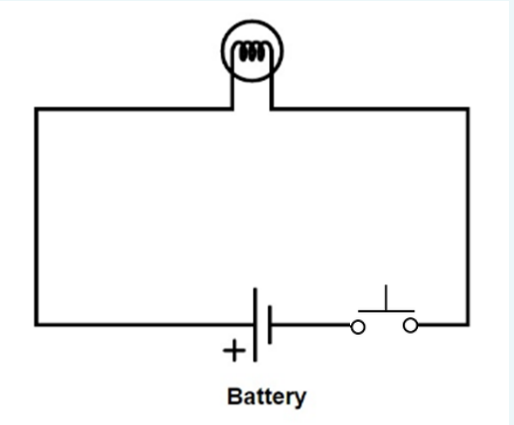
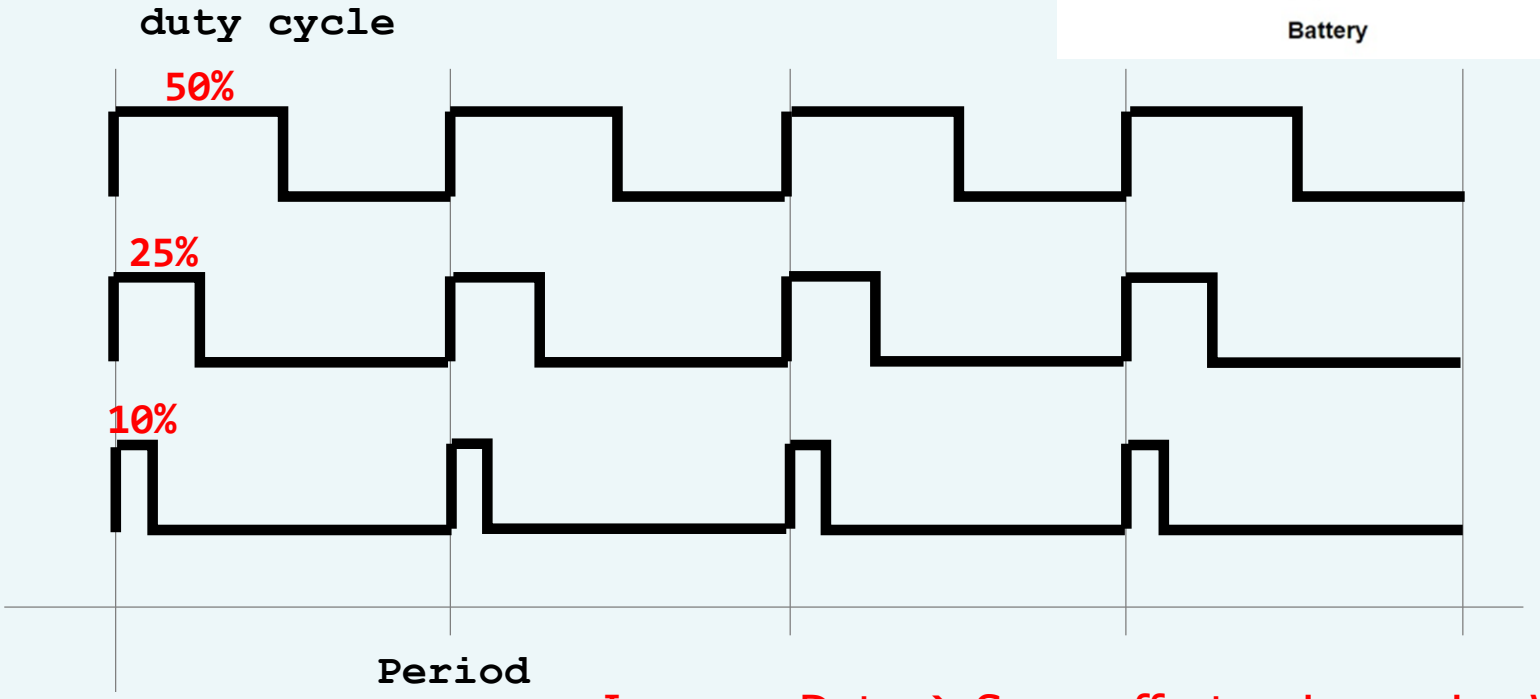
Internet of Things
class 4

ESP32 PWM, ADC, Analog Input, Analog Sensors

Pulse-Width Modulation (PWM)

- PWM: Pulse Width Modulation
  - Adjust output voltage
  - Adjust brightness of lamp,
    control speed of motor





Increase Duty → Same effect as increasing Voltage

C.B.Choi gen1223@kau.ac.kr

ESP32 Pulse-Width Modulation (PWM)

- 16 independent channels with different properties
- Steps (EX: to dim an LED with PWM)
  - Choose a PWM channel (0 ~ 15)
  - Set the PWM signal frequency (ex: 5000Hz for an LED)
  - Set the signal's duty cycle resolution (1 ~ 16bits)
    - Ex: 8-bit resolution controls LED brightness from 0 to 255

ledcSetup(**channel, freq, resolution**)

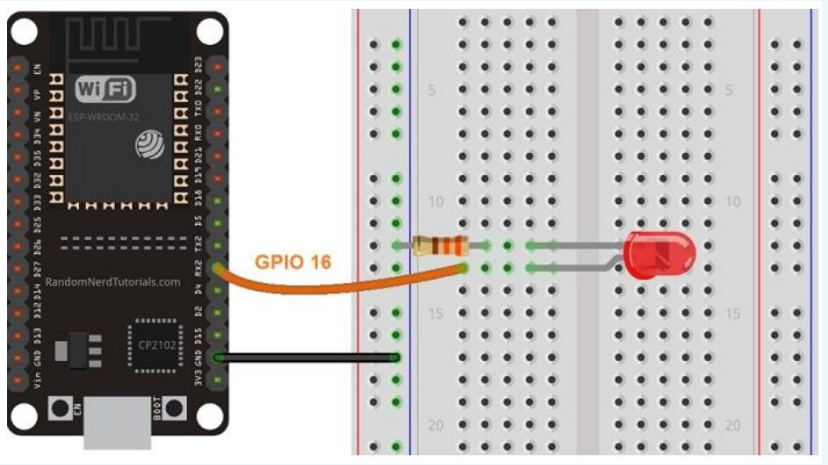
- Specify GPIO to output the signal

ledcAttachPin(**GPIO**, **channel**)

- Control PWM with dutycycle

ledcWrite(**channel**, **dutycycle**)

What effect for High Frequecy ?



C.B.Choi gen1223@kau.ac.kr

Pulse-Width Modulation (PWM)

<Task04-1>

```
// <Task 04-1>
// the number of the LED pin
   const int ledPin = 16;   // GPIO16
// setting PWM properties
   const int freq = 5000;
   const int ledChannel = 0;
   const int resolution = 8;

void setup() {
// configure LED PWM functionalitites
   ledcSetup(ledChannel, freq, resolution);
// attach the channel to the GPIO
   ledcAttachPin(ledPin, ledChannel);
}
```

```
void loop() {
// increase the LED brightness
   for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++) {
      // changing the LED brightness with PWM
      ledcWrite(ledChannel, dutyCycle);
      delay(15);
   }

// decrease the LED brightness
   for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--) {
      // changing the LED brightness with PWM
      ledcWrite(ledChannel, dutyCycle);
      delay(15);
   }
}
```

Same Signal on Different GPIOs

- Attach GPIOs to the same channel on the setup()

Steps (EX: to dim three LEDs with a PWM value)

- Choose a PWM channel (0 ~ 15)
- Set the PWM signal frequency (ex: 5000Hz for an LED)
- Set the signal's duty cycle resolution (1 ~ 16bits)
  - Ex: 8-bit resolution controls LED brightness from 0 to 255

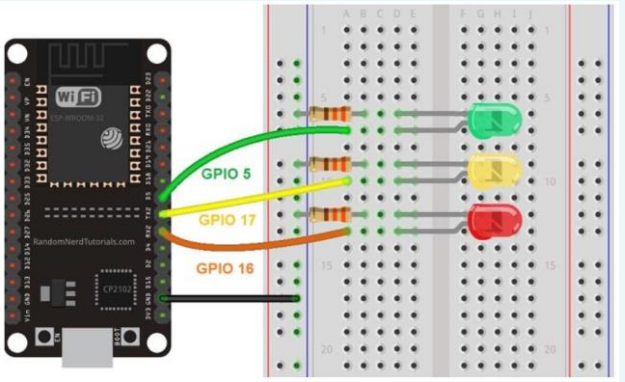
ledcSetup(**channel, freq, resolution**)

