

파이썬으로 배우는 AI 로봇

핑퐁 파이썬 활용 2

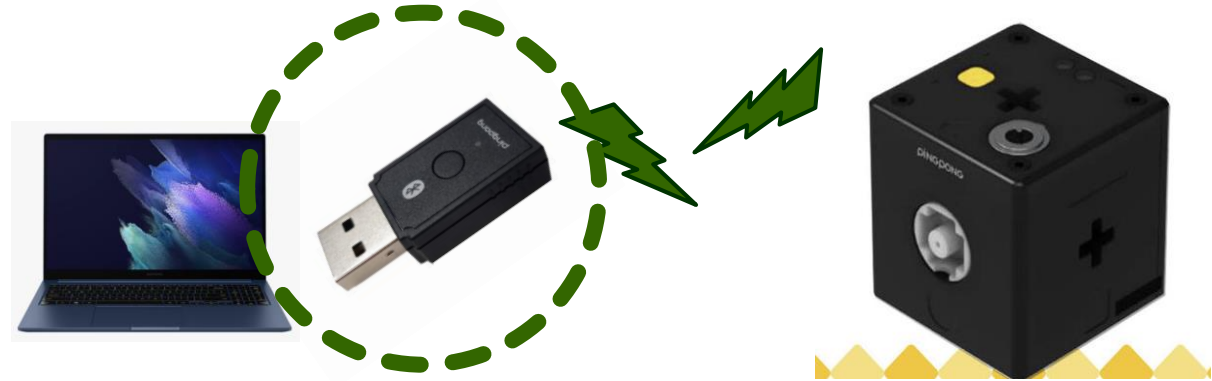
김정래 (Solrootec)

주요 내용

- ▶ Example1: chapter 2
- ▶ Example2: chapter 4
- ▶ Example3: chapter 5
- ▶ Example4: chapter 6
- ▶ Example5: chapter 8
- ▶ Example6: chapter 9
- ▶ Example7: chapter 10

핑퐁 가상환경 설정

핑퐁 USB 동글 활용
핑퐁 큐브와 노트북 연결



노트북 캠 연결 활용

```
# 웹캠 열기.  
PingPongThreadInstance.webcam_open(0) # 노트북 웹캠 이용
```

Chapter 2

● 2. 인공지능 마스크 감지 로봇

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.

● 2. 인공지능 마스크 감지로봇

 Quit Logout

Files Running Clusters

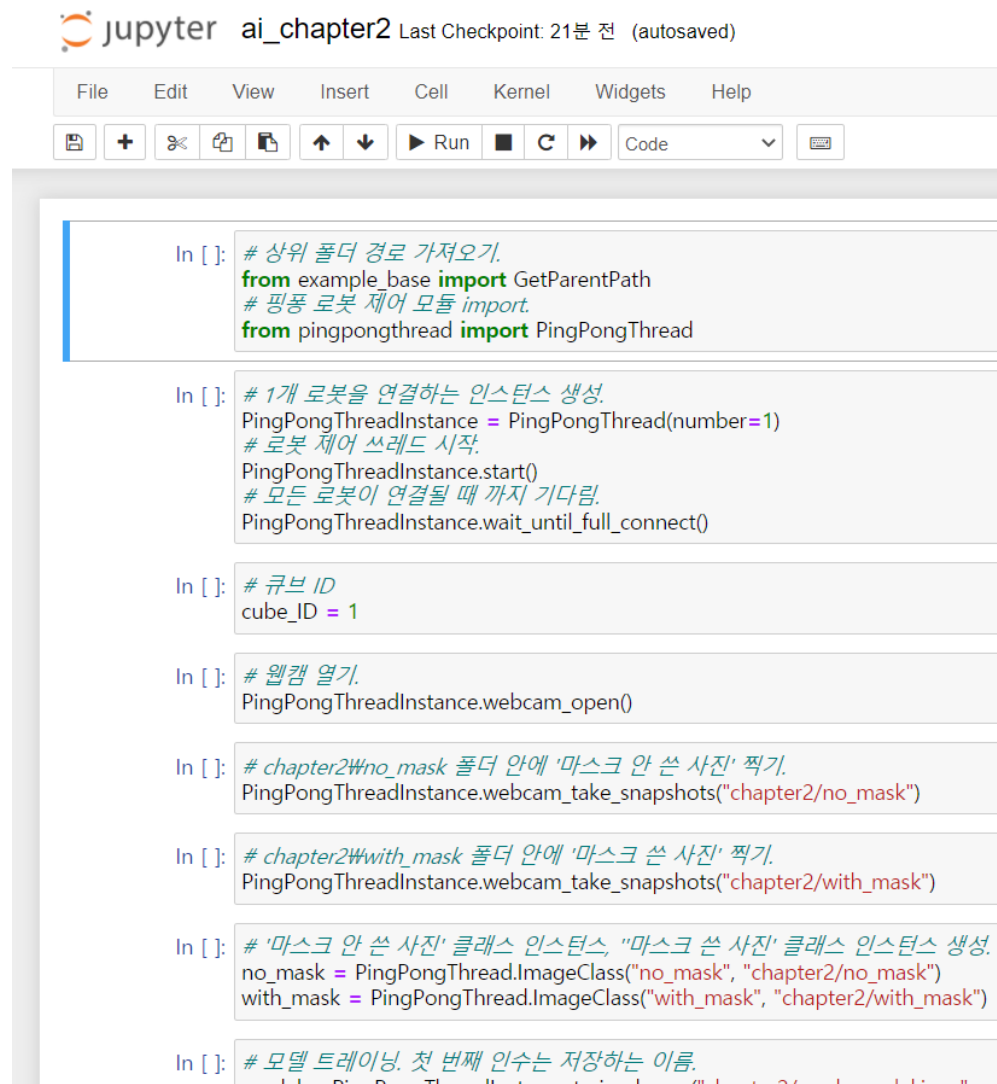
Select items to perform actions on them. Upload New ▾ ↺

<input type="checkbox"/> 0 ▾	/	Name ▾	Last Modified	File size
<input type="checkbox"/>	chapter2		16분 전	
<input type="checkbox"/>	ex_dev		10시간 전	
<input type="checkbox"/>	 ai_chapter2.ipynb		Running 3분 전	463 kB
<input type="checkbox"/>	 ai_chapter2.py		한 시간 전	2.39 kB
<input type="checkbox"/>	 ai_weekx.py		5일 전	1.34 kB
<input type="checkbox"/>	 example_base.py		11일 전	204 B
<input type="checkbox"/>	 get_sensor_example.py		한 시간 전	943 B
<input type="checkbox"/>	 ledmatrix_example.py		11일 전	1.93 kB
<input type="checkbox"/>	 music_example.py		11일 전	487 B
<input type="checkbox"/>	 servo_example.py		11일 전	1.39 kB
<input type="checkbox"/>	 servo_example2.py		11일 전	1.2 kB
<input type="checkbox"/>	 servo_example3.py		11일 전	1.22 kB
<input type="checkbox"/>	 servo_example4.py		11일 전	1.29 kB
<input type="checkbox"/>	 stepper_example.py		3일 전	946 B
<input type="checkbox"/>	 stepper_example2.py		11일 전	1.07 kB
<input type="checkbox"/>	 stepper_example3.py		11일 전	1.1 kB

● 2. 인공지능 마스크 감지 로봇

3.

- ▶ 주피터 노트북에서 'ai_chapter2.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.



The screenshot shows a Jupyter Notebook titled 'ai_chapter2' with a 'Last Checkpoint: 21분 전 (autosaved)' status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, navigation, and execution. The notebook contains several code cells, each starting with 'In []:' and containing Python code with comments in Korean. The code is as follows:

```
In [ ]: # 상위 폴더 경로 가져오기.
from example_base import GetParentPath
# 핑퐁 로봇 제어 모듈 import.
from pingpongthread import PingPongThread

In [ ]: # 1개 로봇을 연결하는 인스턴스 생성.
PingPongThreadInstance = PingPongThread(number=1)
# 로봇 제어 스레드 시작.
PingPongThreadInstance.start()
# 모든 로봇이 연결될 때 까지 기다림.
PingPongThreadInstance.wait_until_full_connect()

In [ ]: # 큐브 ID
cube_ID = 1

In [ ]: # 웹캠 열기.
PingPongThreadInstance.webcam_open()

In [ ]: # chapter2\#no_mask 폴더 안에 '마스크 안 쓴 사진' 찍기.
PingPongThreadInstance.webcam_take_snapshots("chapter2/no_mask")

In [ ]: # chapter2\#with_mask 폴더 안에 '마스크 쓴 사진' 찍기.
PingPongThreadInstance.webcam_take_snapshots("chapter2/with_mask")

In [ ]: # '마스크 안 쓴 사진' 클래스 인스턴스, '마스크 쓴 사진' 클래스 인스턴스 생성.
no_mask = PingPongThread.ImageClass("no_mask", "chapter2/no_mask")
with_mask = PingPongThread.ImageClass("with_mask", "chapter2/with_mask")

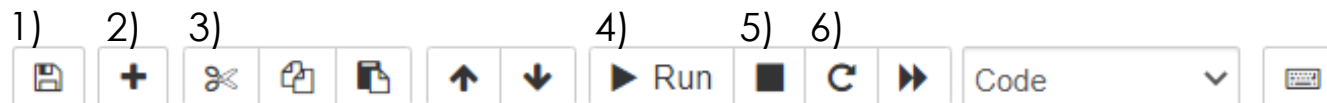
In [ ]: # 모델 트레이닝. 첫 번째 인수는 저장하는 이름.
```

● 2. 인공지능 마스크 감지 로봇

* 주의: 실행한 셀이 끝나기 전에 다른 셀을 실행하지 않는다. 실행 중인 셀은 셀 왼쪽에 'In[*]'이라고 뜨며, 실행이 완료되면 'In[숫자]'라고 뜬다.

- 주피터 노트북 설명

- 1) 현재 노트북을 저장한다.
- 2) 선택한 셀 다음에 새로운 셀을 생성한다.
- 3) 선택한 셀을 잘라낸다.
- 4) 선택한 셀을 실행한다.
- 5) 실행되고 있는 셀을 중지한다.
- 6) 실행되는 셀을 중지하고 모든 메모리를 정리한다.



● 2. 인공지능 마스크 감지 로봇

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
from example_base import GetParentPath  
# 핑퐁 로봇 제어 모듈 import.  
from pingpongthread import PingPongThread
```

● 2. 인공지능 마스크 감지 로봇

5.

- ▶ 큐브 1개 연결하는 인스턴스 생성하고, 로봇 제어 스레드 시작. 'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 처음 생성시 인터넷 연결 필요.

```
In [2]: # 1개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=1)  
# 로봇 제어 스레드 시작.  
PingPongThreadInstance.start()  
  
# 큐브 ID  
cube_ID = 1
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.  
Write data: DD DD 00 00 00 00 DA 00 0B 00 00  
Connected with a master robot.  
Fully connected.
```

● 2. 인공지능 마스크 감지 로봇

6.

- ▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

```
In [3]: # 웹캠 열기.  
PingPongThreadInstance.webcam_open()
```

● 2. 인공지능 마스크 감지로봇

7.

- ▶ 마스크를 쓰지 않은 사진을 찍어 저장한다. 저장 위치는 현재 폴더('pingpong-master\examples') 아래에 'chapter2\no_mask' 폴더에 저장된다. 키보드에서 'S'를 누르면 캡처가 되고, 'Q'를 누르면 캡처를 종료한다.
- ▶ 10~20개 정도 찍는다. 사진을 잘못 찍었으면 해당 폴더에 들어가 삭제하다

```
In [ ]: # chapter2\no_mask 폴더 안에 '마스크 안 쓴 사진' 찍기.  
PingPongThreadInstance.webcam_take_snapshots("chapter2/no_mask")
```

In [4]:



Q hit, closing...

*참고: 윈도우에서는 하위 폴더를 'w', 또는 '\'으로 표기 하며, 리눅스와 맥OS에서는 '/'로 표기 한다. 하지만 윈도우에서 입력할 때 '/'라고 해도 하위 폴더라고 인식하기 때문에, '/'로 표기하는 것이 좋다.

● 2. 인공지능 마스크 감지로봇

8.

- ▶ 마스크를 쓴 사진을 'chapter2\with_mask' 폴더에 저장한다. 방식은 이전과 같다.

```
In [5]: # chapter2\with_mask 폴더 안에 '마스크 쓴 사진' 찍기.  
PingPongThreadInstance.webcam_take_snapshots("chapter2\with_mask")
```



Q hit, closing...

● 2. 인공지능 마스크 감지로봇

9.

- ▶ 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 생성한다. 여기서는 '마스크를 안 쓴 사진'과 '마스크를 쓴 사진'의 묶음이다.
- ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다. 예를 들어, 'no_mask' 인스턴스의 첫 번째 인수는 "no_mask"이므로 이미지 클래스의 이름은 'no_mask'이고, 두 번째 인수는 "chapter2/no_mask"이므로 'chapter2\no_mask' 폴더의 사진들을 모두 불러온다.

