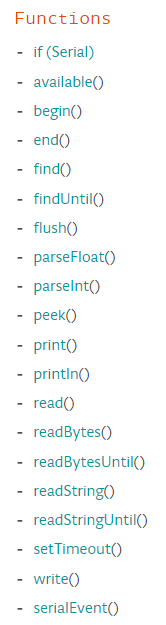
**아두이노 Serial 함수의 명령어 알아보기 (1)**

기본적으로 Serial함수에 대해 알고 계시고, 자주 사용하는 함수는 Serial.begin(), Serial.print(), Serial.println(), Serial.available(), Serial.read() 함수일 것입니다.

저 위의 함수들만 사용할 수 있으면 기본적으로 시리얼 통신을 이용하는데 전혀 문제가 없지만, 밑의 사진을 보시면 아두이노에서 Serial을 사용하는데 다양한 도움을 주는 함수들을 많이 제공합니다.

밑의 함수들을 보시면 시리얼 통신할 때 알아두면 유용한 함수들이 많은데요, 이번 컨텐츠에서는 한국형 아두이노 오렌지보드에서 사용할 수 있는 다양한 Serial함수들에 대해 알아보고 어떻게 동작하는지 알아보겠습니다.



위의 함수들을 컨텐츠에서 전부 다루지는 못하지만 알아두시면 매우 유용한 함수들 위주로 하나씩 소개하도록 하겠습니다.

1. Serial.Write()

사용 방법

 - Serial.write(val)

 - Serial.write(str)

 - Serial.write(buf, len)

**변수**

 - val : byte형 데이터

 - str : string형 문자열

 - buf : byte형 배열

 - len : 배열의 길이

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  void loop(){    const uint8\_t temp[5] = {'1', '2', '3', '4', '5'};    **Serial.write(1);**  // write()함수로 1 전송   delay(500);    Serial.write(49); // write()함수로 49 전송   delay(500);    **Serial.print(1);**  // print()함수로 1 전송   delay(500);    Serial.write('a');  // write()함수로 'a' 전송   delay(500);    Serial.write(temp, 5);  // write()함수로 temp배열을 5만큼 전송   delay(500);    Serial.write("12345"); // write()함수로 string값 전송   delay(500);    Serial.write('\n');  // 줄바꿈   delay(500);  } | 그림입니다.  원본 그림의 이름: mem0000211c0002.png  원본 그림의 크기: 가로 172pixel, 세로 280pixel |

설명

Serial.write()함수는 Serial.print() 함수와 마찬가지로 데이터 값을 시리얼 통신으로 송신하는 기능을 합니다.

위의 소스코드 결과 화면을 보시면 write()함수와 print() 함수가 모두 1 을 전송하는데(첫번 째와 세 번째) write()함수는 이상한 값을 출력합니다. write(49)를 출력해야 제대로 값이 1이 나오는 것을 확인 할 수 있습니다.

print()함수는 1값을 아스키 코드로 해석해서 데이터를 전송하지만, write()함수는 1값을 그대로 전송해서 시리얼 모니터에서 아스키 코드로 해석해서 보여주므로 다르게 보이는 것입니다.

 아스키 코드 값 49가 문자 ‘1’이어서 49를 write()함수로 전송해야 시리얼 모니터에서 아스키 코드 문자 1로 해석하고 문자 1을 보여주는 것입니다.

write()함수는 기본적으로 시리얼 모니터에 표시할 때는 사용하지 않지만 블루투스나 지그비 간 통신을 할 때 문자 외에 직접 데이터 값을 보내야 할 경우 사용합니다.

2. Serial.end()

 사용방법

 - Serial.end();

설명

Serial.end는 Serial통신을 사용하지 않도록 하는 함수입니다. Serial.begin()이 Serial 통신을 사용하기 위해 사용되는 것이라면, Serial.end()는 Serial에 사용되는 핀(디지털 0, 1번 핀)을 일반 디지털 핀으로 사용하게 하는 함수입니다.

3. Serial.parseInt()

사용방법

 - Serial.parseInt();

**반환값**

 - Long타입 숫자 반환(입력 받은 값이 숫자일 경우)

 - 숫자 값이 아닐 경우 0을 반환

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  void loop(){    if(Serial.available()){      long value1 = Serial.parseInt();      long value2 = Serial.parseInt();                    Serial.print("value 1 : ");      Serial.println(value1);      Serial.print("value 2 : ");      Serial.println(value2);      Serial.print("value1 + value2 = ");      Serial.println(value1 + value2);    }      } | 그림입니다.  원본 그림의 이름: mem0000211c0003.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.parseInt()함수는 Serial.read()함수와 기능은 비슷합니다. Serial.read()함수는 보내준 데이터 값을 그대로 읽어오지만 parseInt()함수는 보낸 값을 long 형태의 숫자 값으로 바꿔서 저장하게 됩니다.

위의 예제 결과 화면을 보시면 모니터 창에 15 25를 입력 하면 그 값을 parseInt()로 받고 그 값을 숫자로 바꿔서 저장하고, 저장한 2개의 값을 더해서 결과 값을 보여주게 됩니다.

위에 15 값을 read()로 받게 되면 '1' '5' 두개의 문자를 받지만 parseInt()로 받아서 long값 15를 받게 되었습니다. 참고로 숫자 값이 아닐 경우 0을 반환하게 되니 이를 이용하여 숫자와 문자를 구분하는 것으로도 이용할 수 있습니다.

4. Serial.parseFloat()

사용방법

 - Serial.parseFloat()

**반환값**

 - Float형 숫자 반환(입력 값이 숫자일 경우)

 - 숫자 값이 아닐 경우 0을 반환

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  void loop(){    if(Serial.available()){      float value1 = Serial.parseFloat();      float value2 = Serial.parseFloat();                    Serial.print("value 1 : ");      Serial.println(value1);      Serial.print("value 2 : ");      Serial.println(value2);      Serial.print("value1 + value2 = ");      Serial.println(value1 + value2);    }      } | 그림입니다.  원본 그림의 이름: mem0000211c0004.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

위에서는 정수형태의 숫자만 처리했다면, 이번엔 소숫점까지 처리할 수 있는 함수입니다.

parseInt()와 같은 기능이지만 반환값이 long값이 아닌 float값 입니다. 간단하게 설명하면 parseInt()는 정수형태의 값으로 변환하고, parseFloat()는 소수형태 값으로 변환합니다.

(두개의 예제 소스 코드 결과화면을 비교해 보시면 간단하게 이해하실 수 있습니다.)

5. Serial.readBytes()

사용방법

 - Serial.readBytes(buffer, length)

**변수**

 - buffer : 입력받은 데이터를 저장할 buffer

 - length : 입력받을 문자의 길이

**반환**

 - 읽어온 데이터의 길이 byte값

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  void loop(){    char temp[100];    if(Serial.available()){      byte leng = Serial.readBytes(temp, 20);            Serial.print("Input data Length : ");      Serial.println(leng);            for(int i = 0; i <leng; i++){        Serial.print(temp[i]);      }      Serial.println();    }      } | 그림입니다.  원본 그림의 이름: mem0000211c0005.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.readBytes() 함수는 지정한 숫자만큼이나 일정 시간동안 읽은 데이터를 buffer에 저장해주는 함수입니다.(일정 시간은 밑의 serial.setTimeout() 함수를 참조하시면 됩니다.)

지정한 숫자만큼 데이터가 입력되지 않으면 입력받은 개수만큼만 저장하고, 지정한 숫자 이상으로 데이터가 들어 올 경우 지정된 숫자만큼 buffer에 저장하고, 그 이후에 들어온 값을 따로 buffer에 저장합니다.

함수의 반환값은 버퍼에 저장한 문자의 갯수를 byte형태로 반환됩니다.

6. Serial.readBytesUntil()

사용방법

 - Serial.readBytesUntil(char, buffer, length)

**변수**

 - char : 종료 문자열

 - buffer : 입력받은 데이터를 저장할 buffer

 - length : 입력받을 문자의 길이

**반환**

 - 읽어온 데이터의 길이 byte값

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면(기준값 'k') |
| void setup(){    Serial.begin(9600);  }  void loop(){    char temp[100];    if(Serial.available()){      byte leng = Serial.readBytesUntil('k', temp, 20);            Serial.print("Input data Length : ");      Serial.println(leng);            for(int i = 0; i <leng; i++){        Serial.print(temp[i]);      }      Serial.println();    }      } | 그림입니다.  원본 그림의 이름: mem0000211c0006.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.readBytesUntil() Serial.readBytes() 함수와 같이 지정한 숫자만큼이나 일정 시간동안 읽은 데이터를 buffer에 저장해주는 함수입니다.(일정 시간은 밑의 serial.setTimeout() 함수를 참조하시면 됩니다.)

 다른 점은 맨 처음 변수에 지장한 특정 문자가 들어올 때까지 데이터를 읽어 오게 됩니다.(혹은 지정한 숫자만큼 데이터가 입력이 되거나)

 기본은 readBytes()함수와 같습니다. 지정한 숫자보다 적게 들어올 경우 입력받은 갯수만큼만 저장되고, 초과될 경우 나눠서 저장이 됩니다. 결과 화면에 보이듯이 특정 문자가 들어올 경우는 특정 문자 앞/뒤로 나뉘어서 버퍼에 저장되게 됩니다. (위에 예제는 'k'값이 특정 문자로 지정해서 실행한 예제 입니다. 입력값은 abcd**k**efgh 입니다.)

함수의 반환값은 버퍼에 저장한 문자의 갯수를 byte형태로 반환됩니다.

**\* 주의할 점은 특정 문자로 지정한 문자는 버퍼에 따로 저장되지 않습니다.**

7. Serial.setTimeout()

사용방법

 - Serial.setTimeout(time)

**변수**

 - time : 설정할 밀리초 단위의 타임아웃 시간(long 타입)

설명

Serial.setTimeout() 함수는 Serial.readBytes(), Serial.readBytesUntil(), Serial.parseInt(), Serial.praseFloat 함수를 이용할 때 데이터 값 수신을 대기하는 최대 시간을 세팅하는 함수 입니다.

기본적으로 1초(1000 millisecond)가 설정되어 있으며, 변수로 입력하는 값의 단위는 밀리초 입니다.(1second = 1000millisecond)

8. Serial.flush()

사용방법

 - Serial.flush()

예제 소스 코드(비교)

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면(flush() 미사용) |
| char \*buf = "This message for checkint a time";  void setup() {    Serial.begin(9600);  }    void loop() {    unsigned long int atime;    for(int i = 0; i <10; i++) {      atime = micros();      Serial.println(buf);      Serial.println(micros() - atime);    }    while(1) ;  } | 그림입니다.  원본 그림의 이름: mem0000211c0007.png  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면(flush() 사용) |
| char \*buf = "This message for checkint a time";  void setup() {    Serial.begin(9600);  }    void loop() {    unsigned long int atime;    for(int i = 0; i <10; i++) {      atime = micros();      Serial.println(buf);      Serial.flush();      Serial.println(micros() - atime);    }    while(1) ;  } | 그림입니다.  원본 그림의 이름: mem0000211c0008.png  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.flush() 함수는 현재 전송하고 있는 시리얼 데이터가 전송완료가 될 때까지 기다리는 함수 입니다.(참고로 아두이노 1.0버전 이전은 버퍼에 들어온 데이터를 삭제하는 함수입니다.)

위의 두 예제를 보시면, 위의 예제는 flush()함수를 사용하지 않은 예제, 밑의 예제는 flush()함수를 사용한 예제 입니다.

atime = micros();

Serial.println(buf);

Serial.println(micros() - atime);

위의 코드 세줄은 Serial.println(buf);를 실행하는데 걸리는 시간을 확인하는 부분입니다. 위와 아래의 예제를 비교하면 Serial.flush()의 사용 유무의 따른 차이를 확인 할 수 있습니다.

9. Serial.find()

사용방법

 - Serial.find(target)

**변수**

 - target : 검색할 String값 문자

**반환**

  - 읽어온 데이터 값 중 타겟이 있으면 true, 없으면 false : boolean값 리턴

예제 소스 코드(비교)

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  long temp = 0;  void loop(){    while(Serial.available()){      if(Serial.find("f")){          temp = Serial.parseInt();      }else{        temp = 0;      }            Serial.print("Temp Value = ");      Serial.println(temp);    }  } | 그림입니다.  원본 그림의 이름: mem0000211c0009.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.find()함수는 지정된 길이의 대상 문자열이 발견 될 때까지 직렬 버퍼로 부터 데이터를 확인하는 함수입니다.

해당 문자열이 발견이 되면 True값을 리턴, 설정한 타임아웃 시간 까지 해당 문자열을 발견하지 못할 경우 False값을 리턴 합니다.

위의 예제 코드는 "f" 값이 들어올 경우 뒤의 값을 int로 받아서 temp값에 저장, 이 외에 다른 값이 들어오면 temp값을 0으로 초기화 한 후 저장하여 저장된 temp값을 시리얼 모니터에 출력해 주는 예제 입니다.

10. Serial.findUntil()

사용방법

 - Serial.findUntil(target, terminal)

**변수**

 - target : 검색할 String값 문자

 - terminal : 검색을 종료할 String값 문자

**반환**

  - 읽어온 데이터 값 중 타겟이 있으면 true, 없으면 false : boolean값 리턴

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  long temp = 0;  void loop(){    while(Serial.available()){       Serial.println(Serial.findUntil("f", "T"));    }  } | 그림입니다.  원본 그림의 이름: mem0000211c000a.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.findUntil()함수는 Serial.find()함수와 같이 target문자열을 검색하는 함수입니다. find함수와 다른점은 종료 문자열을 설정하여 종료문자열을 찾을 때까지만 데이터를 검색합니다.

해당 문자열이 발견이 되면 True값을 리턴, 종료문자열을 발견하거나, 설정한 타임아웃 시간 까지 해당 문자열을 발견하지 못할 경우 False값을 리턴 합니다.

위의 예제 코드는 입력 받은 문자열 중 'f'값이 있는지 확인하는 예제입니다. 종료 문자열을 'T'로 설정했습니다. 처음에 'asdqwe'만 입력했을 경우는 기본 설정된 타임아웃 시간까지 기다렸다가 결과 값 0이 나오지만, 'asdqwe**T**'를 입력 했을 경우 타임아웃 시간까지 대기하지 않고 바로 결과 값 0이 나오는 것을 확인 할 수 있습니다.

 \* 'f'가 들어 갈 경우 f 앞뒤로 나뉘어서 검색을 하게 되므로 결과 값이 1, 0 2개가 나오게 됩니다.

11. Serial.peek()

사용방법

 - Serial.peek()

**반환**

  - 읽어온 데이터 값 중 첫 번째 값

예제 소스 코드(비교)

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면(read() 사용) |
| void setup(){    Serial.begin(9600);  }  long temp = 0;  void loop(){    if(Serial.available()){       Serial.println(Serial.read());    }  } | 그림입니다.  원본 그림의 이름: mem0000211c000b.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면(peek() 사용) |
| void setup(){    Serial.begin(9600);  }  long temp = 0;  void loop(){    if(Serial.available()){       Serial.println(Serial.peek());    }  } | 그림입니다.  원본 그림의 이름: mem0000211c000c.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

Serial.peek()함수는 Serial.read()함수와 같이 데이터를 읽어오는 함수 입니다. read()와 차이점은 read()함수는 데이터를 읽어오면 읽은 데이터는 버퍼에서 삭제하는데 반해, peek()함수는 시리얼 버퍼에 그대로 데이터를 저장합니다.

위의 예제 코드는 Serial.read()와 Serial.peek()함수의 차이점을 보여주는 예제입니다. read()함수를 사용할 경우 데이터를 읽은 후 시리얼 버퍼에서 지우기 때문에 다음에 호출하면 그 다음 값이 나오는데, peek()함수를 사용하면 시리얼 버퍼에서 데이터를 지우지 않기 때문에 다시 peek()함수를 호출하더라도 제일 처음 값 'a'값이 계속 나오게 됩니다.(시리얼 모니터에 출력되는 값은 ASCII코드 값으로 'a' = 97입니다.)

12. serialEvent()

예제 소스 코드

|  |  |
| --- | --- |
| 소스 코드 | 결과 화면 |
| void setup(){    Serial.begin(9600);  }  void loop(){  }  void serialEvent(){    char text = Serial.read();    Serial.print("data : ");    Serial.println(text);  } | 그림입니다.  원본 그림의 이름: mem0000211c000d.gif  원본 그림의 크기: 가로 364pixel, 세로 498pixel |

설명

serialEvent() 함수는 따로 어떤 용도로 인해 호출하는 함수가 아니라 시리얼 통신을 통해서 데이터가 입력될 때 자동으로 호출되는 함수 입니다. 기본적으로 정의하지 않아도 상관없지만, 위의 예제코드와 같이 serialEvent()함수를 정의해주면 시리얼 통신을 통해서 데이터가 입력 될 때 serialEvent함수가 호출되는 것을 확인할 수 있습니다.

위의 예제 코드를 보시면 주된 동작을 실행하는 loop()함수 안에 다른 코드 없이 serialEvent()함수가 실행되는 것을 확인 할 수 있습니다.