

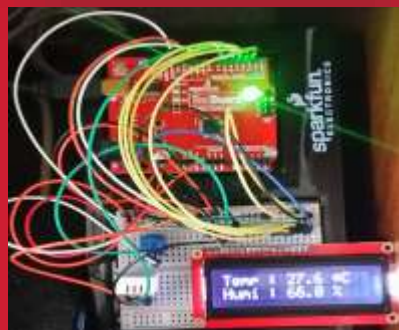


# Arduino-basic

## [wk09]



# Digital input



Learn how to code Arduino from scratch

Comsi, INJE University

2<sup>nd</sup> semester, 2018

Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)

예제제로 쉬게 배우는

# 아두이노

김진환 · 장성웅 지음

# My ID (ARnn)

성명	ID
백동진	AR01
김도훈	AR02
김희찬	AR03
류재현	AR04
문민규	AR05
박진석	AR06
이승현	AR07
이승협	AR08
이후정	AR09
최민구	AR10

김다영	AR11
공진영	AR12
김해인	AR13
류성현	AR14
류재환	AR15
박상현	AR16
박해주	AR17
백지혜	AR18
송원식	AR19
신송주	AR20
윤지훈	AR21
정은성	AR22



# [Practice]

## ◆ [wk07]

- **Arduino LED – III : FND & DM**
- **Complete your project**
- **Submit file : Arnn\_Rpt05.zip**

# wk07 : Practice-05 : ARnn\_Rpt05.zip

## ◆ [Target of this week]

- Complete your works
- Save your outcomes and compress all.

제출파일명 : **ARnn\_Rpt05.zip**

- 압축할 파일들

① **ARnn\_4digit.fzz**

② **ARnn\_4digit.ino**

③ **ARnn\_4digit.png**

④ **ARnn\_dm\_module.fzz**

⑤ **ARnn\_heart.ino**

⑥ **ARnn\_heart.png**

**Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)**

**[ 제목 : id, 이름 (수정) ]**



# 4. LED III

## FND & D.M

4 1EA



7세그먼트 1채널

---

공통 음극 7세그먼트  
시계나 점수 등의 숫자를  
표현 할 때 많이 사용됩니다.

5 1EA



74HC595N

---

기본 메인보드입니다.  
74HC595N LED,  
도트매트릭스, NFD 제어 IC 입니다.

3 1EA



7세그먼트 4채널

---

7세그먼트가 4개 연결된 형태의  
부품입니다.  
총 12개의 핀을 사용합니다.

23 1EA



8x8 도트매트릭스 모듈

---

LED로 다양한 연출을  
할 수 있습니다.

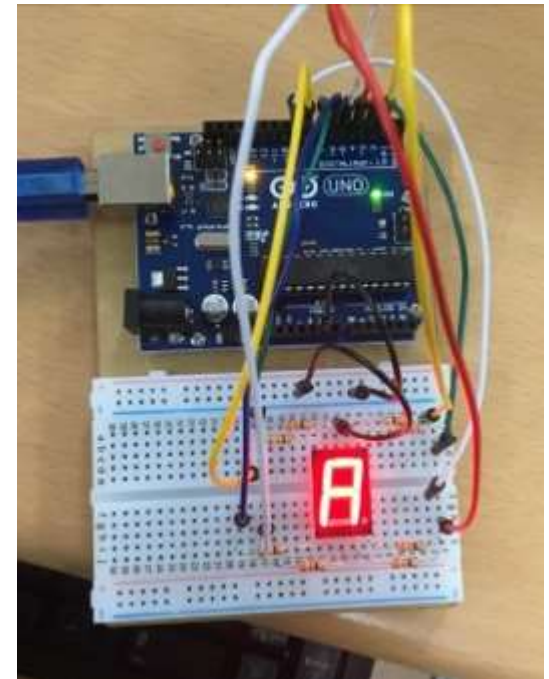
## EX 4.4 FND 제어 (3/3)

**DIY** 위의 예제를 0~F까지의 16진수를 표시하도록 스케치를 수정하여 보자.

(hint: LED 표시를 위한 상수에 A~F를 추가시켜서 불러와 사용하자)

**‘A’가 출력된 화면을 ARnn\_A.png**

로 저장해서 제출. (아두이노 회로를 포함해서 촬영)



## 4.6.5 FND 제어 : 74595 - DIY

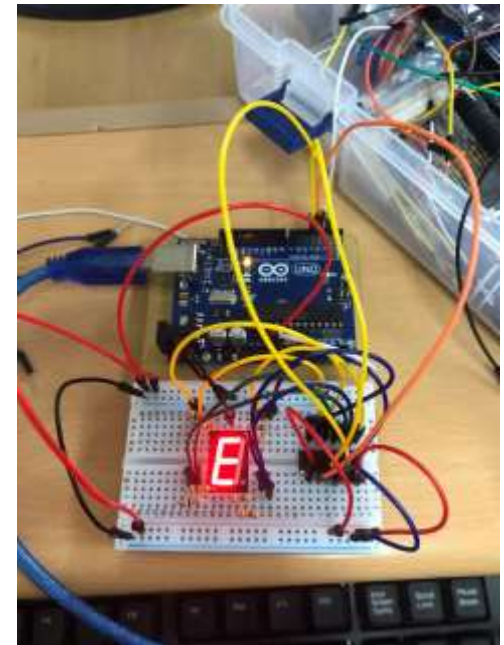
### EX 4.4.2 74595를 이용한 FND 제어 (3/3)

**DIY** 위의 예제를 0~F까지의 16진수를 표시하도록 스케치를 수정하여 보자.

(hint: LED 표시를 위한 상수에 A~F를 추가시켜서 불러와 사용하자)

**‘E’가 출력된 화면을 ARnn\_E.png**

로 저장해서 제출. (아두이노 회로를 포함해서 촬영)

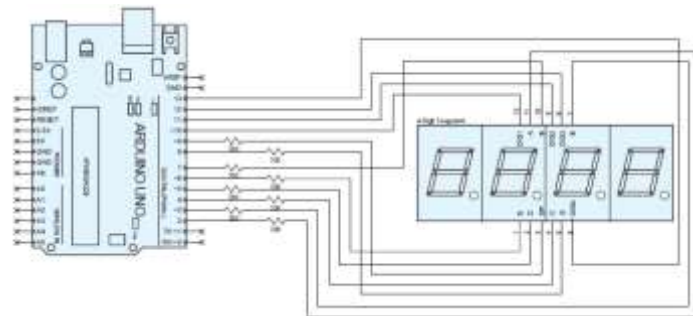


# 4.7.1 4-digit FND 제어

## EX 4.5.1 4-digit FND로 0000~9999 숫자 표시하기 (1/3)

**실습목표** Common Cathode 4-digit FND를 이용하여 0000~9999까지 1초 간격으로 증가하는 스케치를 작성해 보자.

- Hardware**
1. 4-digit FND는 4개의 FND를 연결한 부품이다.
  2. 각각의 FND에는 DIG1~DIG4 네 개의 핀이 각각의 FND의 Common Cathode로 연결되어 있다.
  3. A~G, DP핀은 하나의 FND를 동작시킬 때와 같이 330Ω저항을 통하여 Arduino 2~9번핀에 연결한다.
  4. 맨 왼쪽 FND를 동작시키려면 DIG1에만 LOW신호를 준 상태에서 A~G, DP 핀에 원하는 숫자를 쓰기 위한 신호를 주어야 한다.
  5. 두번째 FND를 동작시키려면 DIG2에만 LOW신호를 준 상태에서 A~G, DP 핀에 원하는 숫자를 쓰기 위한 신호를 주어야 한다.
  6. DIG1~DIG4에 모두 LOW신호를 주면 모두 같은 숫자가 표시된다.

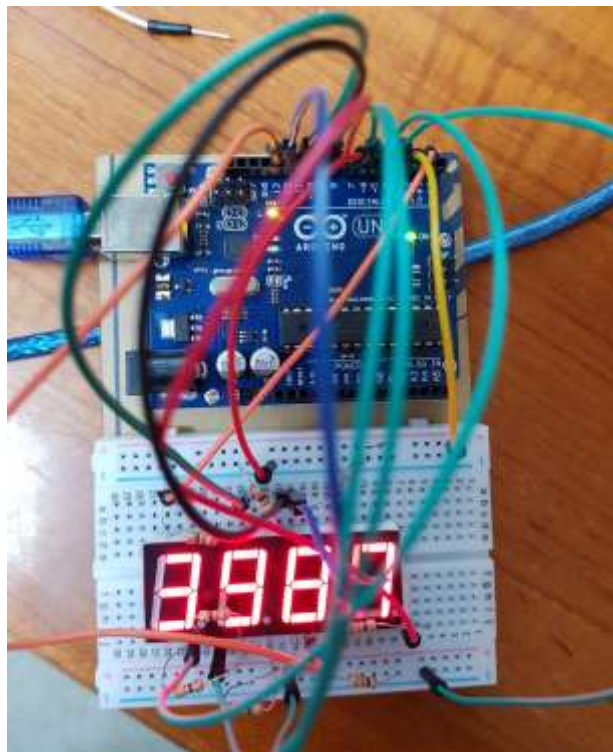




## 4.8.5 4-digit FND 제어 응용 - DIY

DIY      숫자가 증가하는 간격을 0.5초로 변경하여라.

