

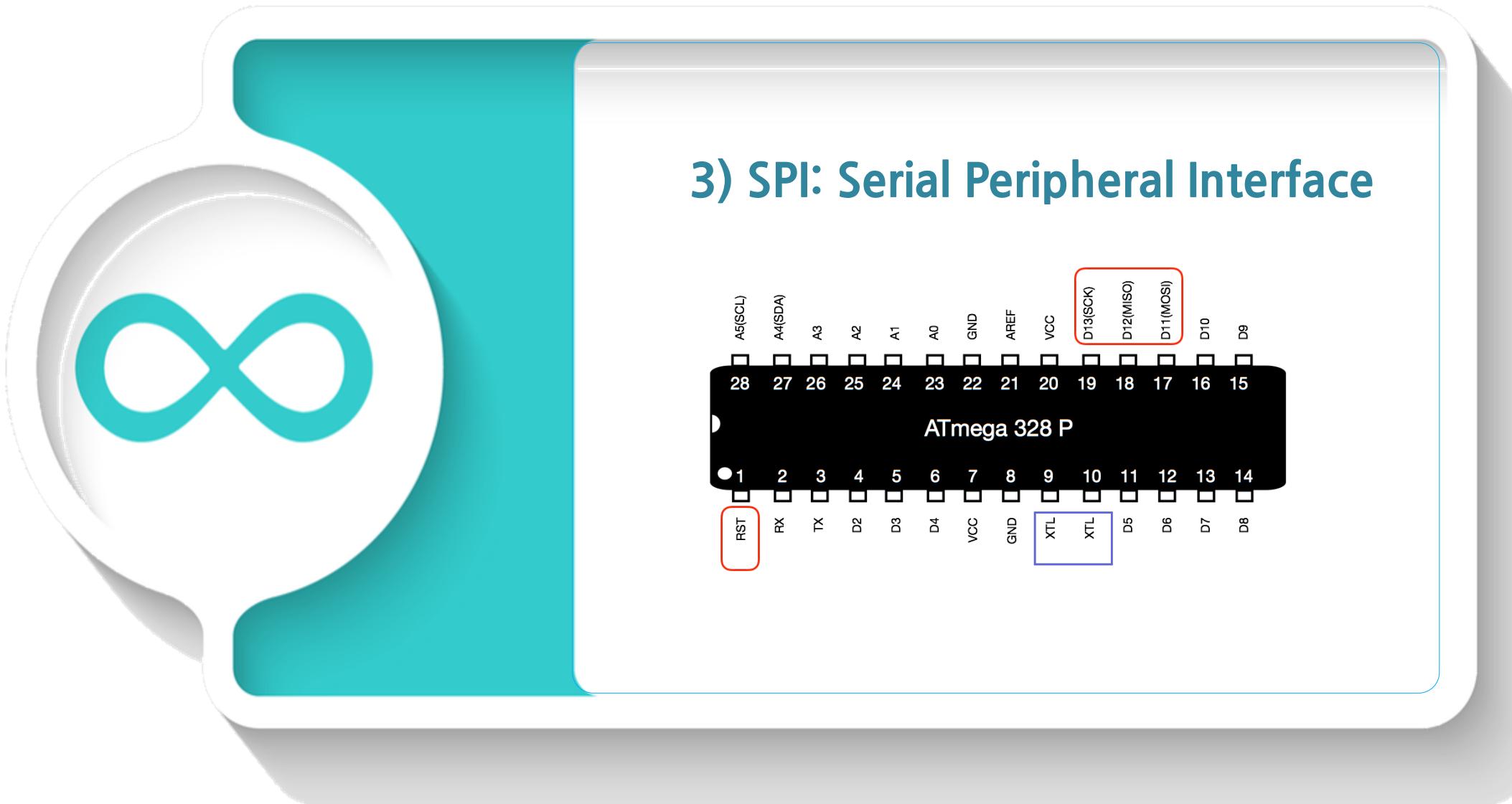
# Ch7. Serial interface, IR

양 세훈



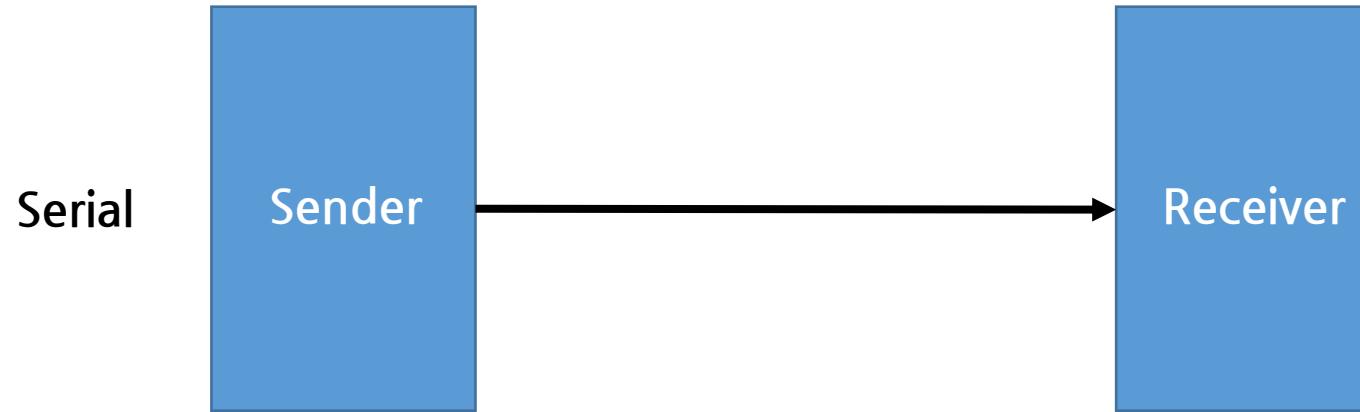
## Ch7. Serial Interface

- 1) UART
- 2) I2C
  - 2-1) I2C : Between Arduino
  - 2-2) TC74XX I2CTemp sensor
  - 2-3) I2C-LCD



### 3) SPI: Serial Peripheral Interface

## Serial vs Parallel : 장단점



# 1) UART

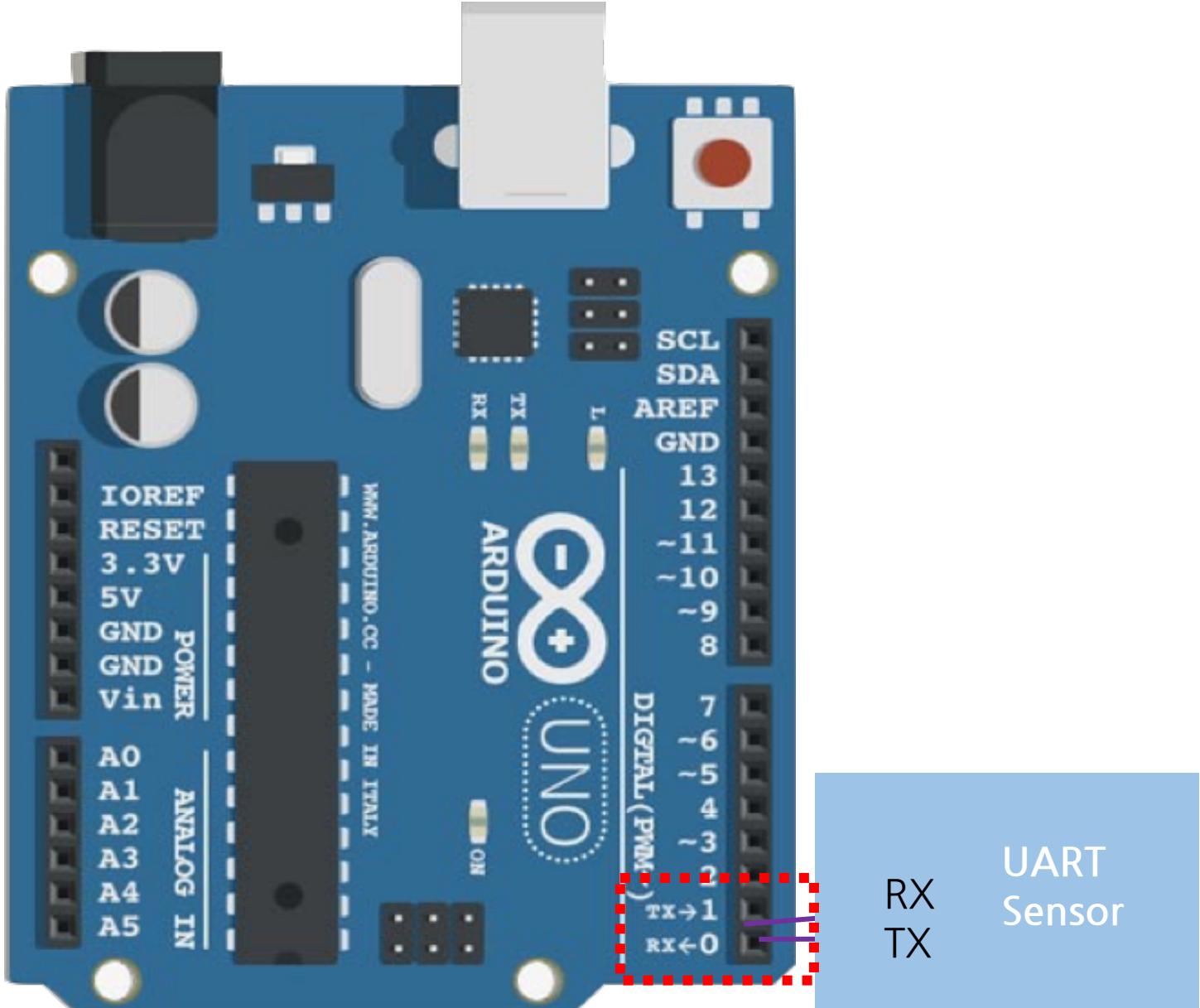
UART: 범용 비동기식 송수신

Universal Asynchronous  
Receive/Transmit

대표적 통신 방법( 1:1 )

0 번 (RX) 핀으로 받고.  
1 번 (TX) 로.

시작	0	1	2	3	4	5	6	7	정지
----	---	---	---	---	---	---	---	---	----



## 2) I2C

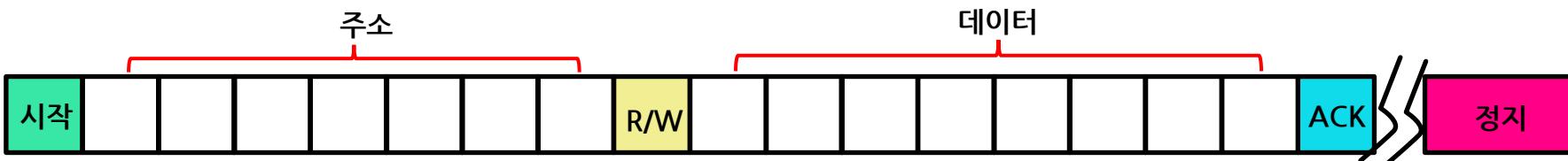
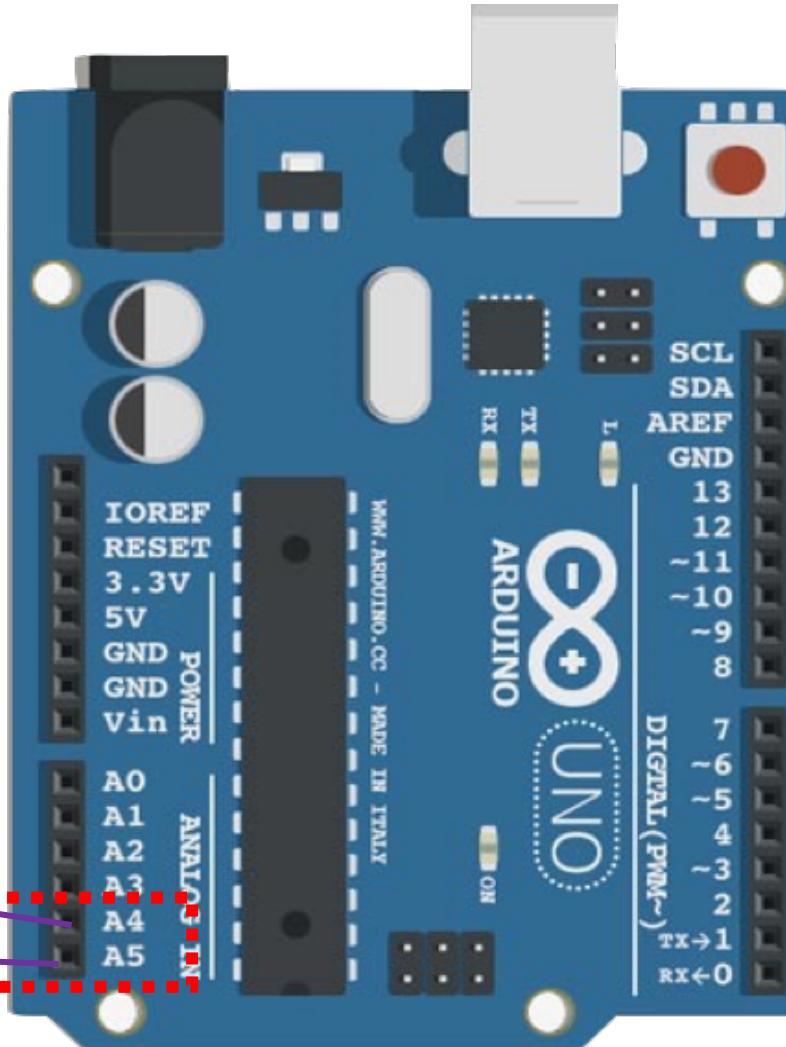
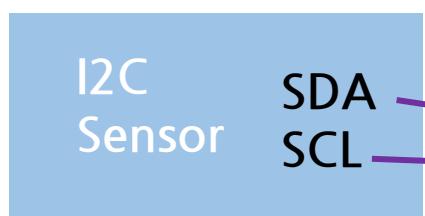
I2C: 일명 TWI

