

Tachy-RT API

가이드북

목차

1. 개요

1-1) Tachy-RT API란?

1-2) 주요 특징 및 장점

2. 요구사항

2-1) 시스템 요구사항

2-2) 지원 플랫폼 및 환경

3. 활용 예제 가이드

〈Tachy-RT API 실행 전 준비사항〉

3-1) 필요한 라이브러리 및 모듈 불러오기

3-2) Tachy-RT 환경 설정 준비

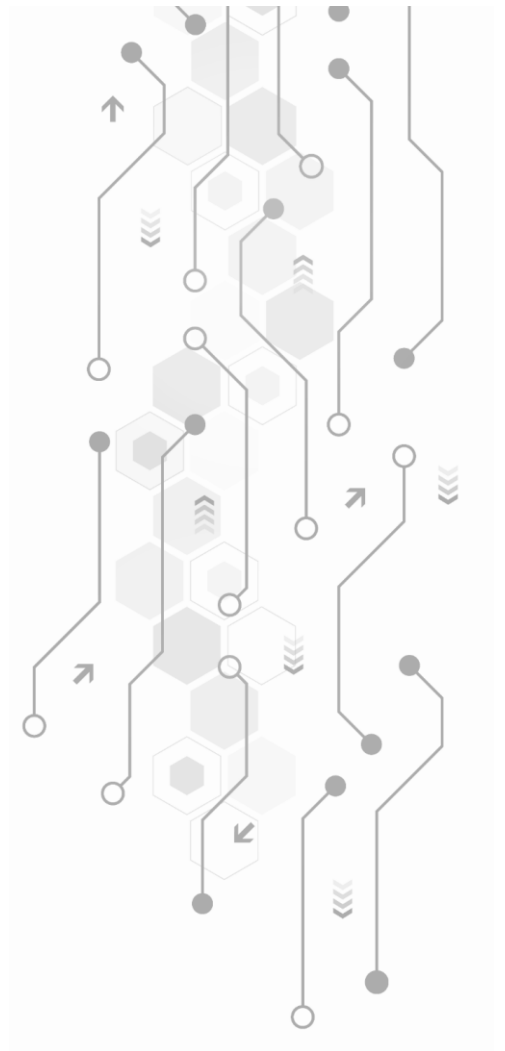
3-3) USB 장치 펌웨어 로딩

3-4) 모델 로딩

3-5) 입력 이미지 불러오기

3-6) 추론 실행

3-7) 후처리 수행

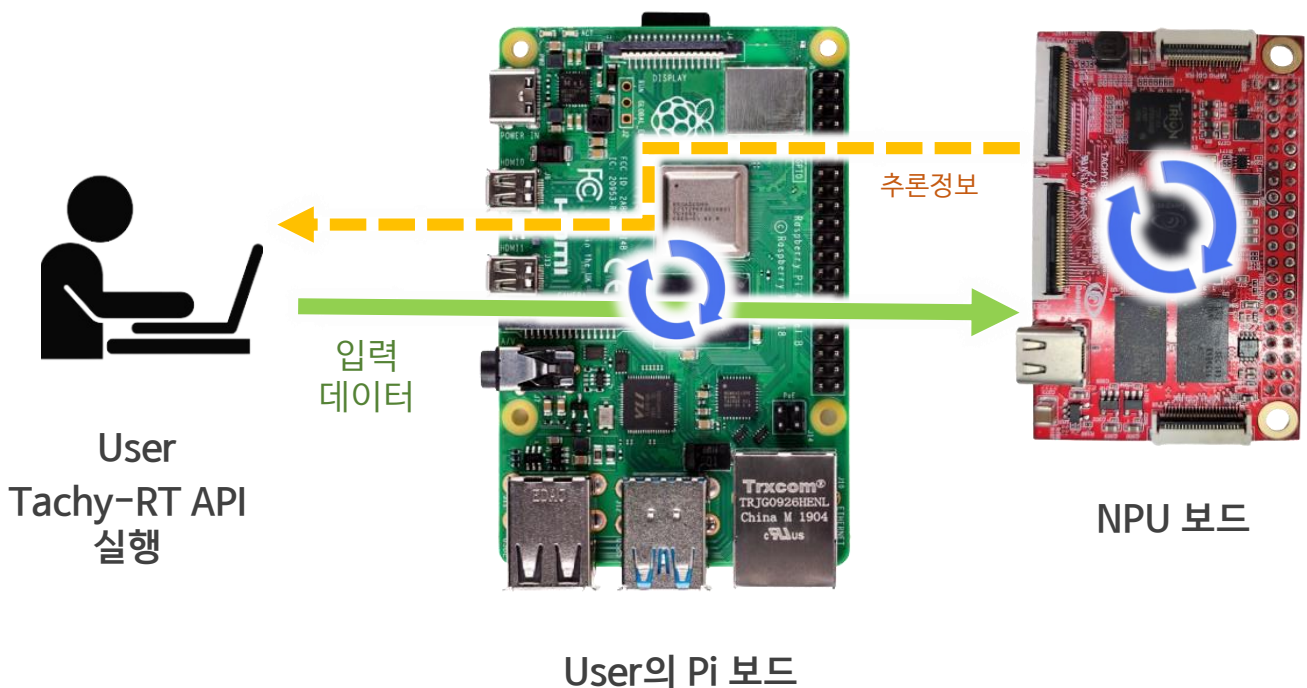


1. 개요

1-1) Tachy-RT API란?

Tachy-RT API는 Tachy Device에서 모델 추론, 센서 제어, 디버깅 등을 CLI 기반으로 제어할 수 있도록 설계된 런타임 도구입니다. Python 환경에서 동작하며, TCP/SPI/Local 등의 다양한 인터페이스를 통해 외부 시스템과 연동이 가능합니다.

〈시스템 사용 흐름도〉



1-2) 주요 특징 및 장점

- 다양한 인터페이스 지원
- 학습된 AI 모델의 실시간 추론
- 임베디드 환경에서 빠른 반응성과 효율적 리소스 사용

2. 요구사항

2-1) 시스템 요구사항

- CPU: Intel i5 이상 또는 등급 AMD
- RAM: 8GB 이상
- GPU (옵션): NVIDIA CUDA 11.x 이상
- OS: Ubuntu 20.04 / Windows 10 이상

2-2) 지원 플랫폼 및 환경

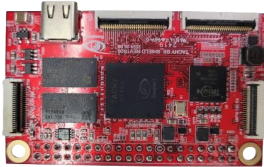
- Linux (Ubuntu 18.04/20.04)
- Windows 10/11 (64bit)
- Python 3.7 이상, C++ 지원

예제) Hello Image Detection

본 예제는 Tachy-RT를 이용하여 장치 설정 → 모델 로드 → 이미지 입력 → 추론 및 후처리 → 검출 결과 시각화까지, AI 기반 Object Detection 추론 전 과정을 다루는 실습 가이드입니다.