4개의 숫자가 다르게 출력된 화면을 [ARnn\\_4digit.png](#)  
로 저장해서 제출. (아두이노 회로를 포함해서 촬영)

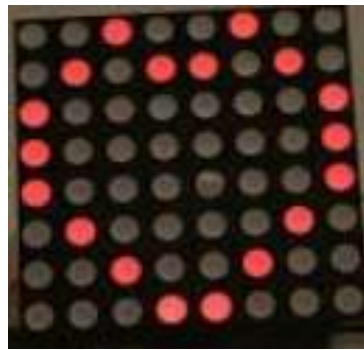


DIY

dot matrix에 ♡ 를 그리는 코드와

♡ 사진을 제출하시오.

완성된 코드를 `ARnn_heart.ino` & `ARnn_heart.png`  
로 저장해서 제출. (실기 시험 2.)





# 5. Digital input

14

6EA



택트스위치 (12x12x7)

스위치를 누르고 있을 경우만  
ON됩니다.

15

각3EA



택트스위치 캡  
(파랑,노랑,초록,빨강,하양)

택트스위치를 사용할 때  
스위치간의 구분을 할 수 있습니다.

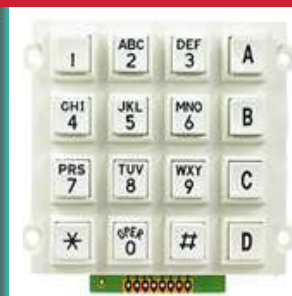
24

1EA



4x4 16 키패드 모듈

16개의 버튼을  
사용할 수 있습니다.



## 디지털 신호

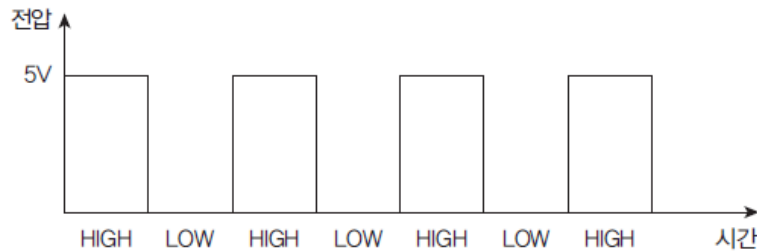


그림 5.1 HIGH LOW 신호가 반복되는 디지털신호

## 풀업 / 풀다운

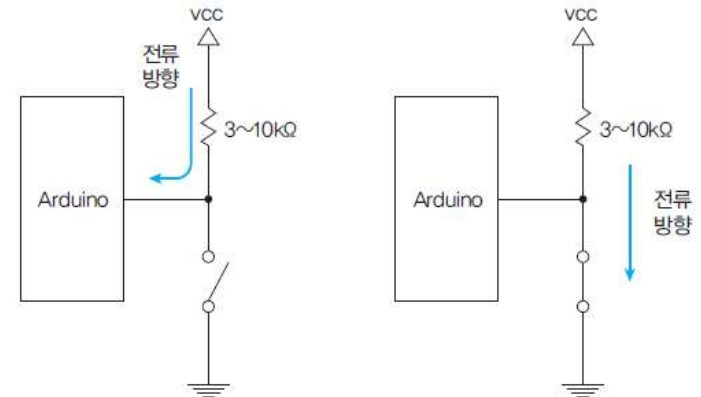


그림 5.2 풀업저항과 스위치 동작. Arduino는 스위치가 오프상태인 (a)에서는 HIGH 신호로, 스위치가 온상태인 (b)에서는 LOW 신호로 인식한다.

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

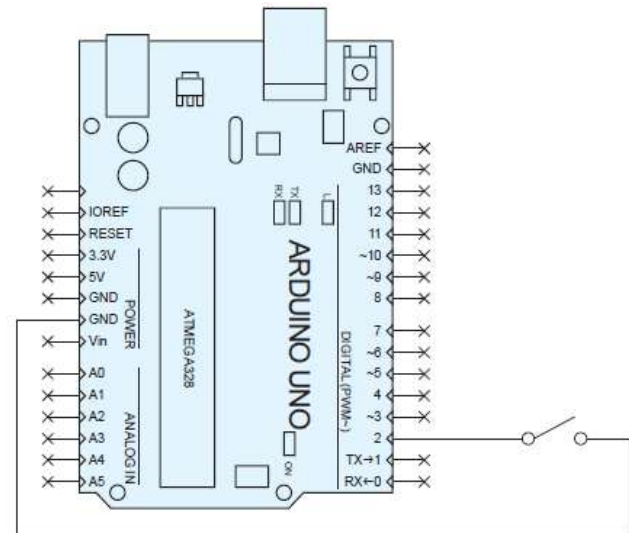
- ✓ 디지털 신호 입력핀에 아무것도 연결되지 않았을 때의 상태를 High 혹은 Low 신호로 만들어 안정시킴

# 5.1.1 디지털 신호 입력 - Switch

## EX 5.1 스위치 입력 (1/3)

**실습목표** 스위치를 이용하여 디지털 신호를 입력 받아 스위치가 눌러졌을 때 LED를 점등시키자.

- Hardware**
1. 디지털 입출력핀인 2번핀으로 스위치 입력을 받는다.
  2. 스위치의 한쪽을 2번핀에 연결하고 다른 한쪽을 GND에 연결한다.
  3. 스위치 입력이 없을 때 2번핀의 상태는 HIGH신호인지 LOW신호인지 알 수가 없다. 그러므로 반드시 핀의 입출력 설정 때 'INPUT\_PULLUP' 명령어를 사용하여 풀업시켜줘야 한다.
  4. 풀업을 해줬다면 스위치 입력이 없을 때 2번핀의 상태는 HIGH신호이고 스위치 입력이 있을 때 2번핀의 상태는 LOW신호이다.



## EX 5.1

## 스위치 입력 (2/3)

### Commands

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. '핀번호' 에는 설정하고자 하는 핀의 번호와 '설정'에는 입력으로 사용하기 위해선 'INPUT', 출력으로 사용하기 위해선 'OUTPUT', **입력이며 풀업 사용시 'INPUT\_PULLUP'을 설정한다.**

- digitalRead(핀번호)

'핀번호' 에 해당하는 핀의 디지털 입력값을 읽는다. LOW 이면 0, HIGH이면 1의 값을 갖는다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. '핀번호' 에는 출력하고자 하는 핀의 번호를, '값'에는 'HIGH' 혹은 'LOW' 를 설정하여 High 혹은 Low 출력을 한다.

### Sketch 구성

1. 2번 핀을 입력으로 설정하고 LED가 내장된 13번 핀을 출력으로 설정한다.
2. 2번 핀은 INPUT\_PULLUP으로서 평상시에는 HIGH 상태로 유지되다가 스위치 입력이 있을 때 LOW로 변화한다.
3. **2번핀에 LOW 신호가 입력될 때 LED를 점등(HIGH)시킨다.**

ex\_5\_1

```

1 /*
2  예제 5.1
3  스위치 입력
4  */
5
6 // 내장된 LED를 사용하기 위해
7 // LED pin을 13번으로 설정
8 const int ledPin = 13;
9 // 2번 핀을 스위치 입력으로 설정
10 const int inputPin = 2;
11
12 void setup() {
13   // Arduino 13번 핀에 내장된 LED를 출력으로 설정한다.
14   pinMode(ledPin, OUTPUT);
15   // 스위치 입력을 위하여 2번핀을 입력으로 설정하고 풀업시킨다.
16   pinMode(inputPin, INPUT_PULLUP);
17 }
18
19 void loop(){
20   // 스위치 입력을 받는다
21   int swInput = digitalRead(inputPin);
22
23   if (swInput == LOW) digitalWrite(ledPin, HIGH); // LED 점등
24   else digitalWrite(ledPin, LOW); // LED 소등
25 }

```

## EX 5.1 스위치 입력 (3/3)

**실습 결과** 스위치를 누르면 LED가 점등되고, 손을 떼면 LED가 소등된다.

**응용 문제** 스위치 입력이 감지될 때 마다 LED가 점등되어있다면 소등시키고, 소등되어 있다면 점등시키도록 loop() 부분을 아래와 같이 수정해 보자.

```
void loop(){  
    int swInput = digitalRead(inputPin); // 스위치 입력을 받는다  
    int ledOutput = digitalRead(ledPin); // LED의 출력 상태를 확인한다  
  
    if (swInput == LOW){  
        if (ledOutput) digitalWrite(ledPin, LOW); // LED가 점등되어 있으면 소등  
        else digitalWrite(ledPin, HIGH); // LED가 소등되어 있으면 점등  
    }  
}
```

이 때 어떠한 현상이 발생하는가? → 측정 결과를 검토하시오.

