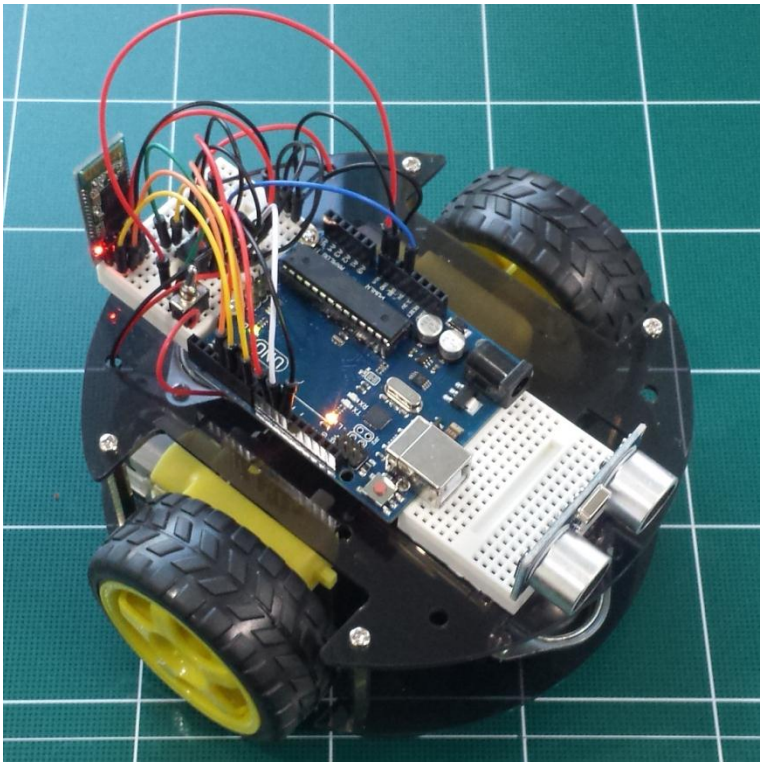
모바일로봇 종결 키트



메카솔루션

2015년 1월 21일

서론

본 모바일로봇 종결키트는 합리적인 가격의 모바일 로봇 플랫폼을 이용하여, 모바일 로보틱스의 기본이 되는 메카트로닉스의 기초, 센서의 신호 및 처리, 모터 제어, 모바일 로봇 기구학 및 실험, 그리고 무선 통신의 이해 및 실습에 이르기까지 기초부터 응용까지 다룰 수 있도록 구성되어 있습니다. 이론 중심의 교육에서 보다 실질적인 부분까지 지식 및 노하우를 담으려 노력하였으나, 물리적 제약으로 인해 싣지 못한 부족한 부분에 대해서는 추가적으로 업데이트가 될 예정입니다. 또한, 모바일로봇에 대해 중점적으로 다루게 될 본 매뉴얼은 불필요한 부분은 최대한 배제하고, "아두이노 종결키트 매뉴얼"과 중복되는 부분은 최소화하였습니다. 따라서, 아두이노의 설치 및 라이브러리 활용 등의 아두이노 관련 정보 및 실습 부분은 "아두이노 종결키트"를 참고해주시기 바랍니다. 본 자료는 교육 목적으로 제작되었으며, 상업적인 이용은 메카솔루션의 동의 하에 이루어질 수 있도록 부탁 드립니다.

이 자료를 통해서 로봇 제작에 관심이 있는 청소년들, 대학생, 그리고 로봇을 배우고자 하는 비전공자들에게도 유용하게 사용되길 바라며, 내용상의 문제점에 대해서는 지적을 해 주시면 감사히 받고 수정하도록 하겠습니다. 토론 및 관련 내용에 대한 보완사항은 roboholic84@gmail.com으로 메일을 보내주시면 감사하겠습니다.

목차

1. 아두이노 기초

A. 아두이노란?

아두이노를 키워드로 말한다면 ‘쉽다’, ‘오픈 소스’, ‘마이크로컨트롤러’ 등으로 표현될 수 있을 것 같습니다. 이렇듯, 아두이노는 여러분의 새로운 아이디어를 쉽게 구현할 수 있도록 디자인된 마이크로 컨트롤러입니다. 이 마이크로컨트롤러와 센서, 모터, 디스플레이, 그리고 무선 통신 모듈과 같은 다양한 전자 소자들을 연결하여 여러분들께서 생각하고 계시는 ‘그것’을 구현할 수 있도록 도와주게 됩니다. 아두이노는 단지 하드웨어를 의미하는 것 뿐만 아니라 프로그래밍을 쉽게 할 수 있도록 돕는 통합개발환경도 일컫게 되는데 많은 사람들이 공유하는 소스코드를 하드웨어에 업로드하거나 변경하여 손쉽게 동작시킬 수 있는 장점을 가지고 있습니다.

오픈소스라는 장점으로 인해서 다양한 모양과 성능의 아두이노 혹은 아두이노 호환보드가 시중에 나와 있는데, 시스템의 사이즈, 요구되는 성능, 필요한 입출력 포트의 수 등, 목적에 맞게 사용하시면 되며 본 매뉴얼에서는 가장 많이 사용되고 있는 우노를 사용하도록 하겠습니다.

Fig. 1. UNO R3

아두이노를 이용하여 할 수 있는 일들은 광범위합니다. 로봇 공학에서부터 스마트 홈에 이르기까지 할 수 있는 영역이 넓으며, 다양한 창의적이고 혁신적인 제품 개발 및 연구에 이르기까지 그 가능성은 더욱 더 커지고 있습니다.

- 로봇공학 (모바일 로봇, 휴머노이드 로봇, 물고기 로봇 등등)

- 음악 및 사운드 장치
- 다양한 센서와의 연동 (온도, 습도, 기울기 등등)
- 게임 분야와의 연동
- 모바일 애플리케이션 (스마트폰과의 통신)
- 네트워킹 시스템 (유선 혹은 무선 통신망 구축)
- 스마트 홈 구현 (조명 제어 및 실시간 모니터링)
- 그 밖에도 무수히 많은 아이디어를 구현할 수 있다

우노 보드에 대한 설명은 다음의 그림을 통해서 이해할 수 있습니다. USB 전원 공급 및 프로그래밍을 위한 USB 포트가 있으며, 배럴잭이라 불리우는 검정색 단자를 통해서 7-12V의 DC 전원을 공급받을 수 있습니다. 전원에 대한 스위치가 없는 것이 단점이라서 보드에 전원을 차단하기 위해서는 별도의 차단 회로 혹은 수동으로 USB 케이블 및 전원 케이블을 뽑아야 하는 점은 있지만, 사용자들로부터 큰 불만은 없는 듯 합니다.

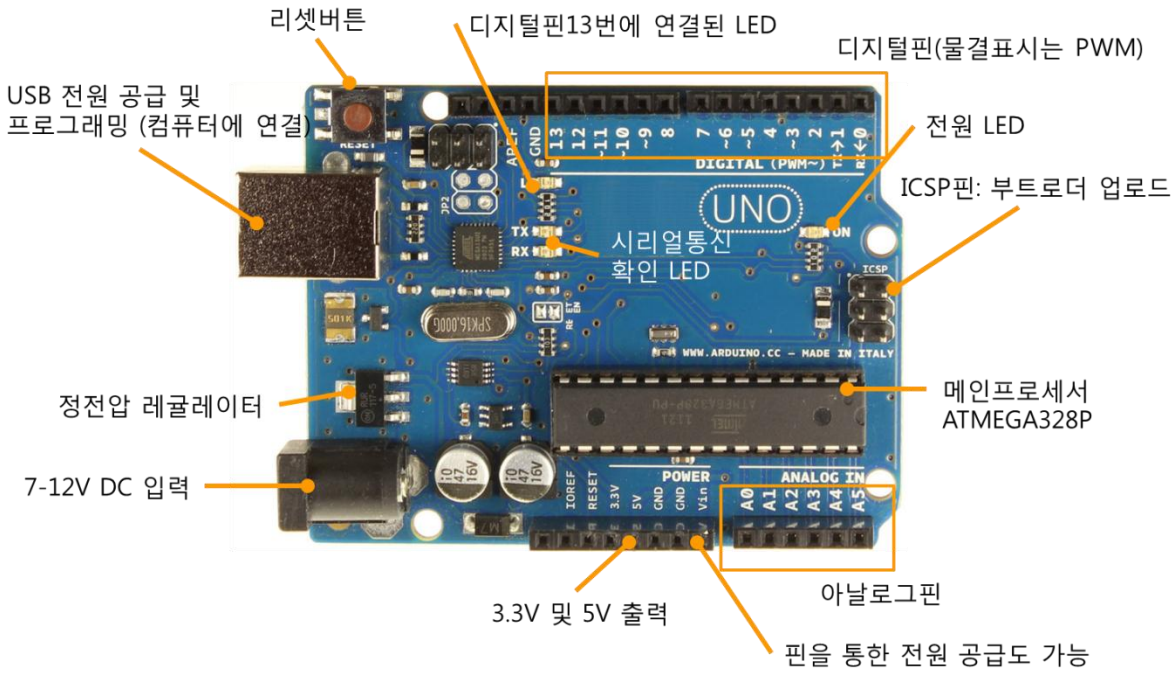


Fig. 2 우노 R3의 기본 부품 및 핀 정보

보드에 전원을 공급하기 위한 방법은 크게 세가지가 있습니다.

