

Jetson TX2 기반 YOLO 응용 과정

- Day 2 -

2020.00



양재 R&CD 혁신허브
Yangjae Innovation Hub

 모두의연구소

목 차

01

Jetson TX2 디바이스 제어

02

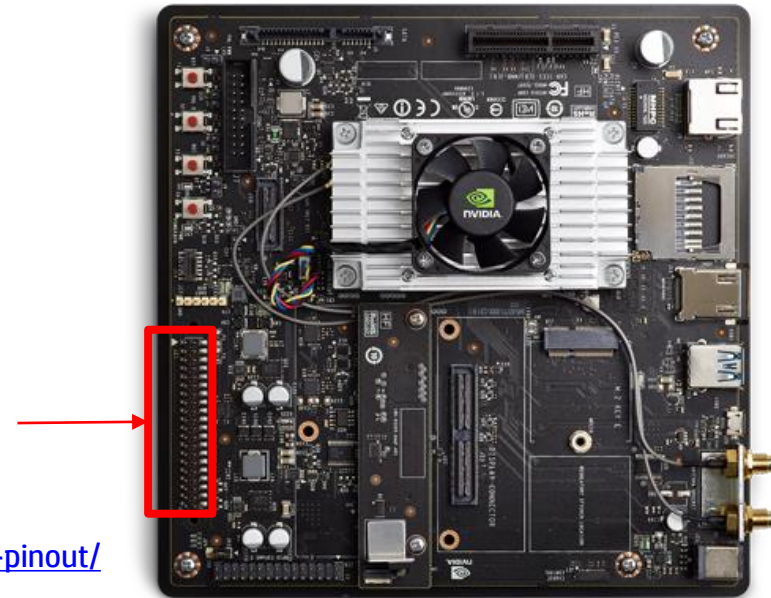
TensorFlow 이론 및 실습



01. Jetson TX2 디바이스 제어

* GPIO

- General Purpose Input Output (범용 입력 출력)
- 마이크로프로세서가 **주변장치와 통신하기 위해 범용으로 사용되는 입력 출력 포트**
- GPIO는 프로그래머가 그 포트에 대해 입력 혹은 출력인지 설정할 수 있으며,
- GPIO포트는 프로그래머가 목적(입력,출력)을 설정하기 이전에 사용이 불가능.



<https://www.jetsonhacks.com/nvidia-jetson-tx2-j21-header-pinout/>

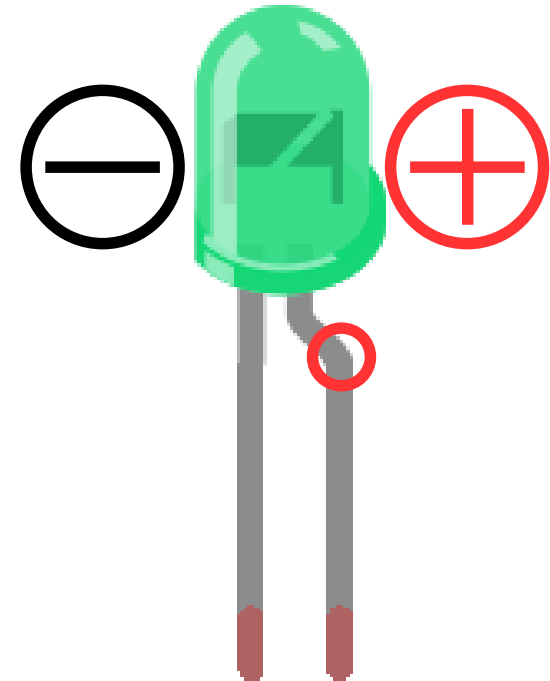
01. Jetson TX2 디바이스 제어

[1] LED 제어(GPIO)

Sysfs GPIO	Connector Label	Pin	Pin	Connector Label	Sysfs GPIO
	3.3 VDC Power	1	2	5.0 VDC Power	
	SDA1 General I2C Data 3.3V, I2C Bus 1	3	4	5.0 VDC Power	
	SCL1 General I2C Clock 3.3V, I2C Bus 1	5	6	GND	
gpio396	GPIO_ECLK Audio Master Clock (1.8/3.3V)	7	8	TXD0 UART #0 Transmit	
	GND	9	10	RXD0 UART #0 Receive	
gpio466	GPIO_GEN0 UART #0 Request to Send	11	12	GPIO_GEN1 Audio I2S #0 Clock	gpio392
gpio397	GPIO_GEN2 Audio Code Interrupt	13	14	GND	
gpio255	GPIO_GEN3 From GPIO Expander (P17)	15	16	GPIO_GEN4 Unused	gpio296
	3.3 VDC Power	17	18	GPIO_GEN5 Modem Wake AP GPIO	gpio481
gpio429	SPI_MOSI SPI #1 Master Out/Slave In	19	20	GND	
gpio428	SPI_MISO SPI #1 Master In/Slave Out	21	22	GPIO_GEN6 From GPIO Expander (P16)	gpio254
gpio427	SPI_SCLK SPI #1 Shift Clock	23	24	SPI_CSN_N SPI #1 Chip Select #0	gpio430
	GND	25	26	SPI_CSN_N SPI #1 Chip Select #1	
	ID_SD General I2C #1 Data (3.3V, I2C Bus 0)	27	28	ID_SC General I2C #1 Clock (3.3V, I2C Bus 0)	
gpio398	GPIO5 Audio Reset (1.8/3.3V)	29	30	GND	
gpio298	GPIO6 Motion Interrupt (3.3V)	31	32	GPIO12 Unused	gpio297
gpio389	GPIO13 AP Wake BT GPIO	33	34	GND	
gpio395	GPIO19 AUDIO I2S #0 Left/Right Clock	35	36	GPIO16 UART #0 Clear to Send	gpio467
gpio388	GPIO26 (3.3V)	37	38	GPIO20 Audio I2S #0 Data in	gpio394
	GND	39	40	GPIO21 Audio I2S #0 Data in	gpio393



