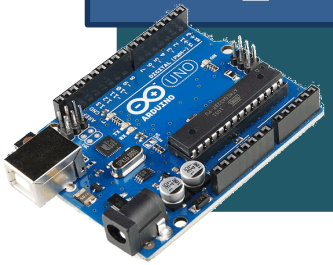
2022년 IoT기반 스마트 솔루션 개발자 양성과정



Firmware [펌웨어]

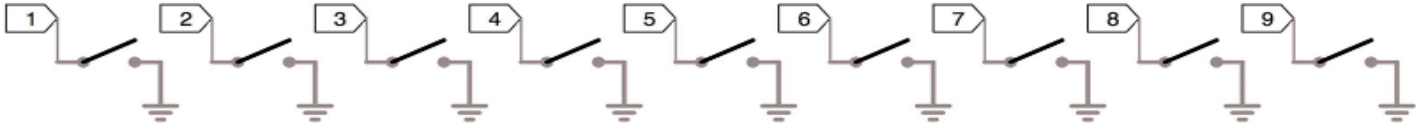
16-KeyPad 4x4

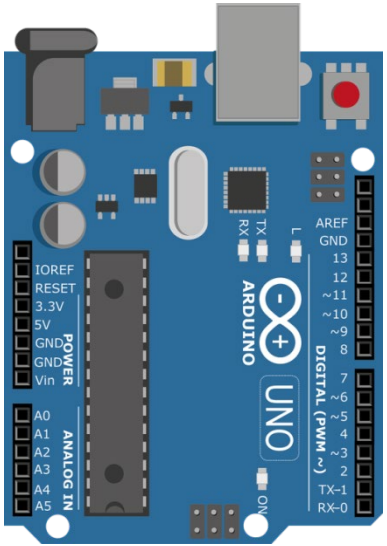
담당 교수 : 유근택

010-5486-5376

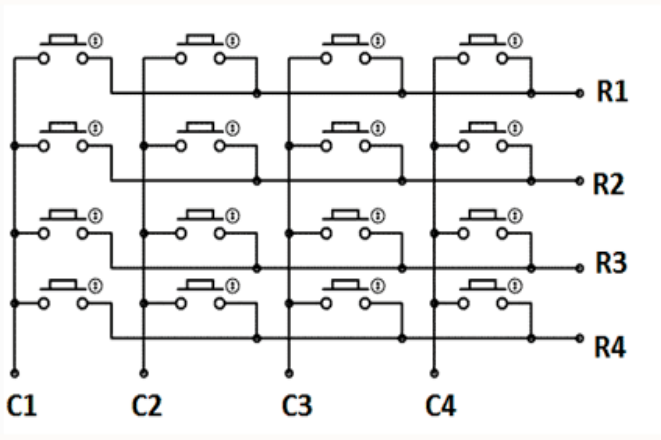
rgt3340@naver.com

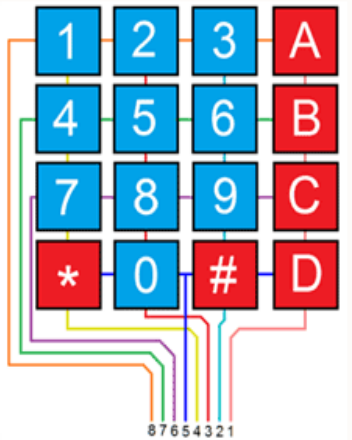
Without a matrix





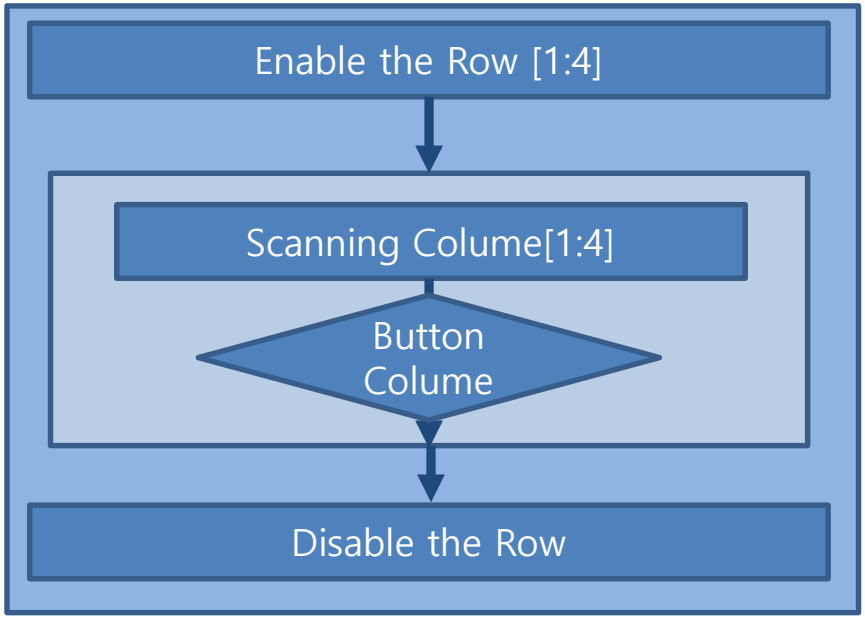
4×4 matrix


R1
R2
R3
R4
C1
C2
C3
C4


1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