- Specify GPIO to output the signal

ledcAttachPin(**GPIO**, **channel**) X 3

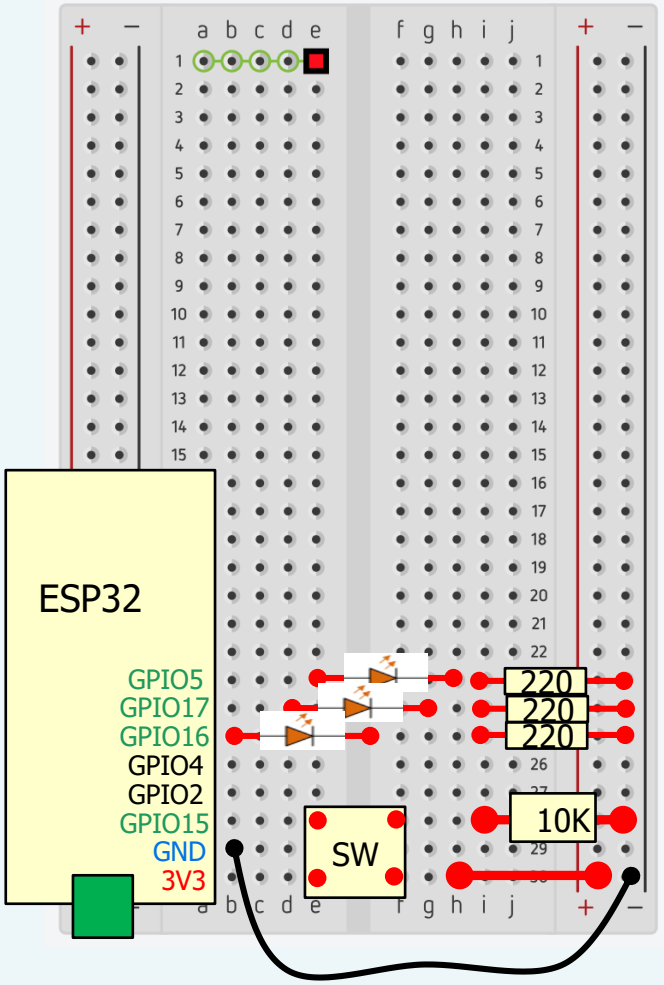
- Control PWM with dutycycle

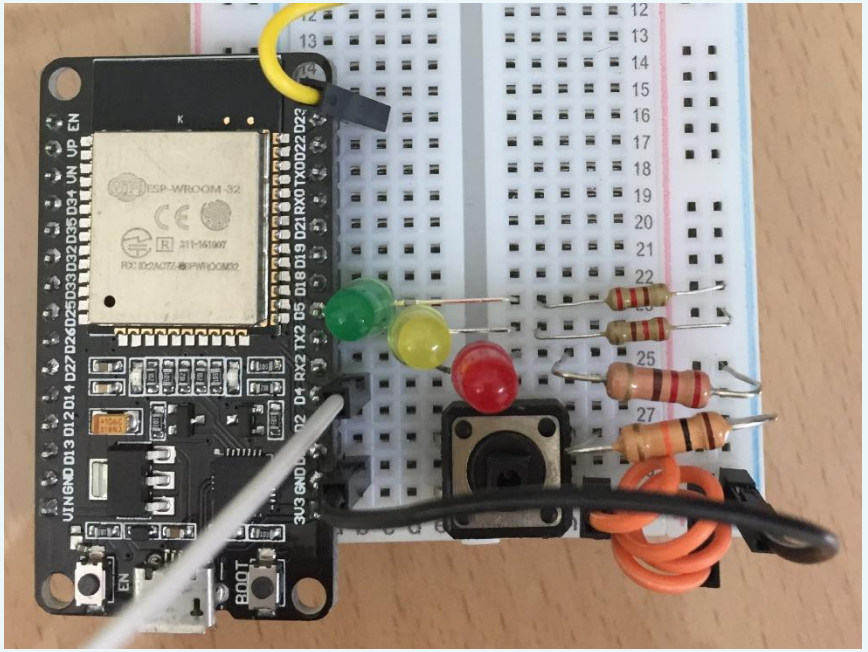
ledcWrite(**channel**, **dutycycle**)



Same Signal on Different GPIOs

** 두개의 LED와 저항 추가: GPIO5, 17 **





Same Signal on Different GPIOs

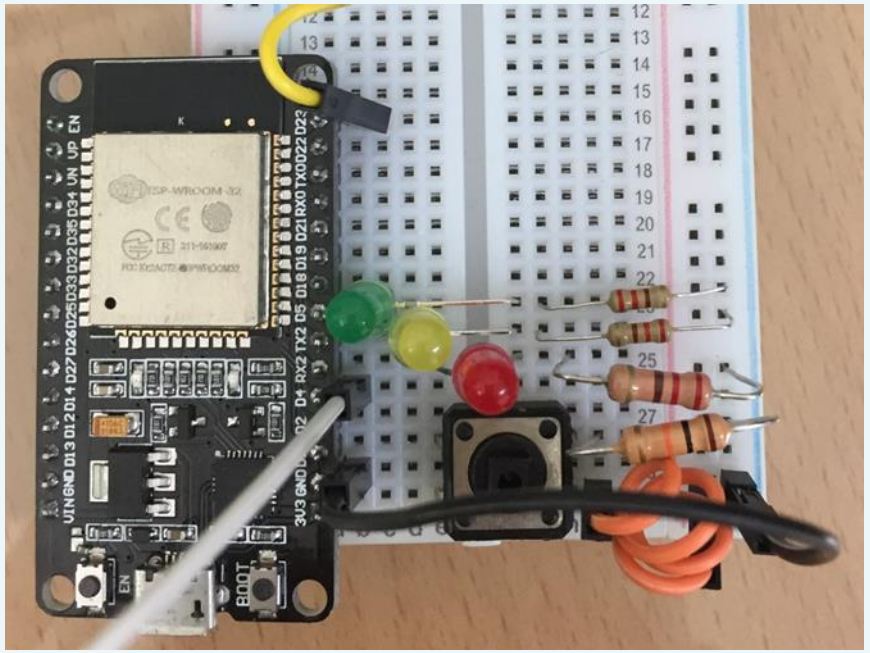
<Task04-2>

```
// the number of the LED pin
const int ledPin = 16;    // 16 corresponds to GPIO16
const int ledPin2 = 17;  // 17 corresponds to GPIO17
const int ledPin3 = 5;   // 5 corresponds to GPIO5
// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

void setup() {
// configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);
// attach the channel to the GPIOs
    ledcAttachPin(ledPin, ledChannel);
    ledcAttachPin(ledPin2, ledChannel);
    ledcAttachPin(ledPin3, ledChannel);
}
```

```
void loop(){
// increase the LED brightness
    for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++) {
        // changing the LED brightness with PWM
        ledcWrite(ledChannel, dutyCycle);
        delay(15);
    }

// decrease the LED brightness
    for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--) {
        // changing the LED brightness with PWM
        ledcWrite(ledChannel, dutyCycle);
        delay(15);
    }
}
```

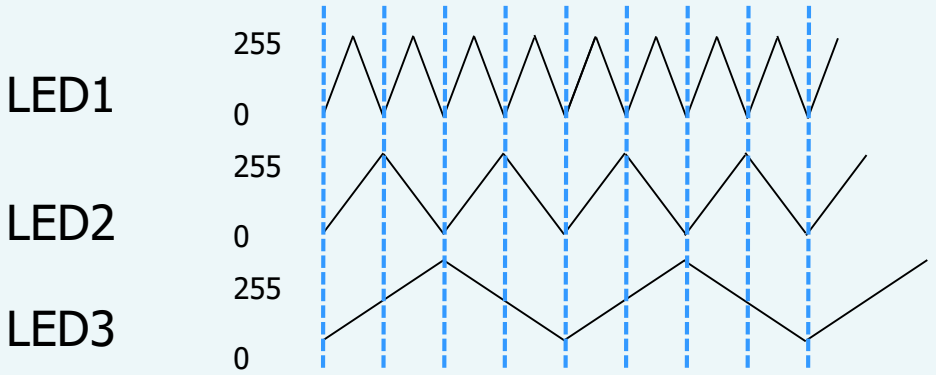
Different Signals on Different GPIOs

- 3개의 PWM 채널을 3개의 LED에 연결하자



<Task04-3>

- 3개의 PWM 채널로 3개의 LED 제어
- freq = 5000, duty-res = 8bits
- pwm 값: 다음과 같은 주기로 0~255
- 스위치로 Pause 기능



C.B.Choi gen1223@kau.ac.kr

Different Signals on Different GPIOs

<Task04-3>

```
// the number of the LED pin
const int ledPin = 16;    // 16 corresponds to GPIO16
const int ledPin2 = 17;  // 17 corresponds to GPIO17
const int ledPin3 = 5;   // 5 corresponds to GPIO5
const int buttonPin = 15;

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int ledChannel1 = 1;
const int ledChannel2 = 2;
const int resolution = 8;

void setup() {
   Serial.begin(115200);
// configure LED PWM functionalitites
   ledcSetup(ledChannel, freq, resolution);
// attach the channel to the GPIOs
   ledcAttachPin(ledPin, ledChannel);
   ledcAttachPin(ledPin2, ledChannel1);
   ledcAttachPin(ledPin3, ledChannel2);
   pinMode(buttonPin, INPUT);
}
```

Different Signals on Different GPIOs

```
void loop(){
// increase the LED brightness
    int j = 0, k = 0, jd = 1, kd = 1;
    for(int i = 0; i <= 1023; i++) {
        // changing the LED brightness with PWM
        ledcWrite(ledChannel, i/4);
        ledcWrite(ledChannel1, j/2);
        if ((j += jd) == 512) jd = -1, j = 511;
        ledcWrite(ledChannel2, k);
        if ((k += kd) >= 256) kd = -1, k = 255;
        else if (k < 0) kd = 1, k = 0;
        checkPause();  delay(5);
    }
// decrease the LED brightness
    j = 0, k = 0, jd = 1, kd = 1;
    for(int i = 1023; i >= 0; i--) {
        // changing the LED brightness with PWM
        ledcWrite(ledChannel, i/4);
        ledcWrite(ledChannel1, j/2);
        if ((j += jd) == 512) jd = -1, j = 511;
        ledcWrite(ledChannel2, k);
        if ((k += kd) >= 256) kd = -1, k = 255;
        else if (k < 0) kd = 1, k = 0;
        checkPause();  delay(5);
    }
}
```

```
void checkPause() {
  // toggle pause state if button pressed
  if (digitalRead(buttonPin)) {
      Serial.println("in pause..");
      delay(250);     // skip glitch
      while (!digitalRead(buttonPin)) ;
      delay(250);     // skip glitch
      Serial.println("exit from pause..");
  }
}
```