버튼 있는 쪽 왼쪽이 시작(1번)

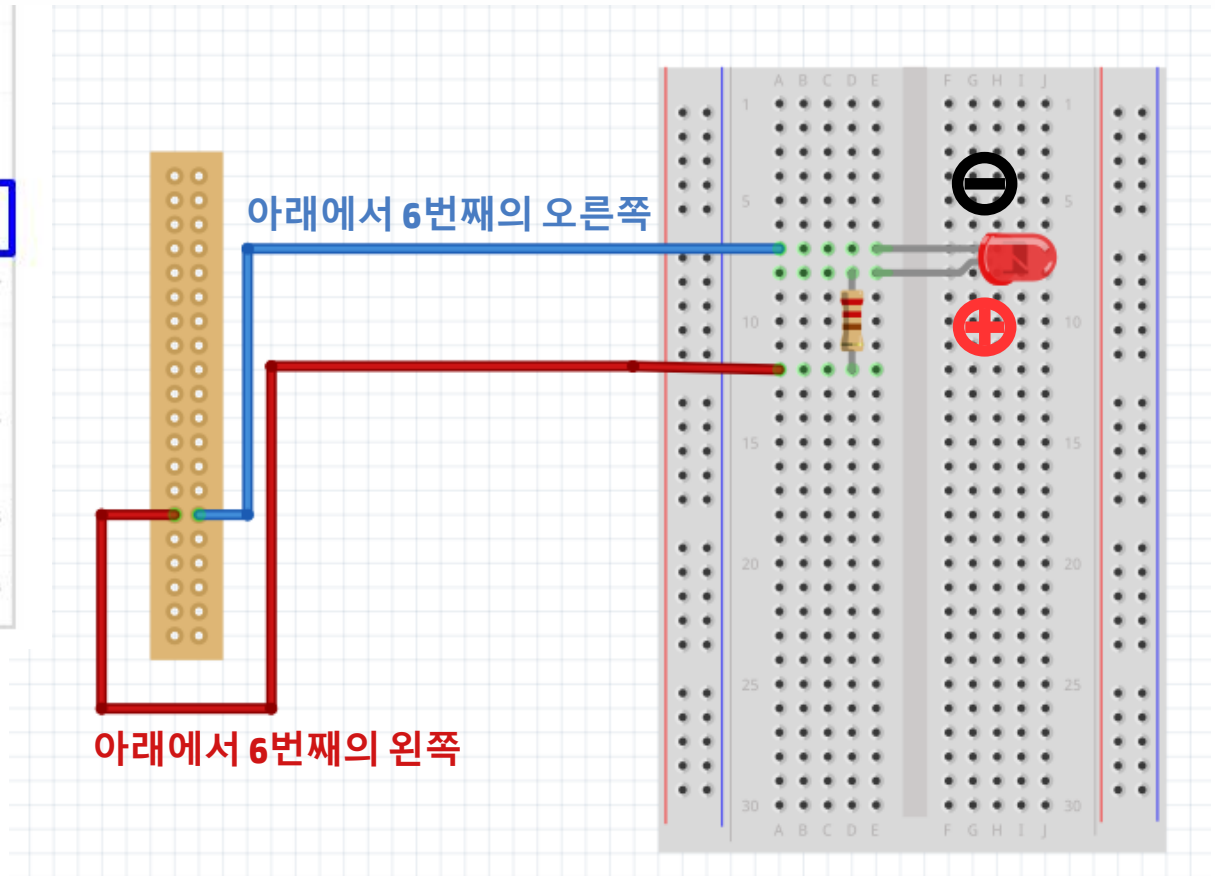


긴 쪽이 +극

01. Jetson TX2 디바이스 제어

[1] LED 제어(GPIO)

	SPI #1 Shift Clock			SPI Chip Select #0
	GND	25	26	SPI_CE1_N SPI #1 Chip Select #1
	ID_SD General I2C #1 Data (3.3V), I2C Bus 0	27	28	ID_SC General I2C #1 Clock (3.3V), I2C Bus 0
gpio398	GPIO5 Audio Reset (1.8/3.3V)	29	30	GND
gpio298	GPIO6 Motion Interrupt (3.3V)	31	32	GPIO12 Unused
gpio389	GPIO13 AP Wake Bt GPIO	33	34	GND
gpio395	GPIO19 AUDIO I2S #0 Left/Right Clock	35	36	GPIO16 UART #0 Clear to Send
gpio388	GPIO26 (3.3V)	37	38	GPIO20 Audio I2S #0 Data in
	GND	39	40	GPIO21 Audio I2S #0 Data in



01. Jetson TX2 디바이스 제어

[1] LED 제어(GPIO)

① 관리자 권한으로 터미널을 접속한다. (사용자로 돌아가기 위해서는 단축키 **ctrl+D**를 누른다.)

```
$ sudo su
```

② 제어하고자 하는 GPIO 핀 헤더 번호(398)를 export 파일에 입력하면 제어가 가능한 작업공간이 생성된다.

```
$ ls /sys/class/gpio/                # 기존 파일 목록
$ echo 398 > /sys/class/gpio/export
$ ls /sys/class/gpio/                # 파일 목록에 gpio398가 새로 생성.
```