87654321

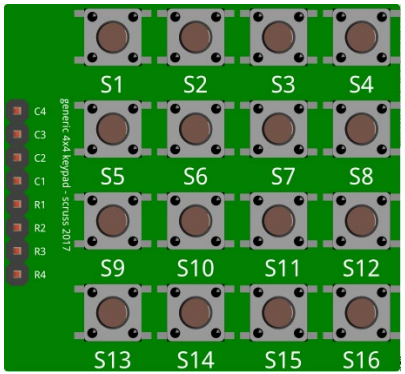

1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D

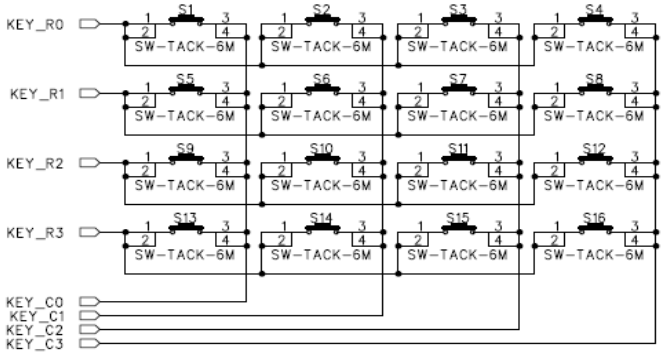
Keyboard Matrix Code



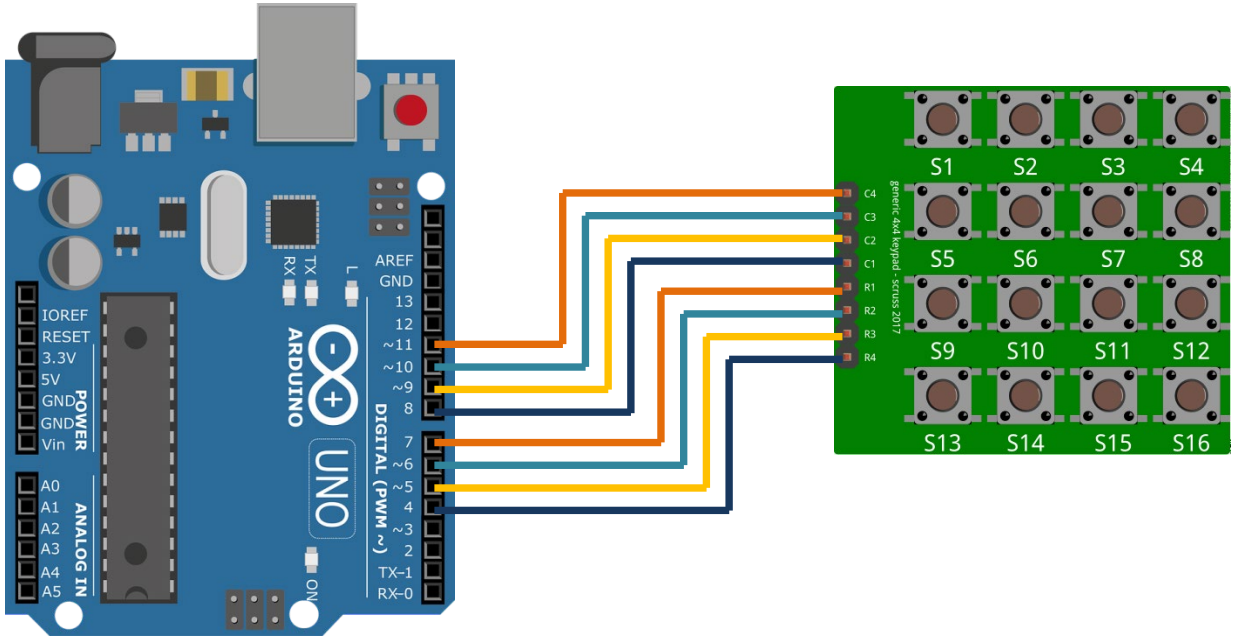
KeyPad 4x4 Module

	kEY_C0=1000	kEY_C1=0100	kEY_C2=0010	kEY_C3=0001
KEY_R0=1000	SW1(0)	SW2(1)	SW3(2)	SW4(3)
KEY_R1=0100	SW5(4)	SW6(5)	SW7(6)	SW8(7)
KEY_R2=0010	SW9(8)	SW10(9)	SW11(A)	SW12(B)
KEY_R3=0001	SW13(C)	SW14(D)	SW15(E)	SW16(F)





