Jetson TX2 기반 YOLO 응용 과정

- 디바이스제어

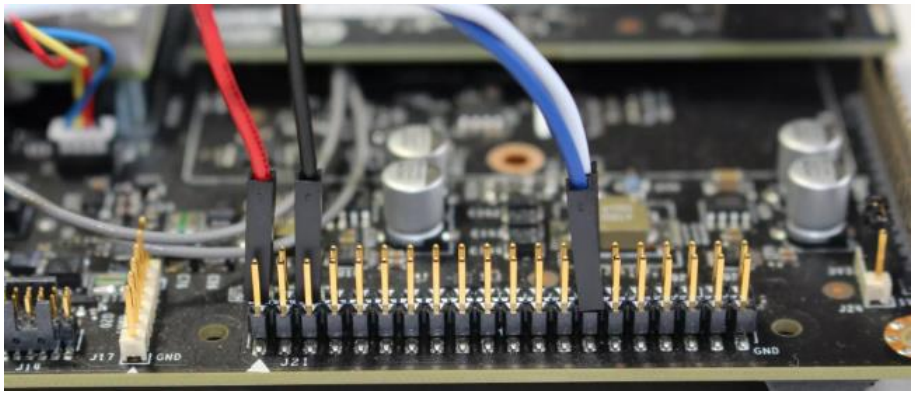
2018. 12

모두의연구소 RL4RWS

목 차

TX2-J21 핀맵

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



Jetson TX2 J21 Header						
Sysfs GPIO	Connector Label	Pin	Pin	Connector Label		
	3.3 VDC *Power*	1	2	5.0 VDC *Power*		
	SDA1 *General I2C Data 3.3.V. I2C Bus 1*	3	4	5.0 VDC *Power*		
	SCL1 *General I2C Clock 3.3.V. I2C Bus 1*	5	6	GND		
gpio396	GPIO_GCLK *Audio Master Clock (1.8/3.3.V)*	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 #(Clock*		
gpio397	GPIO_GEN2 *Audio Code Interrupt*	13	14	GND		
gpio255	GPIO_GEN3 *From GPIO Expander (P17)*	15	16	GPIO_GEN4 *Unused*		
	3.3 VDC *Power*	17	18	GPIO_GEN5 *Modem Wake AP GPIO*		
gpio429	SPI_MOSI *SPI #1 Master Out/Slave In*	19	20	GND		
gpio428	SPI1_MISO *SPI #1 Master In/Slave Out*	21	22	GPIO_GEN6 *From GPIO Expander (P16)*		
gpio427	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*		
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 #(Data in*		
	GND	39	40	GPIO21 *Audio I2S #(Data in*		