```
In [6]: # '마스크 안 쓴 사진' 클래스 인스턴스, "마스크 쓴 사진" 클래스 인스턴스 생성.  
no_mask = PingPongThread.ImageClass("no_mask", "chapter2/no_mask")  
with_mask = PingPongThread.ImageClass("with_mask", "chapter2/with_mask")
```

● 2. 인공지능 마스크 감지로봇

10.

- ▶ 이미지 클래스를 학습시켜 학습 모델을 만든다. 모델은 첫 번째 인수인 'chapter2\\mask_model.json'으로 저장된다. 두 번째 인수부터는 이미지 클래스의 인스턴스를 입력한다.
- ▶ 'get_classification_model' 함수를 이용하여 저장했던 학습 모델을 다시 불러올 수 있다.

```
In [7]: # 모델 트레이닝. 첫 번째 인수는 저장하는 이름.  
model = PingPongThreadInstance.train_classes("chapter2/mask_model.json", no_mask, with_mask)
```

```
Class no_mask : 20 images.  
Class with_mask : 20 images.  
Training done.  
chapter2//mask_model.json saved.
```

● 2. 인공지능 마스크 감지로봇

11.

- ▶ 센서 값을 받는다. 이제 센서 값을 0.5초 간격으로 주기적으로 받게 된다.

```
In [8]: # 센서 값 받기. 0.5초마다 한 번씩 수신.  
PingPongThreadInstance.receive_sensor_data(cube_ID, method="periodic", period=0.5)
```

Write data: FF FF FF 00 00 C8 B8 00 0B 32 01

● 2. 인공지능 마스크 감지 로봇

12.

- ▶ 근접 센서의 기본값을 받는다. 이 기본값은 근접 센서 앞에 아무것도 없을 때를 가정하여 사용하는 기준 값이므로, 이 함수를 실행할 때 근접 센서 앞에 아무것도 없도록 주의한다.

In [9]:

```
# 근접 센서의 디폴트 값 받기. 이 함수를 실행할 때는 근접 센서 앞에 아무것도 없도록 주의.  
proxy_default = PingPongThreadInstance.get_default_proxy(cube_ID)
```

● 2. 인공지능 마스크 감지 로봇

13.

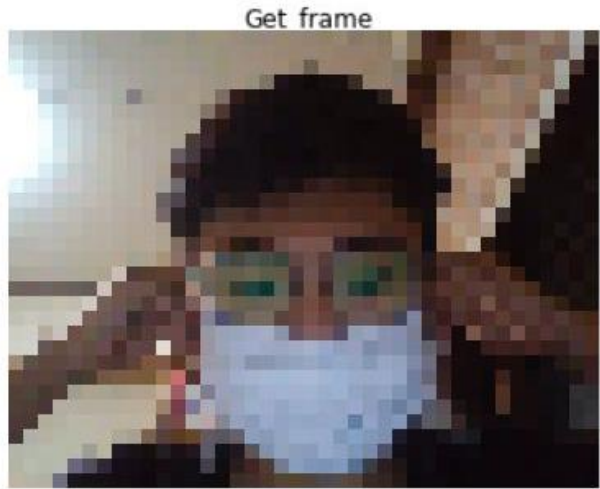
```
In [10]: # 시간 제어를 위한 모듈 import.
import time

# 근접 센서를 체크하는 루프.
while True:
    # 근접 센서 변화량.
    proxy_diff = abs(PingPongThreadInstance.get_current_proxy(cube_ID) - proxy_default)
    # 근접 센서 변화량이 10을 넘으면 3초 쉬고 마스크 검사.
    if 10 < proxy_diff:
        # 센서 데이터 그만 받기.
        PingPongThreadInstance.stop_sensor_data(cube_ID)
        print("마스크를 검사할게요~.")
        time.sleep(3)
        break
    else:
        print("장애물이 감지되지 않았어요.")
        time.sleep(0.1)
        # 출력 비우기.
        PingPongThreadInstance.clear_output()
```

● 2. 인공지능 마스크 감지 로봇

```
# 프레임을 평가하는 인스턴스 생성. 누적 프레임은 1초 동안 보관.
frames_predictor = PingPongThread.FramesPredictor(model=model, timer_sec=3)
# 웹캠을 이용하여 마스크 착용 여부를 예측하는 루프.
while True:
    # 주피터 노트북 출력 비우기.
    PingPongThreadInstance.clear_output()
    # 현재 웹캠 프레임 가져오기.
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")
    # 현재 프레임을 평가하고, 평가 내용을 누적.
    frames_prediction = frames_predictor.image_predict_and_accum(frame)
    print(frames_prediction)
    # 누적된 평가 내용.
    accum_prediction = frames_predictor.accum_predict()
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.
    if accum_prediction == None:
        max_class = None
        continue
    else:
        print("accum_prediction:", accum_prediction)
        # 가장 확률이 높은 클래스.
        max_class = max(accum_prediction, key=accum_prediction.get)
        # 마스크를 착용한 확률이 80% 이상이면 자동문 열림.
        if max_class == "with_mask" and accum_prediction[max_class] > 0.8:
            print("마스크를 잘 착용하고 왔네요~. 등교를 환영합니다!")
            # 15RPM, 시계 반대 방향으로 1/4바퀴(90도) 회전.
            PingPongThreadInstance.run_motor_step(cube_ID, 15, -0.25)
            # 4초 쉬.
            time.sleep(4)
            # 15RPM, 시계 방향으로 1/4바퀴(90도) 회전.
            PingPongThreadInstance.run_motor_step(cube_ID, 15, 0.25)
            break
        else:
            print("이런... 마스크를 깜빡 했나보군요... 마스크를 착용하고 등교해주세요!")
```

● 2. 인공지능 마스크 감지 로봇



{'with_mask': 1.0}

마스크를 잘 착용하고 왔네요~. 등교를 환영합니다!

Write data: FF FF FF 00 10 00 C1 00 13 02 01 00 02 FC 7C 00 00 01 F4

Write data: FF FF FF 00 10 00 C1 00 13 02 01 00 02 03 84 00 00 01 F4

* 참고: 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

● 2. 인공지능 마스크 감지 로봇

코드(알고리즘)설명

- 1) 근접 센서 앞에 장애물이 오면 3초를 쉬고 마스크 검사를 시작한다. 이는 최솟값을 기준으로 근접 센서의 변화량이 10 이상이 되면 장애물이 있다고 감지한다.
- 2) 이미지를 평가하고 평가를 누적하는 클래스('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 1초이다.
- 3) 웹캠을 이용해 마스크 착용 여부를 확인하는 루프에 들어간다.
- 4) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.

'image_predict_and_accum' 함수는 사진이 1초 동안 누적되지 않으면 평가가 진행되지 않고, None이 반환된다.
- 5) 1초 동안 누적된 "마스크를 착용했다고 판단된 평가"가 80%가 넘으면, 자동문이 열린다. 'run_motor_step' 함수로 스텝 모터를 15RPM의 시계 반대 방향으로 1/4바퀴 돌린다. 그리고 4초를 쉬고, 다시 스텝 모터를 15RPM의 시계 방향으로 1/4바퀴 돌린다.

● 2. 인공지능 마스크 감지 로봇

14.

- ▶ 센서 데이터 받는 것을 종료하고, 웹캠을 닫는다. 그리고 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 큐브 노란색 버튼을 2초 이상 눌러 전원을 끈다.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.
- ▶ 전원을 끄지 않으면 다음 연결 시 문제가 발생할 수 있다.

In [10]:

```
# 웹캠 닫기.  
PingPongThreadInstance.webcam_close()  
# 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()
```

Write data: FF FF FF FF 00 00 A8 00 0A 01

Disconnected with a master robot.

Reconnecting with serial...

Disconnect master robot.

End thread.

Found device: Silicon Labs CP210x USB to UART Bridge(COM5)

Write data: DD DD 00 00 00 00 DA 00 0B 00 00

Chapter 4

● 4. AI 감정 인식 로봇

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter4.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 4. AI 감정 인식 로봇

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
from example_base import GetParentPath  
# 핑퐁 로봇 제어 모듈 import.  
from pingpongthread import PingPongThread
```

● 4. AI 감정 인식 로봇

5.

- ▶ 큐브 1개 연결하는 인스턴스 생성하고, 로봇 제어 쓰레드 시작.
'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.
- ▶ 연결할 큐브 숫자를 number 파라미터에, 자신의 그룹 ID를 group_id 파라미터에 넣어 설정. 위는 4번 그룹 연결.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 생성시 인터넷 연결 필요.

```
In [2]: # 1개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=1, group_id=4)  
# 로봇 제어 쓰레드 시작.  
PingPongThreadInstance.start()  
# 모든 로봇이 연결될 때 까지 기다림.  
PingPongThreadInstance.wait_until_full_connect()  
  
# 큐브 ID  
cube_ID = 1
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.Write data: DD DD 04 DD 00 00 DA 00 0B 00 00
```

```
Connected with a master robot.  
Fully connected.
```

● 4. AI 감정 인식 로봇

6.

- ▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

In [4]:

```
# 웹캠 열기.  
PingPongThreadInstance.webcam_open()
```

● 4. AI 감정 인식 로봇

7.

- ▶ 기분 좋은 감정 사진, 기분 나쁜 감정 사진을 각각 찍어 저장한다. 키보드에서 's'키를 누르면 캡처가 되고, 'q'키를 누르면 캡처를 종료한다.
- ▶ 각 그룹에서 30개 정도 찍는다. 사진을 잘못 찍었으면 해당 폴더에 들어가 삭제한다.

```
In [ ]: # chapter4\good 폴더 안에 '기분 좋은 감정' 찍기.  
PingPongThreadInstance.webcam_take_snapshots("chapter4/good")
```

```
In [ ]: # chapter4\bad 폴더 안에 '기분 나쁜 감정' 찍기.  
PingPongThreadInstance.webcam_take_snapshots("chapter4/bad")
```

*참고: 윈도우에서는 하위 폴더를 'w', 또는 '\ '으로 표기 하며, 리눅스와 맥OS에서는 '/'로 표기 한다. 하지만 윈도우에서 입력할 때 '/'라고 해도 하위 폴더라고 인식하기 때문에, '/'로 표기하는 것이 좋다.

● 4. AI 감정 인식 로봇

▶ 기분 좋은 감정 사진 예시



• 기분 나쁜 감정 사진 예시



● 4. AI 감정 인식 로봇

8.

- ▶ 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 생성한다. 여기서는 '기분 좋은 감정' 사진과 '기분 나쁜 감정 사진'의 묶음이다.
- ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다.

```
In [5]: # '기분 좋은 감정' 클래스 인스턴스, '기분 나쁜 감정' 클래스 인스턴스 생성.  
good = PingPongThreadInstance.ImageClass("good", "chapter4/good")  
bad = PingPongThreadInstance.ImageClass("bad", "chapter4/bad")
```

● 4. AI 감정 인식 로봇

9.

- ▶ 이미지 클래스를 학습시켜 학습 모델을 만든다. 모델은 첫 번째 인수인 'chapter4\gb_model.json'으로 저장된다.
- ▶ 두 번째 인수는 KNN 알고리즘의 k 값이고, 여기서는 10으로 설정했다. 세 번째 인수는 모델 모드를 나타내며, 여기서는 2로 설정했다.
- ▶ 네 번째 인수 이후는 이미지 클래스의 인스턴스 입력.
- ▶ 'get_classification_model' 함수를 이용하여 저장했던 학습 모델을 다시 불러올 수 있다.

```
In [6]: # 모델 트레이닝.  
# 첫 번째 인수는 저장하는 이름, 두 번째 인수는 knn 알고리즘의 k 값, 세 번째 인수는 모델 모드  
model = PingPongThreadInstance.train_classes("chapter4/gb_model.json", 10, 2, good, bad)  
  
Class good : 33 images.  
Class bad : 31 images.  
Training done.  
chapter4/gb_model.json saved.
```

● 4. AI 감정 인식 로봇

* 참고

▶ 모델 모드는 특정 이미지 분류 작업마다 잘 동작하는 모드이다.

▶ 모델 모드 1: 2번 Chapter

▶ 모델 모드 2: 4, 5, 6, 8, 9, 10번 Chapter

Chapter 4에서는 KNN 알고리즘의 k 값을 10정도로 크게 설정해야 잘 작동한다.

