

C W/ Arduino

## 아두이노 개요

## 전기,전자회로

## 디지털 입출력

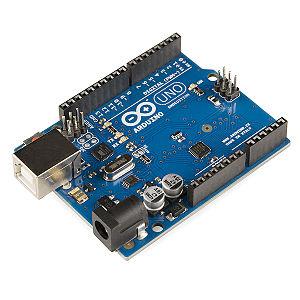
## Serial 통신

## Interrupt(인터럽트)

## 아날로그 입출력

## 센서와 액츄에이터 연동

# 아두이노 개요



**<arduino uno R3>**

**아두이노**([이탈리아어](https://ko.wikipedia.org/wiki/%EC%9D%B4%ED%83%88%EB%A6%AC%EC%95%84%EC%96%B4): Arduino 아르두이노)는 [오픈 소스](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88_%EC%86%8C%EC%8A%A4)를 기반으로 한 단일 보드 마이크로컨트롤러로 구성된 보드와 이를 개발하는 도구 및 환경을 말한다.

2005년[이탈리아](https://ko.wikipedia.org/wiki/%EC%9D%B4%ED%83%88%EB%A6%AC%EC%95%84)의 IDII(Interaction Design Institutelvera)에서 하드웨어에 익숙지 않은 학생들이 자신들의 디자인 작품을 손쉽게 제어할 수 있도록 하기 위해 고안되었다.

아두이노는 다수의 스위치나 센서로부터 값을 받아들여, [LED](https://ko.wikipedia.org/wiki/LED)나 모터와 같은 외부 전자 장치들을 콘트롤함으로써 **환경**과 상호작용이 가능한 물건을 만들어 낼 수 있고 더 나아가 [임베디드 시스템](https://ko.wikipedia.org/wiki/%EC%9E%84%EB%B2%A0%EB%94%94%EB%93%9C_%EC%8B%9C%EC%8A%A4%ED%85%9C) 환경으로써도 이용 가능하다.

아두이노는 컴파일된 펌웨어를 USB를 통해 쉽게 업로드 함으로써 마이크로컨트롤러를 쉽게 동작시킬 수 있다. 또한 아두이노는 다른 모듈에 비해 비교적 저렴하고, [윈도](https://ko.wikipedia.org/wiki/%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C%EC%86%8C%ED%94%84%ED%8A%B8_%EC%9C%88%EB%8F%84)를 비롯해 [맥 OS X](https://ko.wikipedia.org/wiki/%EB%A7%A5_OS_X), [리눅스](https://ko.wikipedia.org/wiki/%EB%A6%AC%EB%88%85%EC%8A%A4)와 같은 여러 [OS](https://ko.wikipedia.org/wiki/OS)를 모두 지원한다.

아두이노 보드의 회로도가 CCL에 따라 공개되어 있으므로, 누구나 직접 보드를 만들고 수정할 수 있다.

##### **아두이노 소프트웨어 설치**

##### <https://www.arduino.cc/> > <https://www.arduino.cc/en/Main/Software> 로 이동하여 각자의 PC 환경에 맞는 소프트웨어 버전을 선택하여 설치한다.

##### 



# 2. 전기, 전자 회로

|  |  |
| --- | --- |
| **전압**  **전류** | 전기 회로에 전류를 흐르게 하는 능력을 **전압**이라고 하며, 단위는 **V**(볼트)를 사용한다.  **전류**는 전자들의 흐름을 의미하며, 단위는 **A**(암페어)를 사용한다.  물의 흐름과 전기 회로의 비유(수압과 전압의 비교)  물의 높이 차에 의한 수압에 의해 물이 흐르듯이 전지의 전압에 의해 전류가 흐른다.  ① 물의 흐름: 펌프가 물을 높은 곳으로 끌어올리면 물의 높이 차(수압)에 의해 수도관에 물이 흐르면서 물레방아가 돌아간다.  물의흐름과전류.jpg  **• 물의 흐름 - 전류**  **• 물레방아 - 전구**  **• 밸브 - 스위치**  **• 수도관 - 도선(전선)**  **• 펌프 - 전지**  **• 물의 높이 차(수압) - 전압**  **②** 전기 회로: 전지의 전압에 의해 도선에 전류가 흘러 전구에 불이 켜진다.  물의흐름과전류2.jpg  [**물의 흐름과 전기회로 비유**](http://tvcast.naver.com/v/767202)**(http://tvcast.naver.com/v/767202)**  **[네이버 지식백과]** [**전압**](http://terms.naver.com/entry.nhn?docId=3344536) **(비상학습백과 중학교 과학 ③, 비상교육) 참조, 발췌** |
| **저항** | 전기 회로에서 전류의 흐름을 방해하는 정도로, 단위로는 **Ω**(옴)을 사용한다.  오디오의 볼륨 스위치를 돌리면 소리가 커지기도 하고 작아지기도 한다. 또 선풍기의 ‘강, 약’ 스위치를 누르면 바람의 세기가 세지기도 하고 약해지기도 한다. 이는 오디오나 선풍기의 저항이 달라졌기 때문이다.  물의흐름과전류3.jpg  경사면을 내려오는 구슬이 못과 충돌하여 이동을 방해받는 것에 비유할 수 있는데, 못이 많아진다면 구슬이 방해를 많이 받을 것이므로, 빨리 굴러 내려갈 수 없다.  못이 박혀있는 정도를 저항과 비유하면, 많이 박혀 있으면 저항이 크고, 적게 박혀 있으면 저항이 작다고 말할 수 있다.  **[네이버 지식백과] 저항 (비상학습백과 중학교 과학 ③, 비상교육) 참조, 발췌** |
| **오옴의 법칙** | 전기 회로에서 전지의 전압이 달라지면 같은 전구라도 밝기가 달라진다.  또 전압이 같더라도 전구의 저항에 따라 밝기가 달라진다.  이는 전압은 전류가 흐르도록 도와주고, 저항은 전류의 흐름을 방해하기 때문이다. 이와 같은 전압, 전류, 저항 사이의 관계를 설명하는 법칙이 **옴의 법칙**이다.  **V(전압) = I(전류) \* R(저항)**  오옴의법칙.jpg  전류의 세기(*I*)는 전압(*V*)에 비례하고, 저항(*R*)에 반비례한다.  즉, 전류는 전압이 클수록 많이 흐르고, 전압이 약할수록 작게 흐르며,  저항이 클수록 작게 흐르고, 저항이 작을수록 많이 흐른다.  **[네이버 지식백과]** [옴의 법칙](http://terms.naver.com/entry.nhn?docId=3344538) (비상학습백과 중학교 과학 ③, 비상교육) |
| **아날로그**  **(Analog)** | 전압이나 전류처럼 연속적으로 변화하는 물리량을 표현한다. 사람의 목소리와 같이 연속적으로 변하는 신호는 아날로그 형태이며 그 양을 계량할 수 있다.  예를 들어 시침과 분침이 돌아가는 시계와 시간이 숫자로 나타나는 시계를 연상해 보자. 시침이나 분침은 연속해서 움직이지만 어느 순간의 시간(양)을 표시해 준다. 그러나 숫자로 나타나는 시계는 시간의 표현이 단속적이며 계수적이다.  이때 바늘이 돌아가는 시계를 아날로그 시계, 숫자가 나타나는 시계를 디지털 시계라고 할 수 있다.  **[네이버 지식백과]** [아날로그](http://terms.naver.com/entry.nhn?docId=1178901) [analog] (두산백과) |
| **디지털**  **(Digital)** | 디지트(digit)는 사람의 손가락이나 동물의 발가락이라는 의미에서 유래한 말이다. 아날로그와 대응하며, 임의의 시간에서의 값이 최소값의 정수배로 되어 있고 그 이외의 중간 값을 취하지 않는 양을 가리킨다.  구체적인 예로 디지털시계의 표시를 들 수 있는데, 시계가 바늘로써 연속적으로 시간을 표시하는 것이 아니라 시 ·분 ·초 등으로 구획하여 문자로 표시한다. 따라서, 디지털이란 일반적으로 [데이터](http://terms.naver.com/entry.nhn?docId=1082441&ref=y)를 한 자리씩 끊어서 다루는 방식이라 할 수 있으며, 애매모호한 점이 없고, 정밀도를 높일 수 있다는 특징이 있다.  **[네이버 지식백과]** [디지털](http://terms.naver.com/entry.nhn?docId=1086177) [digital] (두산백과) |