기본장치 - GPIO 제어

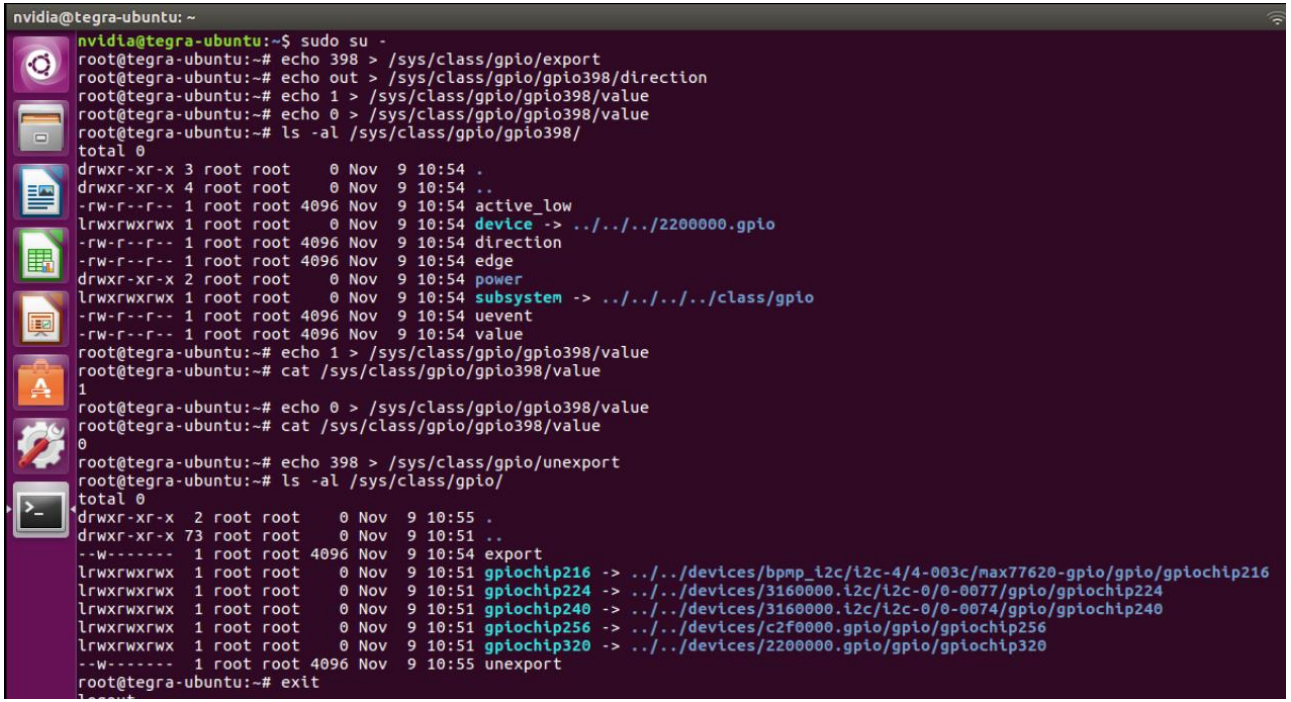
$ sudo su –

$ echo 398 > /sys/class/gpio/export
$ echo out > /sys/class/gpio/gpio398/direction
$ echo 1 > /sys/class/gpio/gpio398/value
$ echo 0 > /sys/class/gpio/gpio398/value

$ ls –al /sys/class/gpio/gpio398/

$ echo 1 > /sys/class/gpio/gpio398/value
$ cat /sys/class/gpio/gpio398/value
$ echo 0 > /sys/class/gpio/gpio398/value
$ cat /sys/class/gpio/gpio398/value

$ echo 398 > /sys/class/gpio/unexport



기본장치 – I2C 제어

I2C 설치

$ sudo dpkg --configure –a
$ **sudo apt-get install libi2c-dev i2c-tools –y**

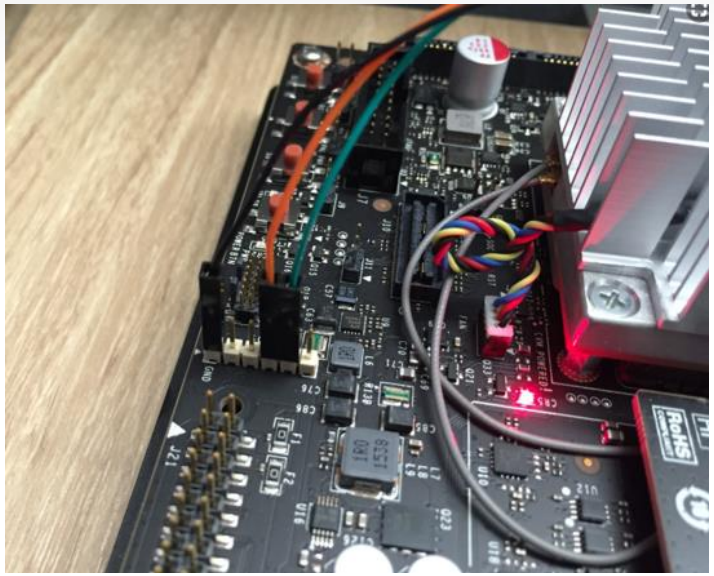
```
Fetched 8,492 B in 0s (13.1 kB/s)
Selecting previously unselected package libi2c-dev.
(Reading database ... 161959 files and directories currently installed.)
Preparing to unpack .../libi2c-dev_3.1.1-1_all.deb ...
Adding 'diversion of /usr/include/linux/i2c-dev.h to /usr/include/linux/i2c-dev.                h.kernel by libi2c-dev'
Unpacking libi2c-dev (3.1.1-1) ...
Setting up libi2c-dev (3.1.1-1) ...
```