1. 배럴잭을 통한 전원 공급 (DC 어댑터를 사용하거나 배터리와 배터리홀더를 사용한 방법)
2. USB 케이블을 통한 전원 공급 (컴퓨터에 USB 케이블을 연결)
3. VIN과 GND를 통한 전원 공급 (배럴잭이 아닌 일반 전선을 사용할 경우)

Fig. 3 전원 공급 방법들

아두이노를 이용한 애플리케이션의 종류는 무한히 커지고 있는데, 그 이유 중 하나가 바로 쉴드의 등장 덕분이라 해도 과언이 아닙니다. 아두이노에서 불리는 쉴드의 개념은 기본적으로 우노 R3에 적층하여서 사용하는 비슷한 모양의 기판으로써, 모터 컨트롤, 통신, 디스플레이, 게임 인터페이스 등 다양한 기능을 전선 연결 혹은 납땜 없이 손쉽게 구현할 수 있도록 하였습니다. 다만, 적층을 하여서 사용하다보면 다른 기능들에 대한 확장성이 어려워지기 때문에 특정 기능이 아닌 범용으로 사용하기를 원하신다면 쉴드보다는 전선을 사용하여 원하는 센서, 모터, 디스플레이 및 통신 모듈과의 연결을 해 보심이 좋을 것입니다.

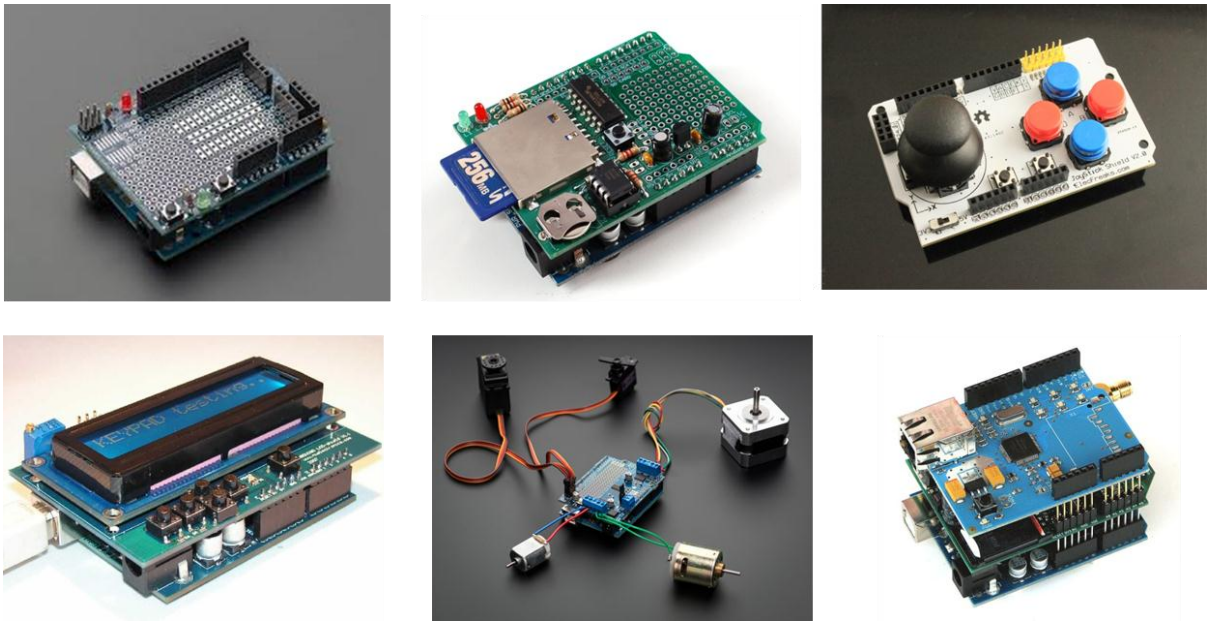


Fig. 4 아두이노 쉴드의 예

B. 소프트웨어의 설치

아두이노 Uno R3(이하 보드)가 알 수 있도록 코드를 써주기 위해서는 소프트웨어가 필요합니다. 다행스럽게도, 이 소프트웨어는 무료입니다. 인터넷이 연결되어 있다면 다음의 순서대로 소프트웨어를 설치할 수 있습니다.

먼저 아래 사이트에 접속합니다. 혹은 구글에서 “Arduino install”로 검색하셔도 됩니다.

http://arduino.cc/en/Main/Software

Windows 사용자

1) 보드를 USB로 연결하고 드라이버 설치과정을 시작할 때까지 기다립니다. 잠시 후 실패했다고 뜨게 되면 정상입니다.

2) 시작메뉴를 클릭하고 제어판을 클릭합니다.

3) 제어판에서 시스템 및 보안을 클릭합니다. 그 후 시스템을 클릭한 후 장치관리

자를 엽니다.

4) 포트부분(COM & LPT)에서 “Arduino Uno(COMxx)”라는 부분을 찾습니다.

5) “Arduino Uno(COMxx)”를 우 클릭한 후 “드라이버 소프트웨어 업데이트”를 클릭합니다.

6). “컴퓨터에서 드라이버 소프트웨어 찾아보기”를 클릭합니다.

7) 최종적으로 아두이노 폴더 안의 “Drivers” - “ArduinoUno.inf”파일을 클릭합니다. (“FTDI USB Drivers”가 아닙니다.)

8) 드라이버 설치가 완료되었습니다.

9) 아두이노 실행파일을 실행시킵니다.

10) LED blink 예제를 불러옵니다. (파일 > 예제 > 1.Basics > Blink)

11) 도구 – 보드 메뉴에서 Arduino Uno를 클릭합니다.

12) 시리얼 포트를 선택합니다. (만약 찾지 못했다면 보드를 연결 해제한 후 빠진 곳이 있는지 체크합니다.

13) 업로드 버튼을 클릭합니다.

14) “Done uploading”이라는 메시지가 나오면 보드의 USB포트부분 위에 있는 L부분이 반짝이는 것을 볼 수 있습니다.

MAC 사용자

1) 보드를 USB로 연결합니다.

2) 하드 드라이브에 아두이노 실행파일을 드래그 합니다.

3) 네트워크 설정에서 적용 혹은 “Apply”를 클릭합니다. (/dev/tty/usb.)

4) 프로그램을 실행합니다.

5) LED blink 예제를 불러옵니다. (파일 > 예제 > 1.Basics > Blink)

6) 도구 – 보드 메뉴에서 Arduino Uno를 클릭합니다.

7) 시리얼 포트를 선택합니다. (만약 찾지 못했다면 보드를 연결 해제한 후빠진 곳이 있는지 체크합니다.

8) 업로드 버튼을 클릭합니다.

9) “Done uploading”이라는 메시지가 나오면 보드의 USB포트부분 위에 있는 L부분이 반짝이는 것을 볼 수 있습니다.

C. 통합개발환경에 대하여

아두이노 소프트웨어가 설치가 되면 다음의 스케치 창을 확인할 수 있습니다. 아두이노 보드를 선택하고, 연결된 시리얼포트를 설정하며, 아두이노를 통해서 어떤 일을 할지 프로그래밍을 쓰고, 보드에 업로드하는 기능을 수행합니다. 이렇듯, 다양한 일을 하기 때문에 통합 개발 환경 (IDE)이라고도 부릅니다. 다음은 메뉴 아이콘들에 대한 설명입니다.

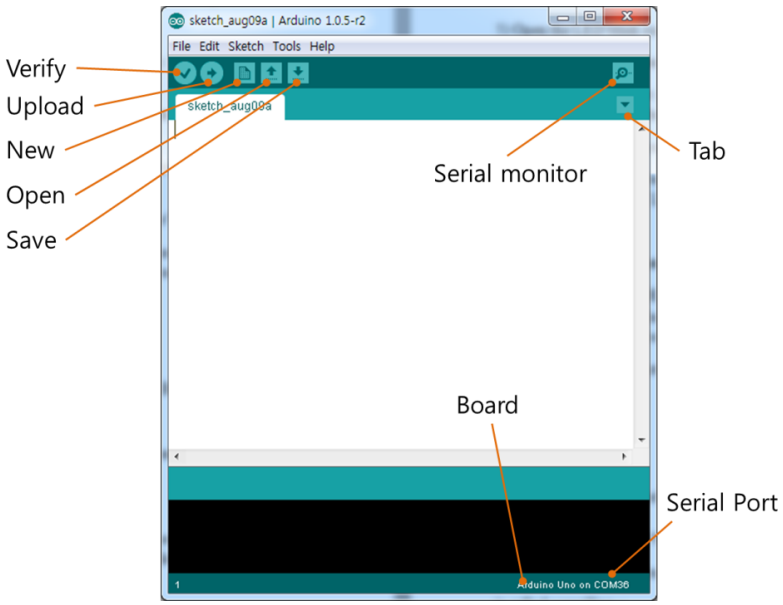


Fig. 5. Arduino IDE 아두이노 통합 개발 환경

• 확인(Verify) : 코드를 보드에 보내기 전에 보드가 이해할 수 있도록 명령어를 변경합니다.

• 업로드(Upload) : 확인, 번역한 후에 USB케이블로 보드에 전송합니다.

• 새 파일(New) : 새 스케치를 만들기 위해 새 창을 띄웁니다.

• 열기(Open) : 컴퓨터에 있는 다른 스케치 파일을 엽니다.