##### **아두이노로 개발하기**

|  |  |
| --- | --- |
| 물의흐름과전류2.jpg | 왼쪽과 같은 전기회로를 아두이노로 만든다고 생각해 보자.스위치를 누르면 전류가 통하게 되어 전구에 불이 들어 오고, 스위치에서 손을 떼게 되면 전류가 흐르지 않게 되어 전구에 불은 꺼지게 된다.만일, 위에서 전지와 스위치 사이에 빛의 밝기를 탐지할 수 있는 조도센서라는 입력장치와 아두이노를 달아서 현재 빛의 밝기를 측정한 후 빛의 밝기에 따라서 스위치를 눌렀다가 뗐다가를 자동으로 처리할 수 있게 한다면 대단히 편리한 자동화시스템이 될 것이다. |

##### arduino.JPG

##### 즉, 외부 환경의 변화에 따라 사람이 일일이 판단하여 어떤 조치를 취하는 일이 있다고 할 때, 더군다나 그러한 판단과 조치가 반복적으로 발생하지만 불규칙적이라고 할 때,

##### 브레드보드 라고 하는 전자부품에 센서와 액츄에이터를 적절히 배치하고 이에 대한 통제를 사람이 미리 프로그램해 둔 아두이노가 맡도록 하는 일련의 과정을 “**아두이노 개발과정**”으로 이해하면 되겠다.

|  |  |
| --- | --- |
| 디지털 입력  Digital Input | **digitalRead(핀번호)** 가능포트 : **D0~D13, A0~A5**  0V 일때 LOW, 5V 일때 HIGH  0V 를 GND(Ground,접지)라고 표현.  예)  digitalRead(3) == HIGH : D3번 포트의 측정 전압이 5V.  digitalRead(3) == LOW : D3번 포트의 측정 전압이 0V.  D3번 포트를 스위치로 생각했을 때, 전체회로 중 D3의 스위치가 눌려져 있다면 5V, 스위치가 눌려져 있지 않다면 0V |
| 디지털 출력  Digital Output | **digitalWrite(핀번호, HIGH/LOW)** 가능포트 : **D0~D13, A0~A5**  예)  digitalWrite(3,HIGH) : D3번 포트에 5V 전압 인가.  digitalWrite(3,LOW) : D3번 포트에 0V 전압 인가.  D3번 포트를 스위치로 생각했을 때, 전체회로중 D3의 스위치를 눌러서 5V 전압을 인가시키는 것 |
| 아날로그 입력  (ADC)  Analog Digital Converter | **analogRead(A0~A5) : 0~1023** 가능포트 : **A0,A1,A2,A3,A4,A5**  예)  analogRead(A0) : A0포트의 전압값(0~5V)을 읽어서 이를 1024개 단위로 쪼개어 각 센서가 보내는 측정값으로 처리.  핀번호 A0번에서 2.5V의 전압이 측정되면, A0번 포트에 연결된 센서에서 511(50%) 의 신호를 보내고 있는 것으로 판단함. |
| 아날로그 출력  (DAC)  Digital Analog Converter | **analogWrite(핀번호, 값) : 0~255** 가능포트 : **A3,A5,A6,A9,A10,A11 (~표시)**  예)  analogWrite(A3) : A3 포트에 0V ~ 5V 의 전압을 ***PWM***으로 256개 단위로 쪼개어 포트에 부하를 가함.  상세내용은 PWM 참조 . |
| 브레드보드  (bread board) | bread1.PNG 아두이노에서 납땜과정 없이 프로토 타입의 회로를 만드는 토대가 되는 전자부품이다.위에서 보는 바와 같이 상단과 하단의 수평라인은 모두 하나의 도선으로 연결되어 있고, 가운데 수직라인 또한 각각 연결되어 있다.상단과 하단의 수평라인은 주로 전원과 GND로 사용되고, 수직라인의 구멍에 구성하고자 하는 부품을 꼽아 회로를 설계하면 된다. [아두이노를 전원 공급 용도로만 연결한 후 회로 만들어 보기 ]  아두이노전원만공급회로.png  [확대한 구성도]  아두이노전원만공급회로1.png  위와 같이 회로를 만들어 보고, 스위치를 누르면 LED에 불이 들어 온다.  LED의 경우 긴쪽이 +극(anode), 짧은 쪽이 -극(cathode)  [저항값 결정]  LED의 데이터시트([datasheet](http://www.alldatasheet.co.kr/))를 살펴보면 최대 허용 전류는 20mA 이고 전류가 흐를 때 LED 양단의 전압 강하는 3V 이다.  아두이노가 5V 전원을 사용하므로 최대 전류를 흘릴 때 전압강하가 3V라고 가정하면 저항값은 오옴의 법칙에 의해서 다음과 같이 간단히 계산할 수 있다.    이 저항값이 허용되는 가장 최소 저항이므로 이것보다 큰 용량의 저항을 선택하면 구동하는데 무리가 없을 것이다.  [저항 색깔로 저항값 알아보는 방법]  resistance-gongbuman_preparetowin.jpg |
| Arduino uno R3 | 아두이노우노.png그림의 오른쪽 헤더핀 중에서 0번부터 13번까지가 디지털 포트에 해당한다. 디지털 입출력 포트는 디지털 신호의 입력과 출력을 담당하는 통로로서, 전기가 통할 경우 HIGH(1), 통하지 않을 경우 LOW(0)의 상태를 나타낸다.D0,D1 : serial 통신D2,D3 : INT0, INT1 (인터럽트)D3,D5,D6,D9,D10,D11 : PWM DAC, ~로 표시D13은 저항이 내장되어 있으므로 digital input 사용에 유의A0,A1,A2,A3,A4,A5 : 아날로그 입력 포트 , 디지털포트로 사용 가능 |

# 3. 디지털 입출력

## **디지털 출력 : LED**

##### 아두이노 우노에는 실습용 LED가 내장되어 있는데 이것은 13번 핀과 연결되어 있고, 자체 저항(220**Ω)**도 가지고 있으므로 브레드보드에 저항과 LED 연결없이 바로 사용 가능하다.

##### (아래 빨간 원이 LED임)

##### 아두이노 스케치 프로그램을 실행 한 후에, 위와 같이 파일>예제>Basics>Blink 파일을 선택하여 읽어들인다.

|  |  |
| --- | --- |
| 디폴트13led.png | 블링크.PNG |

|  |  |
| --- | --- |
| int led = 13;void setup() { pinMode(led, OUTPUT);void loop() { digitalWrite(led, HIGH); delay(1000); digitalWrite(led, LOW); delay(1000);} | **setup() 함수**  아두이노 프로그램이 실행될 때 맨 처음에 단 한 번 호출되어 실행  (단, 시리얼모니터를 실행될 때에도 실행)  각종 장치를 초기화하거나 초기값을 설정하는데 사용.  **loop() 함수**  setup()이 실행된 이후 반복적으로 실행됨.  아두이노에 연결된 sensor나 actuator들을 구동시키는 코드가 위치  **pinMode() 함수**  pinMode(핀번호, INPUT/OUTPUT)  디지털 핀을 입력, 출력으로 사용할 지 설정  (아날로그 핀의 경우 실행할 필요가 없다)  **digitalWrite() 함수**  digitalWrite(핀번호, LOW/HIGH)  핀으로 출력값을 내보내는 작업을 수행  **delay() 함수**  delay(밀리세컨드: ms)  입력된 시간만큼 아무 일도 안하고 멈춰있는 동작을 수행 |

