목차

1. 설계 기초

1. brainstorming -> 프로젝트를 시작하기 위한 시작점.
2. SRS -> 프로젝트를 진행하기 위한 시작점.

2. 아두이노 기초

1. 하드웨어 기초
2. 기본 코드

3. 아두이노

1. LED
2. Analog sensor
3. motor
4. speaker

메모 포함[이1]: 이 수업에서는 소개하는 내용들에 대한 전문적인 지식을 전달하는 것이 아니라, 단순히 내용을 소개하는 것에서 그친다. 자세한 내용은 학년이 높아지면 다른 전공 수업에서 배우고, 이 수업에서는 '이런 것이 있구나' 정도만 얻어 가도록 하자.

1. 설계 기초

1. brainstorming

1. brainstorming

집단적인 발상 기법을 통해 자발적으로 창의적 아이디어를 생산하기 위한 학습 도구이자 회의 기법.

1) 핵심

1. 다수일수록 생성되는 아이디어가 많음.
2. 비판이 가해지지 않으면 아이디어는 계속 생성됨.
3. 생성되는 아이디어의 수가 많을수록 독창적인 아이디어를 발견할 확률이 높음.

메모 포함[이2]: 엉뚱한 주장, 비논리적인 답변, 타당하지 않은 해결책을 모두 수용한다.

메모 포함[이3]: 질보다는 양이 중요한 기법이다.

2) 원칙

1. 비판하기 마라. (판단하지 마라)(의견에 대한 검토는 선별 단계에서 진행함.)
2. 질보다는 양.
3. 엉뚱한 의견도 환영하라.
4. 타인의 아이디어를 발전시켜라. (긍정적인 살을 붙여라.)
5. 주제에 집중하라. (주제를 벗어나지 마라.)
6. 타인이 말할 때 끼어들지 마라.

이를 지키기 위해서 중재자가 필요함.

3) 단점

내성적인 구성원들이나 상하 관계의 구성원들이 참여하기 힘들 수 있음.
무임승차가 발생할 수 있음.
다수의 의견에 끌려갈 수 있음.

단점의 극복을 위해 중재자가 필요함.

4) 중재자

1. 원칙을 지키고 단점을 극복해야 함.
2. 어떤 의견이든 수용해야 함.
3. 발언이 독점되지 않도록 해야 함.

+ 제한시간

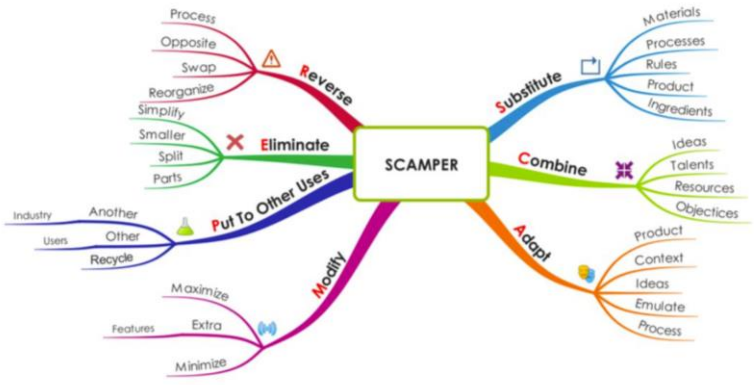
제한시간을 두고 하는 것이 좋음. (15분~1시간 가량 끊어서.)

---

2. 아이디어를 생성하는 방법

SCAMPER(스캠퍼)를 사용할 수 있음.

대체(Substitute)	: 기존의 것 대체.
결합(Combine)	: 기존의 것들을 결합.
적용(Adapt)	: 기존의 것의 성질이나 특징을 다른 것에 적용.
변형(Modify)	: 기존의 것을 변형.
다른 활용(Put to other use)	: 특정 영역의 것을 다른 영역에서 활용.
제거(Eliminate)	: 기존의 것의 성질이나 특징을 제거.
재배열(Reverse)	: 기존의 것의 순서, 모양, 역할 등으로 전환.



3. brainstorming 진행 방식

아이디어 생성. -> 아이디서 선별. -> 아이디어 분류 및 정리. -> 최종 아이디어 도출.

1) 아이디어 생성

가능한 많은 양의 아이디어를 모음.

2) 아이디어 선별

목적, 주제와 너무 동떨어지거나 하는 등의 아이디어들을 제거함.

메모 포함[이4]: 선별 전에 프로젝트의 목적을 정하고, 그 목적에 맞춰 아이디어를 선별하는 것이 좋겠다.

3) 아이디어 분류 및 정리

비슷한 성격의 아이디어끼리 분류하여 정리함.

2. SRS

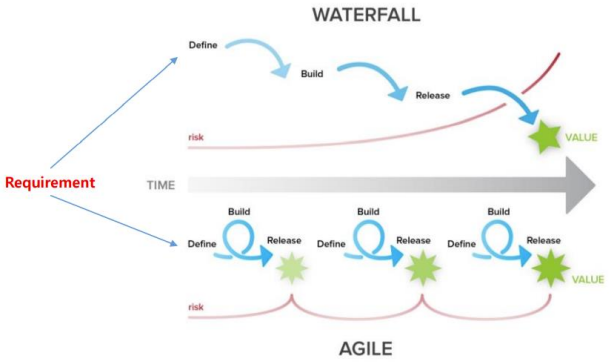
메모 포함[이5]: Software Requirement Specification.
소프트웨어 요구사항 설명서.

1. 소프트웨어 개발 기초

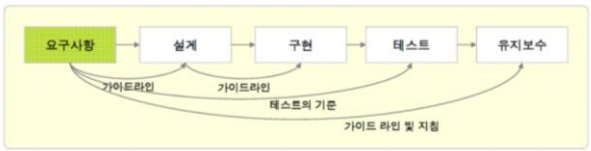
1) 소프트웨어 개발 방법론

waterfall과 agile 등이 있음.

대략적인 방법은 아래 그림과 같음. (실제로는 훨씬 더 많은 세부 기법들이 존재함.)



2) 소프트웨어 개발 과정



2. SRS(Software Requirement Specification)

개발하려는 소프트웨어의 사용자 요구, 필수 요소, 표준, 명세 등을 충족하기 위해 요구사항을 적은 설명서.

개발자들이 이해하고 사용할 수 있도록 요구사항들을 문서화한 것.

메모 포함[이6]: 소프트웨어 요구사항 설명서. 즉, SRS에는 해당 소프트웨어의 요구사항들이 들어 있다.

1) 요구사항의 중요성

1. 팀원들이 동일한 시각/공감대/목표를 가질 수 있게 함.
2. 구체적 설명서가 있어야 관련 계획을 세울 수 있음.

