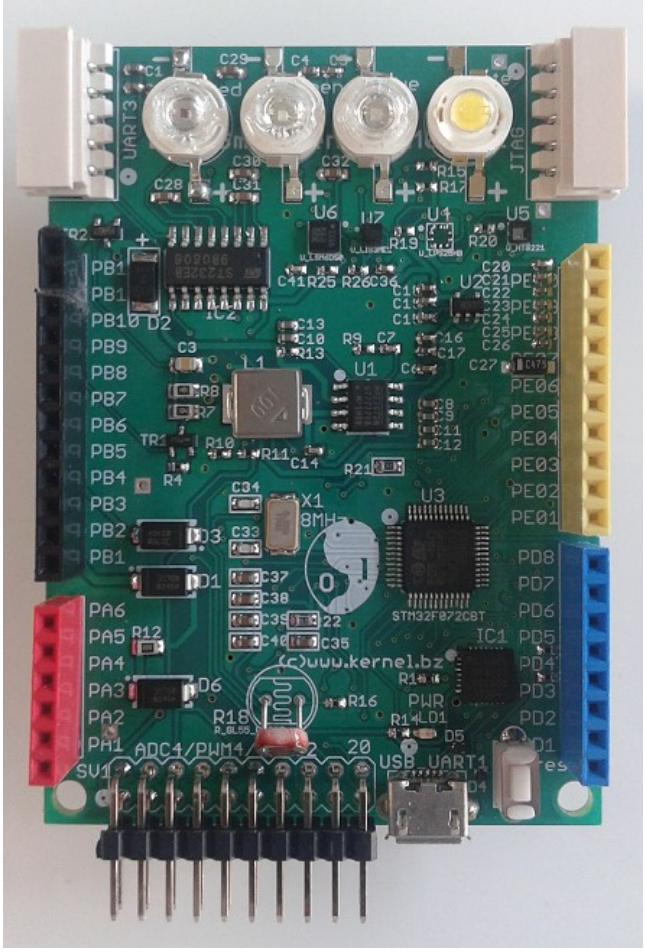
STM32 센서 프로그래밍



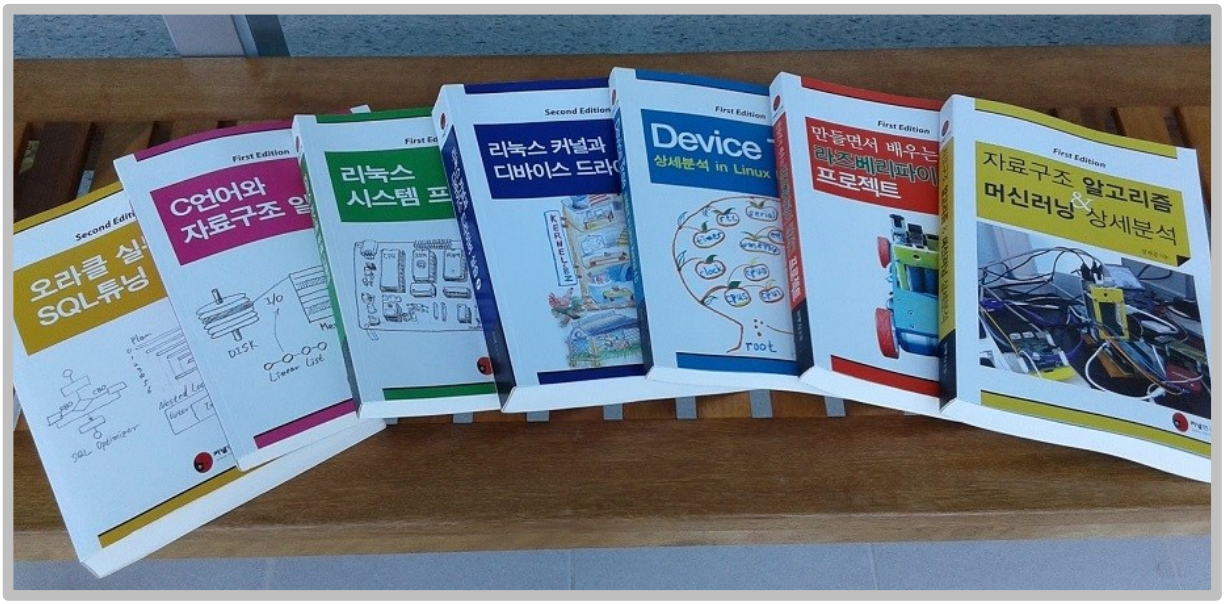
커널연구회(www.kernel.bz)

정재준(rgbi3307@nate.com)

저자 소개

정재준 (rgbi3307@nate.com) / 커널연구회(www.kernel.bz)

저자는 학창시절 마이크로프로세서 제어 기술을 배웠으며 리눅스 커널을 연구하고 있다. 15년 이상 쌓아온 실무 경험을 바탕으로 "C언어와 자료구조 알고리즘", "리눅스 시스템 프로그래밍", "리눅스 커널과 디바이스드라이버 실습2", "자료구조 알고리즘 & 머신러닝 상세분석"등의 책을 집필하고, 월간임베디드월드 잡지에 다수의 글을 기고 하였다. 또한 "맞춤형 문장 자동 번역 시스템 및 이를 위한 데이터베이스 구축방법 (The System for the customized automatic sentence translation and database construction method)" 라는 내용으로 프로그래밍을 하여 특허청에 특허등록 하였다. 최근에는 서울시 버스와 지하철 교통카드 요금결재 단말기에 들어가는 리눅스 커널과 디바이스 드라이버 개발 프로젝트를 성공적으로 수행했고 여러가지 임베디드 제품을 개발했다. 저자는 스탠포드대학교의 John L. Hennessy 교수의 저서 "Computer Organization and Design" 책을 읽고 깊은 감명을 받았으며, 컴퓨터구조와 자료구조 알고리즘 효율성 연구를 통한 기술서적 집필을 해오고 있다. 저자는 커널연구회(www.kernel.bz) 웹사이트를 운영하며 연구개발, 교육, 관련기술 공유 등을 위해 노력하고 있다.



♣ 저자가 집필한 책들

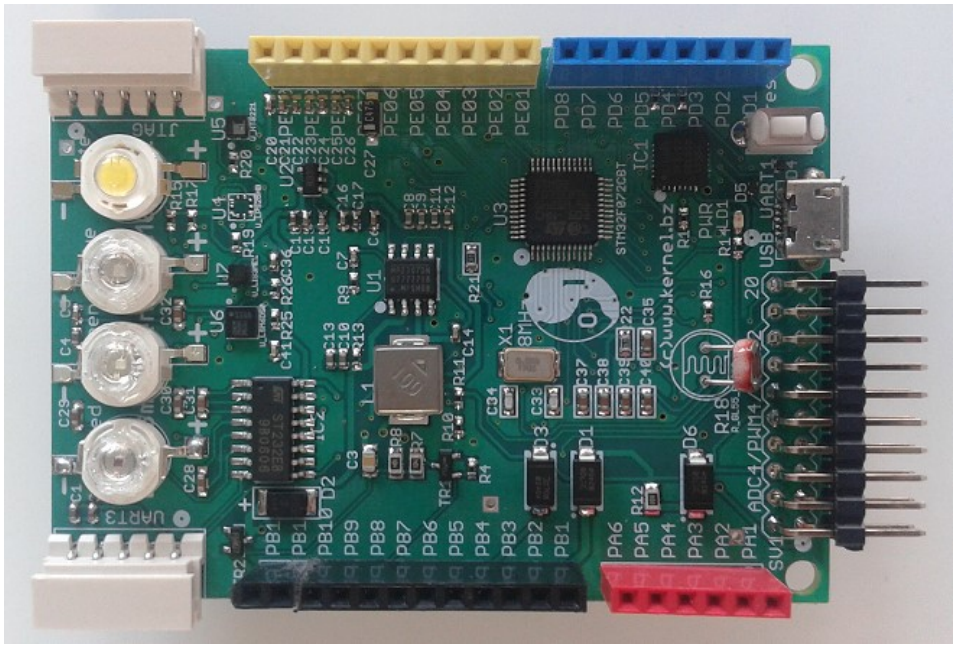
목차

내용

1. SmartSensor(센서제어) 모듈

커널연구회에서 설계하여 제작한 SmartSensor 모듈의 외형은 아래 사진과 같다. 이 모듈의 모델명은 B05_SmartSensor 이다. 이 보드는 SmartPower 모듈에서 라즈베리파이 40 핀 I/O 핀에 장착할 수 있고, 아두이노 헤더핀과도 호환된다. Cortex-M0(STM32F0) MCU 가 내장되어 있어 모듈 독립적으로 센서들로부터 데이터를 수집할 수 있다.

SmartSensor 모듈 외형 사진



SmartSensor 에서 수집하고 있는 센서 데이터들은 다음과 같다.

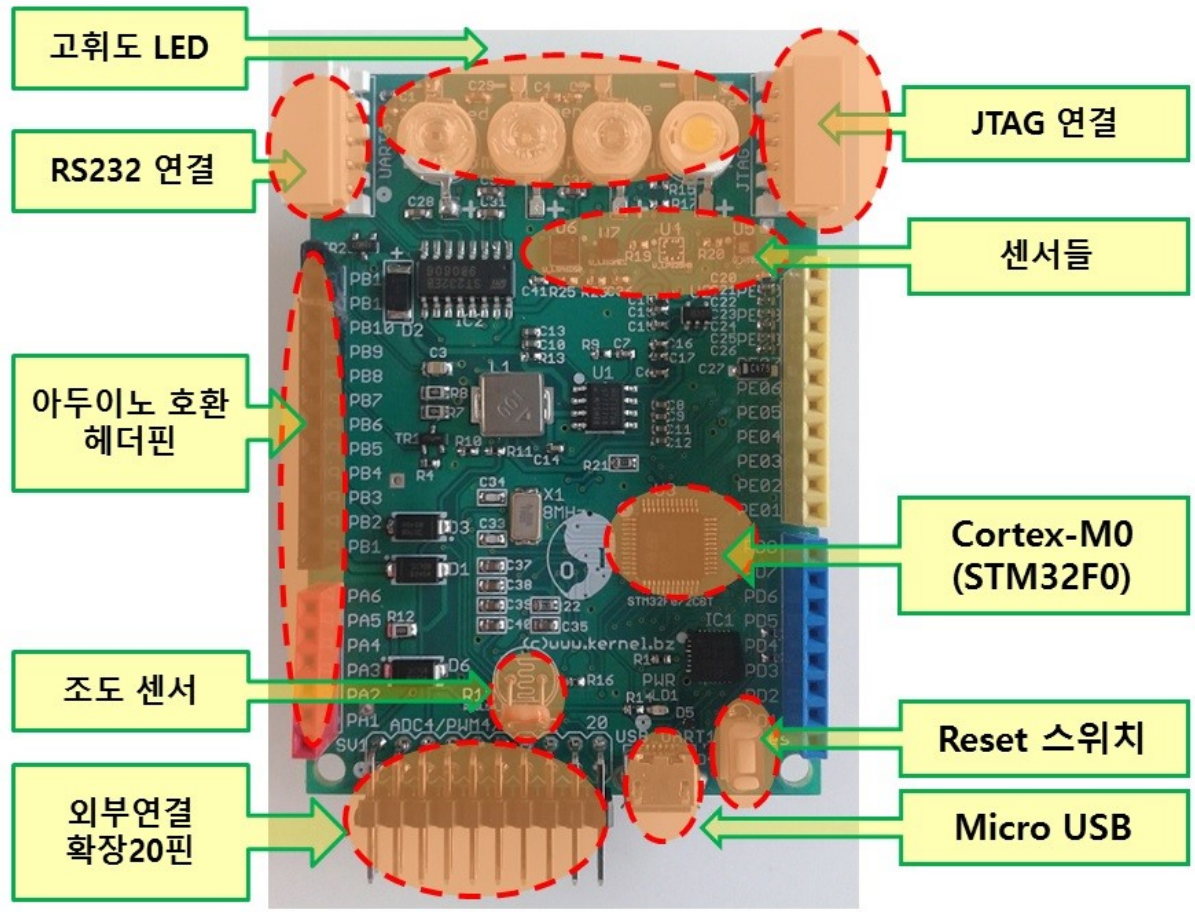
- **Accelerometer 센서: 3 축(X, Y, Z 축) 방향별로 이동속도 데이터를 수집할 수 있다.**
- **Gyroscope 센서: 3 축(X, Y, Z 축) 방향별로 기울기 데이터를 수집할 수 있다.**
- **Magnetic 센서: 3 축(X, Y, Z 축) 방향별로 지구 자기장(나침반) 데이터를 수집할 수 있다.**
- **온도, 습도, 조도 센서: 온도, 습도, 빛의 밝기 데이터를 수집할 수 있다.**
- **ADC 센서: 각종 ADC 센서(초음파, 적외선)를 연결하여 데이터를 수집할 수 있다.**