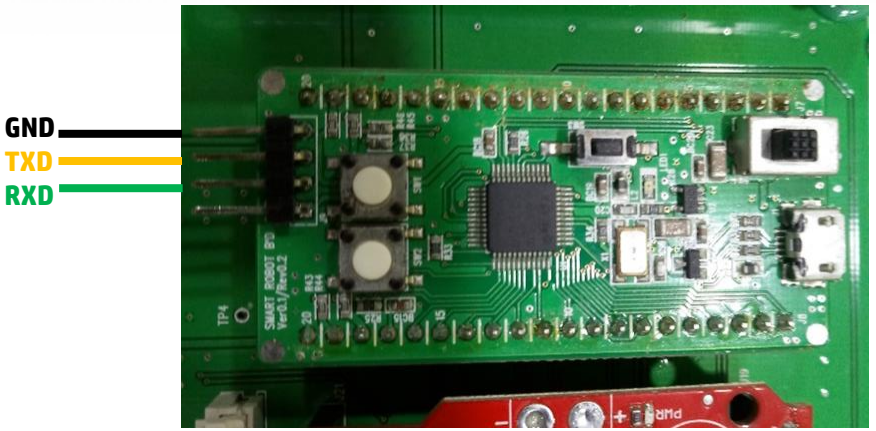
I2C 실행

```
$ 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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Serial 통신 – 회로 구성

UART 회로 연결





STM32모듈

Jetson TX2 J17와 UART 케이블 연결

Jetson TX2 J17		UART 케이블	
PIN 1	GND	GND	Black wire
PIN 4	RXD	TXD	Orange wire
PIN 5	TXD	RXD	Green wire

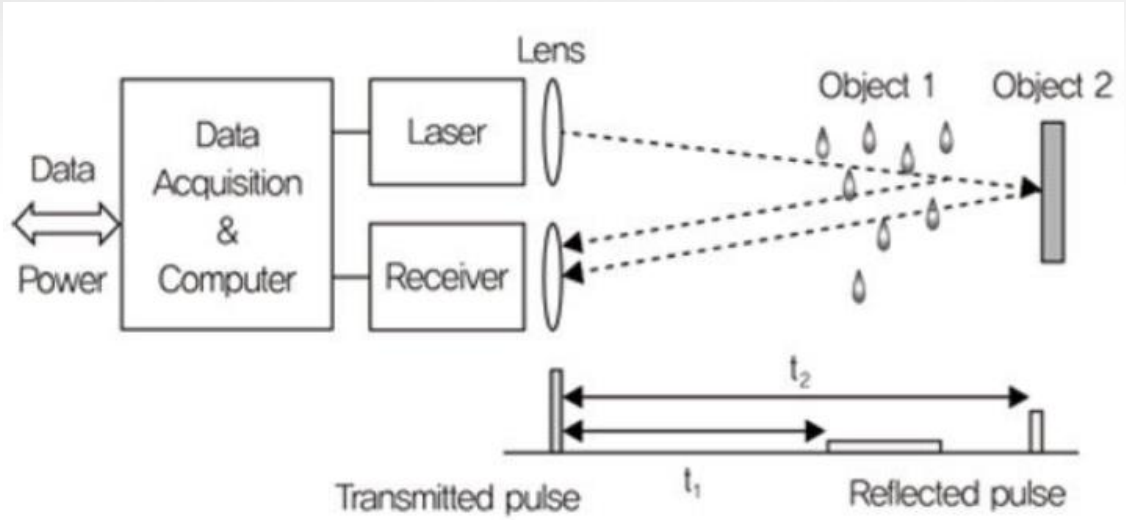
Serial 통신 – TTY

$ ls /dev/tty*
$ dmesg | grep tty
$ sudo su –
$ echo “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa” > /dev/ttyS0
$ echo “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa” > /dev/ttyTHS1
$ echo “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa” > /dev/ttyTHS2
$ exit