메모 포함[이7]: 프로젝트에서 굉장히 중요한 요소 중 하나이다.

SRS 없이 프로젝트를 하는 경우는 거의 없음.
SRS 없이 프로젝트를 하더라도 결국 SRS에서 하는 작업들을 하게 됨.

2) 요구사항의 종류

기능적 요구사항 -> 소프트웨어가 가져야 할 기능들 및 기능들 사이의 처리과정.
비기능적 요구사항 -> 소프트웨어의 성능, 안정성, 편리함 등 시스템의 기능과 관련되지 않은 것들.

메모 포함[이8]: ex. 응답 시간, 속도, 데이터 처리량, 사용의 용이성, 신뢰도, 보안성 등.

3. SRS 작성법(요구사항 관리)

메모 포함[이9]: SRS를 작성하는 것을 요구사항 관리라고도 한다.

1. 요구사항을 추출한다. (요구사항의 종류를 고려.)
2. 요구사항을 분석한다.
3. 요구사항을 명세한다.
4. 요구사항을 검증한다.

이 과정과 관련된 문서 작성 형식이 존재함.
문서 작성 형식은 개발하는 소프트웨어에 따라 다를 수 있지만, 국제 표준 형식이 있기는 함.

1) 요구사항 추출

고객으로부터 필요한 요구사항을 수집한 후, 그것을 바탕으로 요구사항을 도출하는 과정.

고객의 최초 요구사항은 추상적이기 때문에 정확한 요구사항을 파악하는 것이 중요함.

2) 요구사항 분석

추출된 요구사항을 분석 기법을 사용하여 일관성 있는 요구사항들로 정리하는 과정.

요구사항 분석 방식.
1. 시스템을 구조적이고 계층적으로 표현한다. (직접 스케치, UML 등 사용)
2. 외부 사용자와의 상호작용 방식, 시스템 내부 구성요소들끼리의 상호작용 방식 고안.

요구사항 분석 기법의 종류.

메모 포함[이10]: 실제로는 굉장히 많은 분석 기법들이 있다.
또한 분석 기법들에 대해서는 배울 것들이 훨씬 많다.
여기서는 개요만 설명한다.

구조적 분석	-> 시스템의 기능을 중심으로 분석.
객체지향 분석	-> 사용자 중심의 시나리오 분석을 통해 유스케이스 모델 구축.

3) 요구사항 명세

분석된 요구사항을 정확하고 자세하게, 빠짐없이 문서화하는 것.

4) 요구사항 검증

검증 사항	설명
무결성(correctness)과 완전성(completeness)	사용자의 요구를 에러 없이 완전하게 반영하고 있는가?
일관성(consistency)	요구사항이 서로간에 모순되지 않는가?
명확성(unambiguous)	요구분석의 내용이 모호함 없이 모든 참여자들에 의해 명확하게 이해될 수 있는가?
기능성(functional)	요구사항 명세서가 "어떻게" 보다 "무엇을"에 관점을 두고 기술되었는가?
검증 가능성(verifiable)	요구사항 명세서에 기술된 내용이 사용자의 요구를 만족하는가? 개발된 시스템이 요구사항 분석 내용과 일치하는지를 검증할 수 있는가?
추적 가능성(traceable) 및 변경 용이성	시스템 요구사항과 시스템 설계문서를 추적할 수 있는가?

공통 어휘 검증 : 공통 어휘를 사용하고 있는가?

2. 아두이노 기초

1. 하드웨어 기초

1. 하드웨어 기초

1) 핀 맵

회로에서 사용되는 모든 부품은 전선을 연결하기 위한 pin이 존재함. (연결 인터페이스.)
각 핀들은 명칭과 용도가 존재함.

데이터 시트(Datasheet)(핀 맵) : 핀에 대한 정보가 적혀 있는 문서.

회로 연결 전에는 데이터 시트 등을 통해 핀 맵을 숙지해야 함.

2) 외부 전원

아두이노 보드에 USB를 연결하면 전원이 공급되지만, 배터리 등의 외부 전원을 사용하면 USB 연결 없이도 독립적으로 작동할 수 있음.

아두이노는 외부 전원 연결을 위한 커넥터를 가지고 있음.

아두이노는 레귤레이터라는 부품이 있어 전압에 상관없이 연결이 가능함.
레귤레이터 : 임의의 전압을 가진 외부 전원을 연결하면 5V로 변환하여 내부에 전달하는 장치.

3) VCC와 GND(Ground)

전자 회로에서 (+)극은 VCC, (-)극은 GND라 부름.

VCC는 항상 전압의 크기를 함께 표기함. (ex. VCC(5V))
전압의 크기만을 명시하는 경우도 있는데, 이는 VCC가 생략된 것으로 모두 (+)극을 의미함.

GND는 전압의 크기와 상관없이 모두 공통으로 사용함. (GND들은 구별 없이 모두 그냥 GND라고 부름.)
이렇게 사용되는 GND를 공통 그라운드(Common Ground)라고 함.

4) 5V와 3.3V

모든 디지털 전자 회로는 5V 방식과 3.3V 방식 중 하나를 사용함.
아두이노에서는 대부분 5V이지만, 3.3V인 것도 있으므로 유의해야 함.

USB는 5V로 전원을 공급함.

5) 쇼트(Short)(합선)

VCC와 GND가 직접 연결되어 순간적으로 과도한 전류로 인해 부품이 타버리는 현상.

$\Delta V = IR$ 인데 저항(R)이 거의 존재하지 않은 채로 VCC와 GND가 연결되면 전류(I)가 기하급수적으로 높아져 발생한 열에 의한 현상.

6) 플로팅(floating)

스위치를 누른 상태에는 HIGH값이 들어오지만, 스위치를 뗀 상태에서는 LOW값이 들어오는 것이 아니라 HIGH값과 LOW값이 섞여서 들어오는 플로팅 현상이 발생함.

사용 중인 핀 주위의 상태나, 핀 근처에 떠다니는 전자들이 입력으로 들어오기 때문에 발생하는 현상임.

저항을 통해 신호를 일정하게 유지하는 접지 등을 통해 해결함.

메모 포함[이11]: 접지 방법(full down, full up 등)은 설명해 준 것 같기는 한데 잘 못 들었다. ppt는 캡처해 뒀으니 참고. -> pull down 저항. 이거 애들한테 한 번 물어보자.

+ 아두이노 장치 기본 제작 과정

1. 각 부품들의 특성 파악. -> VVC/GND/input/output 등.
2. 브레드보드 구성.
3. 연결.
4. 코딩 후 하드웨어에 올리기.