③ 새로 생성된 gpio398 안에 direction 파일에 out을 입력하여 GPIO를 출력 헤더 핀으로 만든다.

```
$ ls -al /sys/class/gpio/gpio398/    # 경로 gpio398 내의 파일 전부(-a)를 자세히(-l) 확인.
$ echo out > /sys/class/gpio/gpio398/direction
```

④ 작업공간 gpio398의 value 파일에 1을 입력하면 ON, 그리고 0을 입력하면 OFF가 된다.

```
$ echo 1 > /sys/class/gpio/gpio398/value
$ echo 0 > /sys/class/gpio/gpio398/value
$ cat /sys/class/gpio/gpio398/value  # 현재 value 값을 터미널에 출력.
```

⑤ 헤더 핀 398을 더 이상 사용하지 않으면 unexport에 헤더 번호를 입력하여 제어공간을 커널에게 돌려준다.

```
$ echo 398 > /sys/class/gpio/unexport
$ ls /sys/class/gpio/                # 파일 목록에 gpio398가 사라짐.
```

01. Jetson TX2 디바이스 제어

[1] LED 제어(GPIO)

(참고)

GPIO 핀에 할당된 번호를 직접 구하는 방법이다.

1. 필요한 문서들을 다운로드한다:

Jetson TX2 Developer Kit Carrier Board Specification

<https://developer.nvidia.com/embedded/dlc/jetson-tx2-developer-kit-carrier-board-spec>

[참고자료]

Jetson-TX2-Generic-Customer-Pinmux-Template.xmlx

<https://devtalk.nvidia.com/cmd/default/download-comment-attachment/73754/>

2. Board Specification 문서의 제3.4장 Expansion Header에서 핀 번호에 해당하는 Tegra GPIO Port.# 확인.
Tegra GPIO Port.#는 포트 그룹(알파벳)과 핀 번호(숫자)를 알아내기 위함

3. Jetson TX2에서 아래의 경로에 있는 헤더 파일을 연다:

/usr/src/linux-headers-4.9.140-tegra-ubuntu18.04_aarch64/kernel-4.9/include/dt-bindings/gpio/tegra186-gpio.h

4. GPIO 포트 그룹 알파벳이 main인지 aon인지 확인, 그리고 해당 알파벳 옆의 숫자를 확인(포트 번호):
만일 main이면 오프셋 320, aon이면 256이다.

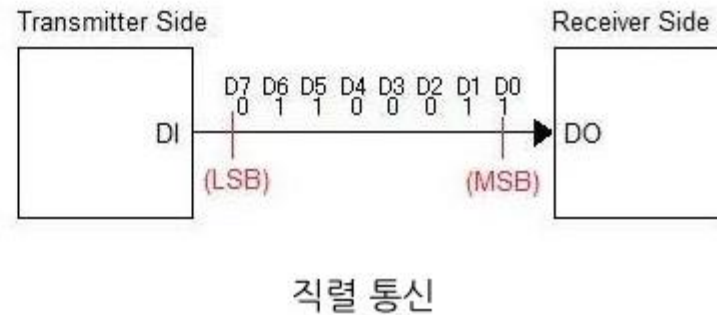
$$\text{GPIO 번호} = \text{오프셋} + (\text{포트번호} \times 8 + \text{핀 번호})$$

예시. J21 헤더 29 → J.06 → TEGRA_MAIN_GPIO_PORT_J 9, 6 → 그룹: MAIN (320), 포트번호: 9, 핀 번호: 6 = $320 + (9 \times 8 + 6) = 398$

01. Jetson TX2 디바이스 제어

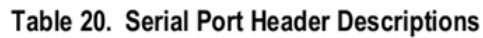
* UART

- 전송 할 Data가 “0101 1100” 일 때,



- UART(Universal asynchronous receiver/transmitter)는 직렬 통신 방법 중 하나
- 직렬 통신방법의 주요 용어

TX	Data를 전송하는 쪽 (Transmitter)
RX	Data를 수신하는 쪽 (Receiver)
Baud rate	TX와 RX에서 데이터를 주고받는 속도



Legend	Ground	Power	Not available on Jetson TX1	Not available on Jetson TX2	Reserved	Unassigned on carrier board
--------	--------	-------	-----------------------------	-----------------------------	----------	-----------------------------

Notes: In the Type/Dir column, Output is to Serial Port header. Input is from Serial Port header. Bidir is for Bidirectional signals.

01. Jetson TX2 디바이스 제어

[2] UART 제어

① UART 송수신을 확인하기 위해 minicom 프로그램을 설치한다.

