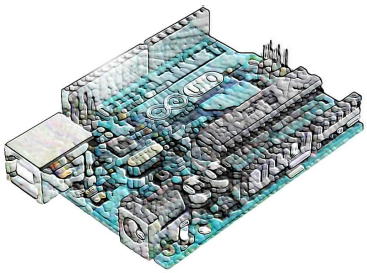
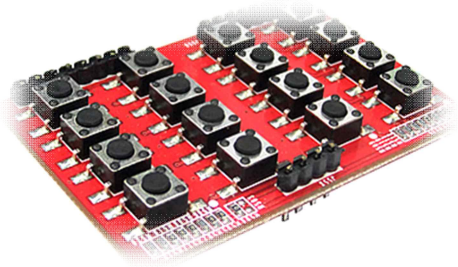


* 본 자료는 서경대학교 아두이노 프로그래밍 수업 수강자를 위해 작성된 강의 교재입니다.

강의교재– 아두이노 프로그래밍

3장 스위치 모듈

서경대학교 김진헌



3

스위치 모듈

내용

3.1 스위치의 사례 및 동작

스위치(switch)는 신호 선로를 연결(close)하거나 개방(open)하는 기계적 장치이다. 그림 3.1.1은 스위치 동작을 회로 작성이나 블록다이어그램에 표시할 때 사용하는 심벌을 보인 것이다. 그림에서 (a)는 평상시에는 개방되어 있다가 누를 때만 연결되는 NO 스위치를 보인 것이며 (b)는 평상시에는 연결되어 있다가 누를 때만 연결이 차단되는 NC 스위치를 보인 것이다. (c)는 스위치의 또 다른 표현으로 푸시 버튼 동작뿐만 아니라 밀어서 연결 여부를 결정하는 슬라이드 스위치 등의 동작을 기술할 때 사용한다.

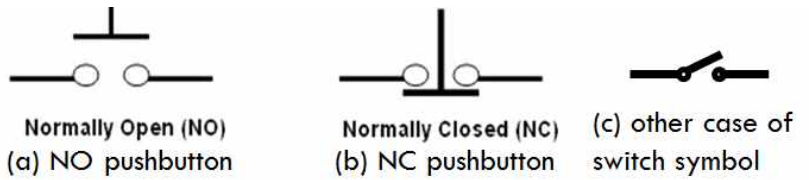


그림 3.1.1 스위치의 심벌

그림 3.1.2에 누르면 작동하는 push button 스위치의 사례를 보였다. 이런 종류의 스쉬치는 내부에 스프링이 있어서 버튼을 놓으면 원래 상태의 연결로 복원되는 특징을 갖는다.

그림 3.1.2 푸시 버튼의 사례

디지털 회로에서 많이 사용되는 소형 스위치의 외형과 그 연결 상태를 그림 3.1.3에 보였다. 본 스위치는 NO(Normally Open) 스위치이므로 버튼을 눌렀을 때만 회로가 연결되는 동작을 수행한다. ①번과 ③④ 중 아무 것이나, 혹은 ②과 ③④번 중 아무 것이나 조합하여 스위치 회로를 구성할 수 있다.

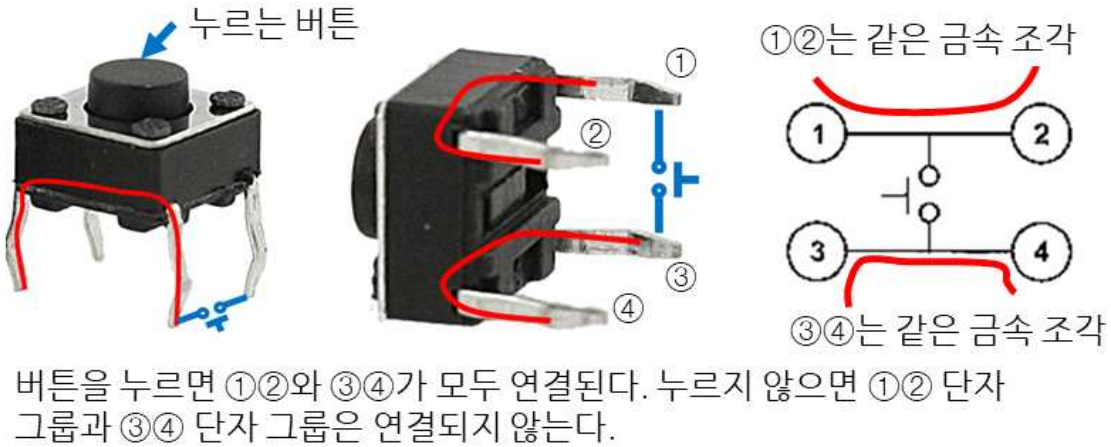
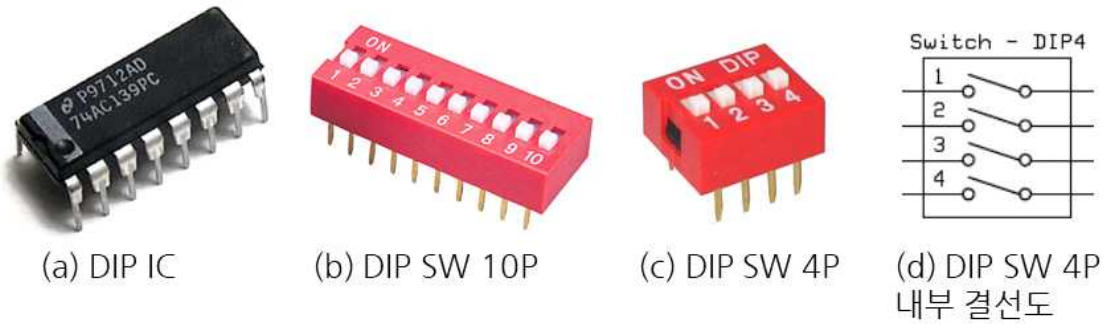