2. 기본 코드

1. 기본 함수들

1) setup(), loop()

setup() : 환경 설정 관련 함수.

loop() : 반복 명령 함수. -> 이 함수 내부의 코드는 반복되어 수행됨.

프로그램이 실행되면 setup() 함수가 먼저 1 회 실행되고, 이후 loop() 함수가 반복 실행됨.

2) pinMode(<pin>, <mode>)

아두이노 보드의 핀을 어떻게 사용할지를 설정하는 함수.

첫 번째 인자에는 사용 방식을 지정할 핀 번호를 지정함.

두 번째 인자에는 사용 방식을 지정함.
사용 방식에는 INPUT, OUTPUT, INPUT_PULLUP 이 있음.
INPUT 과 OUTPUT 이 있다는 정도로만 알면 됨. (INPUT_PULLUP 은 INPUT 에 PULL UP 옵션이 붙은 것.)

+ digital vs. analog

digital 은 0V 와 5V 의 두 가지 상태로 정보를 처리하는 것.
digital 은 2 진법의 0 과 1 로 대체되어 부울 대수에서 True, False 로 사용되거나, On, Off 등으로 사용됨.

analog 는 0V 와 5V 사이의 모든 상태로 정보를 처리하는 것.
analog 상태를 수치화하는 것을 양자화라고 함. 양자화의 세밀함을 해상도(Resolution)라고 함.

digital 은 2 가지 상태로 정보를 처리하는 것이고, analong 는 해상도 만큼의 단계로 정보를 처리하는 것.

+ 핀과 전역변수

핀은 기본적으로 const 전역변수를 사용하여 상수처럼 사용함.
(ex. int const LED = 9;)

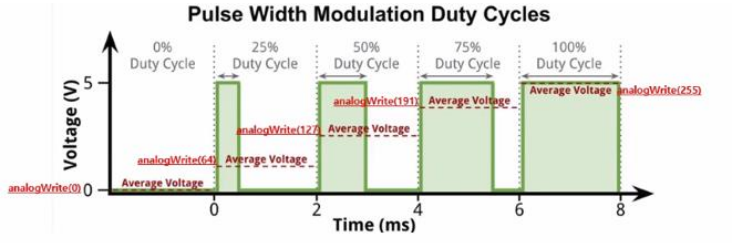
+ PWD(Pulse Width Modulation)

digital 을 이용하여 analog 출력을 흉내내는 방식.

5V(on), 0V(off)와 Duty cycle 을 통해 전압 평균값을 특정 값으로 지정하여 출력할 수 있음.
지정한 Duty cycle 에 해당하는 만큼 5V(on), 0V(off)를 번갈아 가며 사용함.

메모 포함[이12]: 0%, 25%, 50% 75%, 100% 등으로 설정이 가능하다. Duty cycle 값과 전압 값은 비례한다.
-> LED의 밝기나 모터 속도의 조절 등에 사용된다.

analog 출력 생성 장치는 고가이기 때문에 아두이노는 저가인 PWD 를 사용함.



+ 아날로그 핀의 사용 방식 지정

pinMode()로 사용 방식을 지정해주는 게 기본이지만, analogRead() 등 읽거나 쓰는 함수가 있으면 사용 방식이 지정된 것으로 취급되어 생략하기도 함.

사용 시에는 정석대로 pinMode()를 사용해서 핀 지정을 해주자.

3) digitalWrite(<pin>, <value>)

digital 방식(0 과 1)으로 OUTPUT 을 내보내는 함수.

digital 은 0 과 1 만을 가지므로, 아두이노에서는 0 을 LOW, 1 을 HIGH 로 사용함.
해당 핀을 HIGH, LOW 로 전환하여 회로적으로 연결된 부품을 제어할 수 있음.

digitalWrite() 함수를 사용하기 위해서는 해당 핀이 pinMode() 함수에 의해 이미 OUTPUT 으로 지정되어 있어야 함.

4) digitalRead(<pin>)

digital 방식(0 과 1)으로 INPUT 을 가져오는 함수.

인자로 지정한 핀에서 INPUT 을 받아 리턴함.

digital 은 0 과 1 만을 가지므로, digitalRead() 함수는 LOW(0) 또는 HIGH(1) 값을 리턴함.
해당 핀에 연결된 부품의 상태를 확인할 수 있음.

digitalWrite() 함수를 사용하기 위해서는 해당 핀이 pinMode() 함수에 의해 이미 INPUT 으로 지정되어 있어야 함.

5) analogWrite(<pin>, <PWM value>)

analog 방식으로 OUTPUT 을 내보내는 함수.

첫 번째 인자로는 digital 핀 쪽에 물결(~) 표시가 있는 핀 중(3, 5, 6, 9, 10, 11)에 하나를 지정.
PWM 신호는 대부분 490Mhz 이지만 5, 6 번 핀은 980Mhz 임.

두 번째 인자로는 0(always on)~255(always off) 사이의 정수 값을 지정함.
-> 아두이노 Uno 의 PWM Resolution 이 8bit 이기 때문.

함수를 사용하기 위해서는 해당 핀이 pinMode() 함수에 의해 이미 OUTPUT 으로 지정되어 있어야 함.

6) analogRead(<pin>)

analog 방식으로 INPUT 을 가져오는 함수.

첫 번째 인자로는 MCU 쪽에 A(ADC)로 표시된 핀 중(A0, A1, A2, A3, A4, A5) 하나를 지정.

아두이노에는 ADC(Analog to Digital Converter)가 내장되어 있어 analog 를 digital 로 변환해서 받음.
아두이노 Uno 의 ADC Resolution 은 10bit 이기 때문에 ADC 는 analog 값을 0~1023 사이의 수로 리턴함.

메모 포함[이13]: ex.
5V -> 1023
2.5V -> 512
0V -> 0

analog 센서들은 ADC 핀에 연결하여 analogRead() 함수로 센서 정보를 값으로 가져옴.

함수를 사용하기 위해서는 해당 핀이 pinMode() 함수에 의해 이미 INPUT 으로 지정되어 있어야 함.

+ 아두이노 실습 과정

수업 방식

부품 설명 -> 부품 확인 -> 인터페이스 확인 -> 부품 지식 심화(검색 등 활용)

실습 방식

이론 공부 -> 실습에 필요한 부품 확인 -> 부품들 연결 -> 코드 작성 -> 아두이노에 upload -> 실행 -> 성공적으로 실행되었으면 결과를 동영상으로 촬영하여 제출.

오류 발생 원인

1. 코드 오류 2. 연결 실수 3. 접촉불량 4. 케이블 불량 5. 부품 불량

---

3. 아두이노

1. LED

$$\Delta V = IR$$
(옴의 법칙)

1. 저항 (옴의 법칙)

부품이 사용하는 전압의 크기와 부품의 안정성이 유지되는 전류 값을 유지하기 위해 적절한 저항을 연결해 줘야 함.