• 저장(Save) : 현재 작업을 스케치로 변환하여 저장합니다.

• 시리얼 모니터(Serial monitor) : 아두이노에서 매우 유용한 기능입니다. USB를 통해서 보드로 메시지를 보내거나, 받을 수 있습니다.

• 탭(Tab) : 여러 개의 스케치를 작업 할 수 있게 해줍니다. 이 수업에서 하는 것보다 더 고급 프로그래밍을 할 때 사용됩니다.

D. Blink 예제 구현해보기

C 혹은 Java 등의 다른 프로그래밍 언어를 배울 때 기본이 되는 코드가 "Hello World"라는 문구를 모니터에 출력을 해 봤다면, 아두이노를 배울 때 기본이 되는 예제가 LED를 깜빡이는 것일 것입니다. 다른 언어는 소프트웨어 자체를 배우는 것임에 비해, 아두이노는 소프트웨어와 하드웨어를 동시에 배울 수 있기 때문에 소위 코딩 (소프트웨어)을 통해서 LED가 깜빡이는 (하드웨어) 예제를 알아보도록 하겠습니다.

준비물 : Uno R3, USB cable

배경 : Uno에는 디지털 13번 핀과 연결된 내장 LED가 있습니다. USB로 PC와 보드를 연결하면 깜빡이는 LED를 볼 수 있습니다.

스텝 1:

파일 – 예제 – Basics – Blink 에 들어가서 예제 파일을 불러옵니다. 그러면 아래와 같은 새 창이 뜨게 됩니다.

코드:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

설명:

- /* comments */ 소위 주석이라 부르며, /*으로 시작해서 */로 끝납니다. 이 부분은 컴파일러가 무시하며 과정에 포함되지 않습니다. 그래서 이 부분은 Atmega에서 아무런 공간을 차지하지 않습니다.
- 한 줄인 경우에는 //를 사용합니다.
- 보드에 있는 LED의 경우 디지털 13번 핀과 연결되어 있습니다. 그래서 코드에서는 정수로 핀 번호를 13번으로 정의했습니다.

- 모든 맨 끝 문장에는 세미콜론이 붙어 있습니다.
- setup()안에서는 한번만 실행될 것을 넣어주세요. 만약 반복할게 있다면 loop() 안에 넣어주세요,
- pinMode(led,OUTPUT)은 pinMode(13,OUTPUT)과 같습니다.
- digitalWrite(led, HIGH)가 LED를 켜고 뒤에 나오는 delay(1000) 함수가 1000 ms만큼 프로그램을 일시 정지시킵니다.

스텝 2:

도구 – 보드 – Arduino Uno를 클릭하여 보드를 선택합니다. 그 후 도구 – 시리얼 포트에 가면 여러 개의 포트가 있습니다. USB를 뺐다 꽂으면 사라졌다가 나오는 포트가 있습니다.

스텝 3:

확인을 클릭한 후 업로드를 클릭합니다. 그리고 동작을 확인합니다.

E. 구조, 변수, 함수

소위 구조, 변수, 함수가 프로그램을 구성하는 요소인데, 반드시 알았으면 하는 것들을 아래에 설명하였습니다. 모두 다 이해할 필요도 암기할 필요도 없지만, 적어도 setup()과 loop()에 대해서는 이해하도록 합시다.

- 구조

- setup();

  setup() 함수는 스케치를 시작할 때 불러옵니다. 초기 변수, 핀 상태, 사용 라이브러리 시작등에 사용합니다. setup기능은 전원이 켜졌을 때 혹은 리셋 버튼을 눌렀을 때 한번만 실행됩니다.

- loop():

  setup()을 생성한 후에 loop() 함수가 프로그램의 변화와 응답 내에서 연속적으로 작동합니다. 아두이노 보드를 제어하는데 적극적으로 사용하세요.

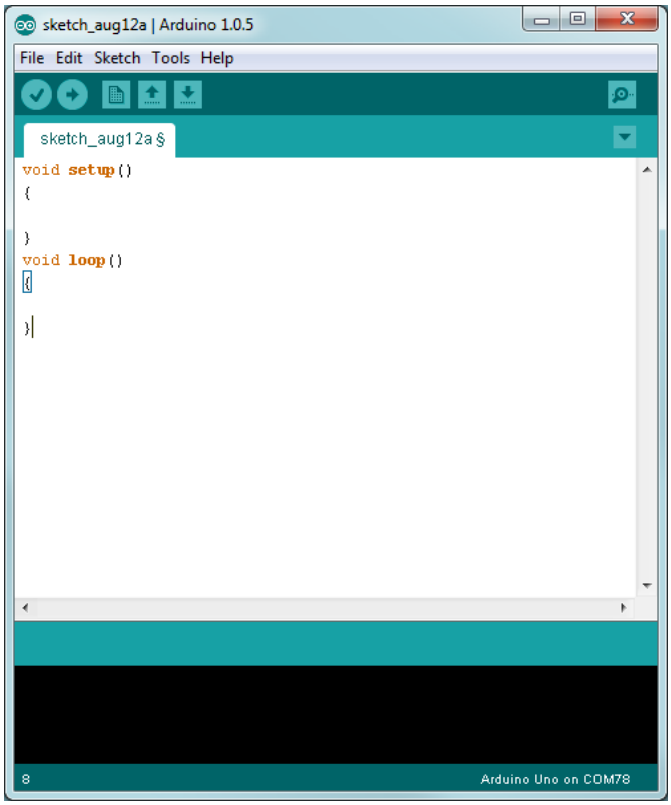


Fig. 6. 기본 구조 – setup and loop

위 Fig. 3에 보이는 바와 같이 setup()과 loop()는 아두이노에서 컴파일과 업로드를 하기 위해 기본적으로 요구되는 부분입니다.

- if

if / else 문은 여러 개의 그룹으로 묶여있는 코드의 흐름을 더 부드럽게 합니다. 예를 들어 아날로그 입력 값이 500미만일 때 A번 행동(action A)이 실행된다면 다른 행동(action B)은 500이상일 때 실행됩니다. 코드는 아래와 같습니다.

```
if (pinFiveInput < 500)
{
  // action A
```

```
}
else
{
   // action B
}
```

else는 if문에서 언급한 조건이 맞지 않을 때 실행되며, 이도 저도 아닐 경우에 실행되는 구문입니다. 즉, 참이 되는 것을 찾을 때까지 진행되며, 그렇지 않으면 else 구문이 실행됩니다.

else if 문은 if문과 접속하고 else문과 접속하기도, 안하기도 합니다. 이 Else if문은 무한하게 쓸 수 있습니다.

다음의 코드는 pinFiveInput이라는 값이 500보다 작으면 A를 실행하고(action A), 1000보다 크거나 같으면 B를 실행합니다 (action B). 그리고 어느 쪽에도 해당되지 않으면 C를 수행합니다 (action C).

```
if (pinFiveInput < 500)
{
   // action A
}
else if (pinFiveInput >= 1000)
{
   // action B
}
else
{
   // action C
}
```

예를 들어, 100이라는 숫자가 pinFiveInput에 할당되면 A, 700이면 B 1100이면 C 이런 형태입니다.

- for:

for 구문은 외부로부터 격리된 채로 반복하기위해 쓰입니다. 증가하는 카운터는 루프의 종료와 증가를 위해 쓰입니다. for 구문은 반복되는 동작에서 매우 유용하며 종종 배열화된 데이터나 핀의 무리에서 동작하기 위해 사용합니다.

for (initialization; condition; increment) {

//statement(s);

}

- 변수

- HIGH

HIGH의 의미(핀에서 사용하는 부분)는 핀의 설정이 INPUT모드냐 OUTPUT모드냐에 따라 다릅니다. pinMode가 INPUT모드일 때, digitalRead를 썼을 경우 3V이상이 들어 올 경우입니다. 또 digitalWrite와 함께 쓸 수도 있습니다. 그러면 내부 20K의 풀업저항이 설정됩니다. 그럼 이 핀은 외부 회로에 의해 LOW가 되기 전까지 HIGH가 됩니다. 이것은 INPUT_PULLUP으로 불립니다.

pinMode가 OUTPUT으로 설정되었을 경우 digitalWrite에서 HIGH로 설정하면 5V가 출력됩니다. 이 상태에선 전류를 흘려 보낼 수 있습니다. 예로 LED등을 킬 수 있습니다. 이것은 저항을 통해 ground로 연결될 때 LOW상태가 됩니다.

- LOW

LOW의 의미도 INPUT과 OUTPUT일 때 다른 의미입니다. pinMode가 INPUT일 경우 digitalRead를 썼을 때, 2볼트 이하인 상태입니다.

OUTPUT상태일 때 digitalWrite에서 LOW로 설정하면 핀은 0볼트가 됩니다. 전류를 받아 들일 수 있습니다. 예로 5V와 저항에 연결된 LED를 연결하게 되면 LED

에 불이 들어옵니다.

- true

true는 종종 1로 정의됩니다. 이것은 '참'이란 의미지만 실제 true는 더 넓은 의미입니다. Boolean에서 0이 아닌 모든 정수는 '참'입니다. 그래서 -1이나 2, -200등은 모두 참으로 정의됩니다.

- false

false는 0으로 정의됩니다.

- Functions

- pinMode(pin, mode)

핀의 상태를 입력 혹은 출력으로 설정합니다. 핀의 목적을 보다 자세하게 묘사해 줍니다.