SmartSensor 모듈에 있는 마이크로 USB 포트를 PC 에 연결하고 PC 에서 시리얼통신 터미널을 실행하여 AT 명령어로 센서 데이터를 쉽게 수집할 수 있다. 또한 라즈베리파이와 아두이노와 결합하여 이동형 로봇의 센서 모듈로 사용할 수도 있다. 좀더 자세한 내용들은 아래부터 설명된다.

1.1 하드웨어 제원 및 기능

먼저, B05_SmartSensor 의 하드웨어 제원 및 기능에 대해서 알아보도록 하자. 아래 사진은 이 모듈의 제원을 부품 위치별로 설명하는 것이다.

B05_SmartSensor 상단부 설명



B05_SmartSensor 모듈 제원 및 기능요약

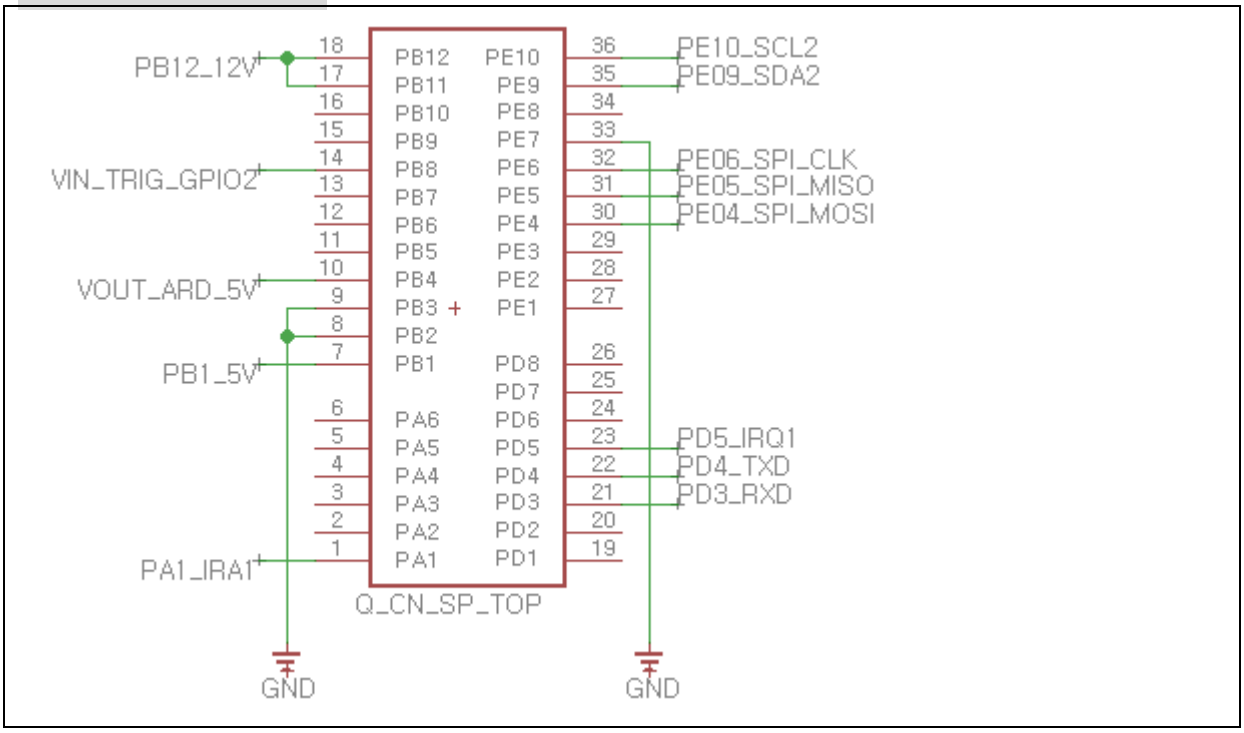
- 보드 크기는 가로 53mm, 세로 75mm (아두이노 헤더핀 호환)
- Cortex-M0(STM32F0) MCU 내장
- 센서(3 축 속도, 기울기, 방향), 온도, 습도, 조도 센서
- 외부 확장 헤더핀(20 핀) 제공하여 ADC 센서 4 개, 외부 GPIO 4 개 연결가능
- RS232 외부 시리얼 통신
- 고휘도 LED 를 색상별(Red, Green, Blue, White) 내장
- 마이크로 USB 포트에 PC 의 시리얼통신 터미널을 연결하여 AT 명령어로 동작 제어

1.2 핀맵 정보

SmartSensor 는 아두이노 헤더핀과 호환된다. 아래의 핀맵 정보를 확인하여 장치들과 연결하여 프로그래밍할 수 있다.

1.2.1 아두이노 호환 헤더핀

아두이노 호환 헤더핀



전원 연결 핀

핀번호	핀명칭	기능
PB1	PB1_5V	5V 입력핀
PB4	VOUT_ARD_5V	아두이노 보드에서 출력되는 5V 입력핀
PB11	PB12_12V	12V 입력핀
PB12	PB12_12V	12V 입력핀
PB8	VIN_TRIG_GPIO2	보드 전원 제어(On/Off)용 핀

보드에 인가되는 전압은 5V, 12V 모두 사용 가능하다. 아두이노 보드를 통하여 5V 만 입력해도 되고, SmartSensor 보드에 12V 을 인가해도 된다. 배터리(리튬이온 18650 타입) 전원은 3.7V 를 2 개 직렬로 연결하여 7.4V 를 12V 핀에 인가한다.

PB8(VIN_TRIG_GPIO2)핀은 SmartSensor 보드에 인가되는 전원을 차단하고자 할 때 사용한다. PB8 에 Low 신호가 입력되면 SmartSensor 보드 전원이 차단된다.

아두이노 및 외부 장치간에 시리얼, I2C, SPI 데이터 통신 및 인터럽트 처리는 아래의 핀번호를 참조하여 프로그래밍할 수 있다.

아두이노 및 외부 장치간 시리얼 통신

핀번호	핀명칭	기능
PD3	PD3_RXD	시리얼 데이터 수신(RxD)
PD4	PD4_TXD	시리얼 데이터 전송(TxD)

아두이노 및 외부 장치간 I2C 통신

핀번호	핀명칭	기능
PE9	PE9_SDA2	I2C 데이터 송수신
PE10	PE10_SCL2	I2C 클럭

아두이노 및 외부 장치간 SPI 통신

핀번호	핀명칭	기능
PE4	PE4_SPI_MOSI	SPI 데이터 전송(Master Out Slave In)
PE5	PE5_SPI_MISO	SPI 데이터 수신(Master In Slave Out)
PE6	PE6_SPI_CLK	SPI 클럭

