



01076101

วิศวกรรมคอมพิวเตอร์เบื้องต้น

Introduction to Computer Engineering

Arduino #3

Analog Read, LDR, 7 Segment



# Analog Read

- นอกจาก Arduino จะสามารถอ่านค่าในแบบ Digital แล้ว ยังให้ขาสำหรับอ่านค่าแบบ Analog (ไม่ใช่แค่ 0,1) มาด้วย จำนวน 6 ขา คือ A0-A5 โดยค่าที่อ่านจะอยู่ระหว่าง 0-1023 โดย 0=0v และ 1023=5v

**Syntax:**

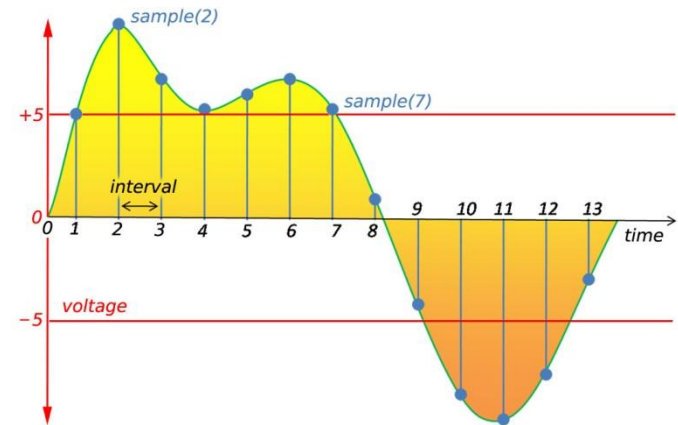
`analogRead(pin)`

**Parameter:**

**pin:** the number of the pin whose mode you wish to set

**Return:**

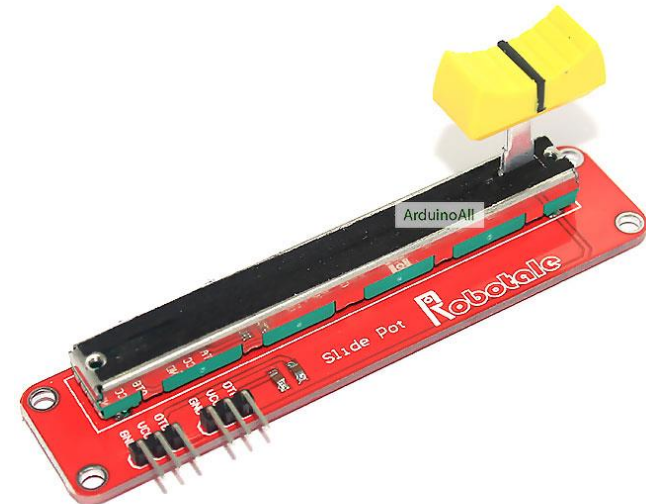
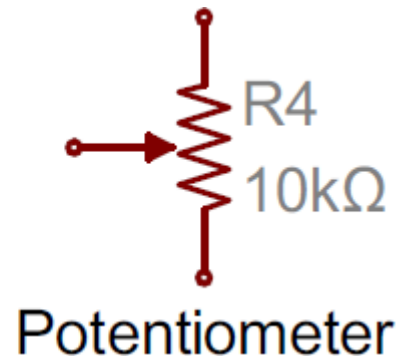
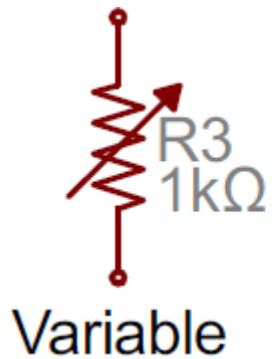
Integer : 0-1023 (0-5V)





# Potentiometer

- คือ ตัวต้านทานแบบปรับค่าได้





# Analog Read

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;       // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor

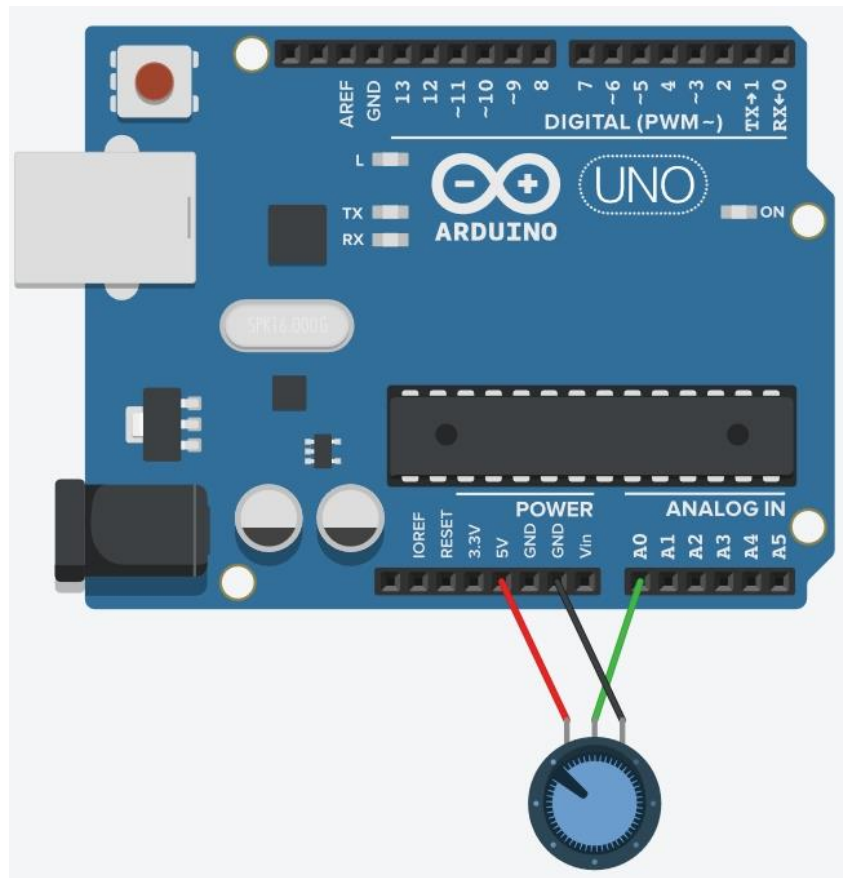
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(sensorPin);    // read the value from the sensor:
  Serial.println(sensorValue);            // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