전류, 전압, 저항 등의 계산은 이 과목에서 중요하지 않으므로 인터넷에 나와있는 계산기를 사용함.

메모 포함[이14]: ex.
아두이노 제공 전압이 5V, LED가 2V를 사용하고 LED가 0.02A 이하에서 안전하다면 (5-2)/0.02=150옴의 저항을 연결해 줘야 한다.
(0.02A만큼의 전류를 유지하기 위해서.)

LED가 9V를 사용할 때 0.02A이하에서 안전한 경우 -> 이 때는 왜 9V로 계산하냐?

전압의 계산 방식이 이해가 안 된다. 배터리와 아두이노의 차이점인가..?

해당 수업 pdf 참고해 보기.

2. LED (Light Emitting Diode, 발광 다이오드)

다리가 긴 쪽이 anode(+), 다리가 짧은 쪽 또는 옆면이 평평한 부분이 cathode(-)임.
anode 에 (+)를, cathode 에 (-)를 연결해야 함.

3. 코드

1) 핀을 출력으로 지정

1. 사용할 핀을 전역변수로 지정. -> const int LED = 9; //9번 핀 사용.
2. setup()에서 출력 핀 지정. -> pinMode(LED, OUTPUT); //LED(9번 핀)을 출력으로 지정.

2) LED 켜기/끄기(digital)

setup()/loop()에서 LED 핀을 HIGH/LOW로 지정 -> digitalWrite(LED, HIGH); //LED(9번 핀)를 HIGH로 지정.

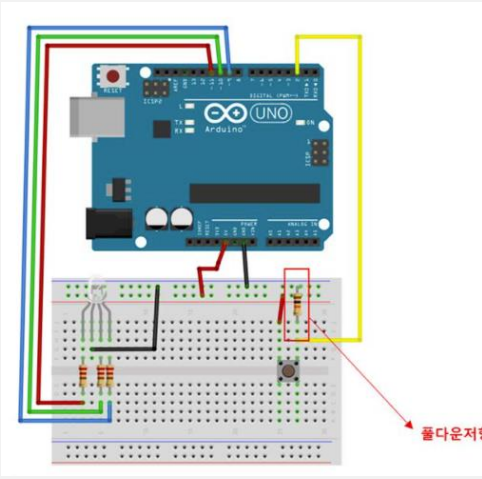
HIGH : ON 상태 / LOW : OFF 상태.

3) LED 밝기 조절하기

loop()에서 LED 핀을 analog로 조정. -> analogWrite(LED, i);

4) 프로그램 일시정지

loop()에서 일시정지. -> delay(100); //인자로 작성한 수 만큼 일시정지. ms 단위.
-> 하드웨어의 변화를 관찰하기 위해서는 일시정지가 필요할 수 있음,



5) RGB LED 사용하기

이해가 좀 더 필요하다.

LED 3개가 붙어 있다고 생각하면 쉬움. 이 LED들의 색이 조합되어 특정 색을 생성함.

6) serial monitor 사용하기

1. serial monitor의 속도 설정하기. -> 일반적으로 9600 보드레이트 사용.
2. setup()에서 속도 맞추기. -> Serial.begin(9600); // serial monitor의 것과 동일하게 지정.
3. loop()에서 출력할 값 지정. -> Serial.println(i); //인자로 serial monitor에서 확인할 값 지정.

+ serial monitor

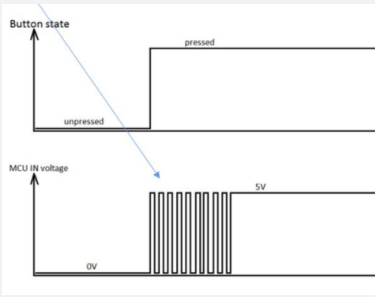
작업 중 주고받는 값들을 확인하기 위한 장치.

아두이노 IDE에서 툴->시리얼 모니터로 들어가서 사용 가능.

serial monitor의 속도와 코드의 속도를 맞춰주는 것이 필수적임.
일반적으로 9600 보드레이트를 사용하지만, 속도는 코드의 것과 맞춰 주기만 하면 됨.

+ bouncing

스위치 등의 기계적 접점이 붙었다가 떨어질 때 기계적/물리적 진동에 의해 아주 짧은 시간 안에 접점이 붙었다가 떨어지는 것이 반복되는 현상.



이 현상을 해결하기 위해 debouncing해줘야 함.

0. "이전 버튼 상태"와 "현재 버튼 상태"를 위한 변수를 사용함. (각각 A, B라고 부르겠음.)

1. B에 현재 버튼 상태 저장.
2. A와 B의 값이 다른 경우, 5ms 동안 프로그램 일시정지. -> bouncing이 끝날 때까지 기다림.
3. 5ms 후에 다시 B에 현재 버튼 상태 저장.
4. A와 B의 값이 다른 경우 작업 진행. (ex. LED 상태 전환 등.)
5. A에 B 값을 저장.

이 과정을 반복함.

debouncing 예시 함수.

디바운싱

```
const int LED=9; //9번 핀을 사용하는 LED 상수 정의
const int BUTTON=2; //버튼을 2번 핀에 지정

boolean lastButton=LOW; //이전 버튼의 눌림 상태를 Boolean 으로 선언
Boolean currentButton=LOW; //현재 버튼의 눌림 상태를 Boolean으로 선언

Boolean ledOn=false; //현재 LED의 점멸 상태를 Boolean형 변수로 선언

Void setup()
{
    pinMode(LED, OUTPUT); //LED(9번 핀)를 출력으로 지정
    pinMode(BUTTON, INPUT); //BUTTON(2번 핀)을 입력으로 지정
}

Void loop()
{
    currentButton=debounce(lastButton); //디바운싱 처리된 버튼값을 읽음
    if(lastButton == LOW && currentButton == HIGH) //버튼을 누름
    {
        ledOn=!ledOn; //LED 점멸 상태를 나타내는 ledOn 변수값을 바꿈
    }
    lastButton=currentButton; //이전 버튼값을 현재 버튼값으로 변경

    digitalWrite(LED, ledOn); //LED 점멸 상태를 바꿈
}
```

```
/*
  디바운싱 처리된 현재 버튼 상태값을 반환
*/

boolean debounce(boolean last)
{
    // 현재 버튼 상태를 확인
    boolean current=digitalRead(BUTTON);

    // 이전 버튼 상태와 현재 버튼 상태가 다름
    if(last != current)
    {
        // 5ms 동안 기다림
        delay(5);

        // current에 현재 버튼 상태를 저장
        current=digitalRead(BUTTON);
    }
    return current; //current 값을 반환
}
```

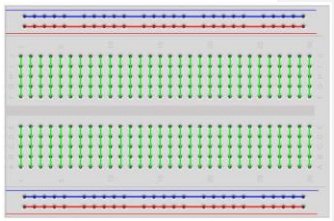
메모 포함[이15]: 버튼이 눌렸을 때 실행되어야 하는 명령들은 대체로 한 번만 수행되면 되기 때문에 lastButton의 값이 LOW이고 currentButton의 값이 HIGH인 경우에만 수행되도록 한 것으로 보인다.