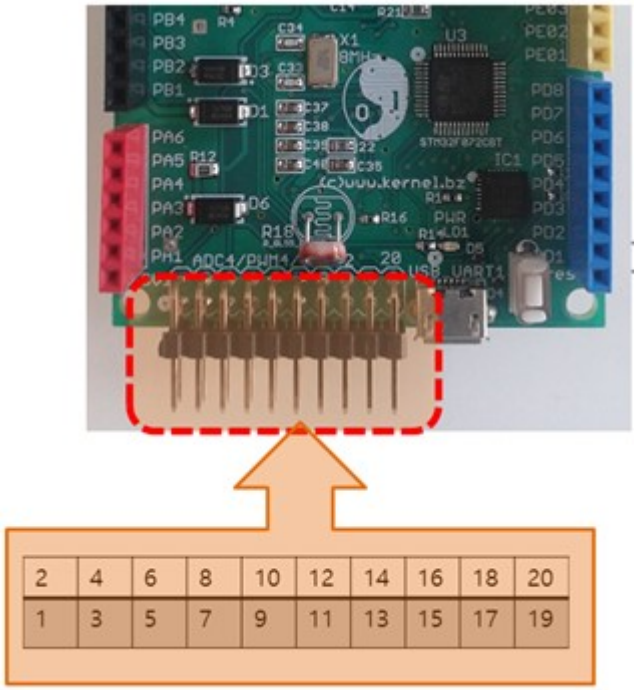
아두이노 및 외부 장치간 인터럽트 핀

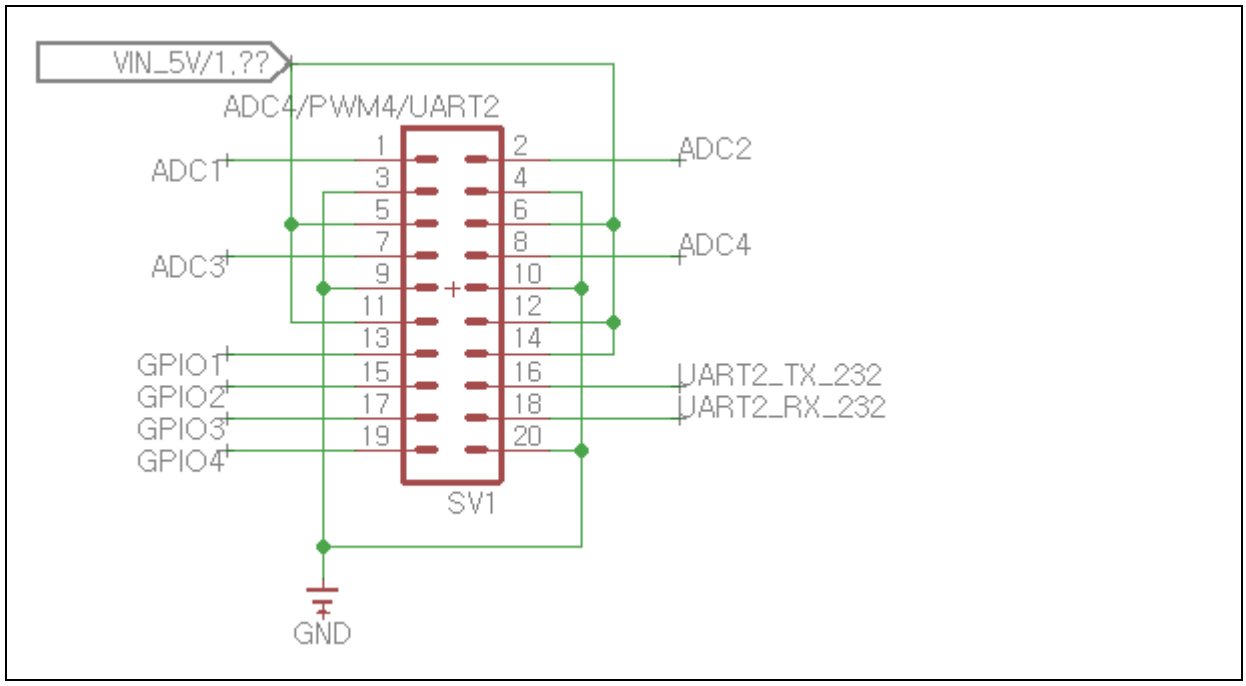
핀번호	핀명칭	기능
PD5	PD5_IRQ1	인터럽트 수신용 핀
PA1	PA1_IRA1	인터럽트 응답용 핀

1.2.2 외부 연결 확장핀

SmartSensor 보드의 하단부에 외부 확장핀(20핀: 2열 10핀)을 제공하여 ADC 센서 4개, 외부 GPIO 4개, RS232 시리얼 통신을 연결할 수 있다.

외부확장 핀위치

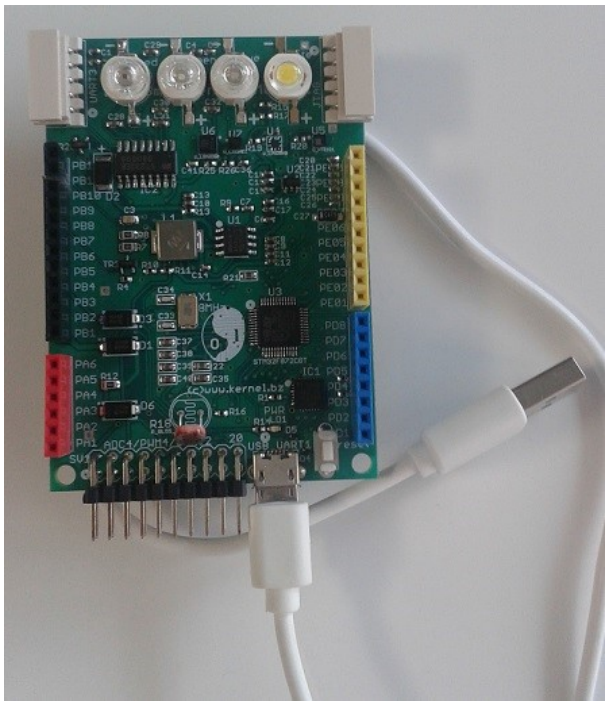




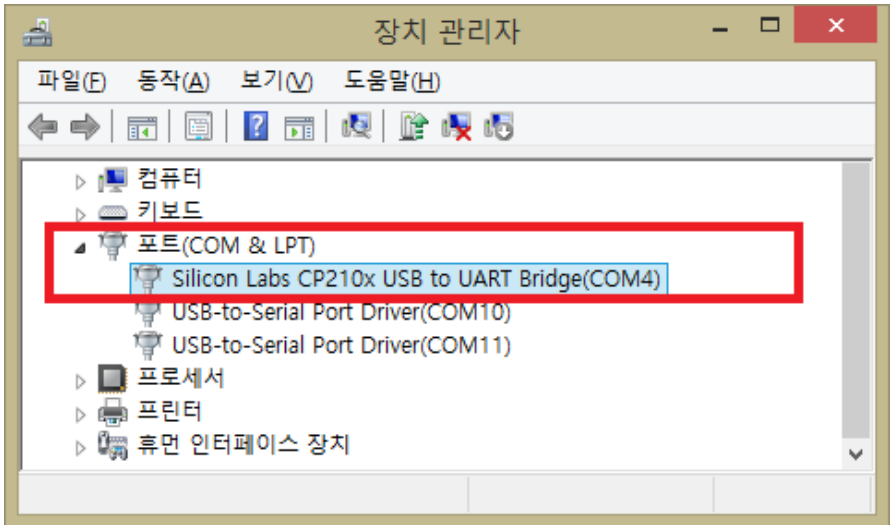
1.3 윈도우 PC에 연결방법

아래 사진처럼, SmartSensor 모듈에 있는 마이크로 USB 포트를 윈도우가 설치되어 있는 PC 에 연결한다. USB 연결선을 PC 에 장착만 하면, USB 로부터 5V 전원이 공급되므로 SmartSensor 모듈 하나는 충분히 동작된다.

마이크로 USB 포트를 윈도우PC에 연결



윈도우의 장치 관리자를 확인해 보면 아래와 같이 SmartSensor 의 시리얼 장치가 자동 인식된다.

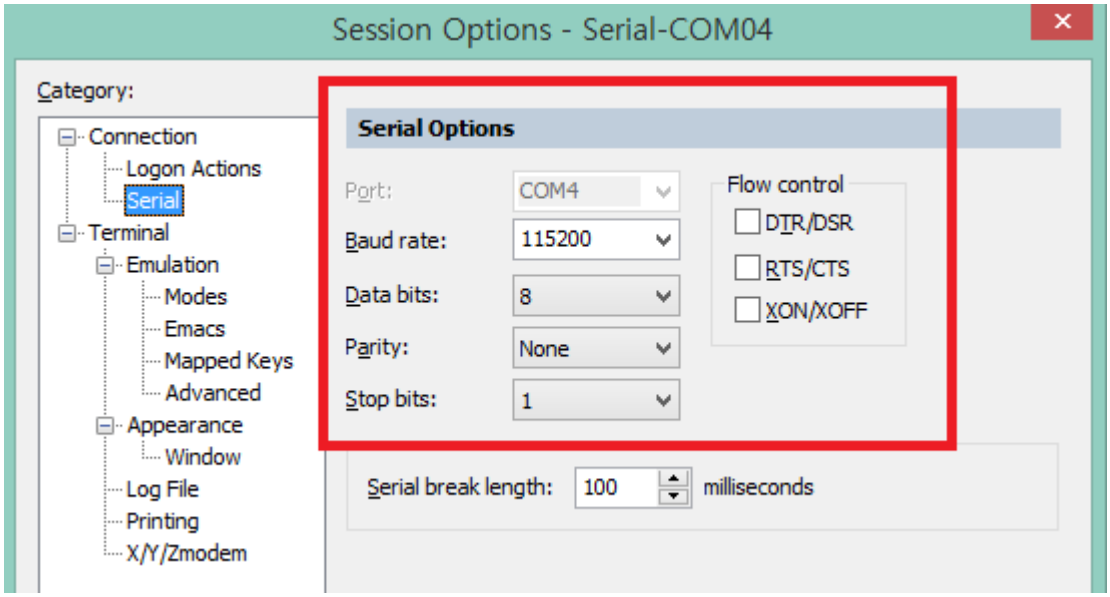


커널연구회는 SmartSensor 모듈에 USB 시리얼(UART) 장치로 Silicon Labs 사의 CP2102 IC 를 사용했다. 이 장치는 PC 에 기본적으로 드라이버가 설치되어 있으므로 별도로 드라이버를 설치하지 않아도 자동으로 인식된다.

PC 의 장치관리자에서 USB 시리얼(UART) 통신 장치의 포트 번호를 확인한다. 위의 그림에서는 COM4 포트이다. (포트 번호는 PC 마다 달라질 수 있다.)

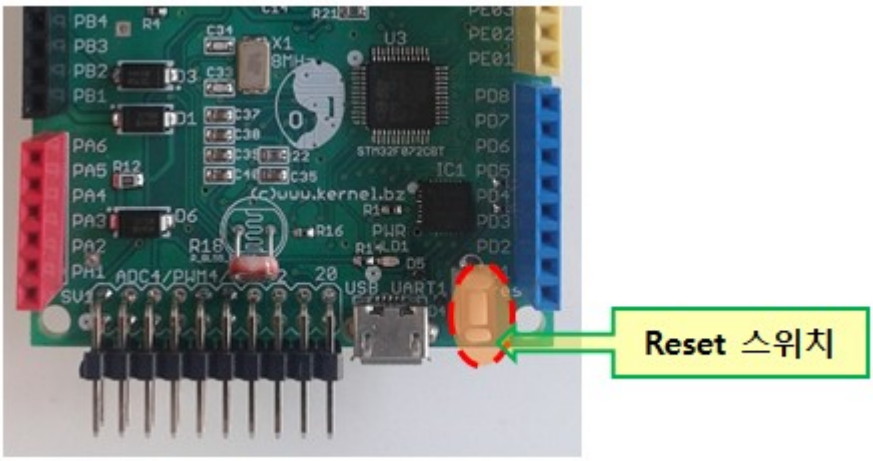
PC 에서 시리얼 터미널 프로그램을 실행한다. 윈도우 시리얼 터미널이나 Putty, SecureCRT 등 사용하기 편한 시리얼 터미널 프로그램을 실행하면 된다. 실행한 시리얼 터미널 프로그램의 시리얼 통신 설정(옵션) 메뉴에서 시리얼 옵션들을 다음과 같이 설정한다.

시리얼 통신 옵션 설정



시리얼 Baud rate 는 115200, Data bits 는 8, Parity 는 None, Stop bits 는 1 로 설정하고, Flow control 는 모두 체크하지 않는다.

위와 같이 시리얼통신 옵션들을 설정한후, SmartSensor 보드의 리셋 버튼을 한번 눌러주면 시리얼 터미널 화면에 아래와 같은 메시지들이 출력되면서 동작을 시작한다.



부팅 로그 메시지

```
user_uart_init(USART1): [OK]
user_uart_init(USART2): [OK]
user_uart_init(USART3): [OK]
user_timer6_init(): [OK]
user_timer7_init(): [OK]
user_spi_init(SPI2): [OK]
user_i2c_init(I2C1): [OK]
user_i2c2_init(I2C2): [OK]
usr_adc_dma_init(): [OK]
(c)www.kernel.bz B05_SmartSensor_V02 Started
Sensor[0]:LSM, Addr:D6, ID:68==68 [OK]
Sensor[1]:LIS, Addr:3C, ID:3D==3D [OK]
Sensor[2]:HTS, Addr:BE, ID:BC==BC [OK]
Sensor[3]:LPS, Addr:BA, ID:BD!=FF [FAIL]
ADC user1 = 1997
ADC user2 = 2119
ADC user3 = 2037
ADC user4 = 2245
ADC user5 = 2700
```

시리얼 터미널에 키보드로부터 AT? 라고 명령을 입력하면 다음과 같은 AT 명령어 사용법(도움말)이 출력된다. 이 명령어들의 자세한 사용법은 다음절부터 자세히 설명한다.