〈Tachy RT API 실행 전 준비사항〉

〈보드 준비〉

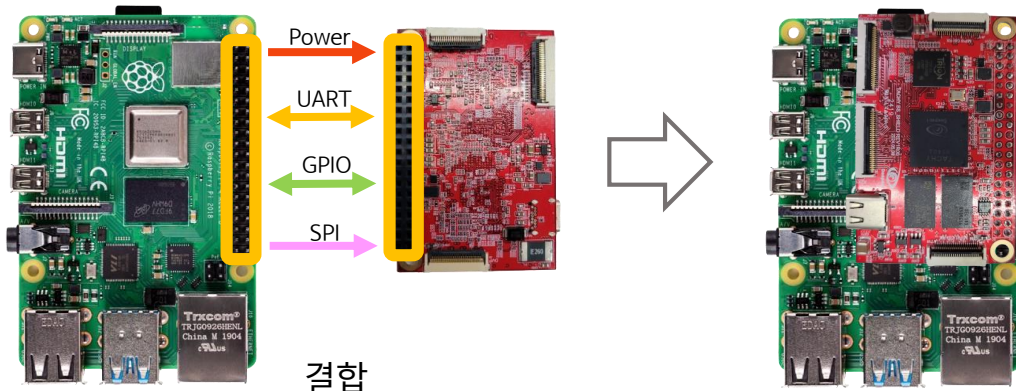


NPU 보드

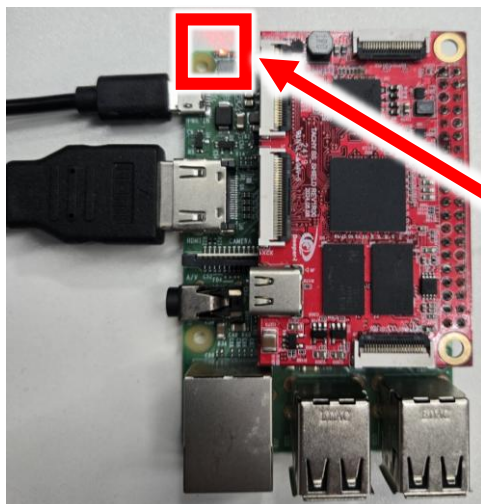


Pi 보드

〈NPU 보드, Pi 결합〉



〈이더넷 선연결〉



Cheak

붉은 LED 발광

→ 연결 ON

※ 본 사진은 이해를 돕기 위한 예시 사진입니다.

3-2) 필요한 라이브러리 및 모듈 불러오기

```
import sys
sys.path.append('.././post')
sys.path.append('.././utils')
import cv2
import argparse
import numpy as np
from yolov4 import Decoder
from functions import *
import tachy_rt.core.functions as rt_core
```

3-3) Tachy-RT 환경 설정 준비

```
config = W
{
  "INTERFACE": {
    # # On board
    # "mode": "local",

    # # On host (connected with ethernet)
    # "mode": "ethernet",
    # "ip": "192.168.3.121",

    # On host (connected with usb)
    "mode": "spi:ftdi"
  },
  "GLOBAL": {
    "bs_ver": 2,
    "std": 255.0,
    "mean": 0.0,
    "input_dump": True
  },
  "NETWORK": {
    "network_path":
    "./model/object_detection_yolov4-20220918-
    288x160-bs2.tachyrt"
  }
}
```

3-4) USB 장치 펌웨어 로딩

```
if "ftdi" in config["INTERFACE"] ["mode"]:  
    ret = rt_core.boot(path='../firmware', spi_type='ftdi')  
    if ret:  
        print("Success to boot. Check the status via uart or  
other api")  
    else:  
        print("Failed to boot")  
        exit(-1)
```

3-5) 모델 로딩

```
engine = rt_core.make_engine(config)  
if engine is None:  
    print("make engine fail")  
    exit()
```

3-6) 입력 이미지 불러오기

```
# The Object Detection model expects images  
in RGB format.  
image = cv2.imread("./image/image_3cls.jpg")  
org_h, org_w, _ = image.shape  
  
# Input size of Object Detection model is  
288x160(h x w)  
image_input = cv2.resize(image, (288,  
160))[..., ::-1]  
  
# Shape of input must be (B, H, W, D)  
input_image = np.expand_dims(image_input, 0)  
display_img_plt(image)
```