Wiring

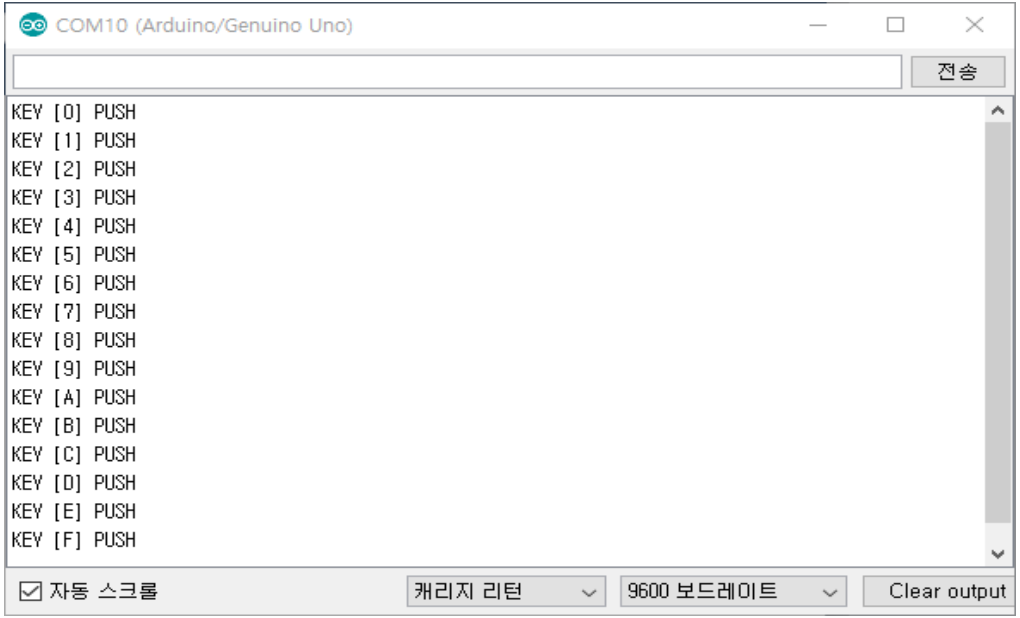

IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
DIGITAL (PWM~)
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
L
TX
RX
ON
C4
C3
C2
C1
R1
R2
R3
R4
generic 4x4 keypad - sctuss 2017
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16

KeyPad-1

```
int KeyRow[4] = {7,  6,  5,  4 };
int KeyCol[4] = { 8,  9, 10, 11};

void setup( ) {
  Serial.begin(9600);
  for(int k=0; k<4; k++)  {
    pinMode(KeyRow[k], OUTPUT);
    digitalWrite(KeyRow[k], HIGH);
    pinMode(KeyCol[k], INPUT_PULLUP);
  }
}
```

```
void loop( ) {
  int nRow, nNumber;
  for(int k=0; k<4; k++) {
    digitalWrite(KeyRow[k], LOW);
    nRow = k*4;
    for(int m=0; m<4; m++) {
      if ( !digitalRead(KeyCol[m]) ) {
        nNumber=nRow+m;
        Serial.print("KEY [");
        Serial.print(nNumber, HEX);
        Serial.println("] PUSH");
      }
    }
    digitalWrite(KeyRow[k], HIGH);
    delay(100);
  }
}
```

Serial Monitor

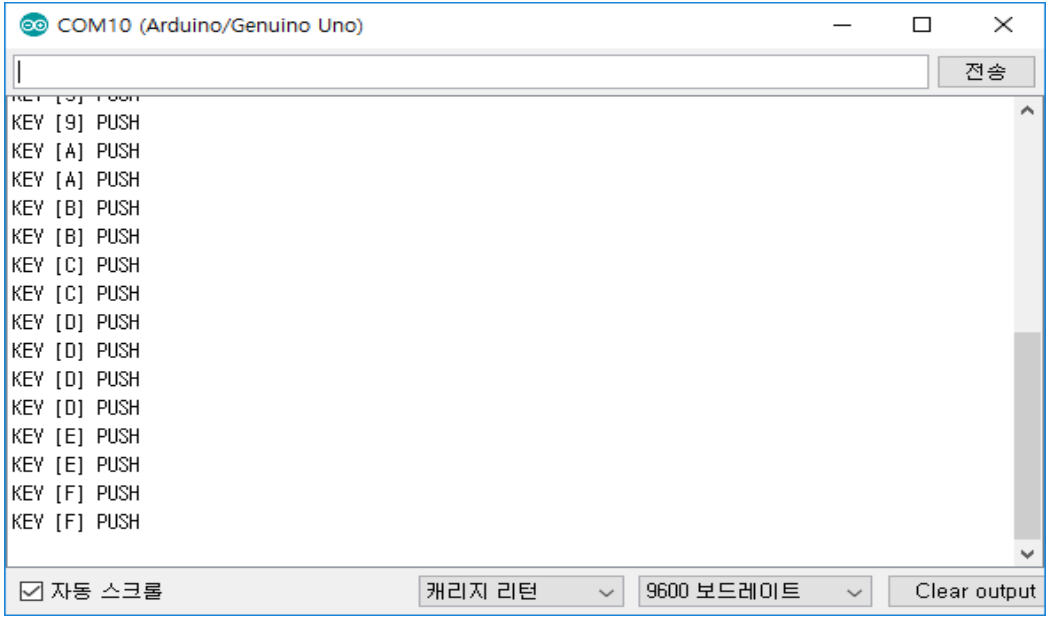


KeyPad-2 : KeyScan( )