● 4. AI 감정 인식 로봇

10.

```
In [7]: # 시간 제어를 위한 모듈 import,
import time
# 키보드 제어를 위한 모듈 import,
import keyboard

print("스페이스 키를 누르고 당신의 감정을 표현해 보세요!")
# 4초 후에 스페이스 바 키를 누르면 루프에서 나감.
time.sleep(4)
while True:
    if keyboard.is_pressed(" "):
        break
    else:
        time.sleep(0.01)

# now(현재 상태), flag(과거 상태) 초기화
now = 1
flag = 1

# 프레임을 평가하는 인스턴스 생성, 누적 프레임은 3초 동안 보관.
frames_predictor = PingPongThreadInstance.FramesPredictor(model=model, timer_sec=3)
# 감정 인식을 위한 루프.
while True:
    # q를 누르면 종료.
    if keyboard.is_pressed("q"):
        break
    # 주피터 노트북 출력 비우기.
    PingPongThreadInstance.clear_output()
    # 현재 웹캠 프레임을 보여주고 가져오기.
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")
    # 현재 프레임을 평가하고, 평가 내용을 누적.
    frames_prediction = frames_predictor.image_predict_and_accum(frame)
    print(frames_prediction)
    # 누적된 평가 내용.
    accum_prediction = frames_predictor.accum_predict()
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.
    if accum_prediction == None:
        max_class = None
        continue
    else:
        print("accum_prediction:", accum_prediction)
        max_class = max(accum_prediction, key=accum_prediction.get)
```

● 4. AI 감정 인식 로봇

```
# '기분 좋은 감정'일 확률이 90% 이상이면 승패 계산.  
if max_class == "good" and accum_prediction[max_class] > 0.9:  
    now = 1  
# '기분 나쁜 감정'일 확률이 90% 이상이면 승패 계산.  
elif max_class == "bad" and accum_prediction[max_class] > 0.9:  
    now = 2  
# 모두 아니면 돌아가기.  
else:  
    continue  
  
# now(현재 상태)와 flag(과거 상태)가 같으면 pass, 다르면 180도 회전  
if now == flag:  
    pass  
else:  
    angle = 180/360  
    PingPongThreadInstance.run_motor_step(cube_ID, 15, angle)  
    time.sleep(angle/15*60)  
    flag = now
```

• 결과:



```
{ 'good': 2, 'bad': 8}  
accum_prediction: { 'bad': 1.0}
```

● 4. AI 감정 인식 로봇

코드(알고리즘) 설명

- 1) "스페이스 키를 누르고 당신의 감정을 표현해 보세요!"를 출력하고 4초를 쉰다. 그리고 스페이스 키를 누를 때까지 루프를 돈다.
- 2) 스페이스 키를 누르면 루프에서 탈출한다. 그리고 now 변수와 flag 변수를 각각 1로 초기화시킨다.
- 3) 이미지를 평가하고 평가를 누적하는 클래스('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 3초이다.
- 4) 웹캠을 이용해 감정을 확인하는 루프에 들어간다.
- 5) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.

● 4. AI 감정 인식 로봇

6) 평가 내용이 없으면 돌아가고, 평가 내용이 있으면 확률이 가장 높은 클래스를 확인한다.

* 'image_predict_and_accum' 함수는 이미지가 누적된 지 3초 전까지는 평가를 진행하지 않는다.

7)

- ▶ 3초 동안 누적된 평가 중 "기분 좋은 감정이라고 판단된 평가"가 90%를 넘으면, now 변수를 1로 설정한다.
- ▶ 3초 동안 누적된 평가 중 "기분 나쁜 감정이라고 판단된 평가"가 90%를 넘으면, now 변수를 2로 설정한다.
- ▶ 3초 동안 누적된 평가 중 90%를 넘은 것이 없으면 다시 누적 평가한다.

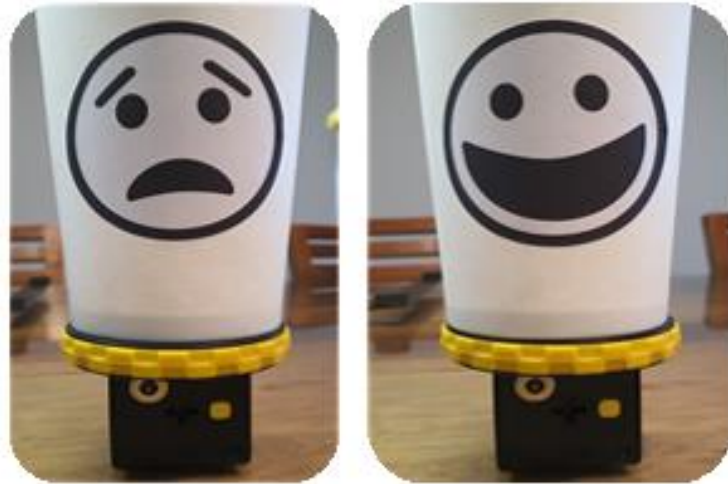
8) now 변수와 flag 변수의 값이 같으면 패스하고, 값이 다르면 180도 회전한다. 그리고 flag 변수를 now 변수와 같은 값으로 바꿔준다.

9) 루프 반복 → 'q' 키를 누르면 루프에서 빠져나온다.

● 4. AI 감정 인식 로봇

* 참고

- ▶ now 변수와 flag 변수는, now 변수가 1이 되면 웃는 얼굴이 앞쪽에 나오도록 하고, now 변수가 2가 되면 우는 얼굴이 앞쪽에 나오도록 하는 역할을 한다.



● 4. AI 감정 인식 로봇

11.

- ▶ 웹캠을 닫는다. 그리고 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 각 큐브 노란 버튼을 2초 이상 눌러 전원 OFF.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.

* 주의: 중간에 오류가 생겼으면, 반드시 이 버튼을 눌러준다. 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

In [10]:

```
# 웹캠 닫기.  
PingPongThreadInstance.webcam_close()  
# 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()
```

Write data: FF FF FF FF 00 00 A8 00 0A 01

Disconnected with a master robot.

Reconnecting with serial...

Disconnect master robot.

End thread.

Found device: Silicon Labs CP210x USB to UART Bridge(COM5)

Write data: DD DD 00 00 00 00 DA 00 0B 00 00

Chapter 5

● 5. 재활용 분리 수거 로봇

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter5.ipynb'를 클릭하여 연다.
또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 5. 재활용 분리 수거 로봇

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
from example_base import GetParentPath  
# 핑퐁 로봇 제어 모듈 import.  
from pingpongthread import PingPongThread
```

● 5. 재활용 분리 수거 로봇

5.

- ▶ 자신의 그룹 ID에 큐브 2개 연결하는 인스턴스 생성하고, 로봇제어 쓰레드 시작. 'Serial connected.' -> 각 큐브 노란 버튼 두 번씩 눌러 연결. 'Fully connected.' -> 큐브 연결 완료
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 생성시 인터넷 연결 필요.
- ▶ 큐브 ID 변수를 등록한다. 첫 번째로 연결한 로봇만 조작.

```
In [2]: # 2개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=2)  
# 로봇 제어 쓰레드 시작.  
PingPongThreadInstance.start()  
  
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.Write data: FF FF 00 FF 20 00 AD 00 0B 0A 00  
  
Connected with a master robot.  
Connected robots: 2  
Fully connected.
```

```
In [3]: # 큐브 ID  
cube_ID = 1
```

● 5. 재활용 분리 수거 로봇

6.

▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

```
In [4]: # 웹캠 열기.  
PingPongThreadInstance.webcam_open()
```

● 5. 재활용 분리 수거 로봇

7.

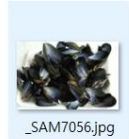
- ▶ 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 생성한다. 여기서는 '일반 쓰레기 사진'과 '재활용 쓰레기 사진'의 묶음이다.
- ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다.

In
[5]:

```
# '일반 쓰레기' 사진 클래스 인스턴스, '재활용 쓰레기' 클래스 인스턴스 생성.  
general = PingPongThreadInstance.ImageClass("general", "chapter5/general")  
recycle = PingPongThreadInstance.ImageClass("recycle", "chapter5/recycle")
```

● 5. 재활용 분리 수거 로봇

일반 쓰레기



_SAM7056.jpg



145A6D0C4A00FA4CC6.jpg



161D473F507ACFFB24.jpg



348_shop1_922409.jpg



46860183-호두-검찰-근접-촬영의-제비.jpg



70312685-흰색-배경에-치킨-뼈.jpg



87801554-깨진-거울.jpg



9867865325598.jpg



GettyImages-821157812.jpg



unnamed.jpg

재활용 쓰레기



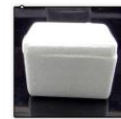
1a76e33bb6.jpg



161d6a28f6.jpg



352_shop1_653870.jpg



903_shop1_350718.jpg



1605311_1.jpg



1490249452814.png



1458174404756746.jpg



d941e6be6.jpg



다운로드.jpg



다운로드2.jpg

● 5. 재활용 분리 수거 로봇

8.

- ▶ 이미지 클래스를 학습시켜 학습 모델을 만든다. 모델은 첫 번째 인수인 'chapter5\\waste_model.json'으로 저장된다. 두 번째 인수부터는 이미지 클래스의 인스턴스를 입력한다.
- ▶ 'get_classification_model' 함수를 이용하여 저장했던 학습 모델을 다시 불러올 수 있다.

```
In [6]: # 모델 학습. 첫 번째 인수는 모델을 저장하는 이름.  
model = PingPongThreadInstance.train_classes("chapter5/waste_model.json", general, recycle)
```

```
Class general : 10 images.  
Class recycle : 10 images.  
Training done.  
chapter5//waste_model.json saved.
```

● 5. 재활용 분리 수거 로봇

9.

- ▶ 센서 값을 받는다. 이제 센서 값을 0.5초 간격으로 주기적으로 받게 된다.
- ▶ 그리고 근접 센서의 기본값을 받는다. 이 기본값은 근접 센서 앞에 아무것도 없을 때를 가정하여 사용하는 기준 값이므로, 이 함수를 실행할 때 근접 센서 앞에 아무것도 없도록 주의한다.

```
In [7]: # 센서 값 받기. 0.5초마다 한 번씩 수신.  
PingPongThreadInstance.receive_sensor_data(cube_ID, method="periodic", period=0.5)
```

Write data: FF FF FF 00 00 C8 B8 00 0B 32 01

```
In [8]: # 근접 센서의 디폴트 값 받기. 이 함수를 실행할 때는 근접 센서 앞에 아무것도 없도록 주의.  
proxy_default = PingPongThreadInstance.get_default_proxy(cube_ID)
```

● 5. 재활용 분리 수거 로봇

10.

```
In ]: # 시간 제어를 위한 모듈 import.
[9]: import time

# 근접 센서를 체크하는 루프.
while True:
    # 근접 센서 변화량.
    proxy_diff = abs(PingPongThreadInstance.get_current_proxy(cube_ID) - proxy_default)
    # 근접 센서 변화량이 10을 넘으면 쓰레기 검사.
    if 10 < proxy_diff:
        print("장애물이 감지되었습니다.")
        # 센서 데이터 그만 받기.
        PingPongThreadInstance.stop_sensor_data(cube_ID)
        print("안녕하세요? 생활에서 중요한 재활용 분리수거. 제가 도와드릴게요!")
        time.sleep(4)
        print("이미지를 보여주세요.")
        time.sleep(2)
        break
    else:
        print("장애물이 감지되지 않았어요.")
        time.sleep(0.1)
        # 출력 비우기.
        PingPongThreadInstance.clear_output()
# 프레임을 평가하는 인스턴스 생성. 누적 프레임은 5초 동안 보관.
frames_predictor = PingPongThreadInstance.FramesPredictor(model=model, timer_sec=3)
# KNN 알고리즘의 k를 1로 설정.
frames_predictor.set_knn_k(1)
# 웹캠을 이용하여 쓰레기 분류를 예측하는 루프.
```

* 참고: k를 1로 설정하는 이유는 뒤에 설명할 것이다.

● 5. 재활용 분리 수거 로봇

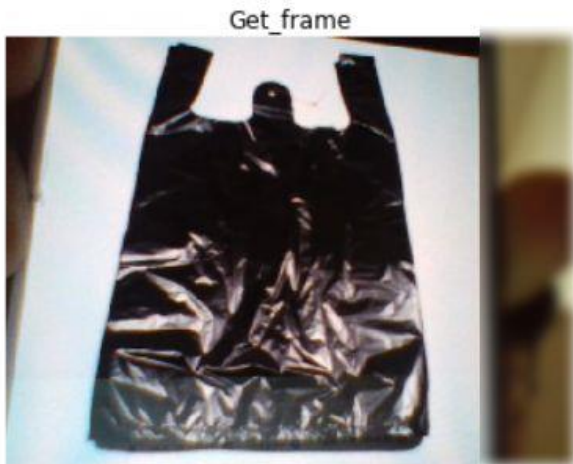
```
# 웹캠을 이용하여 쓰레기 분류를 예측하는 루프.
while True:
    # 주피터 노트북 출력 비우기.
    PingPongThreadInstance.clear_output()
    # 현재 웹캠 프레임을 보여주고 가져오기.
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")
    # 현재 프레임을 평가하고, 평가 내용을 누적.
    frame_prediction = frames_predictor.image_predict_and_accum(frame)
    print(frame_prediction)
    # 누적된 평가 내용.
    accum_prediction = frames_predictor.accum_predict()
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.
    if accum_prediction == None:
        max_class = None
        continue
    else:
        print("accum_prediction:", accum_prediction)
        # 가장 확률이 높은 클래스.
        max_class = max(accum_prediction, key=accum_prediction.get)
    # 일반 쓰레기일 확률이 80% 이상이면 왼쪽을 가리킴.
    if max_class == "general" and accum_prediction[max_class] > 0.9:
        print("일반쓰레기")
        # 5초 쉬.
        time.sleep(5)
        # 15RPM, 시계 반대 방향으로 1/4바퀴(90도) 회전.
        PingPongThreadInstance.run_motor_step(cube_ID, 15, -0.25)
        # 2초 쉬.
        time.sleep(2)
        # 15RPM, 시계 방향으로 1/4바퀴(90도) 회전.
        PingPongThreadInstance.run_motor_step(cube_ID, 15, 0.25)
        # 1초 쉬.
        time.sleep(1)
    # 재활용 쓰레기일 확률이 80% 이상이면 오른쪽을 가리킴.
    elif max_class == "recycle" and accum_prediction[max_class] > 0.9:
        print("재활용 쓰레기")
        # 5초 쉬.
        time.sleep(5)
        # 15RPM, 시계 방향으로 1/4바퀴(90도) 회전.
        PingPongThreadInstance.run_motor_step(cube_ID, 15, 0.25)
        # 2초 쉬.
        time.sleep(2)
        # 15RPM, 시계 반대 방향으로 1/4바퀴(90도) 회전.
        PingPongThreadInstance.run_motor_step(cube_ID, 15, -0.25)
        # 1초 쉬.
        time.sleep(1)
    else:
        continue
```

● 5. 재활용 분리 수거 로봇