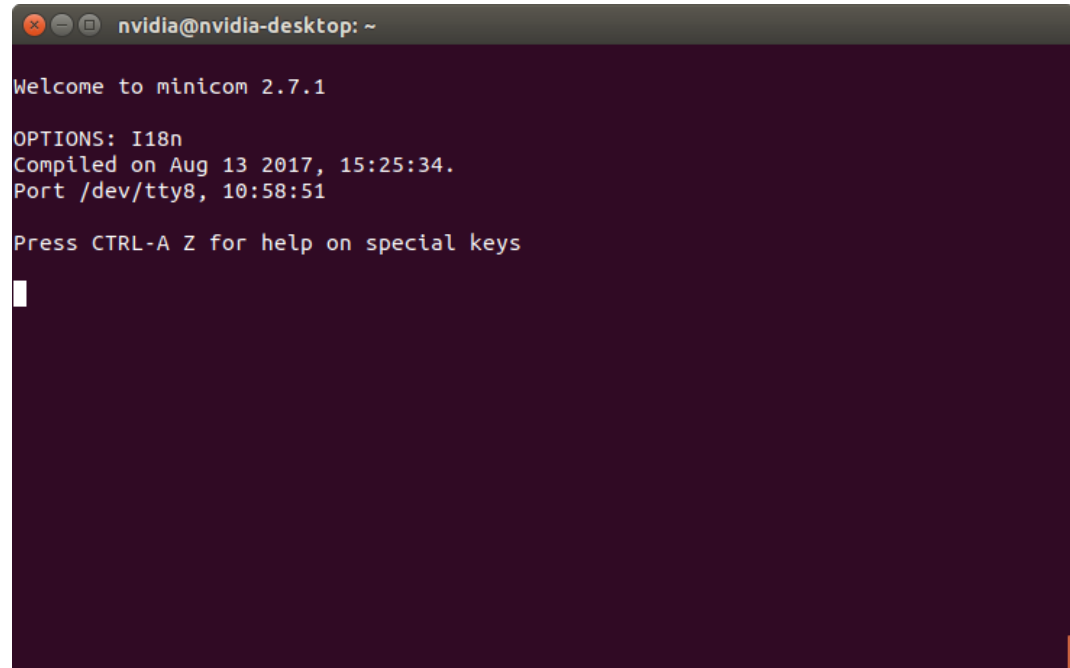
```
$ sudo apt-get install minicom
```

② Minicom 프로그램을 실행한다.

```
$ sudo minicom
```

현재 포트	J18 UART
/dev/tty8	/dev/ttyTHS2

- 키보드를 입력해도 아무런 문자가 나타나지 않는다.
- 단축기 **ctrl + A** 그리고 **X**를 눌러 종료.



```
nvidia@nvidia-desktop: ~  
Welcome to minicom 2.7.1  
OPTIONS: I18n  
Compiled on Aug 13 2017, 15:25:34.  
Port /dev/tty8, 10:58:51  
Press CTRL-A Z for help on special keys  
|
```

01. Jetson TX2 디바이스 제어

[2] UART 제어

③ J17 UART 확인을 위해 현재 설정된 포트를 /dev/ttyTHS2로 변경한다.

```
$ sudo minicom -s
```

④ Serial port setup 선택.

⑤ A 버튼 입력

⑥ tty8 → ttyTHS2 로 변경

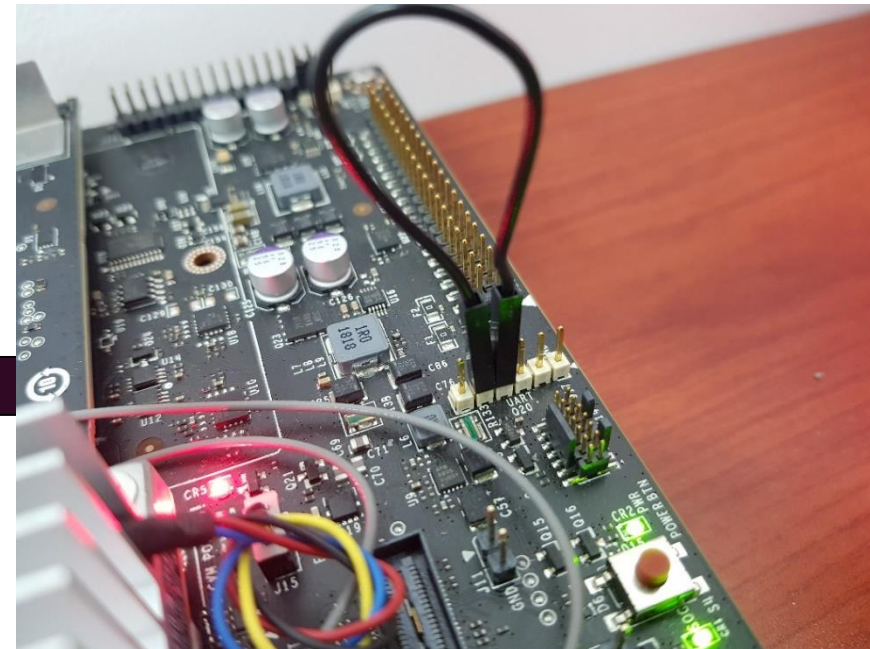
⑦ save setup as dfl 선택

⑧ Exit 선택

⑨ J17 헤더 핀 4번(RX)과 5번(TX)을 연결하고 minicom을 실행한다.

```
$ sudo minicom
```

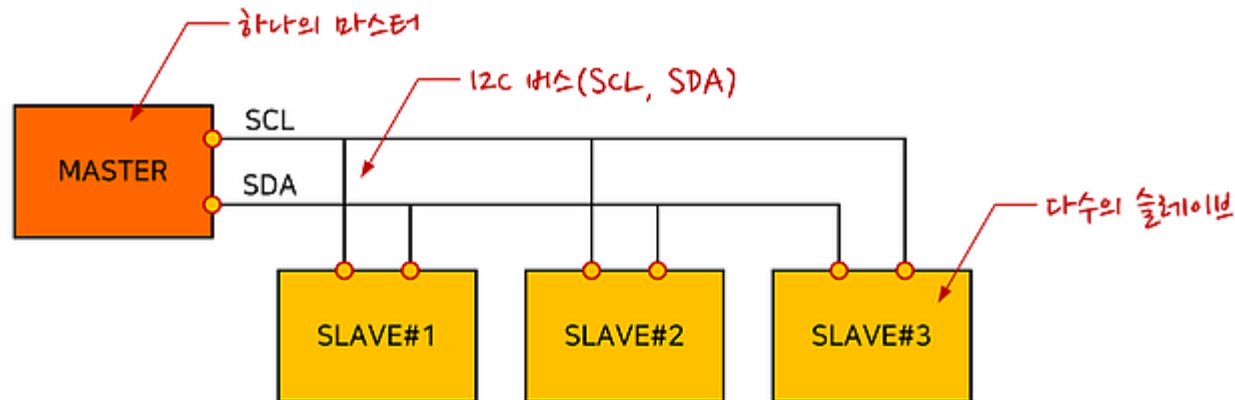
- 키보드를 입력하면 터미널에 문자가 나타난다.
- 키보드 입력이 TX로 통해 송신하여, RX로 수신하여 터미널에 보여준다.
- 루프백(loopback)로 통신상태 확인.