Making tones with PWM

- PWM을 이용한 Tone 발생
  - 부저 출력 (수동부저)
  - Tone별 설정 주파수
  - 극성이 있다:
    +: GPIO23에 연결
    -: GND에 연결

음별 주파수

C3 131	**C4 262 - 도**	**C5 523**
CS3 139	CS4 277	CS5 554
D3 147	**D4 294 - 레**	**D5 587**
DS3 156	DS4 311	DS5 622
E3 165	**E4 330 - 미**	**E5 659**
F3 175	**F4 349 - 파**	**F5 698**
FS3 185	FS4 370	FS5 740
G3 196	**G4 392 - 솔**	**G5 784**
GS3 208	GS4 415	GS5 831
A3 220	**A4 440 - 라**	**A5 880**
AS3 233	AS4 466	AS5 932
B3 247	**B4 494 - 시**	**B5 988**

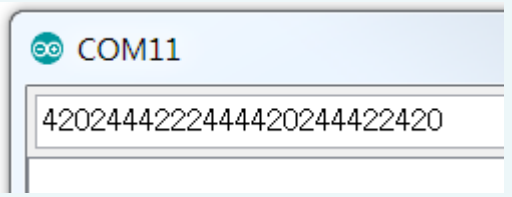
<Task04-4>

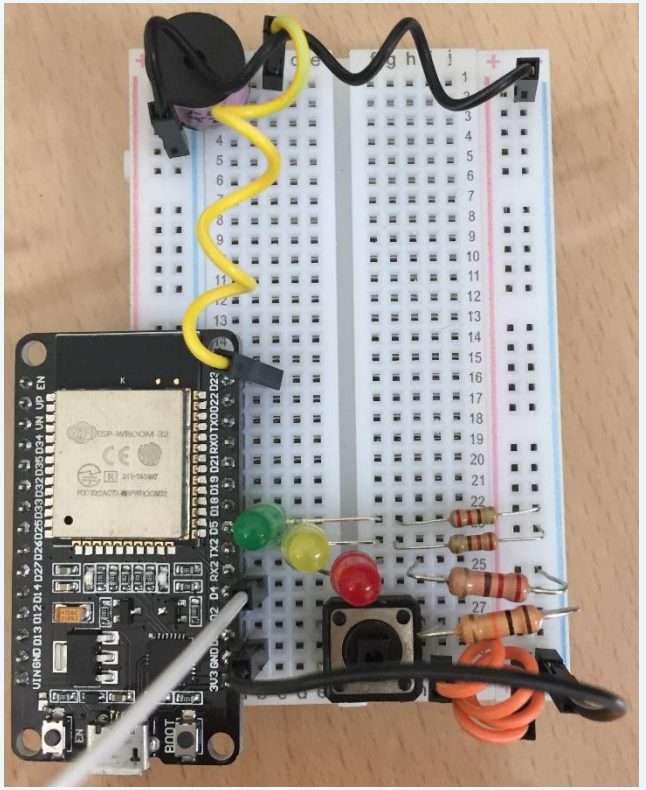
```
// make tones by using pwm
//   freq -> pitch
//   duty -> volume (?)

// setting PWM properties
//const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;
const int buzPin = 23;
const int duty = 128;

// variables
int sVal;

// notes
//enum Notes {C3=0, CS3, D3, DS3, E3, F3};
int nFrq[] = {/*131, 139, 147, 156, 165, 175*/
            262, 277, 294, 311, 330, 349};
```

Making tones with PWM

- Serial 입력을 통해 연주
  - 모든 음의 재생 시간을 250ms로 한다

```
void playNote(int note, int dur) {
    ledcSetup(ledChannel, nFrq[note], resolution);
    ledcWrite(ledChannel, duty);
    Serial.println(note);
    delay(dur);
}

void setup() {
    Serial.begin(115200);
    ledcAttachPin(buzPin, ledChannel);
}

void loop(){
    if (Serial.available() > 0) {
        sVal = Serial.read();
        playNote(sVal-0x30, 250);
    }
}
```





Analog Sensors

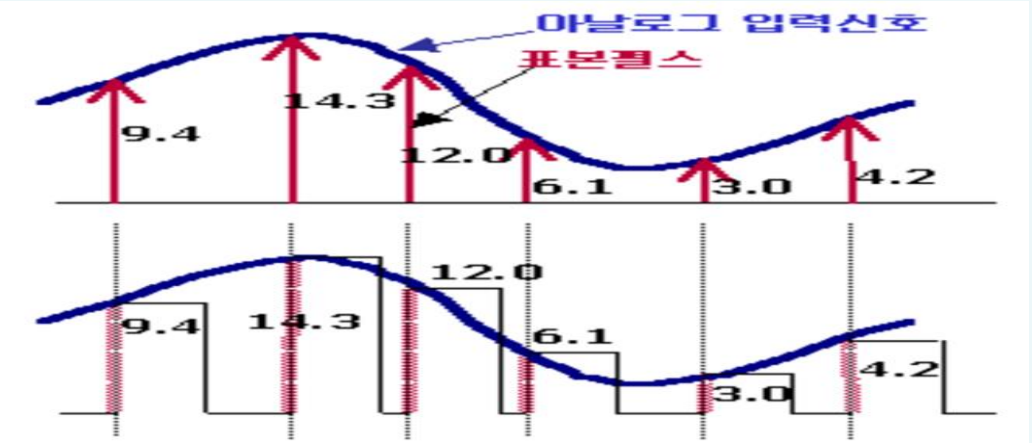
- Analog Sensors
  - Illuminance, Temperature(TMP35/36/37), Gas, Sound, Accelerometer, Potentiometer, joystick, etc.