```
# 입력 값 받기. 1이면 다시 이미지를 인식하고, 나머지면 종료.
text = "더 버릴 쓰레기가 있으신가요? (있으면 1 아니면 0): "
x = input(text)
if x == "1":
    print("이미지를 보여주세요!")
    # 누적된 프레임 비우기.
    frames_predictor.clear_accum()
    time.sleep(2)
    continue
else:
    print("분리수거를 생활화합시다!")
    break
```

● 5. 재활용 분리 수거 로봇

* 참고: 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.



```
{'general': 0, 'recycle': 1}
accum_prediction: {'recycle': 1.0}
Write data: FF FF FF AA 20 00 CD 00 20 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 03 84 00 00 01 F4
Aggregator set.
Write data: FF FF FF AA 20 00 CD 00 20 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 01 F4
Aggregator set.
더 버릴 쓰레기가 있으신가요? (있으면 1 아니면 0): 0
분리수거를 생활화합시다!
```

● 5. 재활용 분리 수거 로봇

- 1) 근접 센서 앞에 장애물이 오면 쓰레기 분류를 시작한다. 이는 최솟값을 기준으로 근접 센서의 변화량이 10 이상이 되면 장애물이 있다고 감지한다.
- 2) 이미지를 평가하고 평가를 누적하는 클래스('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 5초이다.
- 3) KNN 알고리즘의 k를 1로 설정해 준다.
- 4) 웹캠을 이용해 마스크 착용 여부를 확인하는 루프에 들어간다.
- 5) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.

● 5. 재활용 분리 수거 로봇

6) 평가 내용이 없으면 돌아가고, 평가 내용이 있으면 확률이 가장 높은 클래스를 확인한다.

'image_predict_and_accum' 함수는 이미지가 누적된 지 5초가 지나지 않으면 평가를 진행하지 않는다.

7) 5초 동안 누적된 "일반 쓰레기라고 판단된 평가"가 90%를 넘으면, 'run_motor_step' 함수로 스텝 모터를 돌려 왼쪽을 가리킨다. 5초 동안 누적된 "재활용 쓰레기라고 판단된 평가"가 90%를 넘으면, 'run_motor_step' 함수로 스텝 모터를 돌려 오른쪽을 가리킨다. 둘 다 아니면 다시 누적 평가한다.

8) 입력 값을 받아 '1'이면 누적된 프레임을 비우고 다시 이미지 프레임을 받아 평가를 시작한다. 나머지면 끝낸다.

● 5. 재활용 분리 수거 로봇

- * 참고: KNN 알고리즘의 k 를 1로 설정하는 이유?
- ▶ KNN 알고리즘은, 평가할 대상과 가장 가까운 학습 이미지 k 개를 뽑아, k 개 중에 가장 많이 속해 있는 클래스를 뽑는 것이다.
- ▶ 그런데 Chapter 5의 클래스에서는 같은 클래스 안에서 서로 비슷한 이미지가 없으므로, 1개의 가장 가까운 이미지만 골라내면 되는 것이다.
- ▶ PingPongThread 모듈의 k 의 디폴트 값은 5이다.

● 5. 재활용 분리 수거 로봇

11.

- ▶ 웹캠을 닫는다. 그리고 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 각 큐브 노란 버튼을 2초 이상 눌러 전원 OFF.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.
- ▶ 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

In [10]:

```
# 웹캠 닫기.  
PingPongThreadInstance.webcam_close()  
# 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()  
  
Write data: FF FF FF FF 00 00 A8 00 0A 01  
Disconnected with a master robot.  
Reconnecting with serial...  
Disconnect master robot.  
End thread.  
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Write data: DD DD 00 00 00 00 DA 00 0B 00 00
```

Chapter 6

● 6. 위조 지폐 판독기

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter6.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 6. 위조 지폐 판독기

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
        from example_base import GetParentPath  
        # 핑퐁 로봇 제어 모듈 import.  
        from pingpongthread import PingPongThread
```

● 6. 위조 지폐 판독기

5.

- ▶ 자신의 그룹 ID에 큐브 2개 연결하는 인스턴스 생성하고, 로봇제어 쓰레드 시작. 'Serial connected.' -> 각 큐브 노란 버튼 두 번씩 눌러 연결. 'Fully connected.' -> 큐브 연결 완료
- ▶ 연결할 큐브의 수를 number 파라미터에, 자신의 그룹 ID를 group_id 에 설정. 위의 예시에서는 4번 그룹 연결 상황.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 생성시 인터넷 연결 필요.

```
In [2]: # 2개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=2, group_id=4)  
# 로봇 제어 쓰레드 시작.  
PingPongThreadInstance.start()  
# 모든 로봇이 연결될 때 까지 기다림.  
PingPongThreadInstance.wait_until_full_connect()
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.  
Write data: FF FF 04 FF 2D 00 AD 00 0B 1A 04  
Connected with a master robot.  
Connected robots: 2  
Fully connected.
```

● 6. 위조 지폐 판독기

6.

▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

```
In [4]: # 웹캠 열기.  
PingPongThreadInstance.webcam_open()
```

● 6. 위조 지폐 판독기

7.

- ▶ 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 생성한다. 여기서는 '우리나라 지폐 사진'과 '위조 지폐 사진'의 묶음이다.
- ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다.

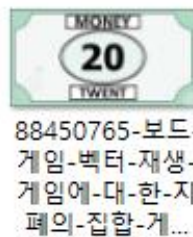
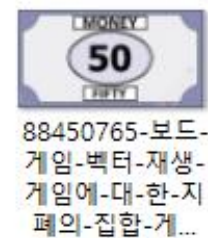
```
In [4]: # '우리나라 지폐' 사진 클래스 인스턴스, '위조 지폐' 클래스 인스턴스 생성.  
bill = PingPongThreadInstance.ImageClass("bill", "chapter6/bill")  
fake = PingPongThreadInstance.ImageClass("fake", "chapter6/fake")
```

● 6. 위조 지폐 판독기

우리나라 지폐



위조 지폐



● 6. 위조 지폐 판독기

8.

- ▶ 이미지 클래스를 학습시켜 학습 모델을 만든다. 모델은 첫 번째 인수인 'chapter4\wgb_model.json'으로 저장됨.
- ▶ 두 번째 인수는 KNN 알고리즘의 k 값이고, 여기서는 1로 설정했다. 세 번째 인수는 모델 모드를 나타내며, 여기서는 2로 설정했다.
- ▶ 네 번째 인수 이후는 이미지 클래스 인스턴스를 입력한다.
- ▶ 'get_classification_model' 함수를 이용하여 저장했던 학습 모델을 다시 불러올 수 있다.

```
In [5]: # 모델 학습.  
# 첫 번째 인수는 저장하는 이름, 두 번째 인수는 knn 알고리즘의 k 값, 세 번째 인수는 모델 모드.  
model = PingPongThreadInstance.train_classes("chapter6/bill_model.json", 1, 2, bill, fake)  
  
Class bill : 8 images.  
Class fake : 8 images.  
Training done.  
chapter6/bill_model.json saved.
```

● 6. 위조 지폐 판독기

* 참고

- ▶ 모델 모드는 특정 이미지 분류 작업마다 잘 동작하는 모드이다.
 - ▶ 모델 모드 1: 2번 Chapter
 - ▶ 모델 모드 2: 4, 5, 6, 8, 9, 10번 Chapter
- ▶ KNN 알고리즘은, 평가할 대상과 가장 가까운 학습 이미지 k개를 뽑아, k개 중에 가장 많이 속해 있는 클래스를 뽑는 것이다. 그런데 Chapter 6의 클래스에서는 같은 클래스 안에서 서로 비슷한 이미지가 없으므로, 1개의 가장 가까운 이미지만 골라내면 되는 것이다.

● 6. 위조 지폐 판독기

9.

- ▶ 몇 가지 대사를 출력하고, 스페이스 바 키를 누를 때까지 기다리는 루프를 생성한다.
- ▶ 스페이스 바 키가 눌리면 루프를 탈출하고 다시 대사를 출력한다.

```
In [6]: # 시간 제어를 위한 모듈 import,  
import time  
# 키보드 제어를 위한 모듈 import,  
import keyboard  
  
# 대사 출력.  
print("여기에 지폐를 인식시키면")  
time.sleep(1)  
print("위조지폐 여부를 알 수 있대")  
time.sleep(1)  
# '스페이스 바' 키를 누르면 진행.  
while True:  
    if keyboard.is_pressed(" "):  
        break  
    else:  
        time.sleep(0.01)  
time.sleep(4)  
print("재미있겠다~ 빨리 해보자!")  
time.sleep(2)
```

여기에 지폐를 인식시키면
위조지폐 여부를 알 수 있대
재미있겠다~ 빨리 해보자!

● 6. 위조 지폐 판독기

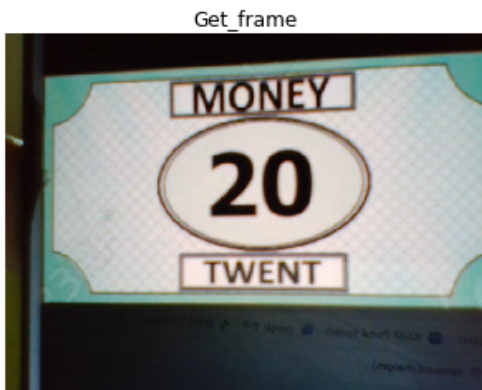
```
10. In [9]: # 대사 읽어주기.  
PingPongThreadInstance.tts_ko("지폐를 인식시켜 주세요", True)  
time.sleep(2)  
  
# 프레임을 평가하는 인스턴스 생성. 누적 프레임은 3초 동안 보관.  
frames_predictor = PingPongThreadInstance.FramesPredictor(model=model, timer_sec=3)  
# 웹캠을 이용하여 지폐를 예측하는 루프.  
while True:  
    # 주피터 노트북 출력 비우기.  
    PingPongThreadInstance.clear_output()  
    # 현재 웹캠 프레임을 보여주고 가져오기.  
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")  
    # 현재 프레임을 평가하고, 평가 내용을 누적.  
    frame_prediction = frames_predictor.image_predict_and_accum(frame)  
    print(frame_prediction)  
    # 누적된 평가 내용.  
    accum_prediction = frames_predictor.accum_predict()  
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.  
    if accum_prediction == None:  
        max_class = None  
        continue  
    else:  
        print("accum_prediction:", accum_prediction)  
        # 가장 확률이 높은 클래스.  
        max_class = max(accum_prediction, key=accum_prediction.get)
```