01. Jetson TX2 디바이스 제어

* I2C

- Inter-Integrated Circuit
- I2C 통신은 데이터를 주고받기 위한 선(SDA) 하나, 송수신 타이밍 동기화를 위한 클럭선(SCL) 하나로 이루어진다.
- 한번에 하나의 마스터와 하나의 슬레이브만 통신 가능.
- 데이터의 송수신은 마스터에서 주도한다.



01. Jetson TX2 디바이스 제어

[3] I2C 제어

① I2C 설치

```
$ sudo apt-get install libi2c-dev i2c-tools -y
```

② I2C 실행 (Bus 0번 직렬 통신 확인)

```
$ sudo i2cdetect -y -r 0
```

-y	-r	0
명령어 YES 자동 입력	읽기모드 장치 탐색 방식	I2C Bus 0 탐색 (27번, 28번 핀)

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:			--	--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	UU	UU	UU	UU	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	UU	--	--	UU								



01. Jetson TX2 디바이스 제어

[3] I2C 제어 _ VL53L0X 레이저 거리 센서

① I2C 명령어로 Bus 0 직렬 통신 확인

```
$ sudo i2cdetect -y -r 0
```

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:			--	--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	29	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	UU	UU	UU	UU	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	UU	--	--	UU								

3.3 VDC Power	17	18	GPIO_GEN5 Modem Wake AP GPIO
SPI_MOSI SPI #1 Master Out/Slave In	19	20	GND
SPI1_MISO SPI #1 Master In/Slave Out	21	22	GPIO_GEN6 From GPIO Epander (P16)
SPI_SCLK SPI #1 Shift Clock	23	24	SPI_CE0_N SPI Chip Select #0
GND	25	26	SPI_CE1_N SPI #1 Chip Select #1
ID_SD General I2C #1 Data (3.3V), I2C Bus 0	27	28	ID_SC General I2C #1 Clock (3.3V), I2C Bus 0

② 레이저 거리 센서 코드 다운로드 및 실행

```
$ mkdir ~/Workspace
$ cd ~/Workspace
$ git clone https://github.com/jetsonhacks/JHVL53L0X.git
$ cd JHVL53L0X
$ ./installPre.sh
```

01. Jetson TX2 디바이스 제어

[3] I2C 제어 _ VL53L0X 레이저 거리 센서

③ 소스코드 수정

파일 ./JHVL53L0X/src/VL53L0X.cpp

(1) LINE 52 (수정): `ki2CBus(0)` <<< bus 0인 경우 생략
→ `ki2CBus(1)`

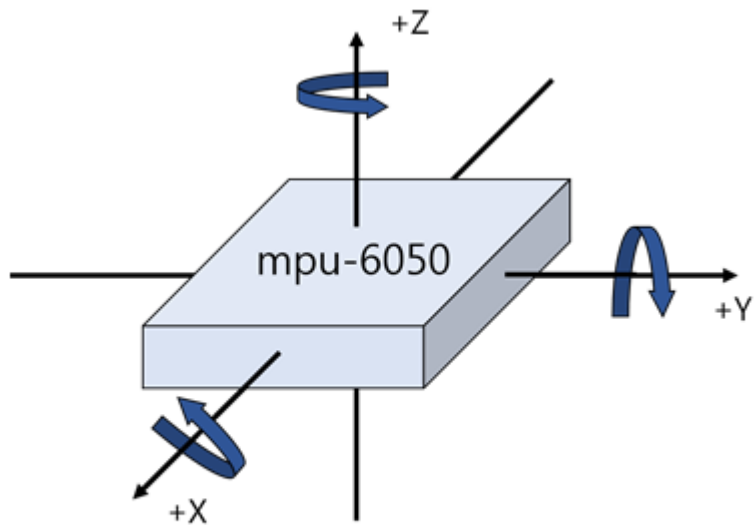
(2) LINE 18 (삽입): `extern "C" { #include <i2c/smbus.h> }`

파일 ./JHVL53L0X/example/Makefile

(1) LINE 1 (추가): `g++ example.cpp ../src/VL53L0X.cpp -I../src -o example`
→ `g++ example.cpp ../src/VL53L0X.cpp -I../src -o example -li2c`

