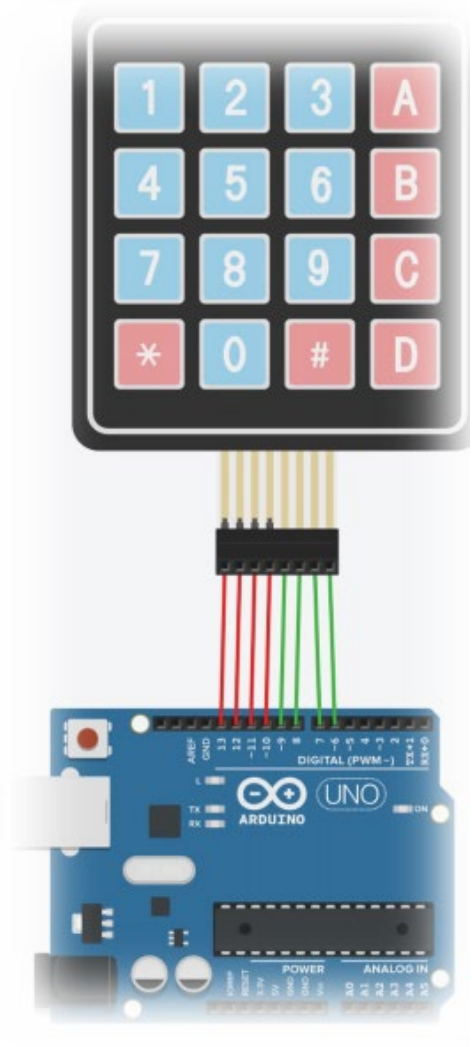


아두이노 프로그래밍 1 차과제

4 x 4 키패드 모듈 보고서

2019305059 이현수

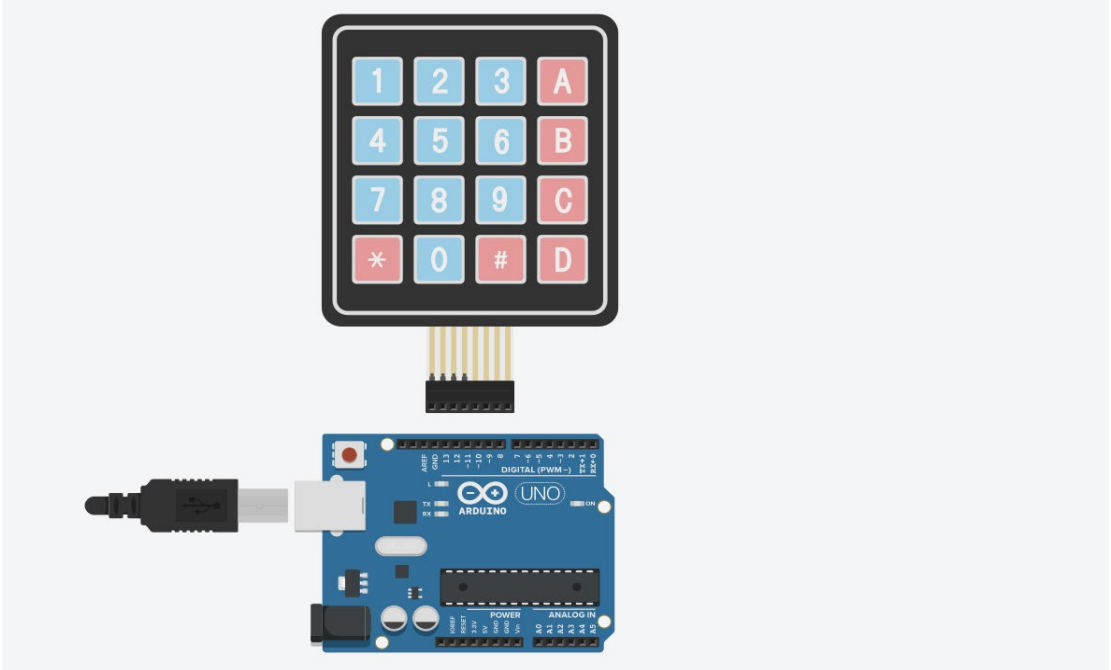
서론

4 x 4 키패드 모듈 완성하기

- 실험환경 Tinkercad circuit 을 이용하기
- 교과서 3.5 스캔방식의 키패드 모듈 사용하기
- 키보드 입력함수 _getch()를 만들어 키패드의 문자 출력하기
- 아두이노 표준함수만 사용하기

본론

1. 실험환경에 Tinkercad circuit 에 들어가서 Arduino Uno R3 와 Keypad 4 x 4 를 설치한다.