```
nvidia@tegra-ubuntu:~$ ls /dev/tty*
/dev/tty   /dev/tty17  /dev/tty26  /dev/tty35  /dev/tty44  /dev/tty53  /dev/tty62   /dev/ttyTHS1
/dev/tty0  /dev/tty18  /dev/tty27  /dev/tty36  /dev/tty45  /dev/tty54  /dev/tty63   /dev/ttyTHS2
/dev/tty1  /dev/tty19  /dev/tty28  /dev/tty37  /dev/tty46  /dev/tty55  /dev/tty7    /dev/ttyTHS3
/dev/tty10 /dev/tty2   /dev/tty29  /dev/tty38  /dev/tty47  /dev/tty56  /dev/tty8

/dev/tty15 /dev/tty24  /dev/tty33  /dev/tty42  /dev/tty51  /dev/tty60  /dev/ttyS2
/dev/tty16 /dev/tty25  /dev/tty34  /dev/tty43  /dev/tty52  /dev/tty61  /dev/ttyS3
nvidia@tegra-ubuntu:~$ dmesg | grep tty
[   0.000000] Kernel command line: root=/dev/mmcblk0p1 rw rootwait console=ttyS0,115200n8 console=tty0 OS=l4t fbcon=map:0
net.ifnames=0 memtype=0 video=tegrafb no_console_suspend=1 androidboot.serialno=0423318020535
bl_prof_dataptr=0x10000@0x277040000 sdhci_tegra.en_boot_part_access=1 root=/dev/mmcblk0p1 rw rootwait rootfstype=ext4
[   0.014453] console [tty0] enabled
[   0.472831] console [ttyS0] disabled
[   0.472890] 3100000.serial: ttyS0 at MMIO 0x3100000 (irq = 37, base_baud = 25500000) is a Tegra
[   2.600422] console [ttyS0] enabled
[   2.602390] 3110000.serial: ttyTHS1 at MMIO 0x3110000 (irq = 38, base_baud = 0) is a TEGRA_UART
[   2.603538] c280000.serial: ttyTHS2 at MMIO 0xc280000 (irq = 39, base_baud = 0) is a TEGRA_UART
[   2.604664] 3130000.serial: ttyTHS3 at MMIO 0x3130000 (irq = 40, base_baud = 0) is a TEGRA_UART
[   6.766962] systemd[1]: Created slice system-serial\x2dgetty.slice.
[   6.788478] systemd[1]: Created slice system-getty.slice.
nvidia@tegra-ubuntu:~$ sudo su-
root@tegra-ubuntu:~# echo "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" > /dev/ttys0
root@tegra-ubuntu:~# echo "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" > /dev/ttyTHS1
root@tegra-ubuntu:~# echo "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" > /dev/ttyTHS2
root@tegra-ubuntu:~# exit
```

레이저센서 원리



LiDAR 시스템 기본 구성 및 동작원리

- 라이다 센서 시스템의 구성은 레이저 송신부, 레이저 검출부, 신호 수집 및 처리와 데이터를 송수신하기 위한 부분으로 단순하게 구분될 수 있다. 라이다 센서는 레이저 신호의 변조 방법에 따라 time-of-flight(TOF) 방식과 phase-shift 방식으로 구분될 수 있다.