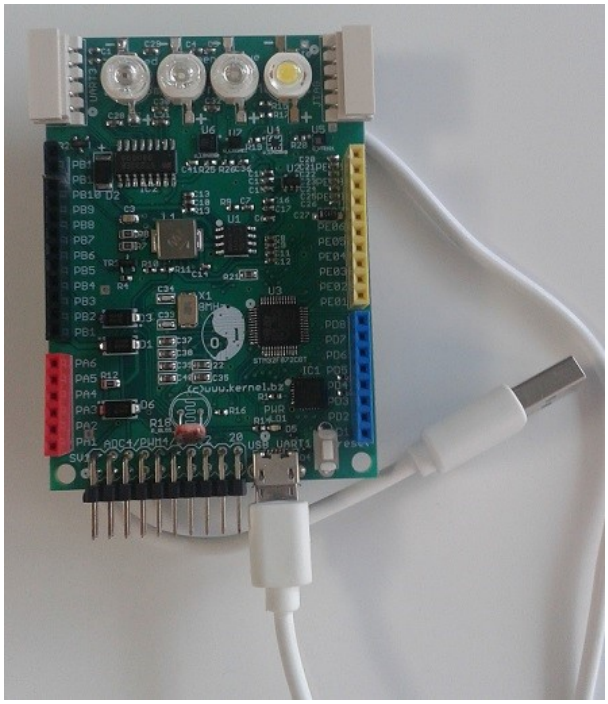
AT 명령어 도움말

```
AT?
-------- Module Version ----------------------------------------
B05_SmartSensor_V02 (c)www.kernel.bz

-------- Sensor Control Command -------------------------------
AT+SLSM?n       Getting Accelerometer/Gyroscope(LSM) Data
AT+SLIS?n       Getting Magnetic(LIS) Data
AT+SHTS?n       Getting Humidity/Temperature(HTS) Data

-------- ADC Data Command -------------------------------------
AT+SADC?n       Getting ADC Data(n:0=ALL, 1=ADC1, 2=ADC2, 3=ADC3, 4=ADC4, 5=ADC5)

-------- LED Control Command ----------------------------------
AT+SLED=n,0     LED OFF(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)
AT+SLED=n,1     LED ON(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)

OK
```

1.4 리눅스 PC에 연결방법

아래 사진처럼, SmartSensor 모듈에 있는 마이크로 USB 포트를 리눅스가 설치되어 있는 PC 에 연결한다. USB 연결선을 PC 에 장착만 하면, USB 로부터 5V 전원이 공급되므로 SmartSensor 모듈 하나는 충분히 동작된다.

마이크로 USB 포트를 리눅스 PC에 연결



위와 같이 연결했다면, 리눅스 터미널에서 다음과 같은 리눅스 명령어를 입력하여 SmartSensor 모듈의 USB 시리얼통신 장치가 연결되었는지 확인한다. (별도로 설정하지 않아도 자동으로 연결된다)

```
$ dmesg | grep tty

[    0.000000] console [tty0] enabled
[  247.957798] usb 2-2: cp210x converter now attached to ttyUSB0
```

아래 명령어로 SmartMotor 모듈의 USB 시리얼 장치가 /dev/ttyUSB0 노드에 연결되어 있음을 확인할 수 있다.

```
$ ll /dev/ttyUSB*

crw-rw---- 1 root dialout 188, 0 Mar 31 18:44 /dev/ttyUSB0
```

이제, 리눅스 명령창에서 시리얼 터미널 프로그램을 실행한다. 리눅스에서는 minicom 을 시리얼 터미널 프로그램으로 많이 사용한다. 이 프로그램이 설치되어 있지 않다면, 다음과 같이 설치한다.

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

이제 minicom 을 다음과 같이 실행한다. (루트 권한으로 실행)

```
$ sudo minicom
```

실행 메시지가 다음과 같이 화면에 출력된다.

```
Welcome to minicom 2.6.1

OPTIONS: I18n
Port /dev/ttyUSB0

Press CTRL-A Z for help on special keys
```

시리얼통신 포트가 위와 같이 /dev/ttyUSB0 로 설정되어 있지 않고, 통신 속도(baud rate)도 115200 으로 설정되어 있지 않다면, 키보드에서 Ctrl-A 를 누른후 O 키를 눌러서 다음과 같이 옵션 설정 메뉴로 진입한다.

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
```

```
| Serial port setup       |
| Modem and dialing       |
| Screen and keyboard     |
| Save setup as dfl       |
| Save setup as..         |
| Exit                    |
+-------------------------+
```

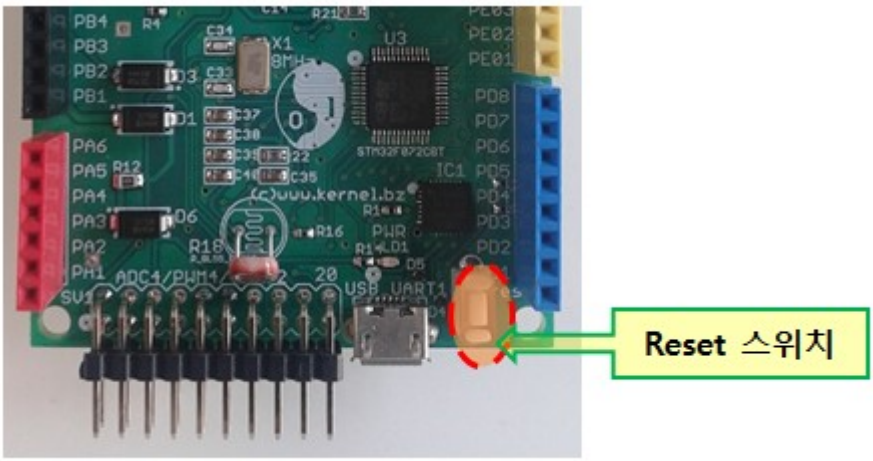
화살표를 아래로 눌러서 세번째 메뉴인 Serial port setup 을 선택한후 엔터하면 아래와 같이 설정 옵션 메뉴가 나타난다.

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

화면에 나타난 옵션들을 위와 같이 맞추어 준다. 그런다음 다시 상위 메뉴로 돌아가서 Save Setup as dfl 메뉴를 엔터하여 설정내용을 저장하고 Exit 하여 빠져 나온다.

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
+--------------------------+
```

위와 같이 시리얼통신 옵션들을 설정한후, SmartSensor 보드의 리셋 버턴을 한번 눌러주면 minicom 시리얼 터미널 화면에 아래와 같은 메시지들이 출력되면서 동작을 시작한다.



부팅 로그 메시지

```
user_uart_init(USART1): [OK]
user_uart_init(USART2): [OK]
user_uart_init(USART3): [OK]
user_timer6_init(): [OK]
user_timer7_init(): [OK]
user_spi_init(SPI2): [OK]
user_i2c_init(I2C1): [OK]
user_i2c2_init(I2C2): [OK]
usr_adc_dma_init(): [OK]
(c)www.kernel.bz B05_SmartSensor_V02 Started
Sensor[0]:LSM, Addr:D6, ID:68==68 [OK]
Sensor[1]:LIS, Addr:3C, ID:3D==3D [OK]
Sensor[2]:HTS, Addr:BE, ID:BC==BC [OK]
Sensor[3]:LPS, Addr:BA, ID:BD!=FF [FAIL]
ADC user1 = 1997
ADC user2 = 2119
ADC user3 = 2037
ADC user4 = 2245
ADC user5 = 2700
```

시리얼 터미널에 키보드로부터 AT? 라고 명령을 입력하면 다음과 같은 AT 명령어 사용법(도움말)이 출력된다. 이 명령어들의 자세한 사용법은 다음절부터 자세히 설명한다.

AT 명령어 도움말

```
AT?
-------- Module Version ----------------------------------------
B05_SmartSensor_V02 (c)www.kernel.bz

-------- Sensor Control Command --------------------------------
AT+SLSM?n       Getting Accelerometer/Gyroscope(LSM) Data
AT+SLIS?n       Getting Magnetic(LIS) Data
AT+SHTS?n       Getting Humidity/Temperature(HTS) Data

-------- ADC Data Command --------------------------------------
AT+SADC?n       Getting ADC Data(n:0=ALL, 1=ADC1, 2=ADC2, 3=ADC3, 4=ADC4, 5=ADC5)

-------- LED Control Command -----------------------------------
AT+SLED=n,0     LED OFF(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)
AT+SLED=n,1     LED ON(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)

OK
```

1.5 AT 명령어 사용법

SmartSensor 모듈과 PC간에 USB 시리얼 라인이 연결되고 옵션들이 정상적으로 설정되었다면 AT 명령어들을 사용하여 SmartSensor 모듈에 프로그램되어 있는 펌웨어 기능들을 동작시킬 수 있다. (혹시 동작하지 않는다면 앞의 연결과정을 다시한번 점검해 주기 바란다)

참고로, AT 명령어들은 소문자/대문자 구분없이 입력하면 되고, SmartSensor 모듈 내부에서는 모두 대문자로 변환하여 동작된다.

1.5.1 도움말 확인하기

시리얼 터미널 창에서 키보드로부터 AT? 을 입력하고 엔터하면 다음과 같은 도움말이 화면에 출력된다. B05_SmartSensor_V02 는 모듈의 모델명과 버전이다. 명령어들은 크게 3 가지로 나누어져 있다. 첫째는 센서들의 데이터를 가져오는 명령들이고, 둘째는 외부 확장핀에 연결되어 있는 ADC 센서들의 데이터를 가져오는 명령어들(AT+SADC?)이고, 셋째는 고휘도 LED 를 on/off 하는 명령어들(AT+SLED=) 이다. 이것들에 대해서 아래부터 자세히 사용법을 설명한다.

AT 명령어 도움말

```
AT?
-------- Module Version ----------------------------------------
B05_SmartSensor_V02 (c)www.kernel.bz

-------- Sensor Control Command --------------------------------
AT+SLSM?n       Getting Accelerometer/Gyroscope(LSM) Data
AT+SLIS?n       Getting Magnetic(LIS) Data
AT+SHTS?n       Getting Humidity/Temperature(HTS) Data

-------- ADC Data Command --------------------------------------
AT+SADC?n       Getting ADC Data(n:0=ALL, 1=ADC1, 2=ADC2, 3=ADC3, 4=ADC4, 5=ADC5)

-------- LED Control Command -----------------------------------
AT+SLED=n,0     LED OFF(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)
AT+SLED=n,1     LED ON(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)

OK
```

1.5.2 센서 데이터 가져오기

센서들의 데이터를 가져오는 AT 명령어는 다음과 같이 세가지 종류가 있다. 첫째로 AT+SLSM?n 명령어는 3 축 가속도와 3 축 기울기 데이터를 가져온다. 두번째 AT+SLIS?n 명령어는 지구 자기장(지자계) 센서로부터 방향 정보를 가져온다. 세번째 AT+SHTS?n 명령어는 온도와 습도 센서로부터 데이터를 가져온다.