그림 3.1.3 푸시 버튼의 사례: 붉은 선으로 표기한 단자는 연결되어 있다.

그림 3.1.4(b)(c)는 8P, 4P 딥 스위치(DIP switch)를 보인 것이다. 스위치의 크기와 외형이 DIP type IC[1]와 크기와 모양이 같아서 붙인 이름이다. 이 스위치는 스위치 상단의 돌출된 부분을 볼펜 심 등으로 밀어서 각 단자의 연결 상태를 제어한다. ON으로 표시된 곳에 돌출 부분을 위치하게 하면 회로가 연결(closed)된다. 푸시 버튼 스위치는 버튼을 놓으면 원래 설정으로 복원된다. 그러나 내부에 스프링이 없는 이런 종류의 슬라이드(slide) 스위치는 한 번 설정되면 다시 바꾸지 않는 이상 그 상태가 계속 유지된다.

1) Dual In-line Package의 약자이다. IC 소자의 package type을 말하는 용어로 IC의 단자가 2개 열로 구성되어 있다. 이 단자가 PCB의 구멍에 삽입되어 납땜 과정을 거쳐 보드에 장착된다. DIP 스위치는 일반 DIP 형의 IC와 동일 모양과 크기를 가진 슬라이드 스위치를 일컫는 말이다.



(a) DIP IC (b) DIP SW 10P (c) DIP SW 4P (d) DIP SW 4P 내부 결선도

그림 3.1.4 DIP 스위치의 사례

그림 3.1.5에는 스위치를 상태를 읽어 들이는 회로를 구성한 사례를 보였다. (a)의 경우에는 스위치 ON(closed) 상태에서는 입력 단자에는 L(Low, 논리 0)가 입력되며 개방(OFF, open)하면 H(High, 논리 1)의 상태가 입력된다. (b)의 경우에는 버튼을 눌러 closed 상태가 되면 입력 단자에는 H가 입력되며, 개방하면 L가 입력된다. (c)의 경우에도 (b)와 같은 동작을 수행한다. 다만 (b)가 입력단에 저항이 연결되어 있으므로 H가 입력될 때 인가되는 전류의 양을 저항이 통제할 수 있으므로 다소 안전한 회로라고 할 수 있을 것이다.

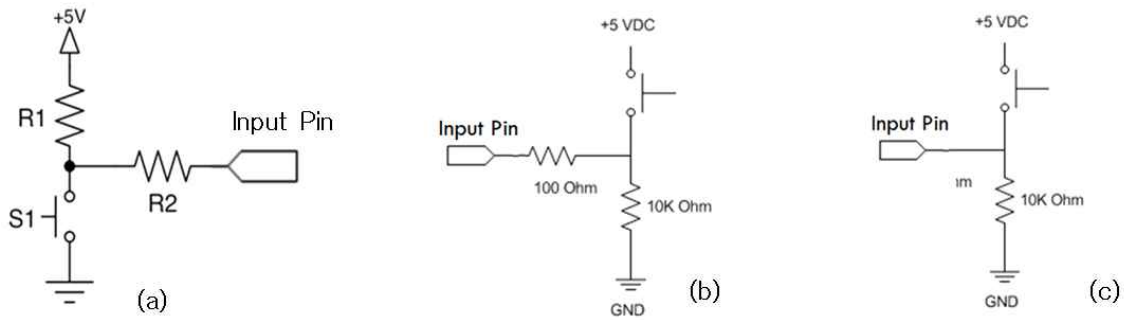


그림 3.1.5 스위치를 통해 H 혹은 L의 상태를 입력할 수 있는 회로

3.2 스위치 모듈과 그 인터페이스

스위치 실험에는 그림 3.2.1과 같이 푸시 버튼 스위치가 4x4의 배열로 나열되어 있는 스위치 모듈이 사용된다.