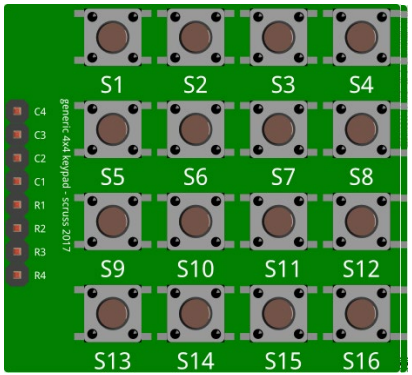
```
int KeyRow[4] = {7,  6,  5,  4 };
int KeyCol[4] = { 8,  9, 10, 11};
int KeyValue=-1;

void setup( ) {
  Serial.begin(9600);
  for(int k=0; k<4; k++)  {
    pinMode(KeyRow[k], OUTPUT);
    digitalWrite(KeyRow[k], HIGH);
    pinMode(KeyCol[k], INPUT_PULLUP);
  }
}

void loop( ) {
  KeyValue=KeyScan();
  if (KeyValue>=0){
    Serial.print("KEY=");
    Serial.print(KeyValue, HEX);
    Serial.println(" Pressed");
  }
  delay(100);
}
```

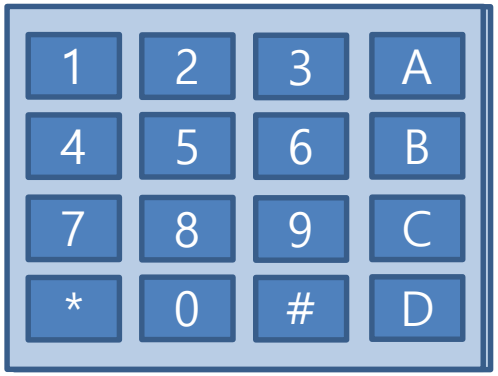
```
int KeyScan( ){
  int nRow, nNumber=-1;
  for(int k=0; k<4; k++) {
    digitalWrite(KeyRow[k], LOW);
    nRow = k*4;
    for(int m=0; m<4; m++) {
      if ( !digitalRead(KeyCol[m]) ) nNumber=nRow+m;
    }
    digitalWrite(KeyRow[k], HIGH);
  }
  return nNumber;
}
```

Serial Monitor



KeyPad-3 : user Assign Key



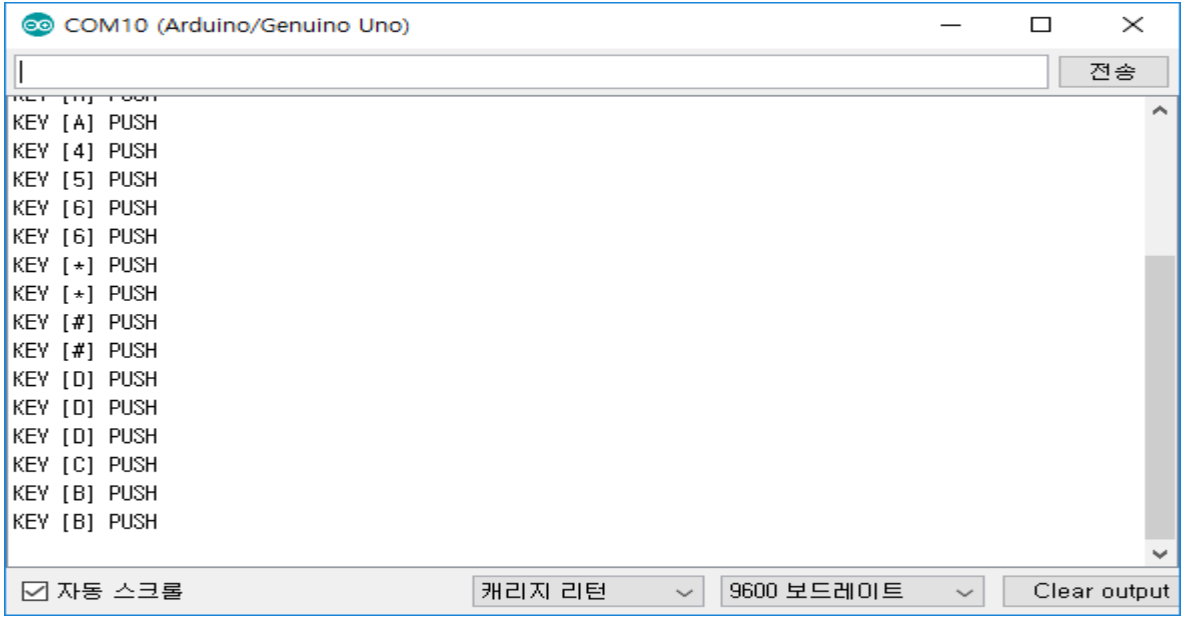


KeyPad-3 : user Assign Key

```
int KeyRow[4] = {7,  6,  5,  4 };
int KeyCol[4] = { 8,  9, 10, 11};
int KeyValue=-1;
char KeyCode[16]={ '1', '2', '3', 'A',
                   '4', '5', '6', 'B',
                   '7', '8', '9', 'C',
                   '*', '0', '#', 'D'  };

void setup( ) {
  Serial.begin(9600);
  for(int k=0; k<4; k++)  {
    pinMode(KeyRow[k], OUTPUT);
    digitalWrite(KeyRow[k], HIGH);
    pinMode(KeyCol[k], INPUT_PULLUP);
  }
}
```

