[25.01] Pinky Blue 장치 사용해보기

Pinky Contents

Exported on 01/13/2025

Table of Contents

1 Pinky Blue의 장치들

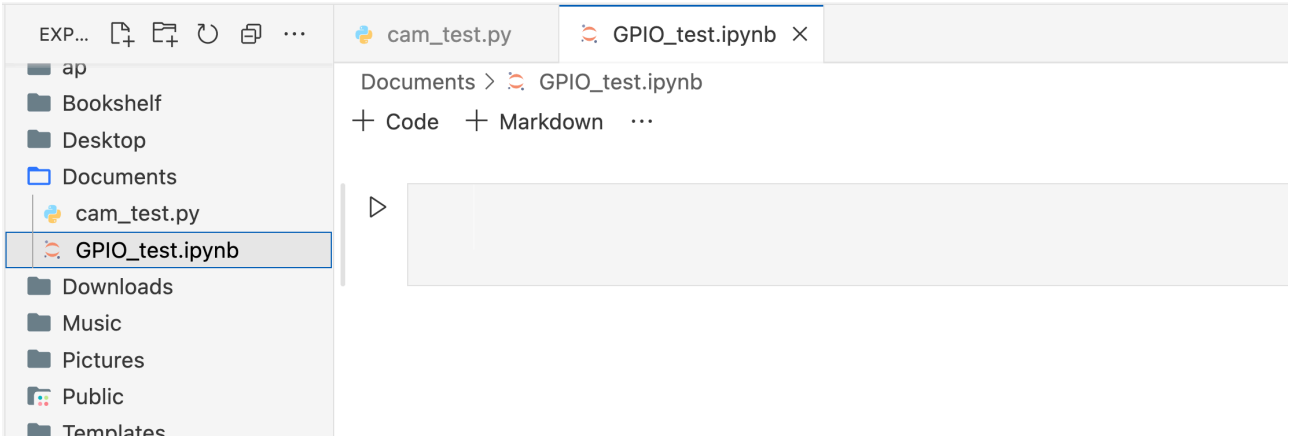
1.1 먼저 Raspberry pi 5 GPIO 모듈 설치

```
pinky@raspberrypi: ~
pinky@raspberrypi:~ $
pinky@raspberrypi:~ $ source ~/venv/my_venv/bin/activate
(my_venv) pinky@raspberrypi:~ $
(my_venv) pinky@raspberrypi:~ $ pip install rpi-lgpio
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting rpi-lgpio
  Downloading https://www.piwheels.org/simple/rpi-lgpio/rpi_lgpio-0.6-py3-none-any.whl (11 kB)
Requirement already satisfied: lgpio>=0.1.0.1 in /usr/lib/python3/dist-packages (from rpi-lgpio) (0.2.2.0)
Installing collected packages: rpi-lgpio
Successfully installed rpi-lgpio-0.6
(my_venv) pinky@raspberrypi:~ $
```

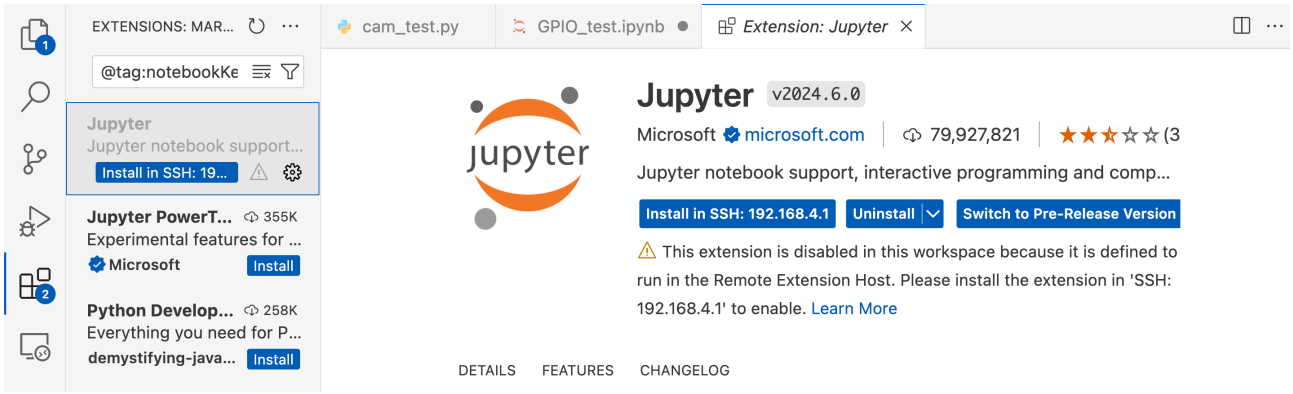
1.2 그리고 Jupyter notebook 실행

```
pinky@raspberrypi: ~
(my_venv) pinky@raspberrypi:~ $ jupyter notebook
```