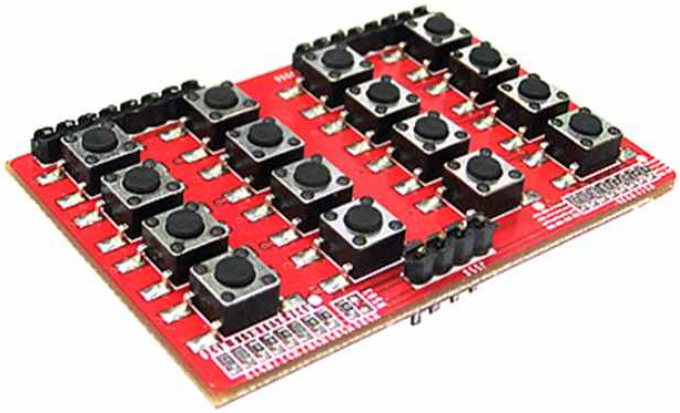


그림 3.2.1 스위치 장치의 사진

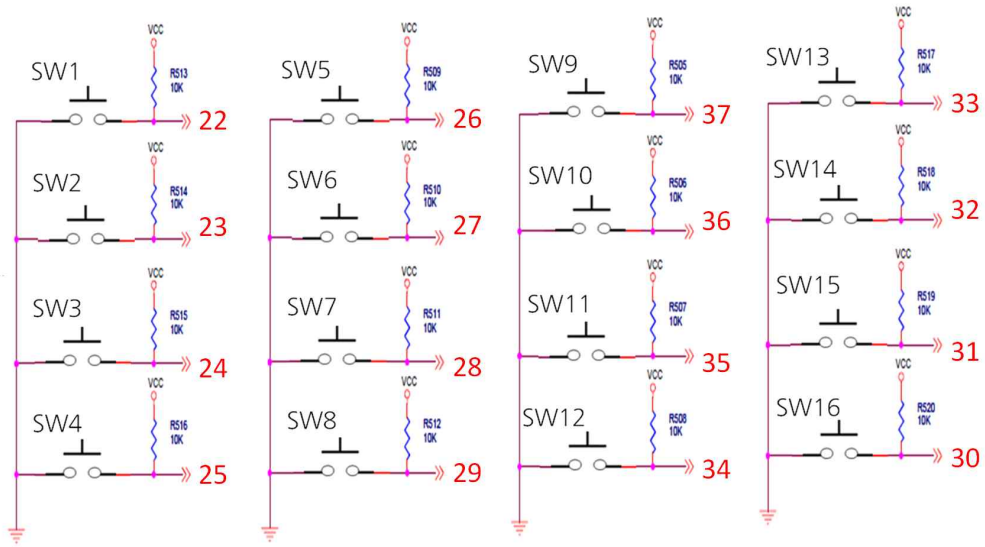


그림 3.2.1 스위치 모듈의 회로도.

그림 3.2.1에는 본 스위치 모듈의 회로도를 보였다. 각 스위치는 연결된 16개의 스위치의 한쪽은 아두이노의 16개 GPIO 단자에 연결된다. 아두이노에서는 GPIO 입력으로 설정하여 해당 단자에 유입되는 신호를 모니터링한다.

회로에서 보면 해당 단자는 저항을 통해 Vcc 전원에 연결되어 있으므로 평상시에는 단자에 High(논리1)가 공급된다[2]. 버튼을 누르면 단자는 GND(0V에 연결되므로 해당 단자는 Low(논리 0)가 된다. 프로그램에서는 이들 16개 단자의 상태를 모니터링하다가 L 상태가 발견되면 해당 단자가 눌린 것으로 판단하면 될 것이다.

2) 이렇게 신호를 저항을 통해 Vcc에 연결하는 동작을 pull up이라고 한다. 근래의 GPIO의 장점을 잘 활용한다면 구태여 외부에 pull up 저항을 붙이지 않아도 된다. pinMode() 함수에서 INPUT 모드로 설정할 때 INPUT_PULLUP으로 설정하면 내부에서 해당 단자를 pull up 처리한다.