## **디지털 출력 : Buzzer**

##### 부저(buzzer)는 소리를 내는 전자부품으로서 2가지 종류가 있다.

##### 수동부저(passive buzzer) : tone() 함수를 이용해 소리 파형을 만들어 사용

##### 능동부저(active buzzer) : 내부에 회로가 내장되어 있어서 digitalWrite() 함수를 이용해 전압을 걸면 소리 발생

##### 

##### 

##### 능동 부저

##### 

##### 부저는 극성이 있으며 조금 긴 다리가 (+)극이고 짧은 쪽은 (-)이다. + 극성에 표기되어 있음.

##### 아두이노 우노의 11번 핀에 부저의 (+)를 꽂으면 (-)핀은 GND에 들어맞게 되므로, 브레드보드와의 연결 없이 간단하게 테스트 가능하다.

|  |  |
| --- | --- |
| 부저.ps.png | |
| #define BUZ 11void setup() { pinMode(BUZ, OUTPUT);}void loop() { tone(BUZ,1000); delay(50); //0.005초 동안 정지 noTone(BUZ); delay(1000); //1초 동안 정지} | tone(pin, freq [, duration]);   * pin : 부저가 연결된 핀번호 * freq : 주파수 (범위 : 31 ~ 65535) * duration : (옵션) 음의 발생 시간   duration은 주파수 지속시간으로서 이 시간이 지난 후에 noTone()함수가 자동으로 호출된다.  duration이 생략되면 noTone()함수가 호출될 때까지 음이 계속 발생  tone()함수는 하나의 주파수만 발생시킬 수 있으며 여러 핀에 동시에 다른 음을 발생시킬 수 없다 |

##### 

|  |
| --- |
| **변수 ( variable )** |
| -값(상수값)을 저장하는 장소  -자료형 변수명 의 형식으로 선언, 변수명은 반드시 영어 알파벳으로 시작해야 한다.  - int : 정수, float: 실수, boolean: true or false  ex)  int a=10; // 정수를 저장하는 장소로 a를 선언하고 상수값으로10을 저장.  float b = 58.7; // 실수를 저장하는 장소로 b를 선언하고 상수값으로 58.7을 저장. |
| **배열 ( array )** |
| -같은 자료형 변수들의 모임  -메모리의 번지(주소)를 연속적으로 할당 받는다.  -배열을 만든 후 배열의 요소들을 구분할때 인덱스를 이용하며 인덱스는 [0] 부터 [개수-1] 까지 사용.  ex) a배열을 3개 만들었을 경우 --> a[0] , a[1] , a[2]  array.JPG |
| **조건문 ( if ~ else / switch ~ case )** |
| if.JPG  switch.JPG |
| 반복문 ( for ) |
| for.JPG |
| 함수 ( function ) |
| -미리 만들어 놓은 작은 프로그램으로 혼자서 실행되지 않고 다른 함수에 의해서 호출을 받아야만 실행된다.-리턴을 하지 않는 경우에는 자료형 자리에 void 를 사용한다. 함수.JPG  함수2.JPG |

|  |  |
| --- | --- |
| int BUZ = 11;int note[] = {2093, 2349, 2637, 2793, 3136, 3520, 3951, 4186}; //도레미파솔라시도void setup() { pinMode(BUZ, OUTPUT); melody();}void loop() {}void melody() { int nSize = sizeof(note)/sizeof(int); int i; for (i = 0; i < nSize; i++) { tone(BUZ, note[i]); delay(500); noTone(BUZ); } for (i = nSize-1; i >=0 ; i--) { tone(BUZ, note[i]); delay(500); noTone(BUZ); } noTone(BUZ);} | ‘도’음을 발생시키려면,tone(11, 2093);delay(500);noTone(11);또는tone(11, 2093, 500);를 실행.이렇게 하면 음 발생 직후 다른 작업을 수행할 수 있으며 500ms후에는 내부 인터럽트가 발생하여 음발생이 자동으로 정지된다. 하지만 단음만 생성할 경우에는 상관없지만 연속음을 생성할 경우에는 주의해야 한다.tone(11, 262, 500); // ‘도’음tone(11, 294, 500); // ‘레’음tone(11, 330, 500); // ‘미’음위와 같이 실행하면 맨 마지막의 ‘미’음만 발생하게 된다. 따라서 올바른 동작을 위해서는 다음과 같이 해야 한다.tone(11, 262); // ‘도’음 발생delay(500);tone(11, 294); // ‘레’음 발생delay(500);tone(11, 330); // ‘미’음 발생delay(500);noTone(11); |
| #include "pitches.h"int BUZ = 11;void setup() { schoolSong();}void loop() {}void schoolSong() {//음계 데이터 (24개)int melody[] = {NOTE\_G7,NOTE\_G7,NOTE\_A7,NOTE\_A7,NOTE\_G7, NOTE\_G7,NOTE\_E7,NOTE\_G7,NOTE\_G7,NOTE\_E7, NOTE\_E7,NOTE\_D7,NOTE\_G7,NOTE\_G7,NOTE\_A7, NOTE\_A7,NOTE\_G7,NOTE\_G7,NOTE\_E7,NOTE\_G7, NOTE\_E7,NOTE\_D7,NOTE\_E7,NOTE\_C7};//음의길이, 4:4분음표, 2:2분음표int noteDurations[] = {4,4,4,4,4,4,2,4,4,4,4,1,4,4,4,4,4,4,2,4,4,4,4,1}; int i; for (i = 0; i < 24; i++) { int noteDuration = 1000 / noteDurations[i]; tone(BUZ, melody[i]); int pauseBetweenNotes = noteDuration \* 1.30; delay(pauseBetweenNotes); noTone(BUZ); }} | 아두이노가 연주하는 “학교종”을 감상해보자  특정 음에 해당되는 주파수를 일일이 숫자로 기입하는 것은 가독성 면에서 좋지 않으며 이를 개선하기 위해서 아두이노에서는 음의 주파수만을 모아놓은 “pitches.h”라는 헤더파일을 제공  tone()함수는 내부적으로 타이머를 사용하므로 다음과 같은 점들을 주의해야 한다.   * 이 함수를 사용할 경우 3번 11번 핀의 PWM이 정상적으로 동작되지 않음.. * 한 번에 하나의 음만 출력 가능 |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Public Constants\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/#define NOTE\_B0 31#define NOTE\_C1 33#define NOTE\_CS1 35#define NOTE\_D1 37#define NOTE\_DS1 39#define NOTE\_E1 41#define NOTE\_F1 44#define NOTE\_FS1 46#define NOTE\_G1 49#define NOTE\_GS1 52#define NOTE\_A1 55#define NOTE\_AS1 58#define NOTE\_B1 62#define NOTE\_C2 65#define NOTE\_CS2 69#define NOTE\_D2 73#define NOTE\_DS2 78#define NOTE\_E2 82#define NOTE\_F2 87#define NOTE\_FS2 93#define NOTE\_G2 98#define NOTE\_GS2 104#define NOTE\_A2 110#define NOTE\_AS2 117#define NOTE\_B2 123#define NOTE\_C3 131#define NOTE\_CS3 139#define NOTE\_D3 147#define NOTE\_DS3 156#define NOTE\_E3 165#define NOTE\_F3 175#define NOTE\_FS3 185#define NOTE\_G3 196#define NOTE\_GS3 208#define NOTE\_A3 220#define NOTE\_AS3 233#define NOTE\_B3 247#define NOTE\_C4 262#define NOTE\_CS4 277#define NOTE\_D4 294#define NOTE\_DS4 311#define NOTE\_E4 330#define NOTE\_F4 349#define NOTE\_FS4 370#define NOTE\_G4 392#define NOTE\_GS4 415#define NOTE\_A4 440#define NOTE\_AS4 466#define NOTE\_B4 494#define NOTE\_C5 523#define NOTE\_CS5 554#define NOTE\_D5 587#define NOTE\_DS5 622#define NOTE\_E5 659#define NOTE\_F5 698#define NOTE\_FS5 740#define NOTE\_G5 784#define NOTE\_GS5 831#define NOTE\_A5 880#define NOTE\_AS5 932#define NOTE\_B5 988#define NOTE\_C6 1047#define NOTE\_CS6 1109#define NOTE\_D6 1175#define NOTE\_DS6 1245#define NOTE\_E6 1319#define NOTE\_F6 1397#define NOTE\_FS6 1480#define NOTE\_G6 1568#define NOTE\_GS6 1661#define NOTE\_A6 1760#define NOTE\_AS6 1865#define NOTE\_B6 1976#define NOTE\_C7 2093#define NOTE\_CS7 2217#define NOTE\_D7 2349#define NOTE\_DS7 2489#define NOTE\_E7 2637#define NOTE\_F7 2794#define NOTE\_FS7 2960#define NOTE\_G7 3136#define NOTE\_GS7 3322#define NOTE\_A7 3520#define NOTE\_AS7 3729#define NOTE\_B7 3951#define NOTE\_C8 4186#define NOTE\_CS8 4435#define NOTE\_D8 4699#define NOTE\_DS8 4978 | |