센서 데이터 수집 명령어

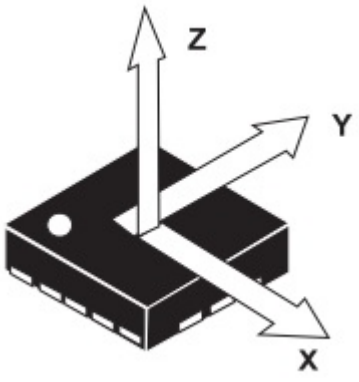
```
-------- Sensor Control Command --------------------------------
AT+SLSM?n       Getting Accelerometer/Gyroscope(LSM) Data
AT+SLIS?n       Getting Magnetic(LIS) Data
AT+SHTS?n       Getting Humidity/Temperature(HTS) Data
```

먼저 AT+SLSM? 명령어를 시리얼 터미널에 입력하면 다음과 같은 데이터를 출력한다.
AT+SLSM?n 에서 n 은 생략가능하다. 하지만 n 에 숫자를 입력하면 그 숫자만큼 데이터를 반복 수집하여 평균값을 출력한다. 예를들어 n 에 6 을 입력하면, 내부적으로 센서 데이터를 6 번 반복 수집하여 합산한후 6 으로 나눈 평균값을 출력한다. 이렇게 하면 오차가 줄어든다.

```
acc_x=-35866, acc_y=3355, acc_z=1464671, roll=15067500, pitch=126236248, yaw=-103459999
```

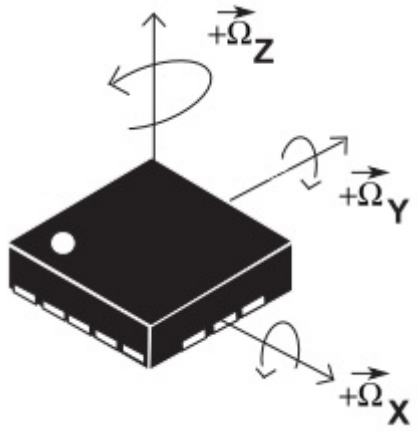
위의 데이터는 SmartSensor 보드에 있는 3 축 가속도와 3 축 기울기 센서인 LSM6DS0 칩으로부터 데이터를 수집하여 출력한 것이다. acc_x 는 X 축 방향으로 이동할 때 발생하는 가속도 데이터이고, acc_y 는 Y 축 방향으로 이동할 때 발생하는 가속도 데이터이고, acc_z 는 Z 축 방향으로 이동할 때 발생하는 가속도 데이터이다. 아래 그림은 이것을 설명하는 것이다.

3축 가속도



roll 은 X 축을 기준으로 기울어진 기울기 데이터이고, pitch 는 Y 축 기준의 기울기 데이터이고, yaw 는 Z 축 기준의 기울기 데이터이다. 아래 그림은 이것을 설명하는 것이다.

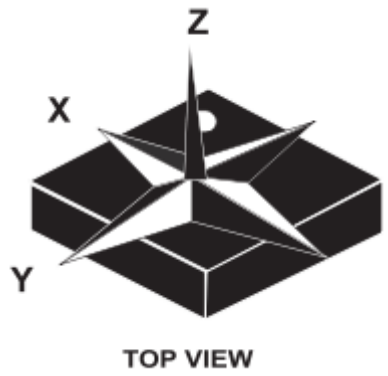
3축 기울기



다음으로 AT+SLIS? 명령어를 시리얼 터미널에 입력하면 다음과 같은 데이터를 출력한다.
AT+SLIS?n 에서 n 은 생략가능하다. n 은 데이터를 내부적으로 반복 수집하는 회수이다.

```
mag_x=146146, mag_y=-866362, mag_z=-1402036
```

위의 데이터는 SmartSensor 보드에 있는 3 축 지자계 센서인 LIS3MDL 칩으로부터 지구 자기장을 기준으로 이 칩이 놓여진 방향 데이터를 출력한 것이다. mag_x 는 X 축 기준의 방향 데이터이고, mag_y 는 Y 축 기준의 방향 데이터이고, mag_z 는 Z 축 기준의 방향 데이터이다. 아래 그림은 이것을 설명하고 있는 것이다.

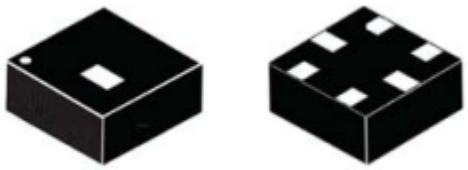


다음으로 AT+SHTS? 명령어를 시리얼 터미널에 입력하면 다음과 같은 데이터를 출력한다. AT+SHTS?n 에서 n 은 생략가능하다. n 은 데이터를 내부적으로 반복 수집하는 회수이다.

```
temper=21353, humty=58543
```

위의 데이터는 SmartSensor 보드에 있는 온도 습도 센서인 HTS221 칩으로부터 온도와 습도 데이터를 가져와서 출력한 것이다.

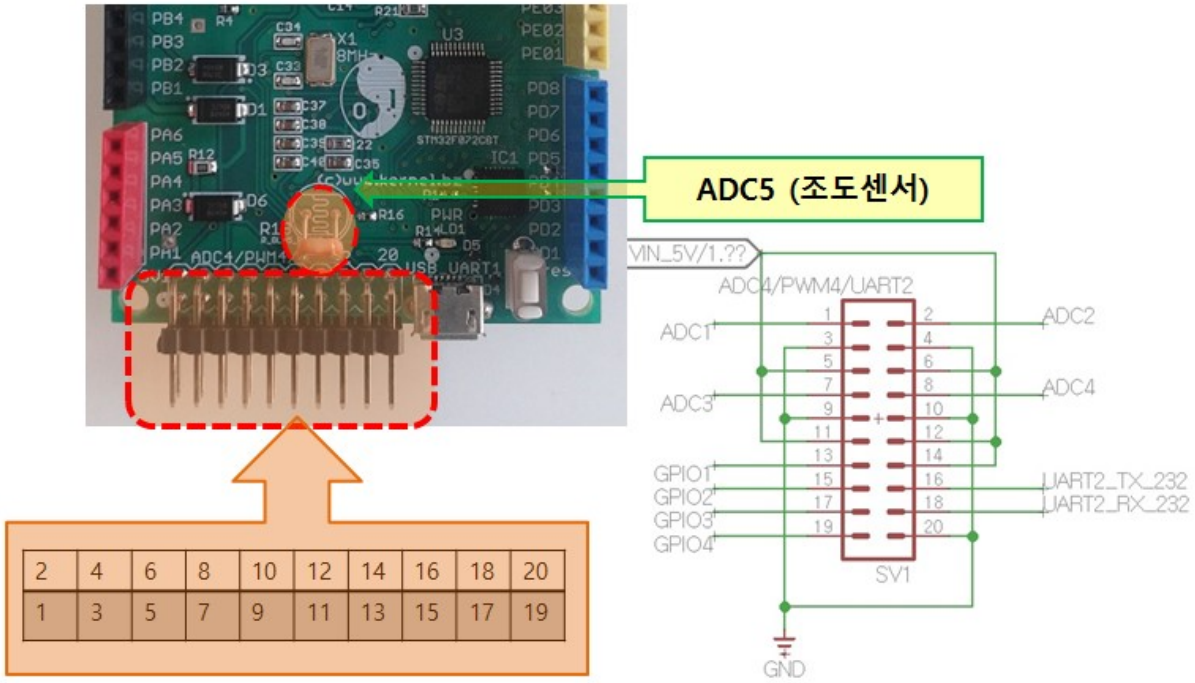
온도 습도 센서



temper 는 온도 데이터이고 humty 는 습도 데이터이다.

1.5.3 ADC 센서들의 데이터 가져오기

ADC 센서들은 아래 그림과 같이 SmartSensor 보드의 하단부에 있는 외부 확장 헤더핀(2 열 40 핀)에 연결할 수 있다. ADC 센서들의 데이터에 해당하는 ADC1, 2, 3, 4 는 외부 확장 헤더핀번호 1, 2, 7, 8 에 각각 연결하고 전원(5V)은 5, 6, 11, 12 번 핀에 연결하고 0V 그라운드는 3, 4, 9, 10 번 핀에 각각 연결하면 된다. 특별히 ADC5 는 SmartSensor 보드에서 조도(빛의 밝기)센서에 연결되어 있다.(내장)



ADC 센서들로부터 데이터를 가져오는 명령어는 다음과 같다.

```
-------- ADC Data Command -------------------------------------
AT+SADC?n       Getting ADC Data(n:0=ALL, 1=ADC1, 2=ADC2, 3=ADC3, 4=ADC4, 5=ADC5)
```

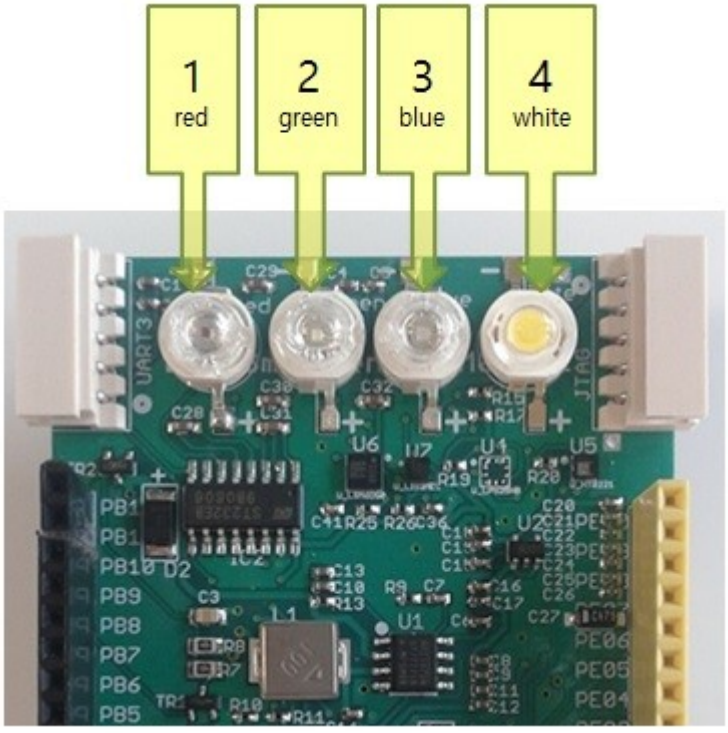
AT+SADC? 를 시리얼 터미널에 입력하면 다음과 같이 ADC 센서 번호별로 각각의 데이터가 출력된다. AT+SADC?n 에서 n 은 생략가능하다. n 은 데이터를 내부적으로 반복 수집하는 회수이다.

```
adc1=2195, adc2=1978, adc3=2071, adc4=2320, adc5=2846
```

adc1 은 ADC1 센서의 데이터이고 adc2 은 ADC2 센서의 데이터이고 adc3 은 ADC3 센서의 데이터이고 adc4 은 ADC4 센서의 데이터이고 adc5 은 ADC5 센서의 데이터이다. 특별히 adc5 는 SmartSensor 보드에 내장되어 있는 조도센서 데이터 값이다. 이값은 최소 0 부터 최대 4095 사이에서 출력된다. 빛이 밝을수록 큰값이 출력되고 빛이 어두어질수록 작은값이 출력된다.