2. 교과서 3.5 스캔방식의 모듈을 사용해야 한다.

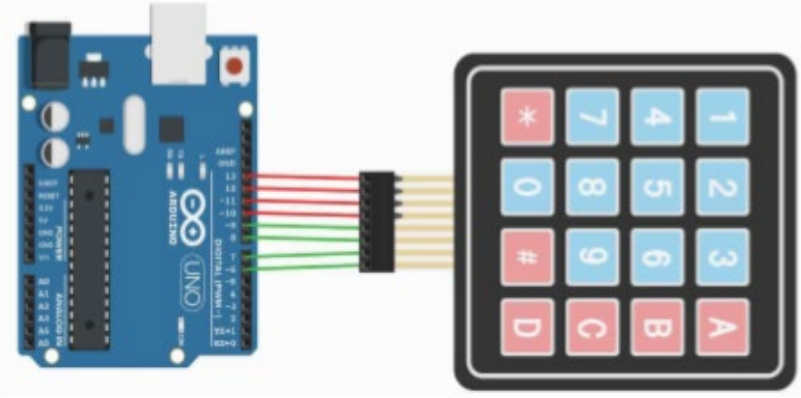
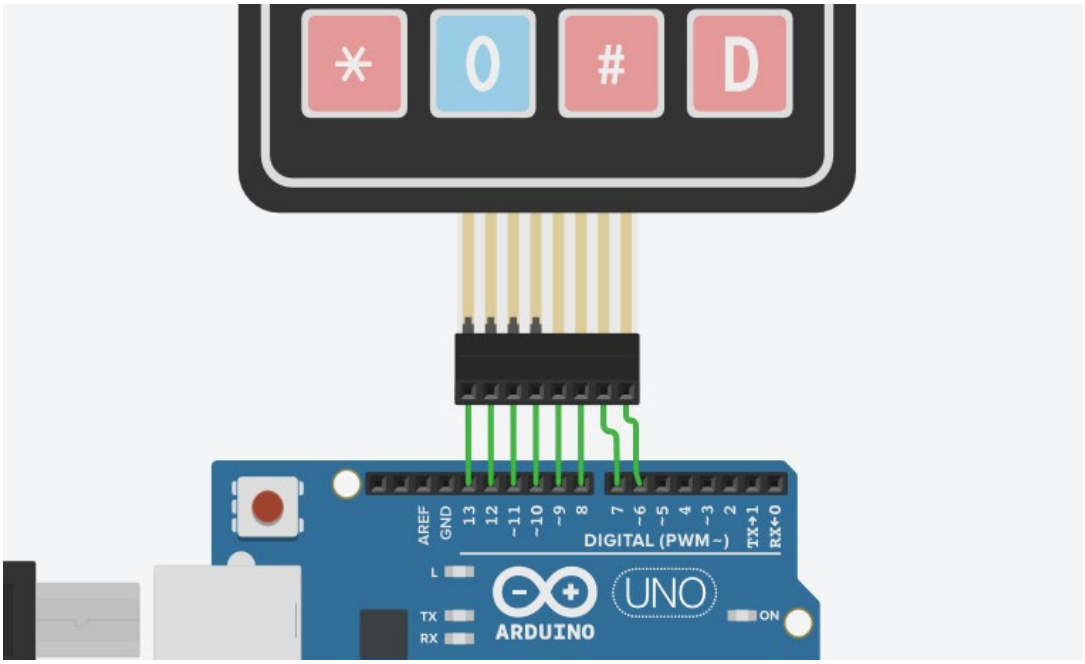