# Activity



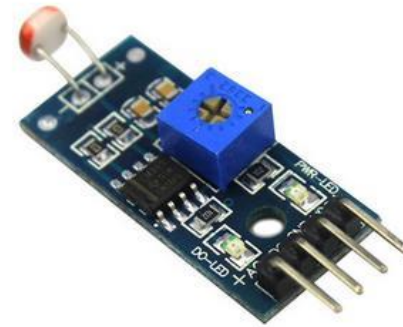
- ให้ต่อวงจรตามรูป และเขียนโปรแกรมเพื่ออ่านค่ามาแสดงใน Serial Plotter และลองหมุนดู



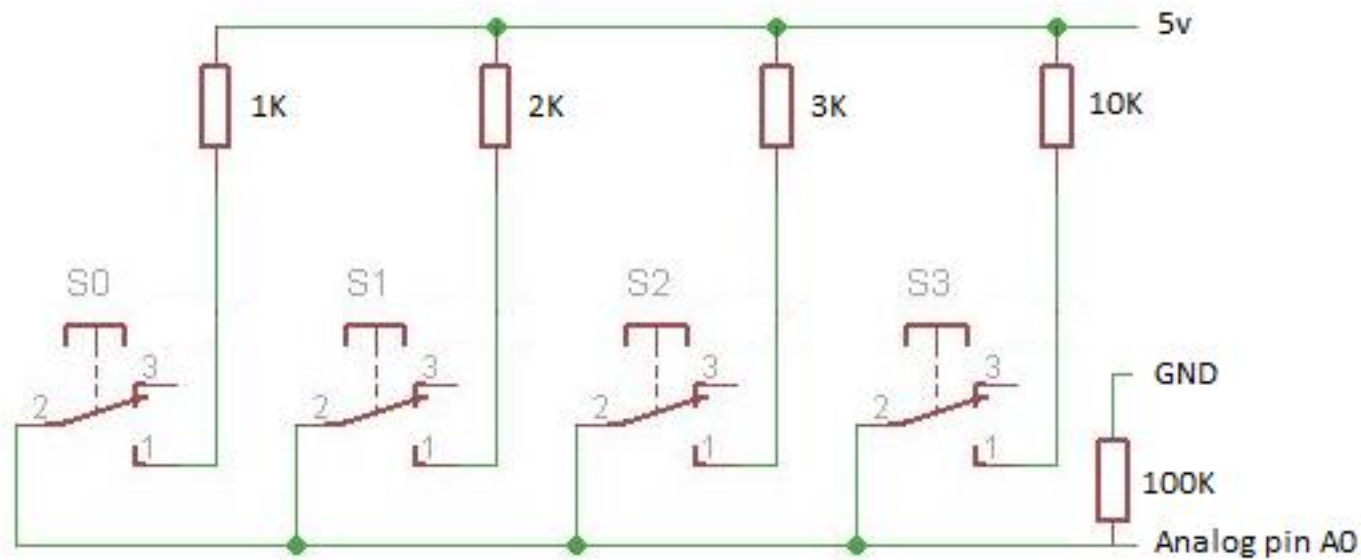


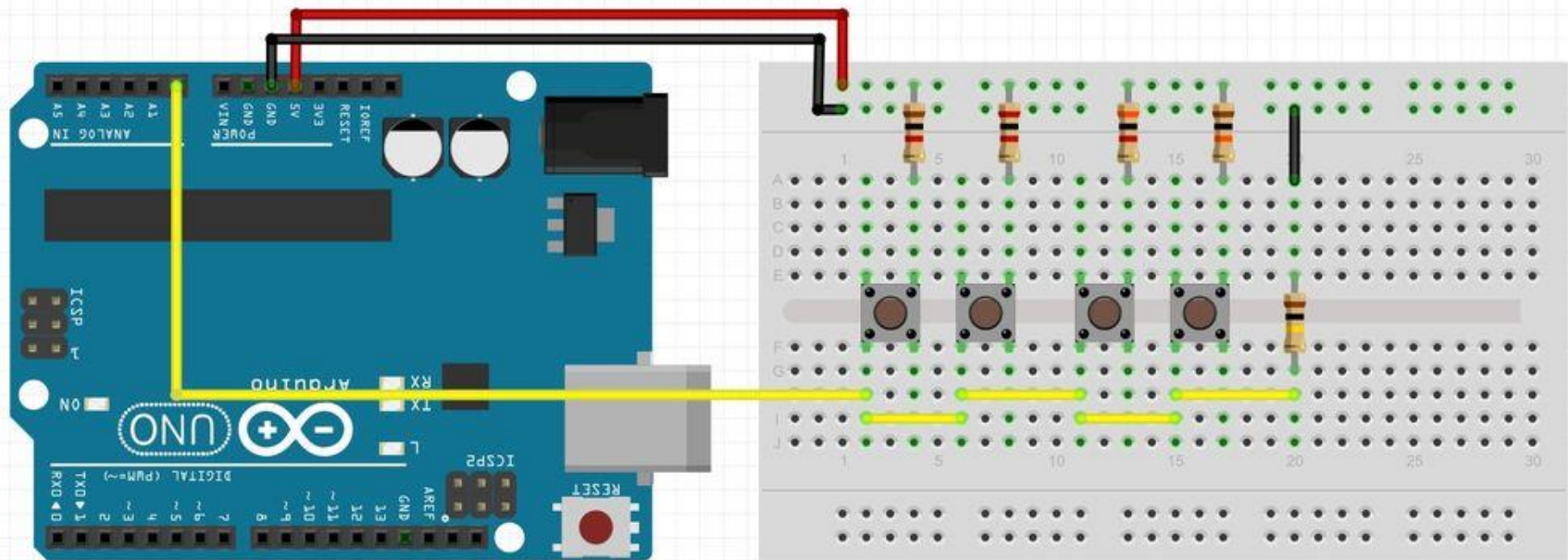
## LDR module

- เป็นโมดูลสำหรับใช้วัดความสว่างของแสง
- LDR ย่อมาจาก Light Detector Resister
- LDR จะเปลี่ยนค่าความต้านทานไปตามความสว่างของแสง
- โมดูลจะมี 4 ขา คือ
  - Vcc ต่อกับ 5V เพื่อเลี้ยงวงจร, Gnd ต่อกับ Ground ของ Arduino