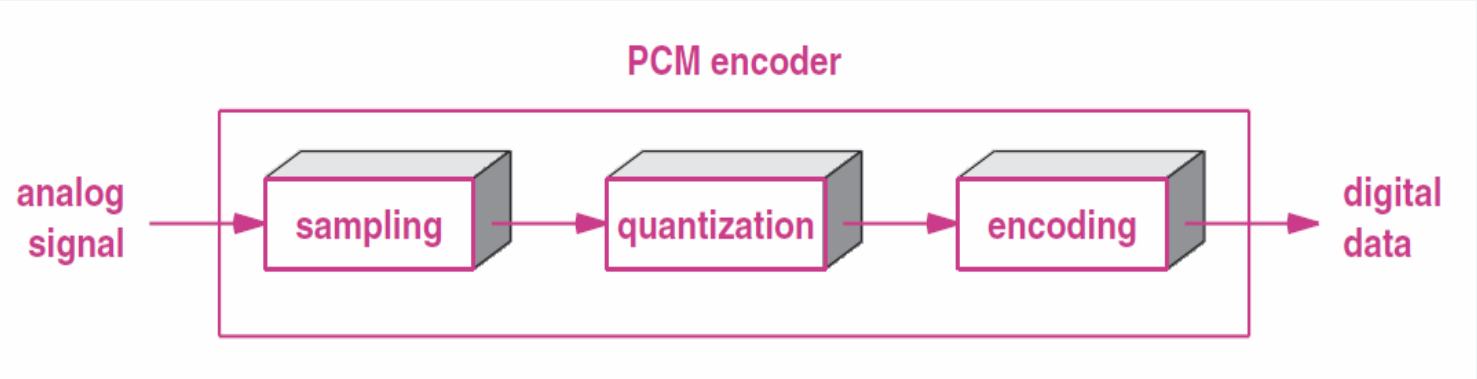





Analog Input - Pulse Code Modulation

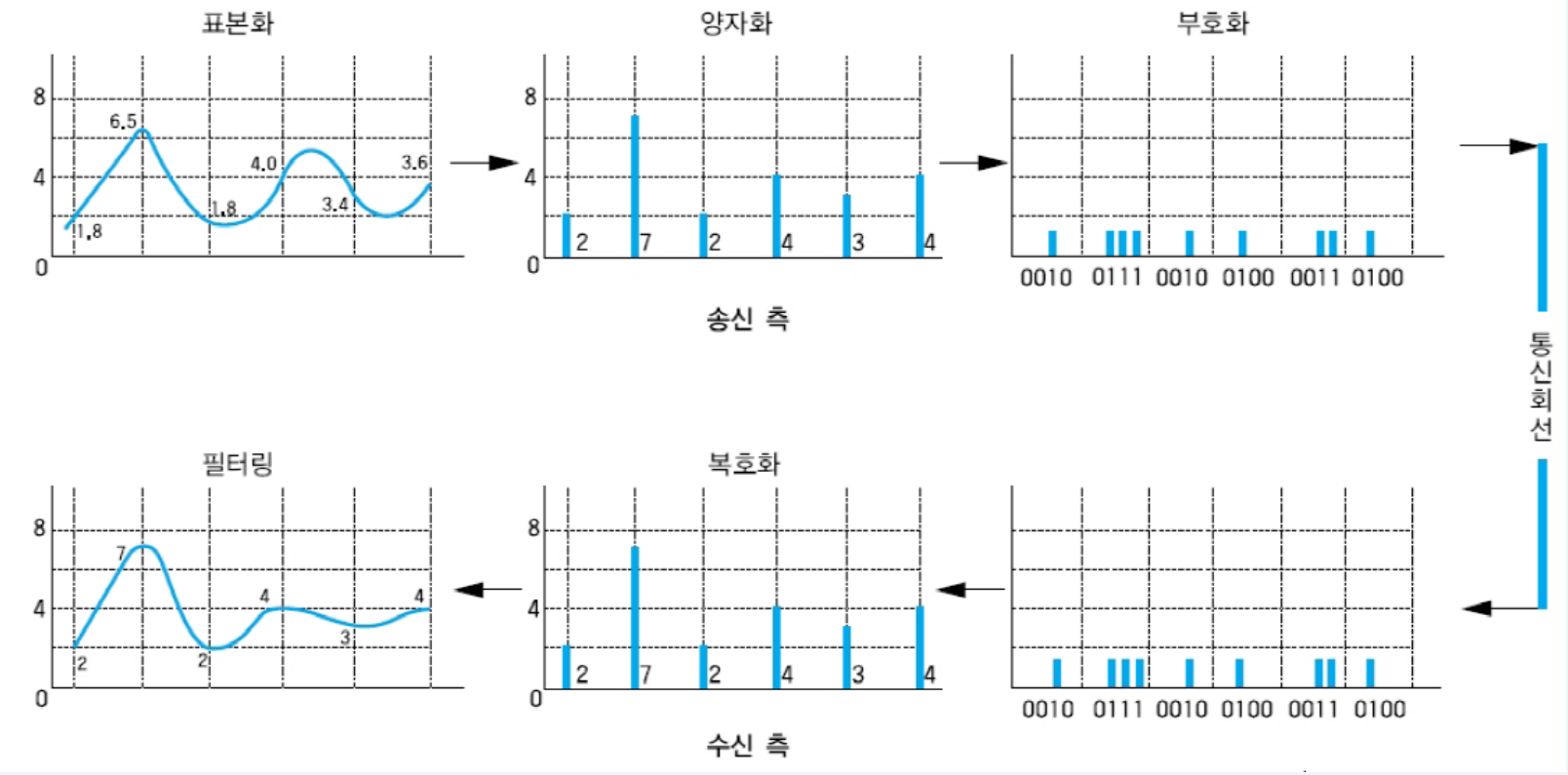
- Analog Signal
  - Pulse Code Modulation (PCM)





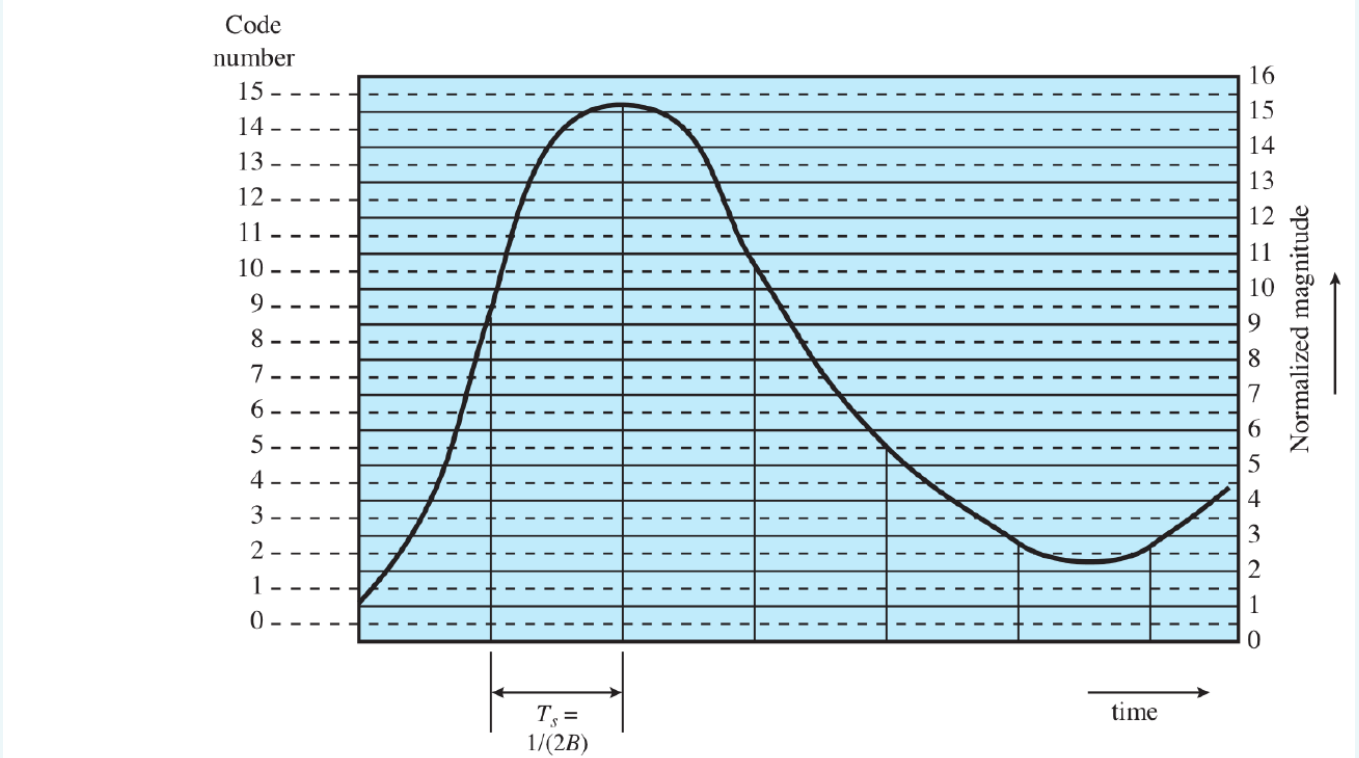
Pulse Code Modulation

- Analog Signal
  - Pulse Code Modulation (PCM)
  - Sampling -> Quantizing -> Encoding -> Send/Receive -> Decoding -> Filtering



Pulse Code Modulation

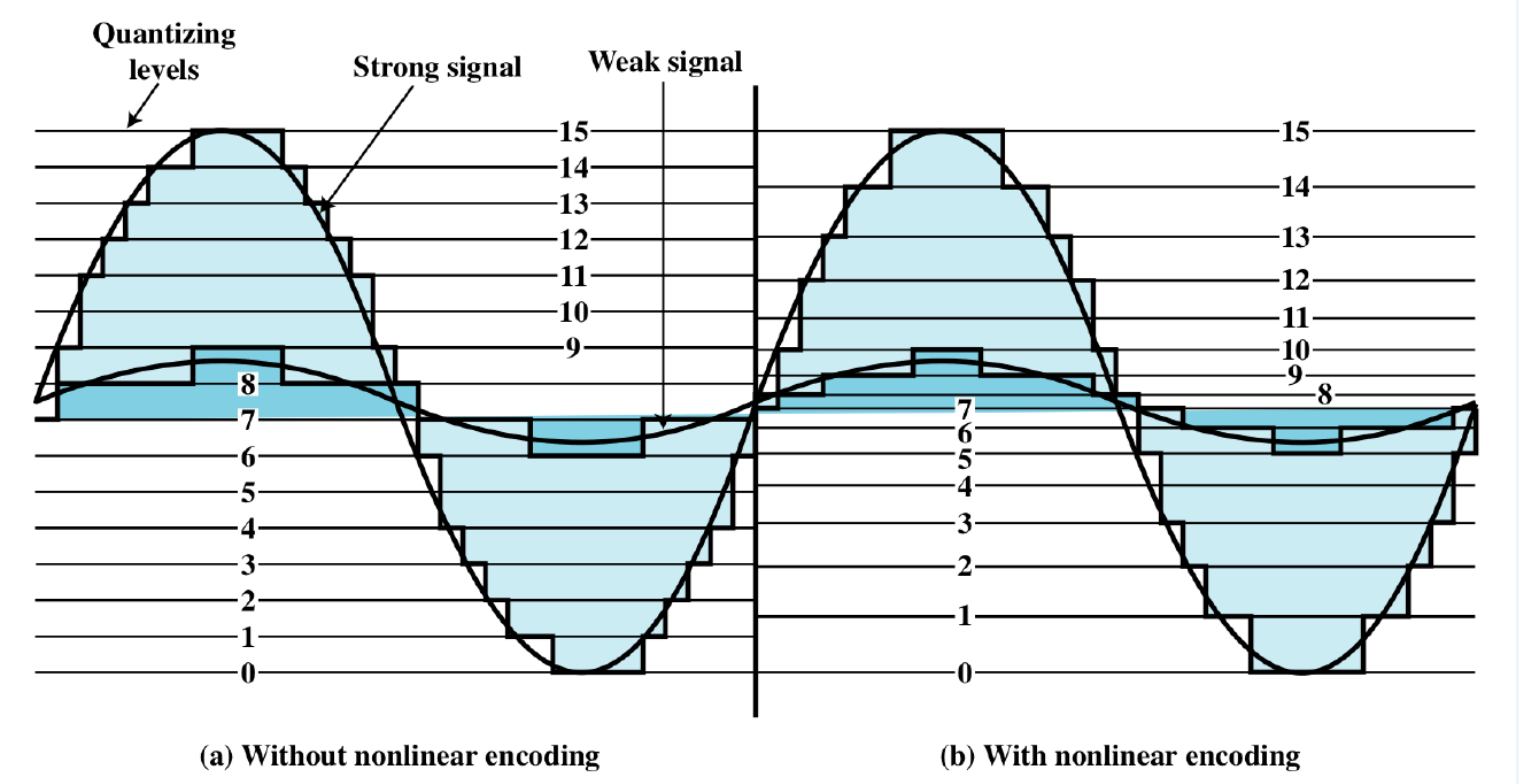
- Analog Signal
  - PCM Example



PAM value	1.1	9.2	15.2	10.8	5.6	2.8	2.7
quantized code number	1	9	15	10	5	2	2
PCM code	0001	1001	1111	1010	0101	0010	0010

