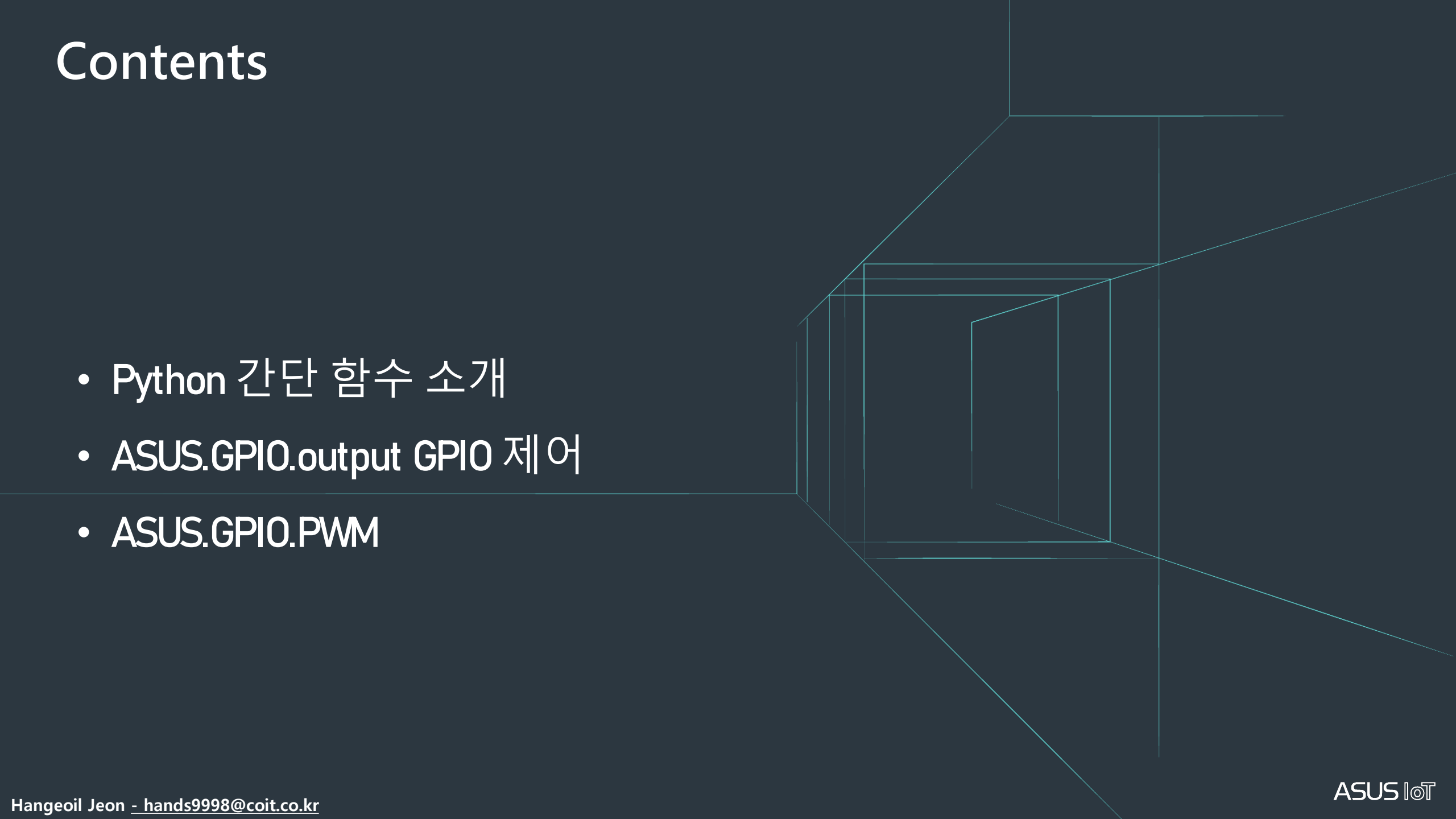
ASUS IoT

Tinker Board

IoT를 위한 최선의 엣지 단말기

Kun Woo Park – poli52ppp@gmail.com

Contents



- Python 간단 함수 소개
- ASUS.GPIO.output GPIO 제어
- ASUS.GPIO.PWM

Print 함수

print 함수에 대한 사용법 학습
print 함수는 문자열과 숫자를 출력해 주는 함수
파이썬에서 문자열을 출력하고자 할 경우엔 print 함수 사용