  - AO (Analog Out) จะให้ Output เป็น Analog (0-1023)
  - DO (Digital Out) จะให้ Output เป็น **HIGH** เมื่อความสว่างมากกว่าที่กำหนด (สามารถกำหนดโดย R ปรับค่าได้ (สไล์ฟ้า))



# การประยุกต์ใช้ Analog Read ในการอ่าน Switch





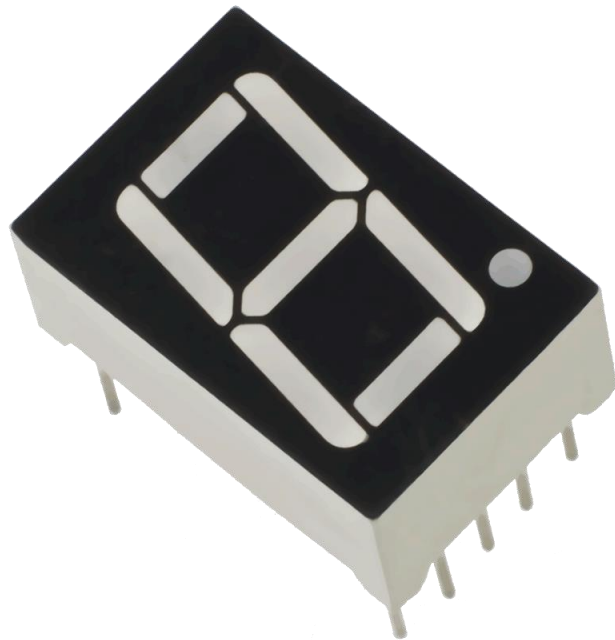
fritzing





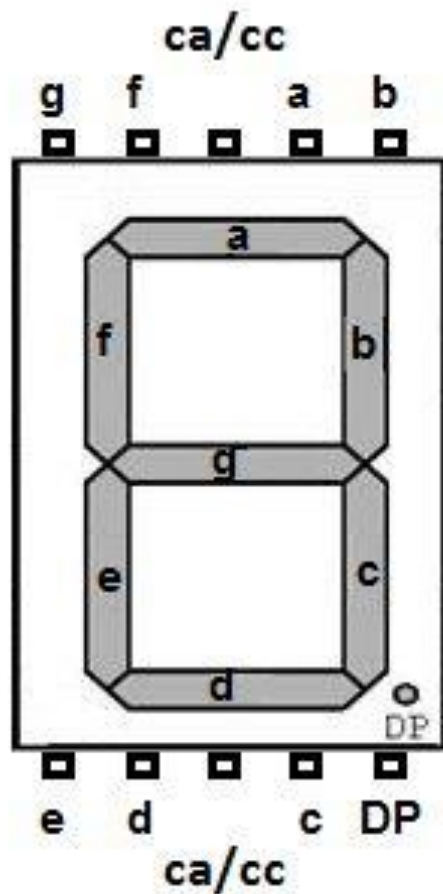
## Activity

- ให้นำ LDR ต่อกับ Arduino
- ให้ต่อขาทั้ง Digital และ Analog
- เขียนโปรแกรมแสดงค่าที่อ่านได้ ทั้ง Analog และ Digital
- เมื่อปรับ Trimpot (สีฟ้า) แล้วเกิดอะไรขึ้น