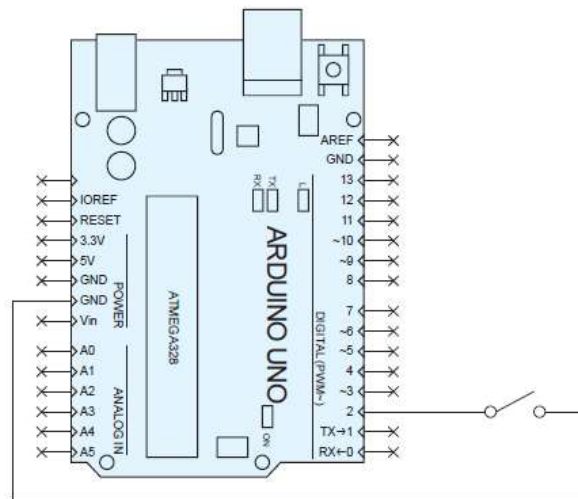


## EX 5.2

## 안정적인 스위치 입력 (1/3)

- 실습목표**
1. 스위치를 이용하여 디지털 신호 입력을 받아 LED를 점멸시킨다.
  2. 스위치가 입력된 횟수를 **시리얼 통신으로 전송**해 보자.

- Hardware**
1. 디지털 입출력핀인 2번핀으로 스위치 입력을 받는다.
  2. 스위치의 한쪽을 2번핀에 연결하고 다른 한쪽을 GND에 연결한다.
  3. 스위치 입력이 없을 때 2번핀의 상태는 HIGH 신호인지 LOW신호인지 알 수가 없다. 그러므로 반드시 핀의 입출력 설정 때 '**INPUT\_PULLUP**' 명령어를 사용하여 풀업시켜줘야 한다.
  4. 풀업을 해줬다면 스위치 입력이 없을 때 2번핀의 상태는 HIGH신호이고 스위치 입력이 있을 때 2번핀의 상태는 LOW신호이다.



### EX 5.2

### 안정적인 스위치 입력 (2/3)

- Sketch 구성
1. LED의 핀 번호를 설정한다.
  2. setup()에서는 LED 출력으로 사용할 핀을 출력핀으로 설정한다.
  3. 스위치 누른 횟수를 저장할 변수를 설정한다. (count)
  4. 시리얼모니터로 스위치 누른 상태를 점검한다.

- 실행 결과
1. 스위치를 누르면 LED가 점등되고, 손을 떼면 LED가 소등된다.
  2. 스위치를 누르는 동안에 시리얼 모니터로 카운트 값이 증가하는 것을 볼 수 있다.

실행 시 어떤 문제가 발생하는가? → 오작동을 해결하는 방법을 찾아 보시오.

## 5.2.3 디지털 신호 입력 - 안정된 Switch : code

```

7 // 내장된 LED 사용을 위해 13번핀을 출력으로 설정
8 const int ledPin = 13;
9 // 2번핀을 스위치 입력으로 설정
10 const int inputPin = 2;
11
12 // 스위치 입력 횟수 저장 변수
13 int count = 0;
14
15 void setup() {
16   // 13번 핀에 내장된 LED를 출력으로 설정한다
17   pinMode(ledPin, OUTPUT);
18   // 스위치 입력을 위하여 2번핀을 입력으로 설정하고 풀업시킨다
19   pinMode(inputPin, INPUT_PULLUP);
20   // 시리얼 통신을 설정한다
21   Serial.begin(9600);
22 }

```

```

24 void loop(){
25   // 스위치 입력을 받는다
26   int swInput = digitalRead(inputPin);
27   // LED의 출력 상태를 확인한다
28   int ledOutput = digitalRead(ledPin);
29
30   // 스위치가 눌렸을 때
31   if (swInput == LOW){
32     if (ledOutput) digitalWrite(ledPin, LOW); // LED가 점등되어 있으면 소등
33     else digitalWrite(ledPin, HIGH); // LED가 소등되어 있으면 점등
34     ++count;
35     // 스위치 입력 횟수를 시리얼 통신으로 전송한다.
36     Serial.println(count);
37   };
38 }

```

**ARnn\_Dinput\_count.ino**

## 5.2.4 디지털 신호 입력 - 안정된 Switch

EX 5.2

### 안정적인 스위치 입력 (3/3)

**응용 문제** 스위치를 누르고 있는 동안에는 'if (swInput == LOW){ }' 내의 명령어는 수없이 반복한다. 실제 디지털 입력을 받을 때에는 단 **한번의 명령만 실행되어야** 한다. 이를 위해서 이 예제의 스위치가 눌렀을 때 처리되는 부분을 다음과 같이 수정해 보자.

```
// 스위치가 눌렀을 때
if (swInput == LOW){
    delay(100);
    if (ledOutput) digitalWrite(ledPin, LOW); // LED가 점등되어 있으면 소등
    else digitalWrite(ledPin, HIGH); // LED가 소등되어 있으면 점등
    ++count;
// 스위치 입력 횟수를 시리얼 통신으로 전송한다.
    Serial.println(count);
}
```

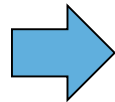
**실행 시 어떤 문제가 발생하는가? → 해결 방법을 찾아 보시오.**

## 5.2.5 디지털 신호 입력 : 안정된 Switch - DIY

[DIY] 현재의 LED 상태 (LOW or HIGH)와 스위치 누른 횟수를 아래와 같이 출력하는 code를 만들어서 **ARnn\_Switch.ino** 로 저장하시오.

```
COM6

LED: 0, count: 1
LED: 1, count: 2
LED: 0, count: 3
LED: 1, count: 4
LED: 0, count: 5
LED: 1, count: 6
LED: 0, count: 7
LED: 1, count: 8
LED: 0, count: 9
LED: 1, count: 10
```



```
COM6

LED: LOW, count: 1
LED: HIGH, count: 2
LED: LOW, count: 3
LED: HIGH, count: 4
LED: LOW, count: 5
LED: HIGH, count: 6
LED: LOW, count: 7
LED: HIGH, count: 8
LED: LOW, count: 9
LED: HIGH, count: 10
```

```
void loop(){
  int swinput = digitalRead(inputPin); // 스위치 입력을 받는다
  int ledOutput = digitalRead(ledPin); // LED의 출력 상태를 확인한다

  if (swinput == LOW){
    delay(400);
    if (ledOutput) {
      insert your code here!
    }
    ++count;
    // 스위치 입력 횟수를 시리얼 통신으로 전송한다.
    Serial.print(", count: ");
    Serial.println(count);
  }
}
```

**ARnn\_Switch.ino** 로 저장  
, 제출

## 5.2.6.1 디지털 신호 입력 : 안정된 Switch - DIY

[DIY] 스위치 입력 마다 LED On/Off 바꾸면서, 바운싱 없는 스위치 입력 받으면서  
스위치 입력 횟수를 시리얼 통신으로 전송하는 code를 만들어서

ARnn\_Switch\_good.ino 로 저장하시오.

ARnn\_Switch\_good.ino 제출

```

8 // 내장된 LED 사용을 위해 13번핀을 출력으로 설정
9 const int ledPin = 13;
10 // 2번핀을 스위치 입력으로 설정
11 const int inputPin = 2;
12 // 실제 스위치가 눌린 후 지연되는 시간
13 const int swCountTime = 10;
14
15 // 스위치 입력 횟수 저장 변수
16 int count = 0;
17 // 실제 스위치가 눌린 시간을 계산하기 위한 변수
18 int swCountTimer = 0;
19
20 void setup() {
21   // 13번 핀에 내장된 LED를 출력으로 설정한다
22   pinMode(ledPin, OUTPUT);
23   // 스위치 입력을 위하여 2번핀을 입력으로 설정하고 풀업시킨다
24   pinMode(inputPin, INPUT_PULLUP);
25   // 시리얼 통신을 설정한다
26   Serial.begin(9600);
27 }

```

```

29 void loop(){
30
31   // LED의 출력 상태를 확인한다
32   int ledOutput = digitalRead(ledPin);
33
34   // swCheck(핀번호) 루틴에서 HIGH, LOW 값을 받는다.
35   if (swCheck(inputPin)){
36     if (ledOutput) digitalWrite(ledPin, LOW); // LED가 점등되어 있으면 소등
37     else digitalWrite(ledPin, HIGH); // LED가 소등되어 있으면 점등
38     ++count;
39     Serial.println(count); // 스위치 입력 횟수를 시리얼 통신으로 전송한다.
40   };
41 }

```

## [DIY] swCheck() function

```
43 boolean swCheck(int pin){
44
45     // 스위치 입력을 받는다
46     boolean swInput = digitalRead(pin);
47     // 스위치 입력을 리턴할 변수
48     boolean state;
```

COM11 (Arduino/Genuino Uno)