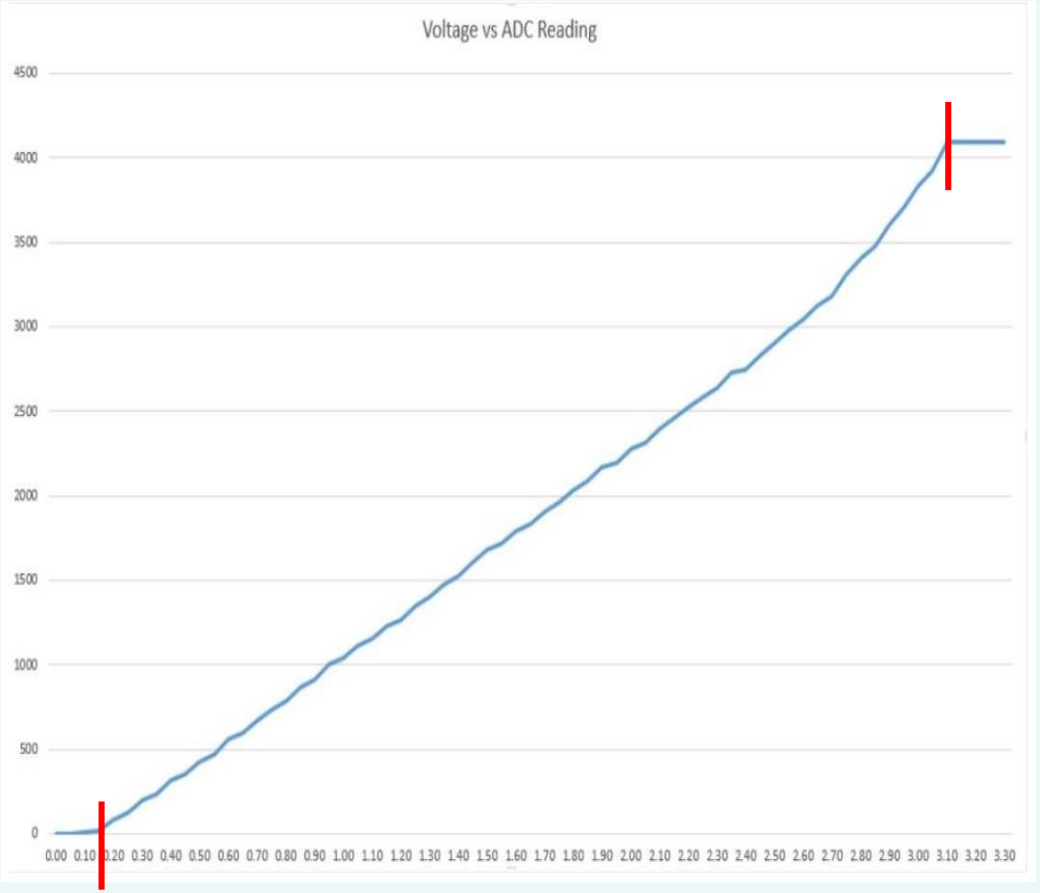
Pulse Code Modulation

- Analog Signal
  - PCM with Nonlinear Coding



ESP32 Analog Inputs

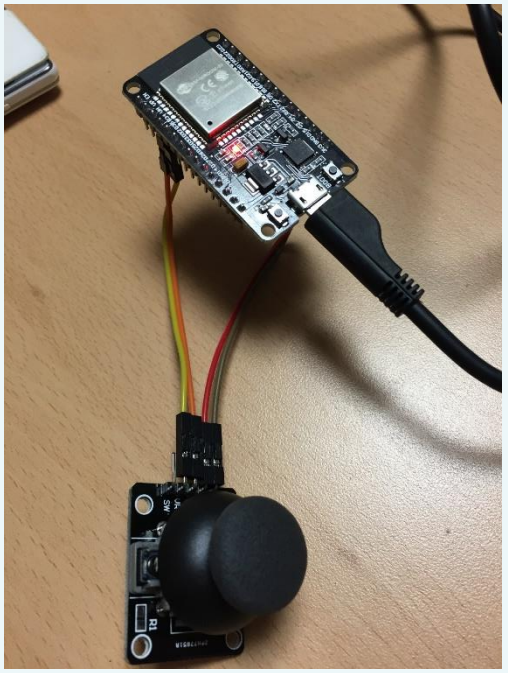
- Can measure Voltage 0V ~ 3.3V
- Assigned to a value 0 ~ 4095
- ESP32 ADC is not linear
  - cannot distinguish:
    - 0V ~ 0.1V
      (value = 0)
    - 3.2V ~ 3.3V
      (value = 4095)
  - Keep in mind!!

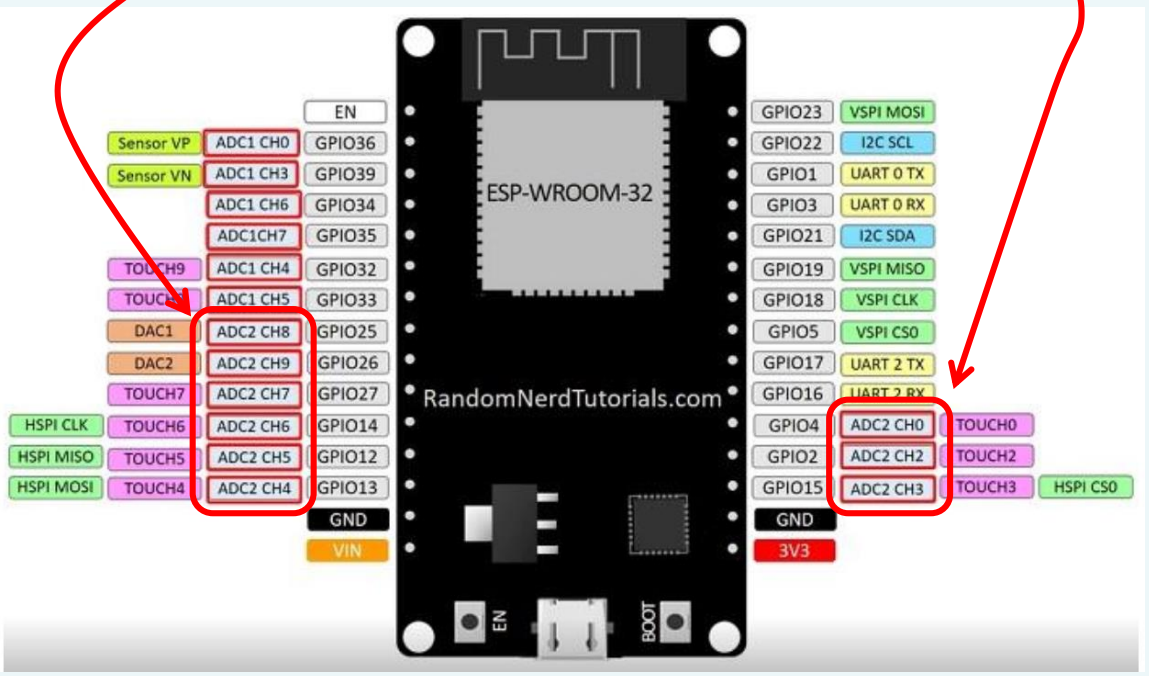


C.B.Choi gen1223@kau.ac.kr

Reading Analog Inputs

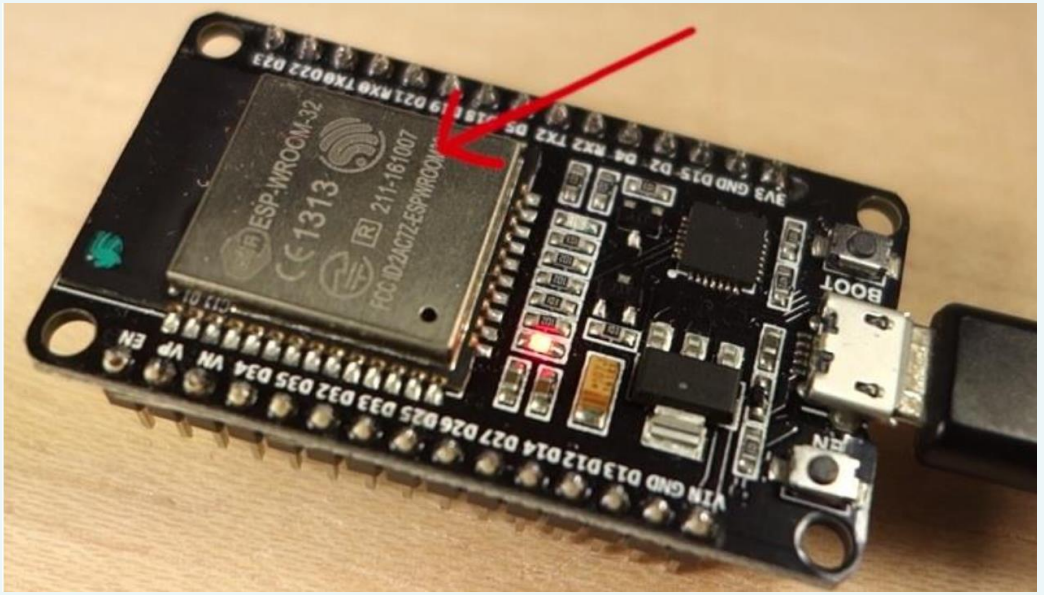
- ESP32 has 18 different channels
- But only 15 are available in DEVKIT V1 DOIT
- 12 bits resolution: 0 ~ 4095
- analogRead(**GPIO**)