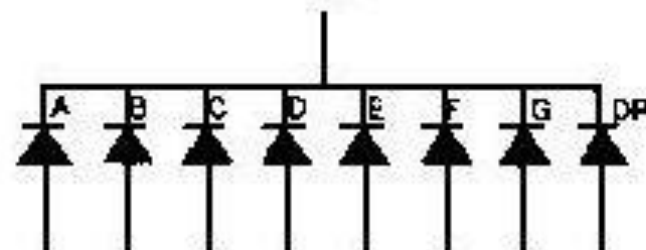


7-Segments

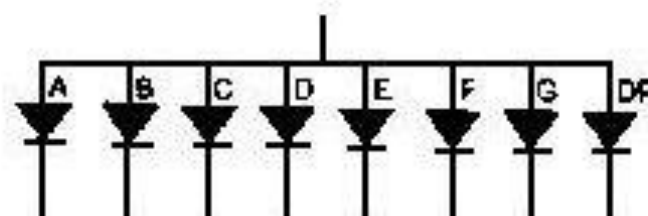
# รายละเอียดแต่ละ Segment ใน 7 Segment



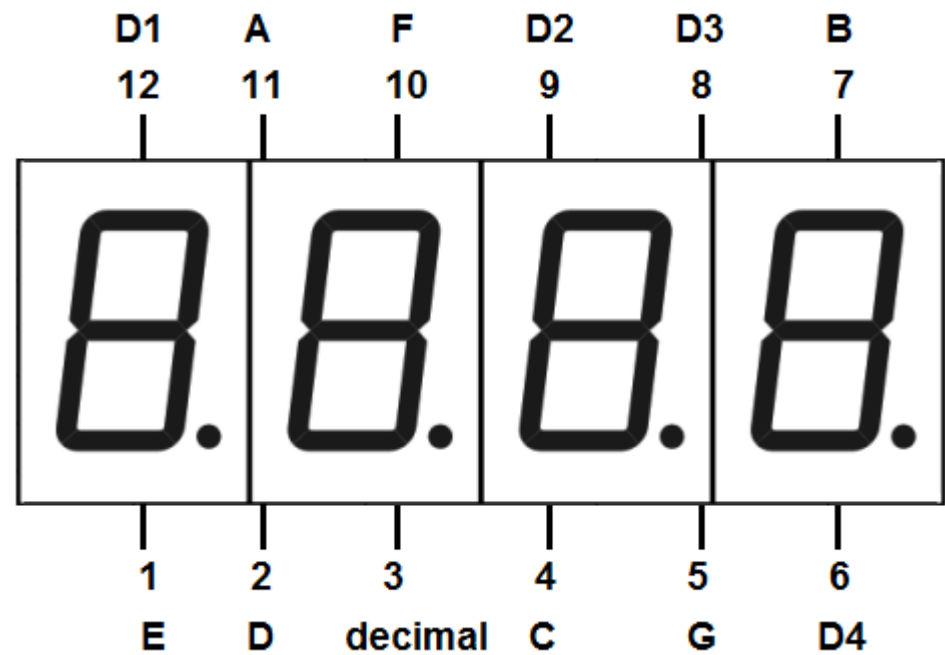
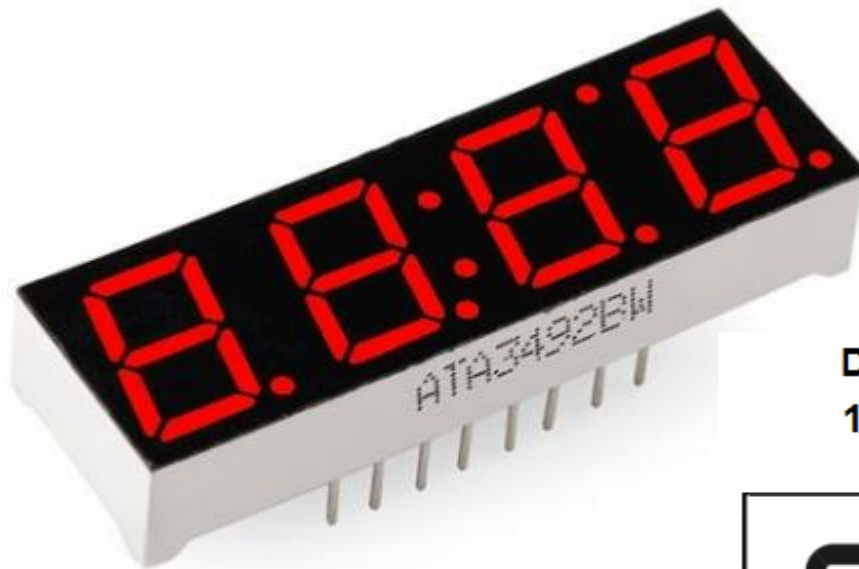
Common Cathode

