GPIO

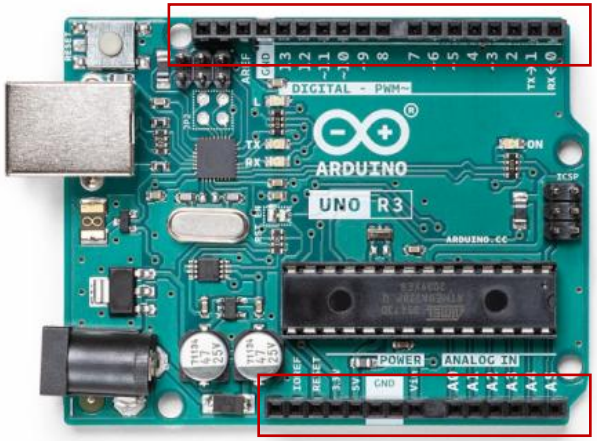
GPIO

- ## GPIO(General Purpose Input/Output)란?
- GPIO는 컴퓨터와 외부 장치 간에 데이터를 주고받기 위한 범용 입출력 인터페이스로, 마이크로프로세서와 마이크로컨트롤러 같은 임베디드 하드웨어에서 디지털 또는 아날로그 신호를 주고받기 위해 사용될 수 있다. 예를 들어, 라즈베리파이 4 모델 B에는 총 40개의 GPIO 핀이 있어 다양한 외부 장치와 연결할 수 있다.
- ## GPIO의 주요 특징

1. **다양한 핀 구성**: GPIO는 수 개에서 수십 개의 핀으로 구성되어 있으며, 라즈베리파이 4 B의 경우 40개의 핀이 있다.
2. **유연한 사용**: 몇몇 필수 핀을 제외하고는 모든 핀의 기능이 미리 정해져 있지 않아, 사용자가 필요에 따라 입출력 핀의 역할을 설정할 수 있다.
3. **입출력 선택**: 각 핀은 입력 또는 출력 중 하나의 기능을 선택하여 사용할 수 있다.
4. **전원 공급 가능**: GPIO 핀을 통해 외부 하드웨어에 전원을 공급할 수 있다.
5. **소프트웨어 제어**: GPIO 핀으로 송수신되는 신호는 소프트웨어를 통해 제어될 수 있다. 이를 통해 하드웨어와의 상호작용이 가능하다.

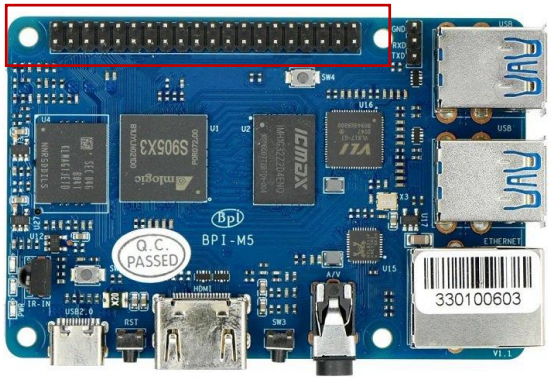
GPIO

아두이노의 GPIO



아두이노의 GPIO

바나나파이의 GPIO



라즈베리파이의 GPIO

GPIO

핀번호	이름		이름	핀번호
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1, I²C)		DC Power 5v	04
05	GPIO03 (SCL1, I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI0_MOSI)		Ground	20
21	GPIO09 (SPI0_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI0_CLK)		(SPI0_CE0_N) GPIO08	24
25	Ground		(SPI0_CE1_N) GPIO07	26
27	ID_SD (I²C ID_EEPROM)		(I²C ID_EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

GPIO

- 소프트웨어를 통해 GPIO에 연결된 장치와 데이터를 주고받을 수 있다. GPIO는 디지털 신호를 주고받는데, 이 신호는 전압 수준에 따라 결정된다.

- **디지털 값 1(이진수 1)**: 3.3V 또는 5V로 표현되며, 이 상태를 HIGH라고 부른다.
- **디지털 값 0(이진수 0)**: 0V 또는 GND로 표현되며, 이 상태를 LOW라고 부른다.

이러한 전압 차이를 이용해 외부 장치와의 통신 및 제어가 가능하다.