## 

## 

## **디지털 입력 : Tact switch**

|  |  |
| --- | --- |
|  | 택스위치(tact switch)는 기계적인 접점을 가지는 스위치로서 누르면 접점이 닫히고 누르지 않은 상태에서는 접점이 열려있는 소자이다.기계적 접점을 가짐으로 인해서 회로 구성시 PullDown, PullUp 저항의 구성과 바운싱 현상에 대한 처리가 필요하다. |

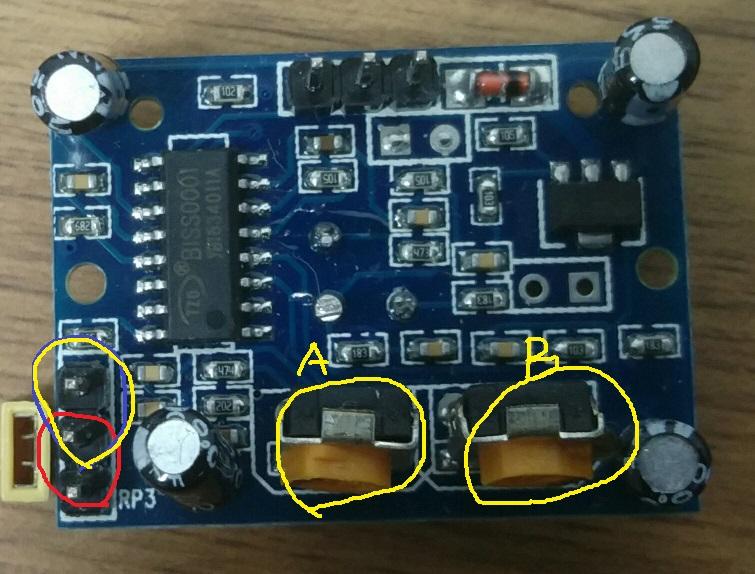
|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| **Floating state(플로팅 상태)**  스위치와 같은 전자 부품은 누르지 않았을 때에는 회로 자체가 연결된 상태가 아니기 때문에 전기적으로 on/off 상태를 정확하게 판단할 수 없다. 이와 같이 0이나 1로 정확하게 구분하지 못하는 상태를 플로팅 상태라고 한다.  만일 아두이노의 sensor로서 스위치가 사용될 경우 HIGH도 아니고 LOW도 아닌 상태라면 이에 따른 정확한 통제가 불가능할 것이므로 풀업이나 풀다운 저항을 이용하여 이런 플로팅 상태를 방지할 수 있다. | |
| pullup.png | PULLUP 저항 : 스위치가 열려 있을 때 전압을 HIGH로 만들기 위한 저항 배치  입력핀은 스위치와 저항 사이의 전압을 체크하여 스위치 on, off 값이 LOW, HIGH 둘 중 하나의 값만을 가지게 하여 floating상태를 막는 것이 목적이다.   * 스위치 off : input pin 5V (저항에 의한 전압강하가 발생되나 3V이상이면 HIGH로 인식) * 스위치 on : input pin 0V (GND와 연결) * 보통 10k**Ω**을 사용 |
| 82.png | PULLDOWN 저항 : 스위치가 열려 있을 때 전압을 LOW로 만들기 위한 저항 배치  입력핀은 스위치와 저항 사이의 전압을 체크   * 스위치 off : input pin 0V (GND와 연결) * 스위치 on : input pin 5V * 보통 100**Ω**-1k**Ω**을 사용 |
| button_led_pullup_bb.ps.png | |
| const int buttonPin = 2;  const int ledPin = 13;  int buttonState = 0;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT);  pinMode(ledPin, HIGH);  }  void loop() {  buttonState = digitalRead(buttonPin);  if (buttonState == LOW) {  digitalWrite(ledPin, HIGH);  } else {  digitalWrite(ledPin, LOW);  }  } | 택 스위치와 LED로 구성된 회로    기계적인 스위치를 핀에 연결할 때는 보통 내부 풀업저항을 연결해서 사용하는데,  내부에 풀업 저항이 연결되어 있으므로 스위치가 안 눌렸으면(switch off) 2번 핀으로 HIGH신호가 나오고, 눌렸다면(switch on) LOW 신호가 나온다. 따라서 이 디지털 신호를 읽으면 택스위치가 눌렸는지 아닌지를 판별할 수 있다. |

|  |  |
| --- | --- |
| 아두이노에서 기계적인 스위치 사용시 발생하는 floating 상태를 방지하기 위해서pinMode(핀번호,INPUT\_PULLUP); 를 제공하므로회로에 스위치가 필요한 경우 풀업, 풀다운 저항등을 따로 구성할 필요가는 없다. | |
| button_led_pullup_bb.png | const int buttonPin = 2;  const int ledPin = 13;  int buttonState = 0;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT\_PULLUP);  pinMode(ledPin, HIGH);  }  void loop() {  buttonState = digitalRead(buttonPin);  if (buttonState == LOW) {  digitalWrite(ledPin, HIGH);  } else {  digitalWrite(ledPin, LOW);  }  } |

## **디지털 입력 : PIR(인체 감지 센서)**

##### 인체감지센서 모듈은 적외선을 감지하는 센서를 장착하고 있으며 약 5미터 이내의 인간이나 동물의 움직임을 감지할 수 있다. 인체나 동물의 몸과 같이 열을 발산하는 곳에서 발생하는 적외선을 감지하는 원리이다. 전원과 신호를 전달하는 세 핀을 가지고 있고 5V 전원을 받아서 인체가 감지되면 신호선은 약 3.3V 전압을 띠게된다. 따라서 신호선을 아두이노의 디지털 핀에 연결하면 쉽게 인체감지 여부를 알 수 있게 된다.