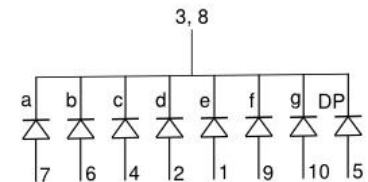
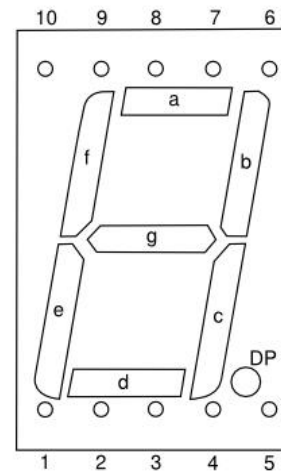
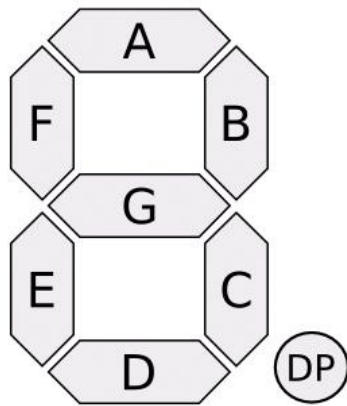
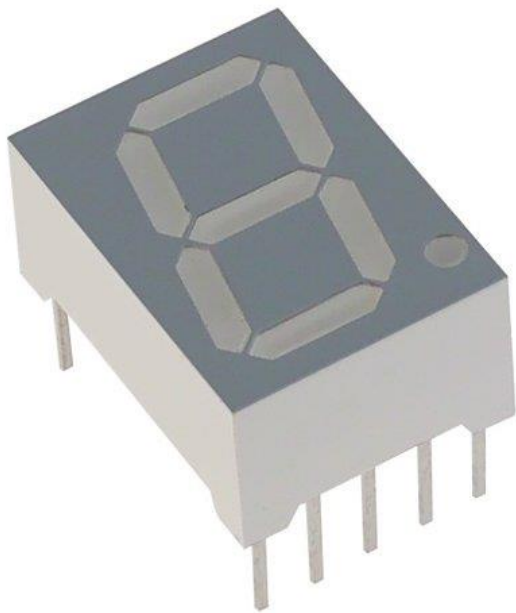