Nano를 이용해 하단 예제 작성 (파일명 : 01_print.py)

```
print("Hello. I'm a Tinker Board~")
```

다음과 같이 예제를 실행

```
$ python 01_print.py
```

다음과 같은 문자열이 출력

```
Hello. I'm a Tinker Board~
```

while 함수

파이썬에서 어떤 일을 반복하고자 할 경우엔 while 문을 사용

Nano를 이용해 하단 예제 작성 (파일명 : 02_while_true.py)

```
1 while True:
2         print("Hello. I'm a Tinker Board~")
```

여기서는 print 함수가 while 문의 일부로 동작
다음과 같이 예제를 실행

```
$ python 02_while_true.py
```

다음과 같은 문자열이 빠른 속도로 무한 출력

```
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
```

while 함수

프로그램을 강제 종료하기 위해서는 CTRL 키를 누른 채로 C 키 입력
키보드 인터럽트가 발생했다는 메시지와 함께 탈출 가능

```
^CHello. I'm a Tinker Board~
Traceback (most recent call last):
  File "_02_while_true.py", line 2, in <module>
    print("Hello. I'm a Tinker Board~")
KeyboardInterrupt
```

인터럽트 처리 메시지를 보이지 않도록 하기 위해서는 키보드 인터럽트를 직접 처리하는 법을 다음 장에서 배울 예정

try~except 함수

키보드 인터럽트를 처리하기 위해 try~except문 학습
Nano를 이용해 하단 예제 작성 (파일명 : 03_try_except.py)

```
1 try:
2       while True:
3                   print("Hello. I'm a Tinker Board~")
4 except:
5           pass
```

키보드 인터럽트를 처리하기 위해 try~except 문을 사용
try~except 문은 예외 처리를 하고자 할 경우 사용
2 : while 문을 try 문보다 탭 문자 하나만 큼 들여 써서 try문 안에서 동작
5 : pass문을 이용하여 아무것도 수행하지 않음

try~except 함수

다음과 같이 예제를 실행

```
$ python 03_try_except.py
```

다음과 같은 문자열이 빠른 속도로 무한 출력

```
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
```

프로그램을 강제 종료하기 위해서는 CTRL 키를 누른 채로 C 키 입력
키보드 인터럽트에 대해 pass 문을 이용하여 글 없이 종료

```
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
^CHello. I'm a Tinker Board~
```

time.sleep 함수

시간에 대한 지연을 주고자 할 경우엔 time 라이브러리의 sleep 함수를 사용
Nano를 이용해 하단 예제 작성 (파일명 : 04_time.py)

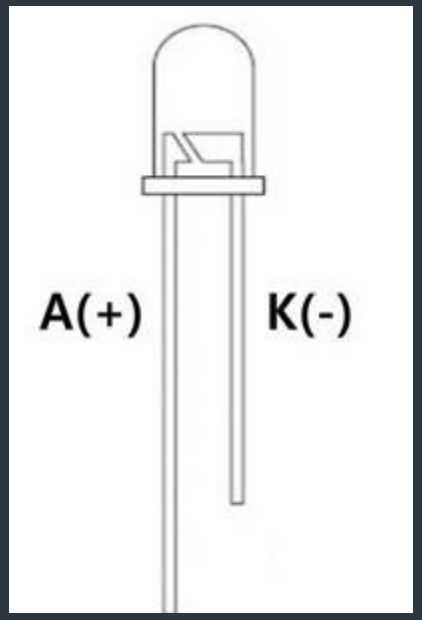
```
1 import time
2
3 try:
4       while True:
5               print("Hello. I'm a Tinker Board~")
6               time.sleep(0.5)
7 except:
8           pass
```

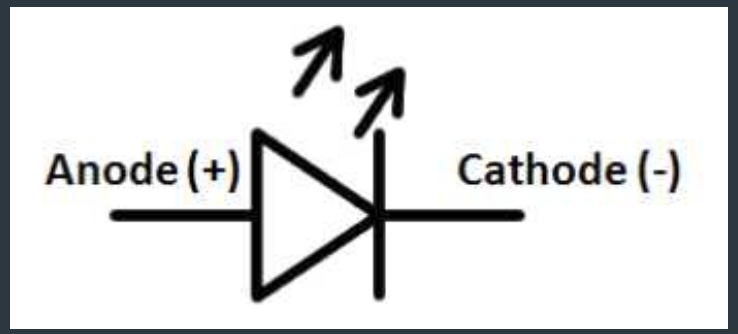
1 : time 모듈을 호출 6줄에서 time 모듈이 제공하는 sleep 함수를 사용하기 위해 필요
6 : time 모듈이 제공하는 sleep 함수를 호출하여 0.5초간 지연
프로그램을 실행시키면 문자열이 0.5초마다 반복해서 출력

```
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
Hello. I'm a Tinker Board~
```

