が表示されます。

```
python3 14-1c_database_select.py
```

次にプログラム（14-1d_database_insert.py）から仮のデータをデータベースに挿入してみます。挿入前と後の結果が表示されるので合わせて確認しましょう。修正箇所は先ほどの 14-1c_database_select.py の修正箇所と同じで、プログラム内のデータベースのパスワードを各自で登録したパスワードに変更して保存してください。

修正して保存したプログラムを以下のコマンドを使ってターミナル上で実行すると検索結果が表示されます。

```
python3 14-1d_database_insert.py
```

9. 判定結果をデータベースに蓄積させる

14-1e_detecRoi_wDB.py は、判別中にキーボードの a を押すとその時の判別結果をデータベースに挿入するように 14-1b_detection_roi.py を改変したものです。プログラム内の中ほど（43 行目）に、これまでと同様にデータベースのパスワードの修正部分があるので、各自で登録したパスワードに変更して保存してください。

それでは以下のコマンドで物体検出を実施しながらキーボードの a を押してその時の結果をデータベースに記録してみましょう。

```
python3 14-1e_detecRoi_wDB.py
```

記録できたかどうか先ほどの 14-1c_database_select.py を使って確認してみましょう。

```
python3 14-1c_database_select.py
```

10. 蓄積した結果の集計を E-mail で任意のメールアドレスに送信する

ここではメーラーとして Gmail を利用する方法を紹介します。まず Gmail の［Google アカウントを管理］からアカウント設定画面を開き、［セキュリティ］の項目にある［Google へのログイン］において、2 段階認証プロセスの設定を有効化した後に、アプリパスワードを生成しましょう。今回はこのアプリパスワードを利用します。なお、フリーの Gmail アドレスにはアプリパスワードの機能がありますが、契約している Gmail の種類によっては表示されない場合もあります。表示されない場合は他のものを利用するか、ここでは確認のみにとどめておいてください。

アプリパスワードの設定では、デバイスにその他を選びましょう。

生成されたアプリパスワードを利用しましょう。

セキュリティの Goggle へのログインが以下のように変更されます。

以下のプログラム内を修正して実行すると、判別結果の総回数と物体を検出した回数、データを蓄積している期間を E-mail で指定したアドレスに送信します。プログラム内のデータベースのパスワードを各自で登録したパスワードに変更して保存してください。また、あなたの Gmail のアドレスと、あなたの Gmail のアプリパスワードを各人ものに変更し、送信先の E-mail アドレスを修正して保存してください。

```
#PyMySQL のインポート
import pymysql.cursors

#データベースへの接続
connection = pymysql.connect(host='localhost',
                             user='pi',
                             password='データベースのパスワード',
                             db='mydb',
                             charset='utf8mb4',
                             cursorclass=pymysql.cursors.DictCursor)

                          （中略）
```

データベースのパスワードに書き換えます

```
print("sending e-mail...")
if __name__ == "__main__":
    # Gmail アカウント設定
    gmail_addr = "あなたの Gmial のアドレス"
    gmail_pass = "あなたの Gmail のアプリパスワード"
    SMTP = "smtp.gmail.com" #SMTP サーバ
    PORT = 587 #ポート番号

    # 送信メール情報
    from_addr = gmail_addr # 送信元メールアドレス
    to_addr = "e-mail を送る相手のメールアドレス"
    subject = "予測結果の集計" #件名

                                （以下省略）
```

Gmail、アプリパスワードに書き換えます

送り先のメールアドレスに書き換えます

修正して保存したプログラムを以下のコマンドを使ってターミナル上で実行すると、集計結果（判別結果の総回数と物体を検出した回数、データを蓄積している期間）をメール本文に記入した E-mail が送信されます。

```
python3 14-1f_sendResult.py
```

以上で（１）TensorFlow を使った物体認証は終了です。演習が順調に進んで時間が余っている人は次の演習にチャレンジしてみてください。

Extra 演習

（１extraA）TensorFlow を使った物体認証の復習 1

各自でリアルタイム検出を行うことができるか、新しい物体を選んで各自で実施してみましょう。

（１extraB）TensorFlow を使った物体認証の復習 2

microSD カードに raspberry pi OS のイメージを書き込んで、各自でインストールの最初からセットアップができるか実施してみましょう。
※これまで構築したものが削除されるので注意してください。前述したイメージファイルのバックアップをとったり、別の microSD カードを使ったりするなど、完成版の状態を残すようにしてください。

（１extraC）TensorFlow を使った物体認証（VGG-16 版 1）

10/02 に実施した「VGG-16 演習」を各自でラズベリーパイ上に再現してみましょう。具体的には、vgg16_camera.py(10/02)をラズベリーパイ上で動作させてみましょう。そのとき必要があれば必要なセットアップを各自で設定してください。

（１extraD）TensorFlow を使った物体認証（VGG-16 版 2）

10/02 に実施した「VGG-16 演習」の機能を今回実施したプログラムに組み込んでみましょう。具体的には、14-1b_detection_roi.py のモデルを vgg16_camera.py (10/02)のモデルと入れ替えて、14-1b_detection_roi.py のフレーム内で vgg16 を動かしてみましょう。そのとき必要があれば必要なセットアップを各自で設定してください。実現方法はいろいろ考えられるので各自で自由にプログラミングしてみてください。例えば、カメラで読み込んだ映像から切り出した画像に対して、リアルタイムで vgg16 の画像認証を動作させ、その結果をコマンドプロンプトに標準出力させる、、などが考えられます。