- digitalWrite(pin, value)

디지털 핀에 위에서 설명한 HIGH 혹은 LOW 를 입력합니다.

- digitalRead(pin)

디지털 핀의 구체적인 값을 읽습니다. HIGH 혹은 LOW 가 나옵니다.

- analogRead(pin)

구체적인 아날로그 핀의 값을 읽습니다. 보드에는 6개의 채널이 있으며(나노와 미니에는 8개) 10비트의 ADC(아날로그 디지털 컨버터)가 있습니다. 이것은 0 ~ 5볼트가 0 ~ 1023의 값으로 변환되는걸 의미합니다.

- analogWrite(pin, value)

핀에 아날로그 값(PWM 형식)을 넣습니다.

2. 모바일 로봇 기초

A. 기구학이란?

기구학 (Kinematics)는 움직이는 물체에 대한 상대적인 운동을 다루는 학문을 말합니다. 즉, 특정한 자세를 기하학적으로 표현한 것을 의미하며, 힘을 다루는 동역학과 달리, 힘이나 토크가 고려되지 않은 움직임을 표현하기 위한 학문을 의미합니다. 특히, 모바일 로봇 연구를 하거나 해석을 할 때, 가장 기본이 되는 스텝으로 Kinematic Analysis (기구학 해석)일 정도로 중요하다고 할 수 있습니다. 이 기구학은 모션 제어를 위해서 더 나아가 동역학적 해석을 하기 위해서 기본이 됩니다.

로봇의 자세(pose)는 위치(position)와 방향(orientation)으로 구성되는데, 일반적으로 모바일 로봇은 다음의 세가지로 로봇의 자세를 표현할 수 있습니다.

$$P = f(x, y, \theta)$$

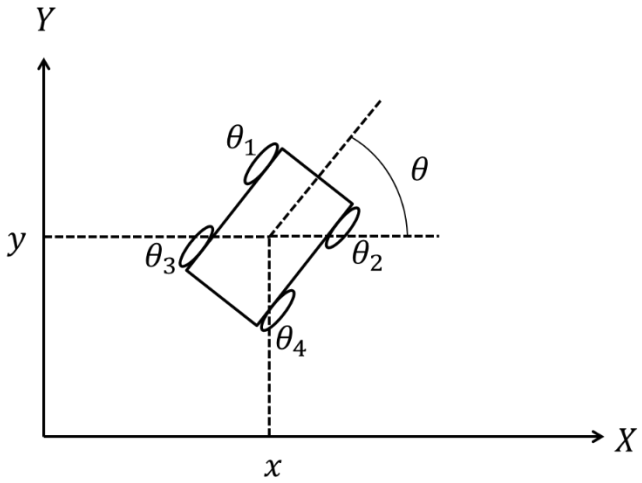


Fig. 7. 4바퀴로 구동되는 모바일 로봇

위 그림 (Fig. 7)에서 로봇의 자세는 직교 좌표계인 $X-Y$ 좌표계에서 로봇은

x, y, θ로 표현될 수 있습니다.

기구학에는 정기구학인 (Forward kinematics, 포워드 키네매틱스)와 역기구학 (Inverse kinematics, 인버스 키네매틱스)가 있는데, 정기구학은 구하려는 로봇의 자세인 x, y, θ 를 표현하기 위해 알고 있는 정보인 바퀴의 회전 각도 $\theta_1, \theta_2, \theta_3, \theta_4$ 를 사용하는 것이고, 역기구학은 x, y, θ를 알고 있는 상태에서 모르고 있는 바퀴의 회전 각도 $\theta_1, \theta_2, \theta_3, \theta_4$를 구하는 것입니다.

B. 자유도 (Degree of Freedom, DOF)란?

자유도란? 공간상에 있는 어떠한 시스템의 위치를 표현하기 위해 필요한 최소 독립 변수의 수를 의미합니다. 2D에서는 x축과 y축의 병진운동 그리고 회전을 포함한 3자유도를 갖게 되고, 3D공간에서는 x, y, z축 방향으로의 병진운동과 x, y, z에 대한 회전 운동을 포함한 6자유도를 갖게 됩니다. 2D상에서 운동하는 모바일 로봇의 경우, 3자유도를 가지게 되면 어느 곳이든 움직일 수 있는 시스템이 되고 (holonomic system), 3자유도가 넘어가면 불필요한 액츄에이터를 가진 시스템이 됩니다 (Redundant), 그리고 로봇의 자유도가 3자유도보다 적으면 nonholomonic system이 됩니다. 홀로노믹과 논홀로노믹에 대해서는 다음장에 설명하도록 하겠습니다.

C. 홀로노믹 & 논홀로노믹(Holonomic & Nonholonomic)이란?

홀로노믹 혹은 논홀로노믹이라는 말은 로봇의 구속조건을 이야기하면서 거론되는 용어입니다. 쉽게 설명하자면, 모바일 로봇에서 홀로노믹은 어느 방향으로도 움직일 수 있는 로봇을 의미하기 때문에 모션 제어가 쉽다는 장점이 있습니다. 이에 반해서 대부분의 모바일 로봇은 논홀로노믹인데, 순간적으로 어떤 방향으로 움직일 수 없기 때문에 어떤 위치로 이동을 하기 위해서는 낙엽이 떨어지듯 왔다갔다 하거나, 자동차를 사이드 주차하기 위해서 핸들을 여러 번 돌리는 것과 같은 과

정이 필요합니다. 홀로노믹 시스템으로 대표적인 것은 메카넘휠과 같은 특수한 바퀴를 사용하여 어느 방향이든 이동할 수 있도록 설계된 Omnidirectional Robot 을 들 수 있습니다.

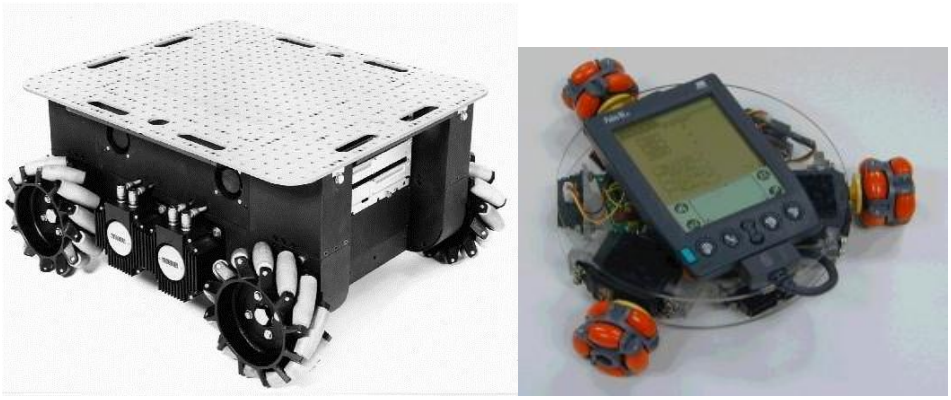


Fig. 8 Omnidirectional Robot

Fig. 9 Omnidirectional Wheels

홀로노믹 및 논홀로노믹 시스템을 수학적으로 설명할 때는 위치 변수들(position variables)을 이용하여 자명한 함수로 표현할 수 있을 때, 홀로노믹 시스템이라고 하고, 위치 변수뿐만 아니라 속도 변수가 포함된 경우 논홀로노믹 시스템

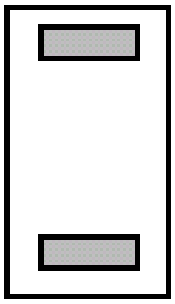
(Nonholonomic system)이라고 부릅니다. 자유도를 이용하여 홀로노믹, 논홀로노믹을 설명할 수도 있는데, 컨트롤할 수 있는 자유도의 개수가 전체 자유도와 같으면 홀로노믹이고, 컨트롤할 수 있는 자유도가 전체 자유도보다 작다면 논홀로노믹, 그리고 컨트롤 자유도가 전체 자유도보다 크면 **리던던트(Redundant)** 시스템이 됩니다.

홀로노믹(Holonomic): Controllable DOF = Total DOF
논홀로노믹(Nonholomomic): Controller DOF < Total DOF
리던던트(Redundant): Controllable DOF > Total DOF

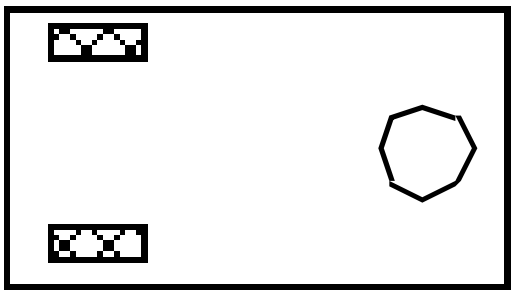
예를 들어, 2차원상을 움직이는 모바일 로봇의 전체 자유도는 3입니다 (x, y, θ). 만약, 위의 그림 Fig. 2의 오른쪽 로봇처럼 Omnidirectional 바퀴가 세 개라면 (Controllable DOF = 3), 우리는 이 바퀴 세개를 컨트롤하게 되고, 세 개만으로 로봇의 위치 (Total DOF = 3)를 모두 표현할 수 있습니다. 반면에 Fig. 2의 왼쪽 로봇처럼 바퀴가 4개가 있다면, 이는 Controllable DOF가 4라서 전체 자유도 3보다 큰 시스템, 즉 리던던트한 시스템이 됩니다. 리던던트라는 말은 불필요하게 많다는 의미입니다.