버튼이 눌린 상태에서 떨어질 때 명령을 수행하려면 조건을 (lastButton == HIGH) && (currentButton == LOW)로 해서 수행하면 될 것이다.

+ 브레드보드

아두이노에서 회로 연결을 위해 사용하는 판.

오른쪽 그림에서 같은 색의 선이 위치하는 구멍끼리는 연결되어 있음.



2. Analog sensor

1. 전압 분배 법칙

저항에 따른 입력 전압의 변화를 나타낸 공식.
저항이 2개 있을 때, 입력 전압에 따른 출력 전압은 오른쪽 공식과 같음

$$V_{out} = \frac{Z_2}{Z_1 + Z_2} V_{in}$$

전압은 V 단위, 저항은 kΩ 단위임.

이해하려 하지 말고 암기해라.

ADC는 이런 방식으로 계산된 전압 값을 digital 값으로 전환함.

메모 포함[이16]: ex.
5V -> 1023
2.5V -> 512
0V -> 0

2. 가변 저항

전자 회로에서 임의로 저항 값을 바꿀 수 있는 저항기.

전압이 바뀌지 않고 저항 값이 변경되는 경우, 전압 분배회로를 만들어 별도로 저항 값의 변화를 측정함.

무게 센서, 온도 센서, 포토레지스터 등에서 가변 저항이 사용됨.
외부 조건에 따라 저항 값이 바뀜.

+ 포토레지스터

빛의 총량에 따라 저항 값이 바뀌는 가변 저항.

(ex. 200kΩ짜리 포토레지스터는, 완전히 어두운 상태일 때 약 200kΩ, 밝은 빛을 받을 때는 약 0Ω.)

3. 값 보정을 위한 함수들

보통 map() 함수와 constrain() 함수를 연달아 사용함.

(ex.
sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
sensorValue = constrain(sensorValue, 0, 255);)

1) map(<값>, <org_min>, <org_max>, <your_min>, <your_max>)

값의 orginal min~max 값을 your min~max로 매핑하는 함수.

매핑 : 하나의 값을 다른 값으로 대응시키는 것.

analogWrite() 함수와 analogRead() 함수를 보면 알 수 있듯이, 아두이노 UNO가 analog 값을 입력받을 때의 해상도는 10bit이고, analog 값을 출력할 때의 해상도는 8bit임.
즉, 10bit 단위로 받은 값을 사용하다가 출력할 때는 8bit 단위로 전환해 줘야 하는데, 이때 map() 함수가 사용되는 것.

값을 반대로 전환할 때도 사용됨.
(ex.
주변이 어두울수록 LED는 밝아져야 하는 경우.
ADC로 입력 받은 값이 작을수록(어두울수록) LED로는 큰 값을 넣어 밝게 출력되도록 map() 함수를 설정.
reverseValue = map(value, 0, 255, 255, 0);)

2) constrain(<값>, <min>, <max>)

값을 min~max 사이의 값으로 한정하는 함수. (min 값이 최소, max 값이 최대.)

첫 번째 인자로 지정한 값이 min 이상 max 이하면 해당 값을 그대로 리턴함.
첫 번째 인자로 지정한 값이 min 미만이면 min을 리턴하고, max 초과이면 max를 리턴함.

map() 함수 이후에 constrain()은 왜 하는 것? 매핑을 하면 최소값과 최댓값을 지정하는데, 그러면 해당 범위에서 벗어나는 값이 없지 않나?

메모 포함[이17]: ex.
val = map(val, 0, 1023, 0, 255);
이 명령은 10bit(0~1023) 기준이었던 기존의 val 값을 8bit(0~255)를 기준으로 변환하여 다시 배정하는 것이다.

메모 포함[이18]: 아날로그 값을 입력 받고 그 값을 사용해서 아날로그 값을 출력할 때는 매핑이 반드시 필요하다.

4. Calibration

센서 값을 맞추는 과정.

이 과정 없이 10bit 해상도로 값을 받아서 매핑하면 값이 끝부터 끝까지 안 들어갈 수 있음.

메모 포함[이19]: 뇌피셜. 팩트인지는 불명.

작동 전에 센서로 값을 제공해 줘야 함.

(ex.

```
void setup() { // Calibration 후 RGB를 쓰는 걸로 변경할 것.
  pinMode(GLED, OUTPUT);
  Serial.begin(9600);

  digitalWrite(13, HIGH); // the start of the calibration period:

  // calibrate during the first five seconds
  while (millis() < 5000) {
    sensorValue = analogRead(LIGHT);

    if (sensorValue > sensorMax) {
      sensorMax = sensorValue; // record the maximum sensor value
      Serial.println(sensorMax);
    }

    if (sensorValue < sensorMin) {
      sensorMin = sensorValue;      // record the minimum sensor value
      Serial.println(sensorMin);
    }
  }
  digitalWrite(13, LOW); // the end of the calibration period
}
```

)

+ millis() 함수

아두이노의 동작이 시작된 후부터 경과된 시간을 millisecond로 리턴하는 함수.

3. Motor

1. 모터의 종류

1) 직류(DC) 모터

전원이 인가되면 한 방향으로 계속 회전하는 모터.

2) 브러시형(brushed) 직류(DC) 모터

고정 자석과 코일을 이용하는 모터.
값이 싸고 속도 제어가 쉬움.

3) 서보 모터

명령에 따라 특정 위치로 이동하고, 다음 명령 전까지 해당 위치에 머무르는 모터.
방향, 각도를 제어하는 모터.

일반 서브모터의 회전 범위는 0~180도임.

PWM(analog) 신호를 보내 이동할 위치를 지정함.
서보 모터의 위치를 지정하는 컨트롤 핀이 존재함,

2. 트랜지스터와 콘덴서

모터 제어를 위한 장치들.

1) 트랜지스터 (바이폴라 접합 트랜지스터)

이미터(E), 컬렉터(C), 베이스(B)로 구성된 장치.

전류는 C로 들어와서 E로 빠져나감.
B 핀의 값을 바꾸어 전류의 트랜지스터 통과 여부를 결정할 수 있음.
-> 이것을 이용하여 단일 트랜지스터는 스위치처럼 사용할 수 있음.
-> B에 analog 값을 지정하여 사용함.