- TOF 방식은 레이저가 펄스 신호를 방출하여 측정 범위 내에 있는 물체들로부터의 반사 펄스 신호들 이 수신기에 도착하는 시간을 측정함으로써 거리를 측정하는 것이 가능하다.
- Phase-shift 방식은 특정 주파수를 가지고 연속적으로 변조되는 레이저 빔을 방출하 고 측정 범위 내에 있는 물체로부터 반사되어 되돌아 오는 신호의 위상 변화량을 측정하여 시간 및 거리를 계산하는 방식이다.
- 레이저 광원은 250nm부터 11μm까지의 파장 영역에서 특정 파장을 가지거나 파장 가변이 가능한 레이저 광원들이 사용되며, 최근에는 소형, 저전력이 가능한 반도체 레이저 다이오드가 많이 사용된다.

레이저센서와 자율주행

실제 도로에서 자동차 자율주행 기술을 시연하고, 기술적으로도 가장 진보한 기술을 확보한 라이다 센서를 장착한 구글(Google) 자율주행차

- 자율주행차 및 스마트카 분야는 구글, 애플, 테슬러 모터스, 우버, 크루즈 오토메이션, 뉴토노미, 리프트 등의 다양한 글로벌 IT업체와 토요다, 현대자동차, BMW, 메르세데스, 볼보, 지엠, 포드 등 완성차 업체
- 자율주행 기술을 연구하는 모든 차량에는 핵심 센서 기술로 라이다가 채택되어 있고, 구글, 포드, GM 등은 높은 해상도의 3차원 위치 정보를 확보하기 위해 고해상도 라이다 센서모듈을 적용하거나 저해상도 라이다 센서모듈을 다수로 배치하여 고해상도 영상 정보를 3차원으로 검출하고 있으며, 추가적인 성능 개선 및 요구사항 만족을 위해 다양한 방식의 실외용 라이다 기술을 적극적으로 활용하고 있다.
- 현재 관련업계에서 자율이동 및 자율주행을 지원하기 위해 기술적 개선의 필요성이 가장 큰 요소로 평가되는 것은 해상도와 센서모듈 가격이다. 즉 최소 비용으로 주변 공간에 대한 최대 분해능의 3차원 정보 획득을 요구하고 있다. 기존의 센서들로는 이러한 높은 수준의 분해능으로 3차원 공간정보를 확보하기 어렵기 때문에 라이다 기술이 자율주행 자동차 시대의 핵심 센서로 인식되고 있다.
- 차량용 라이다 센서는 주.야간 구분 없이 100m 이상 범위의 주변상황에 대한 고해상도 3차원 공간정보 검출이 가능하도록 해야 한다.
- 차량 및 보행자의 안정성 확보를 위하여 충분한 정보(Redundancy)를 제공하는 것은 매우 중요하다. 빛을 조사 후 검출하는 액티브 방식 라이다 기술과 더불어 차량에 탑재된 다양한 센서 기술들을 조합하여 함께 활용하면 매우 우수하면서 안정성 있는 센서 기술을 제공할 수 있지만, 라이다 기술이 제공하는 공간 분해능은 다른 센서 기술의 조합으로도 확보하기 어려운 성능으로 자율주행 자동차의 다양한 센서 중에서 라이다 센서기술은 전략적으로 매우 중요한 요소이다.

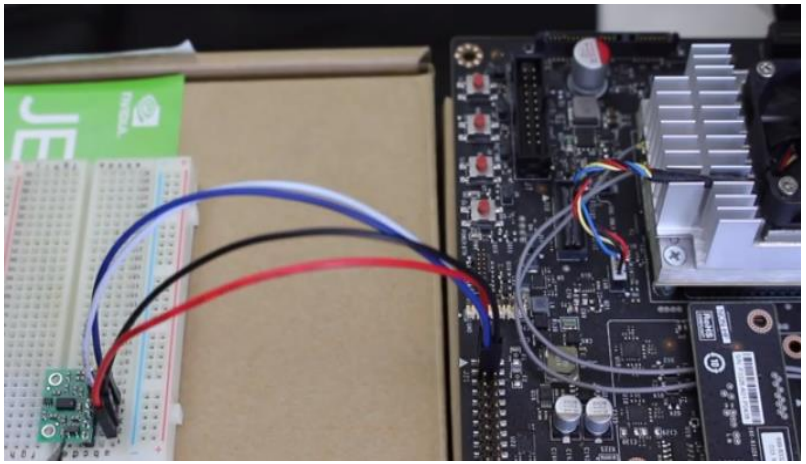
Laser Distance senor(VL53L0X)

ST Microelectronics,a new generation Time-of-Flight (ToF) laser-ranging module
It can measure absolute distances up to 2m

Jetson TX2와 VL53L0X 연결

Jetson TX2 J21 Header

Sysfs GPIO	Connector Label	Pin	Pin	Connector Label	Sysfs GPIO
	3.3 VDC *Power*	1	2	5.0 VDC *Power*	
	SDA1 *General I2C Data 3.3.V, I2C Bus 1*	3	4	5.0 VDC *Power*	
	SCL1 *General I2C Clock 3.3.V, I2C Bus 1*	5	6	GND	
gpio396	GPIO_GCLK *Audio Master Clock (1.8/3.3.V)*	7	8	TXD0 *UART #0 Transmit*	
	GND	9	10	RXD0 *UART #0 Receive*	
gpio466	GPIO_GEN0 *UART #0 Request to*	11	12	GPIO_GEN1 *Audio I2S #0 Clock*	gpio392



I2C을 연결 후 TX2 보드 전원을 켜고 실행한다.

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