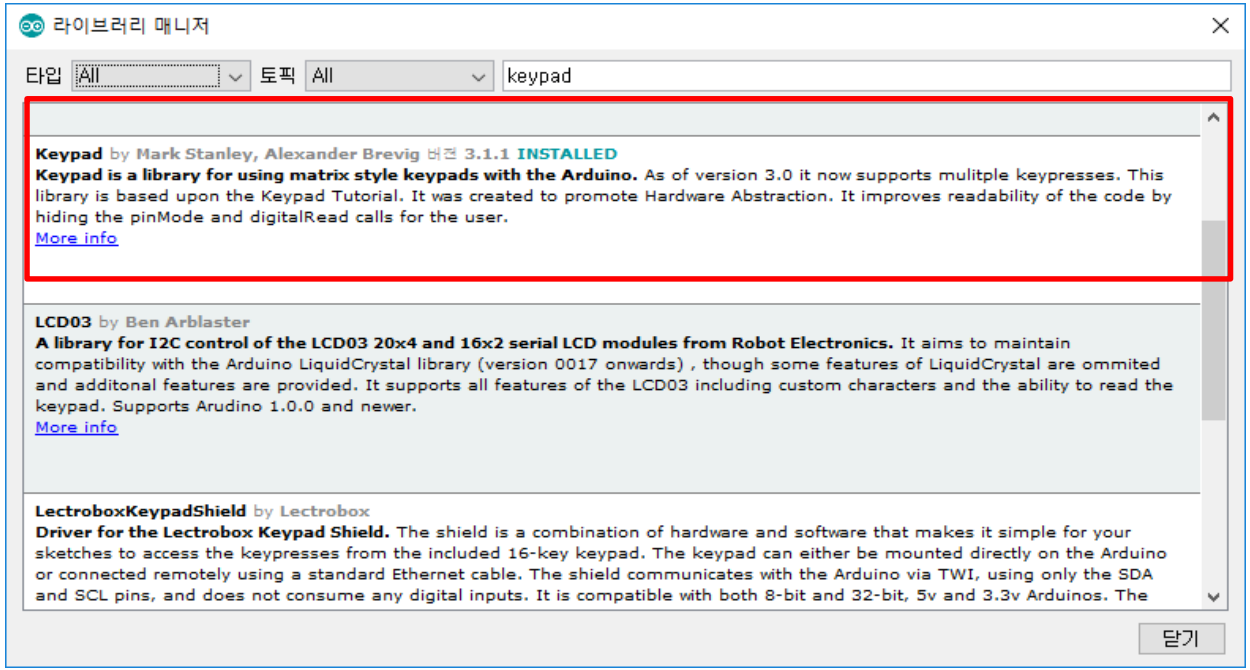
```
void loop( ) {
  KeyValue=KeyScan();
  if (KeyValue>=0){
    Serial.print("KEY=");
    Serial.print(KeyCode[KeyValue]);
    Serial.println(" Pressed");
  }
  delay(100);
}

int KeyScan( ){
  int nRow, nNumber=-1;
  for(int k=0; k<4; k++) {
    digitalWrite(KeyRow[k], LOW);
    nRow = k*4;
    for(int m=0; m<4; m++) {
      if ( !digitalRead(KeyCol[m]) ) nNumber=nRow+m;
    }
    digitalWrite(KeyRow[k], HIGH);
  }
  return nNumber;
}
```

KeyPad-3 : Serial Monitor



KeyPad-4 : KeyPad Library

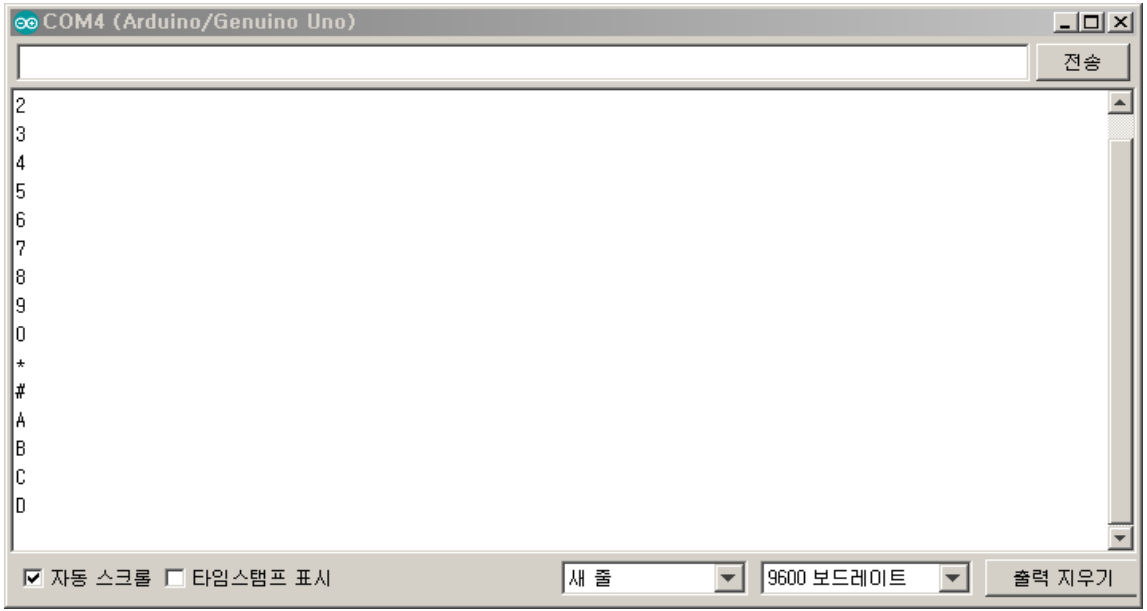
- [스케치] [라이브러리 포함하기] 라이브러리 관리]
- 라이브러리 관리 : keypad