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어

- LED는 방향성이 있음 즉, 회로에 연결할 때 방향을 고려
- 발광 다이오드는 아래와 같이 긴 pin이 +(Anode) 짧은 pin이-(cathode)
- 일반적인 발광다이오드의 구동 전압 1.8V~2.0V 소모 전류는 20~30mA
- GPIO은 3.3V로 구동된다.

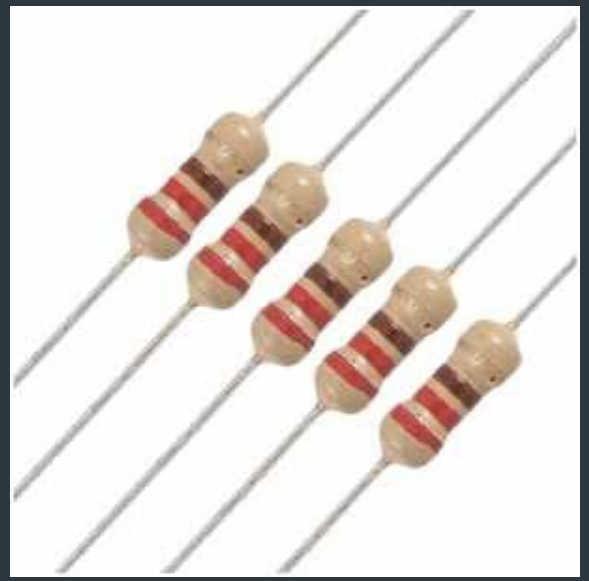


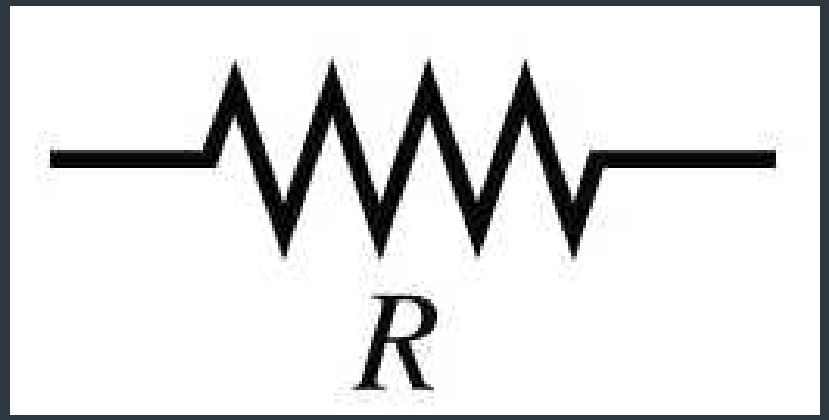


Kun Woo Park – poli52ppp@gmail.com

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어

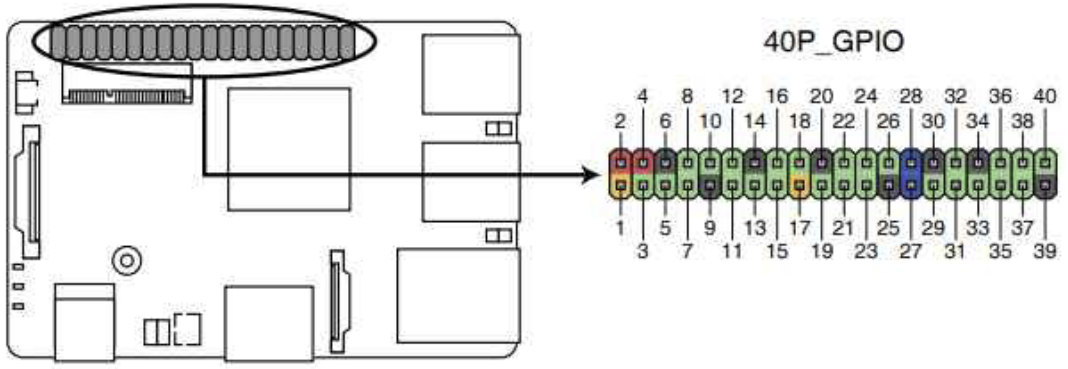
- 하단 저항은 220 Ohm 저항
- 저항은 전류의 양을 조절하는 역할
- 저항은 방향성이 없기 때문에 VCC와 GND에 어떤 방향으로도 연결 가능