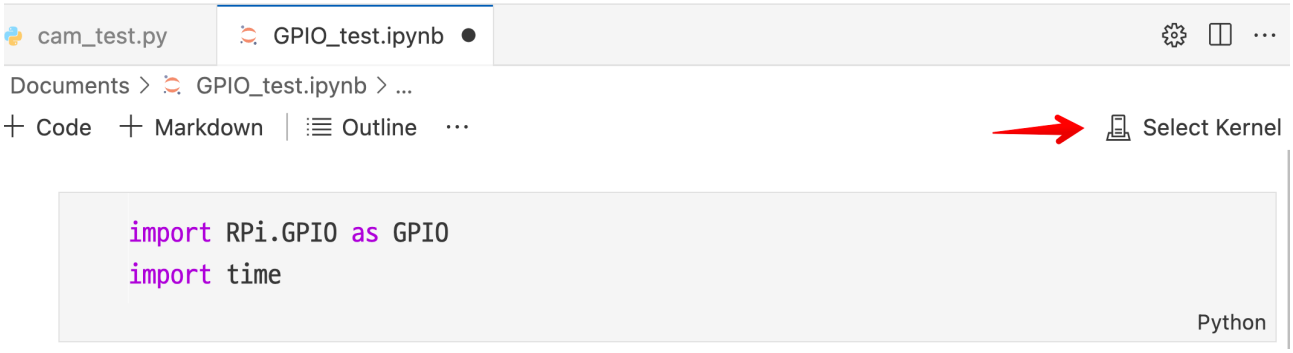
1.3 이번에는 VSCode에서 GPIO_test.ipynb 파일을 생성하자