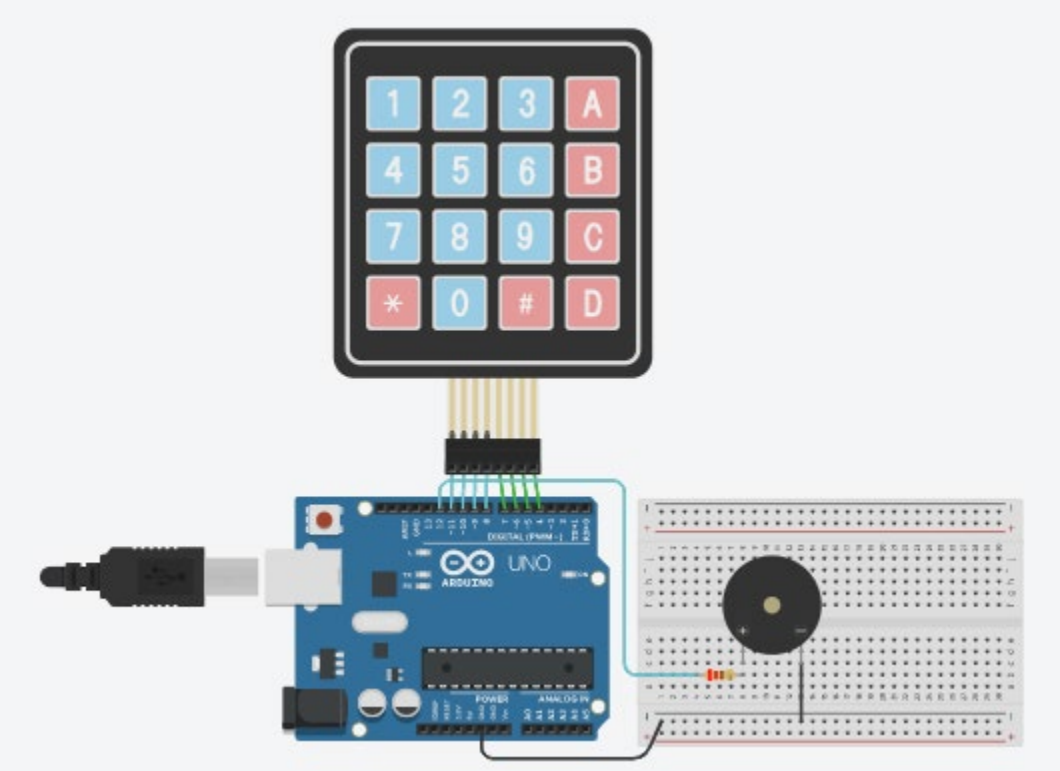
KeyPad-4 : Sketch

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
byte KeyRow[ROWS] = { 7,  6,  5,  4  };
byte KeyCol[COLS] =   { 8,  9, 10, 11 };
char KeyCode[ROWS][COLS]={
                { '1', '2', '3', 'A' },
                { '4', '5', '6', 'B' },
                { '7', '8', '9', 'C' },
                { '*', '0', '#', 'D' }
 };
Keypad keypads = Keypad( makeKeymap(KeyCode), KeyRow, KeyCol, ROWS, COLS);

void setup( ) {
  Serial.begin(9600);
}

void loop( ) {
  char key=keypads.getKey( );
  if (key){ Serial.println(key);
 }
}
```

KeyPad-4 : Serial Monitor



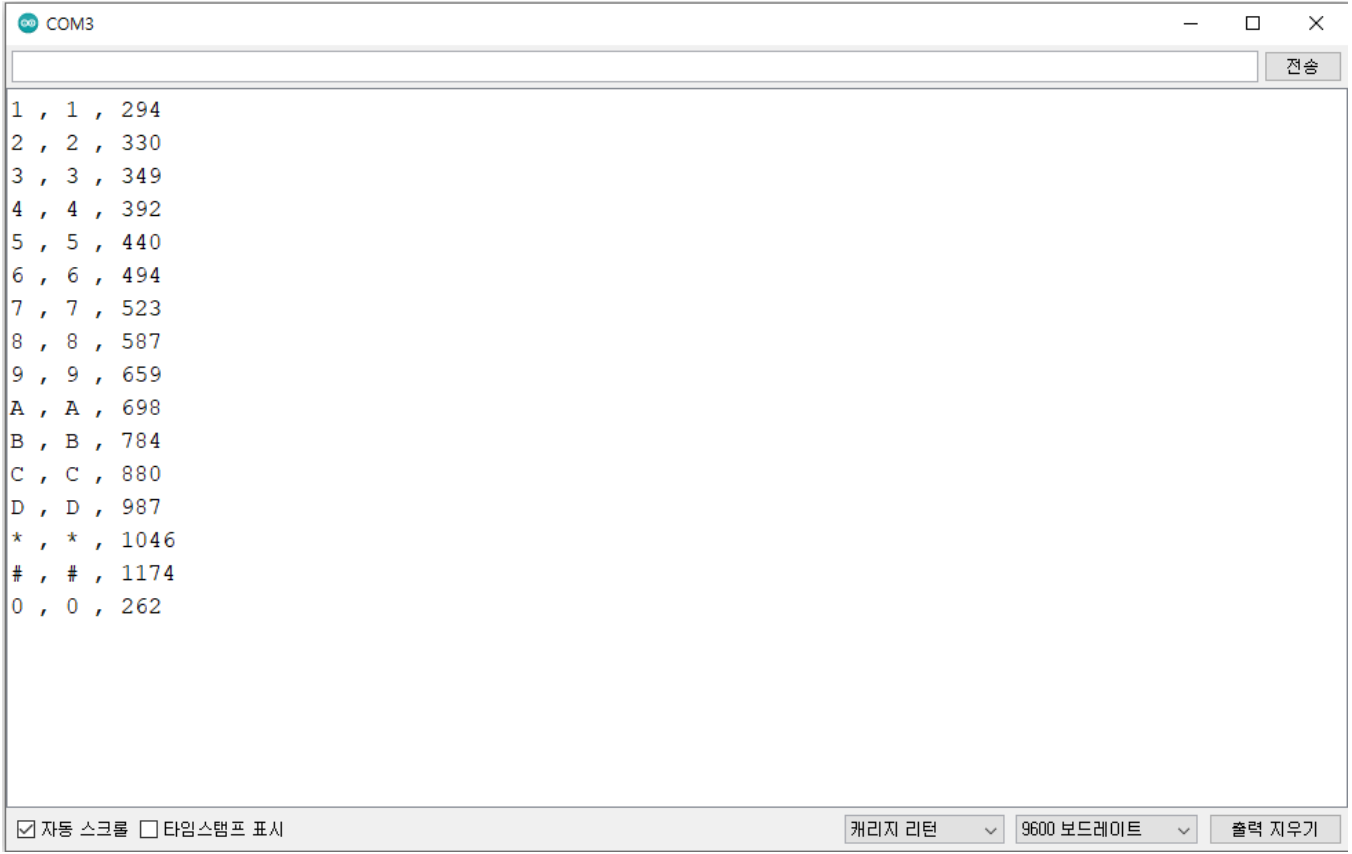
Keypad-5 : Keypad Sound



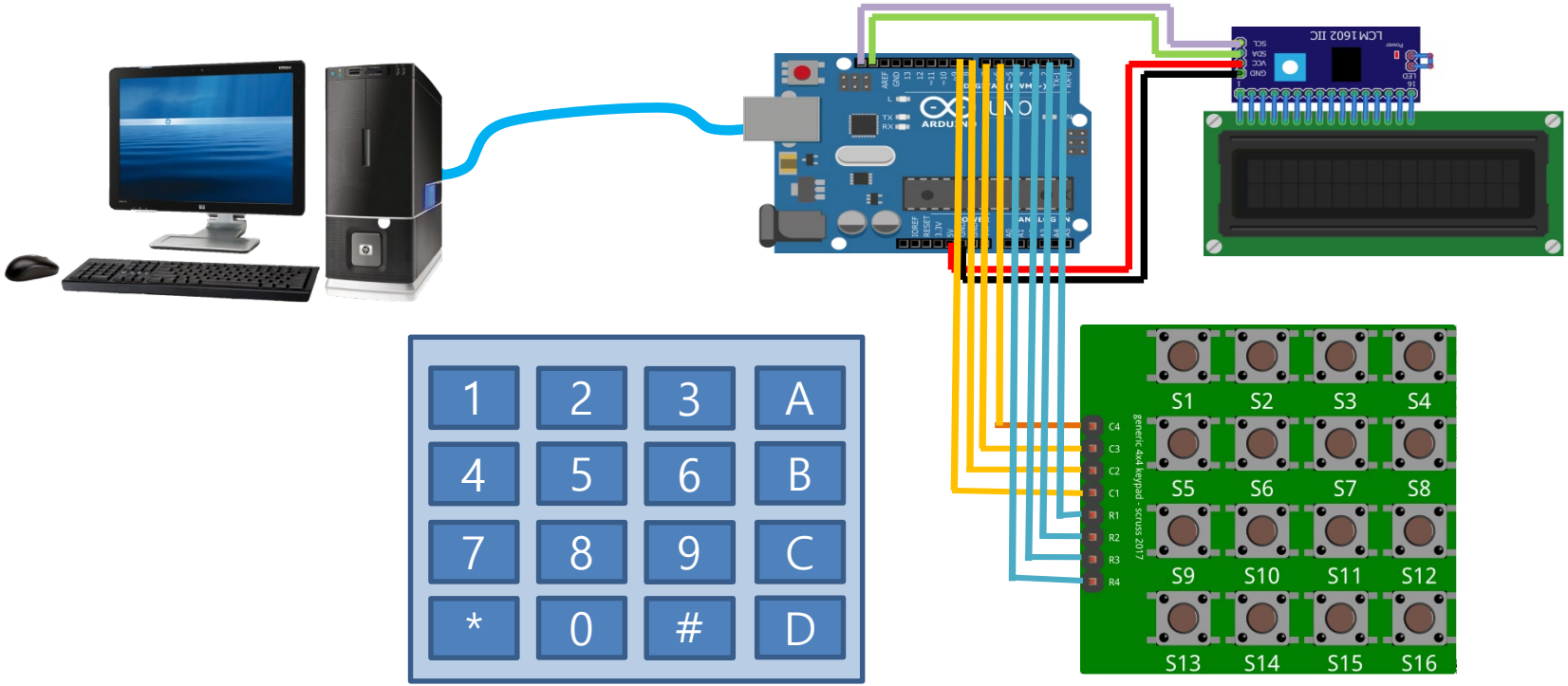
Keypad-5 : keypad Sound code