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17

☒ 자동 스크롤

```
50 // 실제 스위치가 입력되었을 경우
51 if(swInput == LOW){
52     // swCountTimer 변수가 swCountTime 보다 클 때
53     if(swCountTimer >= swCountTime){
54
55         // 두 값이 같아지면 state에 HIGH를 저장
56         if(swCountTimer == swCountTime) state = HIGH;
57         else state = LOW; // 아닐경우 LOW를 저장
58
59         // 다음번 실행 시 LOW에 고정되도록 swCountTimer를 조정함
60         swCountTimer = swCountTime + 1;
61     }
62     else{
63         // 실제 스위치 입력 시간이 swCountTime보다 작을때
64         // swCountTime을 증가시켜준다.
65         ++swCountTimer;
66     }
67 }
68 else{
69     // 실제 스위치 입력이 없을 때 state에 LOW를 저장한다.
70     state = LOW;
71     // 실제 스위치 입력이 없을 때 swCountTimer를 초기화 한다.
72     swCountTimer = 0;
73 }
74 // 이 루틴이 끝날 때 state 값을 리턴한다.
75 return state;
76 }
```

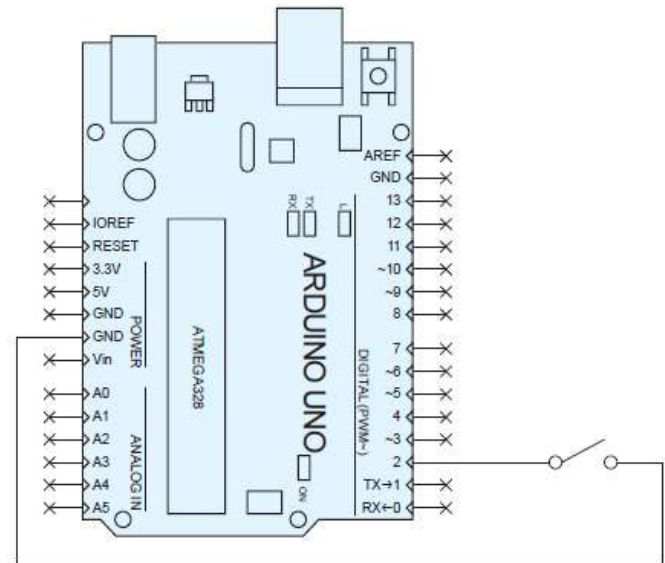
# 5.3.1 디지털 신호 입력 시간 측정하기

## EX 5.3

## 디지털 신호 입력시간 측정하기 (1/3)

- 실습목표**
1. 스위치를 이용하여 디지털 입력을 받는다.
  2. 스위치가 눌러있는 시간을 **0.001초 (ms: 밀리세컨드)단위로 측정**한다.
  3. 측정된 값을 시리얼 통신을 통하여 컴퓨터로 출력시킨다.

- Hardware**
1. 디지털 입출력핀인 2번핀으로 스위치 입력을 받는다.
  2. 스위치의 한쪽을 2번핀에 연결하고 다른 한쪽을 GND에 연결한다.
  3. 스위치 입력이 없을 때 2번핀의 상태는 HIGH신호 인지 LOW신호인지 알 수가 없다. 그러므로 반드시 핀의 입출력 설정 때 'INPUT\_PULLUP' 명령어를 사용하여 풀업시켜줘야 한다.
  4. 풀업을 해줬다면 스위치 입력이 없을 때 2번핀의 상태는 HIGH신호이고 스위치 입력이 있을 때 2번핀의 상태는 LOW신호이다.





## EX 5.3

## 디지털 신호 입력시간 측정하기 (2/3)

### Commands

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. '핀번호' 에는 설정하고자 하는 핀의 번호와 '설정'에는 입력으로 사용하기 위해선 'INPUT', 출력으로 사용하기 위해선 'OUTPUT', 입력이며 풀업 사용시 'INPUT\_PULLUP'을 설정한다.

- digitalWrite(핀번호)

'핀번호' 에 해당하는 핀의 디지털 입력값을 읽는다. LOW 이면 0, HIGH이면 1의 값을 갖는다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. '핀번호' 에는 출력하고자 하는 핀의 번호를, '값'에는 'HIGH' 혹은 'LOW' 를 설정하여 High 혹은 Low 출력을 한다.

- while(조건){ }

조건에 만족할 때까지 괄호 안의 명령을 실행한다.

- millis( )

현재 스케치가 시작된 이후로 경과된 시간 값을 가져온다. 밀리세컨즈(1/1000초) 단위의 값을 갖는다.

## 5.3.3 디지털 신호 입력 시간 측정하기

### EX 5.3

### 디지털 신호 입력시간 측정하기 (3/3)

- Sketch 구성**
1. 2번핀을 디지털 입력핀으로 설정한다.
  2. 현재 시간과 실제 스위치가 눌린 시간을 저장할 변수를 설정한다.
  3. 스위치가 온 되었을 때 Arduino가 실행 한 후의 시간을 확인하는 'millis()'함수를 사용하여 현재 시간을 저장한다.
  4. 스위치가 오프 되었을 때 앞서 저장한 시간과 현재의 시간의 차이를 계산한다.
  5. 측정된 시간을 시리얼 통신으로 출력한다.
- 실습 결과**
1. 스위치를 누르고 뿔 때마다 누르고 있던 시간이 시리얼 모니터에 출력된다.
  2. 시리얼 모니터를 이용하여 확인할 수 있다.

```

6 // 2번핀을 스위치 입력으로 설정
7 const int inputPin = 2;
8
9 // 현재의 시간을 저장하기 위한 변수
10 long startTime = 0;
11 // 실제 스위치가 눌린 후 지연되는 시간
12 long swCountTimer = 0;
13
14 void setup() {
15   // 스위치 입력을 위하여 2번핀을 입력으로 설정하고 풀업시킨다
16   pinMode(inputPin, INPUT_PULLUP);
17   // 시리얼 통신을 설정한다
18   Serial.begin(9600);
19 }
20
21 void loop(){
22   // 스위치 입력이 발생하였을 경우 실행
23   if(digitalRead(inputPin) == LOW){
24     // 현재의 시간을 startTime 변수에 넣는다.
25     startTime = millis();
26     // 스위치가 입력되는 동안 지연시킨다.
27     while(digitalRead(inputPin) == LOW);
28     // swCountTimer 변수에 스위치가 눌러진 시간을 넣는다.
29     // 여기까지 측정된 시간에서 앞서 저장한 시간이 스위치가 눌러진 시간이 된다
30     swCountTimer = millis() - startTime;
31     // 시리얼 통신으로 값을 출력한다.
32     Serial.print(swCountTimer);
33     Serial.println(" ms");
34   };
35 }

```

### [DIY]

3장에서 실습한 16X2 LCD에 스위치 입력시간을 출력해 보자.

(hint: lcd.print 명령어에 'swCountTimer' 변수를 넣어보자.)

→ 윗쪽줄 : ARnn time span

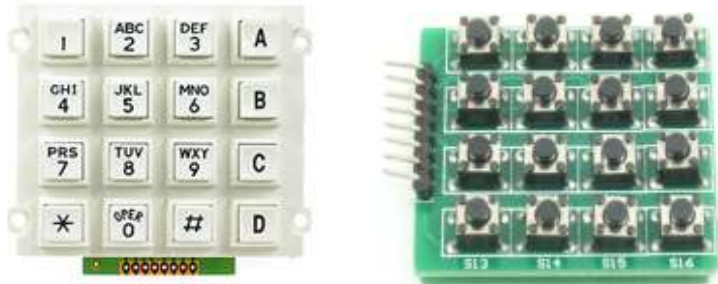
아랫줄: nnnn ms

LCD 출력 화면을 촬영하여 ARnn\_Switch\_time.png 로 저장

아두이노 코드를 ARnn\_Switch\_time.ino 로 저장하여 제출

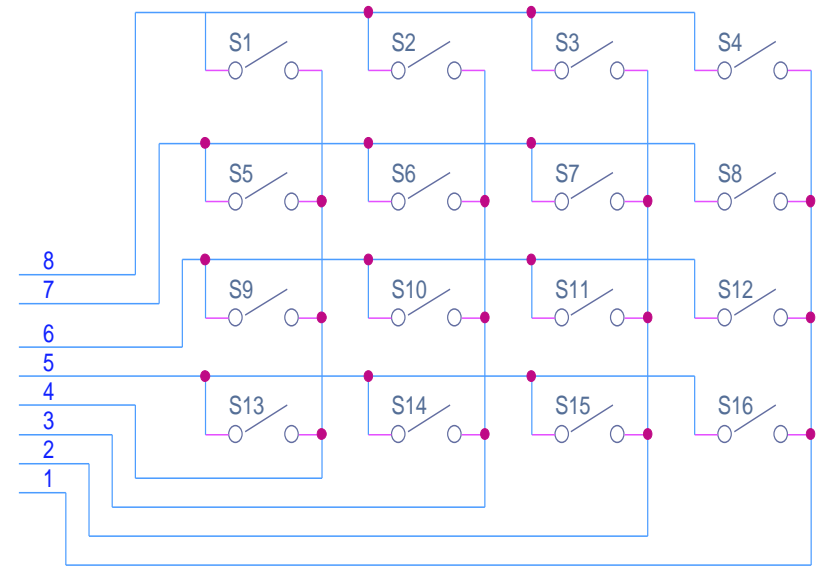
# 5.4.1 키 패드 (Key pad) 입력

## 키 패드 (Key pad)



✓ 여러 개의 스위치를 가로 세로로 연결하여 매트릭스 형태로 조합하여 여러 개의 키 입력을 받을 수 있는 부품

4X4 Key pad



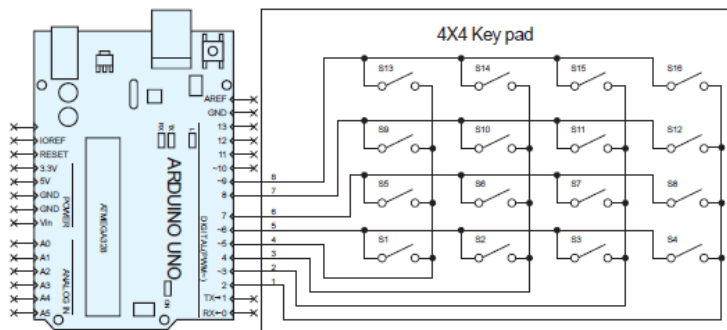
## EX 5.4 키 패드 입력 (1/3)