D. 바퀴의 배치

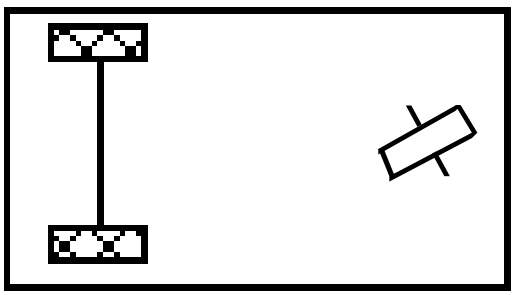
바퀴 1개 + 조향 1개 – 예) 자전거



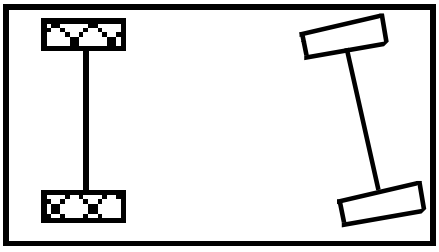
바퀴 2개 – 예) 세그웨이



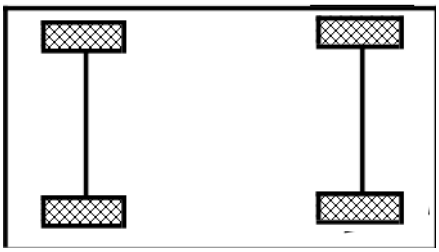
바퀴 2개 + 볼캐스터 1개 (바퀴 2개와 큰 차이는 없지만, 무게 중심이 다르다.)



바퀴 2개 + 조향 1개



바퀴 2개 + 조향 2개



바퀴 4개

E. 모바일 로봇의 인식 (Perception)

모바일 로봇을 공부하면서 중요한 것 중 하나가 로봇을 둘러싼 환경을 인식하는 것입니다. 이는 자율 이동 로봇 (Autonomous robots)에서 더 중요하며 다양한 센서를 통해서 의미 있는 정보를 얻을 수 있습니다. 다양한 센서가 모바일 로봇 연구에 사용되는 가운데, 많은 연구가들에 의해서 사용되는 것이 적외선 센서, 초음파 센서, 카메라, 그리고 자기장 센서들이 있습니다. 장애물을 인식하기 위해서 직접적인 접촉을 통해서 사물을 인식하는 방법도 있고 (피에조 엘레먼트, 압전센서), 비접촉 방식을 통해서 인식하는 방법도 있습니다 (적외선 거리센서, 초음파 거리센서, 카메라). 사물 및 장애물을 인식하는 것 외에도, 방사능 센서와 가스 센서를 통해서 위험한 곳을 감지하고 인명을 구출하는 로봇 및 재난용으로도 센서가 사용되고 있습니다. 모바일 로봇의 방향을 알기 위해서 자기장 센서를 사용하기도 합니다. 자기장 센서는 자구 자기장에 대해서 절대적인 방향을 알려주기 때문에 북쪽이 어디인지, 남쪽이 어디인지 가르쳐주는 유용한 센서 중 하나입니다. 또한, 바퀴가 얼만큼 회전했는지 측정하기 위해 인코더나 홀센서를 사용하기도 합니다. 로봇에 사용되는 센서는 싸게는 몇백원에서 비싸게는 몇백~몇천만원의 고가 센서도 있습니다. 원천적으로 개발된 센서가 비싼 경우도 있지만, 다양한 센서들끼리의 통합(integration) 및 퓨젼(fusion)을 통해서 센서의 성능을 향상시키기도 합니다. 이는 즉, 센서의 부가가치와도 연관이 됩니다.

센서의 퓨젼으로 거론되는 것 중 하나가 칼만필터라는 것입니다. 기본적으로 센서들은 노이즈라는 것을 가지고 있습니다. 이는 센서의 성능과도 연관이 되기 때

문에 노이즈를 줄이기 위한 다양한 알고리즘이 개발되었고, 칼만이라는 사람이 이를 줄이기 위해서 고안한 알고리즘이 칼만필터라는 것입니다. 간단히 설명하자면, 공분산이라는 수학적 도구를 사용하여 보다 정확한 값을 추정하는 과정입니다. 이 밖에도 보상필터라는 것도 자주 사용되는데, 이는 센서간의 단점을 보완하고 장점들을 취함으로써 보다 정확한 값을 얻을 수 있게 합니다. 센서의 통합으로는, A라는 센서가 검출하지 못하는 값을 B라는 센서를 도입함으로써 얻을 수 있도록 하는 과정을 의미하며, 다양한 모바일 로봇에서 센서 퓨전 및 통합을 통해 유용한 정보를 얻고 있습니다.

값 비싼 센서들에도 베스트셀러가 있는데, 다음에는 그 센서들을 거론해보고자 합니다.

거리를 측정하기 위해서는 **1) 적외선 센서와 2) 초음파 센서**가 주로 사용됩니다.

최근에는 저가의 고성능의 센서가 개발되고, 또 대량센서로 인한 좋은 센서가 값싸게 거래되는 등, 가성비로 적외선 센서와 초음파 센서를 비교하는 것이 어려워지고 있습니다. 두 센서 모두 빛 혹은 음파가 사물에 반사되어 돌아오는 것에 대한 정보를 이용하여 거리를 추정하는데, 초음파 센서를 여러 개 사용하다 보면 음파끼리 간섭을 하여 거리 측정에 오차가 생기는 경우도 있고, 적외선 센서는 초음파 센서에 비해서 상대적으로 비싸다는 단점이 있습니다. 일반적으로 초음파 센서가 비싸다고 알려지기도 하는데, 이는 어디까지 브랜드가 있는 센서를 사용하였을 때 이야기이고, HC-SR04 초음파 센서와 같이 중국에서 대량으로 생산되는 센서의 가격과 Sharp사의 적외선 센서를 비교하자면 HC-SR04 초음파 센서가 저렴한 것은 지극히 자연스러운 것일지도 모르겠습니다.

HC-SR04 초음파 센서

http://mechasolution.com/shop/goods/goods_view.php?goodsno=119&category=002002

Sharp GP2Y0A21YK (10~80cm) 적외선 센서

http://mechasolution.com/shop/goods/goods_view.php?goodsno=96&category=002002

회전방향을 측정하기 위해서는 **1) 가속도 센서, 2) 자이로 센서, 그리고 3) 자기장 센서**가 사용됩니다. 가속도 센서는 병진 방향으로의 가속도를 측정하는데 사용되기도 하지만, 각 축에서 나오는 센서 값은 지구의 중력 가속도에 대한 상대적인 값으로써, 센서를 회전하면 각 축으로부터 나오는 값이 변하게 됩니다. 이에 따라서 센서가 어느 정도 회전되었는지 측정하게 되고, 가속도 센서는 회전을 검출하는데 사용되고 있습니다. 자이로 센서는 가속도 센서와 마찬가지로 회전을 측정하기 위해서 사용됩니다. 자이로 센서는 각속도를 검출하며, 회전 각도를 얻기 위해서는 이를 시간에 대해서 적분을 하여 얻게 되는데, 아무리 작은 시간에 대해서 적분을 하여도 오차가 존재하기 때문에 시간이 지나면 가만히 있는 센서라도 얻은 값은 오차를 누적하게 됩니다. 이를 소위 자이로 센서의 드리프트라 부르며 이를 보완하기 위해서 가속도 센서와 자기장 센서를 사용하여 센서 퓨전하곤 합니다. 가속도 센서는 지구 연직 방향에 대해서 회전하는 물체 (예를 들어 팽이와 같은 회전)에 대해서는 회전 각도를 얻을 수 없기 때문에 모든 방향으로의 회전 각도를 얻기 위해서는 자이로 센서와의 센서 퓨전이 필수입니다. 자기장 센서는 지구의 자기장에 대한 값으로 절대 회전 방향을 얻기 위해서 사용되며 줄곧 가속도 및 자이로 센서와 함께 사용되곤 합니다.

MPU-6050 3축 가속도 + 3축 자이로 센서

http://mechasolution.com/shop/goods/goods_view.php?goodsno=6&category=002001

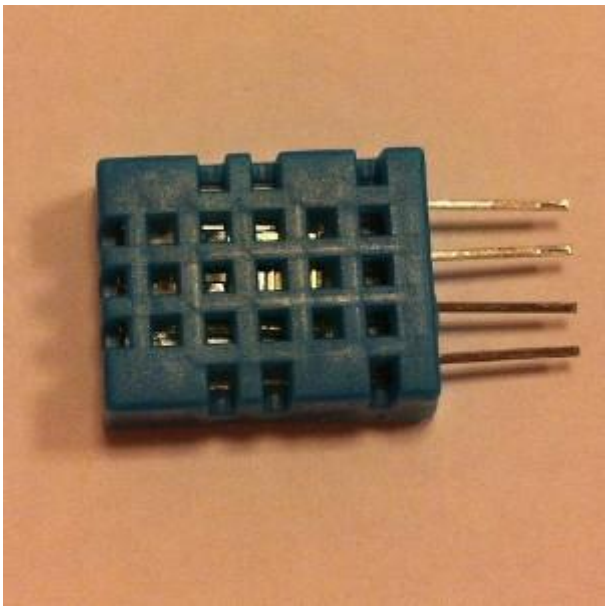
HMC5883L 3축 자기장 센서

http://mechasolution.com/shop/goods/goods_view.php?goodsno=28&category=002001

모바일 로봇은 또한 물리적인 거리 및 회전을 감지하는 것 외에도 온도, 습도, 가스 등을 측정하여 사람이 접근하기 어려운 곳으로부터 환경에 대한 정보를 제공해주기도 합니다.