* 단일감지 : 왼쪽에 보이는 노란색 점퍼를 빨간색 두 개의 점퍼에 연결하는 경우는 Delay time 에는 감지하지 않는다.
* 반복감지 : 왼쪽에 보이는 노란색 점퍼를 노란색 두 개의 점퍼에 연결하는 경우에는 Delay time 에도 감지 한다.
* A 부분 : 감도 조절, CW: 7m, CCW: 3m
* B 부분 : Delay time 조절, CW: 300s, CCW: 5s

|  |  |
| --- | --- |
| led+pir_bb.jpg | |
| #define LED 11#define PIR 4int motion=0;int pirStatus = 0;int val;void setup() { pinMode(PIR, INPUT); pinMode(LED, OUTPUT); Serial.begin(9600);}void loop() { val = digitalRead(PIR); if(val == 1) { if(pirStatus == 0) { digitalWrite(LED, HIGH ); pirStatus = 1; motion++; Serial.print(motion); Serial.println(" motion detected"); } } else { digitalWrite(LED, LOW ); pirStatus = 0; }} | pirStatus의 용도  모션이 감지되면 LED를 밝힌다.  이때의 조건은 pirStatus가 0일때만 LED를 밝히게 된다.  만일, pirStatus가 없다면  PIR sensor의 감도나 지연시간 설정에 상관없이 모션이 감지될 때마다 LED가 껴지고, motion 수가 증가하게 될 것이므로, 정확한 모션 인식이 힘들어진다.  다시 말해서,  모션을 감지하는 처리가 대단히 빠르게 진행되는 상황에서 우리가 집중해야 하는 부분은 pirStatus가 0에서 1로 바뀌는 부분이고 이에 대한 정확한 처리를 위해서 pirStatus가 필요하다.  Serial. 과 관련된 설명은 다음 장에서 진행한다. |

# 4. 시리얼통신

##### 시리얼 통신(serial communication)은 기기들간 데이터를 주고 받는 방법 중 하나인데 병렬 통신 (parallel communication) 방식에 비해서 통신선의 갯수가 적다는 장점이 있다. 아두이노는 UART (universal asynchronous receiver and transmitter), SPI, I2C 방식의 시리얼 통신을 지원하는데 이중 UART는 주로 아두이노와 PC간의 통신을 하는데 사용된다. 아두이노는 하나 이상의 UART 통신기가 내장되어 있는데 라이브러리와 IDE에 내장된 시리얼 모니터 프로그램을 이용하면 손쉽게 PC와의 통신을 수행할 수 있다.

##### 

##### 아두이노 우노의 경우 0번과 1번핀이 시리얼 통신에 사용된다. 이 핀들은 내부적으로 USB통신을 담당하는 칩과 연결되어서 USB신호로 변환된 후 PC에 전달된다. 반대로 PC에서 보내지는 USB신호는 이 칩에서 시리얼 통신 신호로 변환되어 아두이노의 AVR에 전달된다. 따라서 만약 아두이노가 PC와의 통신을 수행하고 있다면 이 핀들을 다른 용도로 사용하면 안된다. 그리고 통신을 수행할 때에는 TX, RX라고 마킹된 LED가 깜빡이는 것을 확인할 수 있다.

##### 아두이노와 PC간에 USB케이블로 연결하는 것으로 준비가 완료된다.

* **시리얼 통신 1**

|  |  |
| --- | --- |
| 아두이노에게 전원을 인가하면 “I’m ready.” 라는 메세지를 PC에 전송하는 예  시리얼1.PNG | |
| void setup() { Serial.begin(9600); Serial.print("I'm ready.");}void loop() {} | 왼쪽 코드를 스케치에서 붙여 넣고, 컴파일과 업로드를 실행한 후 시리얼모니터를 실행한다. |
| **void Serial.begin(long baud\_rate)**   * UART 통신을 초기화 시킨다. * 통신 속도(baud rate)를 입력으로 받는다. * 9600, 19200, 57600, 115200 등 여러 baud rate를 지원한다.   **long Serial.print(val)**   * 입력값을 ASCII값으로 변환하여 PC쪽으로 출력한다. * 전송된 데이터의 바이트 수를 리턴한다. * 비동기 통신 방식이므로 데이터가 전송되기 전에 리턴된다. * 입력 변수 val은 어떤 데이터 타입도 가능하다.   + Serial.print(78) gives "78"   + Serial.print(1.23456) gives "1.23"   + Serial.print('N') gives "N"   + Serial.print("Hello world.") gives "Hello world." * 두 번째 인수로 출력 형식을 지정할 수 있다. 예를 들면   + Serial.print(78, BIN) gives "1001110"   + Serial.print(78, OCT) gives "116"   + Serial.print(78, DEC) gives "78"   + Serial.print(78, HEX) gives "4E"   + Serial.println(1.23456, 0) gives "1"   + Serial.println(1.23456, 2) gives "1.23"   + Serial.println(1.23456, 4) gives "1.2346" | |

##### **시리얼 통신 2**

|  |  |
| --- | --- |
| **PC에서 문자 하나를 받아서 그것이 ‘0’이면 LED를 끄고 ‘1’이면 LED를 켜는 프로그램**LED는 13번 핀에 바로 연결한 상태이고, 아두이노는 PC의 시리얼 모니터와 시리얼통신을 진행한다.아두이노는 PC로부터 데이터가 전송되어 오면 그것을 읽어들여서 ‘1’이면 LED를 켜고, ‘0’이면 끈다. 그 이외의 데이터에 대해서는 잘못된 명령이라는 문자열을 출력한다. | |
| 시리얼2.PNG | |
| #define LED 13void setup() { pinMode(LED, OUTPUT); Serial.begin(9600);}void loop() { if ( Serial.available() ) { char command = Serial.read(); if (command == '0') { digitalWrite(LED, LOW); Serial.println("LED off."); } else if (command == '1') { digitalWrite(LED, HIGH); Serial.println("LED on."); } else { Serial.print("Wrong command :"); Serial.println(command); } }} |  |
| **int Serial.available()**   * 전송되어 *내부버퍼*(64바이트)에 저장된 데이터의 개수를 반환한다. * 입력인수는 없다.   **int Serial.read()**   * 전송되어 내부 버퍼(64바이트)에 저장된 데이터 중 가장 첫 번째 데이터(ASCII코드)를 읽어서 반환한다. * 이 함수가 수행되면 내부 버퍼의 크기는 하나가 줄어든다. * 내부 버퍼가 비었다면 -1을 반환하다.   **int Serial.println()**   * 출력 문자열의 끝에 줄바꿈 기호 ‘\r\n’ 가 자동으로 붙는다는 점 외에는 Serial.print()함수와 동일한 동작을 수행한다.   *내부 버퍼*  내부 버퍼는 전송된 데이터가 일시적으로 저장되는 내부 메모리를 말하며 데이터가 전송된 순서대로 저장된다.  아두이노의 내부 버퍼의 크기는 64바이트이다. 전송되어 저장된 데이터를 사용하려면 내부 버퍼에서 이 데이터를 읽어내야 하는데 이때 사용되는 함수가 Serial.read()함수이다.  가장 먼저 전송된 데이터 하나를 읽어낸 후 그 데이터는 버퍼에서 삭제되며, 만약 버퍼가 비었다면 -1을 반환한다. 따라서 버퍼에 읽어낼 데이터가 있는지 없는지를 먼저 검사하는 것이 일반적이고 이때 사용되는 함수가 Serial.available()이다. 이 함수는 버퍼에 저장된 데이터의 개수를 반환하며 따라서 버퍼가 비어있다면 0을 반환한다. | |