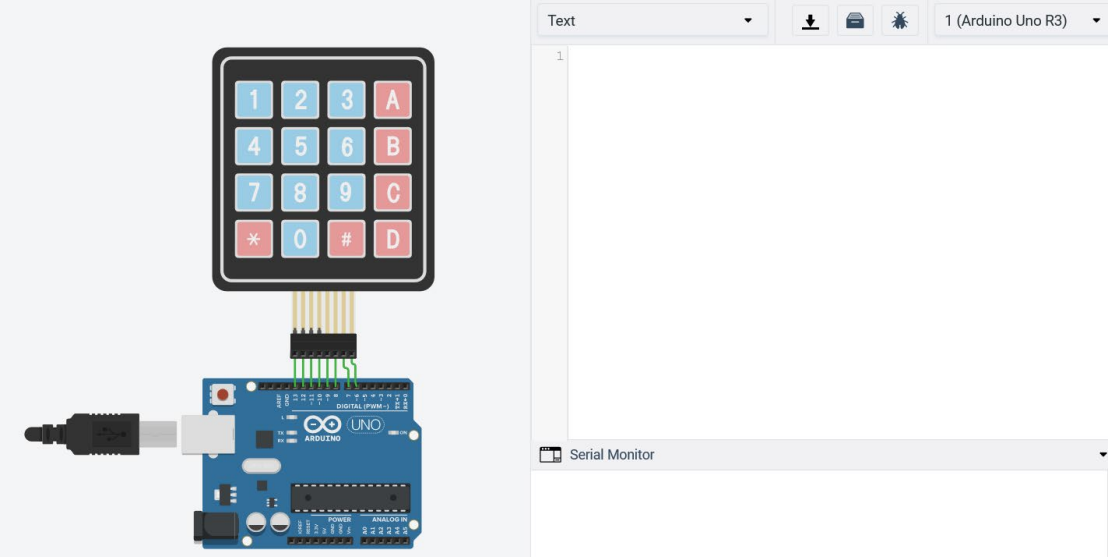
그림 3.5.4 16개의 자판을 모두 지원하기 위한 결선

교과서 3.5 스캔방식 모듈은 R1 은 13 번, R2 는 12 번, R3 는 11 번, R4 는 10 번, C1 은 9 번, C2 는 8 번, C3 는 7 번, C4 는 6 번이다.



위 그림과 같이 단자와 핀을 연결한다.

3. 코드 짜기



프로그래밍을 할 때 아두이노 표준함수만 사용해야 한다.

pinMode / digtalRead / digtalWrite / Serial.begin / Serial.print 등의 함수를 사용하면 된다.

```
1  int R1=13;
2  int R2=12;
3  int R3=11;
4  int R4=10;
5  int C1=9;
6  int C2=8;
7  int C3=7;
8  int C4=6;
9
10 char _getch();
11
```

먼저 전역변수에 핀이름을 정수형 변수로 선언하고 해당되는 단자숫자를 할당한다. 그리고 _getch() 함수 원형을 선언한다.

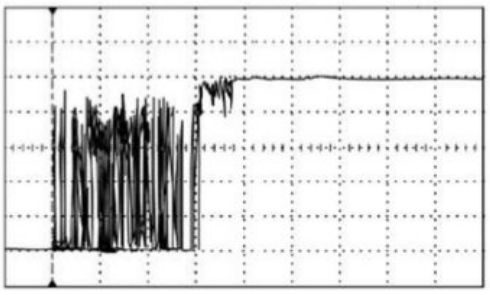
```
12  void setup()
13  {
14    pinMode(C1, INPUT_PULLUP);
15    pinMode(C2, INPUT_PULLUP);
16    pinMode(C3, INPUT_PULLUP);
17    pinMode(C4, INPUT_PULLUP);
18    pinMode(R1, OUTPUT);
19    pinMode(R2, OUTPUT);
20    pinMode(R3, OUTPUT);
21    pinMode(R4, OUTPUT);
22    Serial.begin(9600);
23  }
24
```

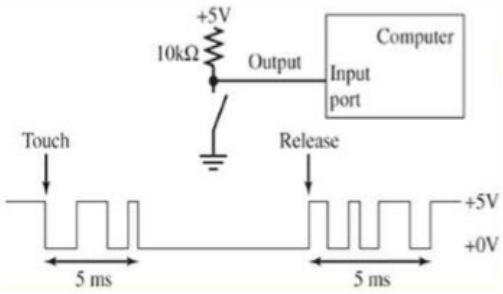
Void setup()에는 pinMode 함수와 Serial.begin 함수를 사용한다.