Analog Input - Hall Effect Sensor

- ESP32 has a built-in hall effect sensor
  - can detect variations in the magnetic field
  - greater the magnetic field, the greater the output voltage
  - Increase or become negative depending on the magnet pole facing the sensor



Hall Effect Sensor

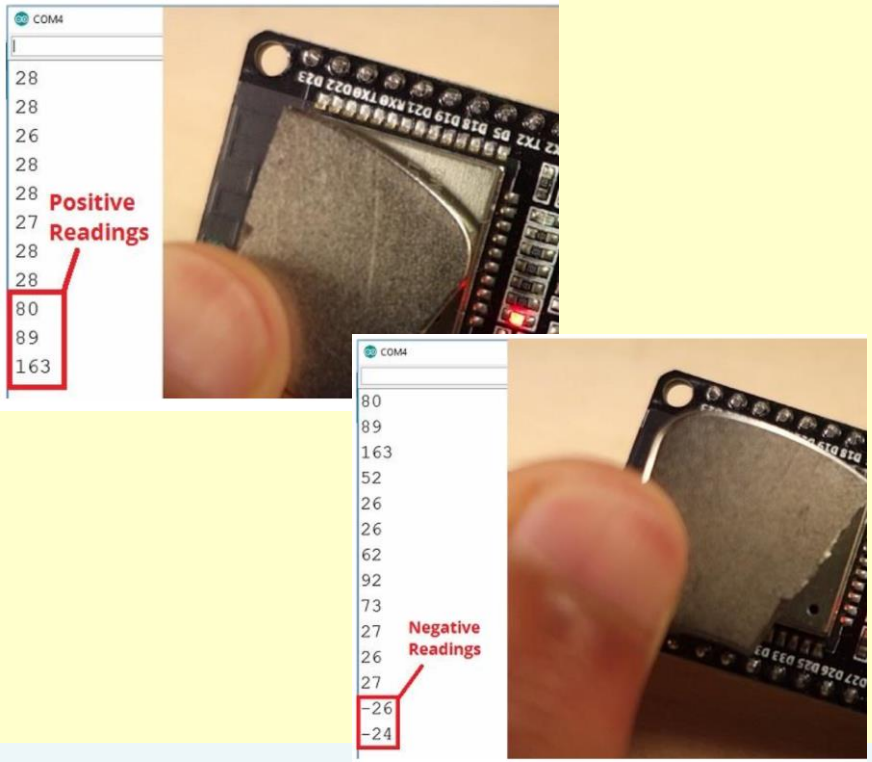
- Threshold detection to act as a switch
- Hall effect sensors are mainly used to:
  - Detect proximity
  - Calculate positioning
  - Count the number of revolutions of a wheel
  - Detect a door closing;

Analog Input - Hall Effect Sensor

- hallRead()
  Examples> ESP32> HallSensor

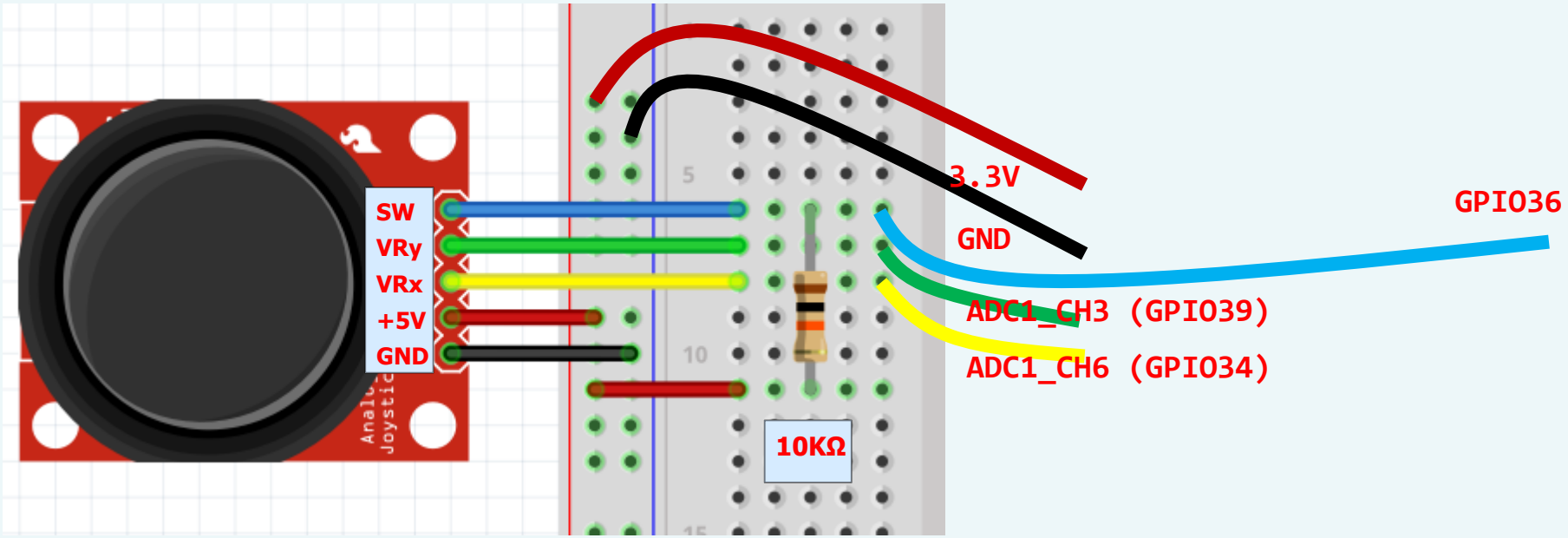
<Task04-5>

```
// Simple sketch to access the internal hall effect detector on the esp32.
// values can be quite low.
// Brian Degger / @sctv
int val = 0;

void setup() {
Serial.begin(115200);
}

// put your main code here, to run repeatedly
void loop() {
    // read hall effect sensor value
    val = hallRead();
    // print the results to the serial monitor
    Serial.println(val);
    delay(1000);
}
```



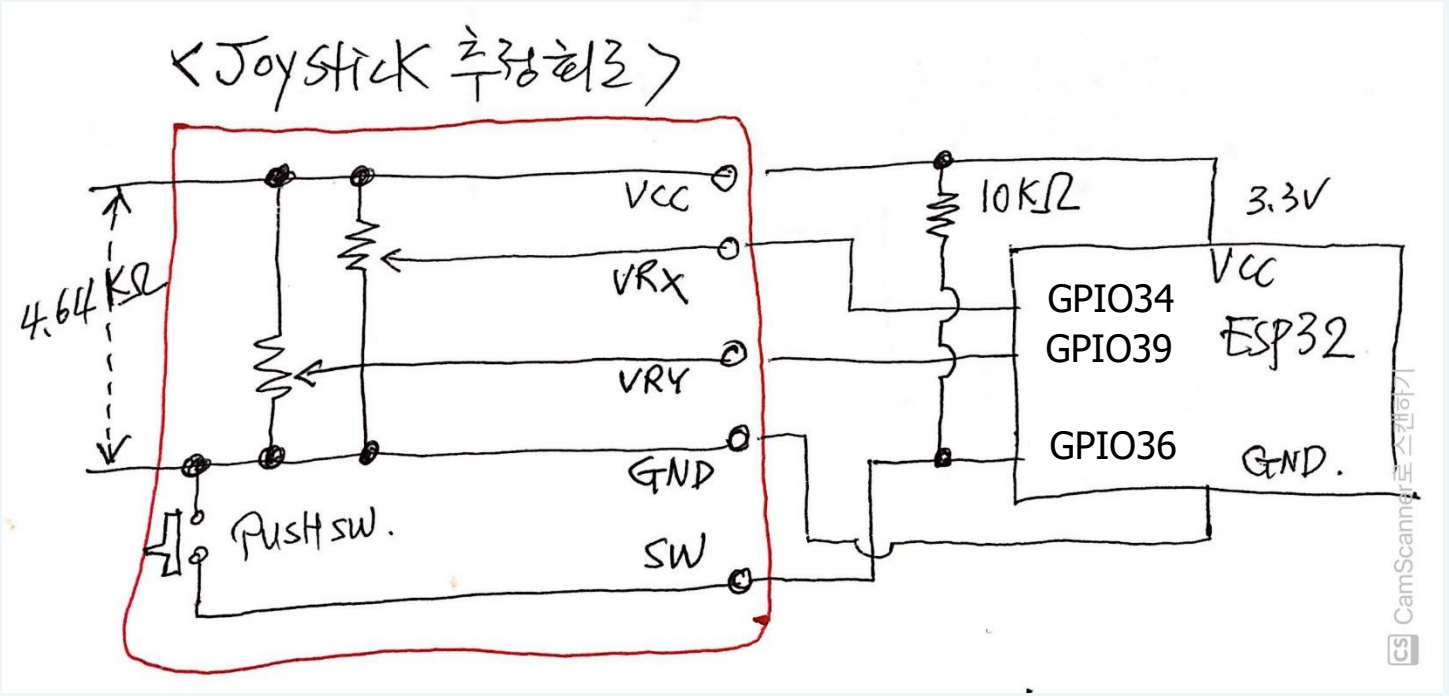
Analog Input - Joystick

- Joystick의 Analog Input .. analogRead(GPIO)
  - VRx → GPIO34 (ADC1_CH6)
  - VRy → GPIO39 (ADC1_CH3)
- Joystick의 Digital Input
  - Switch → GPIO36