##### **시리얼통신 3**

|  |  |
| --- | --- |
| 11번핀에 부저가 연결되어 있고, 13번핀에는 LED가 연결되어 있다. 시리얼 모니터로부터 명령어를 받아서 아래와 같이 동작을 수행하는 코드를 작성한다.  - ‘0’ : LED를 끈다.  - ‘1’ : LED를 켠다.  - ‘2’ : 부저를 짧게 한 번 울린다. (삑)  - ‘3’ : 부저를 짧게 두 번 울린다.(삐삑)  - 그 외의 명령어들은 잘못된 것이라는 메세지를 출력한다. | |
| #define LED 13#define BUZ 11void setup() { pinMode(LED, OUTPUT); pinMode(BUZ, OUTPUT); Serial.begin(9600);}void loop() { if ( Serial.available() ) { char command = Serial.read(); switch(command) { case '0': digitalWrite(LED, LOW); Serial.println("LED off."); break; case '1' : digitalWrite(LED, HIGH); Serial.println("LED on."); break; case '2' : digitalWrite(BUZ, HIGH); delay(50); digitalWrite(BUZ, LOW); break; case '3' : digitalWrite(BUZ, HIGH); delay(50); digitalWrite(BUZ, LOW); delay(50); digitalWrite(BUZ, HIGH); delay(50); digitalWrite(BUZ, LOW); break; default: Serial.print("Wrong command :"); Serial.println(command); } }} | 부저는 안에 회로가 내장된 능동부저를 사용해야만 digitalWrite()으로 제대로 된 테스트가 가능하다. 수동부저인 경우는 tone()을 사용할 것 |

# 5. 인터럽트

|  |  |
| --- | --- |
| 인터럽트.jpg | 아두이노는 loop() 함수가 실행되는 도중에 특정한 이벤트가 발생했을 때에는 수행 중인 loop() 함수 작업을 중단하고 발생한 이벤트를 처리한 다음, 다시 원래 수행하던 작업으로 되돌아가 계속 loop() 함수를 처리하게 된다. 이렇게 ‘즉시 처리해야 하는 특수한 이벤트의 발생’을 인터럽트라고 한다. |

##### 인터럽트가 발생하면 아두이노는 그 이벤트를 처리할 작업들이 수록된 함수를 호출하는데 이 함수를 **인터럽트 서비스 루틴 (interrupt service routine, 이하 ISR)**이라고 한다.

##### 

##### 스위치와 LED가 연결된 회로에서 스위치가 눌렸는지를 감지하려면 기본적으로 아두이노가 loop()함수 내에서 항상 이 스위치의 상태를 검사해야 한다. 하지만 이것을 인터럽트로 처리하면 소프트웨어적으로 이 버튼의 상태를 항상 검사할 필요가 없다.

##### 다른 작업을 하고 있다가 ‘스위치가 눌려지면’ 그 즉시 하드웨어적으로 인터럽트를 발생하여 하던 일을 멈추고 이를 처리할 ISR을 호출한 뒤 ISR이 끝나면 하던 일을 다시 하면 되는 것이다. 이와 같이 주변기기와 아두이노가 통신을 수행하는데 있어서 아두이노가 주변기기의 상태를 loop()함수에서 상시 검사하지 않고 인터럽트로 처리하면 전체적인 시스템의 운용 효율이 매우 높아지게 된다.

##### **attachInterrupt( pin, ISR, mode)** 함수

##### 인터럽트핀에 처리 함수를 붙이는 일을 수행한다.

##### pin 값은 우노의 경우 0(또는 INT0 ), 1 (또는 INT1) 둘 중 하나이다.

##### D2번 핀의 경우 0을, D3번 핀의 경우 1을 입력해야 한다는 것에 주의해야 한다.

##### ISR은 인터럽트가 걸렸을 때 호출할 함수의 이름이다.

##### mode 는 RISING / FALLING / CHANGE / LOW 중 하나이다.

##### attachInterrupt() 함수는 대개 setup()함수 안에서 초기 설정시 호출해서 사용한다.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

##### **LED toggle 스위치 1**

|  |  |
| --- | --- |
| interrupt_bb.png | const int buttonPin = 2;  const int ledPin = 13;  boolean ledOn = false;  int check;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT\_PULLUP);  pinMode(ledPin, HIGH);  }  void loop() {  if (digitalRead(buttonPin) == LOW) {  if (ledOn == false) {  digitalWrite(ledPin, HIGH);  ledOn = true;  } else {  digitalWrite(ledPin, LOW);  ledOn = false;  }  }  } |

위 스케치를 실행하면 제대로 동작하지 않는다. 이유는 무엇인가?

##### **LED toggle 스위치 2**

|  |  |
| --- | --- |
| const int buttonPin = 2;  const int ledPin = 13;  boolean ledOn = false;  int check;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT\_PULLUP);  pinMode(ledPin, HIGH);  }  void loop() {  if (digitalRead(buttonPin) == LOW) {  if (ledOn == false) {  digitalWrite(ledPin, HIGH);  ledOn = true;  } else {  digitalWrite(ledPin, LOW);  ledOn = false;  }  }  } | const int buttonPin = 2;  const int ledPin = 13;  boolean ledOn = false;  boolean checkOnce = false;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT\_PULLUP);  pinMode(ledPin, HIGH);  }  void loop() {  if (digitalRead(buttonPin) == LOW) {  if (checkOnce == false) {  checkOnce = true;  if (ledOn == false) {  digitalWrite(ledPin, HIGH);  ledOn = true;  } else {  digitalWrite(ledPin, LOW);  ledOn = false;  }  }  } else {  checkOnce = false;  }  } |

##### **LED toggle 스위치 3 :** 인터럽트의 ISR 을 이용

|  |  |
| --- | --- |
| const int buttonPin = 2;  const int ledPin = 13;  boolean ledOn = false;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT\_PULLUP);  pinMode(ledPin, HIGH);  attachInterrupt(0, myInterrupt, FALLING);  }  void loop() {  delay(1000);  }  void myInterrupt() {  if(ledOn == false) {  digitalWrite(ledPin, HIGH);  ledOn = true;  } else {  digitalWrite(ledPin, LOW);  ledOn = false;  }  } | 코드안에서 버튼이 눌렸는지, 아닌지를 조사하는 부분이 없다.  하드웨어적으로 버튼이 눌린다면 인터럽트에 의해서 자동으로 LED의 불이 켜진다. |

##### **디바운싱 : 바운싱(채터링) 제거**

##### 터치센서와 다르게 기계적인 접점을 갖는 스위치는 점점이 붙거나 떨어지는 순간에 바운싱(bouncing), 채터링(chaterring) 현상이 있다. 이것은 접점이 붙거나 떨어지는 그 짧은 순간에 접점이 고속으로 여러 번 on/off 되는 현상을 말하며 기계적인 스위치라면 반드시 발생하는 현상이다.