3-7) 추론 실행

```
engine.process(input_image)
ret = engine.get_result()
```

3-8) 후처리 수행

1. Prepare post processing configuration
2. Create post processing instance
3. Do post processing

```
# load post configuration
post_config =
read_json('./config/post_3cls.json')

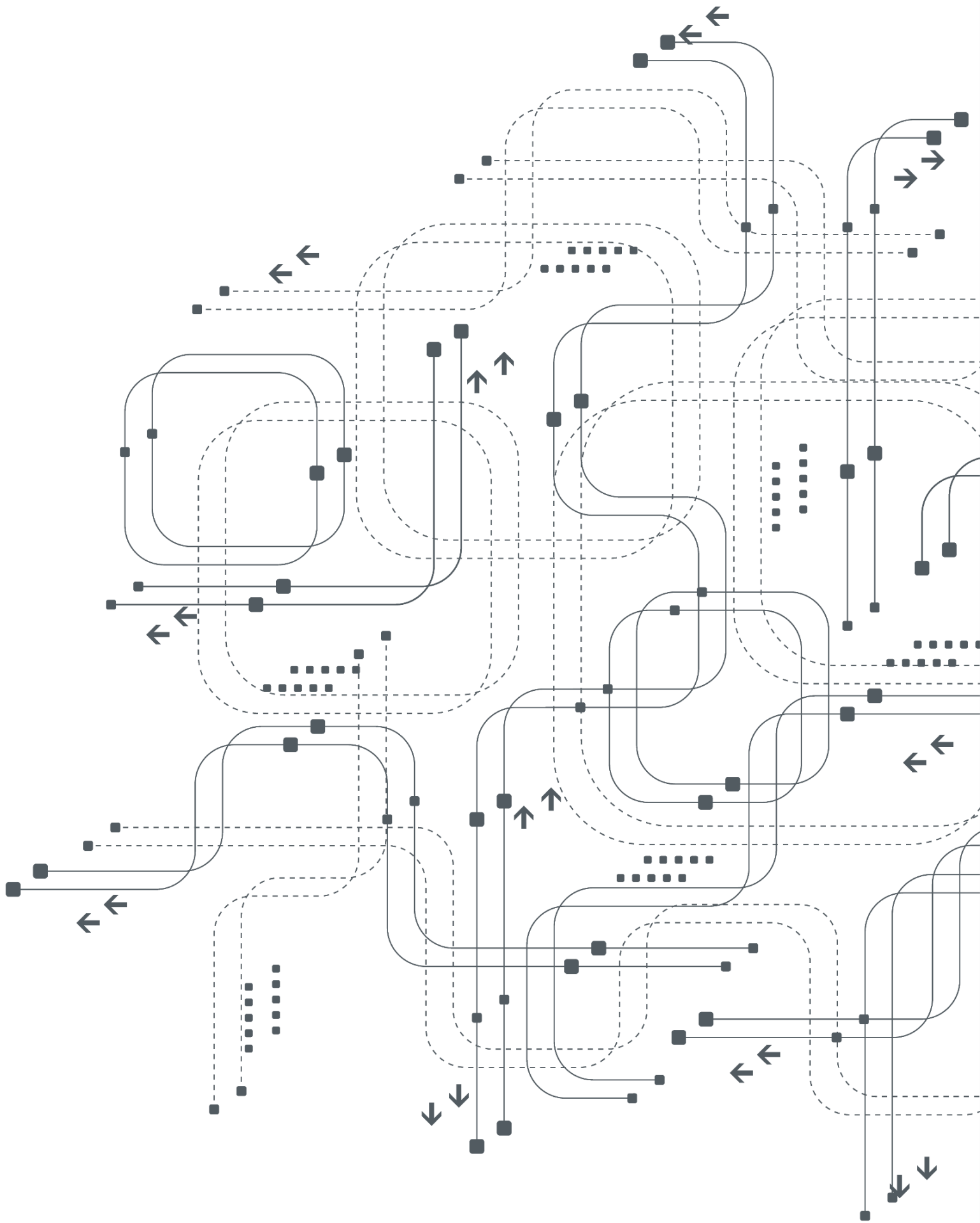
# Init post decoder
output_depth =
int(np.mean(np.asarray(post_config["INF_SHAPES_OUTPUT"])[..., -1]))
post_processing = Decoder(post_config)

# Do post process for annotation
anno =
post_processing.main(ret['buf'].reshape(-1,
output_depth), np.array([[0, 0, org_w-1,
org_h-1]], dtype=np.float32))
```

Draw Annotations

1. Prepare labels for detection result
2. Draw detection

```
# Draw box and show image
labels = read_json('./label/label_3cls.json')
img = draw_detection(image, anno, labels)
display_img_plt(img)
```



고객 지원 (Customer Support)

- 문의처: partner@deeper-i.ai
- 웹사이트: <https://www.deeper-i.ai>