1.5.4 고휘도 LED 스위칭

고휘도 LED 는 아주 밝은 빛을 출력한다. SmartSensor 보드 상단에 색상별로 4 개가 내장되어 있다. 첫번째 고휘도 LED 는 적색(red) 빛을 출력하고 두번째는 녹색(green), 세번째는 푸른색(blue), 네번째는 흰색(white) 빛을 출력한다.



고휘도 LED 를 색상별로 출력하는 AT 명령어는 다음과 같다.

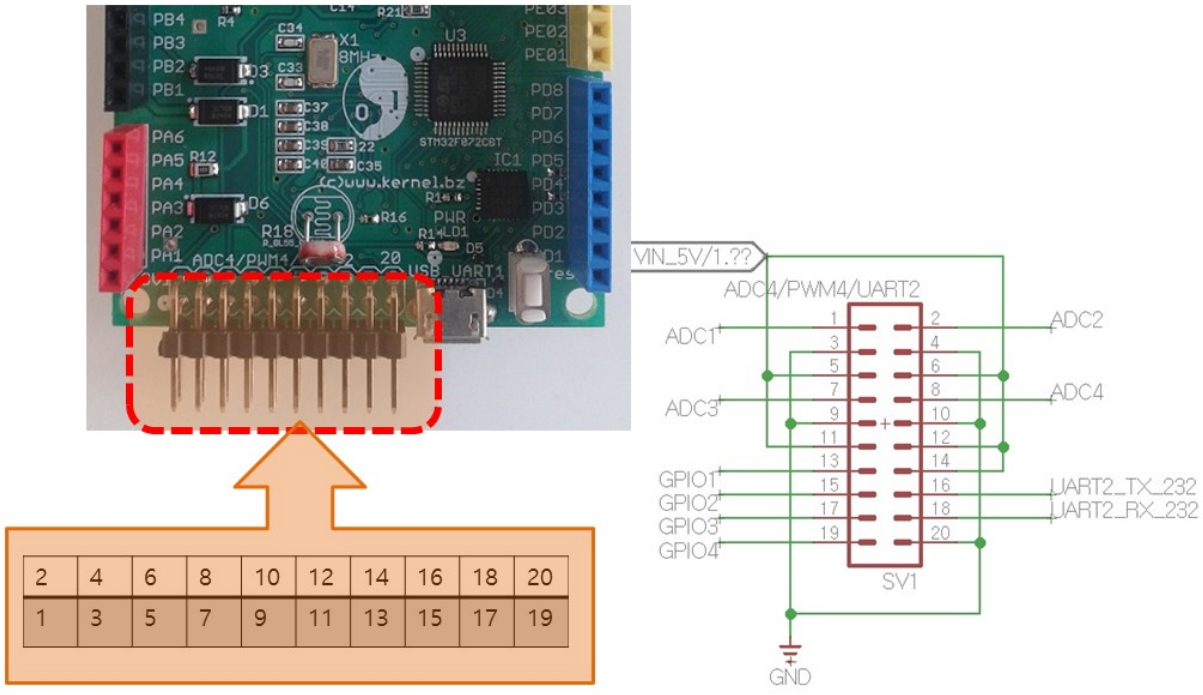
```
-------- LED Control Command ------------------------------------
AT+SLED=n,0      LED OFF(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)
AT+SLED=n,1      LED ON(n:1=(RED), 2=(GREEN), 3=(BLUE), 4=(WHITE), 5=ALL)
```

AT+SLED=n,1 명령어는 LED 을 켜는(On) 것이고 AT+SLED=n,0 명령어는 LED 를 크는(Off) 명령어이다. 여기서 n 은 LED 를 선택하는 번호이다. n 에 1 을 입력하면 첫번째 적색 LED 이고 2 는 두번째 녹색, 3 은 세번째 푸른색, 4 는 네번째 흰색 LED 을 선택하는 것이다.

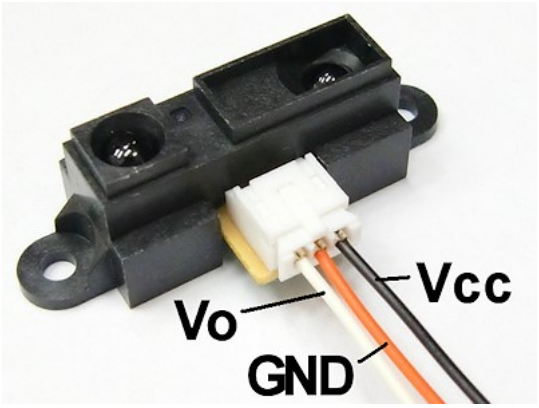
2. 센서 소스 설명

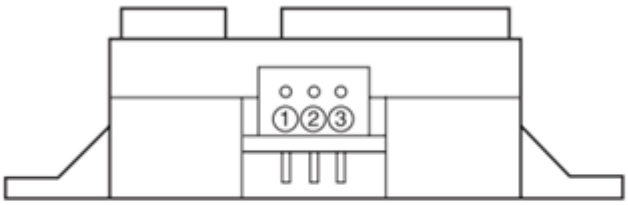
외부 센서들(적외선, 초음파)을 SmartSensor 보드 하단부에 있는 외부 확장핀(20핀: 2열 10핀)에 다음과 같이 연결한다.

SmartSensor 외부확장핀에 외부 센서들 연결



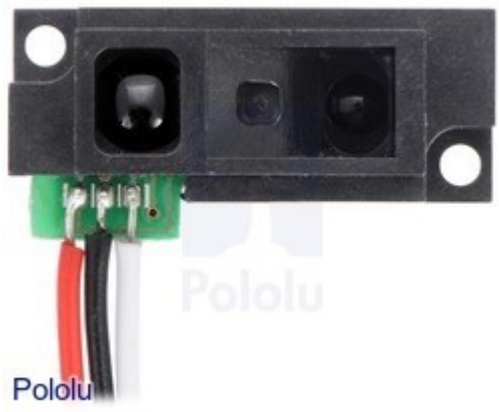
수십센티미터(10~80cm) 내의 물체를 감지하기 위해서 다음과 같은 적외선 센서(GP2Y0A21YK)를 사용한다.

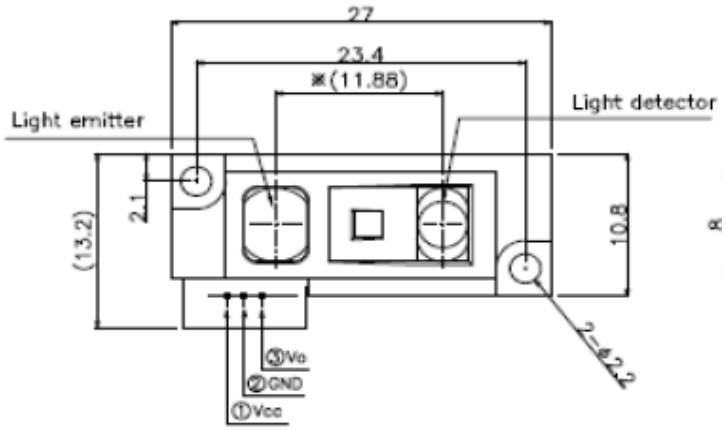




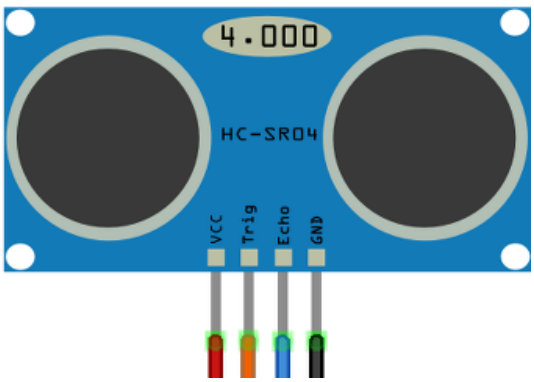
PIN	SIGNAL NAME
①	V_O
②	GND
③	V_{CC}

위의 센서 1번핀(Vo)은 SmartSensor 외부 확장핀 1번인 ADC1에 연결하고 2번핀(GND)은 3번에 연결하고 3번핀(Vcc)는 5번에 연결한다. 참고로, 좀더 짧은거리(1~8센티미터)를 감지하기 위해서 GP2Y0A51SK0F 모델를 사용한다. 아래 그림을 보면, 1번핀이 Vcc이고 2번핀은 GND, 3번핀은 Vo 이다.



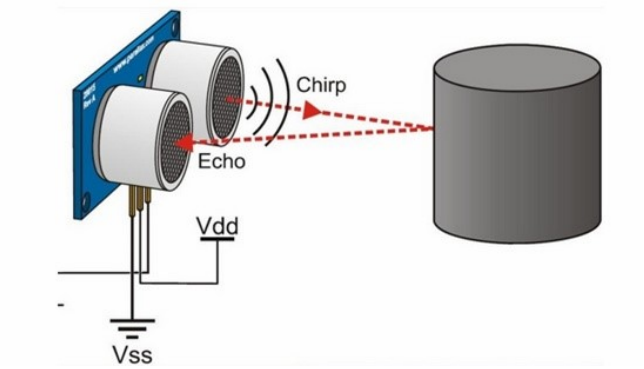


또한, 좀더 먼거리(최대 4미터)에 있는 물체를 감지하기 위해서 다음과 같은 초음파 센서(HC-SR04)를 사용한다.



이 센서의 Vcc는 SmartSensor 외부 확장핀 11번(5V)에 연결하고 GND는 9번(GND)에 연결한다.

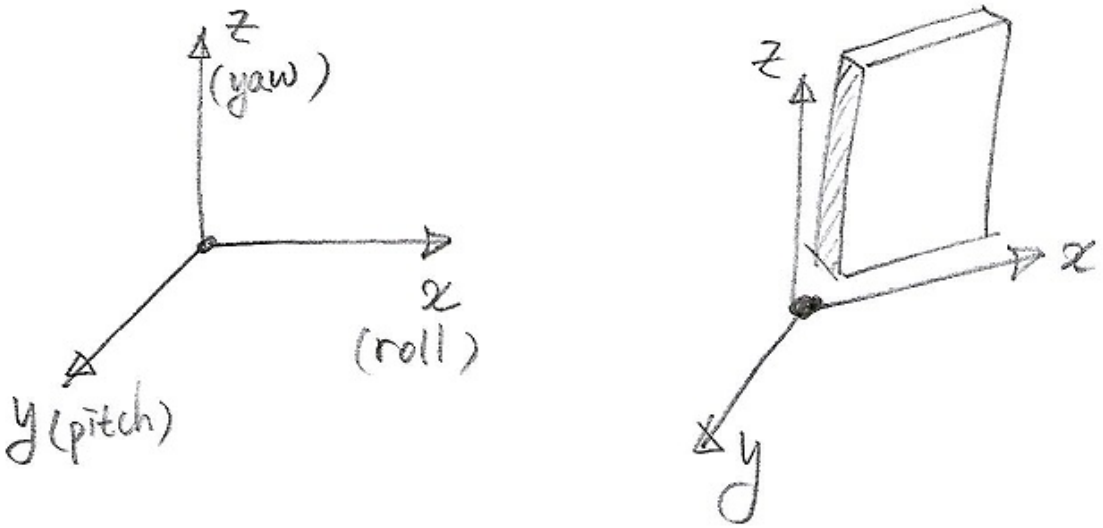
Trig(Chirp)는 초음파를 출력(발신)하고 Echo는 초음파를 입력(수신) 받는다. Trig는 13번(GPIO1)에 연결하고 Echo는 7번(ADC4)에 연결한다.