Kun Woo Park – poli52ppp@gmail.com

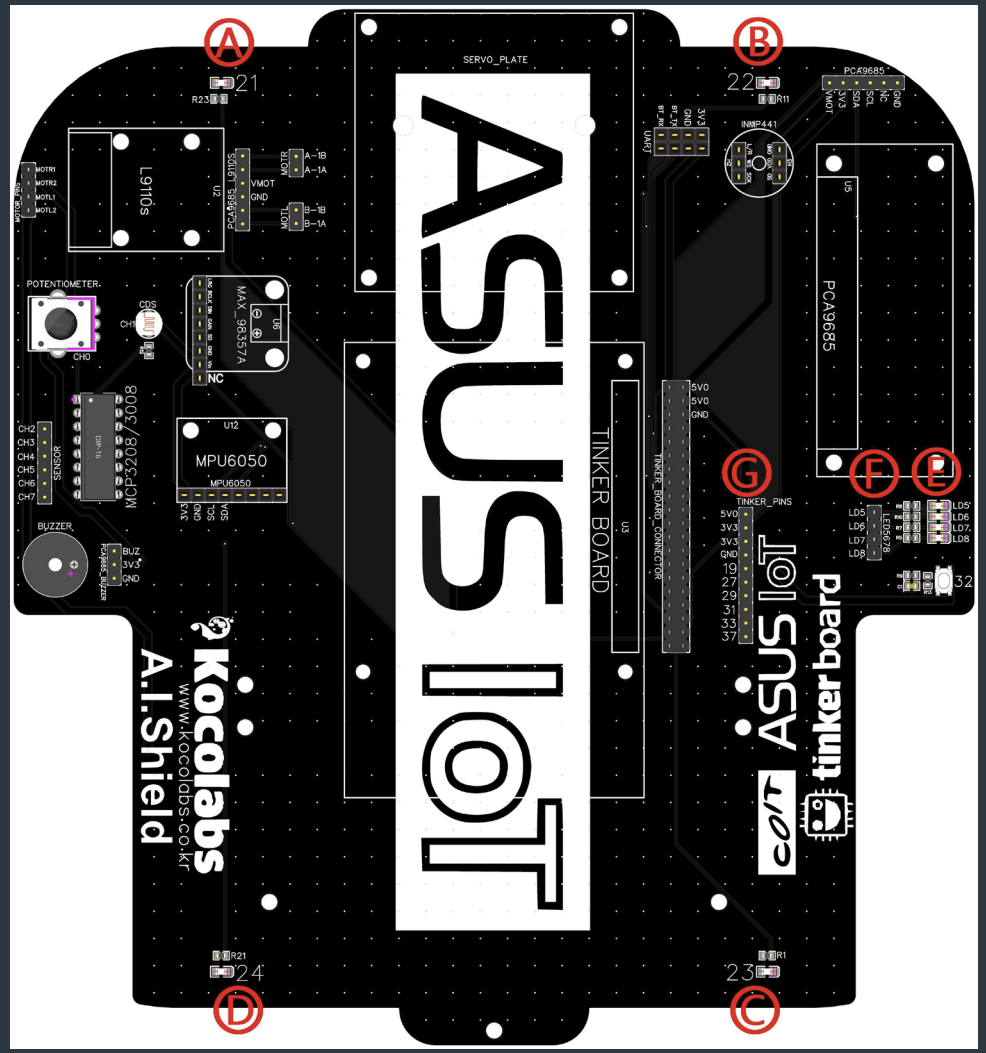
ASUS.GPIO.output를 통해 LED(발광다이오드) 제어



Pin definition	40P GPIO		Pin definition
VCC3.3_IO	1	2	VCC5V
GPIO2_B1/I2C6_SDA	3	4	
GPIO2_B2/I2C6_SCL	5	6	GND
GPIO0_B0/CLKOUT	7	8	GPIO2_C1/UART0_TX
GND	9	10	GPIO2_C0/UART0_RX
GPIO2_C3/UART0_RTSN	11	12	GPIO3_D1/I2S0_SCLK
GPIO2_C5/SPI5_TXD	13	14	GND
GPIO2_C4/SPI5_RXD	15	16	GPIO2_C6/SPI5_CLK
VCC3.3_IO	17	18	GPIO2_C7/SPI5_CSN0
GPIO1_B0/SPI1_TXD/UART4_TX	19	20	GND
GPIO1_A7/SPI1_RXD/UART4_RX	21	22	GPIO3_D4/I2S0_SDI1SDO3
GPIO1_B1/SPI1_CLK	23	24	GPIO1_B2/SPI1_CSN0
GND	25	26	GPIO0_A6/PWM3A_IR
GPIO2_A7/I2C7_SDA	27	28	GPIO2_B0/I2C7_SCL
GPIO3_D6/I2S0_SDI1SDO1	29	30	GND
GPIO3_D5/I2S0_SDI1SDO2	31	32	GPIO4_C2/PWM0
GPIO4_C6/PWM1	33	34	GND
GPIO3_D1/I2S0_LRCK	35	36	GPIO2_C2/UART0_CTSN
GPIO4_C5/SPDIF_TX	37	38	GPIO3_D3/I2S0_SDI0
GND	39	40	GPIO3_D7/I2S0_SDO0

- ASUS.GPIO.output 함수를 통해 제어할 수 있는 핀은 초록색으로 표시된 GPIO 핀
- Board SOC(System on Chip) 내부에 있는 CPU, UART, I2C, Timer 등의 하드웨어 모듈에 할당하여 사용

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어



- LD1, LD2, LD3, LD4의 경우 각각 팅커 보드의 21, 22, 23, 24 번 핀으로 연결되고 LD5, LD6, LD7, LD8의 경우 Ⓕ 부분의 확장 핀으로 연결

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어

ASUS.GPIO.output 함수를 이용하여 LED를 제어
Nano를 이용해 하단 예제 작성 (파일명 : 06_gpio_output.py)

```
01 import ASUS.GPIO as GPIO
02 import sys
03 import time
04
05 led_pin = int(sys.argv[1])
06
07 GPIO.setmode(GPIO.BOARD)
08 GPIO.setwarnings(False)
09
10 GPIO.setup(led_pin, GPIO.OUT)
11