● 6. 위조 지폐 판독기

```
# '우리나라 지폐'일 확률이 90% 이상일 때,
if max_class == "bill" and accum_prediction[max_class] > 0.9:
    # 클래스 읽어주기.
    PingPongThreadInstance.tts_ko("우리나라 지폐", True)
    # 50도만큼 1번 큐브를 반시계 방향으로 회전, 2번 큐브를 시계 방향으로 회전.
    angle = 50/360
    PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, angle])
    time.sleep(angle/15*60)
    # 50도만큼 1번 큐브를 시계 방향으로 회전, 2번 큐브를 반시계 방향으로 회전.
    PingPongThreadInstance.run_motor_step([1, 2], 15, [angle, -angle])
    time.sleep(angle/15*60)
    break
# '위조 지폐'일 확률이 90% 이상일 때,
elif max_class == "fake" and accum_prediction[max_class] > 0.9:
    # 반박자로 '미, 도, 미, 도' 연주.
    PingPongThreadInstance.play_music(1, ["mi", "do", "mi", "do"], ["half"]*4)
    # 클래스 읽어주기.
    PingPongThreadInstance.tts_ko("위조 지폐", True)
    PingPongThreadInstance.tts_ko("위조 지폐", True)
    # 45도만큼 1번 큐브를 시계 방향으로 회전, 2번 큐브를 시계 방향으로 회전.
    angle = 45/360
    PingPongThreadInstance.run_motor_step([1, 2], 15, [angle, angle])
    time.sleep(angle/15*60)
    # 45도만큼 1번 큐브를 반시계 방향으로 회전, 2번 큐브를 반시계 방향으로 회전.
    PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, -angle])
    time.sleep(angle/15*60)
    # 45도만큼 1번 큐브를 반시계 방향으로 회전, 2번 큐브를 반시계 방향으로 회전.
    PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, -angle])
    time.sleep(angle/15*60)
    # 45도만큼 1번 큐브를 시계 방향으로 회전, 2번 큐브를 시계 방향으로 회전.
    PingPongThreadInstance.run_motor_step([1, 2], 15, [angle, angle])
    time.sleep(angle/15*60)
    break
# 모두 아니면 돌아가기.
else:
    continue
```

● 6. 위조 지폐 판독기

결과:



```
{'bill': 0, 'fake': 1}
accum_prediction: {'fake': 1.0}
Write data: FF FF 04 00 00 A1 E8 00 17 00 00 2C 3C 00 28 3C 00 2C 3C 00 28 3C 00
위조 지폐
위조 지폐
Write data: FF FF 04 AA 20 00 CD 00 33 02 01 00 00 FF FF 04 00 20 00 C1 00 13 02 01 00 02 03 84 00 00 00 FA FF FF 04 01 20 00 C1 00 13 02
01 00 02 03 84 00 00 00 FA
Timeout. Initialize buffer.
Timeout buffer: DE 5D 04 00 00 A1 E8 00 17 00 00
Aggregator set.
Write data: FF FF 04 AA 20 00 CD 00 33 02 01 00 00 FF FF 04 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 00 FA FF FF 04 01 20 00 C1 00 13 02
01 00 02 FC 7C 00 00 00 FA
Aggregator set.
Write data: FF FF 04 AA 20 00 CD 00 33 02 01 00 00 FF FF 04 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 00 FA FF FF 04 01 20 00 C1 00 13 02
01 00 02 FC 7C 00 00 00 FA
Aggregator set.
Write data: FF FF 04 AA 20 00 CD 00 33 02 01 00 00 FF FF 04 00 20 00 C1 00 13 02 01 00 02 03 84 00 00 00 FA FF FF 04 01 20 00 C1 00 13 02
01 00 02 03 84 00 00 00 FA
Aggregator set.
```

● 6. 위조 지폐 판독기

코드(알고리즘) 설명

- 1) "지폐를 인식시켜 주세요"를 TTS로 말하고 2초를 쉰다.
- 2) 이미지를 평가하고 평가를 누적하는 클래스('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 3초이다.
- 4) 웹캠을 이용해 지폐를 확인하는 루프에 들어간다.
- 5) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.
- 6) 평가 내용이 없으면 돌아가고, 평가 내용이 있으면 확률이 가장 높은 클래스를 확인한다.

* 'image_predict_and_accum' 함수는 이미지가 누적된 지 3초가 지나지 않으면 평가를 진행하지 않는다.

● 6. 위조 지폐 판독기

7)

- ▶ 3초 동안 누적된 평가 중 "우리나라 지폐로 판단된 평가"가 90%를 넘으면, "우리나라 지폐"를 TTS로 말하고, 큐브들을 특정 각도로 회전시키는 몇 가지 작업을 수행한다. 그리고 루프를 빠져나온다.
- ▶ 3초 동안 누적된 평가 중 "위조 지폐로 판단된 평가"가 90%를 넘으면, "위조 지폐"를 TTS로 두 번 말하고, 1번 큐브가 경고음(반박자로 "미도미도")를 재생한다. 그리고 큐브들을 특정 각도로 회전시키는 몇 가지 작업을 수행하고 루프를 빠져나온다.
- ▶ 3초 동안 누적된 평가 중 90%를 넘은 것이 없으면 다시 누적 평가한다.

● 6. 위조 지폐 판독기

* 참고

- ▶ 'tts_ko' 함수를 실행할 때는 반드시 인터넷에 연결이 되어 있어야 한다.
- ▶ 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

● 6. 위조 지폐 판독기

* 참고

▶ 'play_music'으로 연주할 수 있는 음과 박자는 아래 내용과 같다.

▶ 음높이:

▶ A3, A#3(= Bb3), B3

▶ C4, C#4(= Db4), D4, D#4(= Eb4), E4, F4, F#4, Gb4, G4, G#4(= Ab4), A4, A#4(= Bb4), B4

▶ C5, C#5(= Db5), D5, D#5(= Eb5), E5, F5, F#5(= Gb5), G5, G#5(= Ab5), A5, A#5(= Bb5), B5, C6

▶ do(= C4), re(= D4), mi(= E4), fa(= F4), sol(= G4), la(= A4), ti(= B4)

▶ 박자:

▶ metronome: 60, 100, 150

▶ tempo: whole, half, dotted half, quarter, dotted quarter, eighth, sixteenth

● 6. 위조 지폐 판독기

11.

- ▶ 웹캠을 닫는다. 그리고 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 각 큐브 노란 버튼을 2초 이상 눌러 전원 OFF.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.

* 주의: 중간에 오류가 생겼으면, 반드시 이 버튼을 눌러준다. 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

```
In [10]: # 웹캠 닫기.  
PingPongThreadInstance.webcam_close()  
# 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()  
  
Write data: FF FF FF FF 00 00 A8 00 0A 01  
Disconnected with a master robot.  
Reconnecting with serial...  
Disconnect master robot.  
End thread.  
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Write data: DD DD 00 00 00 00 DA 00 0B 00 00
```

Chapter 8

● 8. 우리 지역 특산물을 알려줘!

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter8.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 8. 우리 지역 특산물을 알려줘!

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
from example_base import GetParentPath  
# 핑퐁 로봇 제어 모듈 import.  
from pingpongthread import PingPongThread
```

● 8. 우리 지역 특산물을 알려줘!

5.

- ▶ 자신의 그룹 ID에 큐브 2개 연결하는 인스턴스 생성하고, 로봇제어 쓰레드 시작. 'Serial connected.' -> 각 큐브 노란 버튼 두 번씩 눌러 연결. 'Fully connected.' -> 큐브 연결 완료
- ▶ 연결할 큐브 수를 number 파라미터에, 자신의 그룹 ID를 group_id 파라미터에 넣어 설정. 위 예시는 4번 그룹 연결 예.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 처음 생성시 인터넷 연결 필요.

```
In [2]: # 2개 로봇을 연결하는 인스턴스 생성.
PingPongThreadInstance = PingPongThread(number=2, group_id=4)
# 로봇 제어 쓰레드 시작.
PingPongThreadInstance.start()
# 모든 로봇이 연결될 때 까지 기다림.
PingPongThreadInstance.wait_until_full_connect()
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)
Serial connected.Write data: FF FF 04 FF 20 00 AD 00 0B 1A 04
```

```
Connected with a master robot.
Connected robots: 2
Fully connected.
```

● 8. 우리 지역 특산물을 알려줘!

6.

▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

```
In [4]: # 웹캠 열기.  
PingPongThreadInstance.webcam_open()
```

● 8. 우리 지역 특산물을 알려줘!

7.

- ▶ 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 생성한다. 예시는 '강원도 특산물 사진'과 '충청남도 특산물 사진'의 묶음이다.
- ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다.

```
In [4]: # '강원도 특산물' 사진 클래스 인스턴스, '충청남도 특산물' 사진 클래스 인스턴스 생성.  
gangwondo = PingPongThreadInstance.ImageClass("gangwondo", "chapter8/gangwondo")  
chungcheongnamdo = PingPongThreadInstance.ImageClass("chungcheongnamdo", "chapter8/chungcheongnamdo")
```

● 8. 우리 지역 특산물을 알려줘!

강원도 특산물



충청남도 특산물



● 8. 우리 지역 특산물을 알려줘!

8.

- ▶ 이미지 클래스를 학습시켜 학습 모델을 만든다. 모델은 첫 번째 인수 'chapter4\specialties_model.json'로 저장됨.
- ▶ 두 번째 인수는 KNN 알고리즘의 k 값이고, 여기서는 1로 설정했다. 세 번째 인수는 모델 모드를 나타내며, 여기서는 2로 설정했다.
- ▶ 네 번째 인수 이후는 이미지 클래스의 인스턴스 입력.
- ▶ 'get_classification_model' 함수를 이용하여 저장했던 학습 모델을 다시 불러올 수 있다.

```
In [5]: # 모델 학습.  
# 첫 번째 인수는 저장하는 이름, 두 번째 인수는 knn 알고리즘의 k 값, 세 번째 인수는 모델 모드, 네 번째 인수 이후는 클래스 이름  
model = PingPongThreadInstance.train_classes("chapter8/specialties_model.json", 1, 2, gangwondo, chungcheongnamdo)  
  
Class gangwondo : 16 images.  
Class chungcheongnamdo : 16 images.  
Training done.  
chapter8/specialties_model.json saved.
```

● 8. 우리 지역 특산물을 알려줘!

* 참고

- ▶ 모델 모드는 특정 이미지 분류 작업마다 잘 동작하는 모드이다.
 - ▶ 모델 모드 1: 2번 Chapter
 - ▶ 모델 모드 2: 4, 5, 6, 8, 9, 10번 Chapter
- ▶ KNN 알고리즘은, 평가할 대상과 가장 가까운 학습 이미지 k개를 뽑아, k개 중에 가장 많이 속해 있는 클래스를 뽑는 것이다. 그런데 Chapter 8의 클래스에서는 같은 클래스 안에서 서로 비슷한 이미지가 없으므로, 1개의 가장 가까운 이미지만 골라내면 되는 것이다.

● 8. 우리 지역 특산물을 알려줘!

9.

- ▶ 몇 가지 대사를 출력하고, 스페이스 바 키를 누를 때까지 기다리는 루프를 생성한다.
- ▶ 스페이스 바 키가 눌리면 루프를 탈출한다.

```
In [6]: # 시간 제어를 위한 모듈 import,
import time
# 키보드 제어를 위한 모듈 import,
import keyboard

# 대사 출력,
print("어느 지역의 특산품이지??")
time.sleep(1)
# '스페이스 바' 키를 누르면 진행,
while True:
    if keyboard.is_pressed(" "):
        break
    else:
        time.sleep(0.01)
```

어느 지역의 특산품이지??

● 8. 우리 지역 특산물을 알려줘!

10.

- ▶ 강원도 특산물을 감지했을 때 실행시키는 'scene2' 함수를 정의한다.
- ▶ 'scene2' 함수는 "강원도 특산품입니다."를 TTS로 말하고, 로봇이 지도에서 충청남도를 가리켰다가 다시 원래대로 돌아오도록 큐브를 회전시킨다.

```
In [7]: # 'scene2' 함수 정의.  
def scene2():  
    # 대사 읽어주기.  
    PingPongThreadInstance.tts_ko("강원도 특산품입니다.", True)  
    # 20도만큼 1번 큐브를 반시계방향으로 회전, 이후 1초 쉬기.  
    angle = 20/360  
    PingPongThreadInstance.run_motor_step(1, 15, -angle)  
    time.sleep(angle/15*60+1)  
    # 120도만큼 2번 큐브를 반시계방향으로 회전, 이후 1초 쉬기.  
    angle = 120/360  
    PingPongThreadInstance.run_motor_step(2, 15, -angle)  
    time.sleep(angle/15*60+1)  
    # 120도만큼 2번 큐브를 시계방향으로 회전, 이후 1초 쉬기.  
    angle = 120/360  
    PingPongThreadInstance.run_motor_step(2, 15, angle)  
    time.sleep(angle/15*60+1)  
    # 20도만큼 1번 큐브를 시계방향으로 회전.  
    angle = 20/360  
    PingPongThreadInstance.run_motor_step(1, 15, angle)  
    time.sleep(angle/15*60)
```

● 8. 우리 지역 특산물을 알려줘!

* 참고

- ▶ 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

● 8. 우리 지역 특산물을 알려줘!

11.

- ▶ 충청남도 특산물을 감지했을 때 실행시키는 'scene2' 함수를 정의한다.
- ▶ 'scene2' 함수는 "충청남도 특산품입니다."를 TTS로 말하고, 로봇이 지도에서 충청남도를 가리켰다가 다시 원래대로 돌아오도록 큐브를 회전시킨다.

