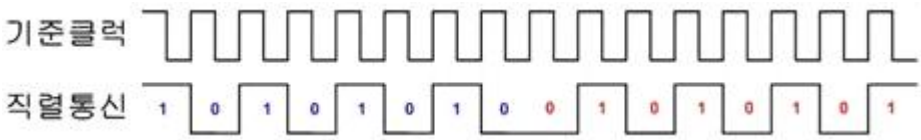
4. 시리얼 통신

1) 개요

직렬 통신이라고도 부른다. 컴퓨터와 컴퓨터 간 혹은 컴퓨터와 주변장치 간에 데이터를 주고 받을 때에 비트 단위로 전송을 하는 방식을 취하는데 하나의 통신 선으로 한 주기에 한 비트씩 차례로 보내는 방식이 바로 시리얼 통신 방식이다.



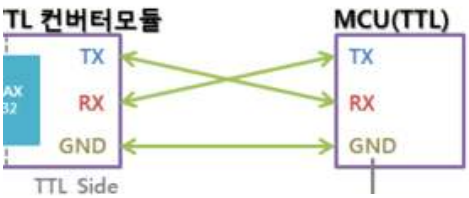
위 그림에서처럼 일정한 주기를 가진 클럭이 주어지면 이 클럭을 기준으로 0과 1의 신호를 주기에 맞추어서 보내게 된다. 따라서 기준 클럭의 주기를 짧게 하면 통신 속도가 빨라지고 반면에 주기를 길게 하면 통신 속도 역시 느려지게 된다.

2) 핀 배열

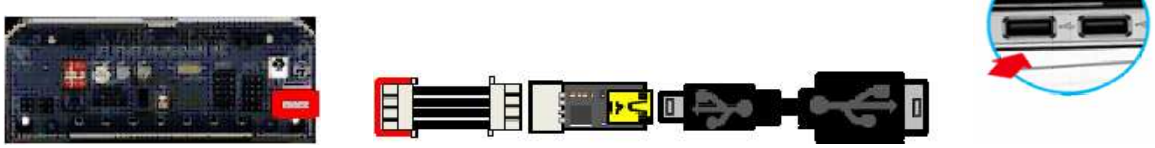
시리얼 통신을 하는 장치들은 보통 4개의 핀으로 이루어진다(비동기식). 전원핀(V와G), 그리고 Tx 핀과 Rx 핀이다. Tx는 데이터를 송신하는 송신측(transceiver)핀이고 Rx는 데이터를 수신하는 수신측(Receiver)핀이다. 혹은 V핀을 구성하지 않고 Tx와 Rx 그리고 GND핀으로만 되어 있는 경우도 있다.

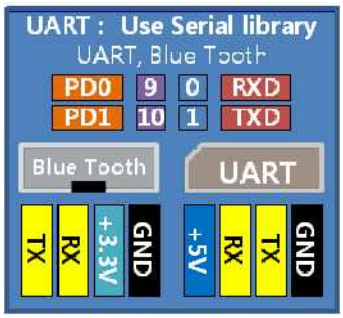
3) 연결 방법

시리얼 통신의 배선 예가 그림 4-2에 나타나 있다. 이 예에서처럼 컴퓨터와 컴퓨터 간, 혹은 컴퓨터와 주변장치간에 시리얼 통신 장치를 배선할 때에는 아래 그림에서처럼 MCU의 Tx를 통신하고자 하는 주변장치의 Rx와 연결하여야 한다. 하지만 스마트 인벤터 보드에서는 시리얼 통신 장치들을 연결할 수 있도록 커넥터들을 구성해 놓았으므로 배선 방법을 고민하지 않고 편리하게 사용할 수 있다.



PC와 시리얼 통신하기 위해 연결하는 경우, 아래 그림에서처럼 스마트 인벤터 보드의 UART 소켓에 USB 업로더를 연결한 것으로 배선은 완료된다.





스마트 인벤터 보드에서 시리얼 통신 소켓은 두 개가 설치되어 있다. 하나는 블루투스 모듈을 연결할 수 있으며 블루투스 구동 전압인 3.3V를 공급할 수 있도록 하였다. 다른 하나는 USB 업로더나 임의의 시리얼 통신 장치를 연결하기 위하여 사용하는 소켓으로 스마트 인벤터 보드의 기본 구동 전압인 5V를 사용한다. 유의해야 할 것은 사용자의 편의를 위해서 소켓을 두 개로 구성하였지만 하나의 시리얼 통신 포트를 공유한다는 점이다. 따라서 두 개의 소켓에 모두 통신 장치를 연결하여 동시에 사용할 수는 없다.

4) 아두이노 함수 설명

Serial.begin(bps);

시리얼 통신 포트를 열고 통신 속도(기준 클럭)를 결정하는 함수이다. 여기서 bps(bit per second)는 통신 속도인데 2400 / 4800 / 9600 / 14400 / 19200 / 38400 / 57600 / 115200 bps를 사용할 수 있다.

Serial.available();

현재 수신된 바이트의 개수를 반환한다. 아두이노는 64바이트의 용량을 지닌 시리얼 데이터 수신 버퍼를 운영하는데 수신된 바이트들을 우선 이 버퍼에 저장한 후 아직 읽지 않은 데이터의 개수를 이 함수를 통해 알려준다.

Serial.read();

버퍼에 저장된 수신 바이트 중 가장 먼저 저장된 1개의 바이트를 읽어 온다.

Serial.readbytes(buffer, length);

Buffer: 수신된 바이트들을 저장할 버퍼 이름
Length: 읽어올 바이트의 개수

수신된 바이트들을 동시에 여러 개 읽어올 때 사용한다. 배열 변수로 정의된 버퍼의 이름과 읽기를 희망하는 바이트의 개수를 지정해서 함수를 호출하면 실제로 읽어온 바이트

의 개수를 반환하게 된다.

Serial.println(val);

시리얼 통신 포트를 통해 ASCII 문자열 데이터를 출력하는 함수이다. 예를 들면
Serial.print(50); -> ASCII 문자열 "50"을 송신한다. (값 50을 송신하는 것이 아님)
Serial.print("Hello world") -> ASCII 문자열 "hello world"를 송신한다.
이렇게 송신을 마친 후에는 라인피드와 캐리지 리턴에 해당하는 ASCII코드를 전송하여 줄바꿈을 해준다. 만약 이러한 줄바꿈을 원치 않는 경우에는 Serial.print()를 사용하면 된다.

Serial.write(val);

하나의 바이트를 송신한다. 따라서 피라메터 val에는 0 ~ 255의 숫자가 들어가야 한다. ASCII 문자열을 보내는 print()와는 달리 실제 이진 데이터를 송신한다.

void serialEvent() { 문장1, 문장2,... }

이 함수를 사용하면 바이트가 수신될 때마다 자동으로 중괄호 내의 문장들이 실행된다.
보통 이 함수 안의 문장들에는 Serial.read()가 포함되어 수신된 바이트를 읽는 코드가 포함
되어 있다. 이렇게 데이터를 수신하면 데이터를 수신하기 위해 특정 루프를 계속 돌리거나,
혹은 계속 대기하는 방식보다 더 효율적인 방식이라고 할 수 있을 것이다.

5) 프로그래밍

시리얼 통신은 매우 다양하고 폭넓은 사용 예들을 갖고 있다. 그 중 몇 가지 예제들을 살
펴 보기로 한다.

예제 1 : 전방 적외선 센서의 감지 여부를 시리얼 모니터로 출력하기

전방에 있는 3개의 적외선 센서의 물체 감지 여부를 나타내는 예제이다. 물체가 감지되면
3개 중 어느 물체에 감지되었는지의 여부를 알려준다. PC와 제어기가 USB 업로더로 연결되어 있는 상태여야 하며 실행 전 시리얼 모니터를 화면에 띄워 놓도록 한다.

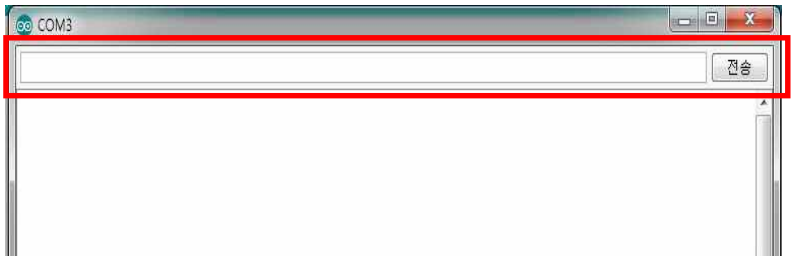
```
void setup() {
    Serial.begin(9600);
}

void loop() {
   int   frontSensor0 = analogRead(A0);
   int   frontSensor1 = analogRead(A1);
   int   frontSensor2 = analogRead(A2);

   if(frontSensor0 < 100) { Serial.println("found on the left"); }
   if(frontSensor1 < 100) { Serial.println("found on the center"); }
   if(frontSensor2 < 100) { Serial.println("found ont the right"); }

}
```

예제 2 : 수신 받은 값에 따라 8개의 내부 LED를 깜빡이기

시리얼 모니터의 입력 창에 0 ~ 9까지의 번호를 입력하고 전송버튼을 누르면 그 횟수에 따라 8개의 내부 LED가 깜빡이는 예제이다.

시리얼 모니터 창을 띄우고 전송창에 0부터 9까지 숫자를 입력하고 엔터를 누르거나 전송

버튼을 누르면 그 숫자의 횟수만큼 LED가 깜빡이는 것을 볼 수 있다.




```
int recvBytes;
int num;
char inChar[0];

void setup() {
   pinMode(11, OUTPUT);
   pinMode(12, OUTPUT);
   pinMode(13, OUTPUT);
   pinMode(14, OUTPUT);
```

```
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(17, OUTPUT);
  pinMode(18, OUTPUT);
  Serial.begin(9600);
  Serial.println("Ready... ");
  inChar[0] = '0';                          // 캐릭터형으로 초기화
}

void loop() {
 if(recvBytes)                              // 새로 들어온 데이터가 있다면 LED 깜빡이기
 {
    for(int j = 0; j < num; j++)
    {
        digitalWrite(11, HIGH);             // 0.1초 간격으로 깜빡이기
        digitalWrite(12, HIGH);
        digitalWrite(13, HIGH);
        digitalWrite(14, HIGH);
        digitalWrite(15, HIGH);
        digitalWrite(16, HIGH);
        digitalWrite(17, HIGH);
        digitalWrite(18, HIGH);
        delay(100);
        digitalWrite(11, LOW);
        digitalWrite(12, LOW);
        digitalWrite(13, LOW);
        digitalWrite(14, LOW);
        digitalWrite(15, LOW);
        digitalWrite(16, LOW);
        digitalWrite(17, LOW);
        digitalWrite(18, LOW);
        delay(100);
    }
 }
}

void serialEvent()                          // 데이터가 수신되면 실행되는 이벤트 함수
{
  recvBytes = Serial.available();           // 수신된 데이터의 개수를 수신함
```

```
  if(recvBytes)
  {
    inChar[0] = (char)Serial.read();      // 수신된 데이터를 캐릭터형 배열 변수에 저장
    num = atoi(inChar);                   // 받은 캐릭터를 정수로 변환함
  }
}
```

예제 검토: 수신된 데이터는 정수형이 아닌 ASCII 캐릭터 값이다. 따라서 캐릭터 값(0~9)을
정수로 바꾸어주기 위해 atoi(char*)를 사용하였다. atoi함수의 파라메터 데이터 타입은 캐릭
터 배열이므로 캐릭터를 저장하는 변수(inChar)의 데이터 타입 역시 배열형으로 하였다.