Inter Integrated Circuit

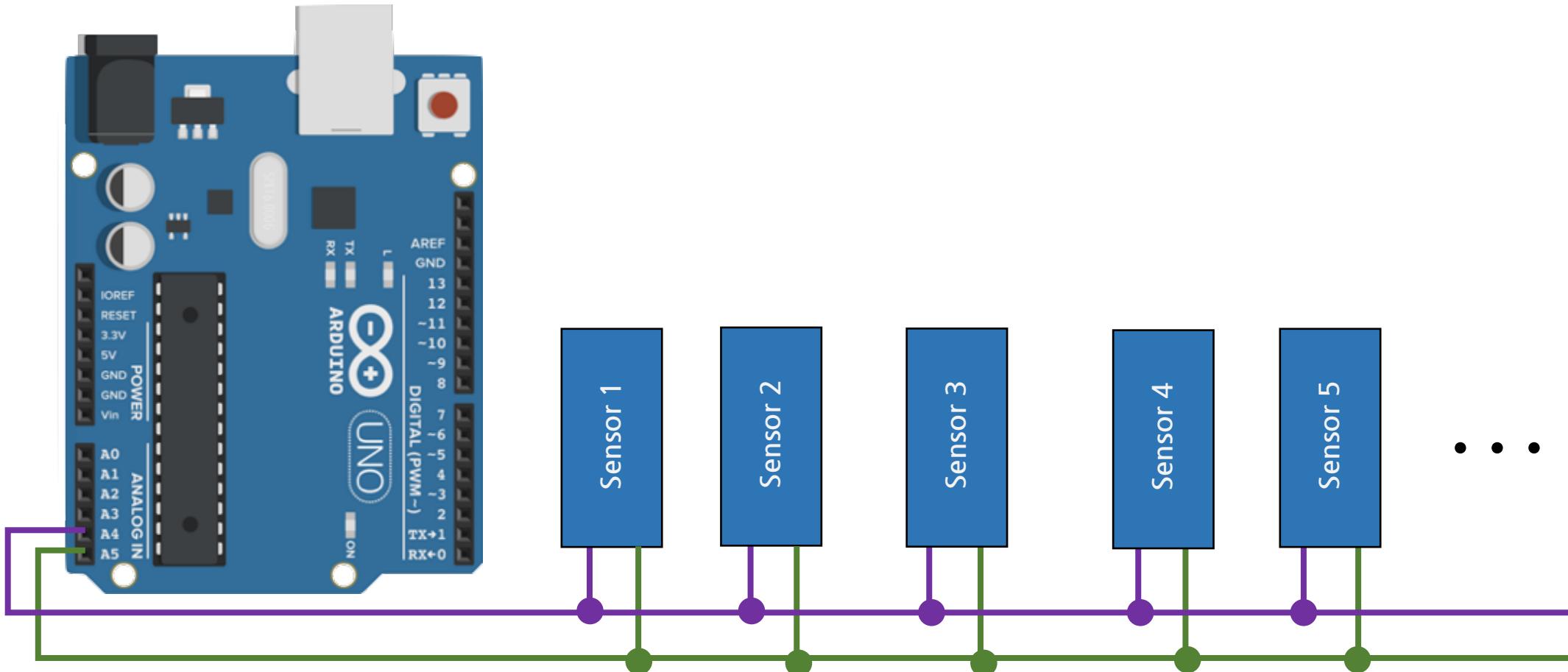
여러개 센서와 동시에 통신( 1:N )

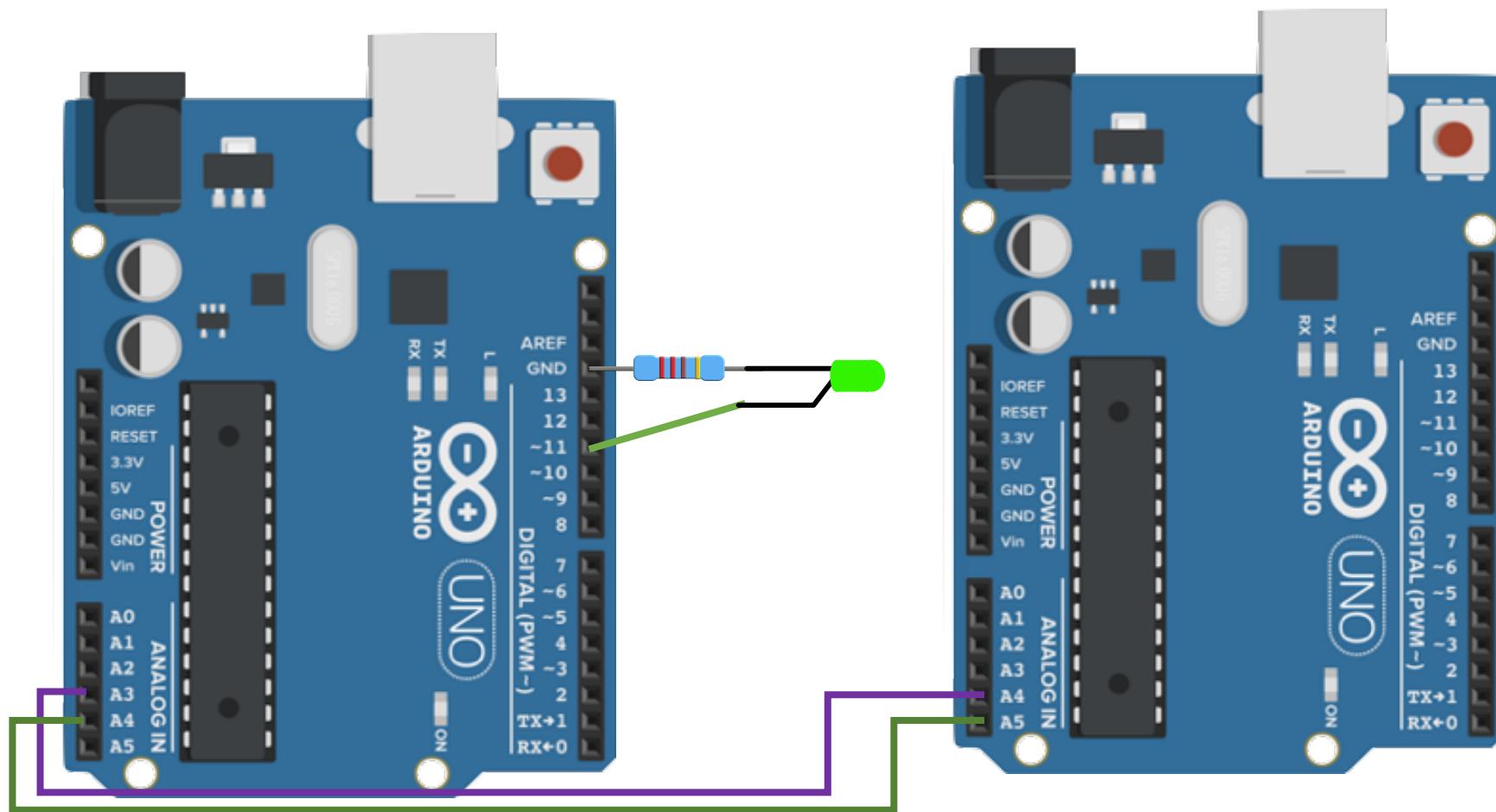
A4 핀 SDA

A5 핀 SCL



# I2C Sensor making





## 2-1) I2C : Between Arduino

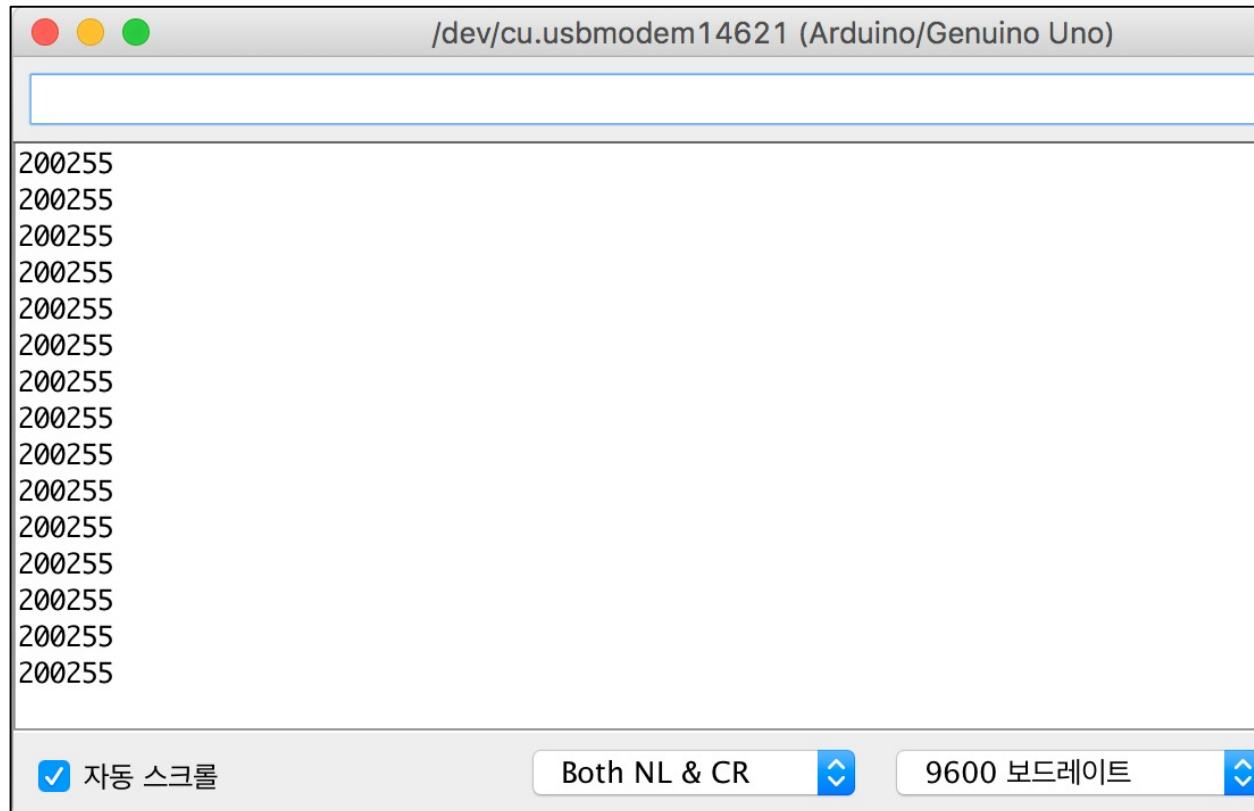
The screenshot shows the Arduino IDE interface with the title bar "I2C\_2\_Way\_Server". The code is as follows:

```
1 // I2C_2_Way_Server :
2 // Send big number and Receive 'H' or 'L'
3
4 #include <Wire.h>
5
6 void setup() {
7     Wire.begin(0);
8     Serial.begin(9600) ;
9     Wire.onReceive(Action);
10    pinMode(11,OUTPUT);
11 }
12
13 void loop() {
14     Wire.beginTransmission(1);
15     Wire.write(200);
16
17     Wire.write(255);
18     Wire.endTransmission();
19     delay(200);
20 }
21
22 void Action() {
23     char letter;
24
25     while(Wire.available() > 0) {
26         letter = Wire.read();
27         Serial.println(letter) ;
28         if (letter == 'H')
29         {
30             digitalWrite(11, HIGH);
31         }
32         else if (letter == 'L')
33         {
34             digitalWrite(11, LOW);
35         }
36     }
37 }
```

The screenshot shows the Arduino IDE interface with the title bar "I2C\_2\_Way\_Slave". The code is as follows:

```
1 // I2c_2_Way_Slave :
2 // Send 'H' or 'L' and Receive Big number
3
4 #include <Wire.h>
5
6 void setup() {
7     Wire.begin(1);
8     Serial.begin(9600) ;
9     Wire.onReceive(Action);
10    pinMode(11,OUTPUT);
11 }
12
13 void loop() {
14     Wire.beginTransmission(0);
15     Wire.write('H');
16     Wire.endTransmission();
17
18     delay(200);
20
21     Wire.beginTransmission(0);
22     Wire.write('L');
23     Wire.endTransmission();
24
25     delay(200);
26 }
27
28 void Action() {
29     int num1 ;
30     int num2 ;
31
32     if(Wire.available() <= 2)  {
33         num1 = Wire.read();
34         num2 = Wire.read();
35     }
36     Serial.print(num1) ;
37     Serial.println(num2) ;
38 }
```

## 2-1) I2C : Between Arduino



## 2-1) I2C :

### Between Arduino

Using onRequest

The screenshot shows the Arduino IDE interface with two tabs open. The left tab is titled "I2C\_2\_WAY\_Server\_m1" and contains the server-side code. The right tab is titled "I2C\_2\_Way\_Slave\_m1" and contains the slave-side code.

```
1 // UNO_1 : address=1
2 #include <Wire.h>
3
4 int LED=11 ;
5 char x = 0;
6
7 void setup() {
8 Wire.begin(1);
9 pinMode(LED,OUTPUT);
10 Serial.begin(9600) ;
11 Wire.onReceive(read_Character) ; // to receive from 5
12 Wire.onRequest(requestEvent_5) ;
13 delay(500) ;
14 }
15
16 =====
17 void read_Character() {
18 x = Wire.read(); // read one character from the I2C
19 Serial.println(x) ;
20 }
21
22 void loop() {
23 LED_Control() ; // Receive from 5 then control LED
24 delay(300) ;
25 }
26
27 =====
28 void requestEvent_5() {
29 Wire.write("12345678") ;
30 }
31
32 =====
33 void LED_Control() {
34 if (x == 'H') {
35 digitalWrite(LED, HIGH);
36 }
37
38 else if (x == 'L') {
39 digitalWrite(LED, LOW);
40 }
41 }
42
```

The screenshot shows the Arduino IDE interface with two tabs open. The left tab is titled "I2C\_2\_WAY\_Server\_m1" and contains the server-side code. The right tab is titled "I2C\_2\_Way\_Slave\_m1" and contains the slave-side code.

```
1 // Server address=5
2 #include <Wire.h>
3
4 void setup() {
5 Wire.begin(5);
6 Serial.begin(9600) ;
7 }
8
9 void loop() {
10 request_1() ; // request data from UNO_1
11
12 Send_to_1() ; // send to UNO_1
13 }
14
15 =====
16 void request_1() {
17 Wire.requestFrom(1,5) ;
18
19 while(Wire.available()) {
20 char c =Wire.read() ;
21 Serial.print(c) ;
22 }
23 Serial.println("") ;
24 }
25
26 =====
27
28 void Send_to_1() {
29 Wire.beginTransmission(1);
30 Wire.write('H');
31 Wire.endTransmission();
32 delay(500) ;
33 Serial.println("To UNO_1 ");
34
35 Wire.beginTransmission(1);
36 Wire.write('L');
37 Wire.endTransmission();
38 delay(500) ;
39 }
40
```

## 2-1) I2C : Between Arduino

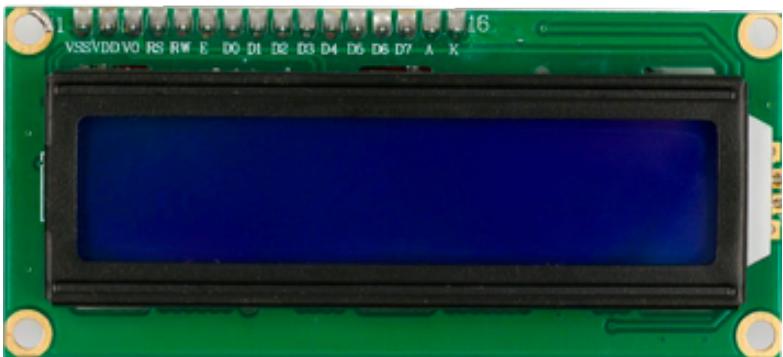


The screenshot shows a terminal window titled "/dev/cu.usbmodem14621 (Arduino/Genuino Uno)". The window displays a series of identical messages, each consisting of "12345" followed by "To UNO\_1". There are ten such messages stacked vertically. The terminal interface includes standard Mac OS X window controls (red, yellow, green buttons) at the top left. At the bottom, there is a checkbox labeled "자동 스크롤" (Auto Scroll) with a checked mark, and a dropdown menu labeled "Both NL & CR" with up and down arrows.