- 실습목표**
1. 4X4 키 패드로 입력된 값을 시리얼 통신으로 출력한다.
  2. 행은 초기값이 모두 HIGH인 출력으로 설정하고, 열은 초기값은 풀 업이 된 입력으로 설정한다.
  3. LOW로 설정된 행을 하나씩 LOW로 바꾸고 이 때 각 열에 연결된 핀을 확인하여 어느 열에 키 입력이 되었는지 확인한다.
  4. 만일 키 입력이 있다면 해당하는 문자를 시리얼 통신으로 출력하고, 없으면 출력하지 않는다.

- Hardware**
1. 4X4 키패드를 사용하려면 4개의 출력핀과 4개의 입력핀이 필요하다.
  2. 행을 출력으로 사용한다. 행에 해당되는 핀을 6~9번핀에 연결한다.
  3. 열을 입력으로 사용한다. 열에 해당되는 핀을 2~5번핀에 연결한다.
  4. 입력으로 사용할 핀은 반드시 풀업(INPUT\_PULLUP) 설정을 해야 한다.
  5. 네 개의 행 중 첫번째 행에만 LOW신호를 준다. 이 때 각 열에 연결된 네개의 스위치로부터 LOW신호 여부를 확인한다. 즉 6번핀 LOW신호, 7~9번핀 HIGH신호를 준 뒤 S1~S4 스위치의 동작을 확인하는 것이다. 만일 S2에 스위치 입력이 발생했을 경우 Arduino의 4번핀이 LOW신호로 확인될 것이다.
  6. 각 행마다 이러한 동작을 반복하면 어느 위치의 스위치가 동작했는지 확인할 수 있다.

## EX 5.4 키 패드 입력 (2/3)

### Hardware



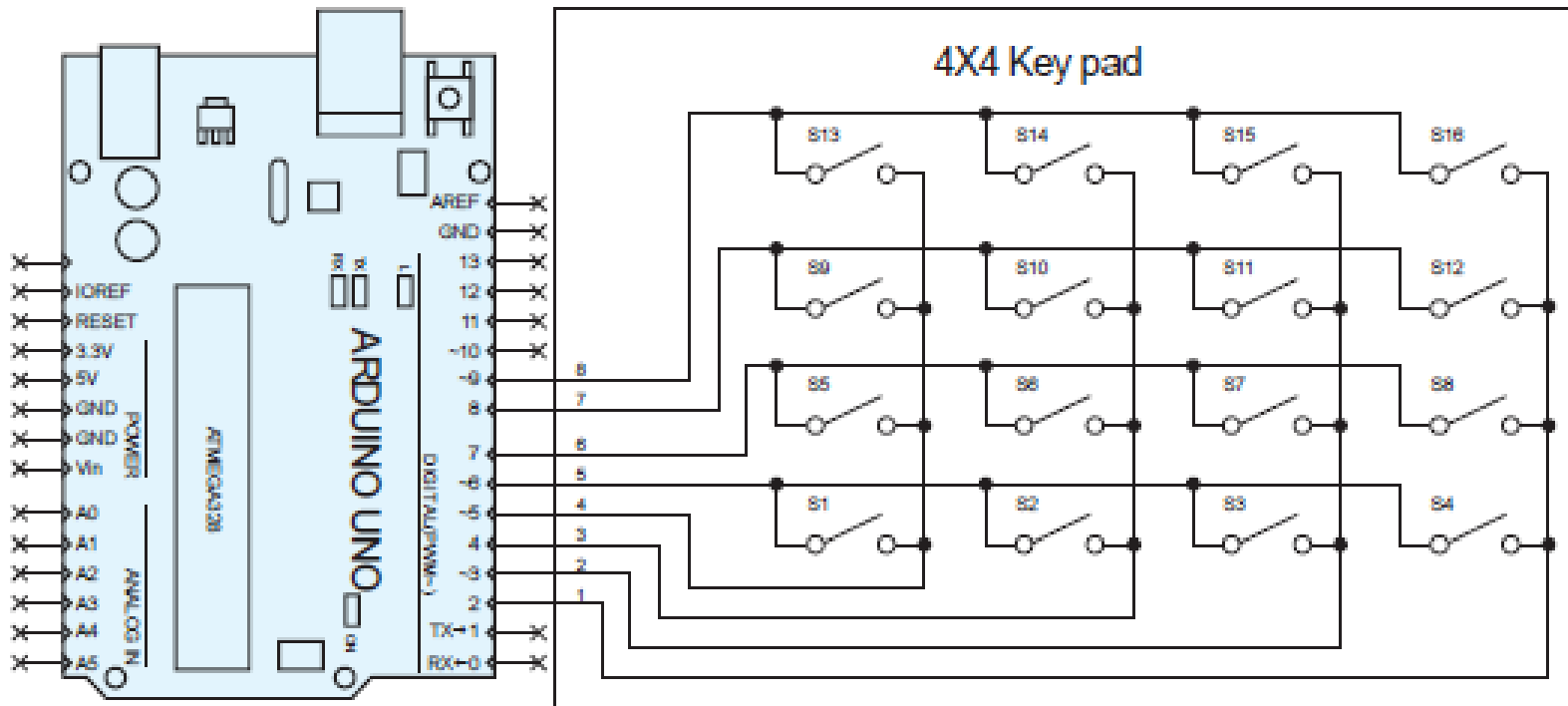
### Commands • pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. '핀번호'에는 설정하고자 하는 핀의 번호와 '설정'에는 입력으로 사용하기 위해선 'INPUT', 출력으로 사용하기 위해선 'OUTPUT', 입력이며 풀업 사용시 'INPUT\_PULLUP'을 설정한다.

### • digitalRead(핀번호)

'핀번호'에 해당하는 핀의 디지털 입력값을 읽는다. LOW 이면 0, HIGH이면 1의 값을 갖는다.