3.3 아두이노의 GPIO 데이터 입력 표준 함수

아두이노에서는 다음과 같이 특정 단자에 입력되는 디지털 신호선의 상태를 읽어내는 GPIO 입력 함수를 제공한다.

GPIO 장치는 입출력에 앞서 ① 사용할 단자 번호를 지정하고, ② 이것을 입력 모드로 사용할 것인지 출력 모드로 사용할 것인지 결정해야 한다. 이러한 모드 설정은 작동 중에 수시로 변경할 수 있다.

void pinMode(pinNum, mode)		
기 능	해당 핀을 입력으로 사용할지 출력으로 사용할지 설정 (GPIO 장치 모드 설정)	
매개변수	uint8_t pinNum	설정하고 싶은 아두이노 단자(pin) 번호 GPIO가 가능한 단자 번호를 선택해야 한다.
	unit8_t mode	mode=INPUT : 입력 모드 설정 mode=INPUT_PULLUP : 풀업저항이 달린 입력 모드로 설정 mode=OUTPUT : 출력 모드 설정
리턴 값	void	없음

지정한 단자의 상태 값을 다음 함수를 통해 읽어 들인다.

void digitalRead(pinNum)		
기 능	지정된 핀으로 입력된 디지털 값 읽어오기 (GPIO 장치 읽어 들이기)	
매개변수	uint8_t pinNum	값을 읽어올 아두이노 단자 번호
리턴 값	uint8_t value	논리 값 0 혹은 1의 값이 반환된다.

3.4 스위치 모듈의 프로그래밍

예제(1) - 스위치 버튼 번호 출력하기

스위치를 누를 때마다 스위치의 핀 번호를 시리얼 모니터[3]를 통해 출력한다.

❐ 예제 1 : 스위치 모듈 버튼의 아두이노 핀 번호 출력

switch1.ino : 아두이노 표준 함수 – pinMode(), digitalRead()

```
01  int startPushPin=22;
02  int endPushPin=37;
03  void setup( ) {
04          for(int i=startPushPin; i<=endPushPin; i++) {
05                  pinMode(i, INPUT);
06          }
07          Serial.begin(9600);
08  }
09
10  void loop( ) {
11          String tmp="Pushed SW : " ;
12          for(int i=startPushPin; i<=endPushPin; i++) {
13                  if(digitalRead(i) == LOW) {
14                  Serial.println(tmp+i);
15                  }
16          }
17  }
```

본 실험을 위해서는 부록 C에서 다룬 COM 포트의 설정과 시리얼 포트 번

3) 시리얼 모니터(serial monitor)는 아두이노와 직렬통신의 결과를 출력하는 창이다. 시리얼 모니터를 활성화시키기 전에는 COM 포트의 번호를 드라이버가 배정해 준 가상 포트번호(본 사례의 경우에는 COM3)와 일치시켜야 한다.

호 설정이 일치하도록 그림 3.4.1과 같이 설정하여야 한다[4]. 시리얼 모니터 창을 열기 위해서는 "툴->시리얼 모니터"를 선택하거나 hot key (Control+Shift+M)를 입력하면 된다.

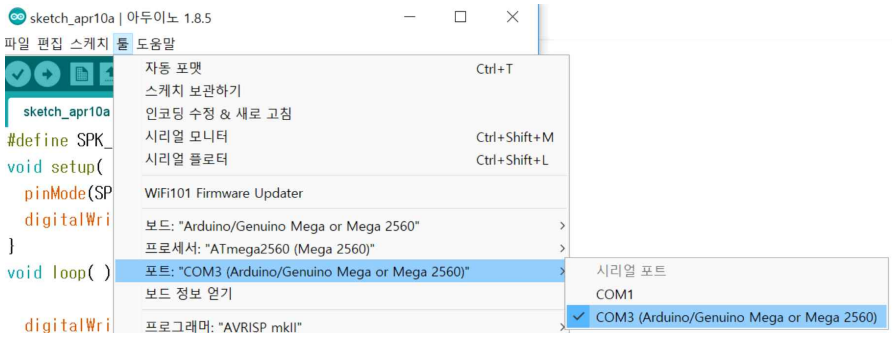


그림 3.4.1 통신(COM) 포트의 설정과 시리얼 모니터의 가동

예제 1의 프로그램을 수행하고 시리얼 모니터 창을 활성화시킨 후 버튼을 입력하면 그림 3.4.2와 같이 모니터 창에 누른 버튼의 아두이노 단자 번호가 출력된다. 그림의 사례는 SW1만을 누른 사례이다.