Common Anode



# LED 7 Segment ชนิดหลายหลัก



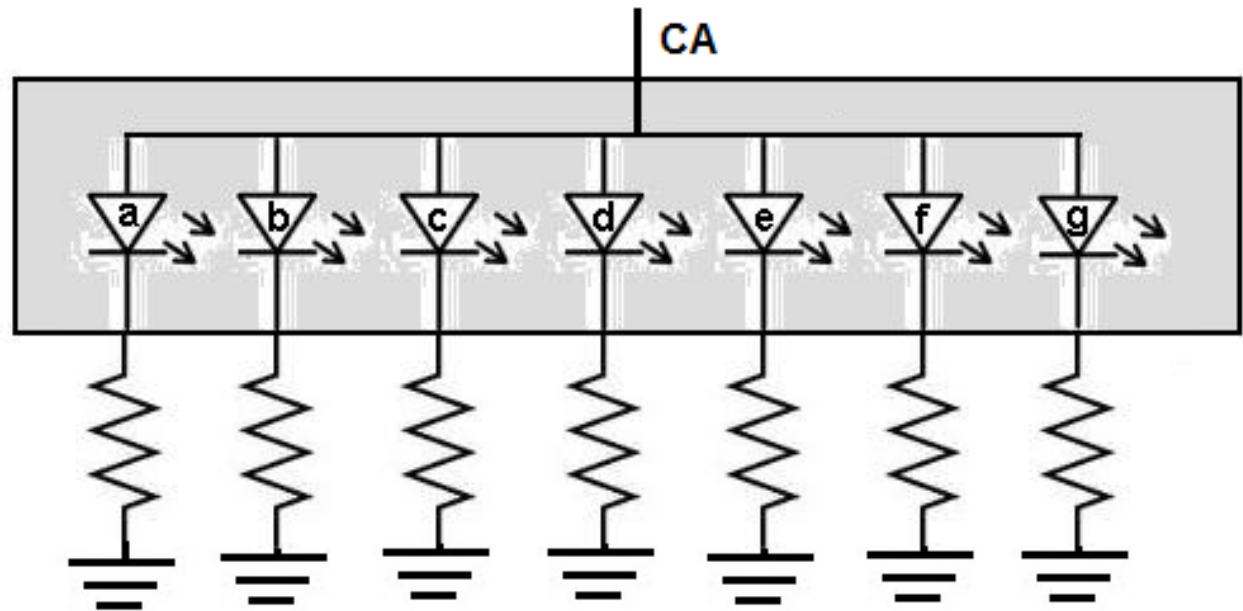


อยากให้ 7-Segments แสดงเลข 3 ค่า a-g จะต้องมียค่าเป็น  
เท่าใดตามลำดับ

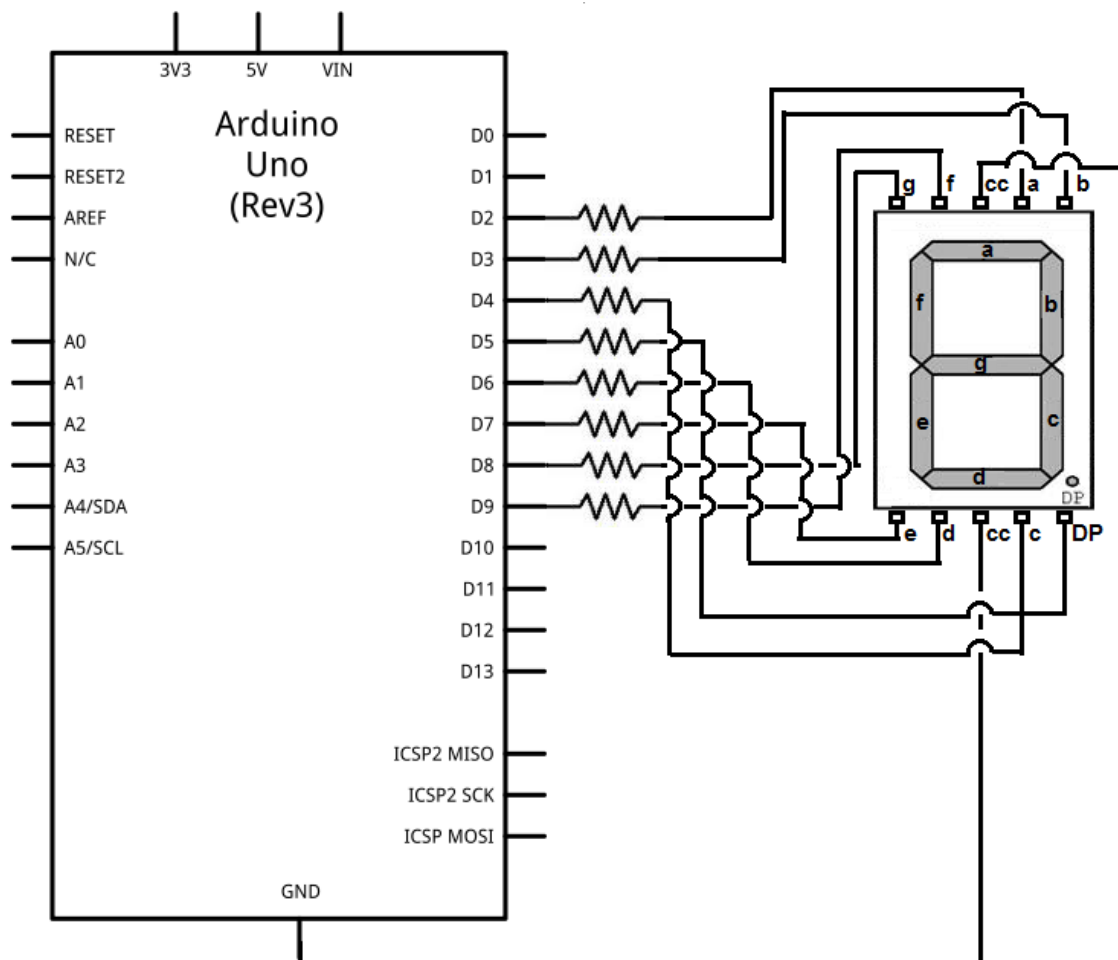


## การคำนวณตัวต้านทานสำหรับ 7-Segments

- Input voltage จากวงจร 5 Volts
- 7-Segment รับกระแส 15mA และ forward voltage drop ที่ 2 Volts
- ต้องใช้ R เท่าไหร่ - สีอะไร



# ตัวอย่างการต่อ 7 Segment กับ Arduino





# Operator ระดับบิตในภาษา C/C++

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

NOT Truth Table

A	B
0	1
1	0





## Operator ระดับบิตในภาษา C/C++

- ในการทำงานระดับ Hardware การประมวลผลระดับบิตมีความสำคัญ เช่นใน 7 segment แทนที่เราจะใช้ตัวแปร 8 ตัว หรือ array ขนาด 8 ไบต์ ในการแทนแต่ละ segment เราสามารถใช้แต่ละบิตแทนได้ ซึ่งทำให้เราใช้ข้อมูลเพียงไบต์เดียว
- $a \mid b$  (Bitwise Or) เป็นการ or ระดับบิต เช่น ตัวอย่าง  $0x56 \mid 0x32$  ได้  $0x76$

```
  0 1 0 1 0 1 1 0 <----- 0x56
| 0 0 1 1 0 0 1 0 <----- 0x32
-----
  0 1 1 1 0 1 1 0 <----- 0x76
```



## Operator ระดับบิตในภาษา C/C++

- $a \& b$  (Bitwise And) เป็นการ and ในระดับบิต ตัวอย่าง  $0x56 \& 0x32$  ได้  $0x12$

	0	1	0	1	0	1	1	0	<-----	0x56
&	0	0	1	1	0	0	1	0	<-----	0x32
-----										
	0	0	0	1	0	0	1	0	<-----	0x12

- $a \wedge b$  (Bitwise Exclusive or) เป็นการทำ exclusive or ในระดับบิต "เหมือนกันเป็นศูนย์ ต่างกันเป็นหนึ่ง" ตัวอย่าง  $0x56 \wedge 0x32$  ได้ผลลัพธ์  $0x64$

	0	1	0	1	0	1	1	0	<-----	0x56
^	0	0	1	1	0	0	1	0	<-----	0x32
-----										
	0	1	1	0	0	1	0	0	<-----	0x64



## Operator ระดับบิตในภาษา C/C++

- $\sim a$  (Bitwise Complement) ทำหน้าที่ กลับบิต ให้ตรงกันข้าม จากหนึ่งเป็นศูนย์ หรือ จากศูนย์เป็นหนึ่ง

```
~ 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0
-----
  1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1
```



## Operator ระดับบิตในภาษา C/C++

- << (Shift Bit Left) ทำให้ลำดับบิตเลื่อนไปทางซ้าย แล้วนำศูนย์มาต่อบิตทางขวา

```
      0 0 0 0 0 1 1 0    <---- 0x06
0 0 0 0 0 1 1 0 _ _    <---- 0x06 << 2
-----
      0 0 0 1 1 0 0 0    <---- 0x18
```

- >> (Shift Bit Right) ทำให้ลำดับบิตเลื่อนไปทางขวา แล้วนำศูนย์มาต่อบิตทางซ้ายมือ (ส่วนที่ Shift Out ตัดทิ้ง)

```
      0 0 0 0 0 1 1 0    <---- 0x06
_ _ 0 0 0 0 0 1 1 0    <---- 0x06 >> 2
-----
      0 0 0 0 0 0 0 1    <---- 0x01
```



# Operator ระดับบิตในภาษา C/C++

- การเซตบิต (Set Bit)

```
uint8_t a = 0x08;    /* 00001000 */
                    /* เซต บิตที่2 */
a |= (1<<2);         /* 00001100 */
```

- การเคลียร์บิต (Clear Bit)

```
uint8_t a = 0x0F;    /* 00001111 */
                    /* เคลียร์บิตที่2 */
a &= ~(1<<2);        /* 00001011 */
```



# Operator ระดับบิตในภาษา C/C++

- การเคลียร์บิต (Clear Bit) หลายบิต

```
uint8_t a = 0x0F;          /* 00001111 */
                             /* เคลียร์บิตที่1 และบิตที่2 */
a &= ~( (1<<2) | (1<<1) ); /* 00001001 */
```

- การกลับบิต (Toggle Bit)

```
uint8_t a = 0x0F; /* 00001111 */
                             /* สลับบิตที่2 */
a ^= (1<<2);        /* 00001011 */
a ^= (1<<2);        /* 00001111 */
```



# Operator ระดับบิตในภาษา C/C++

- `bitRead()` อ่านบิตจากข้อมูล

## Syntax

**`bitRead(x, n)`**

- `bitRead(5,2)` จะ return ค่า บิตที่ 2 นับจากขวาสุดของ 00000101 นั่นก็คือ 1
- เทียบเท่ากับ `((5) >> (2)) & 0x01`
- ใน arduino หากต้องการกำหนดข้อมูลเป็นฐาน 2 ให้ใช้ B00000101



# Operator ระดับบิตในภาษา C/C++

```
int number;    // number to display
int bitPattern = 'B11111100'
const byte numPins = 8;
const int segmentPins[8] = {7, 12, 11, 5, 6, 8, 9, 10};
void setup () {
    Serial.begin(19200);
    for (int i = 0; i < numPins; i++)
        pinMode(segmentPins[i], OUTPUT);
}

void loop() {
    boolean isBitSet;

    for (int segment = 0; segment < 8; segment++){
        isBitSet = bitRead(bitPattern, segment);
        digitalWrite(segmentPins[segment], isBitSet);
    }
}
```



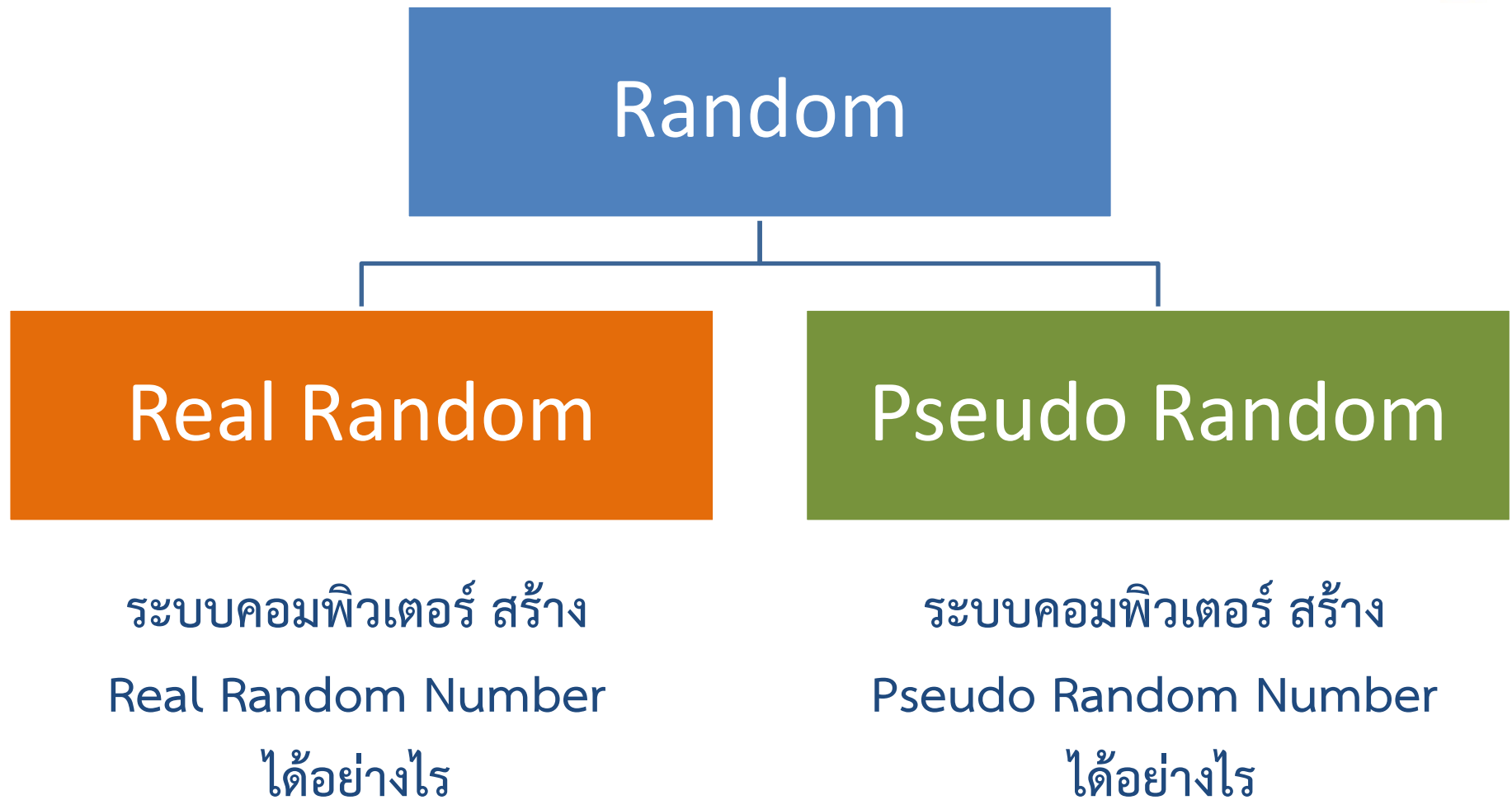
# Activity



- ให้เขียนฟังก์ชันที่แสดงเลข 0-9 ใน 7 segment โดยรับพารามิเตอร์เป็นเลข 0-9



ตัวเลขสุ่ม (Random Number) มีลักษณะอย่างไร





# Random

## Real Random

วัดค่าจากแหล่งภายนอก  
เช่น Key Stroke , Voltage ที่มีการ  
สุ่มค่าจริงๆ

## Pseudo Random

Pseudo Random Number  
Generator Algorithm โดยใช้  
สูตรคณิตศาสตร์ มาสร้างตาราง  
ที่มีชุดตัวเลขที่เดาค่าได้ยาก



# randomseed()

- Initial random number generator
- Start point of random sequences
- **Parameter**
  - **Long int** : parameter to generate the seed



# random()

- Generate pseudo-random number
- Syntax
  - random(max)
  - random(min,max)
- Return
  - A random number between min and max-1 (long)






# random() และ randomseed()

- ให้ใช้ randomseed() ใน setup ()

```
void setup () {  
    randomSeed(analogRead(A0));  
    randomNo = random(1, 10);  
}
```



# Assignment #3 : Number Gues

- ต่อวงจรโดยใช้สวิตช์ 2 ตัว A และ B และต่อ 7 Segment จำนวน 1 ตัว
- เริ่มต้นการทำงานให้สุ่มเลข 1-9
- จากนั้นเมื่อกดสวิตช์ A ให้แสดงผลใน 7 Segment เริ่มจาก 1 และเพิ่มครั้งละ 1 ถ้าเกิน 9 ให้กลับมาเริ่มที่ 1 ใหม่
- จากนั้นกดสวิตช์ B ซึ่งเป็นการทาย ให้เปรียบเทียบกับค่าที่สุ่มเอาไว้ตั้งแต่แรก
- ถ้าตรงกันให้แสดง  ถ้ามากกว่าให้แสดง  (G) ถ้าน้อยกว่าให้แสดง 
- ทายได้เรื่อยๆ ถ้าทายถูกให้สุ่มเลขใหม่
- ให้กำหนดตัวเลขใน Array และการแสดงผลให้ทำเป็นฟังก์ชันเดียว ห้ามทำเป็นฟังก์ชัน แสดงเลข 0,1,2,3,4,5,6,7,8,9 แยกกันไป ห้ามทำเป็น if หรือ case
- คะแนน 5 คะแนน





# Guess Number Game

```
void setup() {  
    //setup pin mode and randomseed  
}  
  
loop() {  
    handle_guess_button();  
    handle_start_button();  
}  
  
void handle_guess_button() {  
    //increment guess number in pressed.  
}  
  
void handle_start_button() {  
    // if start pressed :  
}
```



*For your attention*