메모 포함[이20]: 어떤 식으로 값이 반영되는지는 잘 모르겠다.

2) 콘덴서

세라믹, 마이카, 필름 콘덴서는 극성이 없음.

전해 콘덴서는 극성이 있음.
긴 다리가 (+), 짧은 다리가 (-)임.
다리가 잘려 있는 경우, 흰색 띠가 있는 쪽이 (-)임. 탄탈 콘덴서는 띠가 있는 쪽이 (+)임.

이 단원에서 콘덴서는 역전압 스파크 발생 방지를 위해 사용됨.

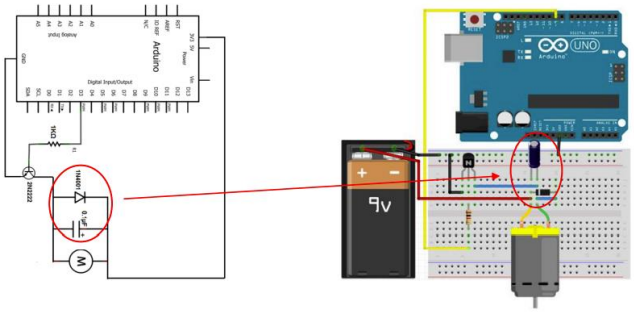
3. 모터 사용 시 회로 보호

관성 전류와 그 반대 방향의 전류가 충돌하면 역전압 스파크가 발생하여 회로가 손상될 수 있음.

1. 모터를 갑자기 정지하는 경우.
2. 모터의 전원을 갑자기 제거하는 경우.
3. 모터의 회전 방향 전환을 위해 전류 방향을 변경하는 경우.

이럴 때 역전압 스파크가 발생할 수 있음.

다이오드와 콘덴서로 보호 회로를 구성할 수 있음. (이해 X. 암기.)



+ 모터 쉴드

일반적으로는 모터 사용 시에 외부 전원을 공급하기보단 모터 쉴드를 사용함.

프로젝트 시에도 외부 전원보다는 모터 쉴드를 사용하는 것이 좋음.

---

4. 모터에 외부 전원 공급

일반적인 DC 모터와 서보 모터는 아두이노가 제공할 수 있는 전류보다 많은 전류를 소모함.
아두이노의 전원과 별개로 외부 전원을 DC 모터에 공급해줘야 함.

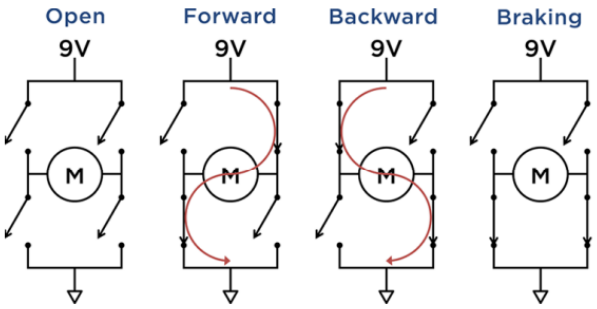
5. H-브리지

모터 1개와 스위치 4개가 H 모양으로 연결된 장치.
일반적으로는 스위치로 트랜지스터를 사용하고, 회로 보호용 다이오드를 연결함.

DC 모터를 양방향으로 구동하기 위해 H-브리지를 사용함.

H-브리지에는 정지, 브레이크, 전진, 후진의 4가지 주요 동작 상태가 있음.

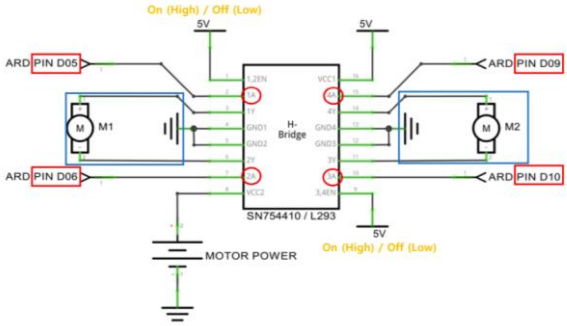
H-브리지 한 쪽의 스위치를 모두 연결해서 외부 전원이 그대로 GND에 합선되는 것을 주의해야 함.



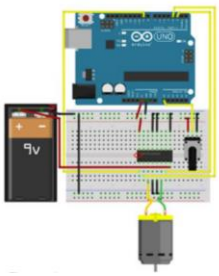
+ H-브리지 SN754410 사용법

SN754410으로 하나의 IC에 반 H-브리지 4 개가 내장됨
반 H-브리지 두 개를 합치면 완전한 풀H-브리지를 만들 수 있음

The motors should be connected to PWM pins



실습 6-3



필요한 부품

- 가변저항 1개
- DC 모터 1개
- 9V 배터리 1개
- 9V 배터리 클립 1개
- SN754410 H-브리지 IC 1개

```
const int EN=9;
const int MC1=3;
const int MC2=2;
const int POT=0;

int val=0;
int velocity=0;

void setup() {
    Serial.begin(9600);
    pinMode(EN, OUTPUT);
    pinMode(MC1, OUTPUT);
    pinMode(MC2, OUTPUT);
    brake();
}
void loop( ) {
    val=analogRead(POT);
    Serial.println(val);
    if(val>562) { //정회전
        velocity=map(val, 563, 1023, 0, 255);
        forward(velocity);
    }
    else if(val<462) {//역회전
        velocity=map(val, 461, 0, 0, 255);
        reverse(velocity);
    }
    else {
    brake(); //브레이크
    }
}
```

⇨

```
// 지정된 속도로 회전(from 0-255)
void forward(int rate)
{
    digitalWrite(EN, LOW);
    digitalWrite(MC1, HIGH);
    digitalWrite(MC2, LOW);
    analogWrite(EN, rate);
}

// 지정된 속도로 역회전(from 0-255)
void reverse(int rate)
{
    digitalWrite(EN, LOW);
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, HIGH);
    analogWrite(EN, rate);
}

// 정지
void brake()
{
    digitalWrite(EN, LOW);
    digitalWrite(MC1, LOW);
    digitalWrite(MC2, LOW);
    digitalWrite(EN, HIGH);
}
```

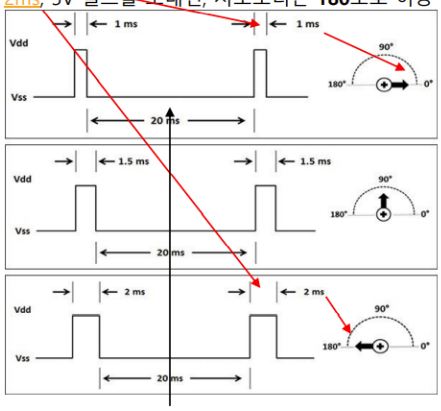
6. 서보 모터

서보 모터는 컨트롤 핀을 사용하여 PWM 신호를 보내서 제어함.