12 GPIO.output(led_pin, True)
13 time.sleep(2.0)
14 GPIO.output(led_pin, False)
15
16 GPIO.cleanup()
```

1 : ASUS.GPIO 모듈을 GPIO라는 이름으로 로드
2 : sys 모듈을 로드
3 : 13줄에서 sleep 함수를 사용하기 위하여 time 모듈 로드
7 : GPIO.setmode 함수를 호출하여 BOARD 핀 번호를 사용하도록 설정
8 : GPIO.setwarnings 함수에 False 값을 인자로 주어 경고 메시지 출력을 막음
10 : GPIO.setup 함수를 호출하여 led_pin을 GPIO.OUTPUT으로 설정
12 : GPIO.output 함수를 호출하여 led_pin을 True로 설정
16 : GPIO.cleanup 함수를 호출하여 GPIO 핀의 상태를 초기화

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어

다음과 같이 예제를 실행

```
$ python 06_gpio_output.py 21
```

명령행의 3 번째에 오는 숫자 21은 21 번 핀을 의미
21 번 핀에 연결된 Ⓐ 부분의 LED가 켜졌다 2.0초 후에 꺼지는 것을 확인

21을 다른 핀 값으로 변경하여 테스트

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어

ASUS.GPIO.output 함수를 이용하여 LED를 제어
Nano를 이용해 하단 예제 작성 (파일명 : 06_gpio_output_2.py)

```
01 import ASUS.GPIO as GPIO
02 import sys
03 import time
04
05 led_pin = int(sys.argv[1])
06 DELAY = float(sys.argv[2])
07
08 GPIO.setmode(GPIO.BOARD)
09 GPIO.setwarnings(False)
10
11 GPIO.setup(led_pin, GPIO.OUT)
12
13 try:
14     while True:
15         GPIO.output(led_pin, True)
16         time.sleep(DELAY)
17         GPIO.output(led_pin, False)
18         time.sleep(DELAY)
19 except:
20     pass
21
22 GPIO.cleanup()
```

6 : DELAY 변수를 선언, 프로그램 실행 시 넘겨받게 될 2 번 인자를 실수로 변환한 값으로 초기화
16,18 : time.sleep 함수를 호출하여 DELAY 초간 지연

ASUS.GPIO.output를 통해 LED(발광다이오드) 제어

다음과 같이 예제를 실행

```
$ python 06_gpio_output_2.py 21 0.5
```

스크립트의 1 번 인자로 21, 2 번 인자로 0.5 입력
1초 주기로 21 번 핀에 연결된 LED가 켜졌다 꺼졌다 하는 것을 확인
0.5는 1Hz