2.1 자이로 센서 데이터 처리

자이로 센서는 아래 그림과 같이 X, Y, Z축 세가지 방향에 대해서 기울기 값을 출력한다.

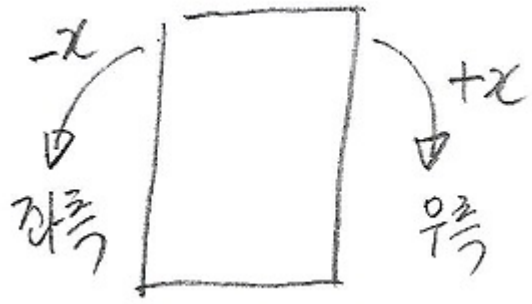
3축 기울기



x축은 아래 그림과 같이 좌우측면으로 기울기를 측정한다. 우측 방향은 x축의 기울기 값이 양수로 증가하고 좌측 방향으로는 x축의 기울기 값이 음수로 증가한다. 그리고 x축 방향으로 값이 변화는 각속도(angular rate)를 roll 이라 한다. X축 방향으로 기울기가 빠르게 변화면 roll은 증가하고 기울기가 천천히 변화면 roll은 감소한다. Roll은 기울기가 변화는 순간의 속도 변화량인

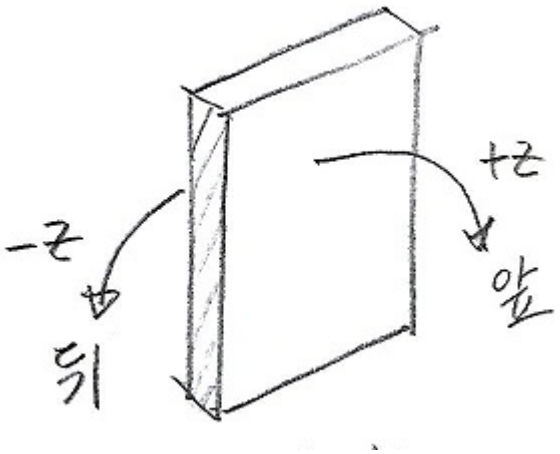
가속도로 표현된다.

X축 방향의 기울기 변화



z축은 아래 그림과 같이 앞뒤(전후방)로 기울기를 측정한다. 앞(전방) 방향은 z축의 기울기 값이 양수로 증가하고 뒤(후방) 방향으로는 z축의 기울기 값이 음수로 증가한다. 그리고 z축 방향으로 값이 변화는 각속도(angular rate)를 yaw라고 한다. z축 방향으로 기울기가 빠르게 변화면 yaw는 증가하고 기울기가 천천히 변화면 yaw는 감소한다. Yaw는 기울기가 변화는 순간의 속도 변화량인 가속도로 표현된다.

Z축 방향의 기울기 변화



Y축 방향은 z축 방향이 180도 이상 완전히 뒤집어진 경우에 의미있는 값을 가진다. 직립형태로 구동하는 2륜구동 로봇에서 180도 뒤집어지면 물구나무를 서서 이동하는 형태가 되므로 y축 방향에 대한 설명은 논외로 제외 하기로 한다.

직립형태의 이륜 구동 로봇에서 z축 방향의 기울기 값이 가장 의미있는 중요한 데이터이다. 우리 사람도 직립하여 걸어갈 때 앞뒤로 넘어지지 않는 방향으로 가장 균형을 유지하려고 노력한다.

실제 구현에서도 z방향의 기울기 데이터를 가장 가중치 있게 프로그래밍 하였다.

아래는 실제로 측정한 기울기 값을 테이블로 정리한 것이다.

X축 방향의 실측 기울기 값

각도	우측방향 기울기 값(+x)	좌측방향 기울기 값(-x)
10도	180	-210
20도	250	-420
45도	520	-750
90도	980	-1020

X축 방향의 roll은 평균적으로 7435 값을 기준으로 변화량이 측정 된다.

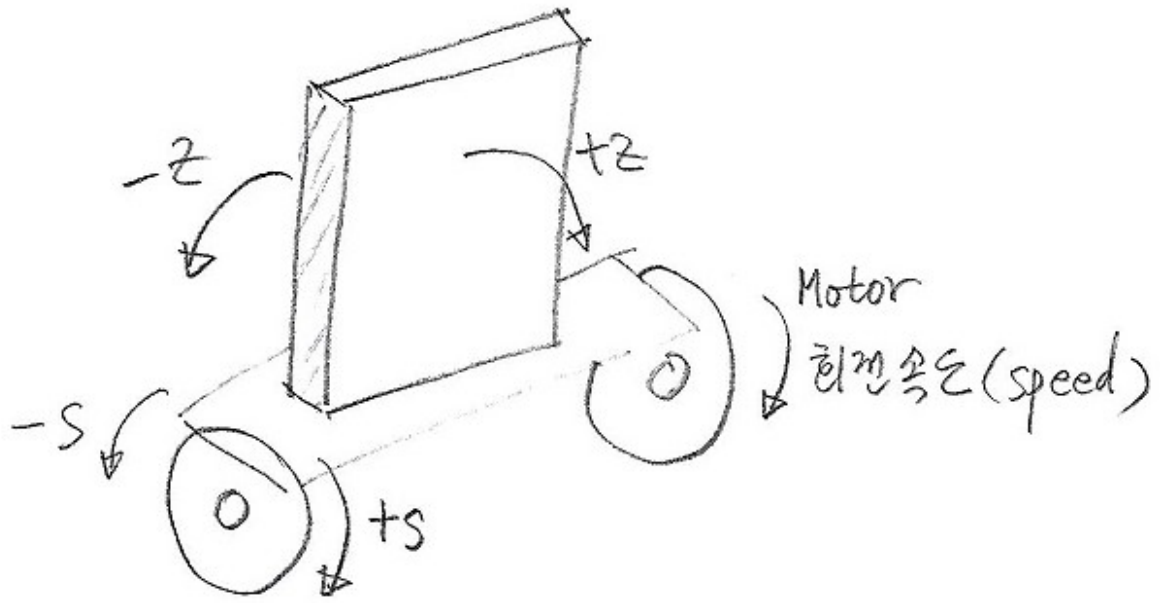
Z축 방향의 실측 기울기 값

각도	우측방향 기울기 값(+z)	좌측방향 기울기 값(-z)
10도	160	-200
20도	220	-400
45도	420	-700
90도	960	-1040

Z축 방향의 yaw는 평균적으로 1120 값을 기준으로 변화량이 측정된다.

위의 실측 데이터를 바탕으로 아래와 같이 이륜 구동 로봇이 넘어지지 않고 직립을 유지하기 위해서는 기울기 값과 이것의 변화량(angular rate) 사이에서 적절한 수식을 도입하여 모터 속도를 제어해야 한다.

이륜 직립유지 로봇



위의 그림에서 이륜구동 로봇이 직립을 유지하기 위해서는 +z방향으로 기울기 변화량에 비례하는 만큼 모터를 +s 속도로 제어하고, -z방향으로 기울기 변화량에 비례하는 만큼 모터를 –s 속도로 제어해야 한다.

위의 관계를 심사숙고하여 실제 프로그래밍 코드에 적용한 내용을 수식으로 정리하면 다음과 같다.

z방향의 모터속도 제어 수식

V = (1120 – yaw) / 9000

S = 2 + Z / 7 + V

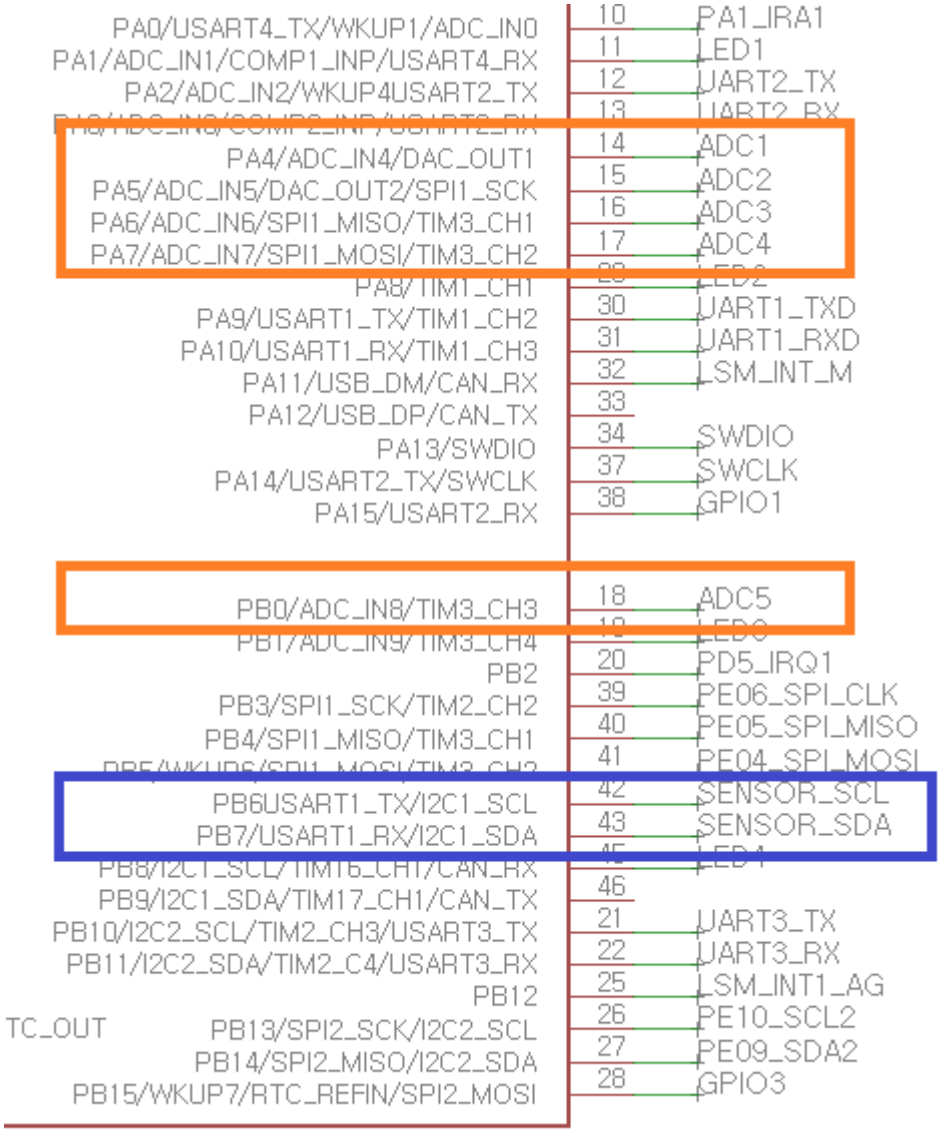
위에서 V는 z축 방향의 yaw가 평균적으로 1120 값을 기준으로 변화는 변화량을 수식으로 계산한 값이다. Z는 z축 방향으로 기울어진 기울기 값이고 S는 위의 내용을 수식으로 적용하여 산출한 모터 제어 속도이다.