C1~C4 핀은 INPUT_PULLUP(풀업저항이 달린 입력모드)모드로, R1~R4 핀은 OUTPUT(출력모드)모드로 설정한다.

```
25  void loop()
26  {
27    char a;
28    a=_getch();
29    Serial.print(a);
30    Serial.println(": pushed");
31    delay(200);
32  }
```

Void loop()에는 문자형 자료형 a 를 선언한다. 그리고 _getch()함수로부터 문자를 받은 후 Serial.print(a)를 이용해 입력받은 a 를 출력 후 Serial.println(": pushed");를 통해 a 다음에 : pushed 문자열을 출력 후 다음줄로 이동한다.





그다음에 delay(200); 구문이 있다. 이것은 채터링현상을 막기위해 사용했다.

처음에 delay(100); 으로 해봤는데 채터링현상이 간혹 발생했다. 그래서 delay(200); 으로 설정했는데 채터링현상은 발생하지 않았다.

```
char _getch(){
  char ch;
  for(;;)
  {
     for(int i=10;i<=13;i++)
     {
       digitalWrite(10,HIGH);
       digitalWrite(11,HIGH);
       digitalWrite(12,HIGH);
       digitalWrite(13,HIGH);
       digitalWrite(i,LOW);
       if(digitalRead(C1)==LOW)
       {
           if(digitalRead(R1)==LOW)
           {
            ch='1';goto AAA;
           }
           else if(digitalRead(R2)==LOW)
           {
             ch='4';goto AAA;
           }
           else if(digitalRead(R3)==LOW)
           {
             ch='7'; goto AAA;
           }
           else
           { ch='*'; goto AAA; }
        }
```

char _getch()함수다. 먼저 문자형 자료형 ch 를 선언한다.

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

4 x 4 키패드의 동작원리는 R 단자 중 하나만 0 이 되고 나머지는 1 이 되는 신호를 발생하는 작업을 무한반복한다.

R 단자 중 한 개만 0 이되는 데이터를 출력하면서 C 단자들의 입력 상태를 관찰한다. C 는 아무런 키도 눌리지 않았다면 1 인 상태가 된다.

하지만 만약 어떤 키를 누르게 되면 그 C 단자는 0 이 된다. 그래서 [R, C]로 이루어진 스캔코드를 해석하면 어떤 키를 입력했는지 알 수 있다.

```
35    for(;;)
36    {
37       for(int i=10;i<=13;i++)
38       {
39         digitalWrite(10,HIGH);
40         digitalWrite(11,HIGH);
41         digitalWrite(12,HIGH);
42         digitalWrite(13,HIGH);
43         digitalWrite(i,LOW);
```

Char _getch()함수의 for 문은 이러한 원리를 구현하기 위한 것이다.

for 문을 무한히 반복하면서 R 단자의 한 개에만 LOW 를 입력한다. 이때 digitalWrite 함수를 사용한다.

```
44          if(digitalRead(C1)==LOW)
45          {
46              if(digitalRead(R1)==LOW)
47              {
48               ch='1';goto AAA;
49              }
50              else if(digitalRead(R2)==LOW)
51              {
52                ch='4';goto AAA;
53              }
54              else if(digitalRead(R3)==LOW)
55              {
56                ch='7'; goto AAA;
57              }
58              else
59              { ch='*'; goto AAA; }
60           }
```

만약 C 단자가 LOW 상태, 즉 키패드를 통해 입력을 받으면, 그 순간
어떤 R 단자가 LOW 상태인지 각각 조건문을 이용해서 판단 후 해당하는 문자를 ch 문자형에 할당한다.

이때 각각의 핀상태를 읽기 위해서 digitalRead 함수를 사용한다.

```
if(digitalRead(C1)==LOW)
{
    if(digitalRead(R1)==LOW)
    {
     ch='1';goto AAA;
    }
    else if(digitalRead(R2)==LOW)
    {
      ch='4';goto AAA;
    }
    else if(digitalRead(R3)==LOW)
    {
      ch='7'; goto AAA;
    }
    else
    { ch='*'; goto AAA; }
 }

 else if(digitalRead(C2)==LOW)
 {
    if(digitalRead(R1)==LOW)
    {
     ch='2';goto AAA;
    }
    else if(digitalRead(R2)==LOW)
    {
      ch='5';goto AAA;
    }
    else if(digitalRead(R3)==LOW)
    {
      ch='8'; goto AAA;
    }
    else
    { ch='0'; goto AAA; }
 }
```

```
else if(digitalRead(C3)==LOW)
{
   if(digitalRead(R1)==LOW)
   {
    ch='3';goto AAA;
   }
   else if(digitalRead(R2)==LOW)
   {
     ch='6';goto AAA;
   }
   else if(digitalRead(R3)==LOW)
   {
     ch='9'; goto AAA;
   }
   else
   { ch='#'; goto AAA; }
}

else if(digitalRead(C4)==LOW)
{
   if(digitalRead(R1)==LOW)
   {
    ch='A';goto AAA;
   }
   else if(digitalRead(R2)==LOW)
   {
     ch='B';goto AAA;
   }
   else if(digitalRead(R3)==LOW)
   {
     ch='C'; goto AAA;
   }
   else
   { ch='D'; goto AAA; }
}
```

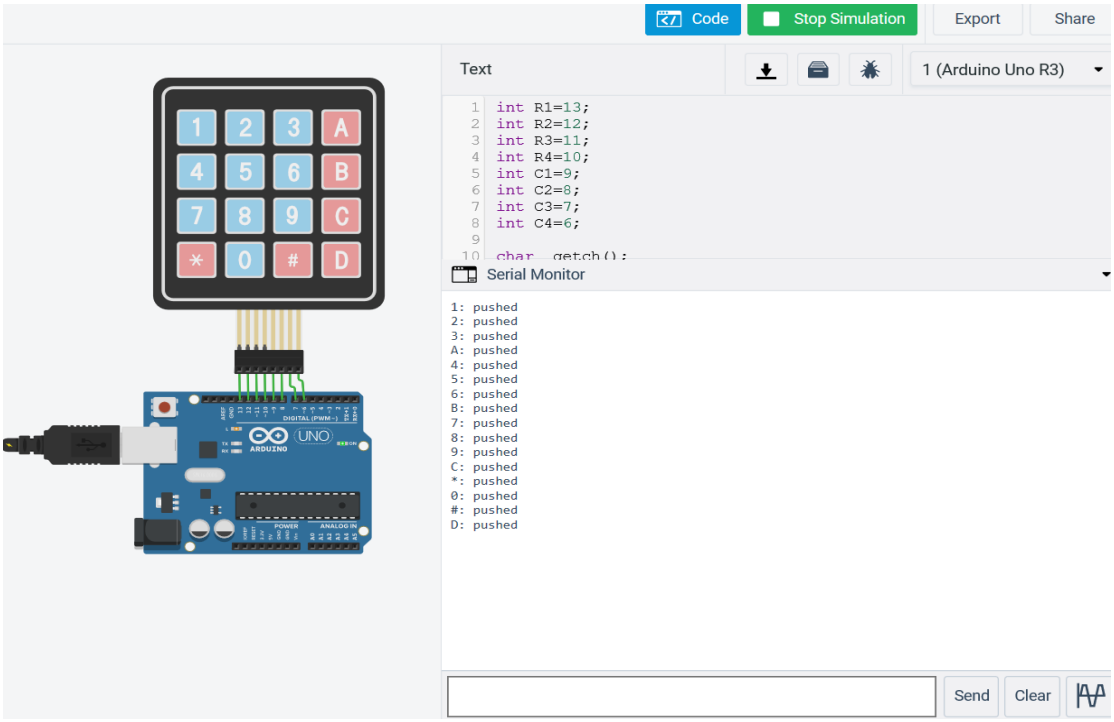
위 그림처럼 C1~C4 가 LOW 상태가 될 때 수행될 조건문과 그 조건문 안에 그 순간 LOW 상태인 R 단자를 알기위해 4 개의 조건문이 각각 있고, 그에 해당하는 알맞은 문자가 ch 문자형에 삽입된다. 그리고 goto 문을 통해 AAA 로 이동한다.