• GPIO 라이브러리

- GPIO와의 상호작용을 간편하게 할 수 있는 다양한 소프트웨어 라이브러리가 존재한다. 이를 통해 코드에서 간단하게 핀 설정, 신호 송수신 등을 제어할 수 있다.

언어	라이브러리 이름	언어	라이브러리 이름
C/C++	pigpio, BCM2835	Python	RPi.GPIO
C#	RaspberryGPIOManager	Scratch	ScratchGPIO
Java	Pi4J	Shell	sysfs
Perl	BCM2835		

GPIO

RPi.GPIO는 라즈베리파이의 GPIO 핀에 연결된 장치들을 제어하기 위한 파이썬 라이브러리이다. 이 라이브러리를 사용하면, 파이썬을 통해 GPIO 핀의 상태를 쉽게 제어할 수 있다.

RPi.GPIO 라이브러리 설치 방법

1.가상 환경 활성화

- 먼저 가상 환경을 활성화하여 작업을 진행한다.

```
pi@pi:~ $ source myenv/bin/activate
(myenv) pi@pi:~ $
```

2.RPi.GPIO 라이브러리 설치

- 가상 환경이 활성화된 상태에서 RPi.GPIO 라이브러리를 설치한다.

```
(myenv) pi@pi:~ $ pip install RPi.GPIO
```

파이썬 프로그램에 RPi.GPIO 활용

RPi.GPIO 라이브러리를 파이썬 코드에 사용하려면 아래와 같이 라이브러리를 불러온다.

```
import RPi.GPIO as GPIO
```

GPIO

함수	기능
GPIO.setmode(GPIO.BOARD)	BOARD 모드의 핀 번호 사용
GPIO.setmode(GPIO.BCM)	BCM 모드의 핀 번호 사용
GPIO.getmode()	setmode()로 설정된 값 리턴(GPIO.BOARD, GPIO.BCM, None 중 하나)
GPIO.setup(pin, GPIO.IN)	pin을 입력으로 설정. pin은 번호를 나타내는 정수
GPIO.setup(pin, GPIO.OUT)	pin을 출력으로 설정
GPIO.input(pin)	pin으로부터 디지털 값 읽어 리턴(0 또는 1)
GPIO.output(pin, GPIO.HIGH)	pin으로 디지털 1(HIGH) 값 출력. GPIO pin에 5V의 전압 출력
GPIO.output(pin, GPIO.LOW)	pin으로 디지털 0(LOW) 값 출력. GPIO pin에 0V(GND)의 전압 출력
GPIO.cleanup()	사용한 핀들의 번호를 기억에서 지우고, 사용한 핀들을 모두 입력으로 초기화
GPIO.setwarnings(True/False)	매개변수가 False이면 이미 사용 중인 핀에 다음 프로그램에서 setmode()를 호출하여 입출력 용도를 바꿀 때 경고 메시지가 출력되지 않게 하고, 매개변수가 True이면 경고 메시지가 출력되게 함

GPIO

- 응용 프로그램에서 GPIO 핀을 사용할 때, 핀 번호를 설정하는 방식에는 두 가지가 있다. 이 두 가지 방법을 통해 개발자는 GPIO 핀을 제어할 수 있다.

1. BOARD 모드

1. **물리적인 핀 번호**로 매겨진 번호를 사용한다.
2. 라즈베리파이 보드에서 핀의 실제 위치를 기준으로 번호가 매겨져 있다.
3. 핀 배치 그대로 번호를 사용할 때 유용하다.

2. BCM 모드

1. 라즈베리파이의 **BCM(Broadcom)** 칩에 의해 매겨진 **용도별 핀 번호**를 사용한다.
2. 라즈베리파이의 내부 칩 설계에 따라 핀에 할당된 고유 번호를 기준으로 사용된다.
3. 특정 기능(예: I2C, SPI 등)을 담당하는 핀을 지정할 때 BCM 모드를 많이 사용한다.