## EX 5.4 키 패드 입력 (2/3)



행에 해당되는 핀을 6~9번핀에 연결

열에 해당되는 핀을 2~5번핀에 연결



# 5.4.3 키 패드 (Key pad) 입력

## EX 5.4 키 패드 입력 (2/3)

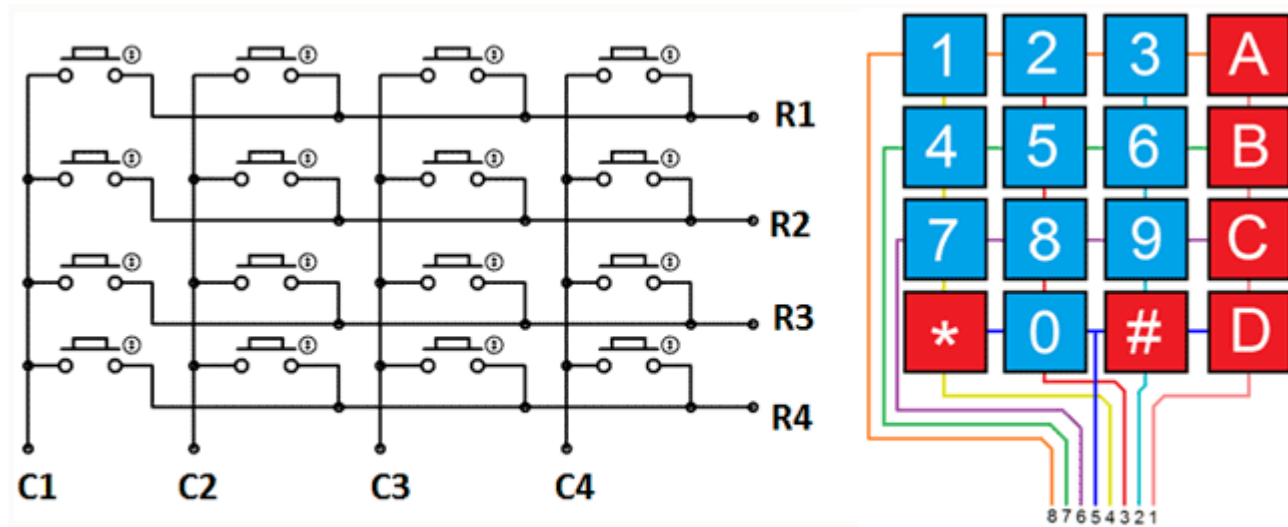
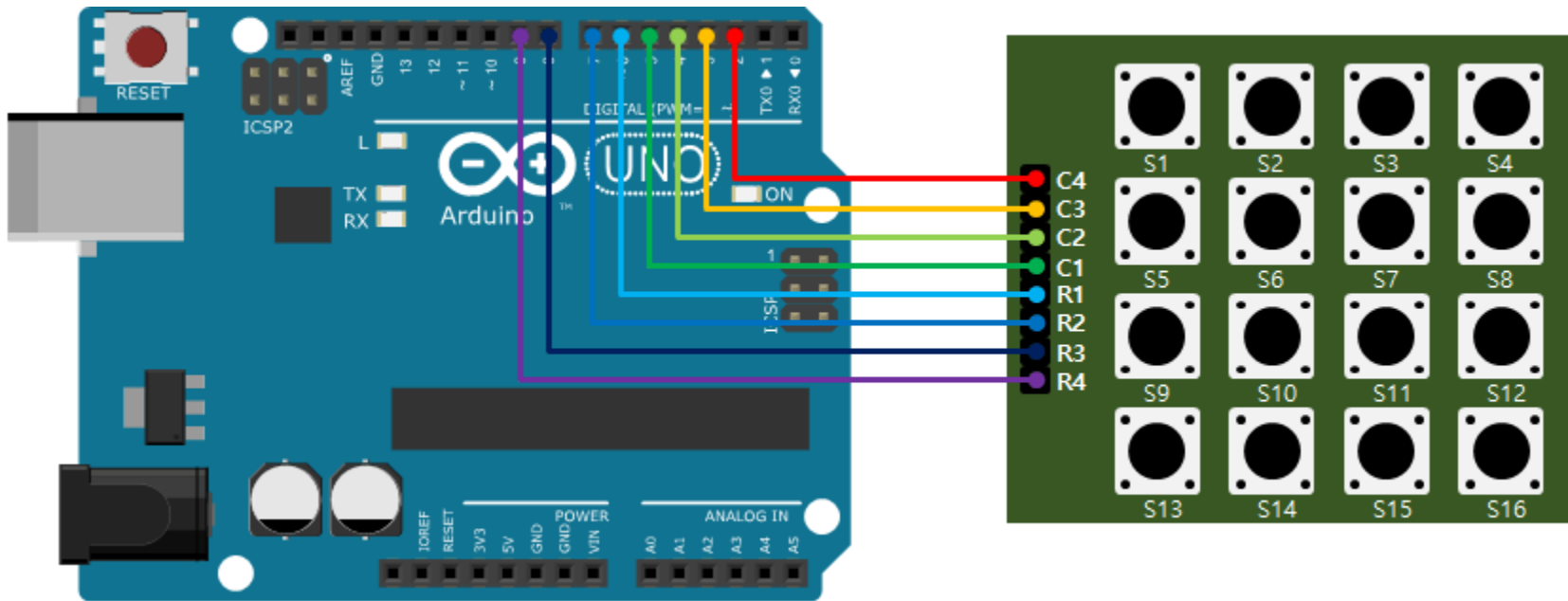


그림 출처: circuitdigest.com

[https://javalab.org/arduino\\_4x4\\_keypad\\_test/](https://javalab.org/arduino_4x4_keypad_test/)

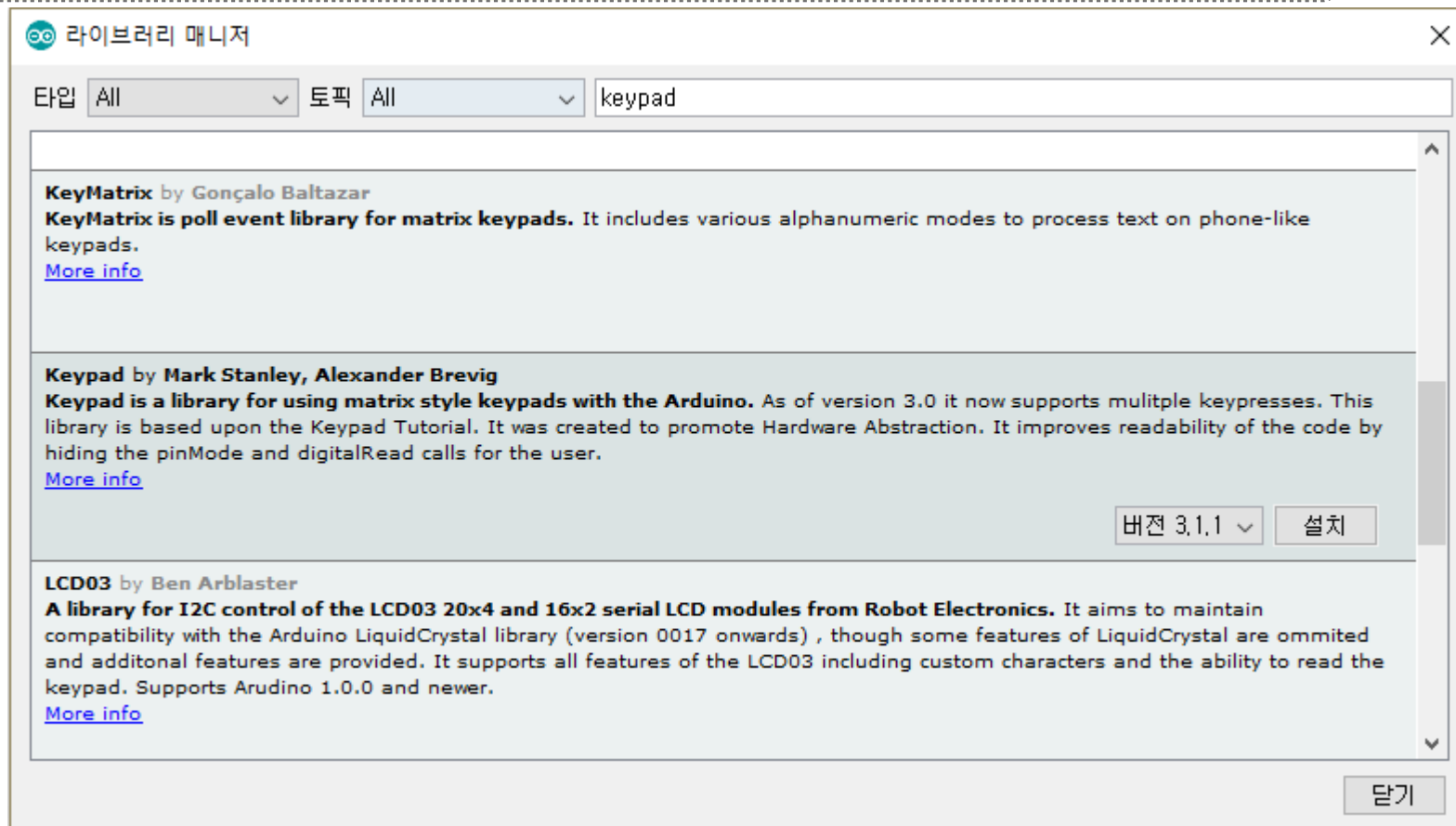
## 5.4.3 키 패드 (Key pad) 입력

### EX 5.4 키 패드 입력 (2/3)



[https://javalab.org/arduino\\_4x4\\_keypad\\_test/](https://javalab.org/arduino_4x4_keypad_test/)

스케치 > 라이브러리 포함하기 > 라이브러리 관리



[https://javalab.org/arduino\\_4x4\\_keypad\\_test/](https://javalab.org/arduino_4x4_keypad_test/)

## 5.4.4 키 패드 (Key pad) 입력 - code 1

```

ARnn_keypad $
1 #include <Keypad.h>
2
3 const byte ROWS = 4;    // 행(rows) 개수
4 const byte COLS = 4;    // 열(columns) 개수
5 char keys[ROWS][COLS] = {
6   {'1','2','3','A'},
7   {'4','5','6','B'},
8   {'7','8','9','C'},
9   {'*','0','#','D'}
10 };
11
12 byte rowPins[ROWS] = {6, 7, 8, 9}; // R1, R2, R3, R4 단자가 연결된 아두이노 핀 번호
13 byte colPins[COLS] = {5, 4, 3, 2}; // C1, C2, C3, C4 단자가 연결된 아두이노 핀 번호
14
15 Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
16
17 void setup() {
18   Serial.begin(9600);
19 }
20
21 void loop() {
22   char key = keypad.getKey();
23
24   if (key) {
25     Serial.println(key);
26   }
27 }

```

```

21 void loop() {
22     char key = keypad.getKey();
23
24     if (key) {
25         switch(key) {
26             case 'A':
27                 Serial.println("Hello?");
28                 break;
29             case 'B':
30                 Serial.println("I love you.");
31                 break;
32             default:
33                 Serial.println(key);
34                 break;
35         }
36     }
37 }

```

## EX 5.4 키 패드 입력 (3/3)

Sketch 구성 1. 각 열과 행에 연결된 키에 대하여 문자를 할당한다.