```
In [8]: # 'scene3' 함수 정의.  
def scene3():  
    # 대사 읽어주기.  
    PingPongThreadInstance.tts_ko("충청남도 특산품입니다.", True)  
    # 15도만큼 1번 큐브를 시계방향으로 회전. 이후 1초 쉬기.  
    angle = 15/360  
    PingPongThreadInstance.run_motor_step(1, 15, angle)  
    time.sleep(angle/15*60+1)  
    # 120도만큼 2번 큐브를 반시계방향으로 회전. 이후 1초 쉬기.  
    angle = 120/360  
    PingPongThreadInstance.run_motor_step(2, 15, -angle)  
    time.sleep(angle/15*60+2)  
    # 120도만큼 2번 큐브를 시계방향으로 회전. 이후 1초 쉬기.  
    angle = 120/360  
    PingPongThreadInstance.run_motor_step(2, 15, angle)  
    time.sleep(angle/15*60+1)  
    # 15도만큼 1번 큐브를 반시계방향으로 회전. 이후 1초 쉬기.  
    angle = 15/360  
    PingPongThreadInstance.run_motor_step(1, 15, -angle)  
    time.sleep(angle/15*60)
```

● 8. 우리 지역 특산물을 알려줘!

12.

In [10]:

```
# 대사 출력.
time.sleep(2)
print("내가 알려주지~")
time.sleep(1)
print("사진을 인식시켜 주면")
time.sleep(1)
print("해당 특산물 지역을 알려줄게")
time.sleep(1)

# 프레임을 평가하는 인스턴스 생성. 누적 프레임은 5초 동안 보관.
frames_predictor = PingPongThreadInstance.FramesPredictor(model=model, timer_sec=5)
# KNN 알고리즘의 k를 1로 설정.
frames_predictor.set_knn_k(1)
# 웹캠을 이용하여 특산물을 분류하는 루프.
while True:
    # 주피터 노트북 출력 비우기.
    PingPongThreadInstance.clear_output()
    # 현재 웹캠 프레임을 보여주고 가져오기.
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")
    # 현재 프레임을 평가하고, 평가 내용을 누적.
    frame_prediction = frames_predictor.image_predict_and_accum(frame)
    print(frame_prediction)
    # 누적된 평가 내용.
    accum_prediction = frames_predictor.accum_predict()
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.
    if accum_prediction == None:
        max_class = None
        continue
    else:
        print("accum_prediction:", accum_prediction)
        # 가장 확률이 높은 클래스.
        max_class = max(accum_prediction, key=accum_prediction.get)
```

● 8. 우리 지역 특산물을 알려줘!

```
if accum_prediction == None:
    max_class = None
    continue
else:
    print("accum_prediction:", accum_prediction)
    # 가장 확률이 높은 클래스.
    max_class = max(accum_prediction, key=accum_prediction.get)
    # '강원도 특산물'일 확률이 90% 이상일 때, scene2 함수를 실행.
    if max_class == "gangwondo" and accum_prediction[max_class] > 0.9:
        scene2()
        break
    # '충청남도 특산물'일 확률이 90% 이상일 때, scene3 함수를 실행.
    elif max_class == "chungcheongnamdo" and accum_prediction[max_class] > 0.9:
        scene3()
        break
    # 모두 아니면 돌아가기.
    else:
        continue
```


● 8. 우리 지역 특산물을 알려줘!

▶ 결과:



```
{'gangwondo': 1, 'chungcheongnamdo': 0}
accum_prediction: {'chungcheongnamdo': 0.07407407407407407, 'gangwondo': 0.9259259259259259}
강원도 특산품입니다.
Write data: FF FF 04 AA 20 00 CD 00 20 02 01 00 00 FF FF 04 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 00 6F
Aggregator set.
Write data: FF FF 04 AA 20 00 CD 00 20 02 01 00 00 FF FF 04 01 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 02 9B
Aggregator set.
Write data: FF FF 04 AA 20 00 CD 00 20 02 01 00 00 FF FF 04 01 20 00 C1 00 13 02 01 00 02 03 84 00 00 02 9B
Aggregator set.
Write data: FF FF 04 AA 20 00 CD 00 20 02 01 00 00 FF FF 04 00 20 00 C1 00 13 02 01 00 02 03 84 00 00 00 6F
Aggregator set.
```

● 8. 우리 지역 특산물을 알려줘!

코드 (알고리즘) 설명

- 1) 몇 가지 대사를 출력한다.
- 2) 이미지를 평가하고 평가를 누적하는 클래스('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 5초이다.
- 4) 웹캠을 이용해 특산물을 확인하는 루프에 들어간다.
- 5) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.
- 6) 평가 내용이 없으면 돌아가고, 평가 내용이 있으면 확률이 가장 높은 클래스를 확인한다.

▶ 'image_predict_and_accum' 함수는 이미지가 누적된 지 5초가 지나지 않으면 평가를 진행하지 않는다.

● 8. 우리 지역 특산물을 알려줘!

7)

- ▶ 5초 동안 누적된 평가 중 "강원도 특산물로 판단된 평가"가 90%를 넘으면, 'scene2' 함수를 실행한다. 그리고 루프를 빠져나온다.
- ▶ 5초 동안 누적된 평가 중 "충청남도 특산물로 판단된 평가"가 90%를 넘으면, 'scene3' 함수를 실행한다. 그리고 루프를 빠져나온다.
- ▶ 5초 동안 누적된 평가 중 90%를 넘은 것이 없으면 다시 누적 평가한다.

● 8. 우리 지역 특산물을 알려줘!

* 참고

- ▶ 'tts_ko' 함수를 실행할 때는 반드시 인터넷에 연결이 되어 있어야 한다.

● 8. 우리 지역 특산물을 알려줘!

13.

- ▶ 웹캠을 닫는다. 그리고 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 각 큐브 노란 버튼을 2초 이상 눌러 전원 OFF.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.

* 주의: 중간에 오류가 생겼으면, 반드시 이 버튼을 눌러준다. 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

In [10]:

```
# 웹캠 닫기.  
PingPongThreadInstance.webcam_close()  
# 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()
```

Write data: FF FF FF FF 00 00 A8 00 0A 01

Disconnected with a master robot.

Reconnecting with serial...

Disconnect master robot.

End thread.

Found device: Silicon Labs CP210x USB to UART Bridge(COM5)

Write data: DD DD 00 00 00 00 DA 00 0B 00 00

Chapter 9

● 9. 가위바위보 로봇

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter9.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 9. 가위바위보 로봇

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
        from example_base import GetParentPath  
        # 핑퐁 로봇 제어 모듈 import.  
        from pingpongthread import PingPongThread
```


● 9. 가위바위보 로봇

5.

- ▶ 큐브 1개 연결하는 인스턴스 생성하고, 로봇 제어 스레드 시작. 'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 처음 생성시 인터넷 연결 필요.
- ▶ 큐브 ID 변수를 등록한다.

```
In [2]: # 1개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=1)  
# 로봇 제어 스레드 시작.  
PingPongThreadInstance.start()  
# 모든 로봇이 연결될 때 까지 기다림.  
PingPongThreadInstance.wait_until_full_connect()
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.  
Write data: DD DD 00 00 00 00 DA 00 0B 00 00  
Connected with a master robot.  
Fully connected.
```

```
In [3]: # 큐브 ID  
cube_ID = 1
```

● 9. 가위바위보 로봇

6.

- ▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

In [4]:

```
# 웹캠 열기.
```

```
PingPongThreadInstance.webcam_open()
```

● 9. 가위|바위|보 로봇

7.

In
[5]:

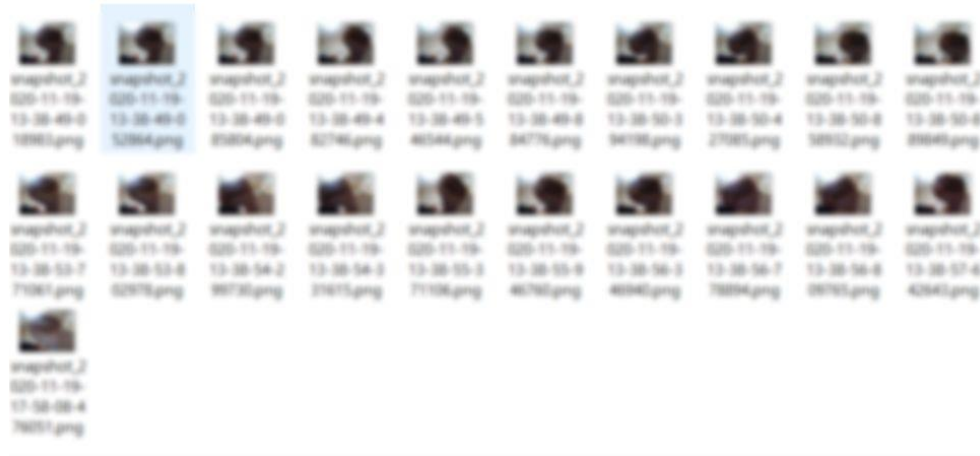
```
# chapter9\rock 폴더 안에 '바위 사진' 찍기.
PingPongThreadInstance.webcam_take_snapshots("chapter9/rock")
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)
Write data: DD DD 00 00 00 00 DA 00 0B 00 00
```

Taking snapshots

Q hit, closing...

캡처한 주먹 사진



● 9. 가위바위보 로봇

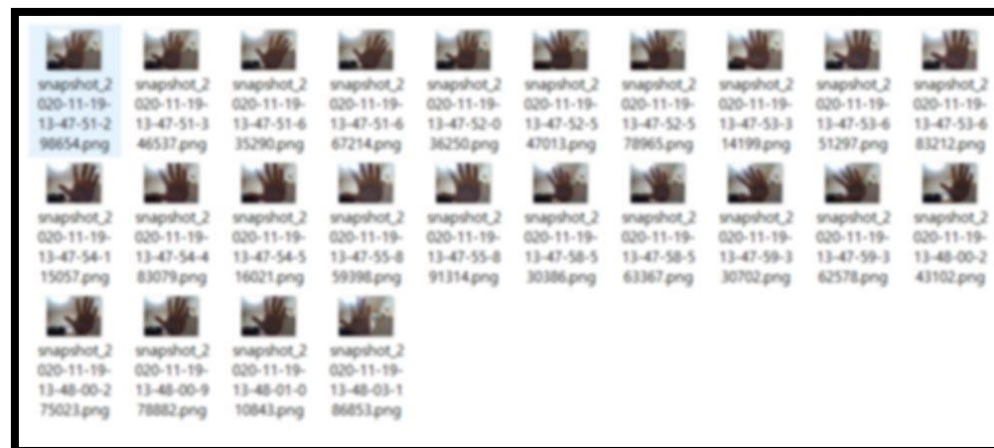
In
[6]: `# chapter9#paper 폴더 안에 '보 사진' 찍기.
PingPongThreadInstance.webcam_take_snapshots("chapter9/paper")`

Taking_snapshots



Q hit, closing...

캡처한 보 사진



● 9. 가위바위보 로봇

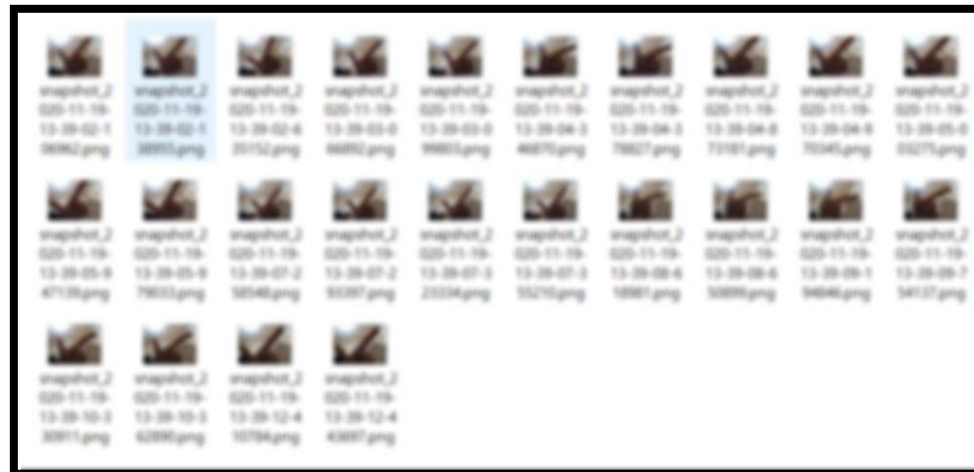
In
[7]: `# chapter9#scissors 폴더 안에 '가위 사진' 찍기.
PingPongThreadInstance.webcam_take_snapshots("chapter9/scissors")`

Taking snapshots



Q hit, closing...

캡처한 가위 사진



● 9. 가위바위보 로봇

- ▶ 주먹 사진, 보 사진, 가위 사진을 각각 찍어 저장한다. 키보드에서 'S'를 누르면 캡처가 되고, 'Q'를 누르면 캡처를 종료한다.
- ▶ 각 그룹에서 10~20개 정도 찍는다. 사진을 잘못 찍었으면 해당 폴더에 들어가 삭제한다.

● 9. 가위바위보 로봇

8.

