파이썬 - 라즈베리파이

□ 코드

○ GPIO

* GPIO : 범용 입/출력 포트
* set 하였다 : 그 핀의 전압을 high로 설정
* clear 하였다 : 그 핀의 전압을 low로 설정
* 에노드(+)를 외부에서 높은 전압(high)에 연결하고 케소드(-)를 GPIO에 연결하였다면,
  => 불을 켜기 위해 GPIO 핀을 low로 설정
* 에노드(+)를 GPIO에 연결하고, 케소드(-)를 GND(low)에 연결하였다면,
  => 불을 켜기 위해 GPIO 핀을 high로 설정

GPIO 함수	설명
GPIO.setmode()	GPIO 입력 및 출력을 설정하는 함수 ex) GPIO.setmode(GPIO.BCM) :
GPIO.setup(해당 핀, GPIO값)	해당 핀에 대해 GPIO값으로 작업한다. // 핀 모드는 한 번만 하면 되기 때문에 setup 함수에서만 사용 ex) GPIO.setup(13, GPIO.OUT) : 13번째 핀을 출력 ex) GPIO.setup(13, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) : 13번째 핀을 입력 설정, 내부 풀업 저항을 사용
GPIO.output(해당 핀, GPIO값)	해당 핀을 출력하는데, 몇 볼트로 출력할 것인지 설정 ex) GPIO.output(13, GPIO.LOW) : 13번째 핀을 0(0V)으로 출력 ex) GPIO.output(13, GPIO.HIGH) : 13번째 핀을 1(5V)으로 출력
GPIO.input(해당 핀)	해당 핀을 입력
GPIO.cleanup()	GPIO를 클린업
GPIO 값	**설명**
GPIO.BCM	BCM 모드로 GPIO를 설정
GPIO.OUT	출력 설정
GPIO.IN	입력 설정
GPIO.PUD_DOWN	입력으로 설정하는데, 내부 풀업 저항을 사용
GPIO.HIGH	1 (5V)를 출력
GPIO.LOW	0 (0V)를 출력

Door Lock 코드

```
<main 함수>
if __name__ == '__main__':
    digit=[None, None, None, None]

    for i in range (0, 4):
        while digit[i] == None:
            #print("Press Key")
            kp = keypad()
            digit.insert(i, kp.getKey())

        print(i, "Complate")
        print(digit[i])
```

Door Lock 코드

```
import RPi.GPIO as GPIO
<키패드 클래스>
class keypad():    #20
    # CONSTANTS
    KEYPAD =
    [               // 키패드 번호
        [1,2,3],
        [4,5,6],
        [7,8,9],
        ["*",0,"#"]
    ]

    ROW          = [18,23,24,25]    // 행
    COLUMN        = [4,17,22]       // 열

    def __init__(self):
        GPIO.setmode(GPIO.BCM)
 <키를 받아오는 함수>
    def getKey(self):
     <행, 열 입출력 준비>
        # Set all columns as output low (출력할 행의 모든 열을 설정)
        for j in range(len(self.COLUMN)):
            GPIO.setup(self.COLUMN[j], GPIO.OUT)    // GPIO.OUT : 출력
            GPIO.output(self.COLUMN[j], GPIO.LOW)  // GPIO.LOW : 0(0V)를 출력

        # Set all rows as input (행 입력을 설정)
        for i in range(len(self.ROW)):
            GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)
            // GPIO.IN : 입력
     <행 설정>
        # Scan rows for pushed key/button (눌려진 키의 행을 확인)
        # A valid key press should set "rowVal"  between 0 and 3.
        rowVal = -1  // rowVal 변수로 눌려진 키의 행 확인
        for i in range(len(self.ROW)):
            tmpRead = GPIO.input(self.ROW[i])
            if tmpRead == 0: // input이 0인 것은 눌려진 상태를 뜻함
                rowVal = i     // 0이면 해당 rowVal에 해당 행을 저장

        # if rowVal is not 0 thru 3 then no button was pressed and we can exit
        // rowVal의 값이 지정된 행 범위를 벗어나면 종료
        if rowVal <0 or rowVal >3:
            self.exit()
            return

        # Convert columns to input
        // 열 입력에 대한 변환
        for j in range(len(self.COLUMN)):
                GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
                // GPIO.PUD_DOWN : 입력으로 설정하는데, 내부 풀업 저항을 사용

        # Switch the i-th row found from scan to output (확인한 출력에 대한 행을 찾음)
        GPIO.setup(self.ROW[rowVal], GPIO.OUT)     // GPIO.OUT : 출력
        GPIO.output(self.ROW[rowVal], GPIO.HIGH)  // GPIO.HIGH : 1(5V)를 출력
     <열 설정>
        # Scan columns for still-pushed key/button
        # A valid key press should set "colVal"  between 0 and 2.
        colVal = -1  // colVal 변수로 눌려진 키의 열 확인
        for j in range(len(self.COLUMN)):
            tmpRead = GPIO.input(self.COLUMN[j]) // j번째 열을 입력
            if tmpRead == 1:  // input이 1인 것은
                colVal=j        // 1이면 해당 colVal에 해당 열을 저장

        # if colVal is not 0 thru 2 then no button was pressed and we can exit
        // colVal의 값이 지정된 열 범위를 벗어나면 종료
        if colVal <0 or colVal >2:
            self.exit()
            return
     <키를 찾았으면 종료>
        # Return the value of the key pressed
        // 눌려진 키를 return 시킴
        self.exit()
        return self.KEYPAD[rowVal][colVal] // 해당 키패드의 행,열번째의 키를 출력
 <exit 함수, 종료시 사용>
    def exit(self):
        # Reinitialize all rows and columns as input at exit
        for i in range(len(self.ROW)): // i번째 행을 입력, 풀업 저항 사용
                GPIO.setup(self.ROW[i], GPIO.IN, pull_up_down=GPIO.PUD_UP)

        for j in range(len(self.COLUMN)):  // j번째 열을 입력, 풀업 저항 사용
                GPIO.setup(self.COLUMN[j], GPIO.IN, pull_up_down=GPIO.PUD_UP)
```