2. 열에 연결된 핀은 인풋 풀업으로 설정하고 행에 연결된 핀은 출력으로 설정한다.

3. 1행에 LOW신호를 주고 열에 입력이 있는지를 확인한다. 입력이 있을 경우 교차되는 지점의 키가 입력된 것이다. 2행, 3행, 4행으로 반복한다.

4. 시리얼 통신을 이용하여 어느 키가 눌렸는지 표시한다.

실행 결과 1. 키 패드를 누를 때 마다 0~9, \*, #, A~B 의 문자가 시리얼 통신으로 전송된다.

2. 시리얼 모니터를 이용하여 확인할 수 있다.

**Keypad.h**를 이용하지 않고 **key** 값 확인

COM11 (Arduino/Genuino Uno)

You push 1 Key  
 You push 2 Key  
 You push 3 Key  
 You push A Key  
 You push 4 Key  
 You push 5 Key  
 You push 6 Key  
 You push B Key  
 You push 7 Key  
 You push 8 Key  
 You push 9 Key  
 You push C Key  
 You push \* Key  
 You push 0 Key  
 You push # Key  
 You push D Key

ex\_5\_4

```
1 /*
2  예제 5.4
3  4X4 키패드 입력
4  */
5
6  // 열의 수를 설정
7  const int numRows = 4;
8  // 행의 수를 설정
9  const int numCols = 4;
10
11 // 열과 행에 대하여 문자를 할당한다
12 char keys[numRows][numCols]={
13   {'1','2','3','A'},
14   {'4','5','6','B'},
15   {'7','8','9','C'},
16   {'*','0','#','D'}};
17
18 // 열에 연결된 핀번호
19 int rowPins[] = {6, 7, 8, 9};
20 // 행에 연결된 핀번호
21 int colPins[] = {5, 4, 3, 2};
```

```
24 void setup() {
25   // 열을 입력 풀업 모드로 설정한다.
26   for(int i = 0; i < numRows; i++){
27     pinMode(rowPins[i], INPUT_PULLUP);
28   }
29
30   // 행을 출력 모드로 설정한다. 초기값을 HIGH로 설정한다.
31   for(int i = 0; i < numCols; i++){
32     pinMode(colPins[i], OUTPUT);
33     digitalWrite(colPins[i], HIGH);
34   }
35
36   // 시리얼 통신을 설정한다.
37   Serial.begin(9600);
38 }
```

## 5.4.5 키 패드 (Key pad) 입력 - code.2

```

40 void loop(){
41
42 // key 변수에 키패드 입력 값을 읽어서 저장한다
43   char key = keypadRead();
44
45 // key 변수가 0일때는 입력이 없는 것이고
46 // 그외의 값에서는 입력이 발생한 것이다.
47   if(key != 0){
48 // 메시지와 눌린 키를 출력한다
49     Serial.print("You push ");
50     Serial.print(key);
51     Serial.println(" Key");
52   };
53 }

```

```

55 char keypadRead(){
56
57   char key = 0;
58
59   for(int i = 0; i < numCols; i++){
60 // 행 중에 하나를 LOW로 설정한다.
61     digitalWrite(colPins[i], LOW);
62 // 열을 하나씩 바꿔가면서 값을 읽는다.
63     for(int j = 0; j < numRows; j++){
64 // 열의 입력이 LOW 일 때 키 입력이 발생한 것이다.
65       if(digitalRead(rowPins[j]) == LOW){
66         delay(10); // 바운싱 방지를 위해 10ms 대기한다.
67
68 // 키를 놓을 때 까지 기다린다.
69         while(digitalRead(rowPins[j]) == LOW);
70
71 // keys 상수에서 위치에 맞는 값을 가져온다.
72         key = keys[j][i];
73       };
74     }
75 // LOW로 설정했던 행을 다시 HIGH로 설정한다.
76     digitalWrite(colPins[i], HIGH);
77   }
78   return key;
79 }

```



## EX 5.4 키 패드 입력 (3/3)

[DIY] ex\_5\_4\_start.ino 코드를  
완성하고 모든 키 값이 출력된 결과를  
ARnn\_all\_keys.png로 저장, 제출하시오.

COM11 (Arduino/Genuino Uno)

```
You push 1 Key
You push 2 Key
You push 3 Key
You push A Key
You push 4 Key
You push 5 Key
You push 6 Key
You push B Key
You push 7 Key
You push 8 Key
You push 9 Key
You push C Key
You push * Key
You push 0 Key
You push # Key
You push D Key
```

**ARnn\_all\_keys.png**



# [Practice]

## ◆ [wk09]

- **Arduino : Digital input**
- **Complete your project**
- **Submit file : Arnn\_Rpt06.zip**

# wk09 : Practice-06 : ARnn\_Rpt06.zip

## ◆ [Target of this week]

- Complete your works
- Save your outcomes and compress all.

제출파일명 : **ARnn\_Rpt06.zip**

- 압축할 파일들

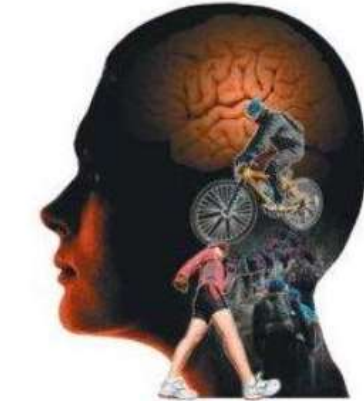
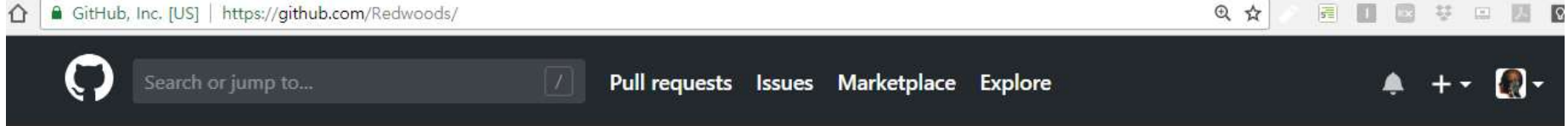
- ① **ARnn\_Switch.ino**
- ② **ARnn\_Switch\_good.ino**
- ③ **ARnn\_Switch\_time.png**
- ④ **ARnn\_Switch\_time.ino**
- ⑤ **ARnn\_all\_keys.png**

**Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)**

**[ 제목 : id, 이름 (수정) ]**

## ● References & good sites

- ✓ <http://www.arduino.cc> Arduino Homepage
- ✓ <http://www.github.com> GitHub
- ✓ <http://www.google.com> Googling
- ✓ <https://www.youtube.com> Youtube



**Redwoods Yi**

Redwoods

Add a bio

GimHae, Republic of Korea

chaos21c@gmail.com

Overview

Repositories 7

Stars 2

Followers 1

Following 0

## Pinned repositories

Customize your pinned repositories

Py

Lectures on coding python from scratch to the advanced level.

Jupyter Notebook

Arduino

Lectures on learning Arduino from scratch to the advanced level in iot environment.

Lec

All lectures by Redwoods in Inje University

Jupyter Notebook

hw-coding

Resource for lecture of Hardware Programming (2017, Inje university)

Arduino

171 contributions in the last year

Contribution settings

Redwoods/Arduino: Lect

GitHub, Inc. [US] | https://github.com/Redwoods/Arduino

Search or jump to... Pull requests Issues Marketplace Explore

Redwoods / Arduino Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Lectures on learning Arduino from scratch to the advanced level in iot environment. Edit

Add topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Redwoods 2018 start	Latest commit 38ca9e0 28 minutes ago
ar-basic	2018 start 28 minutes ago
ar-iot	2018 start 28 minutes ago
README.md	Initial commit 43 minutes ago

README.md

## Arduino

Lectures on learning Arduino from scratch to the advanced level in iot environment.