이와 같이 산출한 S값을 모터 속도 제어값으로 사용했다. 위의 수식은 프로그램 코드를 사용하여 여러 번 테스트를 진행하면서 경험적으로 찾아낸 수식이다. 따라서 이것에는 오차가 있을 수 있으며 앞으로 더 효율적인 수식을 찾아낼 수도 있을 것이다.

그리고, 2륜 구동 로봇의 직립성을 정확히 높이기 위해서는 기울기 값과 변화량에 따라서 모터 속도를 제어하는 수식도 중요하지만 모터의 물리적인 반응속도(민첩성)도 중요하다.

그동안 경험으로 비추어볼 때, 최적의 데이터를 산출하기 위한 수식의 정확성, 이것을 빨리 계산할 수 있는 CPU 성능, 모터 반응속도(민첩성), 로봇 기구물의 무게중심 등의 요소들을 모두 고려해야 최적의 해답을 찾을 수 있다.

2.2. 센서 핀맵 확인



2.3 센서 핀맵 설정 소스

2.3.1 ADC 설정

```
#ifdef USER_CONFIG_ADC_DMA
    //SmartPower: ADC(PA1,2,3,4)
    usr_adc_dma_init();
    user_debug("usr_adc_dma_init(): [OK]");
#endif


void usr_adc_dma_init(void)
{
  ADC_Std_InitTypeDef     ADC_InitStructure;
  DMA_Std_InitTypeDef     DMA_InitStructure;

  /* ADC1 DeInit */
  ADC_DeInit(ADC1);

  /* ADC1 Periph clock enable */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  /* DMA1 clock enable */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1 , ENABLE);

  /* DMA1 Channel1 Config */
  DMA_DeInit(DMA1_Channel1);
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)ADC1_DR_Address;
  DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)RegularConvData_Tab;
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  DMA_InitStructure.DMA_BufferSize = ADC_CNT;
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
```

```
DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
DMA_InitStructure.DMA_Priority = DMA_Priority_High;
DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_Init(DMA1_Channel1, &DMA_InitStructure);

/* DMA1 Channel1 enable */
DMA_Cmd(DMA1_Channel1, ENABLE);

/* ADC DMA request in circular mode */
ADC_DMARequestModeConfig(ADC1, ADC_DMAMode_Circular);

/* Enable ADC_DMA */
ADC_DMACmd(ADC1, ENABLE);

/* Initialize ADC structure */
ADC_StructInit(&ADC_InitStructure);

/* Configure the ADC1 in continous mode withe a resolutuion equal to 12 bits  */
ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_ScanDirection = ADC_ScanDirection_Backward;
ADC_Init(ADC1, &ADC_InitStructure);

ADC_ChannelConfig(ADC1, ADC_Channel_4 , ADC_SampleTime_55_5Cycles);
ADC_ChannelConfig(ADC1, ADC_Channel_5 , ADC_SampleTime_55_5Cycles);
ADC_ChannelConfig(ADC1, ADC_Channel_6 , ADC_SampleTime_55_5Cycles);
ADC_ChannelConfig(ADC1, ADC_Channel_7 , ADC_SampleTime_55_5Cycles);
ADC_ChannelConfig(ADC1, ADC_Channel_8 , ADC_SampleTime_55_5Cycles);

/* ADC Calibration */
ADC_GetCalibrationFactor(ADC1);
```

```
  /* Enable ADC1 */
  ADC_Cmd(ADC1, ENABLE);

  /* Wait the ADCEN falg */
  while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_ADEN));

  /* ADC1 regular Software Start Conv */
  ADC_StartOfConversion(ADC1);
}
```

2.3.2 ADC 센서값 읽기

```
//IR1 Sensor
//5cm:3800, 10cm:3200, 15cm:2200, 20cm:1800
void usr_adc1_get(uint8_t *data)
{
    usr_adc_dma_read();
    sprintf((char *)data, "%u", ADC_Data.adc1);
}
//Call by Timer6
uint32_t usr_adc1_get_int(void)
{
    //Test DMA1 TC flag
    while((DMA_GetFlagStatus(DMA1_FLAG_TC1)) == RESET);
    //Clear DMA TC flag
    DMA_ClearFlag(DMA1_FLAG_TC1);

    return RegularConvData_Tab[0];   //ADC_Data.adc1
}

//IR2 Sensor
void usr_adc2_get(uint8_t *data)
{
```

```
    usr_adc_dma_read();
    sprintf((char *)data, "%u", ADC_Data.adc2);
}

void usr_adc3_get(uint8_t *data)
{
    usr_adc_dma_read();
    sprintf((char *)data, "%u", ADC_Data.adc3);
}

void usr_adc4_get(uint8_t *data)
{
    usr_adc_dma_read();
    sprintf((char *)data, "%u", ADC_Data.adc4);
}

//CDS Sensor: Dark Min:0, Normal:3500, Bright Max:4095
void usr_adc5_get(uint8_t *data)
{
    usr_adc_dma_read();
    sprintf((char *)data, "%u", ADC_Data.adc5); //cds
}

//Call by Timer6
uint32_t usr_adc5_get_int(void)
{
    //Test DMA1 TC flag
    while((DMA_GetFlagStatus(DMA1_FLAG_TC1)) == RESET);
    //Clear DMA TC flag
    DMA_ClearFlag(DMA1_FLAG_TC1);

    return RegularConvData_Tab[1];   //ADC_Data.adc5
}

void usr_adc_all_get(uint8_t *data)
{
```

```
    usr_adc_dma_read();
    sprintf((char *)data, "adc1=%u, adc2=%u, adc3=%u, adc4=%u, adc5=%u"
            , ADC_Data.adc1, ADC_Data.adc2, ADC_Data.adc3, ADC_Data.adc4, ADC_Data.adc5);
}
```

2.3.3 I2C 설정

```
#ifdef USER_CONFIG_I2C
    //I2C1(PB6, PB7)
    if (user_i2c_init() < 0) {
        Error_Handler();
        user_debug_error("user_i2c_init(I2C1): [FAIL]");
        err++;
    } else {
        user_debug("user_i2c_init(I2C1): [OK]");
    }
#endif

int32_t user_i2c_init(void)
{
  /*##-1- Configure the I2C peripheral ######################################*/
  I2cHandle.Instance             = I2C1;

  I2cHandle.Init.Timing          = I2C_TIMING;

  I2cHandle.Init.AddressingMode  = I2C_ADDRESSINGMODE_7BIT;

  I2cHandle.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  I2cHandle.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  I2cHandle.Init.NoStretchMode   = I2C_NOSTRETCH_DISABLE;

  I2cHandle.Init.OwnAddress1     = I2C_ADDRESS;

  if(HAL_I2C_Init(&I2cHandle) != HAL_OK)
```

```
  {
    /* Initialization Error */
    _i2c_error_handler();
    return -1;
  }

  /* Enable the Analog I2C Filter */
  HAL_I2CEx_ConfigAnalogFilter(&I2cHandle,I2C_ANALOGFILTER_ENABLE);

  return 0;
}
```

2.3.4 I2C 센서값 읽기

```
Sensor_T *SensorData; ///sensor data

uint8_t sensor_lsm_get(uint8_t *buf, uint32_t num)
{
    memset(SensorData, 0, sizeof(Sensor_T));
    lsm6ds0_read(LSM6DS0_ADDRESS, SensorData, num);

    user_debug_sensor("Count=%d: acc_x=%d, acc_y=%d, acc_z=%d, roll=%d, pitch=%d, yaw=%d"
        , num, SensorData->acc_x, SensorData->acc_y, SensorData->acc_z, SensorData->roll,
SensorData->pitch, SensorData->yaw );

    sprintf((char *)buf, "acc_x=%d, acc_y=%d, acc_z=%d, roll=%d, pitch=%d, yaw=%d"
        , SensorData->acc_x, SensorData->acc_y, SensorData->acc_z, SensorData->roll,
SensorData->pitch, SensorData->yaw );

    return SENSOR_OK;
}

static uint8_t _sensor_lsm_read(Sensor_T *sdata, uint32_t num)
```

```
{
    lsm6ds0_read(LSM6DS0_ADDRESS, sdata, num);

    user_debug_sensor("Count=%d: acc_x=%d, acc_y=%d, acc_z=%d, roll=%d, pitch=%d, yaw=%d"
                    , num, sdata->acc_x, sdata->acc_y, sdata->acc_z
                    , sdata->roll, sdata->pitch, sdata->yaw );

    return SENSOR_OK;
}

uint8_t sensor_lis_get(uint8_t *buf, uint32_t num)
{
    memset(SensorData, 0, sizeof(Sensor_T));
    lis3mdl_read(LIS3MDL_ADDRESS, SensorData, num);

    user_debug_sensor("Count=%d: mag_x=%d, mag_y=%d, mag_z=%d"
                    , num, SensorData->mag_x, SensorData->mag_y, SensorData->mag_z );

    sprintf((char *)buf, "mag_x=%d, mag_y=%d, mag_z=%d"
                    , SensorData->mag_x, SensorData->mag_y, SensorData->mag_z );

    return SENSOR_OK;
}

static uint8_t _sensor_lis_read(Sensor_T *sdata, uint32_t num)
{
    lis3mdl_read(LIS3MDL_ADDRESS, sdata, num);

    user_debug_sensor("Count=%d: mag_x=%d, mag_y=%d, mag_z=%d₩r₩n"
                    , num, sdata->mag_x, sdata->mag_y, sdata->mag_z);

    return SENSOR_OK;
}

uint8_t sensor_hts_get(uint8_t *buf, uint32_t num)
{
```

```
    memset(SensorData, 0, sizeof(Sensor_T));
    hts221_read(HTS221_ADDRESS, SensorData, num);

    user_debug_sensor("Count=%d: temper=%d, humty=%d", num, SensorData->temper,
SensorData->humty);

    sprintf((char *)buf, "temper=%d, humty=%d", SensorData->temper, SensorData->humty);

    return SENSOR_OK;
}

static uint8_t _sensor_hts_read(Sensor_T *sdata, uint32_t num)
{
    hts221_read(HTS221_ADDRESS, sdata, num);

    user_debug_sensor("Count=%d: temper=%d, humty=%d", num, sdata->temper, sdata->humty);

    return SENSOR_OK;
}
```