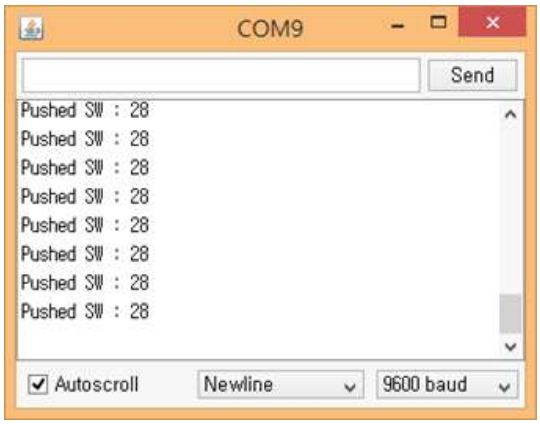


그림 3.4.2 시리얼 모니터 출력 결과

4) 보드 명이 일치하는 COM 번호가 배정되었는지 확인한다. COM1, COM2는 USB에 연결되지 않은 직렬 포트이기 때문에 사용할 수 없다.

3.5 스캔 방식의 키패드 모듈

지금까지 보인 스위치 모듈은 16개의 버튼을 상태를 받기 위해 16개의 입력장치를 이용하였다. 만약 현재의 PC 키보드처럼 100여 개가 넘는 키의 입력을 받아들이기 위해서라면 역시 100여 개의 GPIO 입력이 필요할 것인가?

실제 상황에서는 이런 방식으로 키보드(혹은 key pad)를 설계하지 않는다. 이번 사례에서는 4개의 출력과 4개의 입력으로 16개의 키패드를 구성하는 보이고자 한다. 참고로 이런 종류의 키패드 구동 방식에서는 R개의 출력과 C개의 입력으로 RxC 개의 키를 입력받을 수 있다.

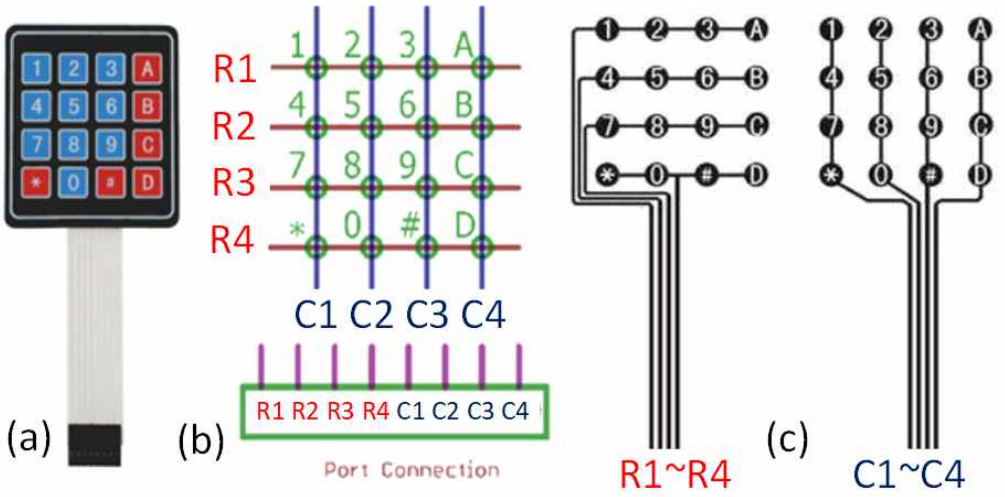


그림 3.5.1 4x4 membrane keypad

그림 3.5.1에 4개의 행과 열로 구성된 키패드의 실물(a)와 그 동작원리를 보여주는 키패드 배열을 보였다. (b)에서 R1~R4와 C1~C4는 서로 교차하는 지점에서는 해당 위치의 키를 누르는 순간 동안은 서로 연결(closed)된다. 누르지 않은 상태에서는 소재의 특성에 의해 서로 개방(open)된다. C1~C4의 모든 신호선은 각각 pull up 저항에 연결되어 있다[5]. (c)에서 보이듯 R1~R4는 4 bit GPIO 출력에 연결하며, C1~C4는 4 bit GPIO 입력에 연결한다. 이를 2개의 접촉점은 서로 포개어져 있다. 아두이노에서는 1비트 GPIO만 지원

5) pinMode()에서 INPUT_PULLUP을 선언하면 저항을 연결하지 않아도 된다.