반복문은 goto 문을 통해서 탈출하게 되는데, goto 문이 실행되기 위해서 사용자가 키패드의 키 중 아무키나 하나 선택을 해야한다. 그래서 사용자가 키패드를 누르기전까지 프로그램은 종료하지 않고 사용자의 입력을 무한히 기다린다.

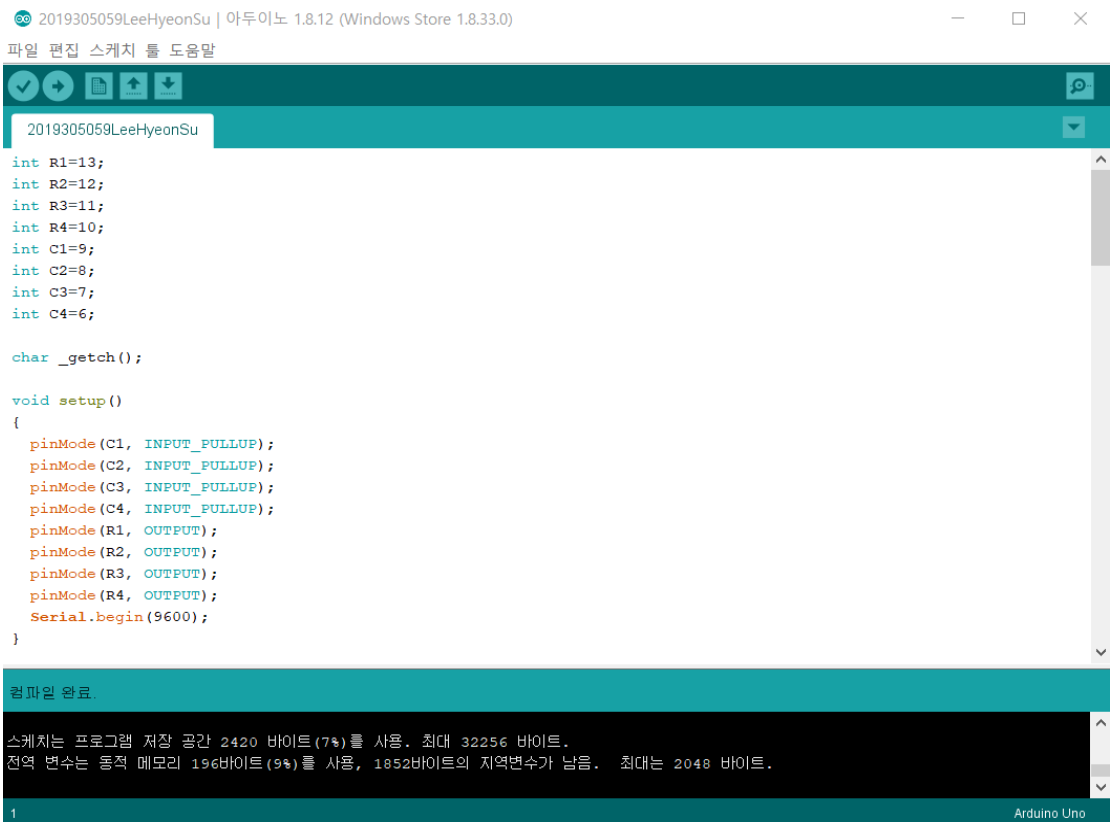
```
111             }
112         }
113     }
114     AAA:
115     return(ch);
116 }
```

AAA 는 반복문 밖에 위치한다. 그리고 ch 를 리턴한다.

4. 결과



실행을 시키면 정상적으로 작동한다.



스케치에서도 정상적으로 컴파일 된다.