```
#include <Keypad.h>
const int BUZZER_PIN = 12;
const int ROW_NUM    = 4; // four rows
const int COLUMN_NUM = 4; // four columns

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'} };
int tones[]={
 262, 294, 330, 349, 392, 440, 494, 523,
 587, 659, 698, 784, 880, 987,1046,1174 };
byte pin_rows[ROW_NUM] = {7, 6, 5, 4};
byte pin_column[COLUMN_NUM] = {8, 9, 10, 11};

Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column,
ROW_NUM, COLUMN_NUM );

void setup() {
  Serial.begin(9600); }

void loop() {
  char key = keypad.getKey();
  if(key) {  Serial.print(key); // prints key to serial monitor
    sound_loop(key);  }
}
void sound_loop(char Key){
    Serial.print(Key);
    if(Key == '0'){PiezoTones(0); }
    if(Key == '1'){PiezoTones(1); }
    if(Key == '2'){PiezoTones(2); }
    if(Key == '3'){PiezoTones(3); }
    if(Key == '4'){PiezoTones(4); }
    if(Key == '5'){PiezoTones(5); }
    if(Key == '6'){PiezoTones(6); }
    if(Key == '7'){PiezoTones(7); }
    if(Key == '8'){PiezoTones(8); }
    if(Key == '9'){PiezoTones(9); }
    if(Key == 'A'){PiezoTones(10); }
    if(Key == 'B'){PiezoTones(11); }
    if(Key == 'C'){PiezoTones(12); }
    if(Key == 'D'){PiezoTones(13); }
    if(Key == '*'){PiezoTones(14); }
    if(Key == '#'){PiezoTones(15); }
    }

void PiezoTones(int t){
    tone(BUZZER_PIN,tones[t]);
    Serial.println(tones[t]);
    delay(200);
    noTone(BUZZER_PIN);
}
```

Result


COM3
전송
1 , 1 , 294
2 , 2 , 330
3 , 3 , 349
4 , 4 , 392
5 , 5 , 440
6 , 6 , 494
7 , 7 , 523
8 , 8 , 587
9 , 9 , 659
A , A , 698
B , B , 784
C , C , 880
D , D , 987
* , * , 1046
# , # , 1174
0 , 0 , 262
자동 스크롤
타임스탬프 표시
캐리지 리턴
9600 보드레이트
출력 지우기

Keypad-I2C LCD



Keypad-I2C LCD code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>

#define Password_Length 8
int signalPin = 12;
char Data[Password_Length];
char Master[Password_Length] = "123A456";
byte data_count = 0, master_count = 0;
bool Pass_is_good;
char customKey;
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'} };
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins,
colPins, ROWS, COLS);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void Setup( ){
lcd.init(); lcd.backlight(); pinMode(signalPin, OUTPUT); }

void loop( ){
 lcd.setCursor(0,0);
 lcd.print("Enter Password:");
 customKey = customKeypad.getKey();
if (customKey){
     Data[data_count] = customKey;
     lcd.setCursor(data_count,1);
     lcd.print(Data[data_count]);
     data_count++; }
  if(data_count == Password_Length-1){
      lcd.clear();
   if(!strcmp(Data, Master)) {
      lcd.print("Correct");
      digitalWrite(signalPin, HIGH);
      delay(5000);
      digitalWrite(signalPin, LOW); }
    else{   lcd.print("Incorrect"); delay(1000); }
     lcd.clear(); clearData(); }
}

void clearData( ){
   while(data_count !=0){ Data[data_count--] = 0; }
       return;
}
```

Result