하므로 실제로는 출력과 입력용으로 각각 4개의 GPIO를 사용한다면 보면 된다.

❑ 동작 원리

4개의 GPIO 출력 장치는 X로 명기된 신호 단자는 GPIO가 4개의 출력단자로 다음과 같이 순차적으로 그 중의 하나만 0이 되고 나머지는 1이 되는 신호를 발생하는 작업을 반복한다.

[R1R2R3R4]: [0111] ⇨ [1011] ⇨ [1101] ⇨ [1110] ⇨ [0111] ⇨ 무한반복

R 단자에 어떤 한 개의 row에 대해서만 0이 되는 데이터를 출력하고 난 후 키보드 모니터링 프로그램에서는 C 단자들의 입력 상태를 관찰한다. C가 모두 1이면 아무런 키를 누르지 않은 것이다. C 단자 중 Low가 되는 것은 그 열과 Low를 출력한 어떤 row와의 교차지점에 있는 버튼이 눌렸다는 것을 의미한다. 예를 들어 R=[0111]일 때 C=[1101]이라면 키[3]이 입력된 것이다. R=[1110]일 때 C=[0111]이 검출되었다면 키[*]가 입력된 것이다. 이렇게 [R, C]로 이루어진 이른바 '스캔코드(scan code)'를 해석하면 어떤 키를 입력했는지 알 수 있다.

PC의 키보드도 바로 이와 같은 방식으로 운영된다. PC 키보드는 내부에 8042 계열의 SCP(Single Chip Processor)가 있어서 같은 방식으로 R을 출력하고, C 값을 읽어 들여 완성된 [R, C]의 조합으로 키를 해석하여 PC로 USB 라인을 통해 그 정보를 전달한다. 이후 메인 보드의 프로세서에서는 이 값을 ASCII 등의 코드로 변환하여 응용 프로그램에 전달하는 과정을 수행한다.

❑ 실험

본 실험 과정은 Arduino Simulator로 불리는 Autodesk tinkercad™ Circuit로 기술하고자 한다. 본 시뮬레이터(소개 동영상 링크)는 간단한 전자회로나 아두이노 실습을 컴퓨터상에서 마치 실물을 가져다 놓고 실습하는 것처럼

연습해 볼 수 있다.

그림 3.5.2에는 16개의 키 중에서 2개의 키 조작을 검출해 내는 실험을 위한 수행하기 위한 팅커캐드의 수행 화면을 보였다.

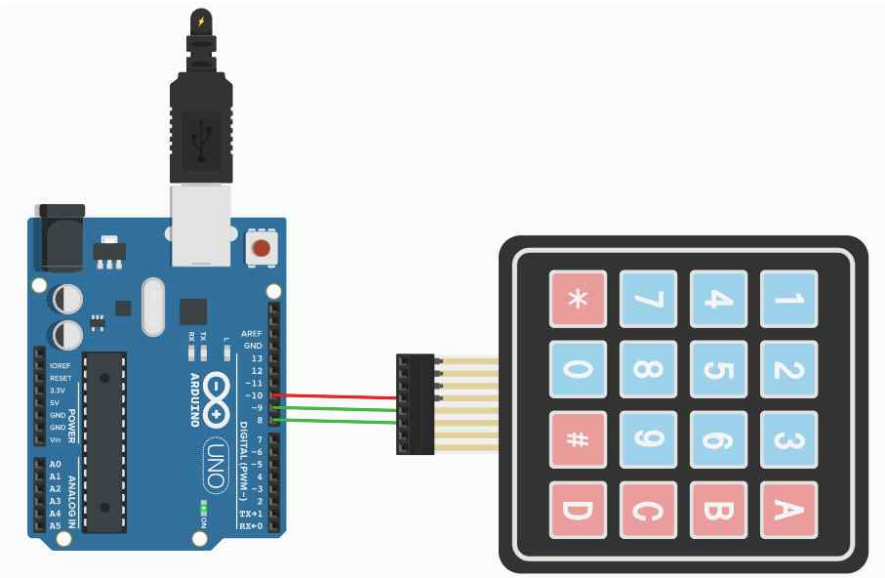


그림 3.5.2 2개의 자판(*, 0)만을 검출하기 위한 배선

아두이노의 GPIO 출력은 R4만을 LOW로 제공하고 있고 버튼 *가 눌리면 C1 단자가 LOW가 되고, 버튼 0가 눌리면 C2 단자가 LOW가 된다. 위의 그림에서 빨간 점퍼선은 키패드의 R4 신호선을 GPIO 10번으로 연결한다. 또한 녹색 점퍼선은 C1, C2를 GPIO 9번과 8번에 연결하고 있다.