##### 

##### 이것은 짧은 순간(약 100ms 이내임)에 여러 번 접점이 붙었다가 떨어지는 것을 반복하므로 의도하지 않게 인터럽트가 여러 번 발생할 수 있으므로 이러한 바운싱 현상을 제거해야 올바르게 동작하는데 이것을 **디바운싱(debouncing)**이라고 한다.

##### 디바운싱하는데 하드웨어적인 방법과 소프트웨어적인 방법이 있는데, 아두이노에서는 부가적인 디바운싱 회로를 만드는 하드웨어적 방법 대신 소프트웨어적으로 이것을 구현하여 불필요한 인터럽트를 억제할 수 있다.

##### 기본적인 아이디어는 ISR 함수 안에서 바운싱이 없어질 때 까지 기다렸다가 스위치 신호를 다시 읽어서 정상 호출된 것인가 아닌가를 판별하는 것이다.

|  |  |
| --- | --- |
| **#include<util/delay.h>** const int buttonPin = 2;  const int ledPin = 13;  boolean ledOn = false;  void setup() {  pinMode(ledPin, OUTPUT);  pinMode(buttonPin, INPUT\_PULLUP);  pinMode(ledPin, HIGH);  attachInterrupt(0, myInterrupt, FALLING);  }  void loop() {  delay(1000);  }  void myInterrupt() {  \_delay\_ms(80);  if(digitalRead(buttonPin) != LOW) {  return;  }  if(ledOn == false) {  digitalWrite(ledPin, HIGH);  ledOn = true;  } else {  digitalWrite(ledPin, LOW);  ledOn = false;  }  } | attachInterrupt(INT0, toggleLed, FALLING/RISING/CHANGE/LOW); \_delay\_ms(milli seconds);  ISR함수 안에서 아두이노의 시간관련 함수인 delay(), delayMicrosecond(), millis(), micros() 가 정상 동작하지 않는다.  반면에 \_delay\_ms()함수는 단순히 소트프웨어적으로 지연을 시켜주는 함수라서 ISR안에서도 정상 작동한다. 단, 이 함수를 사용하기 위해서는 다음과 같이 헤더 파일을 인클루드 시켜야 한다 |

## **volatile 지시자**

##### 지시자(directive)인 volatile 은 변수 선언문의 변수형 앞에 두어서 컴파일러가 그 변수를 접근하는 방식을 지정한다.

##### 일반적으로 변수 데이터는 런타임에 RAM영역에 저장되지만 어떤 변수는 레지스터(register)에 임시로 저장되어 사용하기도 하는데 이 경우 속도면에서는 훨씬 유리하지만 RAM에 저장된 실제 데이터와 레지스터에 저장된 (임시)데이터가 서로 다른 경우가 발생할 수도 있다.

##### 어떤 변수를 volatile로 지정하면 그 변수 데이터는 레지스터의 임시 저장소가 아니라 RAM에서 직접 읽어오도록 컴파일된다. 아두이노 프로그램의 경우 보통은 volatile로 정의할 필요는 없으나 인터럽트 서비스 루틴 (ISR) 내부에서 그 값이 변경되는 변수는 반드시 volatile로 선언해야 실시간으로 변경되는 데이터 값을 ISR 외부에서 정확하게 읽어올 수 있게 된다.

##### 예를 들면 다음과 같다.

|  |
| --- |
| int pin = 13; volatile int state = LOW; void setup() {  pinMode(pin, OUTPUT);  attachInterrupt(0, blink, CHANGE); }  void loop() {  digitalWrite(pin, state); }  void blink() {  state = !state; } |

위의 예에서 blink()함수가 ISR이며 이 안에서 변경되는 변수 state는 volatile로 지정하였음을 확인할 수 있다.

# 6. 아날로그 입출력

##### **조도센서**

##### 

##### 황화 카드뮴 셀( CdS)은 저항의 일종이며 광전자를 이용한 반도체 효과를 이용하여 외부 빛의

##### 조도에 의해 저항값이 결정된다. 즉, 빛이 강해지면 저항값이 작아지므로 회로에서 전압을

##### 측정하게 되면 전위차가 커지게 되고, 반대로 빛이 약해지면 저항값이 증가하여 전위차는

##### 감소하게 된다.

|  |  |
| --- | --- |
| cds_bb.png | |
| const int cdsPin = A0;  const int ledPin = 13;  int sensorValue = 0;  int sensorMin = 1023;  int sensorMax = 0;  void setup() {  pinMode(ledPin, OUTPUT);  digitalWrite(ledPin, HIGH);  Serial.begin(9600);  while (millis() < 5000) {  sensorValue = analogRead(cdsPin);  if (sensorValue > sensorMax) {  sensorMax = sensorValue;  }  if (sensorValue < sensorMin) {  sensorMin = sensorValue;  }  }  digitalWrite(ledPin, LOW);  }  void loop() {  sensorValue = analogRead(cdsPin);  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);  sensorValue = constrain(sensorValue, 0, 255);  analogWrite(ledPin, sensorValue);  Serial.println(sensorValue);  delay(1000);  } | analgWrite()은 PWM에서 상세히 기술  setup 함수에서 입력값에 대한 calibration 처리  즉, 0.5초 동안 입력값을 읽어 들인 후,  그중 최소와 최대값을 구한 뒤에 0에서 255 사이의 값으로 매핑시킴  map(value, fromLow, fromHigh, toLow, toHigh);  fromLow ~ fromHigh 범위의 value값을 toLow ~ toHigh 범위값으로 변환 |

##### **온도센서**

##### LM35 는 사용하기 매우 간편한 온도 센서로써, -55 도에서 +150 도 사이로 0.5 도 간격으로 아날로그 출력을 내보낸다.

|  |  |
| --- | --- |
| lm35_bb.png | |
| int lm35Pin = 0;float tempC;int reading;void setup() {analogReference(INTERNAL);Serial.begin(9600);}void loop() {reading = analogRead(lm35Pin);tempC = reading / 9.31;Serial.print("Tep:");Serial.println(tempC);Serial.println("C");delay(3000);} | 핀에서 측정된 값으로부터 온도(C) 환산  LM35 datasheet(<http://www.ti.com/lit/ds/symlink/lm35.pdf>)에 의하면,  1°C는 10 mV의 전위차에 해당(1V = 100°C)하며  이에 따른 일반적인 환산식은  온도 = 5.0/1024 \* 100.0 \* analogRead(lm35Pin);  그러나, 실제 LM35에서 사용하는 전압은 0V~1.0V까지를 사용하므로,  0~5V 의 범위가 아니라, 0~1.0V 의 범위를 사용하면 더 큰 분해능을 얻을 수 있다.  여유치를 두어 1.1V를 1024로 나누면,  1.1V/1024 = 1.074mV  analogRead(): 1.074mV = X°C: 10mV  X°C = 10/1.074 = ~9.31  즉, analogRead() 가 9.31 단위당 1°C에 해당  analogReference(INTERNAL); 에 의해서 ahref 전압을 1.1V로 설정하면 온도는 0°C~110°C의 범위를 가지게 된다. analogReference(option)analogRead()에서의 기준 전압을 설정 option DEFAULT : 5V, 3.3VINTERNAL : 1.1V  * EXTERNAL : AREF에 연결된 전압   참조: http://playground.arduino.cc/Main/LM35HigherResolution |

##### **PWM**

##### 아두이노의 디지털핀은 오직 HIGH(5V) 아니면 LOW(0V) 두 가지 신호 외에는 출력할 수 없다. 하지만 PWM (pulse width modulation, 펄스 폭 변조) 기능을 이용하면 마치 아날로그 전압처럼 0V와 5V 사이의 전압으로 출력을 낼 수 있다. 따라서 LED의 밝기를 제어한다든가 모터의 회전 속도를 제어하는데 사용할 수 있다.