- 모드 선택

```
GPIO.setmode(GPIO.BOARD) # BOARD 모드 선택
GPIO.setmode(GPIO.BCM) # BCM 모드 선택
```

- 현재 모드 알아내기

```
mode = GPIO.getmode() # GPIO.BOARD, GPIO.BCM, None 중 하나 리턴
```

GPIO

- GPIO 핀의 입출력 용도 선택

```
GPIO.setup(6, GPIO.IN)          # GPIO6 핀을 입력으로 설정
GPIO.setup(21, GPIO.OUT)        # GPIO21 핀을 출력으로 설정
```

- GPIO 핀으로부터 입출력
  - 핀으로부터 디지털 값 읽기

```
value = GPIO.input(6)           # GPIO6 핀으로부터 디지털 신호 읽기
```

  - 핀에 디지털 값 출력
    - GPIO21 핀에 디지털 값 1 출력. 실제 5V 전압 출력

```
GPIO.output(21, 1)
GPIO.output(21, GPIO.HIGH)
GPIO.output(21, True)
```

    - GPIO21 핀에 디지털 값 0 출력. 실제 0V(GND) 전압 출력

```
GPIO.output(21, 0)
GPIO.output(21, GPIO.LOW)
GPIO.output(21, False)
```

G P I O

- RPi.GPIO 라이브러리의 버전 알아내기

```
print(GPIO.VERSION)
```

GPIO

•RPi.GPIO 모듈은 현재 실행 중인 프로그램이 사용 중인 **GPIO 핀 번호를 기억**한다.

•만약 다른 프로그램에서 동일한 핀을 사용하려고 하면, RPi.GPIO 모듈은 **경고 메시지**를 출력하여 개발자에게 핀의 충돌을 알려준다.

•이러한 경고는 두 번째 프로그램이 해당 핀을 사용하지 말도록 경고하는 역할을 한다.

```
RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
```

경고 메시지가 출력되는 상황

• 동일한 핀을 사용하는 경우: 두 프로그램이 동일한 GPIO 핀을 사용하려 할 때 경고가 출력된다.

• 클린업을 하지 않은 경우: 예를 들어, a.py 프로그램이 종료될 때 GPIO.cleanup()을 실행하지 않고 종료된 후, 다시 a.py를 실행하려고 하면 경고 메시지가 출력된다. 이때, RPi.GPIO는 이전 실행에서 사용한 핀을 기억하고 있기 때문이다.

- 방법 1 - GPIO.cleanup()
- 방법 2 - GPIO.setwarnings(False)

GPIO

GPIO cleanup()

GPIO 클린업은 현재 프로그램이 사용한 핀들의 설정을 초기화하는 과정이다. 이를 통해 프로그램이 종료되었을 때에도 핀들이 안전한 상태로 유지될 수 있다.
클린업은 프로그램이 종료되면 GPIO 라이브러리가 **현재 사용 중인 모든 핀의 번호 기억을 삭제**하고, 사용했던 모든 핀을 입력 모드(GPIO.IN)로 초기화한다.
클린업을 제대로 수행하지 않으면 예상치 못한 오류나 하드웨어 손상이 발생할 수 있다.

클린업의 중요성

- **핀 상태 유지**: 프로그램이 종료되더라도, 사용 중이던 핀의 설정(예: 5V 출력)이 유지될 수 있다.
- **안전 문제 발생 가능**: 만약 출력 핀이 여전히 5V 전압을 유지한 채로 응용 프로그램이 종료되고, 사용자가 실수로 그 핀에 Ground(GND)를 연결하면 **합선**이 발생할 수 있다.
- **하드웨어 손상 위험**: 이런 상황이 지속되면 회로가 고장나거나 라즈베리파이 또는 외부 장치에 손상을 줄 수 있다.

클린업 코드 예시

RPi.GPIO 라이브러리에서 클린업을 수행하는 방법은 매우 간단하다.
프로그램의 마지막 부분에 GPIO.cleanup() 함수를 호출하여 사용한 핀을 정리할 수 있다.