서보 모터를 위한 라이브러리와 함수들이 있음.

1ms 동안 5V를 보내면 서보 모터는 0도로 이동함.
2ms 동안 5V를 보내면 서보 모터는 180도로 이동함.

1ms, 5V 펄스를 보내면, 서보모터는 **0**도로 이동
2ms, 5V 펄스를 보내면, 서보모터는 **180**도로 이동



일반적인 서보 모터의 펄스 주기는 20ms임.

메모 포함[이21]: (실습 6-4)

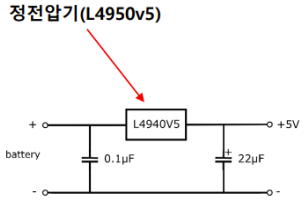
#include <Servo.h>

Servo <변수>; -> 서보 객체 생성
<변수>.attach(<핀>); -> 서보 핀 지정. (pinMode()와 동일한 기능.)
<변수>.write(<값>); -> 서보 모터에 값 지정.

7. 서보 모터에 외부 전원 공급

일반적인 DC 모터는 아두이노가 제공할 수 있는 전류보다 많은 전류를 소모함.
아두이노와 별개로 외부 전원을 DC 모터에 공급해줘야 함.

정전압기를 사용해 전원을 공급함.
정전압기에는 크기가 다른 두 콘덴서를 연결해서 사용함.



+ 거리 센서

적외선(IR) 거리센서

적외선을 보내 **물체에 반사되어 되돌아오는 적외선!!**으로 물체와의 거리를 측정

적외선의 양을 측정하는 방식과 반사각을 측정하는 방식이 있고, 반사각을 측정하는 방식이 측정거리가 더 멀다.

적외선을 쏘는 IR LED와 바로 옆의 포토 센서로 들어온 적외선 빛의 각도를 계산하도록 구성

GP2Y0A21YK (약 10 ~ 80 cm)

초음파 거리센서

초음파를 보내 **물체에 부딪힌 후 되돌아오는데 걸리는 시간으로 물체와의 거리를 측정**

초음파 : 귀에 들리지 않을 정도로 높은 주파수 (**약 20 kHz 이상**)

초음파는 파장이 짧아 지향성과 직진성이 높으며 공기 중에서는 **340m/s의 일정한 속도**로 진행하는 특징을 가짐

HC-SR04 (약 2cm~400cm)

센서를 사용할 때 사용하는 코드와 값은 센서마다 다를 수 있음.
센서를 제작한 곳에서 제공하는 코드와 값을 사용하기 때문에, 이해할 필요는 없음.

실습 6-5

```
#include <Servo.h>

const int SERVO=9; //9번 핀, SERVO
const int IR=0; //아날로그 입력 0번 핀, 거리 센서
const int LED1=3; //3번 핀을 사용하는 LED1
const int LED2=5; //5번 핀을 사용하는 LED2
const int LED3=6; //6번 핀을 사용하는 LED3
const int LED4=11; //11번 핀을 사용하는 LED4

Servo myServo; //서보 객체

int dist1=0; //1번 구역 장애물의 거리를 저장하는 dist1
int dist2=0; //2번 구역 장애물의 거리를 저장하는 dist2
int dist3=0; //3번 구역 장애물의 거리를 저장하는 dist3
int dist4=0; //4번 구역 장애물의 거리를 저장하는 dist4
```

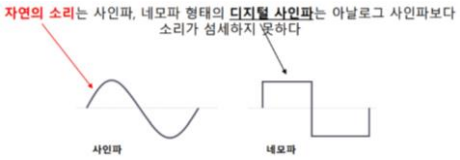
⇨

```
void setup() {
    myServo.attach(SERVO); //9번 핀에 서보모터 추가
    pinMode(LED1, OUTPUT); //LED1(3번 핀)을 출력으로 지정
    pinMode(LED2, OUTPUT); //LED2(5번 핀)를 출력으로 지정
    pinMode(LED3, OUTPUT); //LED3(6번 핀)을 출력으로 지정
    pinMode(LED4, OUTPUT); //LED4(11번 핀)를 출력으로 지정
}

void loop() {
    dist1=readDistance(15); //15도 구역에서 장애물 거리 계산
    analogWrite(LED1, dist1); //장애물 거리에 따라 LED1 밝기
    delay(300); //다음 측정 전까지 300ms 지연
    dist2=readDistance(65); //65도 구역에서 장애물 거리 계산
    analogWrite(LED2, dist2) //장애물 거리에 따라 LED2 밝기
    delay(300); //다음 측정 전까지 300ms 지연
    dist3=readDistance(115); //115도 구역에서 장애물 거리 계산
    analogWrite(LED3, dist3); //장애물 거리에 따라 LED3 밝기
    delay(300); //다음 측정 전까지 300ms 지연
    dist4=readDistance(165); //165도 구역에서 장애물 거리 계산
    analogWrite(LED4, dist4); //장애물 거리에 따라 LED4 밝기
    delay(300); //다음 측정 전까지 300ms 지연
}

int readDistance(int pos) {
    myServo.write(pos); //지정된 위치(pos)로 서보 이동
    delay(600); //서보가 움직일 동안 대기
    int dist=analogRead(IR); //IR 센서값 읽기
    dist=map(dist, 50, 500, 0, 255); //입력값을 8bit 범위로 매핑
    dist=constrain(dist, 0, 255); //변환값을 0-255로 고정
    return dist; //측정한 장애물 거리를 반환
}
```

이 실습은 서보 모터 위에 적외선 거리 센서를 다는 것인가..?

4. speaker

1. 소리



자연의 소리는 아날로그 값으로, sin파로 표현됨.

자연의 소리는 디지털 sin파로 표현함.
해상도가 높을수록 아날로그 값에 근접해짐.

2. 스피커와 저항

LED의 저항과 동일하게 계산하면 됨.

스피커 내부에 있는 코일에서 별도의 내부저항이 발생함.
이 내부저항의 크기는 8옴 정도임. 저항 계산 시 고려해줘야 함.

3. piezo speaker

값싼 스피커.

1) 사용 방법

1. timeHigh 값을 구한다. -> timeHigh = 1 / (2 * frequency) = period / 2
-> timeHigh 값은 frequency 또는 period를 알면 구할 수 있음.
-> 음 마다 나름의 frequency, period를 가지고 있음. (pitches.h에서 확인 가능.)