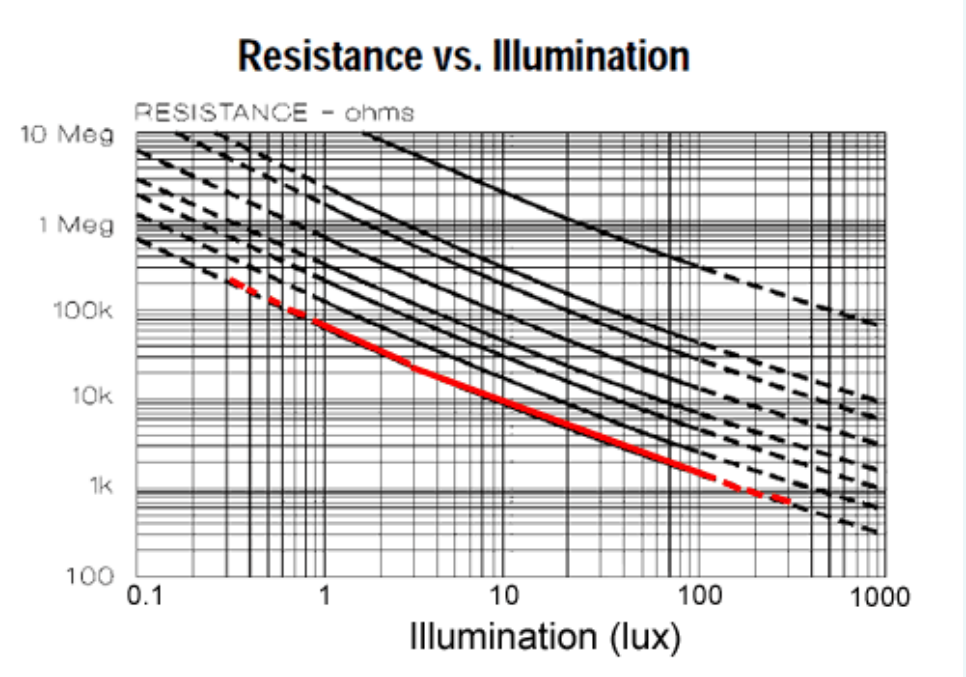
Analog Input - Joystick

- Joystick에는 두개의 Potentiometer와 한 개의 Button이 있다
  - 추정회로.. ESP32 연결 회로.. Pull-up or Pull-down ?



Analog Input - CdS Cell

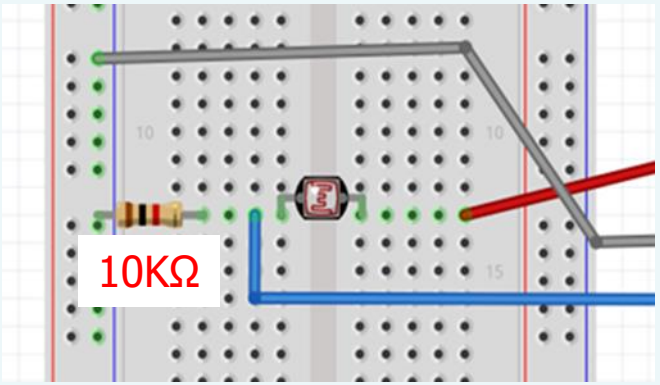
- CdS cell (Cadmium sulphide cell): 조도센서
  - CdS 셀은 가시광선의 양에 따라 저항도 변화
    - 광량이 증가할수록 저항은 감소
    - CdS 셀은 0.3~300 lux 범위에서 가장 밝은 300 lux일 때 저항의 크기가 약 700Ω
    - 가장 어두울 때인 0.3 lux일 때 저항의 크기는 200kΩ



C.B.Choi gen1223@kau.ac.kr

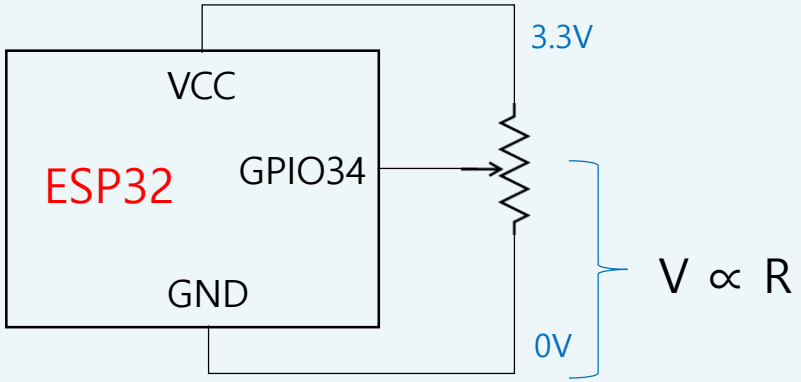
Analog Input – CdS Cell Circuit

- CdS의 저항의 변화에 따른 전압을 ADC로 입력
- GPIO35 → ADC1_CH7 .. analogRead(35)



Analog Input- Potentiometer, CdS

- 가변저항, 조도센서의 신호를 읽는 방법은 유사하다

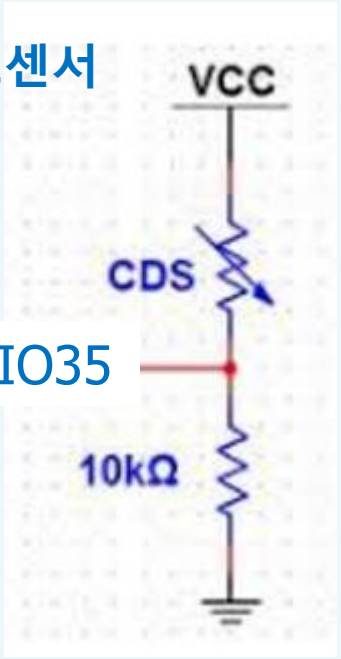


- GPIO34 전압은 GND~GPIO34 저항값에 비례
- analogRead(34) 는 12bit (0~4095) 입력 (ADC)
- 누설전류 (Leakage): 가변저항이 10KΩ 이면,