④ 컴파일 및 실행

```
$ cd example
$ make
$ ./example
```



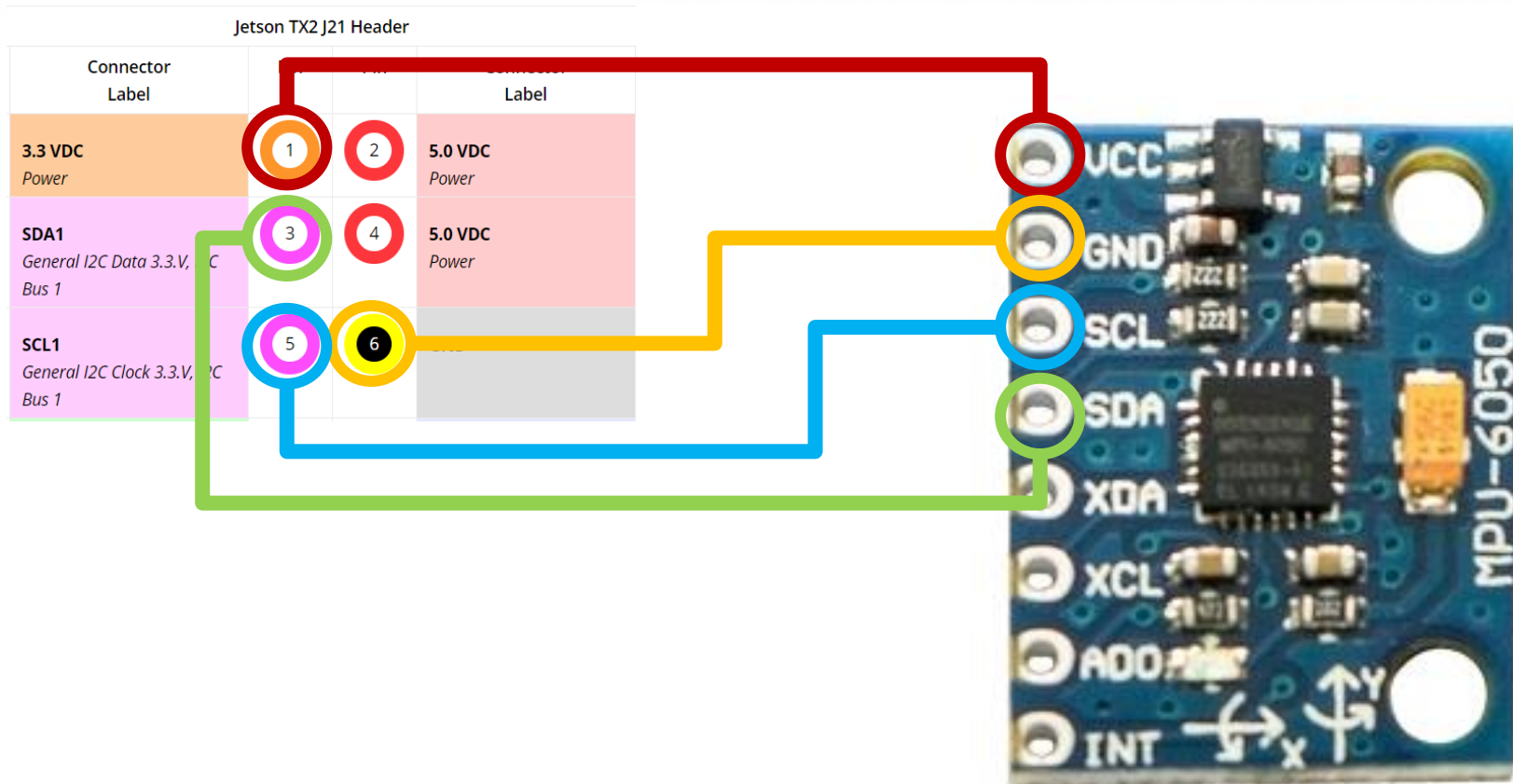
MPU 6050 기울기센서 축 방향

크기	20 X 15 mm
용도	드론, 로봇의 기울기, 위치, 회전 추정
동작 전압	3.3V ~ 5V
통신 방식	I2C
자이로스코프 범위	+/-250 500 1000 2000 degree/sec
가속 범위	+/-2G, 4G, 8G, 16G
핀 구성	8핀 (VCC / GND / SCL / SDA / XDA / CXL / ADO / INT)
핀 간격	2.54mm

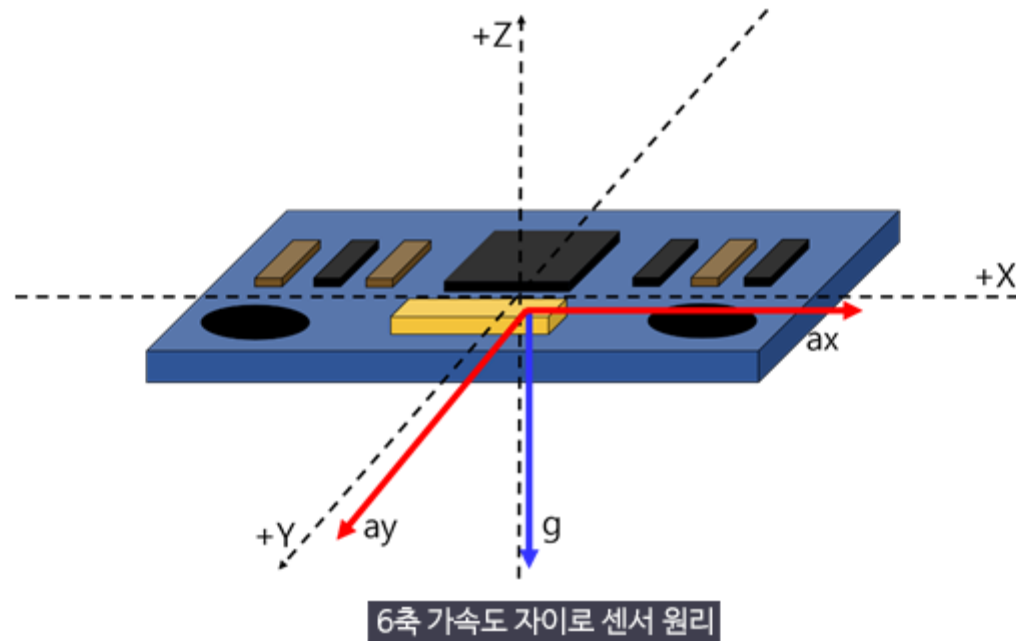
· 기울기 센서인 MPU-6050센서는 6축 자이로 가속도 센서라고 불리는 센서입니다.