##### PWM은 진정한 의미의 아날로그 출력은 아니고 흉내를 내는 것인데 그 원리는 다음 그림과 같다.

##### **Duty rate** : 한 주기 안에서 HIGH를 유지하는 시간의 비율

##### 주파수 (Hz) : 1초당 반복 횟수

|  |  |
| --- | --- |
| pwm.gif | analogWrite(pin, value);  Pin은 pin 번호  Value는 duty rate, 0~255  예를 들어 analogWrite(pin, 127)의 경우  50% Duty cycle을 가지는 아날로그 신호와 유사한데,  한 주기 중 처음 50%만 5V를 인가하고, 나머지는 0V를 인가한다면  Duty rate는 50%가 될 것이고, 평균 소요 전압은 2.5V이므로 value값은 127이 된다.  이렇게 5V, 0V의 상태를 초당 980번 반복한다면 1번 반복하는데 1/980=~0.00102 초 정도 걸릴것이므로 마치 아날로그 출력을 하는 것과 유사해 진다. |

|  |  |
| --- | --- |
| uno_pwm.PNG | PWM 출력 핀별 동작 주파수(~로 표시됨)   * 3, 9, 10, 11번 핀 - 490Hz * 5, 6번 핀 – 980Hz   analogWrite(pin, value)   * pin : 3, 5, 6, 9, 10, 11 * value : 0에서 255 사이의 정수   PWM 의 사용 용도:   * LED 흐리게 하기 * 아날로그 출력하기 * 오디오 신호 만들기 * 모터 공급용 속도조절하기 |

##### **아날로그 출력 PWM :가변저항으로 LED 밝기 조절하기**

|  |  |
| --- | --- |
| led+potentionmeter_bb.png | |
| int potPin = 0;  int ledPin = 11;  int val=0;  void setup() {  pinMode(ledPin, OUTPUT);  Serial.begin(9600);  }  void loop() {  val = analogRead(potPin);  Serial.println(val);  analogWrite(ledPin, val);  delay(10);  } | 아날로그 입력 신호 : 0-1023  디지털 출력 신호 : 0-255  구동 회로는 일반 LED와 다르지 않게 220오옴 저항을 직결하여 전원을 연결  가변저항을 돌림에 따라 아날로그 입력치가 바뀌고 이를 PWM에 의해 아날로그 출력이 달라지게 되어 전구의 밝기가 조절된다. |

# 7. 센서와 액츄에이터 연동

##### **초음파 센서로 장애물 경고하기**

##### HC-SR04는 거리를 측정하는 초음파 센서이다.

##### 거리를 측정하는 원리는 적외선 센서의 경우 빛을, 초음파 센서의 경우 소리를 내 보낸 후,

##### 각각의 빛이나 소리가 물체에 닿으면 반사되며 신호를 보내고 이를 다시 받는 사이의 시간을

##### 측정함으로써 거리를 계산하게 된다.

##### 적외선 센서는 검은 물체에 작동하지 않으며 초음파센서는 스펀지와 같은 흡수성의 물체에 잘

##### 작동하지 않는다.

##### 초음파 센서로 다음과 같이 작동하는 아두이노 프로그램을 작성해 보자.

##### 장애물이 30cm 밖에 있다면 부저는 울리지 않는다.

##### 30cm 이내 10cm 바깥에 있다면 0,5초 주기로 삑삑거린다.

##### 10cm보다 안쪽에 있다면 연속적인 삐~ 신호를 울린다.

##### 마치 자동차의 후방 경고음과 같이 동작한다.

##### 

##### 초음파 센서는 D2,D3 핀에, 부저는 D11번 핀에 달려있다.

|  |  |
| --- | --- |
| buz+ultrasonic_bb.png | |
| #define USS\_E 2  #define USS\_T 3  #define BUZ 11  #define BUZ\_TONE 1000  void setup() {  Serial.begin(9600);  pinMode(BUZ, OUTPUT);  }  void loop() {  float dist = getDist();  int intervalOff = 0;  int buzzerOff = 0;  if (dist > 30) {  intervalOff = -1;  } else if (dist > 10) {  intervalOff = 50;  buzzerOff = 450;  } else {  intervalOff = 0;  }  if (intervalOff == 0) {  //digitalWrite(BUZ, HIGH);  tone(BUZ,BUZ\_TONE);  } else if (intervalOff > 0) {  //digitalWrite(BUZ, HIGH);  tone(BUZ,BUZ\_TONE);  delay(intervalOff);  noTone(BUZ);  //digitalWrite(BUZ, LOW);  delay(buzzerOff);  } else {  digitalWrite(BUZ, LOW);  noTone(BUZ);  }  Serial.print(dist);  Serial.println(" cm");  }  float getDist() {  pinMode(USS\_T, OUTPUT);  pinMode(USS\_E, INPUT);  digitalWrite(USS\_T, LOW);  delayMicroseconds(2);  digitalWrite(USS\_T, HIGH);  delayMicroseconds(10);  digitalWrite(USS\_T, LOW);  long duration, t;  duration = pulseIn(USS\_E, HIGH);  if (duration == 0 ) {  return -1.0;  } else {  return duration / 29.4 / 2;  }  } | 센서에 Trig 신호 입력 => 10micro sec 정도 유지 ⇒ Trig 신호 off  Echo pin에서 HIGH->Low 간격을 측정  거리 환산  음파속도는 1s에 340m  1m에 0.00294초 걸림  1cm 에 0.0000294초==29.4마이크로sec  거리는 왕복이므로 전체 계산식은  거리(cm) = 경과시간(micro sec)/29.4/2 |

##### **조도(cds)센서+Timer+가변저항을 통한 조도값 주기적 모니터링**

|  |  |
| --- | --- |
| simpleTimer <http://playground.arduino.cc/Code/SimpleTimer> | |
| cds_potention_bb.png | |
| #include <SimpleTimer.h>  #define LED 13  #define CDS A0  #define POT A1  SimpleTimer timer;  boolean ledOn = false;  int intervalTime;  int minTime = 500;  int maxTime = 10000;  void setup() {  pinMode(LED, OUTPUT);  Serial.begin(9600);  timer.setInterval(1000, toggle);  }  void loop() {  intervalTime = map(analogRead(POT),0,1023,minTime,maxTime);  timer.setInterval(intervalTime, notifyMe);  timer.run();  }  void toggle() {  if(ledOn == true) {  digitalWrite(LED,LOW);  } else {  digitalWrite(LED,HIGH);  }  ledOn = !ledOn;  }  void notifyMe() {  Serial.print("Brightness : ");  Serial.println(analogRead(CDS));  Serial.print("Interval Time (ms) : ");  Serial.println(intervalTime);  } | 1초 마다 LED가 깜빡이고, 가변저항의 입력값에 따른 주기를 설정하여 조도값을 시리얼 모니터에 전달하는 프로그램  simpleTimer 라이브러리를 다운받고 PC에 저장한 후 헤더파일을 불러온 뒤 코딩한다.  map(value, fromLow, fromHigh, toLow, toHigh);  fromLow ~ fromHigh 범위의 value값을 toLow ~ toHigh 범위값으로 변환 |