MQ 시리즈 가스 센서 (일산화탄소, 메탄가스, 연기, 가연성 가스 등등)

http://mechasolution.com/shop/goods/goods_view.php?goodsno=1308&category=002015



온도 습도 센서

http://mechasolution.com/shop/goods/goods_view.php?goodsno=224&category=002006

카메라는 우리가 바라보는 환경을 로봇이 바라볼 수 있는 가장 정확한 방법이지만, 카메라로부터 얻을 정보를 매 순간 처리하기 위해서는 시스템이 복잡해지기 하며, 단순한 센서로 대체할 수 있는 부분이 있기 때문에 사용할 때 고려를 해봐야 합니다. 예를 들어 컬러를 인식하거나 빛의 밝기 혹은 사물과의 거리를 측정하기 위해서는 카메라 혹은 스테레오 카메라를 사용하기 보다는 컬러 센서, 조도 센서, 혹은 거리 센서를 사용하는 것이 적합합니다. 하지만, 고성능의 프로세서를 통해서 보다 많은 정보를 얻기 위해서는 카메라를 사용하시는 것이 적합할 것입니다. 카메라를 사용하실 경우에는 이미지 프로세싱 혹은 비젼 프로세싱이라는 분야를 공부하시면 도움이 될 것입니다. 이 밖에도 더듬이로 사용할 수 있는 구부림 센서, 빛을 따라갈 수 있도록 조도센서 등을 사용할 수 있습니다. 다양한 센서들을 사용하여 모바일 로봇이 환경에 인식할 수 있도록 구현할 수 있습니다.

F. 모바일 로봇의 위치 추정 (Localization and Navigation)

모바일 로봇의 위치를 추정하는 연구도 활발히 이루어지고 있는데, 실내에서의 위치 추정과 실외에서의 위치 추정은 다른 접근하곤 합니다. 위성 GPS가 동작하지 않는 실내에서는 Indoor GPS라는 방식을 사용하여 로봇의 위치를 추정하기도 하고, 전파, 음파 등을 방출하는 고정 물체 (beacon)들을 통해서 위치를 추정하기도 합니다. 다양한 알고리즘과 테크닉이 개발되고 시현되고 있습니다. 센서로부터 주변 환경을 인식하는 것부터 모터를 구동하여 로봇을 특정 위치에 이동하기까지는 크게 두 가지 방식을 이야기 할 수 있는데, 하나는 행동을 기반으로 한 네비게이션 (Behavior-based navigation) 과 다른 하나는 맵 혹은 모델을 기반으로 한 네비게이션입니다 (Map-based or model-based navigation).

행동을 기반으로 한 네비게이션은 1) 센서로부터 값을 얻은 후에 2) 다양한 정보들을 통합하여 매순간 새로운 이동 명령을 내립니다. 이를 위해서는 센서로 얻은

값들 및 통신을 통한 값들을 업데이트하며 장애물을 회피하거나 새로운 골 포지션으로의 이동 등 변화에 적응하는 시스템에 유리하게 작용할 수 있습니다

맵을 기반으로 하는 네비게이션은 초기 센서로부터 얻은 값을 통해 최종 위치 및 경로를 주어진 모델 및 맵을 통해 계산 한 후에 이동하게 됩니다.

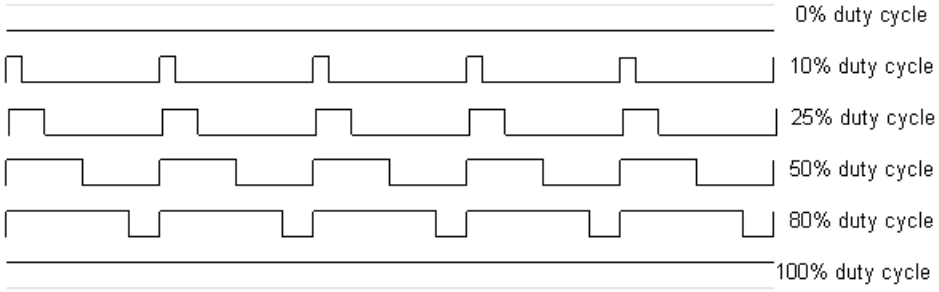
두 기법 모두 맵이 완전하거나 센서값이 신뢰할 수 있는 경우에 사용될 수 있는 방법이기 때문에 네비게이션 이전에 신뢰성있는 센싱 및 맵핑이 중요하다고 볼 수 있습니다.

3. 액츄에이터, 센서, 그리고 통신

A. 서보 모터의 제어

Pulse Width Modulation이란?

Pulse width modulation(이하 PWM)은 펄스의 폭을 컨트롤 하는 주기 제어방법입니다. “On”되는 시간에 따라 그 주기가 달라집니다. 주기가 낮다면 그에 따라 전압이 약해집니다. 왜냐하면 전압이 꺼지는 시간이 대부분이기 때문입니다. 다음 그림은 주기를 퍼센트로 나타낸 것입니다.



PWM의 사용 용도:

- LED 흐리게 하기
- 아날로그 출력하기
- 오디오 신호 만들기
- 모터 공급용 속도조절하기

어떻게 서보모터가 회전하는가

반이중 시리얼 통신을 사용하는 서보모터(Dynamixel 이나 Herkulex)를 제외하면 대부분의 서보모터는 PWM으로 제어됩니다.

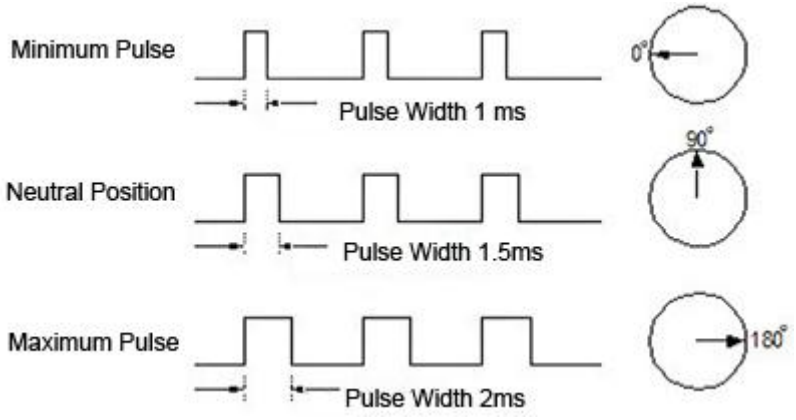


Fig. 10 펄스의 주기로 서보모터의 각 조절하기

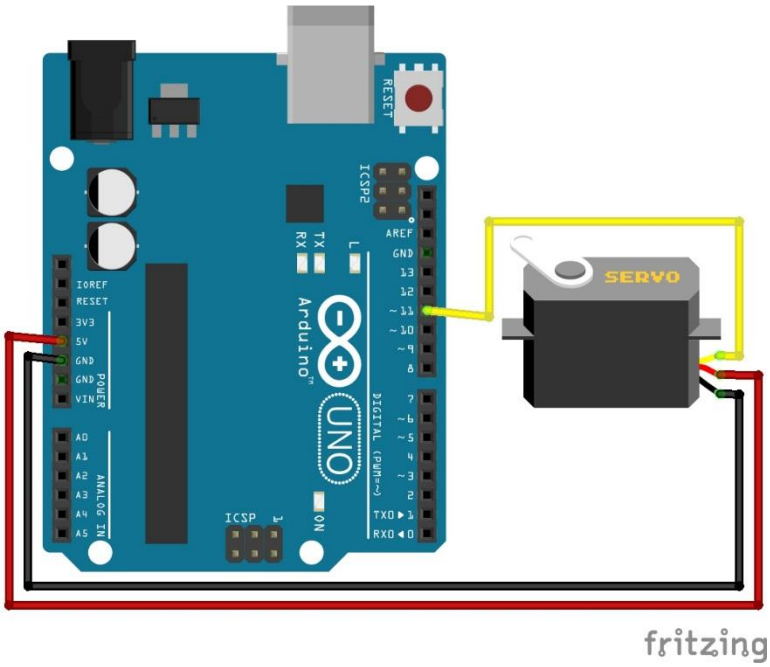
위에서 알아 낸 것은 어떻게 서보모터가 회전하는가 입니다. 서보모터는 공급되는 PWM신호로 위치 제어가 가능하도록 회로가 내장되어 있습니다.

서보모터 조종하기

준비물 : Uno R3, 브레드보드, USB cable, 미니 서보모터 (SG90), 브레드보드 점퍼 와이어

배경 : Servo.h 라이브러리는 서보모터를 제어하기 더 편하게 해줍니다. Servo.h는 아두이노 보드로 12개를 지원하며 메가는 48개를 지원합니다. 서보모터는 3개의 선이 있습니다. 전원, 그라운드, 신호선입니다. 보통 그라운드는 검은색이나 갈색이며 전원선은 빨강, 신호는 노랑이나 주황입니다. 미니 서보는 4 ~ 7.2V를 전원으로 사용할 수 있으며 여기선 약 1.2 kg/cm(16.7 oz/in)의 토크를 가집니다. 만약 이 서보 (SG90)로 더 큰 힘을 내고 싶다면 외부 전원선을 연결하면 됩니다.