본 H/W 결선에 걸맞는 예제 프로그램을 아래 예제 2에 보였다. 입력 모드 설정에 있어서 주의할 점은 INPUT_PULLUP으로 설정해야 한다는 것이다. 그렇지 않으면 버튼을 누르지 않을 때 입력 단자의 로직 레벨이 floating 상태가 되어 어떤 값으로 읽힐지 불확실할 뿐만 아니라 불안정한 회로를 구성하는 것으로 알려져 있다. 만약 C 선로를 외부에 풀업 저항을 연결해 준다면 이런 설정은 필요 없다.

❐ 예제 2 : 스위치 모듈 버튼의 아누이노 핀 번호 출력

simple_4x4.ino : 아두이노 표준 함수 – pinMode(), digitalRead()

```
01  int R4 = 10;
02  int C1 = 9;
03  int C2 = 8;
04
05  void setup() {
06    pinMode(C1, INPUT_PULLUP);
07    pinMode(C2, INPUT_PULLUP);
08    pinMode(R4, OUTPUT);
09    digitalWrite(R4, LOW);
10    Serial.begin(9600);
11  }
12
13  void loop() {
14    if ( digitalRead(C1) == LOW )
15      Serial.println(" * pushed");
16          delay(100);
17
18    if ( digitalRead(C2) == LOW )
19      Serial.println(" 0 pushed");
20          delay(100);
21  }
```

이 프로그램을 작동하고 시리얼 모니터 창을 연 후 키패드의 *, 0 버튼을 눌러 보아서 그림 3.5.3과 같이 올바른 문자가 출력되는지 확인하면 된다. 실제로는 같은 글자가 반복되어 나올 때도 있는데 이는 기계적인 접촉 불안정 때문에 발생하는 chattering 문제 때문이다. H/W 회로를 이용하여 방지하는 방법도 있고, 단순히 적절한 delay()를 조절하여 해결하는 방안도 있다.

만약 16개의 자판을 모두 지원하게 하고자 한다면 그림 3.5.4와 같은 결선

이 필요할 것이다.