여기서 6축의 뜻은 6자유도(DOF)를 의미하며 가속도3축 + 자이로 2축 + 온도1축을 줄여서 6축 기울기 센서라 부릅니다.

기울기 센서(6축 가속도+자이로) 센서



Chapter2. MPU-6050 6축 기울기 센서의 원리



→ 가속도는 시간에 대한 속도 변화의 비율을 뜻 합니다.

가속도센서는 가속도 자체를 측정하는 것이 아닌 중력가속도를 이용하여 가속도를 측정하게 됩니다.

중력 가속도가 3축으로 얼마만큼의 영향을 주었는가를 측정하여 센서의 기울어진 정도를 확인할 수 있습니다.

가속도센서의 민감도는 매우 높아 정적인 상태에서만 정확한 기울기를 측정할 수 있다는 한계를 가지고 있습니다.

→ 자이로 센서는 각속도를 측정 할 수 있는 센서로서 3축의 물리량을 측정하여 센서의 움직임을 감지하고 기울기를 측정할 수 있습니다.

가속도 센서의 비해 비교적 안정적인 값이 출력되지만 각도를 계산하는 과정에서 사용되는 적분에 의해 시간이 지날수록 누적된 오차가 발생하게 됩니다.

따라서 가속도 센서와 자이로 센서의 장점을 적절히 혼합하여 센서를 사용하는 것이 중요합니다.

Chapter3. 3축 자이로 센서로 기울기 측정하기

→ 아두이노와 MPU6050을 연결하여 **센서를 움직였을 때 변화하는 가속도와 자이로 센서 값**을 측정해 보겠습니다.

3.1. 준비물

→ 실습에 앞서 준비물이 필요합니다.

(링크를 클릭하면 해당 제품 페이지로 이동합니다.)

01. Jetson TX2 디바이스 제어

[3] I2C 제어 _ MPU 6050 자이로 센서

① I2C 명령어로 Bus 1 직렬 통신 확인

```
$ sudo i2cdetect -y -r 1
```

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:			--	--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	68	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--								

② python-smbus package 설치

```
$ sudo apt install python3-smbus
```

③ 자이로 센서 코드 다운로드 및 실행

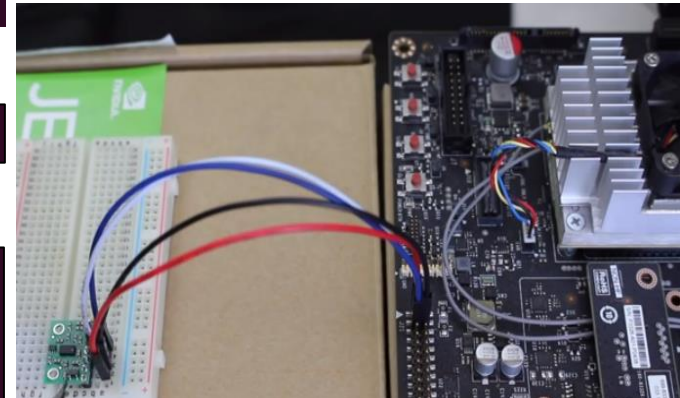
```
$ cd ~/Workspace
$ git clone https://github.com/Tijndagamer/mpu6050.git
$ cd mpu6050
$ python setup.py install
```

④ 센서 동작

```
$ mpu6050-example
```

Jetson TX2 J21 Header

Connector Label	Pin	Pin	Connector Label
3.3 VDC Power	1	2	5.0 VDC Power
SDA1 General I2C Data 3.3V, I2C Bus 1	3	4	5.0 VDC Power
SCL1 General I2C Clock 3.3V, I2C Bus 1	5	6	GND



02. TensorFlow 이론 및 실습

[1] Linear regression (선형 회귀)

① 파이썬 3 전용 numpy 및 matplotlib 라이브러리 설치.

```
$ sudo apt-get install python3-numpy python3-matplotlib
```

② TensorFlow 예제 폴더 생성

```
$ mkdir ~/TensorFlow  
$ cd ~/Tensorflow
```

③ Linear_regression.py 작성

```
$ gedit Linear_regression.py
```

▶ **Linear_regression.py** 파일에 작성해야하는 내용은 본 PPT의 다음 페이지를 참고하세요.

④ 실행

```
$ python3 Linear_regression.py
```

02. TensorFlow 이론 및 실습

[1] Linear regression (선형 회귀)

Linear_regression.py ▶

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

num_puntos = 1000
conjunto_puntos = []

for i in range(num_puntos):
    x1= np.random.normal(0.0, 0.55)
    y0 = x1 * 0.1 + 0.3
    y1 = x1 * 0.1 + 0.3 + np.random.normal(0.0, 0.03)
    conjunto_puntos.append([x1, y1])
x_data = [v[0] for v in conjunto_puntos]
y_data = [v[1] for v in conjunto_puntos]