```
$ sudo i2cdetect -y -r 1
0 1 2 3 4 5 6 7 8 9 a b c d e f
00: — — — — — — — — — — — — — —
10: — — — — — — — — — — — — — — — —
20: — — — — — — — — 29 — — — — — —
30: — — — — — — — — — — — — — — — —
40: — — — — — — — — — — — — — — — —
50: — — — — — — — — — — — — — — — —
60: — — — — — — — — — — — — — — — —
70: — — — — — — — —
```

소스 다운로드 및 실행

```
$ git clone https://github.com/jetsonhacks/JHVL53L0X.git
$ cd JHVL53L0X
$ cd example
$ make
$ ./example
```

모터제어(STM32) - 커맨드

TX2와 STM32 장치간에 시리얼 케이블을 연결하고 TX2에서 ttyTHS2 통신 포트를 사용하여 커맨드를 전송하면 STM32에 연결되어 있는 모터가 제어된다.

TX2 모터 커맨드

```
printf("------------------------------------
printf("                    MAIN MENU\n");
printf("------------------------------------
printf(" 1. RC Motor Stop
printf(" 2. RC Motor Forward
printf(" 3. RC Motor Backward
printf(" 4. Servo Motor Center
printf(" 5. Servo Motor Right
printf(" 6. Servo Motor Left
printf(" 7. Servo Motor Right one step
printf(" 8. Servo Motor Left  one step
```

```
// 화일을 연다.
handle = open( "/dev/ttyTHS2", O_RDWR | O_NOCTTY );
if( handle < 0 )
{
    //화일 열기 실패
    printf( "Serial Open Fail [/dev/t
    exit(0);
}

tcgetattr( handle, &oldtio );  // 현재

memset( &newtio, 0, sizeof(newtio) )
newtio.c_cflag = B115200 | CS8 | CLOC
newtio.c_iflag = IGNPAR;
newtio.c_oflag = 0;

//set input mode (non-canonical, no e
newtio.c_lflag = 0;

newtio.c_cc[VTIME] = 128;     // time-
newtio.c_cc[VMIN]  = 0;      // MIN은 r

tcflush( handle, TCIFLUSH );
tcsetattr( handle, TCSANOW, &newtio )