회로: delay함수를 이용하여 반복적으로 회전하는 서보 운동입니다.



서보모터 핀	아두이노 핀
갈색	GND

노란색	D11
빨간색	5V

코드:

```
#include <Servo.h>

Servo myservo;    // create servo object to control a servo


void setup()
{
   myservo.attach(11);    // attaches the servo on pin 11 to the servo object
}

void loop()
{
   myservo.write(10);              // servo rotates to 10 degree
   delay(1000);                      // waits for 1 second
   myservo.write(170);             // servo rotates to 170 degree
   delay(1000);                      // waits for 1 second
}
```

B. DC 모터의 제어

SN754410은 쉽고 편리한 IC로 DC 모터의 속도와 방향을 컨트롤할 수 있게 합니다. 이번 예제를 통해 쉽고 간단하게 DC모터의 방향을 제어해보겠습니다. 또한, 본 예제와 가속도 센서를 이용해 세그웨이와 같은 밸런스 로봇도 제작할 수 있고, 모터를 이용한 여러 가지 애플리케이션을 제작할 수 있습니다.

준비물: 아두이노 우노, USB 케이블, 브레드보드, SN754410, DC 모터, 점퍼 케이블

배경지식:

SN754410을 이용하여 한 개의 DC모터를 컨트롤 하기 위해서는 1개의 PWM, 2개의 디지털 아웃풋이 요구됩니다. 1개의 PWM은 스피드를 컨트롤하기 위함이며,

2개의 디지털 아웃풋은 모터의 방향을 결정하기 위함입니다. 다음의 SN754410 칩의 함수 테이블을 보면 A가 H(high)일 때, 아웃풋 Y는 HIGH이고, A가 L(low)일 때, 아웃풋은 L이다. 모터 컨트롤러를 왜 사용하는가 생각할 수 있습니다. 아두이노에서 나오는 아날로그 아웃풋을 이용하여 컨트롤할 수도 있고, 혹은 트랜지스터와 다이오드를 이용하여 컨트롤할 수도 있습니다. 하지만, 모터에서 사용하는 전류가 보통 100mA이상이므로 40mA를 출력하는 아두이노에 직접적으로 연결하는 것은 옳지 않으며 간단히 모터 컨트롤러 칩을 사용함으로써 두 개의 모터를 제어하는 편리함 때문에 이 번 예제에서는 트랜지스터와 다이오드를 이용한 모터컨트롤 대신 SN754410을 이용한 모터 컨트롤을 하기로 했습니다. SN754410의 핀 정보를 보면 총 4개의 인풋이 있습니다 – 1A, 2A, 3A, 4A. 이 4개의 인풋을 이용하여 두 개의 모터 방향을 결정할 수 있습니다. 즉, 1A와 2A를 이용하여 1Y, 2Y의 값을 컨트롤하고, 3A와 4A를 이용하여 3Y, 4Y를 컨트롤합니다. 1Y, 2Y는 첫번째 모터에, 그리고 3Y, 4Y는 두번째 모터에 각각 연결됩니다. 그리고, 1,2EN핀과 3,4EN핀에 PWM 값을 입력해주면서 전압에 따른 모터의 속도 컨트롤이 가능합니다.

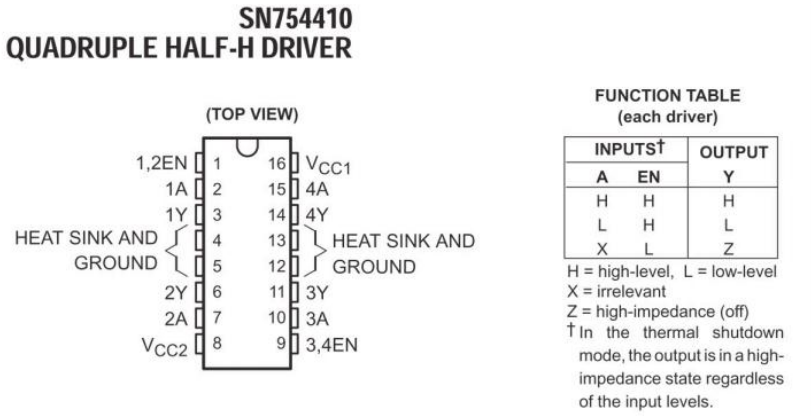


Fig. 11 SN754410의 핀 정보 및 함수 테이블: 모터의 출력 Y

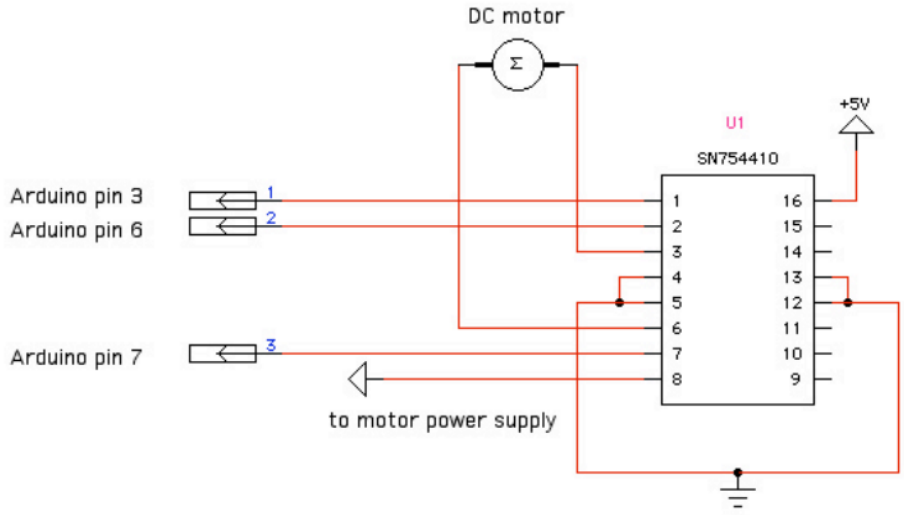


Fig. 12 회로 다이어그램

따라하기:

- 1개의 DC모터를 컨트롤하기 위해서 다음의 회로를 구성해 봅시다. SN754410을 브레드보드에 꼽습니다. 아래 회로에서 왼쪽에 있는 작은 홈이 있는 방향이 1번핀이 시작되는 방향입니다.
- 모터에서 나오는 두 개의 신호선은 + 혹은 –에 대한 정의가 없으니, 임의로 결정을 한 후에 브레드보드에 끼워도 무방하며 방향은 컨트롤러에서 결정할 수 있습니다. 혹은 회전하는 방향에 따라 모터에서 나오는 갈색과 보라색의 선을 바꾸어 끼워도 됩니다.