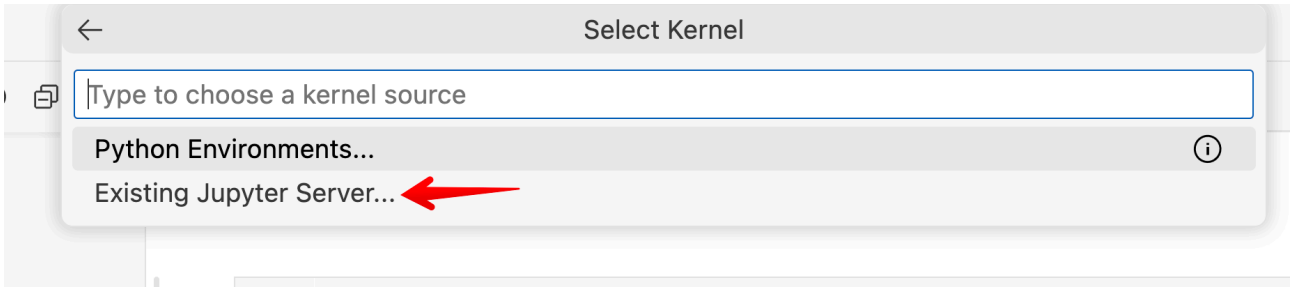
1.4 extensions에서 Jupyter를 꼭 SSH에서 설치하자



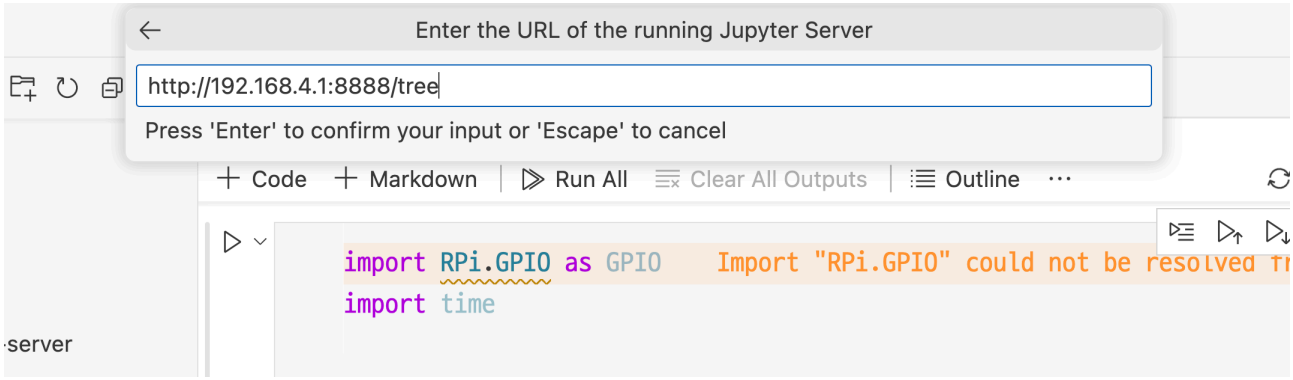
1.5 VSCode에서 Python 환경을 잡아주어야 함



1.6 이걸 누르고



1.7 그리고



1.8 VSCode로 접속이 안되면 웹으로 Jupyter Notebook 접속하자

1.9 GPIO가 import가 된다

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

[1] ✓ 0.0s Python

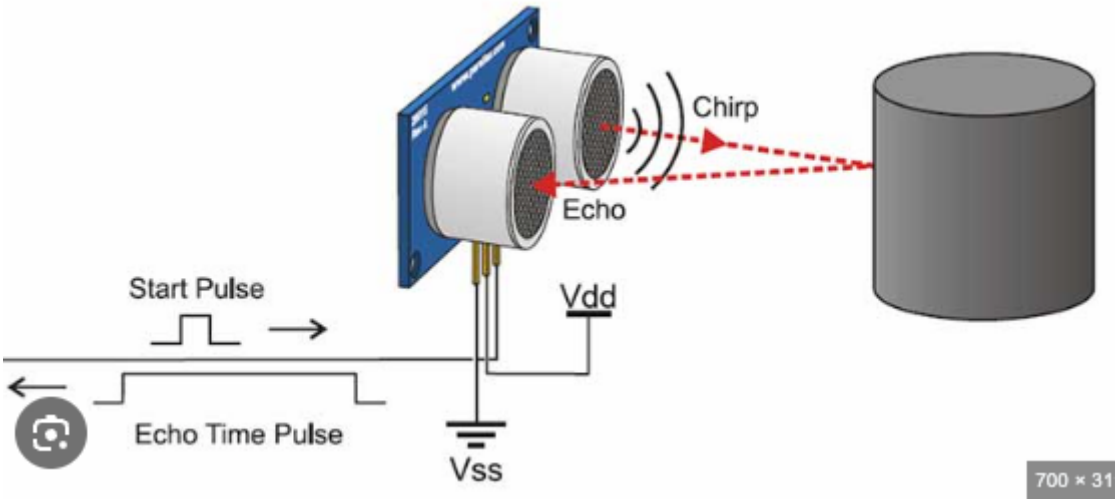
2 초음파 센서

2.1 일단 핀설정 해두고, 이건 약간 아두이노 스러운데~

```
TRIG = 23
ECHO = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

GPIO.output(TRIG, False)
```

[2] ✓ 0.0s Python

2.2 초음파 센서는 음파를 쏘고, 받고, 시간을 측정한다~



2.3 이게 왔다 갔다한 시간

```
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO) == 0:
    pulse_start = time.time()
while GPIO.input(ECHO) == 1:
    pulse_end = time.time()

pulse_duration = pulse_end - pulse_start
pulse_duration
```

✓ 0.0s Python

```
0.002089977264404297
```

2.4 근데 저 시간... 뭐 어쩌라고? 아하~

```
# 거리 계산 (소리의 속도는 34300 cm/s)
distance = pulse_duration * 34300 / 2
distance
```

✓ 0.0s Python

```
35.84311008453369
```

2.5 요걸 함수로 하면

```python
def calc_distance_us():
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start

    return pulse_duration * 34300 / 2
```