아두이노 **상급** 키트

High-Level Kit for Arduino

LEVEL 03



당신의 상상을  
실현해줄  
아두이노스토리

[http://arduinostory.com/goods/goods\\_view.php?goodsNo=1000000306](http://arduinostory.com/goods/goods_view.php?goodsNo=1000000306)



## 상급키트 구성품

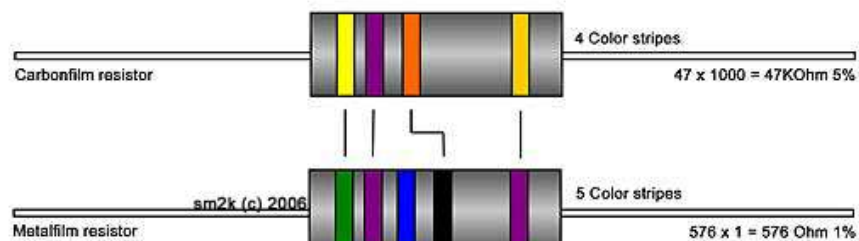
<b>1</b> 1EA  <b>아두이노 우노 R3 DIP</b> 아두이노 우노 R3 (DIP) 호환보드 기본 메인보드입니다.	<b>2</b> 1EA  <b>9V 배터리 홀더</b> 9V 배터리를 연결하여 아두이노에 외부전원을 공급할 수 있습니다.	<b>3</b> 1EA  <b>7세그먼트 4채널</b> 7세그먼트가 4개 연결된 형태의 부품입니다. 총 12개의 핀을 사용합니다.	<b>4</b> 1EA  <b>7세그먼트 1채널</b> 공통 음극 7세그먼트 시계나 점수 등의 숫자를 표현 할 때 많이 사용됩니다.
<b>5</b> 1EA  <b>74HC595N</b> 기본 메인보드입니다. 74HC595N LED, 드로메트릭스, NFD 제어 IC 입니다.	<b>6</b> 1EA  <b>65핀 점퍼 와이어</b> 브레드보드에 연결할 때 사용하는 65핀 점퍼와이어 입니다.	<b>7</b> 1EA  <b>무지개 점퍼선 F-M 20cm</b> M타입과 F타입이 양쪽으로 달린 무지개 점퍼선입니다.	<b>8</b> 1EA  <b>투명 부품 케이스 대,소</b> 키트 구성품을 담을 수 있는 투명 부품 케이스입니다.
<b>9</b> 1EA  <b>가변저항10K</b> 물리변 저항값이 바뀝니다. (0~10KΩ)	<b>10</b> 1EA  <b>1602 I2C LCD</b> 아두이노 16x2 I2C LCD 모듈입니다. LCD입니다.	<b>11</b> 1EA  <b>저항</b> 100, 220, 330, 1K, 2K, 4.7K, 10K, 47K, 100K	<b>12</b> 1EA  <b>브레드 보드 830홀</b> 브레드 보드 830홀(봉무형) 센서 테스트나, 회로 프로토타입을 작성할 때 사용됩니다.

<b>13</b> 1EA  <b>수동부저</b> 아두이노의 tone함수를 통해 소리를 내는 부저입니다.	<b>14</b> 6EA  <b>택트스위치 (12x12x7)</b> 스위치를 누르고 있을 경우만 ON됩니다.	<b>15</b> 3EA  <b>택트스위치 캡 (피랑,노랑,초록,빨강,하양)</b> 택트스위치를 사용할 때 스위치간의 구분을 할 수 있습니다.	<b>16</b> 3EA  <b>조도센서</b> 빛을 감지하거나 빛의 밝기를 아날로그로 출력해주는 CDS 센서입니다.
<b>17</b> 5EA  <b>LED 5mm (빨강,노랑,초록,하양,파랑)</b> 기본으로 사용되는 LED입니다. 동작전압 : 2.2~2.4V 사용전류 : 20mA 미만	<b>18</b> 1EA  <b>헤더핀 1x40/2.54mm</b> 핀 간격은 2.54mm이며 헤더핀의 길이는 약 1.15cm입니다.	<b>19</b> 1EA  <b>USB케이블 50cm</b> PC와 아두이노 우노 보드를 연결하여 프로그램을 다운로드 할 때 사용합니다.	<b>20</b> 1EA  <b>저항값 카드</b> 저항값을 쉽게 확인 할 수 있는 카드입니다. 사이즈 : 60mm x 50mm
<b>21</b> 1EA  <b>능동부저</b> Signal 단자가 HIGH 일 때 약 2.5kHz의 음이 발생됩니다.	<b>22</b> 1EA  <b>5V 1채널 릴레이 모듈</b> 아두이노의 디지털 핀과 모듈 하단의 IN 핀들을 연결해 릴레이를 제어할 수 있는 모듈입니다.	<b>23</b> 1EA  <b>8x8 도트 매트릭스 모듈</b> LED로 다양한 연출을 할 수 있습니다.	<b>24</b> 1EA  <b>4x4 16 키패드 모듈</b> 16개의 버튼을 사용할 수 있습니다.

# 아두이노 키트(Kit) : Part-2

<p>25 1EA</p> <p>무선 리모콘 키트</p> <p>핵파선을 사용해서 리모콘 기능을 구현할 수 있습니다.</p>	<p>26 2EA</p> <p>가을기 센서 스위치</p> <p>센서의 가을기에 따라 스위치 역할을 합니다.</p>	<p>27 1EA</p> <p>사운드 센서 모듈</p> <p>아두이노와 호환되는 사운드센서 모듈입니다.</p>	<p>28 1EA</p> <p>불꽃 센서</p> <p>근거리 화재, 불꽃을 감지하는 센서입니다.</p>	<p>37 1EA</p> <p>DC 5V 스텝 모터</p> <p>28BYJ48 스텝 모터 중 저렴한 편에 속하는 모델입니다. 5개의 핀을 사용합니다.</p>	<p>38 1EA</p> <p>DS1302 RTC 모듈</p> <p>아두이노 등 마이크로컨트롤러에서 사용이 가능합니다.</p>	<p>39 1EA</p> <p>아두이노 우노 프로토 쉴드</p> <p>UNO 보드에서 회로를 간단히 짜기 위해 보드 위에 얹어 사용하는 쉴드입니다.</p>	<p>40 1EA</p> <p>3축 가속도 센서 모듈</p> <p>가속도를 측정할 수 있는 센서입니다.</p>
<p>29 1EA</p> <p>모터 드라이버 모듈</p> <p>ULN2003 스텝 모터 드라이버 모듈 5V ~ 12V를 사용할 수 있습니다.</p>	<p>30 1EA</p> <p>LM35 온도 센서</p> <p>온도를 마닐로그 값으로 출력합니다.</p>	<p>31 1EA</p> <p>수위 센서 모듈</p> <p>센서 역할에 잠긴 정도를 마닐로그 값으로 출력합니다.</p>	<p>32 1EA</p> <p>SG90 서보모터</p> <p>Vcc, GND, 신호선, 총 3개의 핀이 있습니다. 로봇팔이나 자동차, 비행기 조종에 사용됩니다.</p>	<p>41 1EA</p> <p>5V DC모터</p> <p>5V DC모터</p>	<p>42 1EA</p> <p>인체 감지 센서 모듈</p> <p>핵파선을 이용해 움직임 감지하는 센서입니다. 오선이 감지되면 HIGH 신호를 출력합니다.</p>	<p>43 5EA</p> <p>다이오드 1N4001</p> <p>다이오드 1N4001</p>	<p>44 5EA</p> <p>세라믹 캐패시터 (22pF)</p> <p>세라믹 캐패시터 (22pF)</p>
<p>33 1EA</p> <p>초음파 거리 센서 모듈</p> <p>5V를 사용하여 만직 거리는 2cm에서 500cm입니다.</p>	<p>34 1EA</p> <p>조이스틱 모듈</p> <p>기본적으로 조이스틱 모듈은 두개의 가변저항이 서로 수직으로 회전하는 형태로 되어 있습니다.</p>	<p>35 1EA</p> <p>온습도 센서 모듈</p> <p>아두이노 온습도 센서중 가장 대중적으로 사용되는 DHT11 디지털 센서입니다.</p>	<p>36 1EA</p> <p>RGB LED 모듈</p> <p>RGB LED 모듈로 RGB LED 세개를 하나로 묶은 상품입니다.</p>	<p>45 5EA</p> <p>세라믹 캐패시터 (1uF)</p> <p>세라믹 캐패시터 (1uF)</p>	<p>46 5EA</p> <p>트랜지스터 2N2222</p> <p>트랜지스터 2N2222</p>	<p>47 5EA</p> <p>트랜지스터 BC547</p> <p>트랜지스터 BC547</p>	<p>48 5EA</p> <p>트랜지스터 BC557</p> <p>트랜지스터 BC557</p>
<p>49 2EA</p> <p>전해 캐패시터 (50V 10uF)</p> <p>전해 캐패시터 (50V 10uF)</p>	<p>50 2EA</p> <p>전해 캐패시터 (50V 100uF)</p> <p>전해 캐패시터 (50V 100uF)</p>						

# [참고 : 저항 값 읽기]



Color	First	Second	Third	Multiplier	Tolerance
Black	0	0	0	x1	
Brown	1	1	1	x10	1%
Red	2	2	2	x100	2%
Orange	3	3	3	x1000	
Yellow	4	4	4	x10 000	
Green	5	5	5	x100 000	0,50%
Blue	6	6	6	x1 000 000	0,25%
Violette	7	7	7	x10 000 000	0,10%
Gray	8	8	8		
White	9	9	9		
Silver				x0,01	10%
Gold				x0,1	5%