// 타이틀 메세지를 표출한다.
write( handle, TitleMessage, strlen(
```

RC_Control.c 소스

```
while((key=main_menu()) != 0)
{
    switch(key)
    {

        case '1':
            printf("No.1\n");
            Buff[0] = '1';
            write( handle, Buff, 1 );
            break;

        case '2':
            printf("No.2\n");
            Buff[0] = '2';
            write( handle, Buff, 1 );
            break;

        case '3':
            printf("No.3\n");
            Buff[0] = '3';
            write( handle, Buff, 1 );
            break;

        case '4':
            printf("No.4\n");
```

모터제어(STM32) - 커맨드

TX2와 STM32 장치간에 시리얼 케이블을 연결하고 TX2에서 ttyTHS2 통신 포트를 사용하여 커맨드를 전송하면 STM32에 연결되어 있는 모터가 제어된다.

TX2 모터 커맨드

```
printf("----------------------
printf("                    MA
printf("----------------------
printf(" 1. RC Motor Stop
printf(" 2. RC Motor Forward
printf(" 3. RC Motor Backward
printf(" 4. Servo Motor Center
printf(" 5. Servo Motor Right
printf(" 6. Servo Motor Left
printf(" 7. Servo Motor Right
printf(" 8. Servo Motor Left
```

RC_Control.c 소스

```
// 화일을 연다.
handle = open( "/dev/ttyTHS2", O_RDWR | O_NOCTTY );
if( handle < 0 )
{
    //화일 열기 실패
    printf( "Serial Open Fail [/dev/ttyTH
    exit(0);
}

tcgetattr( handle, &oldtio );  // 현재 설정을

memset( &newtio, 0, sizeof(newtio) );
newtio.c_cflag = B115200 | CS8 | CLOCAL |
newtio.c_iflag = IGNPAR;
newtio.c_oflag = 0;

//set input mode (non-canonical, no echo,
newtio.c_lflag = 0;

newtio.c_cc[VTIME] = 128;     // time-out
newtio.c_cc[VMIN]  = 0;      // MIN은 read

tcflush( handle, TCIFLUSH );
tcsetattr( handle, TCSANOW, &newtio );

// 타이틀 메세지를 표출한다.
write( handle, TitleMessage, strlen( Tit
```

```
while((key=main_menu()) != 0)
{
    switch(key)
    {

        case '1':
            printf("No.1\n");
            Buff[0] = '1';
            write( handle, Buff, 1 );
            break;

        case '2':
            printf("No.2\n");
            Buff[0] = '2';
            write( handle, Buff, 1 );
            break;

        case '3':
            printf("No.3\n");
            Buff[0] = '3';
            write( handle, Buff, 1 );
            break;

        case '4':
            printf("No.4\n");
```

모터 제어(STM32) - 실행

TX2 터미널에서 **RC_Control.c** 파일을 gcc 컴파일러를 사용하여 컴파일 및 실행한다.

$ gcc -o RC_Control RC_Control.c
$ ./RC_Control

```
nvidia@tegra-ubuntu:~/Work$ ls
RC_Control.c
nvidia@tegra-ubuntu:~/Work$ gcc -o RC_Control RC_Control.c
nvidia@tegra-ubuntu:~/Work$ ls
RC_Control  RC_Control.c
nvidia@tegra-ubuntu:~/Work$ ./RC_Control


------------------------------------------------
                  MAIN MENU
------------------------------------------------
 1. RC Motor Stop
 2. RC Motor Forward
 3. RC Motor Backward
 4. Servo Motor Center
 5. Servo Motor Right
 6. Servo Motor Left
 7. Servo Motor Right one step
 8. Servo Motor Left  one step
------------------------------------------------
 q. Motor Control application QUIT
------------------------------------------------
```

STM32 모터 제어 명령어

방향	명령어
left	a
right	b
up	c
down	d
stop	i