  I = 3.3 / 10000 = 0.33mA

가변저항 위치	GND~GPIO34 전압	analogRead
0	0V	0
30%	1V	1229
50%	1.7V	2048
100%	3.3V	4095

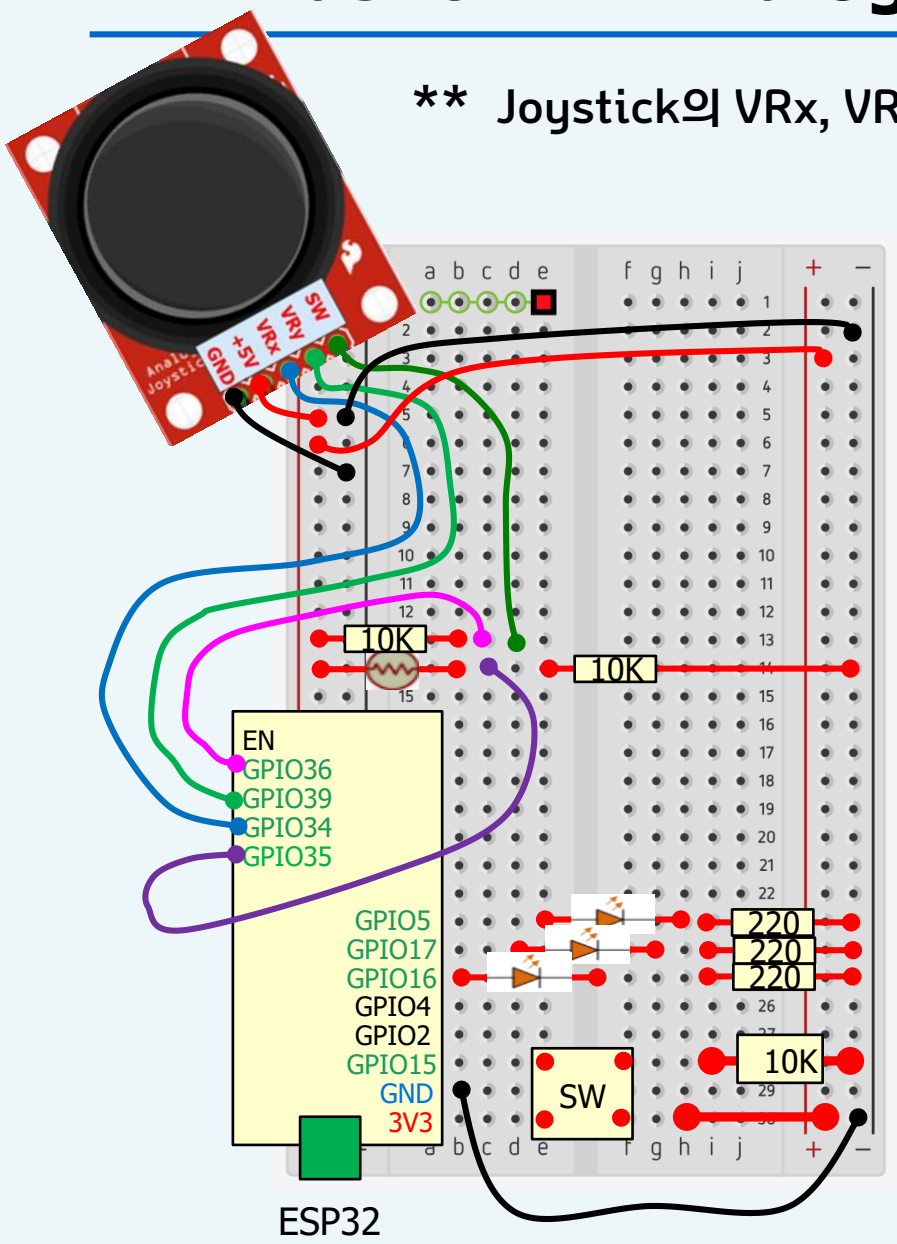
조도센서

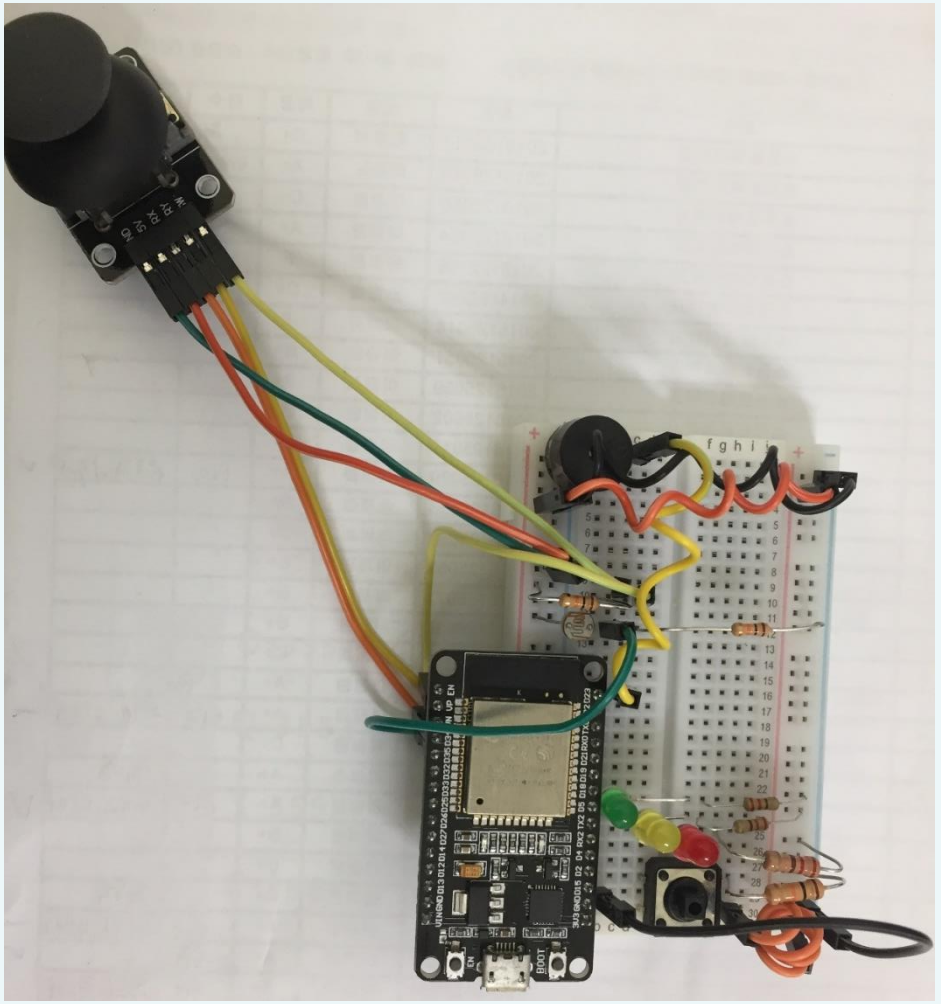


$$V_{A0} = \frac{R}{Rcds+R} * Vcc$$

Task04-A Analog Input- Joystick, CdS

** Joystick의 VRx, VRy, SW와 CdS를 연결하라 **





C.B.Choi gen1223@kau.ac.kr

Task04-A Reading Analog Inputs

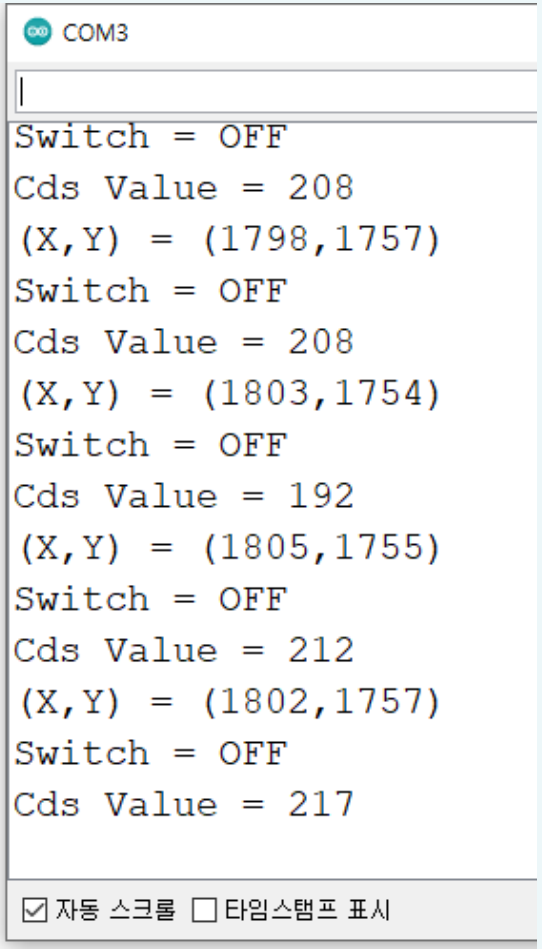
<Task04-A>

```
// Joystick, CdS test
// Joystick is connected to GPIO 34/39 (Analog ADC1_CH6/3)
// Cds is connected to GPIO15 (Analog ADC1_CH7)
const int joyPinX = 34;
const int joyPinY = 39;
const int joySW = 36;
const int cdsPin = 35;

// variable for storing the Joystick values
int xValue = 0, yValue = 0, sValue = 0, cValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(joySW, INPUT);
  delay(1000);
}
```

```
void loop() {
// Reading Joystick values
  xValue = analogRead(joyPinX);
  yValue = analogRead(joyPinY);
  cValue = analogRead(cdsPin);
  sValue = digitalRead(joySW);
  String str = "(X,Y) = (" + String(xValue)
               + ',' + String(yValue) + ")\n";
  Serial.print(str);
  Serial.print("Switch = ");
  Serial.println(sValue?"OFF":"ON"); //
pull-up circuit
  Serial.print("Cds Value = ");
  Serial.println(cValue);
  delay(500);
}
```

Task04-A Reading Analog Inputs - result

- Check CdS value.. Change luminance
- Check Joystick value.. stick left, right, up, down
- Check Joystick switch.. Press button



** Explain the relationship between CdS value and brightness ?

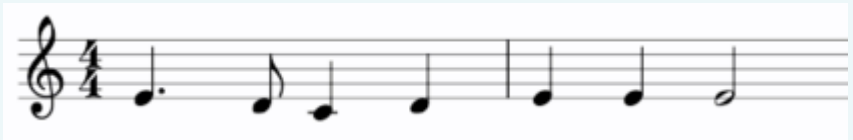
** Why ?

** Change the circuit to get the reverse result

C.B.Choi gen1223@kau.ac.kr

Task04-Step C Play station with PWM

- <Task04-4>를 수정하여 Music Player를 구현하라
  - 입력: 시리얼모니터 사용
  - 한 음을
    <Note#> : 음의 높이
    <Duration#> : 음의 길이
    으로 표현
    Ex)



43 26 04 24 44 44 42

C4	262 - 도
CS4	277
D4	294 - 레
DS4	311
E4	330 - 미
F4	349 - 파
FS4	370
G4	392 - 솔
GS4	415
A4	440 - 라
AS4	466
B4	494 - 시
C5	523

Note #
0
1
2
3
4
5
6
7
8
9
a
b
c
, 쉼표

	Duration#	.. Delay
𝅝	0 ...	2000
𝅗𝅥.	1 ...	1500
𝅗𝅥	2 ...	1000
♩.	3 ...	750
♩	4 ...	500
♪.	5 ...	375
♪	6 ...	250

** 쉼표를 연주하기 위해서는 PWM의 Duty를 0로 한다

Task04-Step C Play station with PWM

- 다음을 연주하라