✓ 0.0s Python

2.6 요렇게 쓰면 되겠네

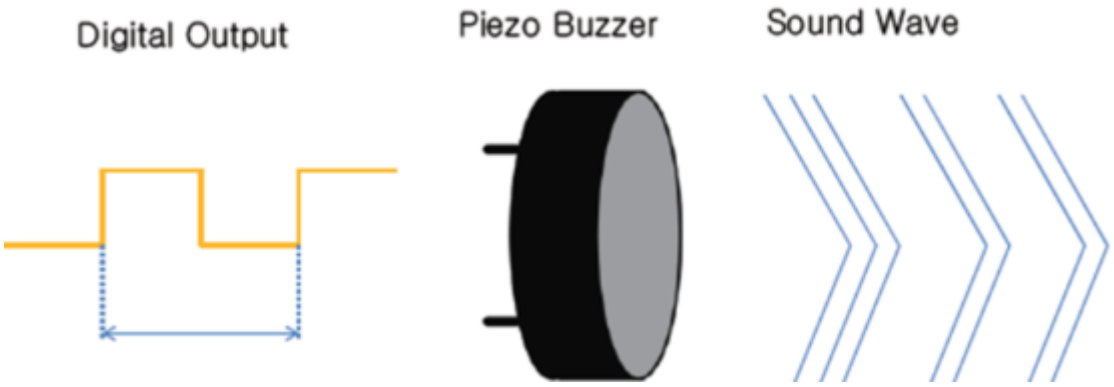
```python
cnt = 0

while cnt < 5:
    dist = calc_distance_us()
    print(str(dist) + ' cm')
    cnt += 1

    time.sleep(1)
```

✓ 5.0s Python

```
43.109047412872314 cm
11.191260814666748 cm
10.234463214874268 cm
7.184159755706787 cm
22.623765468597412 cm
```

3 부저

3.1 부저가 우는 원리?



3.2 부저 핀 세팅

```
BUZZER_PIN = 22

GPIO.setup(BUZZER_PIN, GPIO.OUT)
```

✓ 0.0s Python

3.3 부저 울리기

```
pwm = GPIO.PWM(BUZZER_PIN, 1000)
pwm.start(50)

for i in range(3):
    pwm.ChangeDutyCycle(50)
    time.sleep(0.5)
    pwm.ChangeDutyCycle(0)
    time.sleep(0.5)

pwm.stop()
```

✓ 3.0s Python

3.4 뭐 도미솔 정도는~

```
pwm.start(50)
pwm.ChangeFrequency(262)
time.sleep(0.5)
pwm.ChangeFrequency(330)
time.sleep(0.5)
pwm.ChangeFrequency(392)
time.sleep(0.5)
pwm.ChangeDutyCycle(0)
time.sleep(0.5)
pwm.stop()
```

✓ 2.0s Python

4 모터

4.1 모터 gpio 핀번호 정의 하기

[1]:
```
import RPi.GPIO as GPIO
from time import sleep
```

[2]:
```
# GPIO 핀 번호 정의
AIN1 = 17  # 오른쪽 모터의 AIN1 핀
AIN2 = 27  # 오른쪽 모터의 AIN2 핀
PWMA = 18  # 오른쪽 모터의 PWM 제어 핀
BIN1 = 5   # 왼쪽 모터의 BIN1 핀
BIN2 = 6   # 왼쪽 모터의 BIN2 핀
PWMB = 13  # 왼쪽 모터의 PWM 제어 핀
STBY = 25  # 모터 드라이버의 STBY 핀
```

4.2 모터 gpio설정하기

[3]:
```
# GPIO 설정
GPIO.setmode(GPIO.BCM)

# 모터 제어 핀 설정
GPIO.setup(AIN1, GPIO.OUT)
GPIO.setup(AIN2, GPIO.OUT)
GPIO.setup(PWMA, GPIO.OUT)
GPIO.setup(BIN1, GPIO.OUT)
GPIO.setup(BIN2, GPIO.OUT)
GPIO.setup(PWMB, GPIO.OUT)
GPIO.setup(STBY, GPIO.OUT)
```

4.3 pwm 시작

[4]:
```
# PWM 객체 생성 및 시작
pwm_a = GPIO.PWM(PWMA, 1000)  # 오른쪽 모터 PWM 객체 생성 (주파수: 1000Hz)
pwm_b = GPIO.PWM(PWMB, 1000)  # 왼쪽 모터 PWM 객체 생성 (주파수: 1000Hz)
pwm_a.start(0)
pwm_b.start(0)
```

4.4 모터 활성화 하고

[5]:
```
# 모터 활성화
GPIO.output(STBY, GPIO.HIGH)
```

4.5 로봇 전진

- 코드 실행 전 로봇이 떨어지거나 부딪히지 않게 조심
- 모터 방향이 반대인 로봇은 HIGH->LOW, LOW->HIGH 변경 필요

```
# 직진
GPIO.output(AIN1, GPIO.HIGH)
GPIO.output(AIN2, GPIO.LOW)
pwm_a.ChangeDutyCycle(30) # 속도 설정

GPIO.output(BIN1, GPIO.HIGH)
GPIO.output(BIN2, GPIO.LOW)
pwm_b.ChangeDutyCycle(30)
sleep(1)

pwm_a.ChangeDutyCycle(0)
pwm_b.ChangeDutyCycle(0)
```