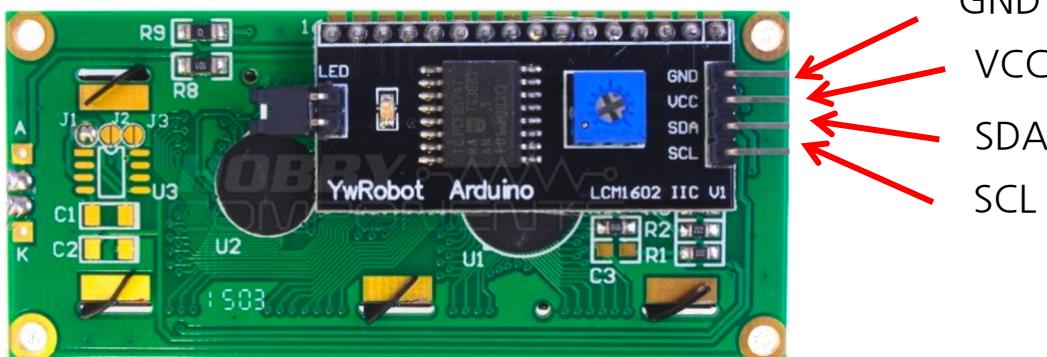
```
12345
To UNO_1
12345
```

## 2-3) I2C LCD

I2C LCD :16X2



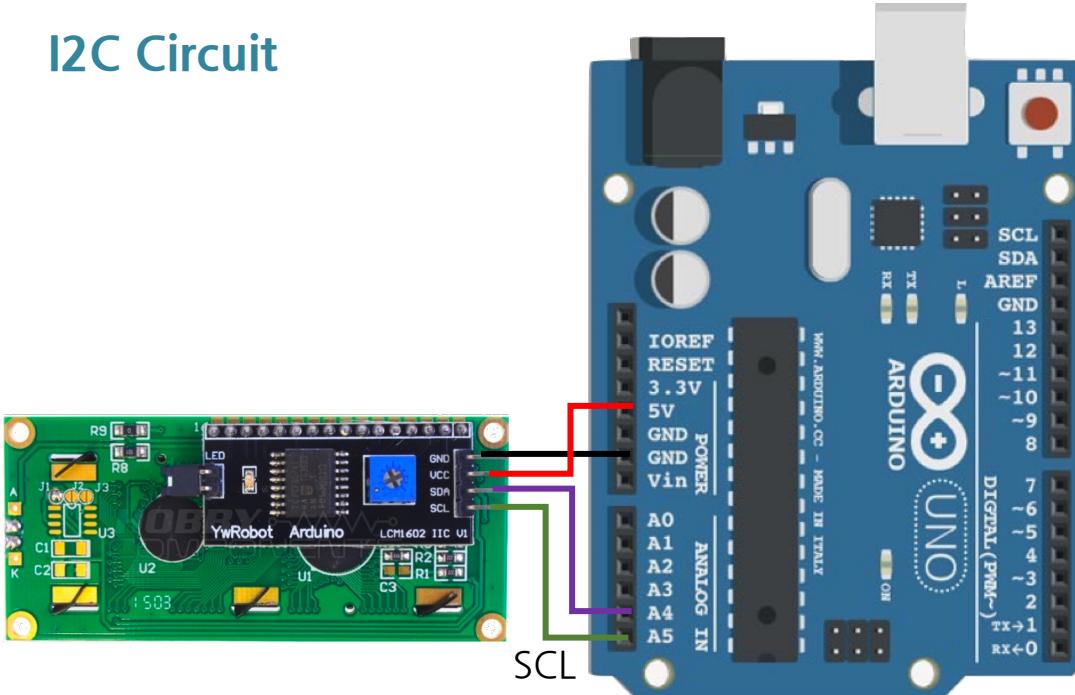
Front view



Rear view

## 2-3) I2C LCD

I2C Circuit



라이브러리 매니저

타입 All 토픽 All liquid crystal i2c

**LiquidCrystal I2C**

by Marco Schwartz

**A library for I2C LCD displays.** The library allows to control I2C displays with functions extremely similar to LiquidCrystal library. THIS LIBRARY MIGHT NOT BE COMPATIBLE WITH EXISTING SKETCHES.

[More info](#)

버전 1.1.2 설치

**LiquidCrystal\_AIP31068**

by Andriy Golovnya

**A library for AIP31068 I2C/SPI LCD displays.** The library allows to control AIP31068 based I2C/SPI displays with functions extremely similar to LiquidCrystal library. THIS LIBRARY MIGHT NOT BE COMPATIBLE WITH EXISTING SKETCHES.

[More info](#)

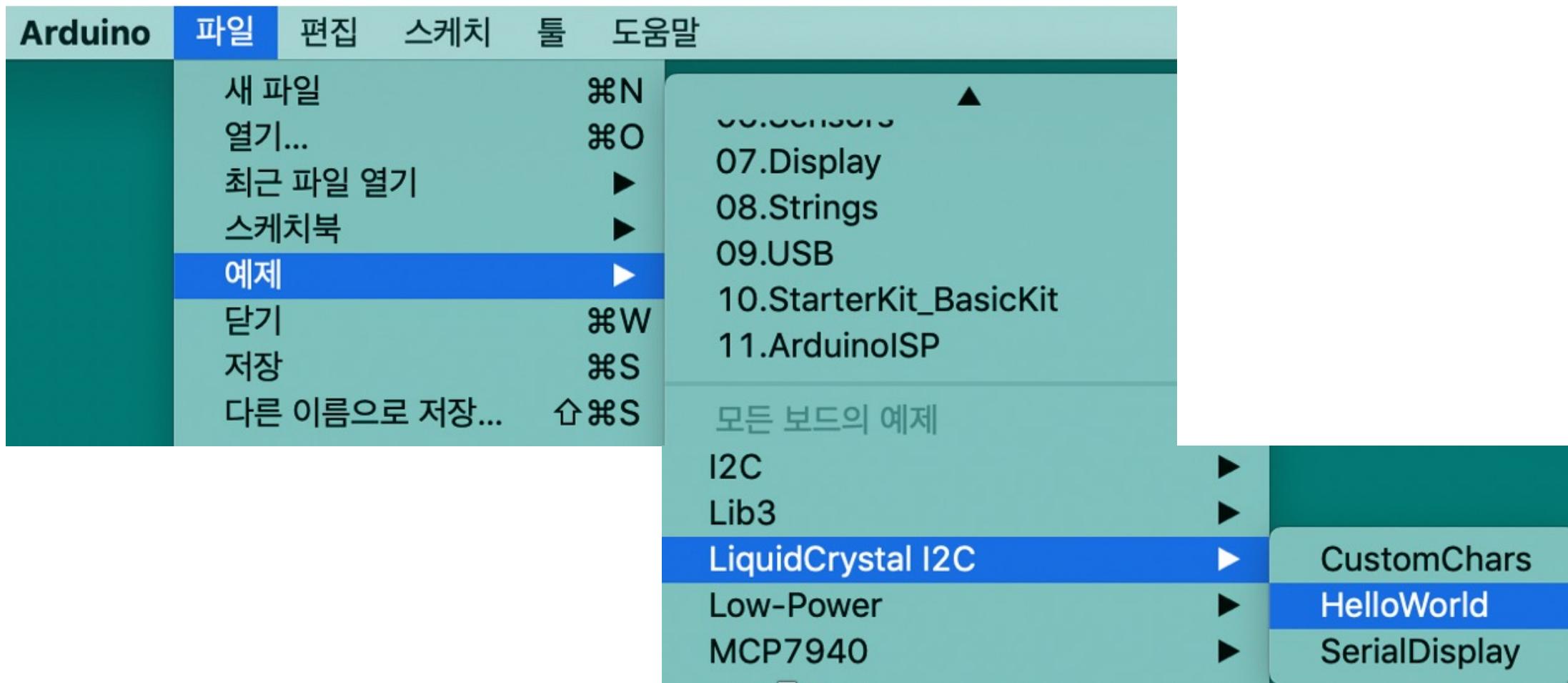
**LiquidCrystal\_I2C\_Hangul**

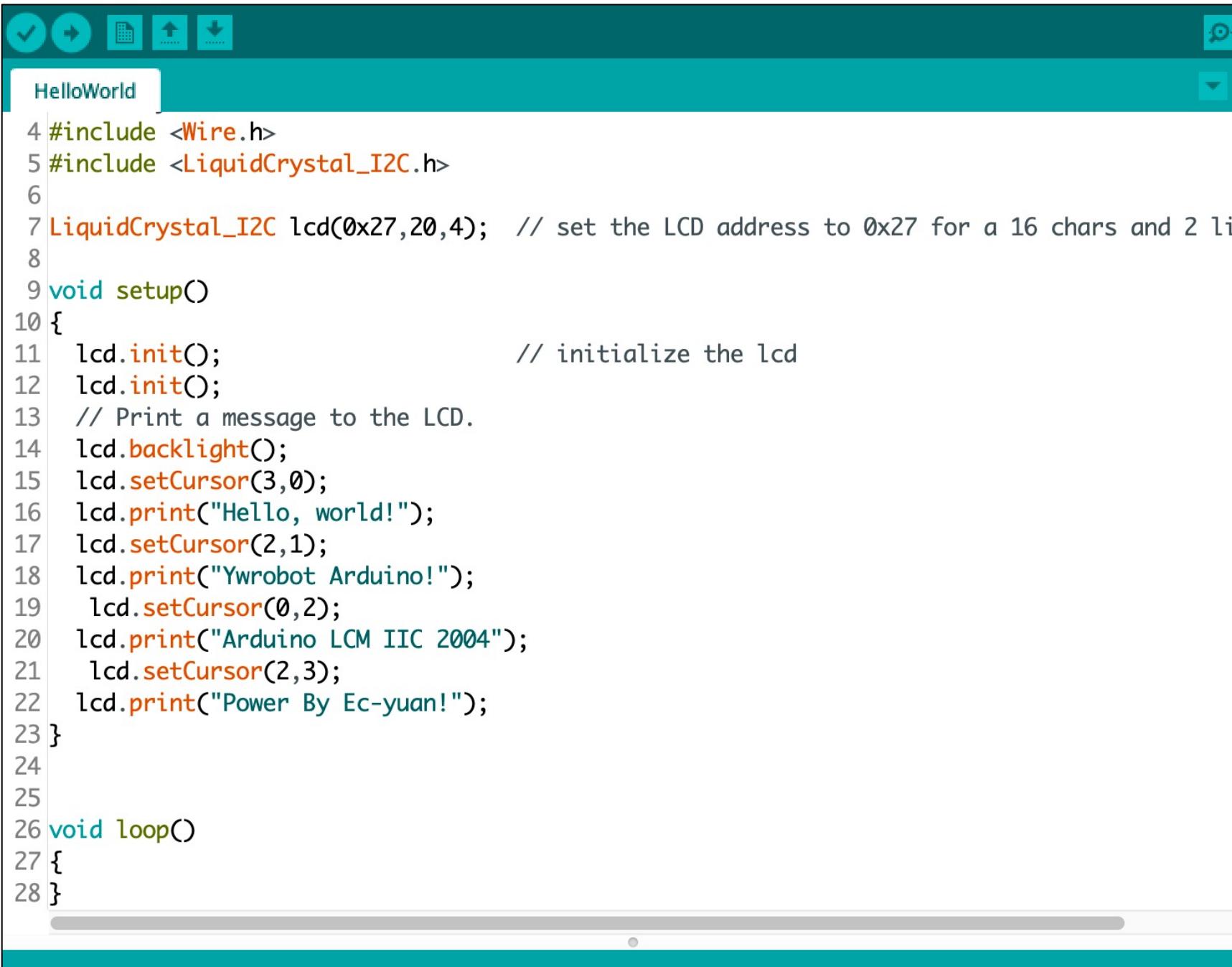
by Junwha Hong

**A library for printing Hangul on I2C LCD displays.** The library allows to control I2C displays with functions extremely similar to LiquidCrystal library. This Library allows to print hangul on LCDs.

[More info](#)