2. 코드를 작성한다.

```
analogWrite(speakerPin, HIGH);
delayMicroseconds(timeHigh);          -> timeHigh 마이크로 초만큼 일시정지.
analogWrite(speakerPin, LOW);
delayMicroseconds(timeHigh);          -> timeHigh 마이크로 초만큼 일시정지.
```

이 4줄의 코드를 반복문에 넣어서 사용함.

짧은 시간간격 동안 HIGH값과 LOW값을 유지함으로써 디지털 sin파를 구현할 수 있음.

메모 포함[이22]: 1000 microsecond = 1 millisecond

4. tone() 함수

1) 타이머 사용

아두이노 nano와 아두이노 uno는 3개의 타이머를 가지고 있음.

아날로그(PWM) 핀들은 타이머를 사용함.
5, 6번 핀 : 타이머0 (8bit) -> delay(), millis(), micros() 등에서 사용.
9, 10번 핀 : 타이머1 (16bit) -> servo lib 등에서 사용.
11, 3번 핀 : 타이머2 (8bit) -> tone() 등에서 사용.

해당 핀이 사용 중이면 다른 곳에서 해당 타이머를 사용할 수 없음.

메모 포함[이23]: 타이머를 사용하지 않는 tone() 함수도 존재한다.

2) 사용 방법

tone(<pin>, <frequency>) 또는 tone(<pin>, <frequency>, <duration>) 으로 사용함.

tone(<pin>, <frequency>) -> 설명을 안 해줘
tone(<pin>, <frequency>, <duration>)-> 해당 기간 동안 해당 핀에 해당 frequency 값으로 소리 제공?

noTone(<pin>) -> 이거는 뭐야 또 시발 설명을 안 해주네

+ 마이크-앰프-스피커

마이크 : 소리를 전기 신호로 변환.
앰프 : 전기 신호를 증폭.
스피커 : 전기 신호를 소리로 변환.

5. serial, processing

1. USB-to-serial

USB-to-serial에서 변환 장치로는 FTDI chip이라는 것을 사용함.

USB를 연결하면 FTDI chip을 통해서 값이 변환되고, tx/rx 핀으로 매핑됨.

USB를 연결하여 사용한다는 것은 USB를 통한 serial 통신을 위해 tx(1번)/rx(0번) 핀을 사용한다는 의미임.
-> USB 연결 중에는 tx/rx 핀을 사용할 수 없음.

2. soft serial

USB 사용 중에 serial을 사용해야 하는 것들은(블루투스, WIFI 등) Softserial을 통해 tx/rx로 0, 1번이 아닌 다른 핀을 사용할 수 있음.

1) Softserial 사용 방법

```
#include <SoftwareSerial.h>

SoftwareSerial <식별자a>(<pin1>, <pin2>);      -> 장치 모듈의 tx/rx를 pin1과 pin2에 매핑하는 것.
                                                -> (장치의 tx를 pin1에, rx를 pin2에 연결하고 사용함.)

void Setup()
{
        Serial.begin(9600);
        Serialprintln.("Enter At something");
        <식별자a>.begin(9600); -> 식별자a에 해당하는 모듈을 serial로 communication할 수 있게 하는 것.
}
```

메모 포함[이24]: ex.
블루투스 모듈.

3. 블루투스 모듈 명령 : HC-06

기기마다 다를 수 있지만, 대부분은 블루투스 모듈 명령으로 HC-06을 사용함.

1) 사용 방법

<command><enter 키>로 명령을 보냄.
명령을 작성하고 enter 키를 보내야 함.

enter 키는 운영체제마다 조금씩 다름.
enter 키를 키보드에서 입력할 때는 상관이 없지만, 코드에 작성할 때는 운영체제에 맞춰 작성해야 enter로 제대로 인식함.

Unix, Linux : ₩r
Mac : ₩r
Dos : ₩n
Windows : ₩r₩n

메모 포함[이25]: ex.
A₩r₩nB₩r₩n를 윈도우용 코드로 작성한 경우 명령 A, 명령 B가 수행된다.

2) 명령들

명령들은 AT+HELP 명령을 통해 확인할 수 있음.

4. serial 관련 함수들

1) Serial.available()

serial 값이 넘어오는 경우 0보다 큰 수를 리턴하는 함수.

2) Serial.read()

serial 값을 문자로 읽어서 리턴하는 함수.

3) Serial.println(<값>)

serial로 값을 출력하는 함수.
값으로는 변수에 저장된 값, 문자열 등을 지정할 수 있음.
두 번째 인자로 DEC을 지정하여 10진수로 출력할 수 있음.

4) Serial.parseInt()

serial 값을 정수로 읽어서 리턴하는 함수.
정수가 아닌 값을 읽으면 0을 리턴함.

```
const int LED=9;
char data; // 수신 문자 저장 문자열 변수 선언

void setup()
{
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}

void loop()
{
    if(Serial.available()>0) { // 값이 있을때 작동
        data=Serial.read();
        if(data=='1') { // 1: LED 켜기
            digitalWrite(LED, HIGH);
            Serial.println("LED ON");
        } else if(data=='0') { // 0: LED 끄기
            digitalWrite(LED, LOW);
            Serial.println("LED OFF");
        }
    }
}
```

메모 포함[이26]: ex.
serial에서 27을 보낸 것을 Serial.read()로 읽으면 '2', '7' 두 개의 문자를 리턴한다.

메모 포함[이27]: ex.
serial에서 27을 보낸 것을 Serial.parseInt()로 읽으면 정수 27을 리턴한다.

5. processing

draw를 사용하기 위해 쓰는 기능인 것 같음.

지금까지 아두이노에서 수행해 온 것과 달리, processing은 pc에서 수행하는 것.

setup()과 loop()로 구성되어 있던 아두이노와 달리, setup()과 draw()로 구성되어 있음.

1) 작동 방식

아두이노에서 serial로 write한 정보는 pc에서 COM port로 들어가고, 이것을 processing이 읽고 draw함..
processing이 draw된 것을 읽어서 COM port에 write하면 serial로 정보가 이동해서 아두이노가 serial의 정보를 읽음.

아두이노 - serial - COM port - draw

COM port(컴 포트)가 뭐 임?

실습 7-4 과정
1. 이미지 다운로드
2. 코드 작성 -> 아두이노 포트 검색 필요.

processing 코드는 아두이노 코드와 동일한 파일에 작성하는가?
-> precessing용

실습 7-5 과정 -> 한 번 이해해보기.
실습 7-5는 안 해도 된다. 시간 되면 하기.

아두이노의 핀들은 pull-up resistor가 내장되어 있음.
pull-up setup이 필요 없음. 직접 연결 가능.