4.6 후진

```
# 후진
GPIO.output(AIN1, GPIO.LOW)
GPIO.output(AIN2, GPIO.HIGH)
pwm_a.ChangeDutyCycle(30) # 속도 설정

GPIO.output(BIN1, GPIO.LOW)
GPIO.output(BIN2, GPIO.HIGH)
pwm_b.ChangeDutyCycle(30)
sleep(1)

pwm_a.ChangeDutyCycle(0)
pwm_b.ChangeDutyCycle(0)
```

4.7 좌회전

[8]:
```
# 좌회전
GPIO.output(AIN1, GPIO.LOW)
GPIO.output(AIN2, GPIO.HIGH)
pwm_a.ChangeDutyCycle(30)

GPIO.output(BIN1, GPIO.HIGH)
GPIO.output(BIN2, GPIO.LOW)
pwm_b.ChangeDutyCycle(30)
sleep(1)

pwm_a.ChangeDutyCycle(0)
pwm_b.ChangeDutyCycle(0)
```

4.8 우회전

```
# 좌회전
GPIO.output(AIN1, GPIO.LOW)
GPIO.output(AIN2, GPIO.HIGH)
pwm_a.ChangeDutyCycle(40) # 속도 설정

GPIO.output(BIN1, GPIO.HIGH)
GPIO.output(BIN2, GPIO.LOW)
pwm_b.ChangeDutyCycle(40)
sleep(1)

pwm_a.ChangeDutyCycle(0)
pwm_b.ChangeDutyCycle(0)
```

4.9 GPIO 종료하기

- 위 코드들을 함수화 하면 나중에 쓰기 편할지도?

[12]:
```
# 모터 비활성화
GPIO.output(STBY, GPIO.LOW)

# PWM 종료
pwm_a.stop()
pwm_b.stop()

# GPIO 종료
GPIO.cleanup()
```

5 IR 센서

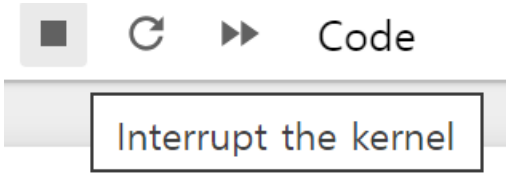
5.1 ir센서 GPIO 설정하기

[1]:
```
import RPi.GPIO as GPIO
import time
```

[2]:
```
# GPIO 핀 번호 설정
IR_PIN_1 = 16
IR_PIN_2 = 20
IR_PIN_3 = 21

# GPIO 설정
GPIO.setmode(GPIO.BCM)
GPIO.setup(IR_PIN_1, GPIO.IN)
GPIO.setup(IR_PIN_2, GPIO.IN)
GPIO.setup(IR_PIN_3, GPIO.IN)
```

5.2 ir센서 읽기

- ir센서에 손을 가져다 대거나 검은색과 흰색이 있는 종이로 확인하기
- ir센서는 검은색이 적외선을 흡수에서 0으로 출력 흰색은 적외선을 반사해서 1로 출력되는 성질이 있다 (이것을 모터와 잘 결합하면 라인트레이싱 코드가 된다)
- 인터럽트로 종료하기



[3]:
```
while True:
    # 각 센서의 상태 읽기
    sensor_1 = GPIO.input(IR_PIN_1)
    sensor_2 = GPIO.input(IR_PIN_2)
    sensor_3 = GPIO.input(IR_PIN_3)

    # 센서 상태 출력
    print(f"Sensor 1: {sensor_1}, Sensor 2: {sensor_2}, Sensor 3: {sensor_3}")

    # 0.1초 대기
    time.sleep(0.1)
```

```
Sensor 1: 0, Sensor 2: 1, Sensor 3: 1
Sensor 1: 1, Sensor 2: 1, Sensor 3: 1
Sensor 1: 1, Sensor 2: 1, Sensor 3: 1
Sensor 1: 1, Sensor 2: 1, Sensor 3: 1
Sensor 1: 1, Sensor 2: 1, Sensor 3: 1
Sensor 1: 1, Sensor 2: 1, Sensor 3: 1
Sensor 1: 1, Sensor 2: 1, Sensor 3: 1
```

5.3 GPIO 종료

[4]:
```
GPIO.cleanup()
```

[ ]: