

Chap03. 입력장치 제어

Contents

3.1 가변저항(VR) 제어하기

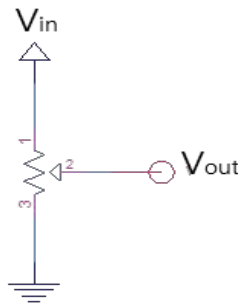
3.2 조도센서(Cds) 제어하기

3.3 버튼(Button) 제어하기

3.1 가변저항 제어하기

◆ 가변저항(Variable Resistor) – 포텐쇼미터(Potentiometer)

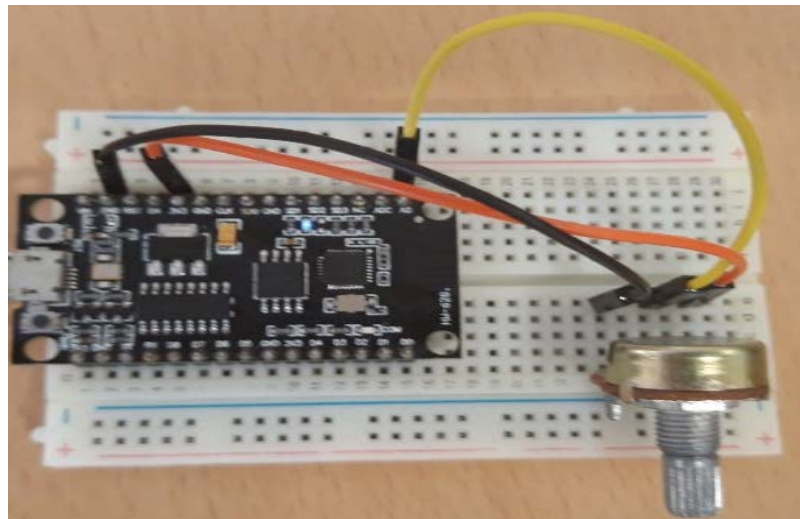
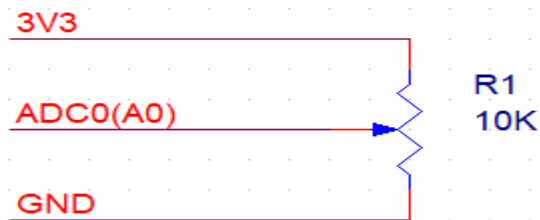
- 회전 및 직선 변위를 감지하는 센서
- 위치에 따라 저항값 변화
- 변화된 저항에 전압을 인가하여 전압 변화 감지(ADC 이용)



3.1 가변저항 제어하기

◆ 3.1.1 가변저항 값 시리얼 프린터로 출력

- 회로도 및 연결도

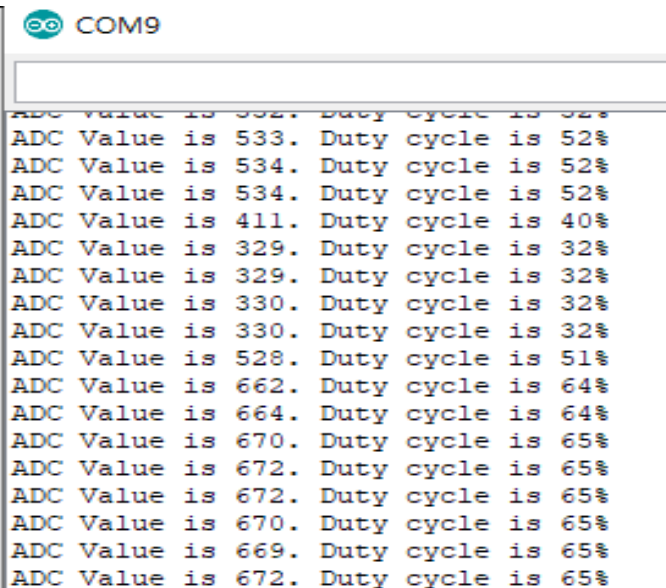


3.1 가변저항 제어하기

◆ 3.1.1 가변저항 값 시리얼 프린터로 출력

● 소스 코드

```
1
2 // 가변저항 Control_Serial -> vr_control
3
4 void setup()
5 {
6     // 시리얼 통신 설정
7     Serial.begin(9600);
8 }
9
10 void loop()
11 {
12     int adcValue; // 실제 센서로부터 읽은 값 (0~1023)
13     int duty;      // LED 점멸 주기 (0~100%)
14
15     // 가변저항 값 읽어들임
16     adcValue = analogRead(A0);
17
18     // 가변저항 값 0~100의 범위로 변경
19     duty = map(adcValue, 0, 1023, 0, 100);
20
21     // 시리얼 통신으로 출력
22     Serial.print("ADC Value is ");
23     Serial.print(adcValue);
24     Serial.print(". Duty cycle is ");
25     Serial.print(duty);
26     Serial.println("");
27     delay(500);
28 }
```



COM9

ADC Value is 532. Duty cycle is 52%

ADC Value is 533. Duty cycle is 52%

ADC Value is 534. Duty cycle is 52%

ADC Value is 411. Duty cycle is 40%

ADC Value is 329. Duty cycle is 32%

ADC Value is 329. Duty cycle is 32%

ADC Value is 330. Duty cycle is 32%

ADC Value is 330. Duty cycle is 32%

ADC Value is 528. Duty cycle is 51%

ADC Value is 662. Duty cycle is 64%

ADC Value is 664. Duty cycle is 64%

ADC Value is 670. Duty cycle is 65%

ADC Value is 672. Duty cycle is 65%

ADC Value is 672. Duty cycle is 65%

ADC Value is 670. Duty cycle is 65%

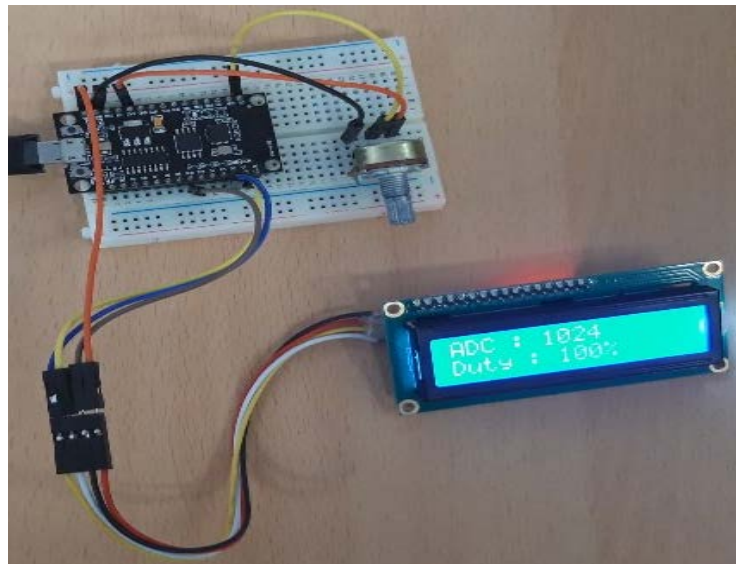
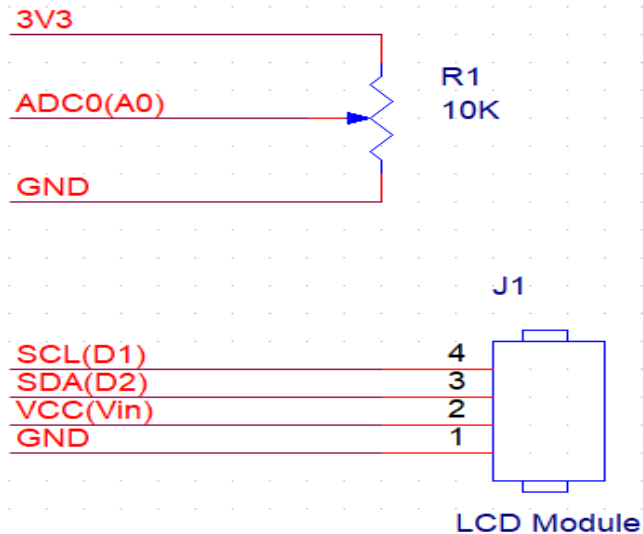
ADC Value is 669. Duty cycle is 65%

ADC Value is 672. Duty cycle is 65%

3.1 가변저항 제어하기

◆ 3.1.2 가변저항 값 LCD에 출력

● 회로도 및 연결도



3.1 가변저항 제어하기

◆ 3.1.2 가변저항 값 LCD에 출력

● 소스 코드

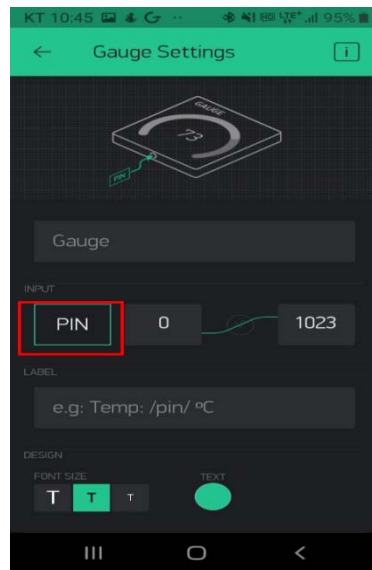
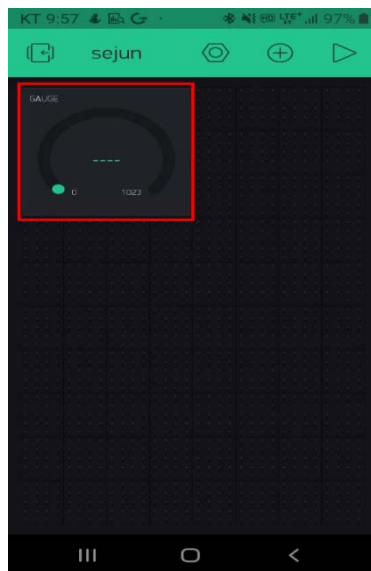
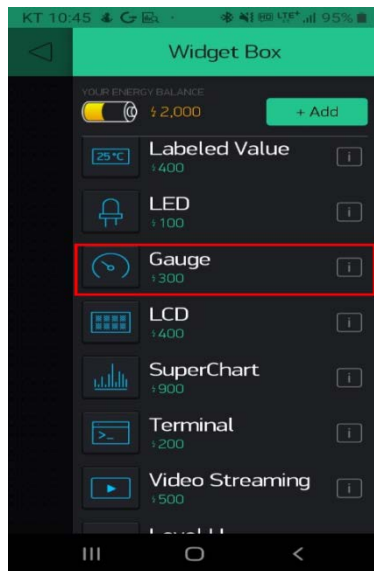
```
2 // 가변저항 Control_lcd -> vr_control)lcd
3
4 // I2C 통신 라이브러리 설정
5 #include <Wire.h>
6 // I2C LCD 라이브러리 설정
7 #include <LiquidCrystal_I2C.h>
8
9 // LCD I2C address 설정 PCF8574:0x27, PCF8574A:
10 LiquidCrystal_I2C lcd(0x20, 16, 2);
11
12 void setup()
13 {
14     // 시리얼 통신 설정
15     Serial.begin(9600);
16     lcd.init();
17     lcd.backlight();
18 }
```

```
20 void loop()
21 {
22     int adcValue; // 실제 센서로부터 읽은 값 (0~1023)
23     int duty;      // LED 점멸 주기 (0~100%)
24
25     // 가변저항 값 읽어들임
26     adcValue = analogRead(A0);
27
28     // 가변저항 값 0~100의 범위로 변경
29     duty = map(adcValue, 0, 1023, 0, 100);
30
31     // 시리얼 통신으로 출력
32     lcd.clear();
33     lcd.setCursor(0,0);
34     lcd.print("ADC : ");
35     lcd.print(adcValue);
36     lcd.setCursor(0,1);
37     lcd.print("Duty : ");
38     lcd.print(duty);
39     lcd.print("%");
40     delay(100);
41 }
```

3.1 가변저항 제어하기

◆ 3.1.3 Blynk 가변저항 값 읽어오기 - 가상핀

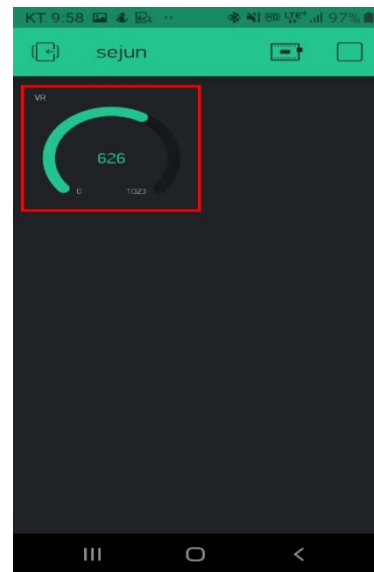
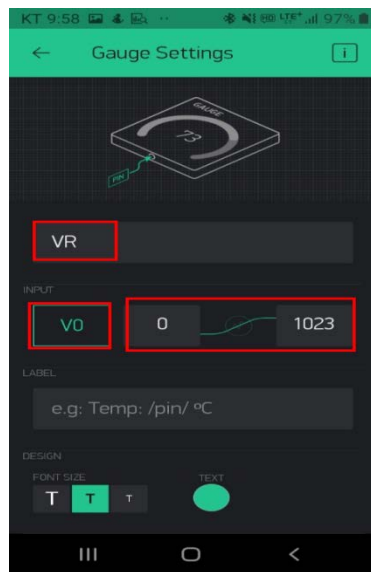
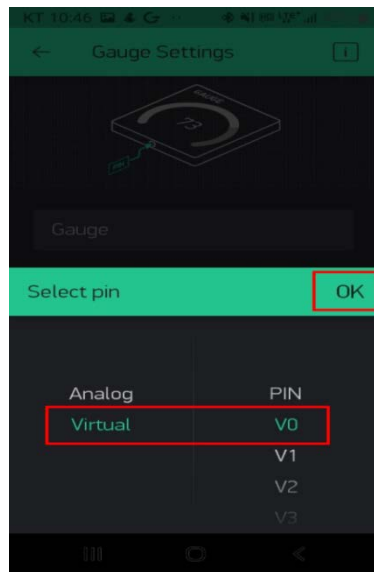
- Widget 설정



3.1 가변저항 제어하기

◆ 3.1.3 Blynk 가변저항 값 읽어오기 - 가상핀

- Widget 설정



3.1 가변저항 제어하기

◆ 3.1.3 Blynk 가변저항 값 읽어오기 - 가상핀

● 소스 코드

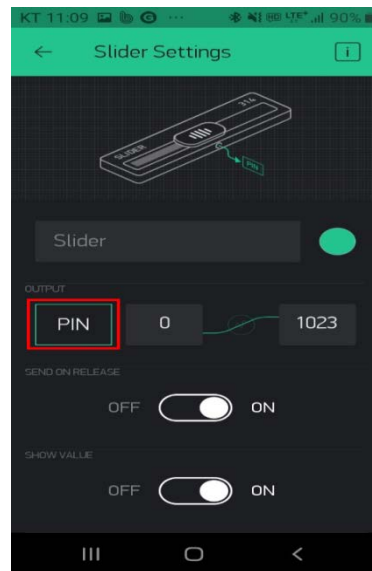
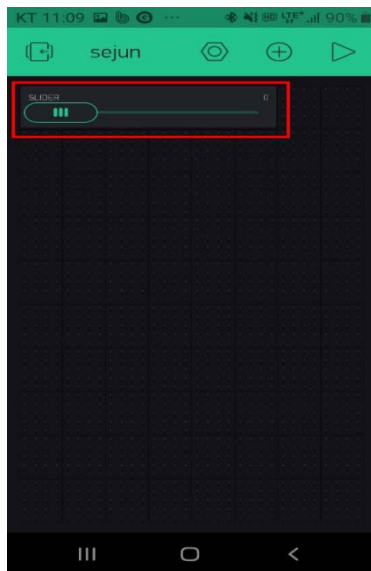
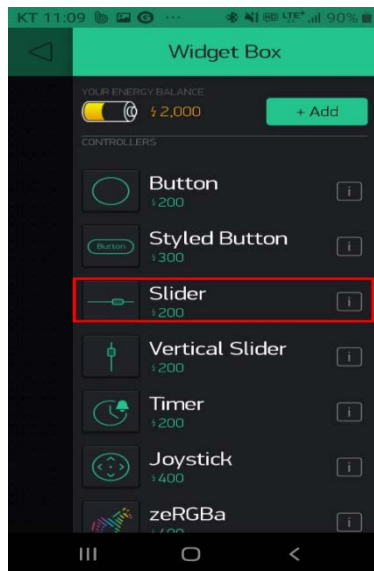
```
2 // 가변저항 Control_vpin -> vr_control_vpin(NodeMCU ->스마트폰)
3
4 // I2C 통신 라이브러리 설정
5 #include <Wire.h>
6 // I2C LCD 라이브러리 설정
7 #include <LiquidCrystal_I2C.h>
8 #include <ESP8266WiFi.h>
9 #include <BlynkSimpleEsp8266.h>
10 #define BLYNK_PRINT Serial
11 // LCD I2C address 설정 PCF8574:0x27, PCF8574A:0x3F
12 LiquidCrystal_I2C lcd(0x20, 16, 2);
13 // You should get Auth Token in the Blynk App.
14 // Go to the Project Settings (nut icon).
15 char auth[] = "KWiEYYjACwL_4t3z-5wNjHSxw3Uftqy4"; /
16
17 // Your WiFi credentials.
18 // Set password to "" for open networks.
19 char ssid[] = "sjpark";
20 char pass[] = "12345678";

22 void setup()
23 {
24     Blynk.begin(auth, ssid, pass);
25 }
26
27 void loop()
28 {
29     Blynk.run();
30
31     int adcValue = analogRead(A0); // 가변저항 값 읽어옴
32
33     Blynk.virtualWrite(V0, adcValue);
34     delay(100);
35 }
```

3.1 가변저항 제어하기

◆ 3.1.4 Blynk 가상(가변저항) LCD에 출력하기 - 가상핀

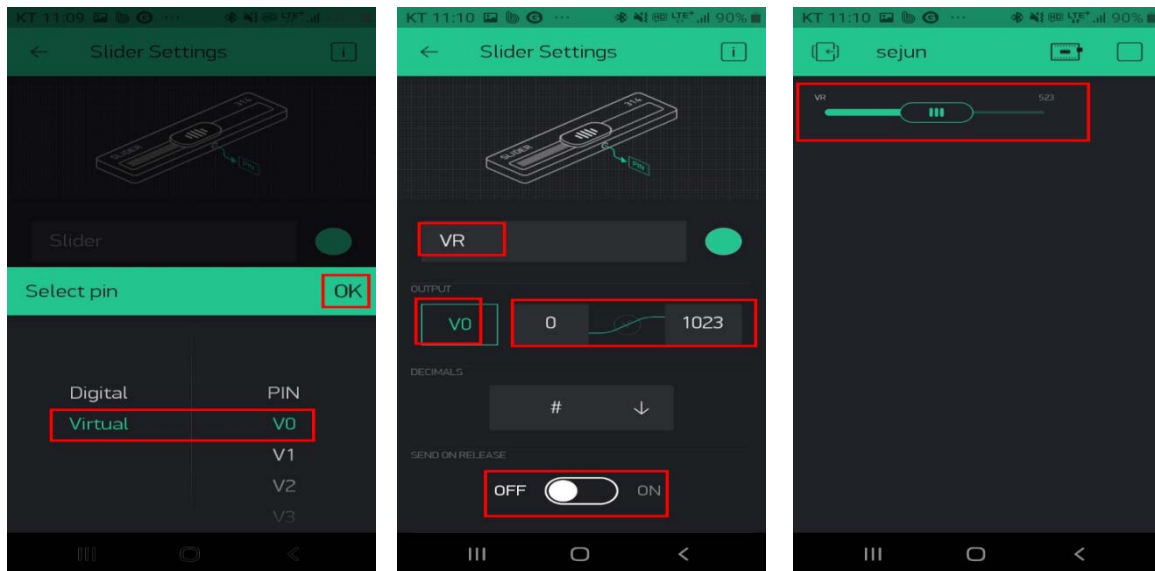
- Widget 설정



3.1 가변저항 제어하기

◆ 3.1.4 Blynk 가상(가변저항) LCD에 출력하기 - 가상핀

- Widget 설정



3.1 가변저항 제어하기

◆ 3.1.4 Blynk 가상(가변저항) LCD에 출력하기 - 가상핀

● 소스 코드

```
2 // 가변저항 Control_lcd_vpin -> vr_control_lcd_vpin(스마트폰 -> NodeMCU)
3
4 // I2C 통신 라이브러리 설정
5 #include <Wire.h>
6 // I2C LCD 라이브러리 설정
7 #include <LiquidCrystal_I2C.h>
8 #include <ESP8266WiFi.h>
9 #include <BlynkSimpleEsp8266.h>
10 #define BLYNK_PRINT Serial
11 // LCD I2C address 설정 PCF8574:0x27, PCF8574A:0x3F
12 LiquidCrystal_I2C lcd(0x20, 16, 2);
13 // You should get Auth Token in the Blynk App.
14 // Go to the Project Settings (nut icon).
15 char auth[] = "KWiEYYjACwL_4t3z-5wNjHSxw3Uftqy4"; // 이메일 토큰으로 변경
16
17 // Your WiFi credentials.
18 // Set password to "" for open networks.
19 char ssid[] = "sjpark";
20 char pass[] = "12345678";
21
22 int adcValue=0;
23 int duty;
```

3.1 가변저항 제어하기

◆ 3.1.4 Blynk 가상(가변저항) LCD에 출력하기 - 가상핀

● 소스 코드

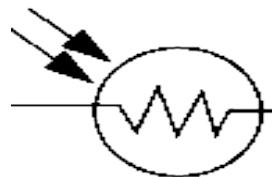
```
25 void setup()
26 {
27   // 시리얼 통신 설정
28   Serial.begin(9600);
29   lcd.init();
30   lcd.backlight();
31   Blynk.begin(auth, ssid, pass);
32 }
33
34 BLYNK_WRITE(V0)
35 {
36   adcValue = param.asInt(); //매개변수로 들어옴
37 }
38 }
```

```
40 void loop()
41 {
42   Blynk.run();
43
44   duty = map(adcValue, 0, 1023, 0, 100);
45
46   lcd.clear();
47   lcd.setCursor(0,0);
48   lcd.print("ADC : ");
49   lcd.print(adcValue);
50   lcd.setCursor(0,1);
51   lcd.print("Duty : ");
52   lcd.print(duty);
53   lcd.print("%");
54   delay(100);
55 }
```

3.2 조도센서 제어하기

◆ 조도센서(Cds)

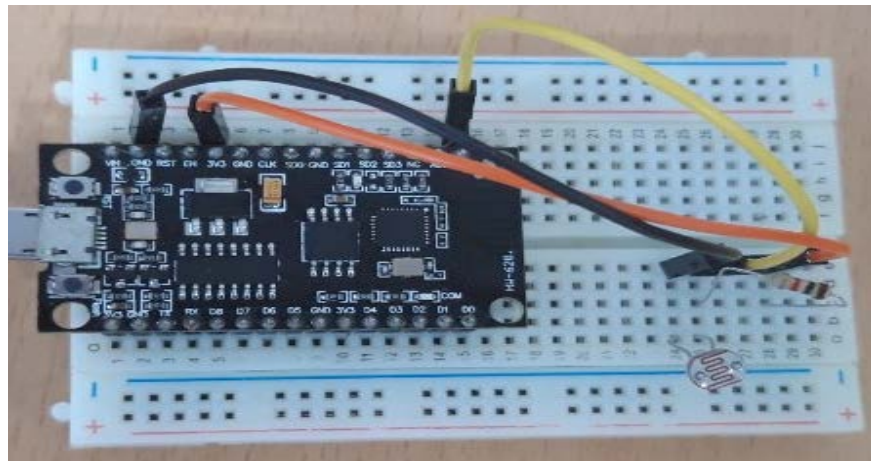
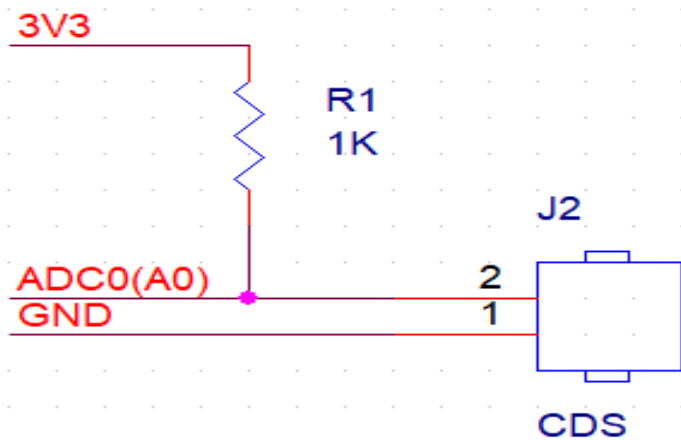
- CdS 분말을 세라믹 기판 위에 압축하여 제작
- 빛이 강할 수록 저항값 감소
- 변화된 저항에 전압을 인가하여 전압의 변화 감지(ADC 이용)
- 자동 조명장치, 조도 측정 등에 사용



3.2 조도센서 제어하기

◆ 3.2.1 조도센서 값 시리얼 프린터로 출력

- 회로도 및 연결도

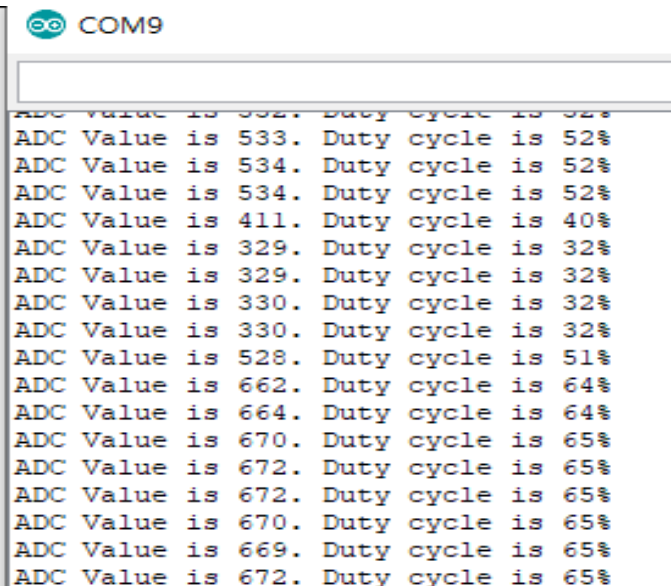


3.2 조도센서 제어하기

◆ 3.2.1 조도센서 값 시리얼 프린터로 출력

● 소스 코드

```
2 // 조도센서 Control_Serial -> cds_control
3
4 void setup()
5 {
6     // 시리얼 통신 설정
7     Serial.begin(9600);
8 }
9
10 void loop()
11 {
12     int adcValue; // 실제 센서로부터 읽은 값 (0~1023)
13     int duty;
14
15     adcValue = analogRead(A0);
16     duty = map(adcValue, 0, 1023, 0, 100);
17
18     // 시리얼 통신으로 출력
19     Serial.print("ADC Value is ");
20     Serial.print(adcValue);
21     Serial.print(". Duty cycle is ");
22     Serial.print(duty);
23     Serial.println("");
24     delay(500);
25 }
```



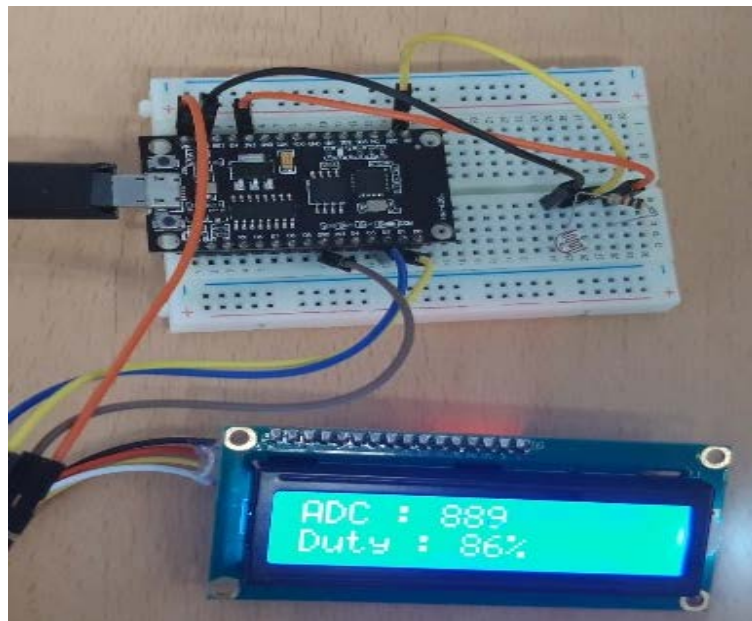
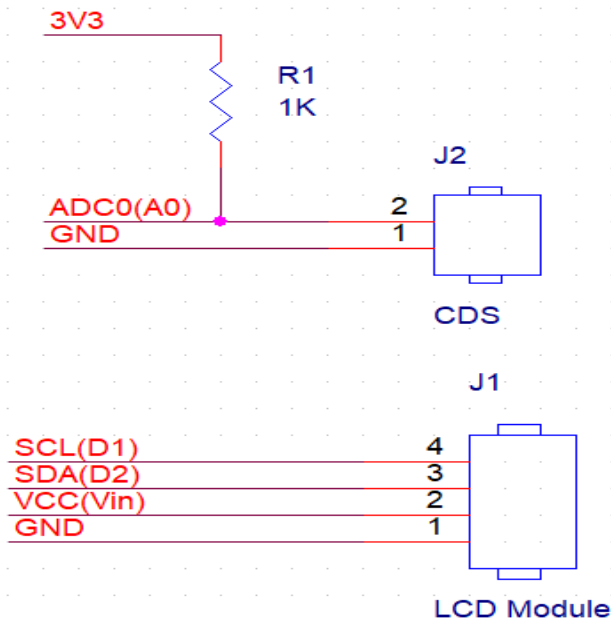
COM9

ADC Value is 533. Duty cycle is 52%
ADC Value is 534. Duty cycle is 52%
ADC Value is 534. Duty cycle is 52%
ADC Value is 411. Duty cycle is 40%
ADC Value is 329. Duty cycle is 32%
ADC Value is 329. Duty cycle is 32%
ADC Value is 330. Duty cycle is 32%
ADC Value is 330. Duty cycle is 32%
ADC Value is 528. Duty cycle is 51%
ADC Value is 662. Duty cycle is 64%
ADC Value is 664. Duty cycle is 64%
ADC Value is 670. Duty cycle is 65%
ADC Value is 672. Duty cycle is 65%
ADC Value is 672. Duty cycle is 65%
ADC Value is 670. Duty cycle is 65%
ADC Value is 669. Duty cycle is 65%
ADC Value is 672. Duty cycle is 65%

3.2 조도센서 제어하기

◆ 3.2.2 조도센서 값 LCD에 출력

- 회로도 및 연결도



3.2 조도센서 제어하기

◆ 3.2.2 조도센서 값 LCD에 출력

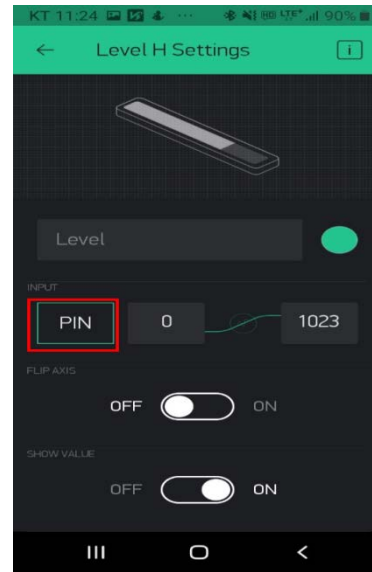
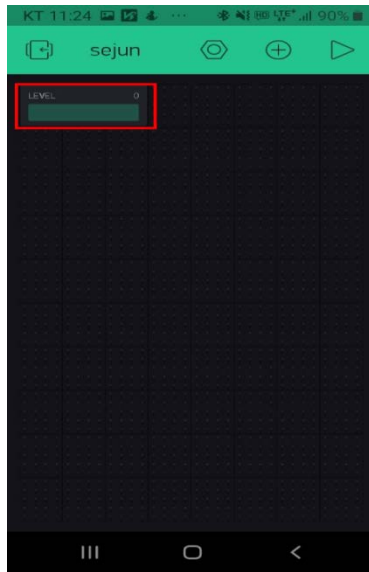
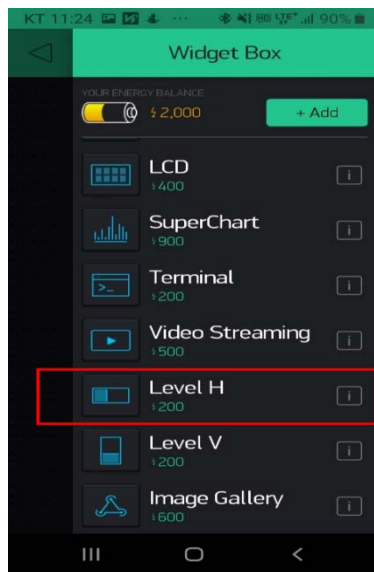
● 소스 코드

```
2 // 조도센서 Control_lcd -> cds_control_lcd
3
4 // I2C 통신 라이브러리 설정
5 #include <Wire.h>
6 // I2C LCD 라이브러리 설정
7 #include <LiquidCrystal_I2C.h>
8
9 // LCD I2C address 설정 PCF8574:0x27, PCF8574A:0x3F
10 LiquidCrystal_I2C lcd(0x20, 16, 2);
11
12 void setup()
13 {
14     Serial.begin(9600);
15     lcd.init();
16     lcd.backlight();
17 }
18
19 void loop()
20 {
21     int adcValue; // 실제 센서로부터 읽은 값 (0~1023)
22     int duty;
23
24     adcValue = analogRead(A0);
25     duty = map(adcValue, 0, 1023, 0, 100);
26
27     lcd.clear();
28     lcd.setCursor(0,0);
29     lcd.print("ADC : ");
30     lcd.print(adcValue);
31     lcd.setCursor(0,1);
32     lcd.print("Duty : ");
33     lcd.print(duty);
34     lcd.print("%");
35     delay(500);
36 }
```

3.2 조도센서 제어하기

◆ 3.2.3 Blynk 조도센서 값 읽어오기 - 가상핀

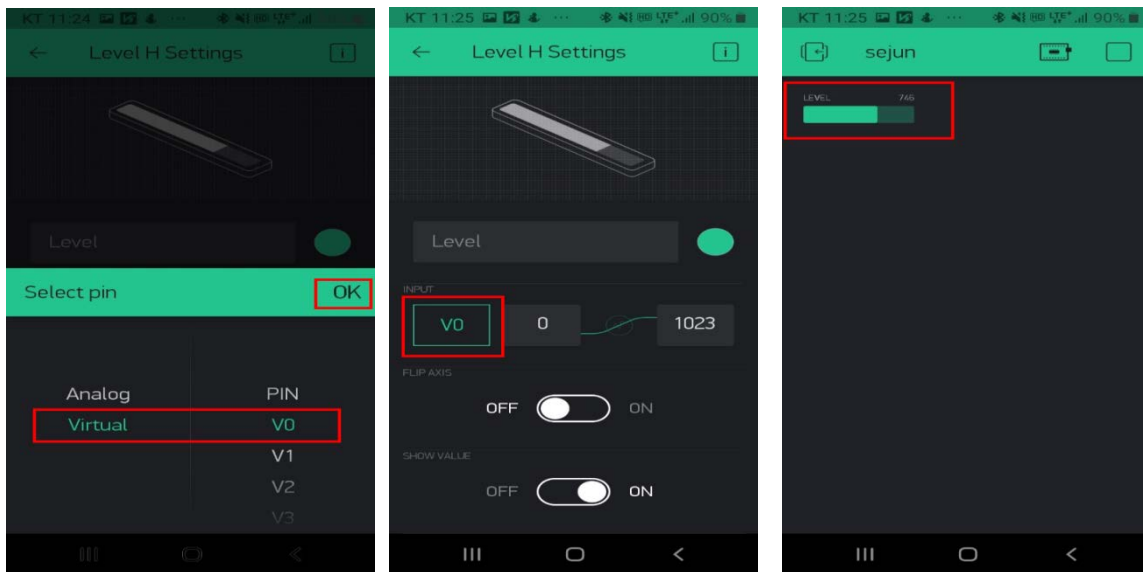
- Widget 설정



3.2 조도센서 제어하기

◆ 3.2.3 Blynk 조도센서 값 읽어오기 - 가상핀

- Widget 설정



3.2 조도센서 제어하기

◆ 3.2.3 Blynk 조도센서 값 읽어오기 - 가상핀

● 소스 코드

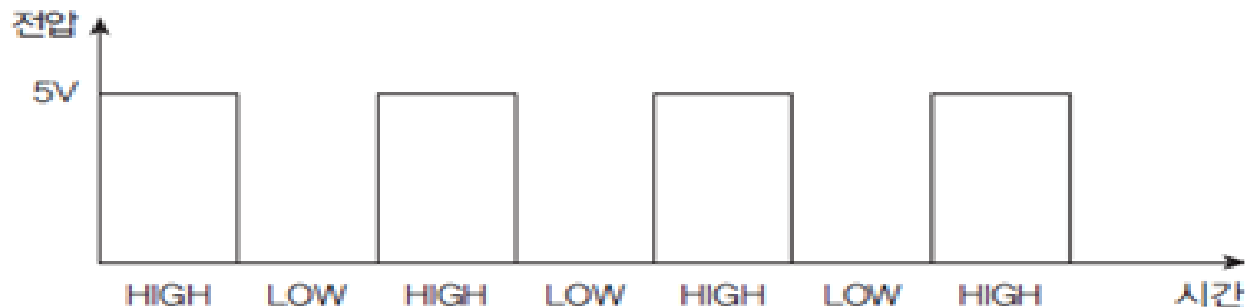
```
2 // 조도센서 Control_vpin -> cds_control_vpin(NodeMCU ->스마트폰)
3
4 // I2C 통신 라이브러리 설정
5 #include <Wire.h>
6 // I2C LCD 라이브러리 설정
7 #include <LiquidCrystal_I2C.h>
8 #include <ESP8266WiFi.h>
9 #include <BlynkSimpleEsp8266.h>
10 #define BLYNK_PRINT Serial
11 // LCD I2C address 설정 PCF8574:0x27, PCF8574A:0x3F
12 LiquidCrystal_I2C lcd(0x20, 16, 2);
13 // You should get Auth Token in the Blynk App.
14 // Go to the Project Settings (nut icon).
15 char auth[] = "KWIEYYjACwL_4t3z-5wNjHSxw3Uftqy4"; //
16
17 // Your WiFi credentials.
18 // Set password to "" for open networks.
19 char ssid[] = "sjpark";
20 char pass[] = "12345678";

22 void setup()
23 {
24     Blynk.begin(auth, ssid, pass);
25 }
26
27 void loop()
28 {
29     Blynk.run();
30
31     int adcValue = analogRead(A0);
32
33     Blynk.virtualWrite(V0, adcValue);
34     delay(100);
35 }
```

3.3 버튼 제어하기

◆ 디지털 신호

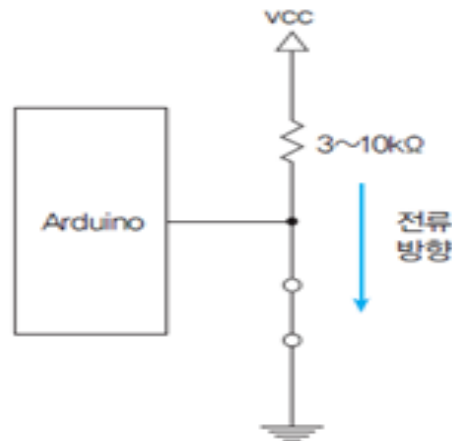
- 0(Low)과 1(High) 두 가지 값으로 표현되는 신호
- 잡음에 강하고 데이터의 저장 및 처리가 용이



3.3 버튼 제어하기

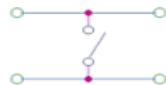
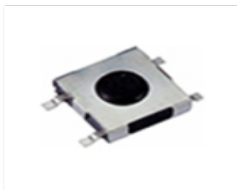
◆ 풀업(Pull-up)/풀다운(Pull-down)

- 디지털 신호 입력핀에 아무것도 연결되지 않았을 때 High 혹은 Low 신호로 만들어 안정시킴



3.3 버튼 제어하기

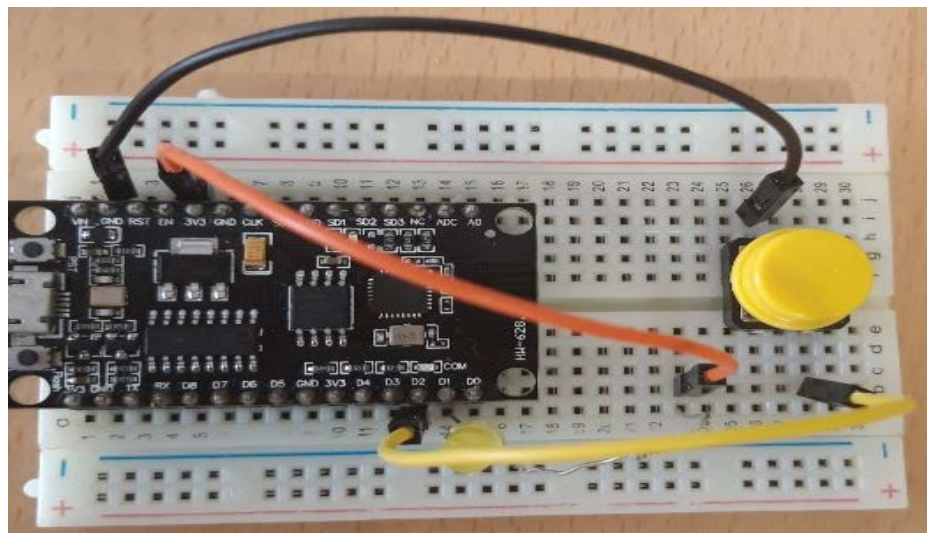
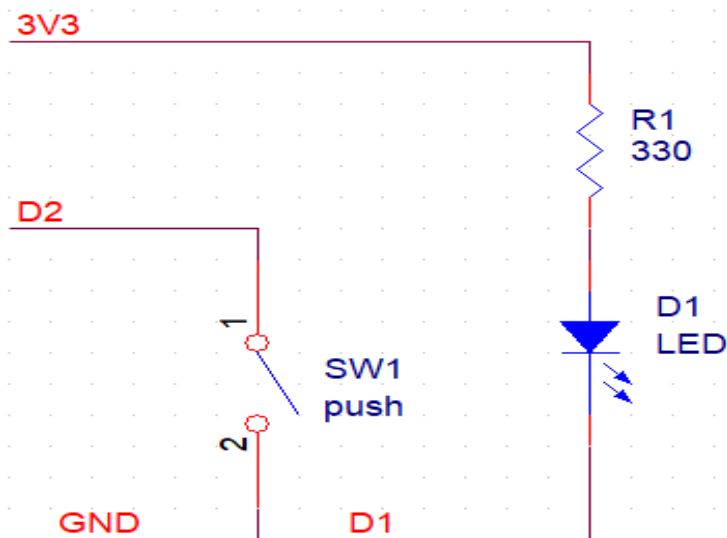
◆ 다양한 형태의 푸쉬 스위치(push switch)



3.3 버튼 제어하기

◆ 3.3.1 버튼으로 LED on/off 제어

- 회로도 및 연결도



3.3 버튼 제어하기

◆ 3.3.1 버튼으로 LED on/off 제어(누르면-on, 떼면-off)

- 소스 코드

```
2 // Button Control -> bt_control
3 // 버튼 누르면 (on), 버튼 떼면 (off)
4
5 const int ledPin = D1;
6 const int inputPin = D2;
7
8 void setup()
9 {
10     pinMode(D1, OUTPUT);
11     pinMode(D2, INPUT_PULLUP);
12 }
13
14 void loop()
15 {
16     int button = digitalRead(D2); // 스위치 입력
17
18     if(button == 0) digitalWrite(ledPin, 0); // 누르면 (LED 점등)
19     else digitalWrite(ledPin, 1);           // 떼면 (LED 소등)
20 }
```

3.3 버튼 제어하기

◆ 3.3.1 버튼으로 LED on/off 제어(누를 면 on/off 변화)

● 소스 코드

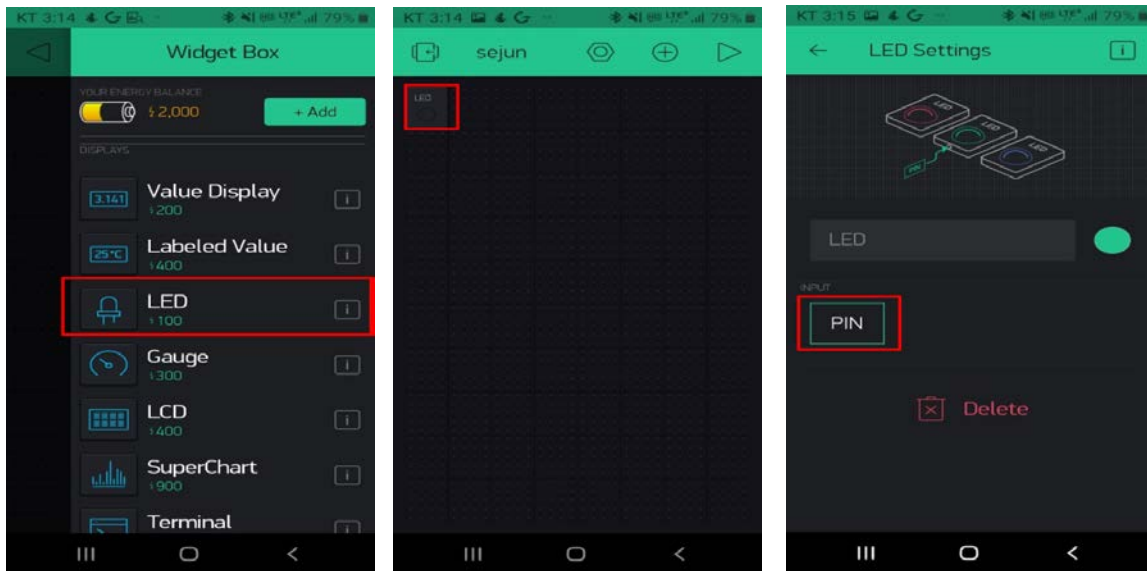
```
2 // Button Control -> bt_control_button
3 // 누를때마다 on/off change
4
5 int btnFlag=0;
6
7 void setup()
8 {
9   pinMode(D1, OUTPUT);
10  pinMode(D2, INPUT_PULLUP);
11  Serial.begin(9600);
12 }
```

```
14 void loop()
15 {
16   int button = digitalRead(D2); // 스위치 입력
17
18   if(button == 0) // 버튼 누르면
19   {
20     if(btnFlag==0) btnFlag=1;
21     else btnFlag=0;
22   }
23
24   Serial.print("Button Flag : ");
25   Serial.println(btnFlag);
26   delay(500);
27
28   if(btnFlag==0)digitalWrite(D1, 1); // LED 소등
29   else digitalWrite(D1, 0); // LED 점등
30 }
```

3.3 버튼 제어하기

◆ 3.3.2 Blynk 버튼으로 가상 LED 제어 - 가상핀

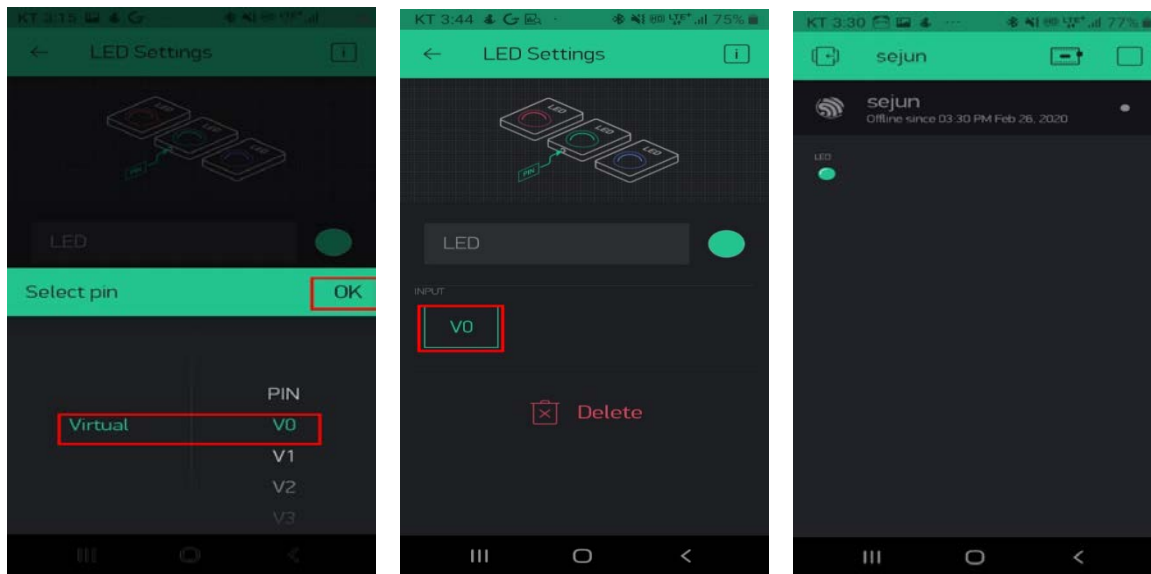
- Widget 설정



3.3 버튼 제어하기

◆ 3.3.2 Blynk 버튼으로 가상 LED 제어 - 가상핀

- Widget 설정



3.3 버튼 제어하기

◆ 3.3.2 Blynk 버튼으로 가상 LED 제어 - 가상핀

- 소스 코드

```
2 // Button Control_vpin -> bt_control_vpin(NodeMCU ->스마트폰)
3 // 누를때마다 on/off change
4
5 #define BLYNK_PRINT Serial
6 #include <ESP8266WiFi.h>
7 #include <BlynkSimpleEsp8266.h>
8
9 // You should get Auth Token in the Blynk App.
10 // Go to the Project Settings (nut icon).
11 char auth[] = "KWiEYYjACwL_4t3z-5wNjHSxw3Uftqy4";
12
13 // Your WiFi credentials.
14 // Set password to "" for open networks.
15 char ssid[] = "sjpark";
16 char pass[] = "12345678";
17
18 int button=0;
19 int btnFlag=0;
20
21 WidgetLED led(V0);
```

3.3 버튼 제어하기

◆ 3.3.2 Blynk 버튼으로 가상 LED 제어 - 가상핀

- 소스 코드

```
23 void setup()
24 {
25     // Debug console
26     Serial.begin(9600);
27     Blynk.begin(auth, ssid, pass);
28     pinMode(D2, INPUT_PULLUP);
29     pinMode(D1, OUTPUT);
30 }
```

```
32 void loop()
33 {
34
35     Blynk.run();
36     button = digitalRead(D2);
37
38     if(button == 0)
39     {
40         if(btnFlag==0) btnFlag=1;
41         else btnFlag=0;
42     }
43
44     if(btnFlag==0)
45     {
46         digitalWrite(D1, 1); // LED 소등
47         led.off();           // LED 소등
48     }
49     else
50     {
51         digitalWrite(D1, 0); // LED 점등
52         led.on();             // LED 점등
53     }
54
55     delay(10);
56 }
```