그림 3.5.3 직렬 모니터의 화면

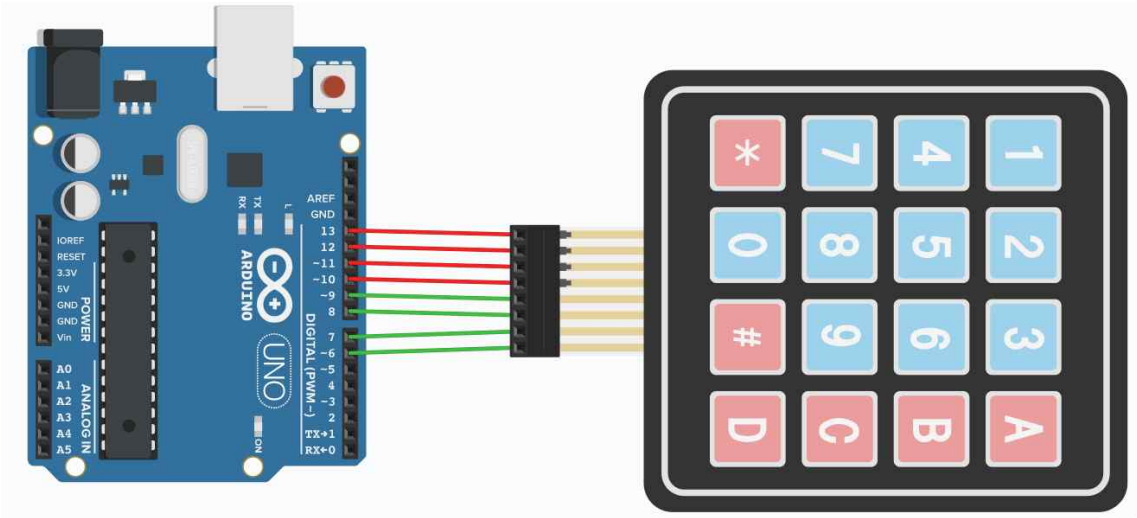


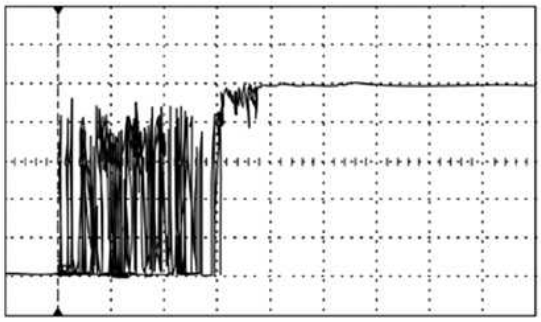
그림 3.5.4 16개의 자판을 모두 지원하기 위한 결선

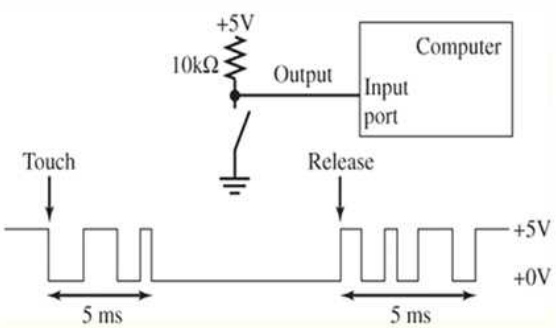
3.6 고찰

다음 주제에 대하여 답할 수 있는지 검토해 보면서 그동안 습득한 지식을 정리해 보자.

1. 토글(toggle) 스위치의 동작을 기술하고 이를 프로그램으로 구현하시오.

2. 채터링(chattering)에 대한 검토





채터링 현상. L에서 H로 천이하는 과도 구간에서 H, L가 반복된다.

기계적 장치에 의한 회로의 연결동작을 미세 시간 단위로 관찰하면 선로의 연결과 개방이 계속 반복되는 채터링 현상이 발생한다. 실험을 통해 채터링이 발생하는 현상을 검증하고자 한다.

목표: 1회 버튼을 누를 때 최대 몇 회의 논리 상태 변화가 일어나는지 계수하여 이를 출력한다.

❒ 1번 스위치를 누른 회수를 시리얼 모니터에 출력하고자 한다. 이때 편의상 16번 스위치를 누르면 그 회수를 출력하기로 한다. 아래 힌트를 참조하여 프로그램을 작성하시오.

setup에서 할일
1. SW1, SW16을 입력 모드로 설정
2. 시리얼 모니터 전송속도 설정
3. 전역변수 count를 0으로 초기화

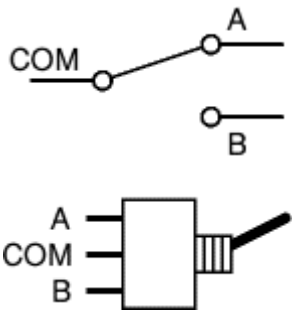
loop에서 할 일
스위치는 누르지 않으면 H, 누르면 원래는 L만 읽혀야 하는데 H->L의 상태 천이가 여러 번 발생한다. 스위치 SW1의 이러한 상태 천이 회수를 count++ 변수에 증가시켜 저장한다.
SW16를 누르면 L가 눌리면 일정시간 delay()한 후 count 값을 출력 & count=0으로 초기화.

☑ 실제로 수행하여 보면 스위치를 1회만 눌렀는데도 상당한 수의 count 값이 발생한다.

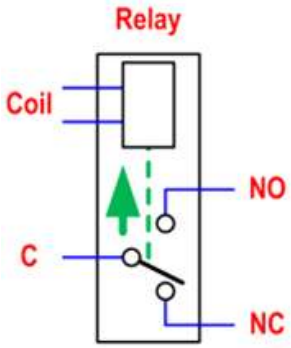
3. PC의 경우 Ctrl+C와 같이 2 버튼을 함께 누르는 동작을 구현하고 있다. 이와 같이 본 실험에서도 버튼 2개를 동시에 누른 사실을 인지할 수 있는 방안에 대하여 기술하시오.

4. 현 스위치 모듈에서는 사용하는 GPIO의 비트 수를 줄이기 위한 방안을 설명하시오.

6. 릴레이 소자는 전기적으로 스위치 동작과 같은 역할을 수행한다. 릴레이 소자의 동작원리와 설명하고 이를 아두이노에 연결하기 위한 방안을 제시하시오. => 구체적 실현 사례: 220V 전구의 점등, 소등 동작을 아두이노 프로그램을 구현한다.



toggle switch



relay

7. 4x4 키패드를 스캐닝하여 눌린 키의 문자(0~9, A~D, #, *)를 반환하는 함수 _getch()를 작성하시오. 키를 누르지 않으면 함수를 종료하지 않는 것으로 한다. 참고 URL: circuitdigest.com, 아두이노 4X4 멤브레인 키패드 설치 사례 및 예제(1), (2)