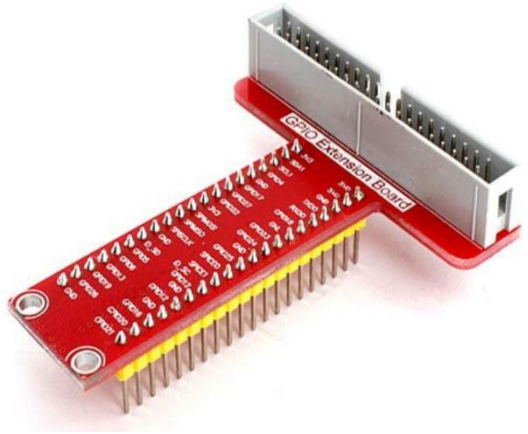
GPIO

- 프로그램이 어떤 상황에서 종료하더라도 클린업하도록 작성
  - 정상적인 종료나,
  - 사용자의 [Ctrl+C] 키로 종료할 때나,
  - 오류로 종료할 때 모두 클린업하도록 프로그램 작성 필요

```
import RPi.GPIO  as GPIO

try:        #
            # 개발자가 작성하고자 하는 목적 코드들
            #
except KeyboardInterrupt: # [Ctrl+C] 키가 입력되어 종료하는 경우
            # 종료 전 처리할 코드

except : # 오류로 인해 종료하는 경우
            # 오류 처리 코드

finally: # 종료하는 모든 경우(정상적이든 오류로 인한 종료 등)에 실행되는 코드
            GPIO.cleanup()
```

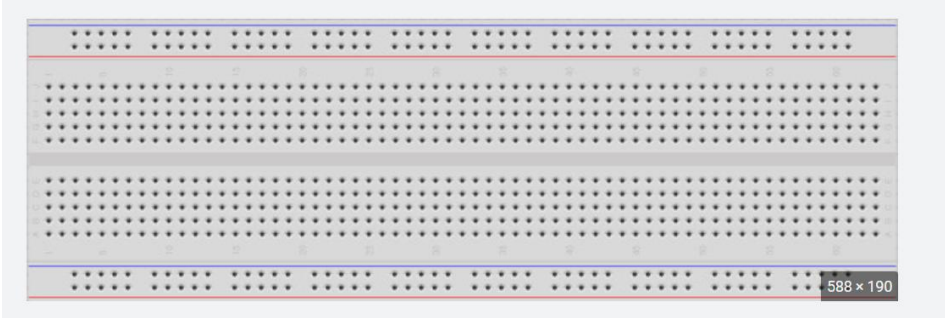
GPIO

- 라즈베리파이의 GPIO 핀에 외부 장치 연결
  - 직접 연결 - 매우 불편
  - 40핀의 플랫 케이블, T자형 GPIO 확장 보드, 브레드보드 활용



T자형 GPIO 확장 보드

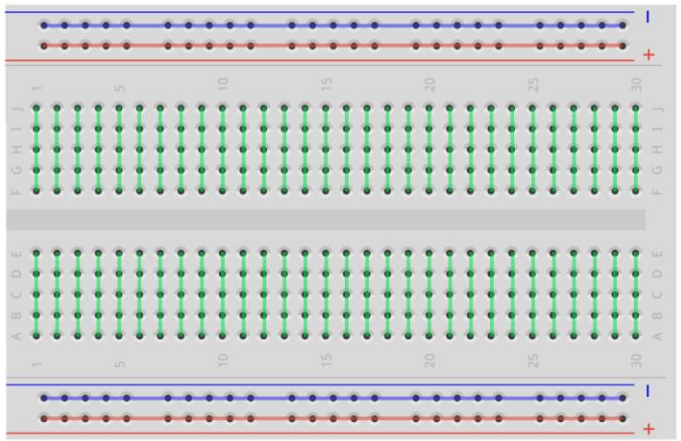
40핀 플랫 케이블



브레드보드

GPIO

- 브레드보드(Bread Board, 빵판)
  - 사용자가 납땜하지 않고 회로를 쉽게 구성할 수 있도록 설계된 장치
- 브레드보드의 구조
  - 수평으로 연결된 라인과 수직으로 연결된 라인들
    - 라인(파란색, 붉은색, 초록색)은 내부적으로 연결되어 있음