- ▶ 바위, 보, 가위 각각의 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 생성한다.
 - ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다.
- ▶ 모델을 트레이닝 시킨다. 트레이닝한 모델을 저장시킨다.

```
In [8]: # '바위 사진' 클래스 인스턴스, '보' 클래스 인스턴스 생성, '가위' 클래스 인스턴스 생성.  
rock = PingPongThreadInstance.ImageClass("rock", "chapter9/rock")  
paper = PingPongThreadInstance.ImageClass("paper", "chapter9/paper")  
scissors = PingPongThreadInstance.ImageClass("scissors", "chapter9/scissors")
```

```
In [9]: # 모델 트레이닝. 첫 번째 인수는 저장하는 이름.  
model = PingPongThreadInstance.train_classes("chapter9/rps_model.json", rock, paper, scissors)  
  
Class rock : 21 images.  
Class paper : 24 images.  
Class scissors : 24 images.  
Training done.  
chapter9//_model.json saved.
```

● 9. 가위바위보 로봇

9.

In [10]:

```
# 시간 제어를 위한 모듈 import
import time
# 키보드 제어를 위한 모듈 import.
import keyboard
# 가위바위보 무작위 선택을 위한 모듈 import.
import random

print("스페이스 키를 누르면 가위바위보 게임을 시작합니다!")
time.sleep(4)
# 스페이스 바 키를 누르면 루프에서 나감.
while True:
    if keyboard.is_pressed(" "):
        break
    else:
        time.sleep(0.1)

print("가위바위보!")
# x는 로봇이 낸 손. 0, 1, 2 중 무작위로 하나를 고름. 0이면 바위, 1이면 가위, 2이면 보.
x = random.randint(0, 2)
# 로봇이 돌려야 할 각도 가져오기.
angle = [60/360, 180/360, 240/360][x]

# 프레임을 평가하는 인스턴스 생성. 누적 프레임은 3초 동안 보관.
frames_predictor = PingPongThreadInstance.FramesPredictor(model=model, timer_sec=3)
while True:
    # 주피터 노트북 출력 비우기.
    PingPongThreadInstance.clear_output()
    # 현재 웹캠 프레임을 보여주고 가져오기.
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")
    # 현재 프레임을 평가하고, 평가 내용을 누적.
    frames_prediction = frames_predictor.image_predict_and_accum(frame)
    print(frames_prediction)
    # 누적된 평가 내용.
    accum_prediction = frames_predictor.accum_predict()
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.
    if accum_prediction == None:
        max_class = None
        continue
    else:
        print("accum_prediction:", accum_prediction)
        # 가장 확률이 높은 클래스.
        max_class = max(accum_prediction, key=accum_prediction.get)
    # 바위일 확률이 90% 이상이면 승패 계산.
```


● 9. 가위바위보 로봇

```
# 바위일 확률이 90% 이상이면 승패 계산.
if max_class == "rock" and accum_prediction[max_class] > 0.9:
    print("인식된 손 모양:", max_class)
    # 로봇이 낸 손 쪽으로 모터를 돌림.
    PingPongThreadInstance.run_motor_step(cube_ID, 15, angle)
    time.sleep(angle/15*60)
    if x == 0:
        print("바위!")
        print("비겼다...")
    elif x == 1:
        print("가위!")
        print("졌다 ㅋㅋ")
    else:
        print("보!")
        print("이겼다~!")
    break
# 보일 확률이 90% 이상이면 승패 계산.
elif max_class == "paper" and accum_prediction[max_class] > 0.9:
    print("인식된 손 모양:", max_class)
    # 로봇이 낸 손 쪽으로 모터를 돌림.
    PingPongThreadInstance.run_motor_step(cube_ID, 15, angle)
    time.sleep(angle/15*60)
    if x == 0:
        print("바위!")
        print("졌다 ㅋㅋ")
    elif x == 1:
        print("가위!")
        print("이겼다~!")
    else:
        print("보!")
        print("비겼다...")
    break
```

● 9. 가위바위보 로봇

```
# 가위일 확률이 90% 이상이면 승패 계산.
elif max_class == "scissors" and accum_prediction[max_class] > 0.9:
    print("인식된 손 모양:", max_class)
    # 로봇이 낸 손 쪽으로 모터를 돌림.
    PingPongThreadInstance.run_motor_step(cube_ID, 15, angle)
    time.sleep(angle/15*60)
    if x == 0:
        print("바위!")
        print("이겼다~!")
    elif x == 1:
        print("가위!")
        print("비겼다...")
    else:
        print("보!")
        print("졌다 ㅋㅋ")
    break
# 모두 아니면 돌아가기.
else:
    continue
```

● 9. 가위바위보 로봇



{'rock': 0, 'paper': 5, 'scissors': 0}

accum_prediction: {'paper': 0.9285714285714286, 'scissors': 0.07142857142857142}

인식된 손 모양: paper

바위!

졌다 πππ

Write data: FF FF FF 00 10 00 C1 00 13 02 01 00 02 03 84 00 00 01 4D

● 9. 가위바위보 로봇

- 1) 스페이스 바 키를 누르면 루프에서 나가 가위바위보 게임을 시작한다.
- 2) 로봇이 0, 1, 2 중에서 무작위로 하나를 고른다. 0이면 바위, 1이면 가위, 2이면 보이다. 하나를 골랐으면 그것으로 부터 각도를 가져온다.
- 3) 이미지를 평가하고 평가를 누적하는 클래스('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 3초이다.
- 4) 웹캠을 이용해 마스크 착용 여부를 확인하는 루프에 들어간다.
- 5) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.

● 9. 가위바위보 로봇

6) 평가 내용이 없으면 돌아가고, 평가 내용이 있으면 확률이 가장 높은 클래스를 확인한다.

'image_predict_and_accum' 함수는 이미지가 누적된 지 3초가 지나지 않으면 평가를 진행하지 않는다.

7) 5초 동안 누적된 "바위라고 판단된 평가"가 90%를 넘으면, 'run_motor_step' 함수로 스텝 모터를 돌려 각도가 바위 쪽으로 가도록 만들고, 승패를 매겨 출력한다. 보, 가위도 같은 방식으로 작동한다. 위 3개에 모두 부합하지 않으면 다시 돌아가서 누적 평가한다.

* 참고: 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

● 9. 가위바위보 로봇

10.

- ▶ 로봇 제어 쓰레드를 종료한다.
- ▶ 셀이 종료되면 로봇의 노란색 버튼을 2초 이상 눌러 전원을 끈다.
- ▶ 메모리를 비우기 위해 Restart kernel 버튼을 누른다.
- ▶ 이 버튼을 누르지 않으면 다음 연결 시 문제가 발생할 수 있다.

```
In [11]: # 로봇 제어 쓰레드 종료.  
PingPongThreadInstance.end()
```

```
Write data: FF FF FF FF 00 00 A8 00 0A 01  
Disconnect master robot.  
Serial disconnected. Sleep 3 seconds.  
End thread.
```

Chapter 10

● 10. 주차장 차량번호 인식하기

1. 파워셸을 열어 'pingpong-master\examples' 디렉터리로 이동한다.
2. 파워셸에 'jupyter notebook'이라고 입력하여 주피터 노트북을 켜다.
3. 주피터 노트북에서 'ai_chapter10.ipynb'를 클릭하여 연다. 또는, New - Python 3를 눌러 새로운 노트북을 만들고, 다음 순서에 나오는 코드들을 순차적으로 입력하고 실행한다.

● 10. 주차장 차량번호 인식하기

4.

- ▶ 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.

```
In [1]: # 상위 폴더 경로 가져오기.  
        from example_base import GetParentPath  
        # 핑퐁 로봇 제어 모듈 import.  
        from pingpongthread import PingPongThread
```

● 10. 주차장 차량번호 인식하기

5.

- ▶ 큐브 1개 연결하는 인스턴스 생성하고, 로봇 제어 쓰레드 시작. 'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.
- ▶ 연결할 큐브의 수를 number 파라미터에, 자신의 그룹 ID를 group_id 파라미터에 넣어 설정. 위 예시는 4번 그룹에 연결한 예.
- ▶ 텐서플로우 라이브러리 로딩 시간 다소 소요됨.
- ▶ 텐서플로우 MobileNet 모델 처음 생성시 인터넷 연결 필요.

```
In [2]: # 2개 로봇을 연결하는 인스턴스 생성.  
PingPongThreadInstance = PingPongThread(number=2, group_id=4)  
# 로봇 제어 쓰레드 시작.  
PingPongThreadInstance.start()  
# 모든 로봇이 연결될 때 까지 기다림.  
PingPongThreadInstance.wait_until_full_connect()
```

```
Found device: Silicon Labs CP210x USB to UART Bridge(COM5)  
Serial connected.Write data: FF FF 00 FF 20 00 AD 00 0B 0A 00
```

```
Connected with a master robot.  
Connected robots: 2  
Fully connected.
```

● 10. 주차장 차량번호 인식하기

6.

▶ 웹캠을 제어할 수 있도록 웹캠을 연다.

```
In [4]: # 웹캠 열기.  
PingPongThreadInstance.webcam_open()
```

● 10. 주차장 차량번호 인식하기

7.

- ▶ 등록 차량 번호 사진, 미등록 차량 번호 사진 이미지 묶음에 대한 클래스('ImageClass') 인스턴스를 각각 생성한다.
- ▶ 인스턴스를 생성할 때, 첫 번째 인수는 이미지 클래스의 이름, 두 번째 인수는 이미지를 불러올 폴더를 의미한다.

```
In [4]: # '등록 차량' 사진 클래스 인스턴스, '미등록 차량' 사진 클래스 인스턴스 생성.  
registered = PingPongThreadInstance.ImageClass("registered", "chapter10/registered")  
unregistered = PingPongThreadInstance.ImageClass("unregistered", "chapter10/unregistered")
```

● 10. 주차장 차량번호 인식하기

등록 차량 번호 사진

00모 0505

Z00ah0505X.jpg

02우 9911

Z02dn9911X.jpg

03더 3214

Z03ej3214X.jpg

04마 6042

Z04ak6042X.jpg

04마 9782

Z04ak9782X.jpg

06여 7746

Z06dj7746X.jpg

06아 0487

Z06dk0487X.jpg

13부 9857

Z13an9857X.jpg

미등록 차량 번호 사진

서울 47
모 7152

A47ah7152X.jpg

경기 58
머 6731

B56aj6731X.jpg

인천 99
마 4050

C99ak4050X.jpg

강원 96
부 9915

D96an9915X.jpg

충남 52
오 8875

E52dh8875X.jpg

대전 32
여 6456

F32dj6456X.jpg

충북 29
아 5454

G29dk5454X.jpg

부산 83
우 0326

H83dn0326X.jpg

● 10. 주차장 차량번호 인식하기

8.

- ▶ 이미지 클래스를 학습시켜 학습 모델을 만든다. 모델은 첫 번째 인수 'chapter10\license_plate_model.json'로 저장.
- ▶ 두 번째 인수는 KNN 알고리즘의 k 값이고, 여기서는 1로 설정했다. 세 번째 인수는 모델 모드를 나타내며, 여기서는 2로 설정했다.
- ▶ 네 번째 인수 이후는 이미지 클래스의 인스턴스 입력.
- ▶ 'get_classification_model' 함수를 이용하여 저장했던 학습 모델을 다시 불러올 수 있다.

```
In [5]: # 모델 학습.  
# 첫 번째 인수는 저장하는 이름, 두 번째 인수는 knn 알고리즘의 k 값, 세 번째 인수는 모델 모드. 네 번째 인수 이후는  
model = PingPongThreadInstance.train_classes("chapter10/license_plate_model.json", 1, 2, registered, unregistered)  
  
Class registered : 8 images.  
Class unregistered : 8 images.  
Training done.  
chapter10/license_place_model.json saved.
```

● 10. 주차장 차량번호 인식하기

* 참고

- ▶ 모델 모드는 특정 이미지 분류 작업마다 잘 동작하는 모드이다.
 - ▶ 모델 모드 1: 2번 Chapter
 - ▶ 모델 모드 2: 4, 5, 6, 8, 9, 10번 Chapter
- ▶ KNN 알고리즘은, 평가할 대상과 가장 가까운 학습 이미지 k개를 뽑아, k개 중에 가장 많이 속해 있는 클래스를 뽑는 것이다. 그런데 Chapter 10은 한 개의 번호판이 클래스에 속해 있는 지를 확인하는 것이기 때문에, 1로 설정했다.

● 10. 주차장 차량번호 인식하기

9.