W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))

y = W * x_data + b

loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for step in range(16):
    sess.run(train)
    plt.plot(x_data, y_data, 'ro', label='train data')
    plt.plot(x_data, sess.run(W) * x_data + sess.run(b))
    plt.xlim(-2,2); plt.ylim(0.1,0.6)
    plt.xlabel('x'); plt.ylabel('y')
    plt.legend(); plt.show()
```

02. TensorFlow 이론 및 실습

[2] MNIST Softmax 및 신경망

◆ MNIST Softmax 예제 다운로드 및 실행

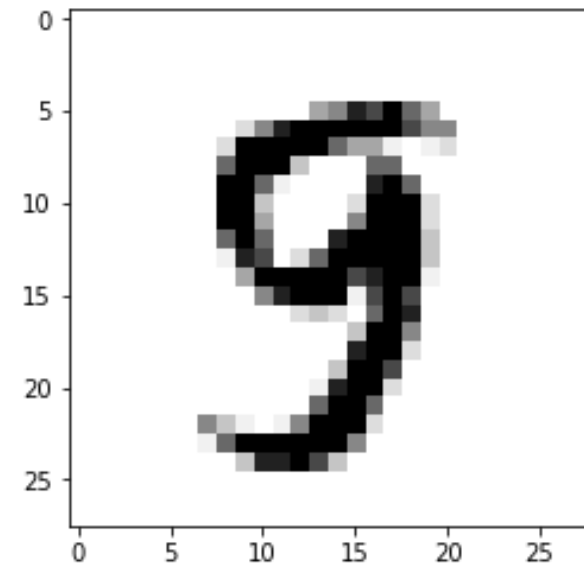
```
$ wget https://raw.githubusercontent.com/hunkim/DeepLearningZeroToAll/master/lab-10-1-mnist_softmax.py
$ python3 lab-10-1-mnist_softmax.py
```

◆ MNIST Neural Network 예제 다운로드 및 실행

```
$ wget https://raw.githubusercontent.com/hunkim/DeepLearningZeroToAll/master/lab-10-2-mnist_nn.py
$ python3 lab-10-2-mnist_nn.py
```

Lab-10-2-mnist_nn.py 소스수정 : Line 4(matplotlib)에서 주석(#)제거
Line 72~74(plt.imshow)의 앞쪽 주석제거

```
nvidia@nvidia-desktop: ~/TensorFlow
stead.
2019-11-19 14:59:28.132201: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successfully opened dynamic library libcublas.so.10.0
Epoch: 0001 cost = 142.458377561
Epoch: 0002 cost = 38.906972299
Epoch: 0003 cost = 24.306909138
Epoch: 0004 cost = 16.975207767
Epoch: 0005 cost = 12.284690088
Epoch: 0006 cost = 9.177114957
Epoch: 0007 cost = 6.941832262
Epoch: 0008 cost = 5.117768896
Epoch: 0009 cost = 3.793526473
Epoch: 0010 cost = 2.951674573
Epoch: 0011 cost = 2.205793865
Epoch: 0012 cost = 1.682234008
Epoch: 0013 cost = 1.272695524
Epoch: 0014 cost = 1.023967305
Epoch: 0015 cost = 0.873731307
Learning Finished!
Accuracy: 0.9427
Label: [9]
Prediction: [9]
```



Description

VNH2SP30 is a full bridge motor driver intended for a wide range of automotive applications. The device incorporates a dual monolithic high side driver and two low side switches. The high side driver switch is designed using the STMicroelectronic's well known and proven proprietary VIPower M0 technology which permits efficient integration on the same die of a true Power MOSFET with an intelligent signal/protection circuitary. The VIN and motor out are pitched for 5mm screw terminals, making it easy to connect larger gauge wires. INA and INB control the direction of each motor, and the PWM pins turns the motors on or off. For the VNH2SP30, the current sense (CS) pins will output approximately 0.13 volts per amp of output current.

Features

Voltage Range : 5.5V - 16V

Maximum Current rating : 30A

Practical Continuous Current: 14 A

Current sense output proportional to motor current

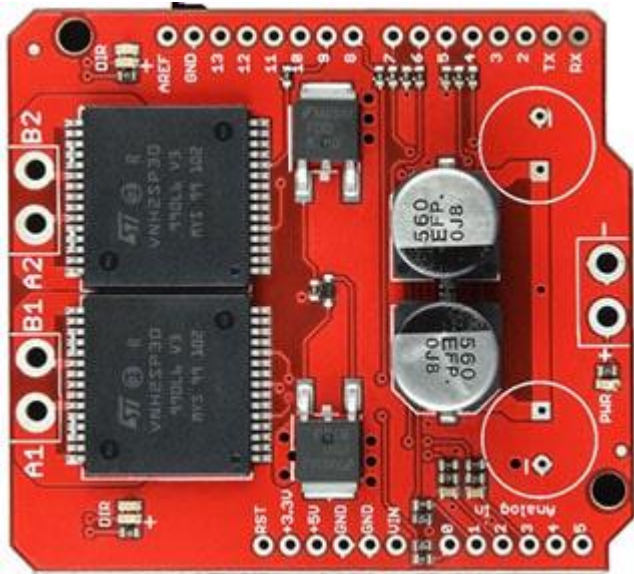
MOSFET on-resistance: 19 mΩ (per leg)

Maximum PWM frequency: 20 kHz

Thermal Shutdown

Undervoltage and Overvoltage shutdown

Monster Motor Shield Driver - 30A



Hardware Pinout

A0 : Enable pin for motor 1

A1 : Enable pin for motor 2

A2 : Current sensor for motor 1

A3 : Current sensor for motor 2

D7 : Clockwise (CW) for motor 1

D8 : Counterclockwise (CCW) for motor 1

D4 : Clockwise (CW) for motor 2

D9 : Counterclockwise (CCW) for motor 2

D5 : PWM for motor 1

D6 : PWM for motor 2

Truthtable to make motor to rotate :

Motor 0

STOP : D7 0, D8 0 & D7 1, D7 1

CCW : D7 0, D8 1

CW : D7 1, D8 0

Motor 1

STOP : D4 0, D9 0 & D4 1, D9 1

CCW : D4 0, D9 1

CW : D4 1, D9 0

Monster Motor Shield Driver - 30A