닫기





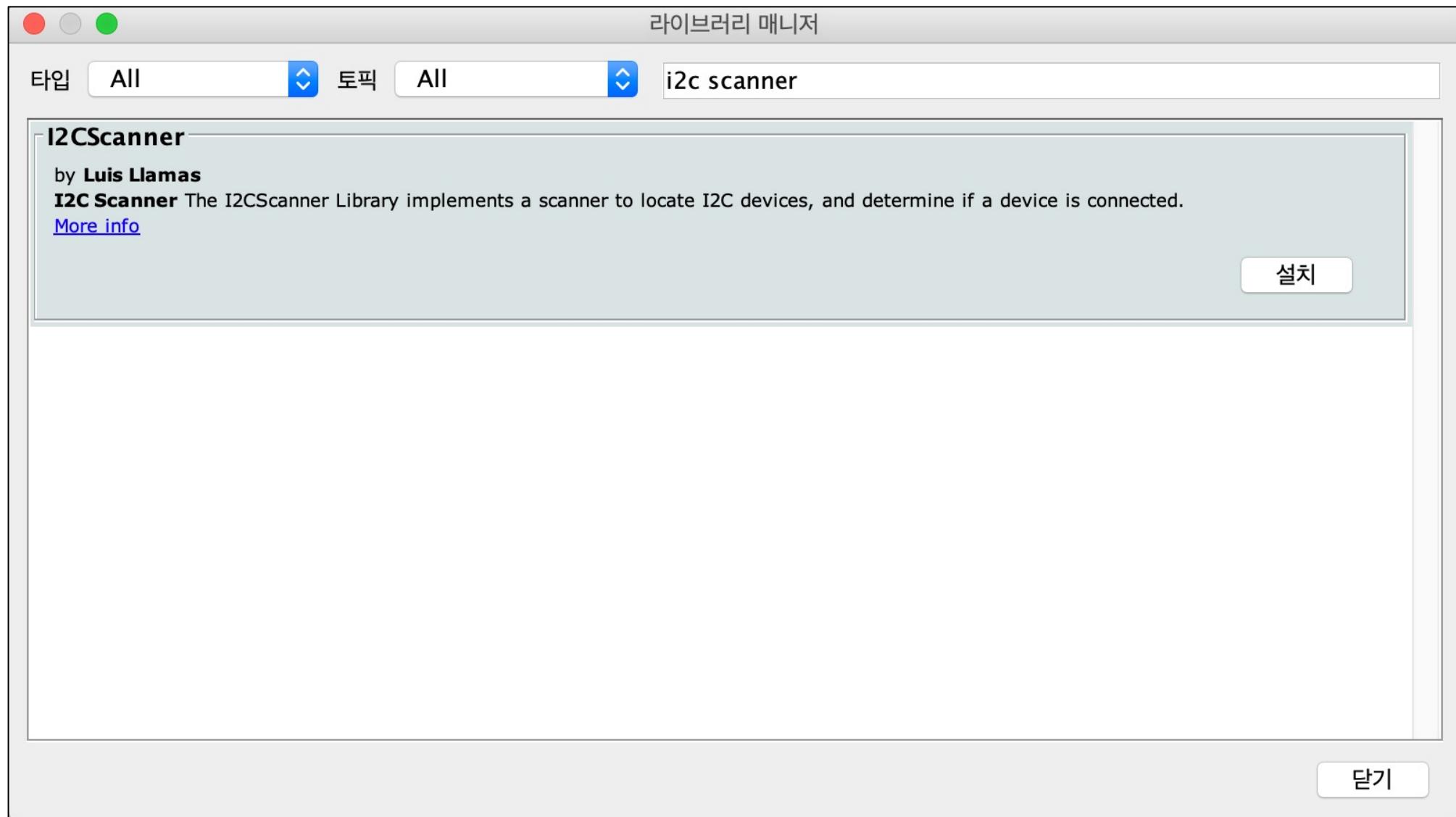
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** The title bar displays the project name "HelloWorld".
- Tool Buttons:** Standard Arduino tool buttons for file operations (New, Open, Save, Print, Copy, Paste) are located at the top left.
- Sketch Area:** The main area contains the C++ code for the "HelloWorld" sketch. The code includes #includes for `Wire.h` and `LiquidCrystal_I2C.h`, initializes an LCD at address 0x27, and prints multiple messages to the screen using the `LiquidCrystal_I2C` library.
- Code Content:** The code is as follows:

```
4 #include <Wire.h>
5 #include <LiquidCrystal_I2C.h>
6
7 LiquidCrystal_I2C lcd(0x27,20,4); // set the LCD address to 0x27 for a 16 chars and 2 lines
8
9 void setup()
10 {
11     lcd.init(); // initialize the lcd
12     lcd.init();
13     // Print a message to the LCD.
14     lcd.backlight();
15     lcd.setCursor(3,0);
16     lcd.print("Hello, world!");
17     lcd.setCursor(2,1);
18     lcd.print("Ywrobot Arduino!");
19     lcd.setCursor(0,2);
20     lcd.print("Arduino LCM IIC 2004");
21     lcd.setCursor(2,3);
22     lcd.print("Power By Ec-yuan!");
23 }
24
25
26 void loop()
27 {
28 }
```

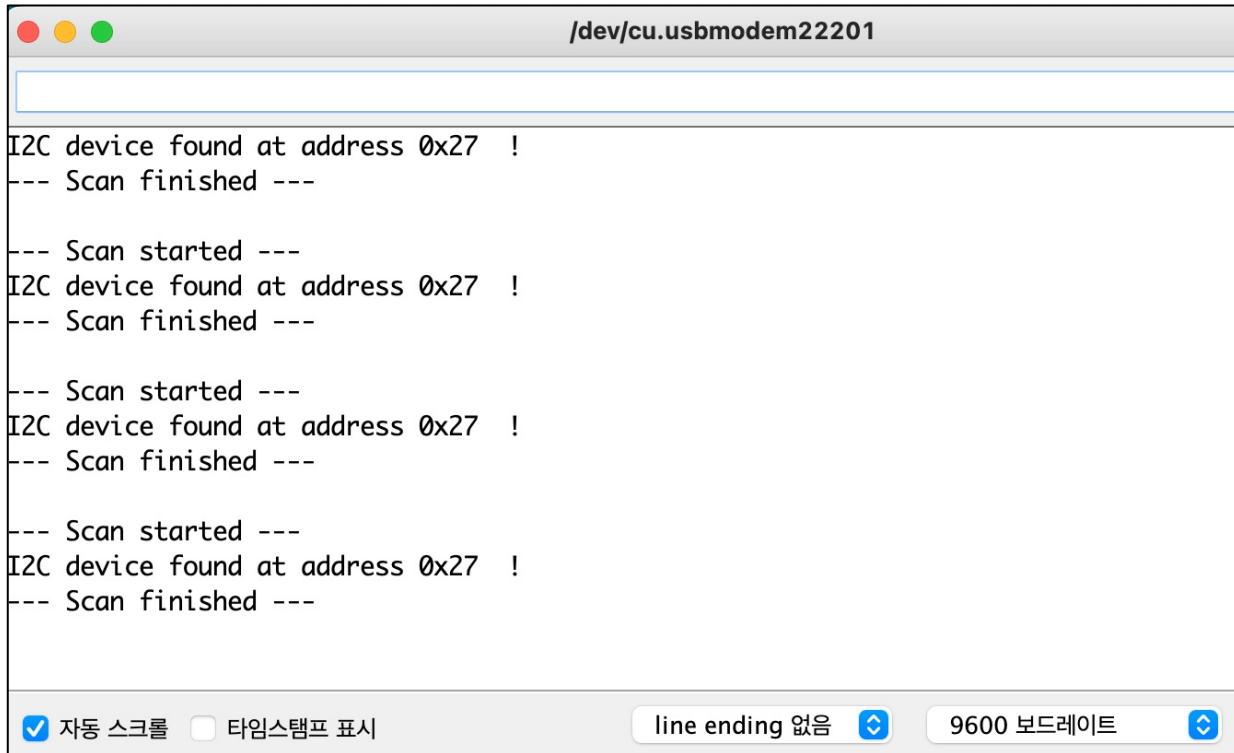
## How to find 0x27

<https://github.com/luisllamasbinaburo/Arduino-I2CScanner/tree/master/examples/Execute>



The screenshot shows a microcontroller development environment with a teal-themed interface. At the top, there are several icons: a checkmark, a circular arrow, a file folder, an upload arrow, a download arrow, and a gear. Below the toolbar is a tab bar with the title "Scanner". The main area displays a C++ code editor with the following content:

```
6 ****
7
8 #include "I2CScanner.h"
9
10 I2CScanner scanner;
11
12 void setup()
13 {
14     Serial.begin(9600);
15     while (!Serial) {};
16
17     scanner.Init();
18 }
19
20 void loop()
21 {
22     scanner.Scan();
23     delay(5000);
24 }
```



The screenshot shows a terminal window titled '/dev/cu.usbmodem22201'. The window contains four identical I2C scan logs, each consisting of two lines: 'I2C device found at address 0x27 !' and '--- Scan finished ---'. The logs are vertically stacked. At the bottom of the window, there are several configuration options: a checked checkbox for '자동 스크롤' (Auto Scroll), an unchecked checkbox for '타임스탬프 표시' (Timestamp Display), a dropdown menu for 'line ending' set to '없음' (None), and a dropdown menu for '9600 보드레이트' (9600 Baud Rate).

```
I2C device found at address 0x27 !
--- Scan finished ---

--- Scan started ---
I2C device found at address 0x27 !
--- Scan finished ---

--- Scan started ---
I2C device found at address 0x27 !
--- Scan finished ---

--- Scan started ---
I2C device found at address 0x27 !
--- Scan finished ---
```

자동 스크롤  타임스탬프 표시 line ending 없음 9600 보드레이트



## Execute

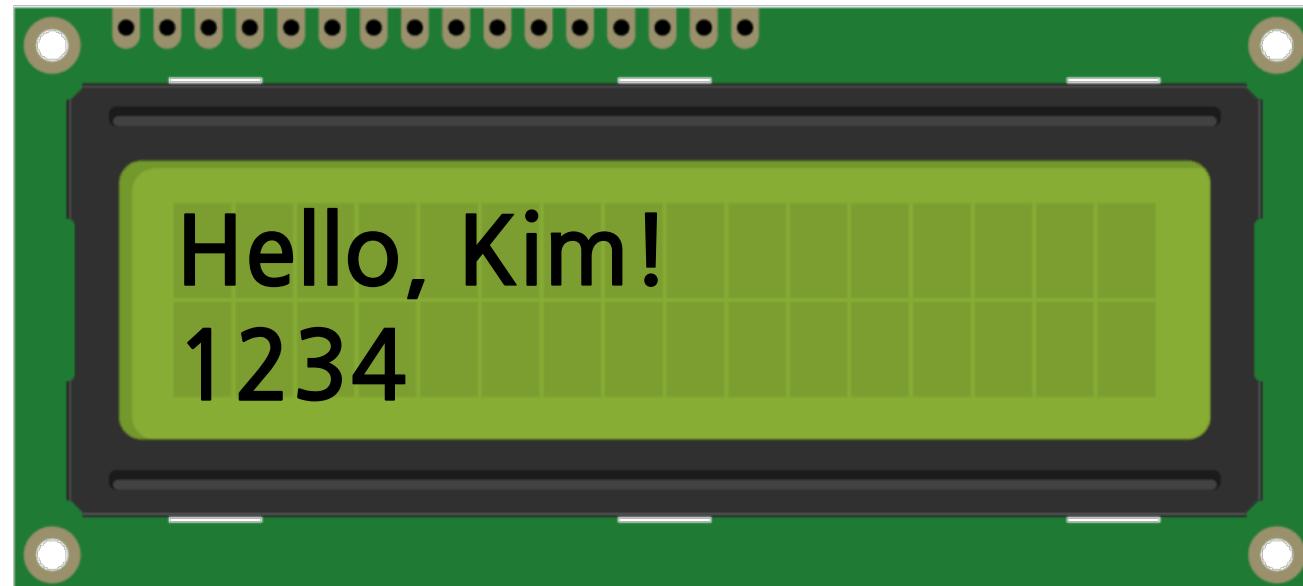
```
1 #include "I2CScanner.h"
2
3 I2CScanner scanner;
4
5 const byte address;
6
7 void debug(byte address)
8 {
9     Serial.print("Found at 0x");
10    Serial.println(address, HEX);
11 }
12
13 void setup()
14 {
15     Serial.begin(9600);
16     while (!Serial) {};
17
18     scanner.Init();
19 }
20
21 void loop()
22 {
23     scanner.Execute(debug);
24     delay(5000);
25 }
```

```
Found at 0x27
Found at 0x27
Found at 0x27
Found at 0x27
```

자동 스크롤  타임스탬프 표시

Project :

Make first line your name and second line second counter



# LiquidCrystal Library

This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

## Examples

- [Autoscroll](#): Shift text right and left.
- [Blink](#): Control of the block-style cursor.
- [Cursor](#): Control of the underscore-style cursor.
- [Display](#): Quickly blank the display without losing what's on it.
- [Hello World](#): Displays "hello world!" and the seconds since reset.
- [Scroll](#): Scroll text left and right.
- [Serial Display](#): Accepts serial input, displays it.

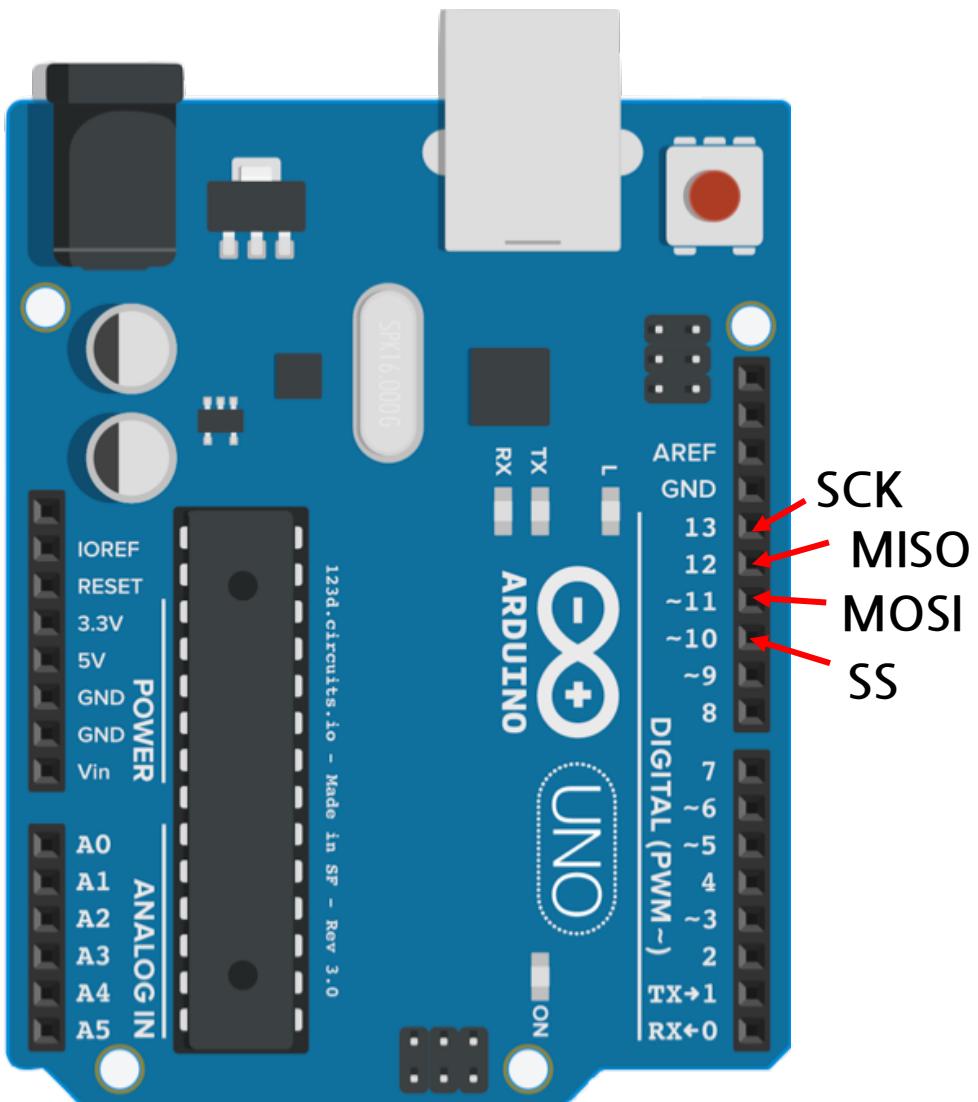
### Function

- [LiquidCrystal\(\)](#)
- [begin\(\)](#)
- [clear\(\)](#)
- [home\(\)](#)
- [setCursor\(\)](#)
- [write\(\)](#)
- [print\(\)](#)
- [cursor\(\)](#)
- [noCursor\(\)](#)
- [blink\(\)](#)
- [noBlink\(\)](#)
- [display\(\)](#)

### 3) SPI

아두이노 와 여러개 센서 사이 **고속** 통신 방법( 1:N )