In [10]:

```
# 시간 제어를 위한 모듈 import.
import time
# 키보드 제어를 위한 모듈 import.
import keyboard

# 대사 출력.
print("안녕하세요? 저는 핑퐁주차로봇입니다.")
time.sleep(2)
print("번호가 등록된 차량만 등록하실 수 있습니다. 차량번호를 인식하겠습니다.")
time.sleep(2)

# 프레임을 평가하는 인스턴스 생성. 누적 프레임은 5초 동안 보관.
frames_predictor = PingPongThreadInstance.FramesPredictor(model=model, timer_sec=5)
# 웹캠을 이용하여 번호판을 인식하는 루프.
while True:
    # 주피터 노트북 출력 비우기.
    PingPongThreadInstance.clear_output()
    # 현재 웹캠 프레임을 보여주고 가져오기.
    frame = PingPongThreadInstance.webcam_get_frame(window="Get_frame")
    # 현재 프레임을 평가하고, 평가 내용을 누적.
    frame_prediction = frames_predictor.image_predict_and_accum(frame)
    print(frame_prediction)
    # 누적된 평가 내용.
    accum_prediction = frames_predictor.accum_predict()
    # 누적된 평가 내용이 없으면 돌아가기. 아니면 가장 확률이 높은 클래스 확인.
    if accum_prediction == None:
        max_class = None
        continue
    else:
        print("accum_prediction:", accum_prediction)
        # 가장 확률이 높은 클래스.
        max_class = max(accum_prediction, key=accum_prediction.get)
```


● 10. 주차장 차량번호 인식하기

```
# '등록 차량'일 확률이 90% 이상일 때, 주차바를 올리는 작업 진행.
if max_class == "registered" and accum_prediction[max_class] > 0.9:
    print("등록된 차량입니다. 주차바가 자동으로 올라갑니다.")
    time.sleep(2)
    # 90도만큼 시계방향으로 회전. 이후 2초 쉬기.
    angle = 90/360
    PingPongThreadInstance.run_motor_step(1, 15, angle)
    time.sleep(angle/15*60+2)
    # 90도만큼 반시계방향으로 회전.
    PingPongThreadInstance.run_motor_step(1, 15, -angle)
    time.sleep(angle/15*60)
    print("진입하십시오! 주차장에 오신 것을 환영합니다.")
    time.sleep(2)
    break
# '미등록 차량'일 확률이 90% 이상일 때, 경고음 재생.
elif max_class == "unregistered" and accum_prediction[max_class] > 0.9:
    # 1박자로 '도' 연주.
    PingPongThreadInstance.play_music(1, ["do"], ["whole"])
    print("죄송합니다. 이 번호는 등록되지 않은 차량이오니 등록 후 주차장을 이용하여 주시기 부탁드립니다.")
    break
# 모두 아니면 돌아가기.
else:
    continue
```

● 10. 주차장 차량번호 인식하기

▶ 결과:



{'registered': 0, 'unregistered': 1}

accum_prediction: {'unregistered': 1.0}

Write data: FF FF 04 00 00 A1 E8 00 0E 00 00 28 78 00

죄송합니다. 이 번호는 등록되지 않은 차량이오니 등록 후 주차장을 이용하여 주시기 부탁드립니다.

● 10. 주차장 차량번호 인식하기

코드(알고리즘) 설명

- 1) 몇 가지 대사를 출력한다.
- 2) 이미지를 평가하고 평가를 누적하는 클래스 ('FramesPredictor') 인스턴스를 만들어준다. 누적하는 시간은 5초이다.
- 4) 웹캠을 이용해 번호판 확인하는 루프에 들어간다.
- 5) 루프 안에서 웹캠의 프레임을 가져와 'image_predict_and_accum' 함수로 평가하고, 평가 내용을 누적한다. 그리고 누적된 평가 내용을 확인한다.

● 10. 주차장 차량번호 인식하기

6) 평가 내용이 없으면 돌아가고, 평가 내용이 있으면 확률이 가장 높은 클래스를 확인한다.

'image_predict_and_accum' 함수는 이미지가 누적된 지 5초가 지나지 않으면 평가를 진행하지 않는다.

7)

- ▶ 5초 동안 누적된 평가 중 "등록 차량 번호로 판단된 평가"가 90%를 넘으면, 몇 가지 대사를 출력하고, 주차바가 올라갔다가 내려가는 몇 가지 동작이 수행된다. 그리고 루프를 빠져나온다.
- ▶ 5초 동안 누적된 평가 중 "미등록 차량 번호로 판단된 평가"가 90%를 넘으면, 경고음을 재생하고 대사를 출력한다.
- ▶ 5초 동안 누적된 평가 중 90%를 넘은 것이 없으면 다시 누적 평가한다.

● 10. 주차장 차량번호 인식하기

* 참고

- ▶ 'run_motor_step' 함수에서, 입력하는 회전 값이 양수이면 시계 방향, 음수이면 반시계 방향으로 회전한다.

* 주의

- ▶ 'MobileNet' 알고리즘은 번호판의 글씨를 인식하는 것이 아니라, 번호판의 형태, 색 등을 인식하는 것이다. 따라서 비슷한 형태, 색의 번호판이 다른 클래스에 들어가 있다면 제대로 인식되지 않을 수 있다.

● 10. 주차장 차량번호 인식하기

* 참고

▶ 'play_music'으로 연주할 수 있는 음과 박자는 아래 내용과 같다.

▶ 음높이:

▶ A3, A#3(= Bb3), B3

▶ C4, C#4(= Db4), D4, D#4(= Eb4), E4, F4, F#4, Gb4, G4, G#4(= Ab4), A4, A#4(= Bb4), B4

▶ C5, C#5(= Db5), D5, D#5(= Eb5), E5, F5, F#5(= Gb5), G5, G#5(= Ab5), A5, A#5(= Bb5), B5, C6

▶ do(= C4), re(= D4), mi(= E4), fa(= F4), sol(= G4), la(= A4), ti(= B4)

▶ 박자:

▶ metronome: 60, 100, 150

▶ tempo: whole, half, dotted half, quarter, dotted quarter, eighth, sixteenth

● 10. 주차장 차량번호 인식하기

10.

- ▶ 웹캠을 닫는다. 그리고 로봇 제어 스레드를 종료한다.

In [10]:

```
# 웹캠 닫기.  
PingPongThreadInstance.webcam_close()  
# 로봇 제어 스레드 종료.  
PingPongThreadInstance.end()
```

Write data: FF FF FF FF 00 00 A8 00 0A 01

Disconnected with a master robot.

Reconnecting with serial...

Disconnect master robot.

End thread.

Found device: Silicon Labs CP210x USB to UART Bridge(COM5)

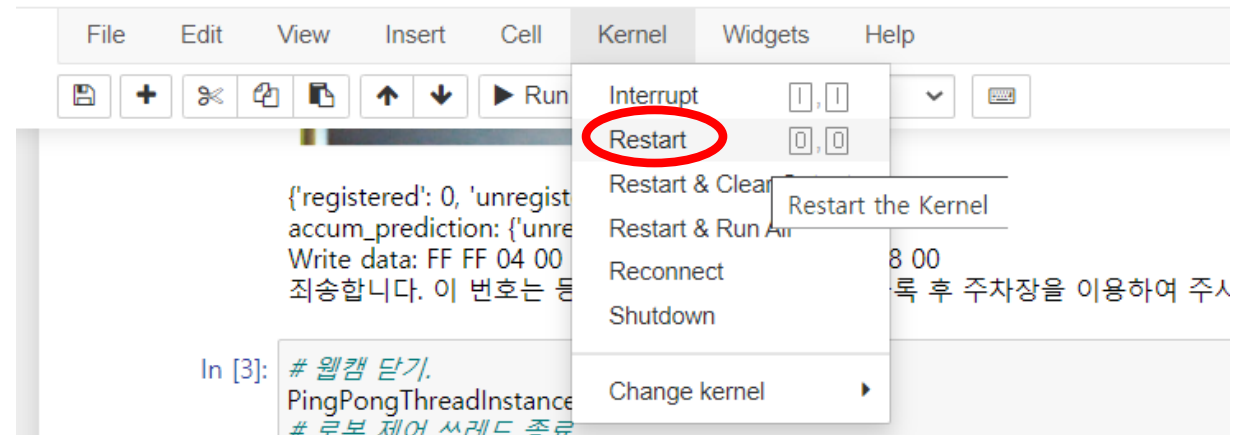
Write data: DD DD 00 00 00 00 DA 00 0B 00 00

● 10. 주차장 차량번호 인식하기

11.

- ▶ 주피터 노트북 커널을 재시작한다. 정의된 모든 변수를 초기화하고, 실행중인 쓰레드를 모두 종료한다.
- ▶ 오류가 난다면 메뉴바의 Kernel - Restart를 하면 제대로 작동한다.

```
In [ ]: # 커널 재시작.  
import os  
os._exit(00)
```



● 10. 주차장 차량번호 인식하기

12.

- ▶ 4.번에서 했던 것과 마찬가지로 상위 경로를 가져오고, 핑퐁 제어 클래스 모듈인 PingPongThread를 가져온다.
- ▶ 5.번과 같이 자신의 그룹 ID에 2개 큐브 연결하는 인스턴스 생성, 로봇 제어 쓰레드 시작. 'Serial connected.' → 큐브 노란 버튼 두 번 눌러 연결. 'Fully connected.' → 큐브 연결 완료.
- ▶ 연결할 큐브의 수를 number 파라미터에, 자신의 그룹 ID를 group_id 파라미터에 넣어 설정. 위 예시는 4번 그룹에 연결한 예.

```
In [1]: # 상위 폴더 경로 가져오기.
from example_base import GetParentPath
# 핑퐁 로봇 제어 모듈 import.
from pingpongthread import PingPongThread

# 2개 로봇을 연결하는 인스턴스 생성.
PingPongThreadInstance = PingPongThread(number=2) # n개 로봇 연결
PingPongThreadInstance.start() # 쓰레드 시작
PingPongThreadInstance.wait_until_full_connect() # 전부 연결될 때까지 기다림

Found device: Silicon Labs CP210x USB to UART Bridge(COM5)
Serial connected.
Write data: FF FF 00 FF 20 00 AD 00 0B 0A 00
Connected with a master robot.
Connected robots: 2
Fully connected.
```

● 10. 주차장 차량번호 인식하기

13.

```
In [2]: # 시간 제어를 위한 모듈 import.
import time
# 키보드 제어를 위한 모듈 import.
import keyboard

# 로봇을 조작하는 루프 생성.
angle = 90/360
while True:
    # '위' 키를 누르면 90도만큼 1번 큐브를 반시계 방향으로 회전, 2번 큐브를 시계 방향으로 회전.
    if keyboard.is_pressed("up"):
        PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, angle])
        time.sleep(angle/15*60)
    # '아래' 키를 누르면 90도만큼 1번 큐브를 시계 방향으로 회전, 2번 큐브를 반시계 방향으로 회전.
    elif keyboard.is_pressed("down"):
        PingPongThreadInstance.run_motor_step([1, 2], 15, [angle, -angle])
        time.sleep(angle/15*60)
    # '왼쪽' 키를 누르면 90도만큼 1번 큐브를 시계 방향으로 회전, 2번 큐브를 시계 방향으로 회전.
    elif keyboard.is_pressed("left"):
        PingPongThreadInstance.run_motor_step([1, 2], 15, [angle, angle])
        time.sleep(angle/15*60)
    # '오른쪽' 키를 누르면 90도만큼 1번 큐브를 반시계 방향으로 회전, 2번 큐브를 반시계 방향으로 회전.
    elif keyboard.is_pressed("right"):
        PingPongThreadInstance.run_motor_step([1, 2], 15, [-angle, -angle])
        time.sleep(angle/15*60)
    # 'q' 키를 누르면 종료.
    elif keyboard.is_pressed("q"):
        break
    else:
        time.sleep(0.1)
```

```
Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 01 F4 FF FF FF 01 20 00 C1 00 13 02 01 00 0
2 03 84 00 00 01 F4
Aggregator set.
Write data: FF FF FF AA 20 00 CD 00 33 02 01 00 00 FF FF FF 00 20 00 C1 00 13 02 01 00 02 FC 7C 00 00 01 F4 FF FF FF 01 20 00 C1 00 13 02 01 00 0
2 03 84 00 00 01 F4
```

● 10. 주차장 차량번호 인식하기

코드(알고리즘)설명

1) 오토카 로봇을 조작하는, 0.1초씩 쉬는 루프에 들어간다.

2)

- ▶ 키보드에서 '위' 키를 누르면 오토카가 앞으로 가도록 만든다.
- ▶ 키보드에서 '아래' 키를 누르면 오토카가 뒤로 가도록 만든다.
- ▶ 키보드에서 '왼쪽' 키를 누르면 오토카가 왼쪽으로 회전하도록 만든다.
- ▶ 키보드에서 '오른쪽' 키를 누르면 오른쪽으로 회전하도록 만든다.
- ▶ 키보드에서 'q' 키를 누르면 루프에서 나온다.



● 10. 주차장 차량번호 인식하기

14.

▶ 로봇 제어 스레드를 종료한다.

```
In [3]: # 로봇 제어 스레드 종료.  
PingPongThreadInstance.end()
```

```
Write data: FF FF FF FF 00 00 A8 00 0A 01  
Disconnect master robot.  
Serial disconnected. Sleep 3 seconds.  
End thread.
```

Thank you