LED 점멸 간격 줄여보기

```
$ python 06_gpio_output_2.py 21 0.05
```

스크립트의 1 번 인자로 21, 2 번 인자로 0.05 입력
0.1초 주기로 21 번 핀에 연결된 LED가 켜졌다 꺼졌다 하는 것을 확인
0.05는 10Hz

ASUS.GPIO.PWM를 통해 PWM 학습

ASUS.GPIO.PWM 클래스를 이용하여 LED 점멸을 반복
Nano를 이용해 하단 예제 작성 (파일명 : 07_pwm_output.py)

```
01 import ASUS.GPIO as GPIO
02 import sys
03
04 led_pin = int(sys.argv[1])
05 FREQUENCY = float(sys.argv[2])
06
07 GPIO.setmode(GPIO.BOARD)
08 GPIO.setwarnings(False)
09
10 GPIO.setup(led_pin, GPIO.OUT)
11
12 pwm = GPIO.PWM(led_pin,
FREQUENCY)
13 pwm.start(50.0) # 0.0~100.0
14
15 try:
```

```
16        while True:
17                pass
18 except:
19        pass
20
21 pwm.stop()
22 GPIO.cleanup()
```

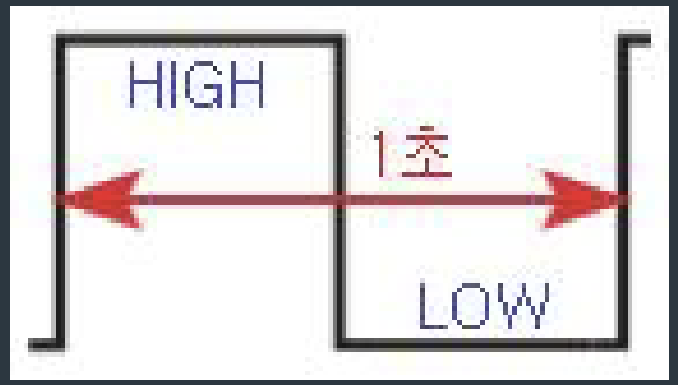
5 : FREQUENCY 변수를 선언한 후, 실행 시 넘겨 받게 될 2 번째 인자를 실수로 변환하여 초기화
12 : GPIO.PWM 객체를 하나 생성한 후, pwm 변수에 할당
GPIO.PWM 객체 생성 시, 1 번 인자는 핀 번호가 되며, 2 번 인자는 주파수 값

ASUS.GPIO.PWM를 통해 PWM 학습

다음과 같이 예제를 실행

```
$ python 07_pwm_output.py 21 1
```

1번 인자로 21(번 핀)을, 2번 인자로 1(Hz)를 입력
1초 주기로 LED가 점멸 하는 것을 확인



GPIO.PWM 도움말 보기

GPIO.PWM 모듈에 대해 파이썬 쉘을 이용하여 확인

```
linaro@linaro-alip:~/pyLabs$ python 1
Python 3.7.3 (default, Oct 31 2022, 14:04:00)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import ASUS.GPIO as GPIO 2
>>> help(GPIO.PWM) 3
```

스페이스 바를 통해 다음 화면으로 넘어가기
q 문자를 통해 도움말 빠져나오기
파이썬 쉘을 빠져 나올 때는 다음과 같이 quit() 함수를 수행

GPIO.PWM 도움말 보기

```
Help on class PWM: (1)

class PWM(builtins.object) (2)
 |  Pulse Width Modulation class (3)
 |
 |  Methods defined here: (4)
 |
 |  ChangeDutyCycle(...) (5)
 |      Change the duty cycle
 |      dutycycle - between 0.0 and 100.0
 |
 |  ChangeFrequency(...) (6)
 |      Change the frequency
 |      frequency - frequency in Hz (freq > 1.0)
 |
 |  __init__(self, /, *args, **kwargs) (7)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  start(...) (8)
 |      Start software PWM
 |      dutycycle - the duty cycle (0.0 to 100.0)
 |
 |  stop(...)
:
```

GPIO.PWM 도움말 보기

```
 |
 |  stop(...) 1
 |      Stop software PWM
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here: 2
 |
 |  __new__(*args, **kwargs) from builtins.type 3
 |      Create and return a new object.  See help(type) for accurate signature.
~
(END)
```