SPI: Serial Peripheral Interface



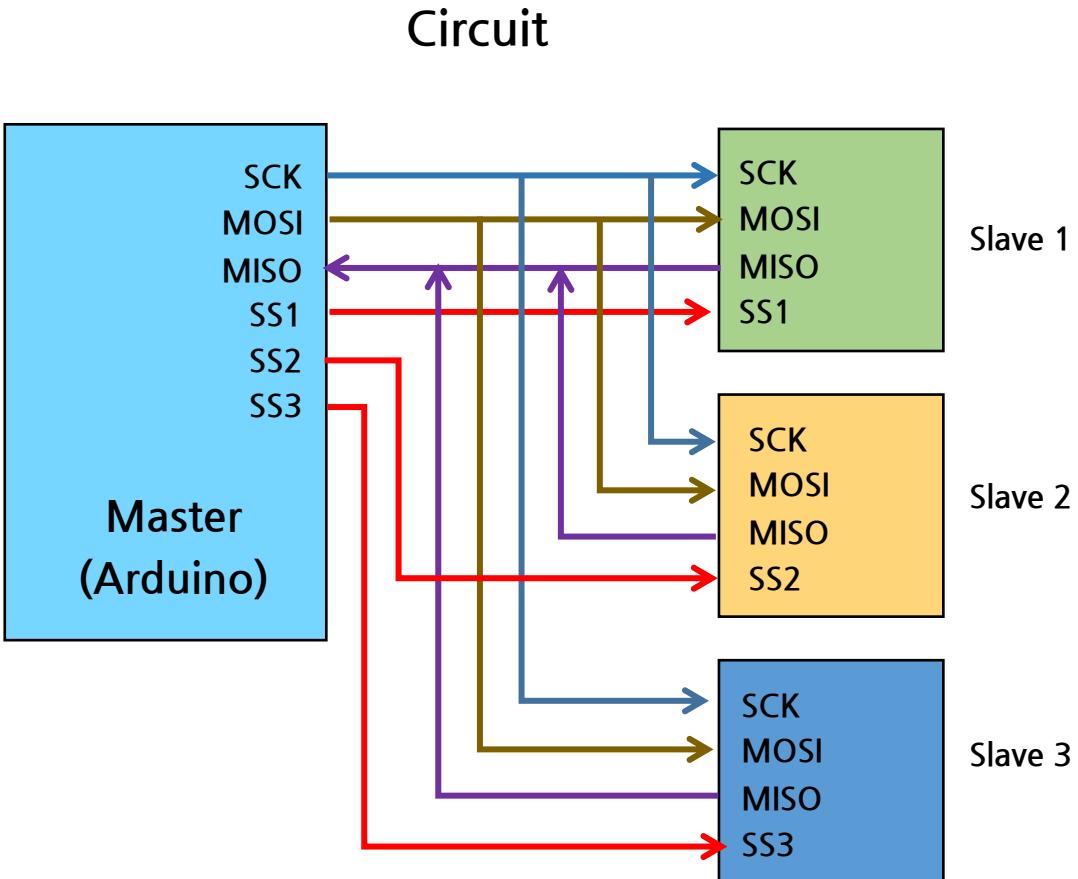
SCK(System ClocK)

MISO(Master In Slave Out)

MOSI(Master Out Slave In)

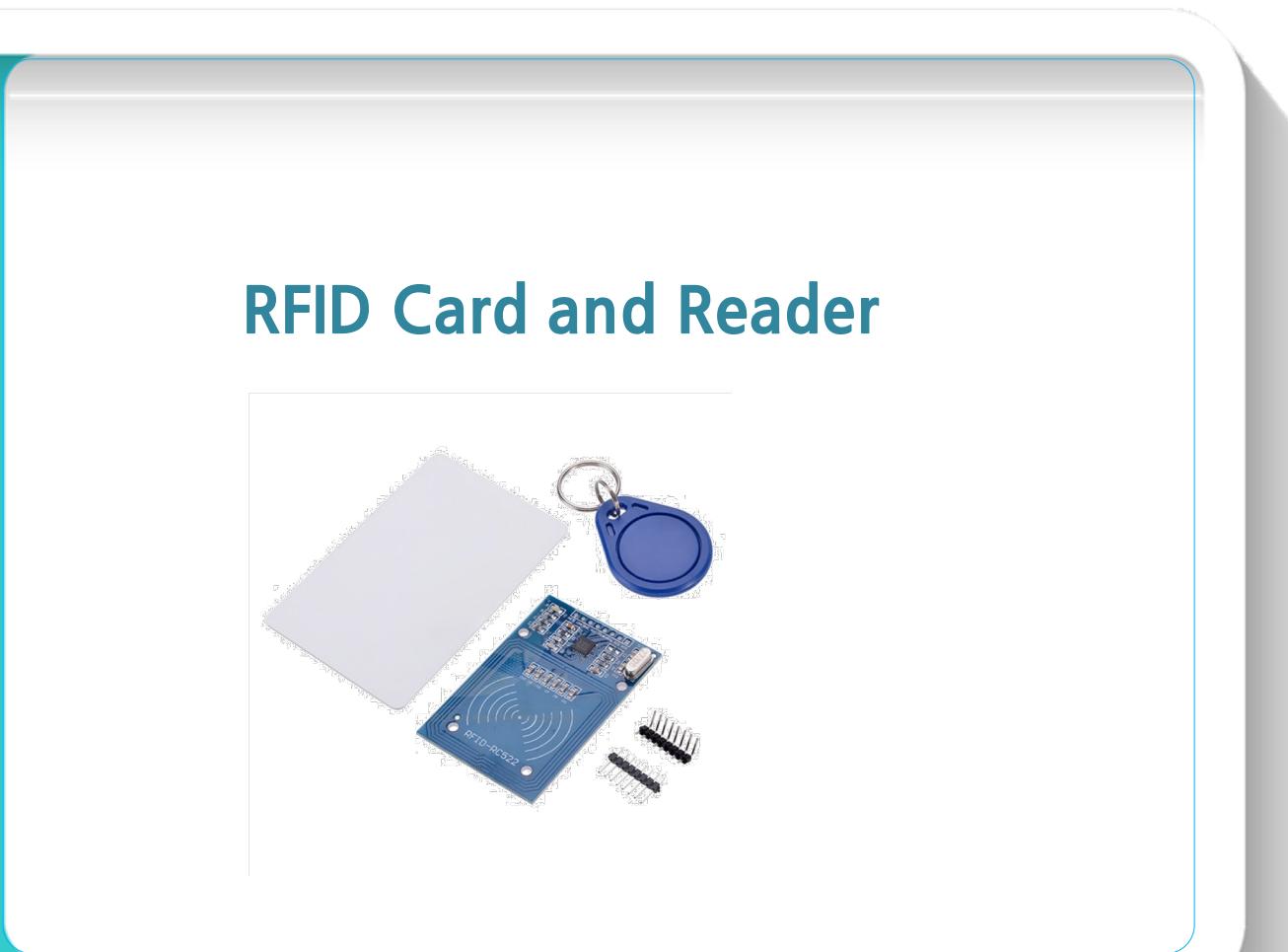
SS(Slave Select)

### 3-1) SPI-circuit



## 유선 통신 방법 비교표

통신방법	최대속도	사용거리	장점	단점
UART	115 Kbs	40m <	사용간편	연결 부품수
I <sup>2</sup> C	400 Kbs	짧다	많은 부품 연결	속도
SPI	20 Mbs	매우 짧다	사용간편	연결 부품수