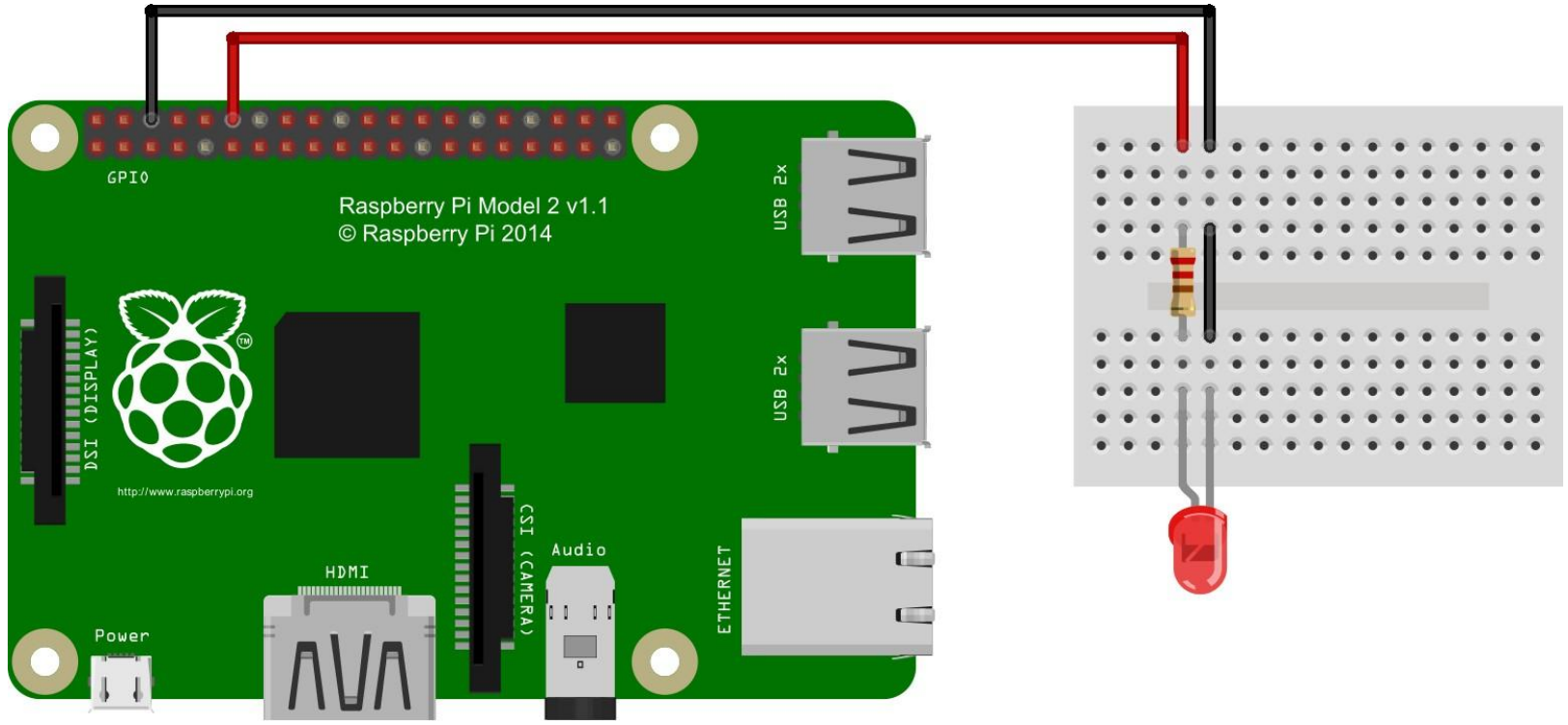


  - 5V와 GND 선
    - 브레드보드에 한 번 연결해두고 계속 사용

LED 점멸 제어

회로구성

- GPIO 18(+), GND(-), 220Ω(적적갈)



LED 점멸 제어

프로그램

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)              # 핀번호방식: BCM -> 채널번호
                                    #             BOARD -> 보드핀번호
LED = 18                            # GPIO 18번으로 설정

GPIO.setup(LED, GPIO.OUT)           # LED(18)번 채널을 출력용으로 설정

try:                                # 예외처리 구문
      while (True):                             # 무한 반복
            GPIO.output(LED, GPIO.HIGH)         # LED(18)에 HIGH 상태 인가
            time.sleep(1)                       # 1초 지연
            GPIO.output(LED, GPIO.LOW)          # LED(18)에 LOW 상태 인가
            time.sleep(1)                       # 1초 지연

except KeyboardInterrupt:             # 예외: 키보드 중단(Ctrl+C 등)
      GPIO.output(LED, GPIO.LOW) # LED(18)에 LOW 상태 인가
      GPIO.cleanup()                # 프로그램 종료
```