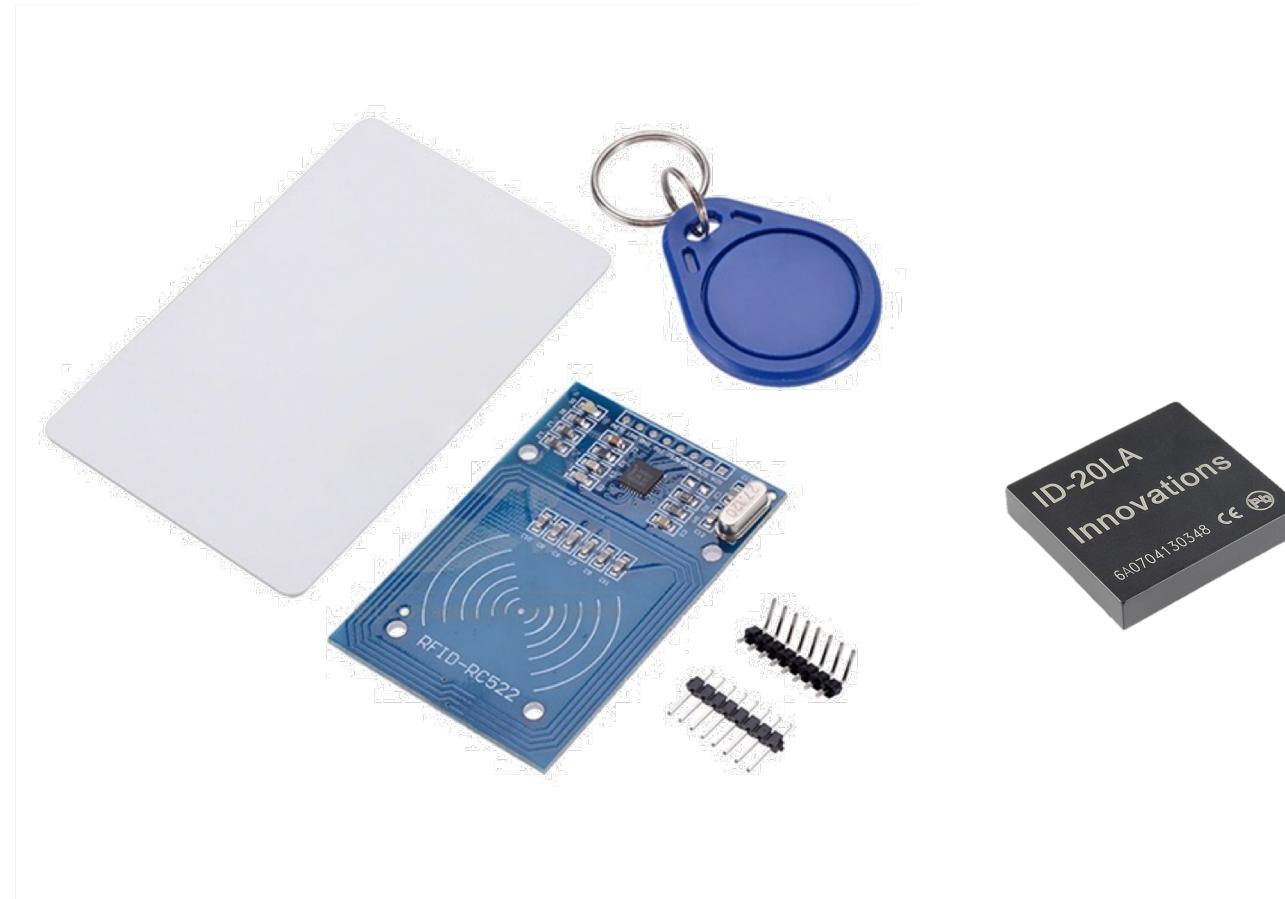
### 3-2) RFID Card and Reader

Active type and Passive type

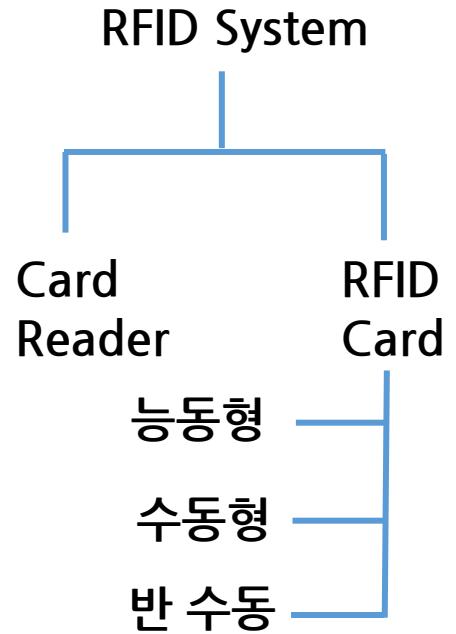
High-pass 는 Active type

Door 카드는 Passive type

RFID 카드 와 리더기



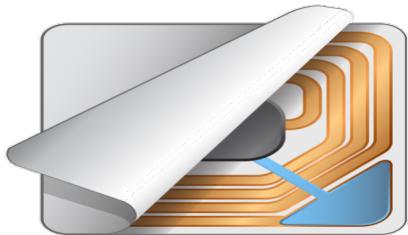
### 3-2) RFID Card and Reader



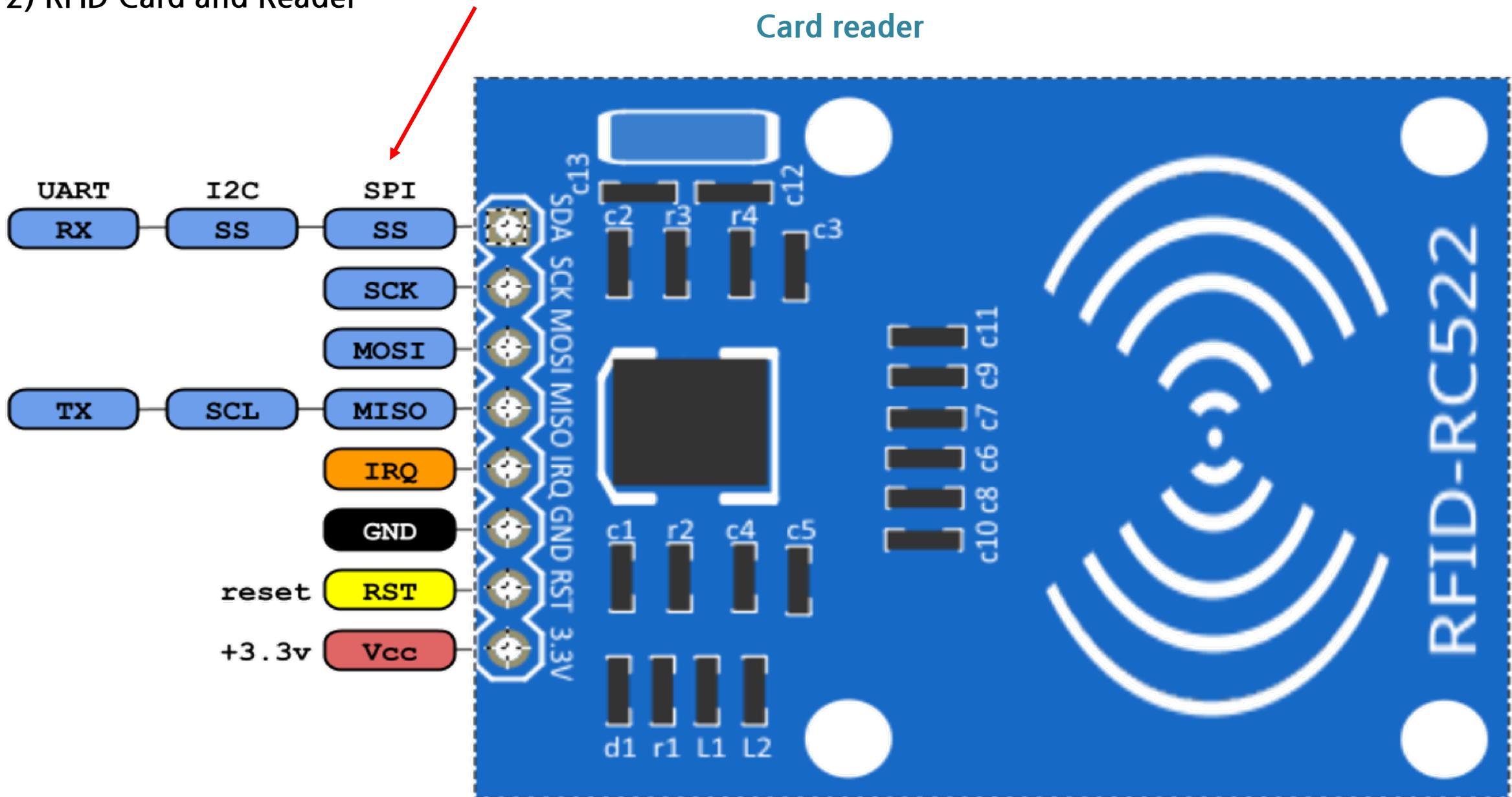
### 3-2) RFID Card and Reader

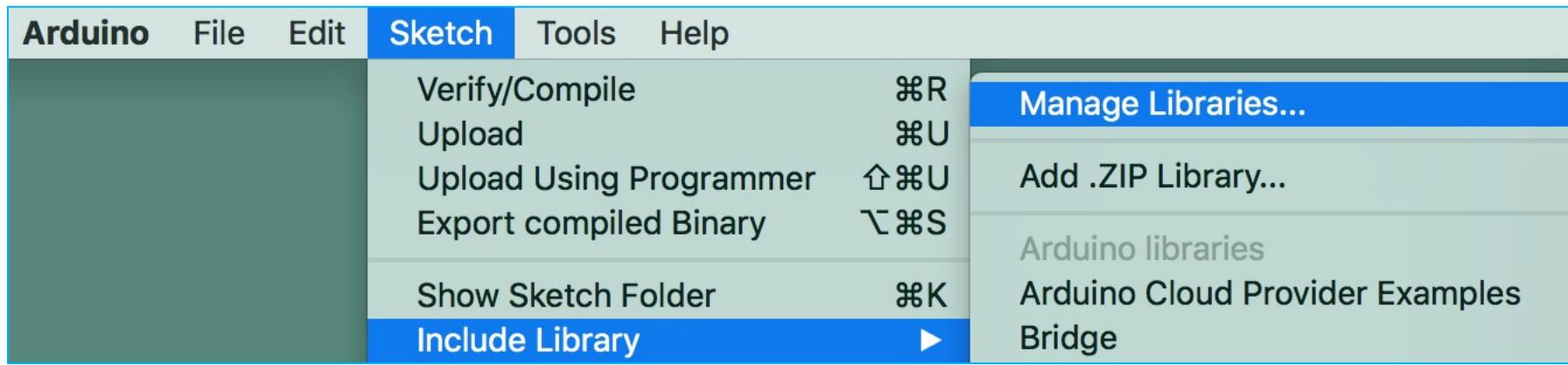
#### 수동 타입 RFID 카드가 작동 방법:

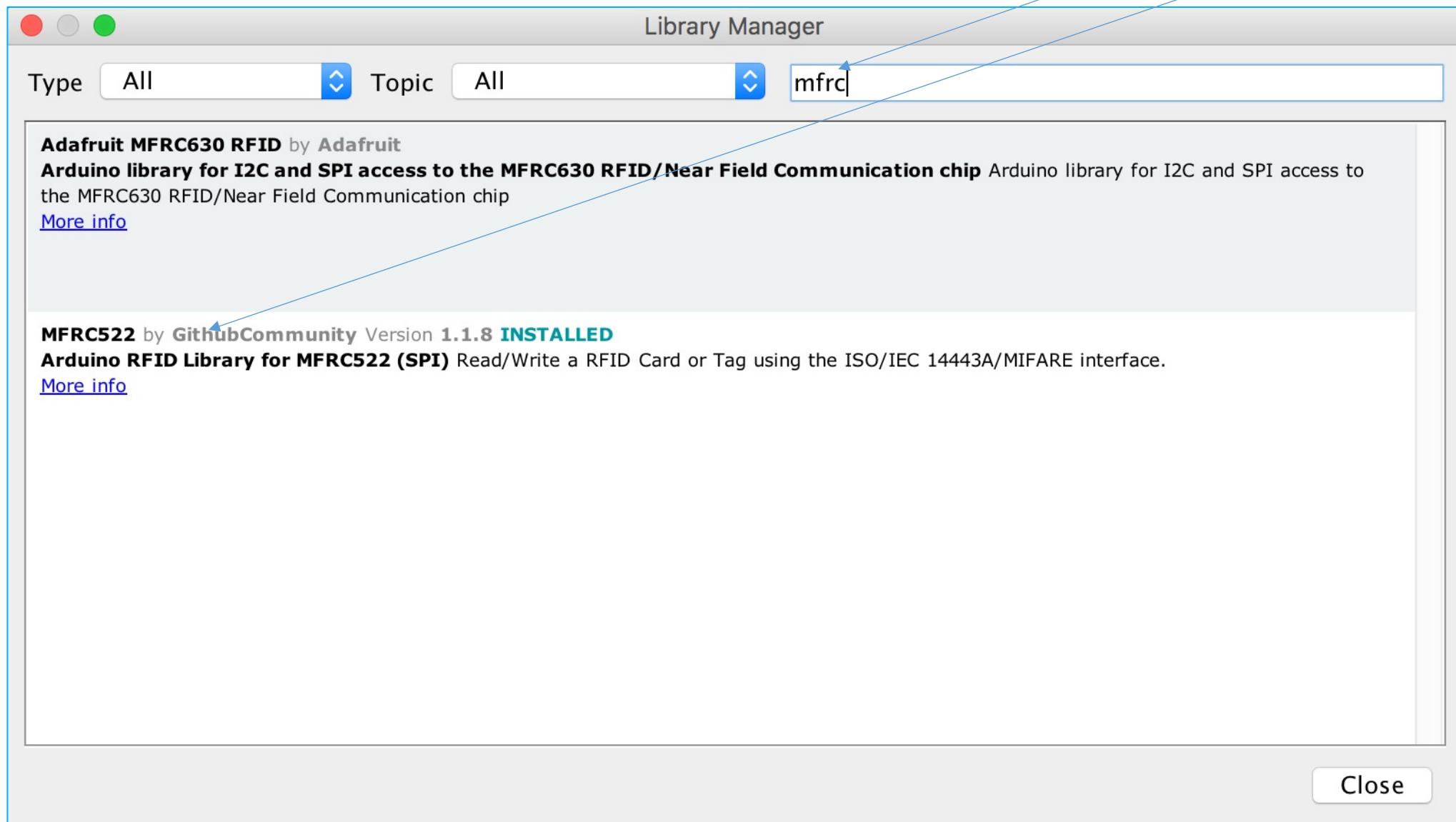
1. 리더기에서 전자기장 발생
2. 카드의 안테나 코일이 전자기파를 받아 전기에너지로 전환
3. 전기에너지로 카드에 있는 정보를 리더기에 전송
4. 리더기는 수신받은 데이터를 읽어 카드의 ID 파악



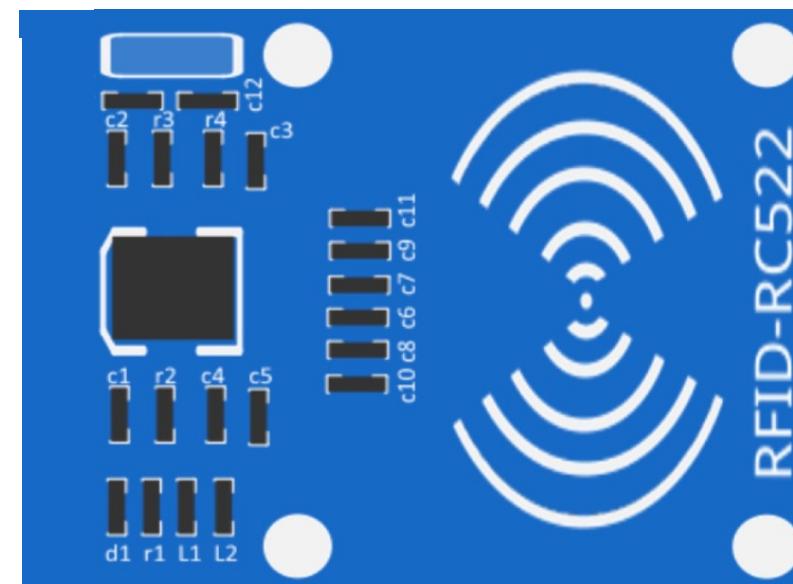
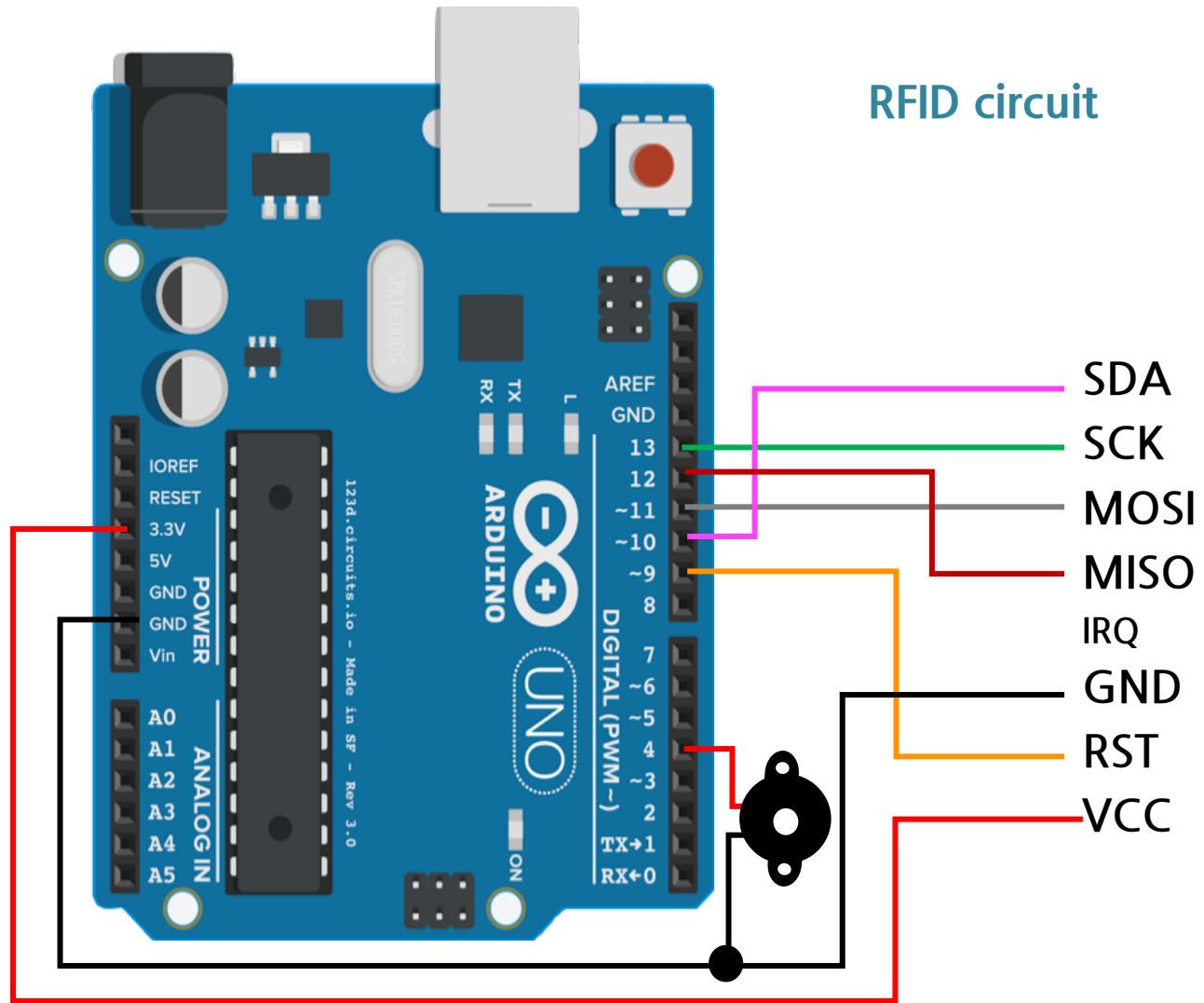
### 3-2) RFID Card and Reader



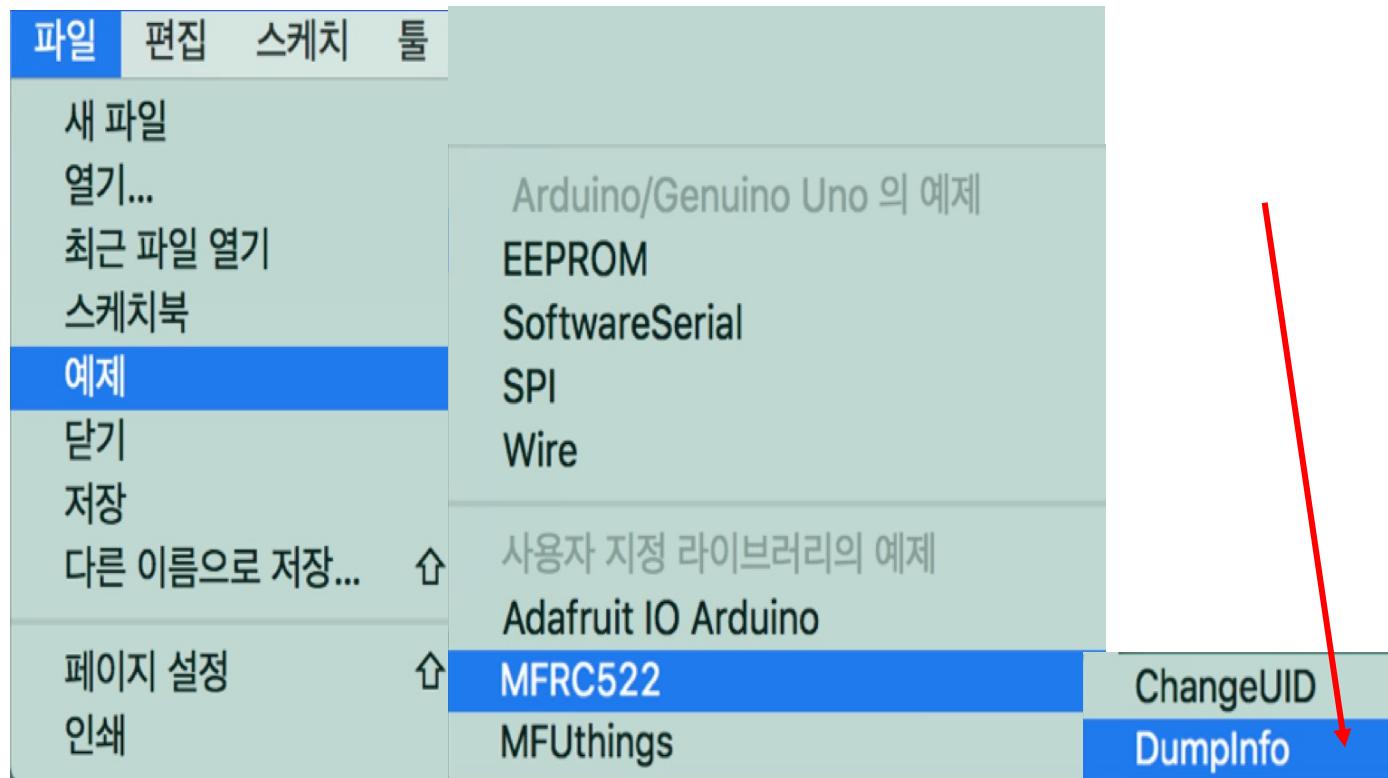




### 3-2) RFID Card and Reader

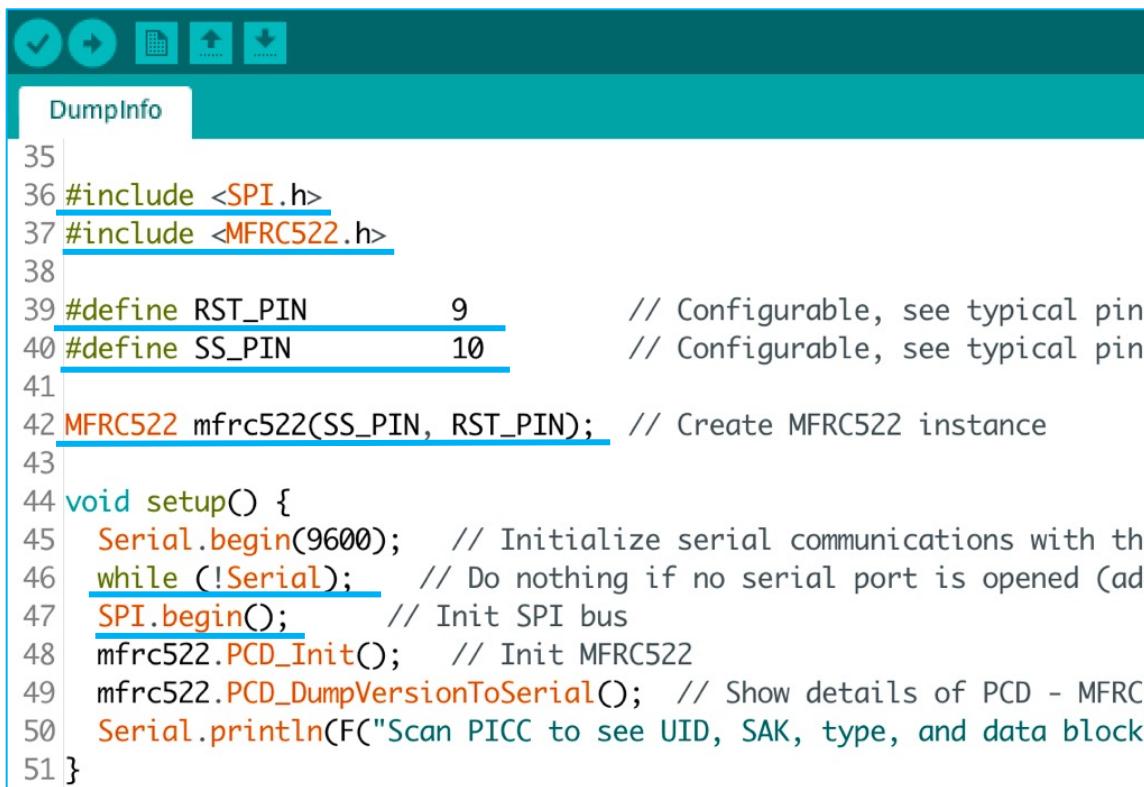


### 3-2) RFID Card and Reader



### 3-2) RFID Card and Reader

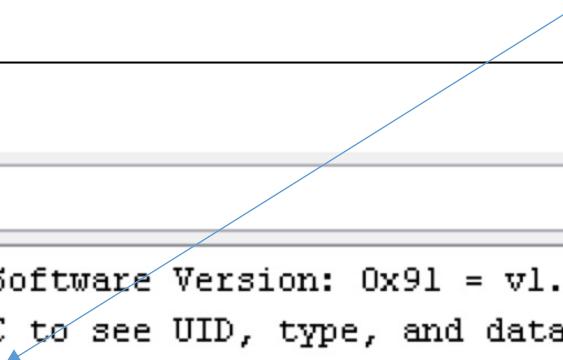
#### Dumpinfo Sketch



The screenshot shows the Arduino IDE interface with a sketch named "DumpInfo". The code is as follows:

```
35
36 #include <SPI.h>
37 #include <MFRC522.h>
38
39 #define RST_PIN 9          // Configurable, see typical pin
40 #define SS_PIN 10          // Configurable, see typical pin
41
42 MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
43
44 void setup() {
45   Serial.begin(9600);    // Initialize serial communications with the
46   while (!Serial);      // Do nothing if no serial port is opened (add
47   SPI.begin();          // Init SPI bus
48   mfrc522.PCD_Init();  // Init MFRC522
49   mfrc522.PCD_DumpVersionToSerial(); // Show details of PCD - MFRC5
50   Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks
51 })
```

## Dumpinfo result



Sector Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 AccessBits																
15	63	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	[ 0 0 1 ]
	62	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	61	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
14	59	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	[ 0 0 1 ]
	58	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	57	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	56	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
13	55	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	[ 0 0 1 ]

## New Sketch

The screenshot shows the Arduino IDE interface with a sketch titled "RFID522\_Tone". The code is written in C++ and performs the following tasks:

- Includes the `SPI.h` and `MFRC522.h` libraries.
- Defines pins: `RST_PIN` (pin 9) and `SS_PIN` (pin 10).
- Creates an instance of the `MFRC522` class named `m522` using the assigned pins.
- Declares variables `rfid` (String) and `rfid1` (String) with the value "224b886f".
- Implements the `setup()` function which initializes the serial port at 9600 bps, sets up the SPI bus, and initializes the `m522` card using the `PCD_Init()` method.
- Implements the `dump()` function which takes a buffer and its size, then iterates through the buffer to build the `rfid` string from individual bytes.

```
1 // RFID 522 Tone
2
3 #include <SPI.h>
4 #include <MFRC522.h>
5
6 int RST_PIN = 9 ; // UNO assigned pin
7 int SS_PIN = 10 ; // UNO assigned pin
8
9 MFRC522 m522(SS_PIN, RST_PIN); // assign name
10
11 String rfid;
12 String rfid1="224b886f" ;
13
14 void setup() {
15   Serial.begin(9600);
16   while (!Serial);
17   SPI.begin();
18   m522.PCD_Init(); // Init m522 card
19 }
20
21 void dump(byte *buffer, byte bufferSize) {
22   rfid="";
23   for (byte i = 0; i < bufferSize; i++) {
24     rfid=rfid + String(buffer[i], HEX);
25   }
26 }
```

```
27
28 void loop() {
29
30 if ( ! m522.PICC_IsNewCardPresent() ) // Look new card
31     return;
32 if ( ! m522.PICC_ReadCardSerial() ) // Select card
33     return;
34 dump(m522.uid.uidByte, m522.uid.size);
35 Serial.print(rfid);
36
37 if (rfid==rfid1) {
38     Serial.println(" Welcome Suzzi !!!");
39     tone(4, 300, 100) ; //Buzzer at PIN 4
40     delay(500);
41 }
42 if (rfid != rfid1) {
43     Serial.println(" Alert ");
44     tone(4, 3000, 1500) ;
45     delay(500);
46 }
47 }
```

## Ch10. IR Sensor

- 1) IR Distance sensor
- 2) IR (TCRT5000)



## IR Sensor

목적 : IR Distance sensor 이해 및 길이 측정 방법

준비물 :

아두이노 우노 1 개

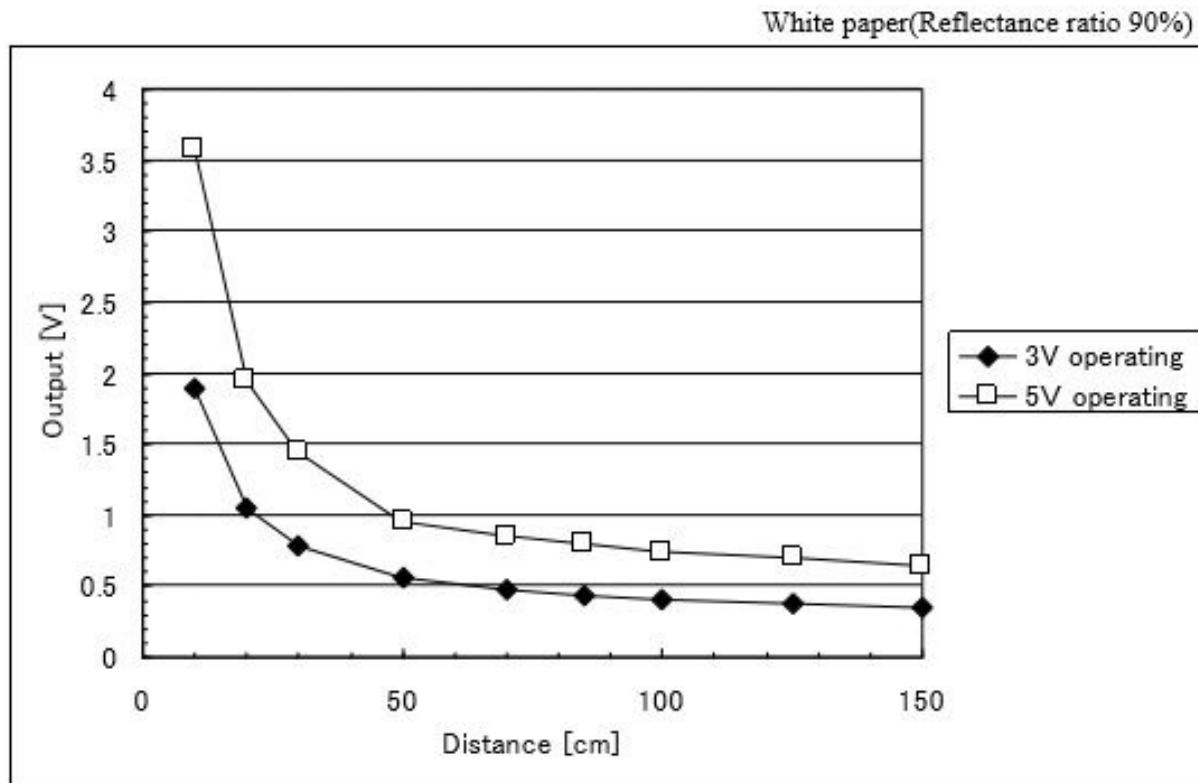
브래드보드 1개

GPY Sharp Distance sensor

점퍼케이블



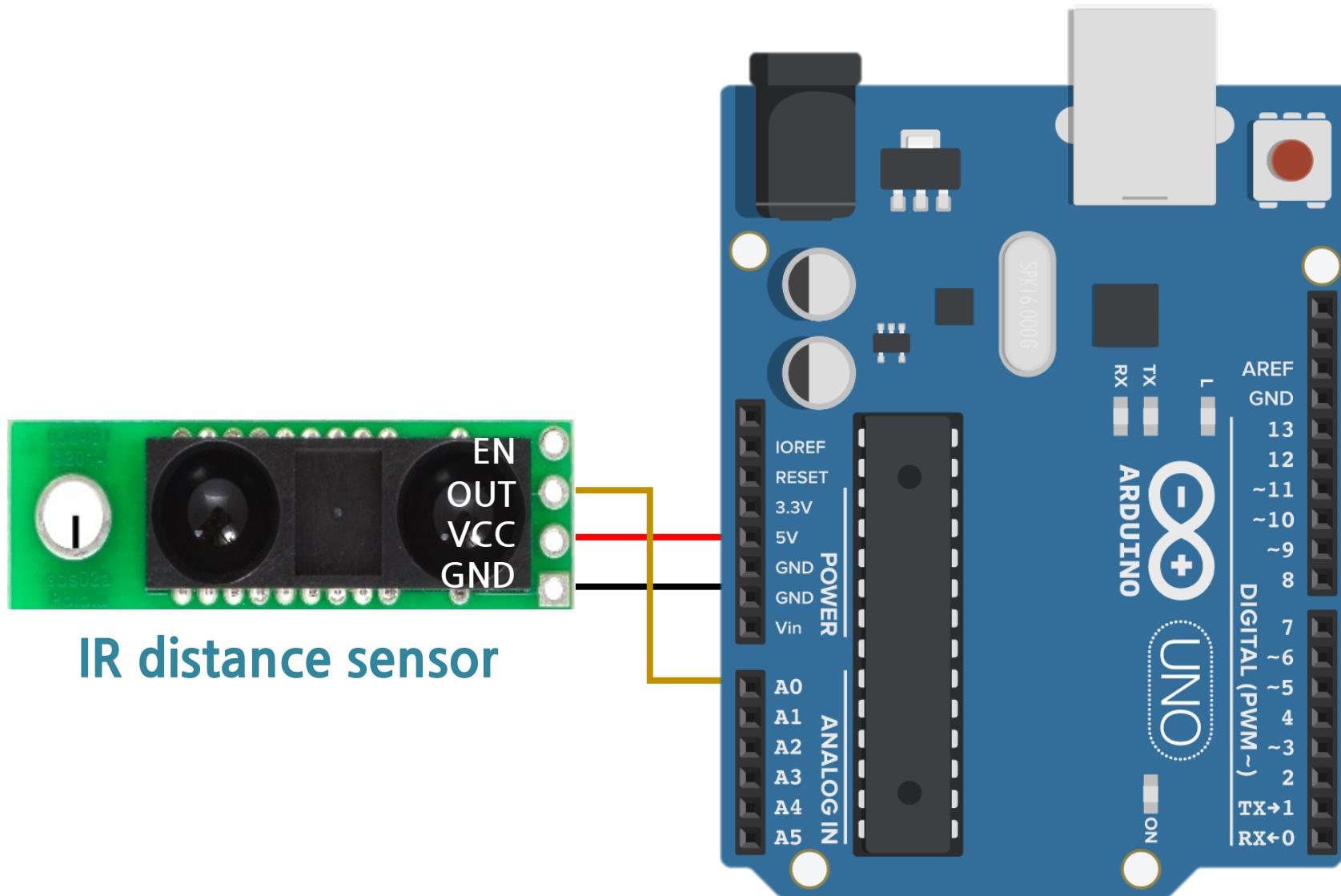
## 1) IR Distance sensor



## IR distance sensor

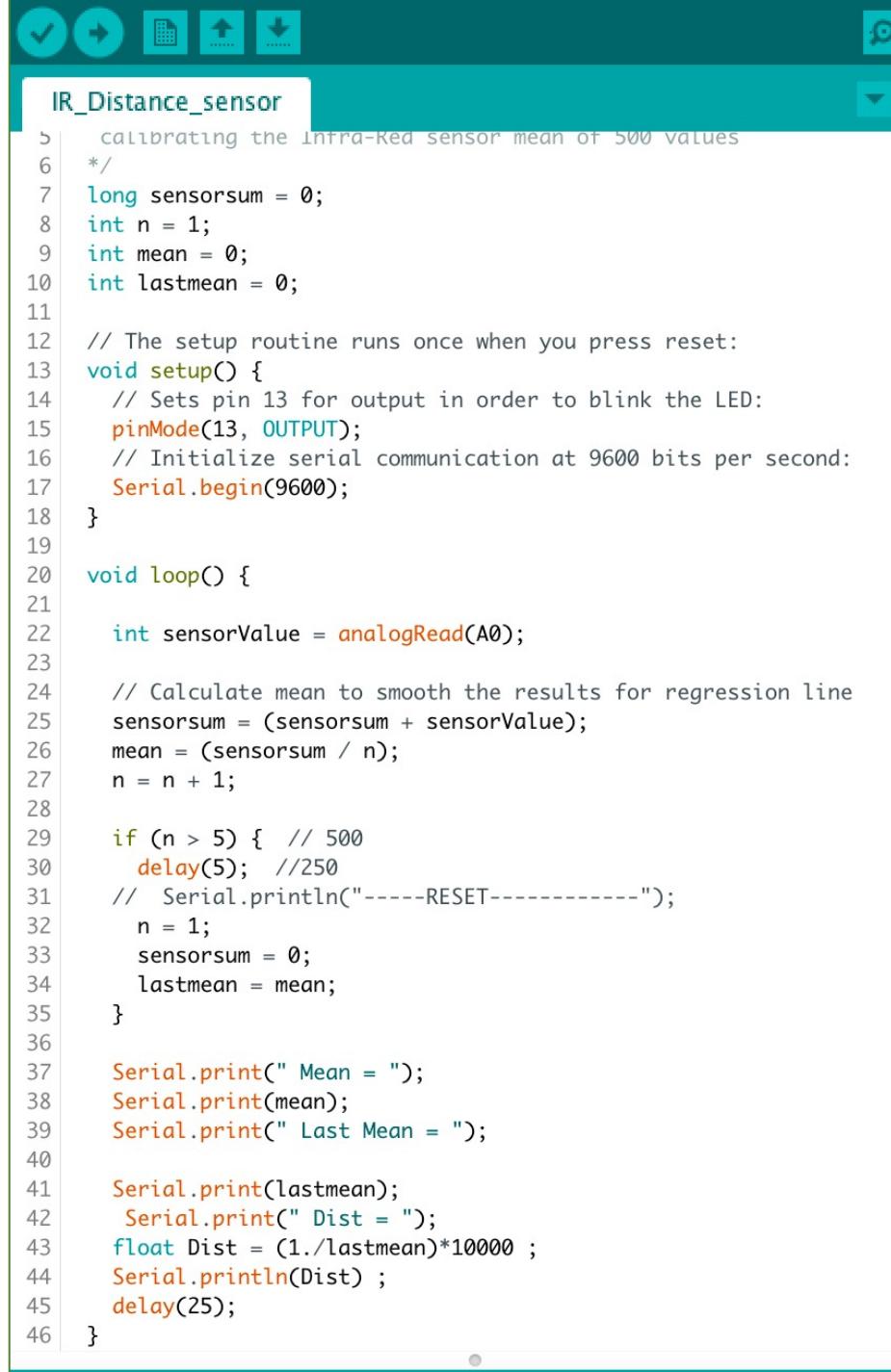
<http://www.instructables.com/id/How-to-setup-a-Pololu-Carrier-with-Sharp-GP2Y0A60S/>

# 1) IR Distance sensor



# 1) IR Distance sensor

For calibration

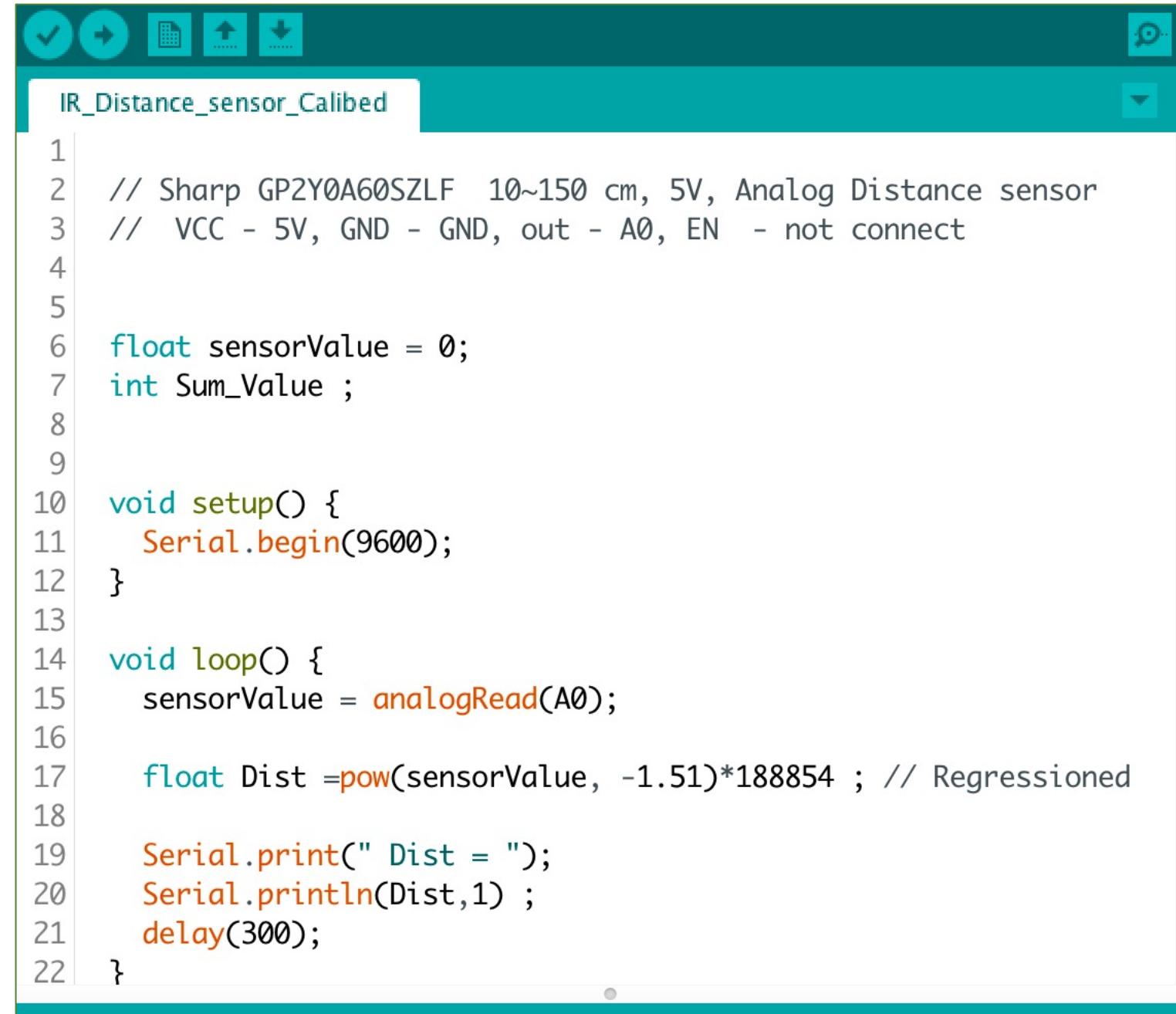


The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** IR\_Distance\_sensor
- Sketch:** A C++ code snippet for an Arduino sketch. The code is used for calibrating an Infrared distance sensor by averaging 500 values.
- Setup:** The setup routine initializes pin 13 as an output for the LED and starts serial communication at 9600 bits per second.
- Loop:** The loop routine reads the analog value from pin A0, calculates the mean of the last 500 values, and prints the current mean and the last mean to the Serial monitor. It also prints the calculated distance in centimeters.
- Comments:** The code includes several comments explaining its functionality.

```
5 	calibrating the Intra-Red sensor mean of 500 values
6 /*
7 	long sensorsum = 0;
8 	int n = 1;
9 	int mean = 0;
10 int lastmean = 0;
11
12 // The setup routine runs once when you press reset:
13 void setup() {
14     // Sets pin 13 for output in order to blink the LED:
15     pinMode(13, OUTPUT);
16     // Initialize serial communication at 9600 bits per second:
17     Serial.begin(9600);
18 }
19
20 void loop() {
21
22     int sensorValue = analogRead(A0);
23
24     // Calculate mean to smooth the results for regression line
25     sensorsum = (sensorsum + sensorValue);
26     mean = (sensorsum / n);
27     n = n + 1;
28
29     if (n > 5) { // 500
30         delay(5); //250
31         // Serial.println("----RESET-----");
32         n = 1;
33         sensorsum = 0;
34         lastmean = mean;
35     }
36
37     Serial.print(" Mean = ");
38     Serial.print(mean);
39     Serial.print(" Last Mean = ");
40
41     Serial.print(lastmean);
42     Serial.print(" Dist = ");
43     float Dist = (1./lastmean)*10000 ;
44     Serial.println(Dist) ;
45     delay(25);
46 }
```

# 1) IR Distance sensor



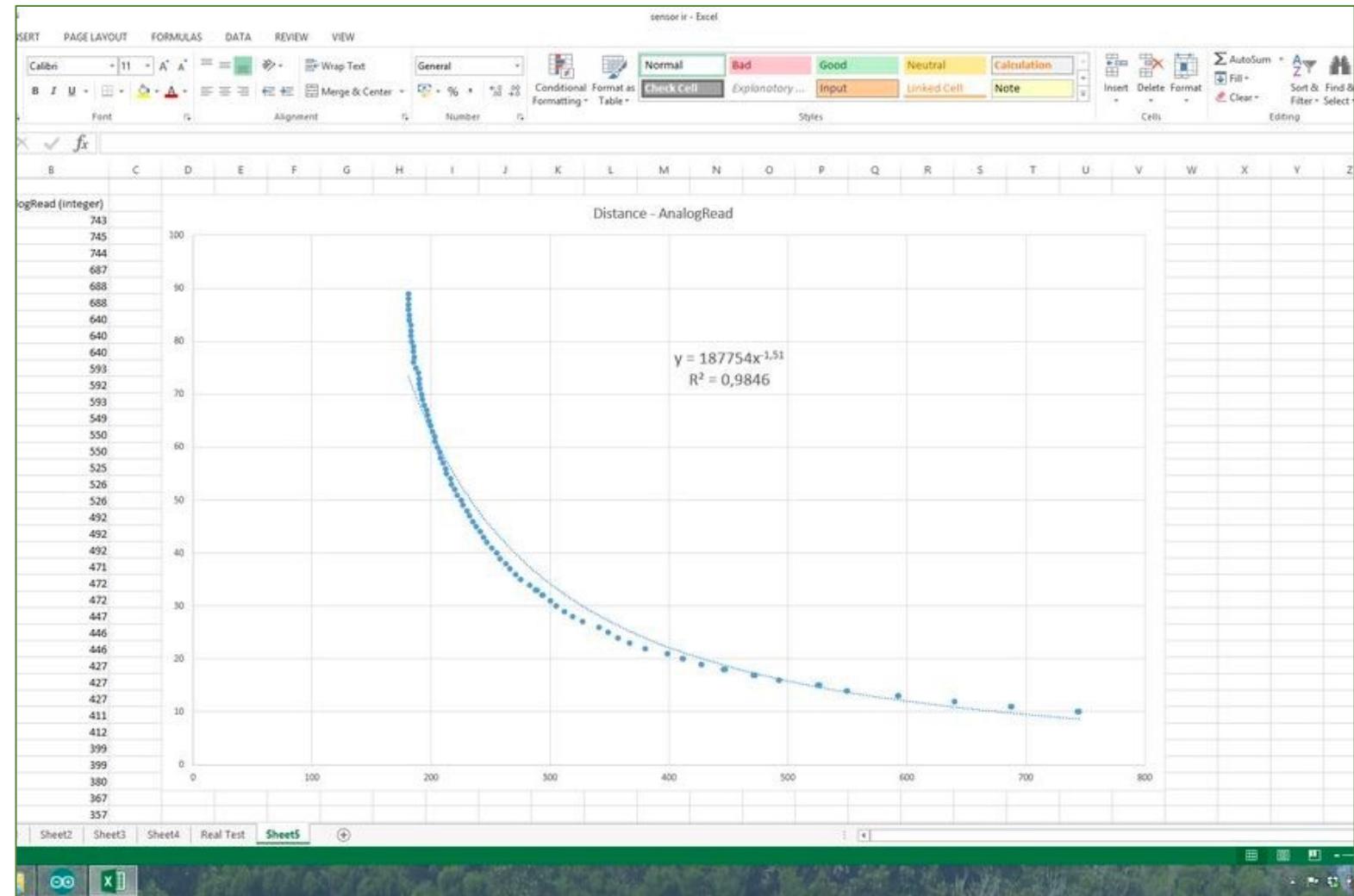
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** IR\_Distance\_sensor\_Calibed
- Tool Buttons:** Checkmark, Refresh, File, Upload, Download, and a gear icon.
- Code Area:** Displays the following C++ code for an IR distance sensor using a Sharp GP2Y0A60SZLF sensor connected to A0:

```
1 // Sharp GP2Y0A60SZLF 10~150 cm, 5V, Analog Distance sensor
2 // VCC - 5V, GND - GND, out - A0, EN - not connect
3
4
5
6 float sensorValue = 0;
7 int Sum_Value ;
8
9
10 void setup() {
11   Serial.begin(9600);
12 }
13
14 void loop() {
15   sensorValue = analogRead(A0);
16
17   float Dist =pow(sensorValue, -1.51)*188854 ; // Regressioned
18
19   Serial.print(" Dist = ");
20   Serial.println(Dist,1) ;
21   delay(300);
22 }
```

# 1) IR Distance sensor

$$Y = 187754X^{-1.51}$$



IR distance sensor