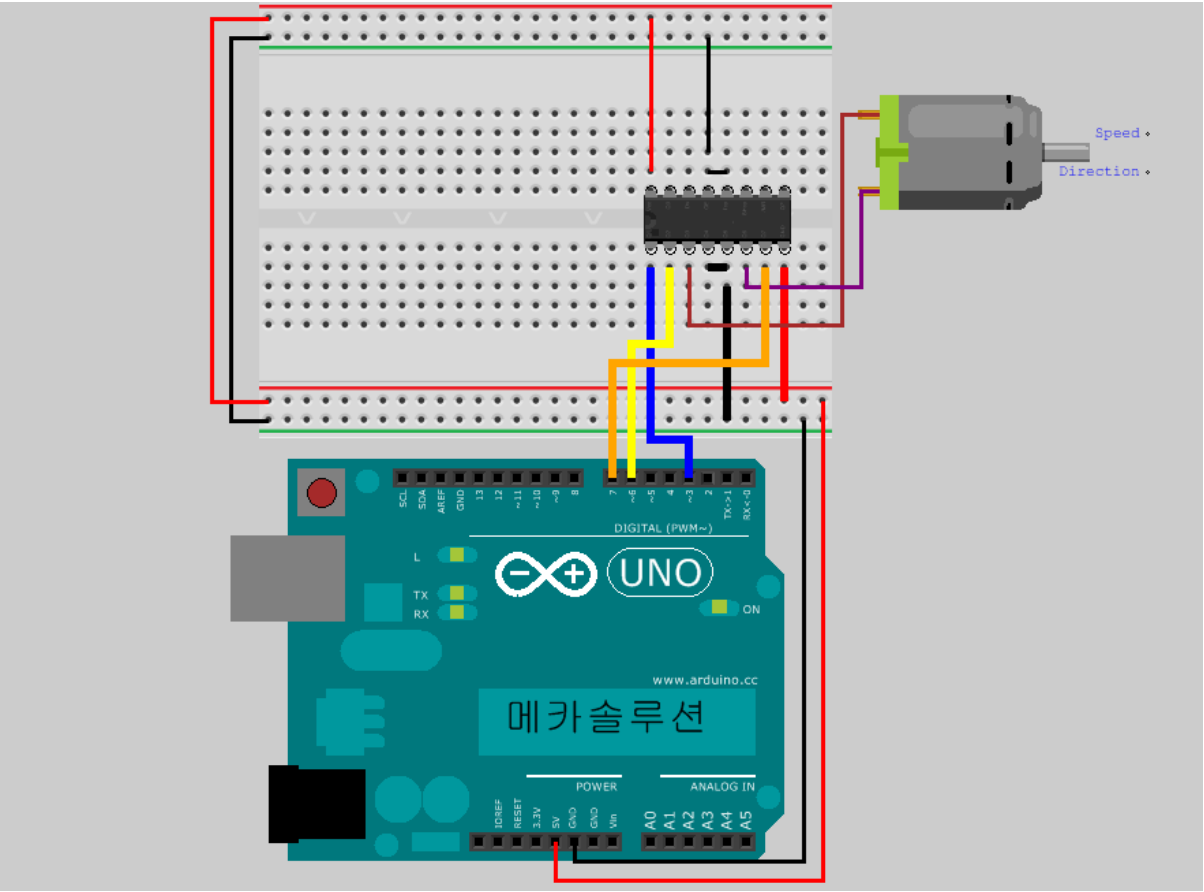


Fig. 13 SN754410 과 DC 모터의 연결

- 회로를 구성하였으면 다음의 프로그램을 컴파일 후 업로드 해보세요.

```
int speedPin = 3;           // SN754410의 1,2EN 핀과 PWM핀에 연결하였다.
속도제어용 핀
int motor1APin = 6;       // 1A
int motor2APin = 7;       // 2A
int speed_value_motor1; // 모터 스피드를 위한 변수
void setup()
{
   pinMode(speedPin, OUTPUT);
   pinMode(motor1APin, OUTPUT);
   pinMode(motor2APin, OUTPUT);
}
void loop()
{
```

```
    digitalWrite(motor1APin, LOW);    // 1A에 LOW
    digitalWrite(motor2APin, HIGH);   // 2A에 HIGH
    speed_value_motor1 = 127; // 0~255의 수 중 선택가능함. 0: 속도 제로, 255: 최고 속도.
    analogWrite(speedPin, speed_value_motor1); // PWM 출력을 하여 정한 속도만큼 모터 회전
}
```

- 위의 코드에서 loop() 내에 있는 명령문 중, digitalWrite(motor1APin, LOW)와 digitalWrite(motor2APin, HIGH)를 다음과 바꾸어보세요. ditialWrite(motor1APin, HIGH), digitalWrite(motor2APin, LOW). 모터가 반대 방향으로 도는 것을 확인할 수 있습니다.
- 또한, speed_value_motor1 = 127의 값을 255로도 바꾸어보세요. 모터가 더 빨리 회전하는 것을 확인할 수 있을 것입니다.

C. 초음파 거리센서 (HC-SR04)

준비물 : Uno R3, USB cable, 브레드보드, HC-SR04 초음파 거리센서, 점퍼선

배경 : 거리를 측정하는 것으로는 보통 적외선이나 초음파를 많이 사용합니다. 각각 센서들은 장점과 단점들이 있습니다. 그러나 이 HC-SR04는 매우 싸며 유명한 거리센서입니다. 적외선과 초음파센서는 각각 빛과 소리를 보냅니다. 신호는 물체에 닿으면 반사되며 신호를 보내고 다시 받는 사이의 시간을 측정합니다. 적외선 센서는 검은 물체에 작동하지 않으며 초음파센서는 흡수성의 물체(예로 스펀지)에 잘 작동하지 않습니다.

회로:

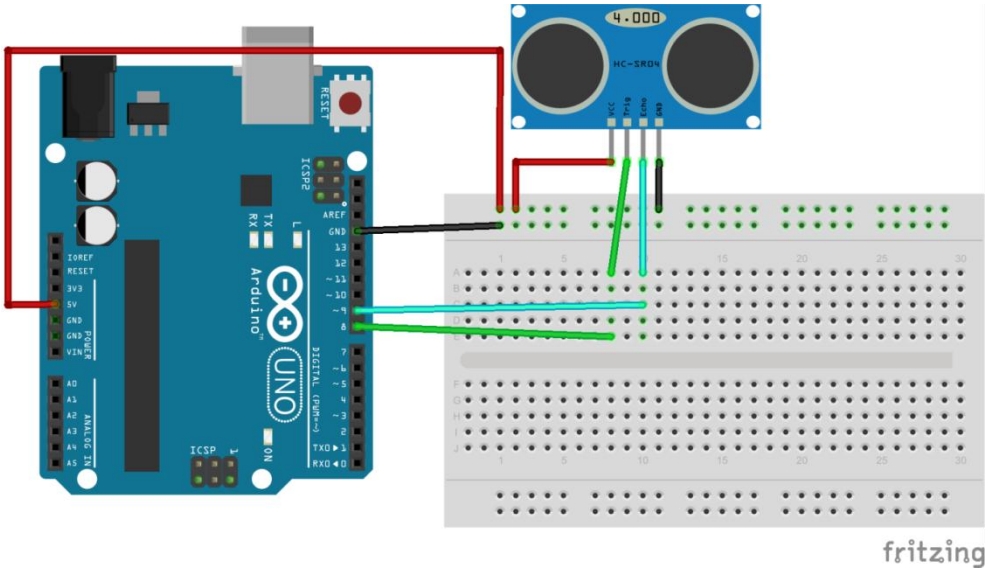


Fig. 14 HC-SR04 초음파 센서와 우노와의 연결

코드:

```
const int TriggerPin = 8;          //Trig pin
const int EchoPin = 9;             //Echo pin
long Duration = 0;

void setup(){
   pinMode(TriggerPin,OUTPUT);    // Trigger is an output pin
   pinMode(EchoPin,INPUT);         // Echo is an input pin
   Serial.begin(9600);             // Serial Output
}

void loop(){
   digitalWrite(TriggerPin, LOW);
   delayMicroseconds(2);
   digitalWrite(TriggerPin, HIGH);            // Trigger pin to HIGH
   delayMicroseconds(10);                      // 10us high
   digitalWrite(TriggerPin, LOW);             // Trigger pin to HIGH

   Duration = pulseIn(EchoPin,HIGH);          // Waits for the echo pin to get high
                                              //  returns the Duration in
microseconds
   long Distance_mm = Distance(Duration);     // Use function to calculate the distance

   Serial.print("Distance = ");               // Output to serial
   Serial.print(Distance_mm);
   Serial.println(" mm");

   delay(1000);                                // Wait to do next measurement
}
```

```
long Distance(long time)
{
      // Calculates the Distance in mm
      // ((time)*(Speed of sound))/ toward and backward of object) * 10

      long DistanceCalc;                              // Calculation variable
      DistanceCalc = ((time /2.9) / 2);         // Actual calculation in mm
      //DistanceCalc = time / 74 / 2;                 // Actual calculation in inches
      return DistanceCalc;                            // return calculated value
}
```

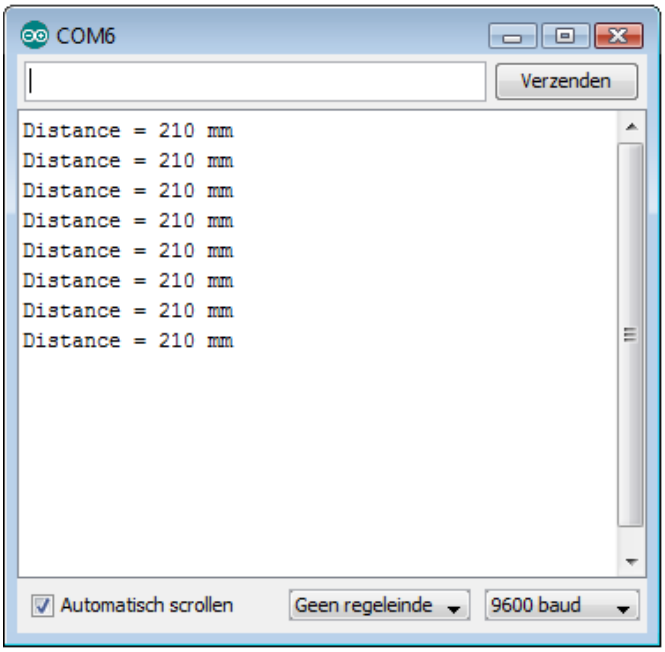


Fig. 15 결과창

D. 자기장 센서 (HMC5883L)

HMC5883L 을 이용하여 절대 방향을 측정할 수 있습니다. 자기장 센서 혹은 방위센서라고 불리는 이 센서는 자석에 의해 값이 왜곡될 수 있으니, 자석이 들어가 있는 모터 등과 가까이 사용하지 않는 것이 좋습니다.

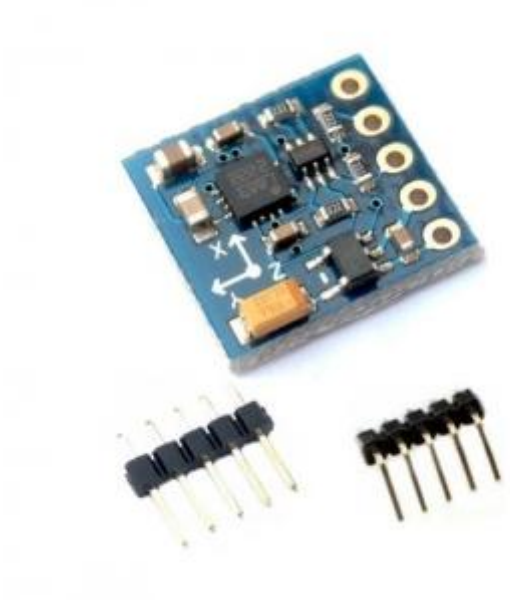
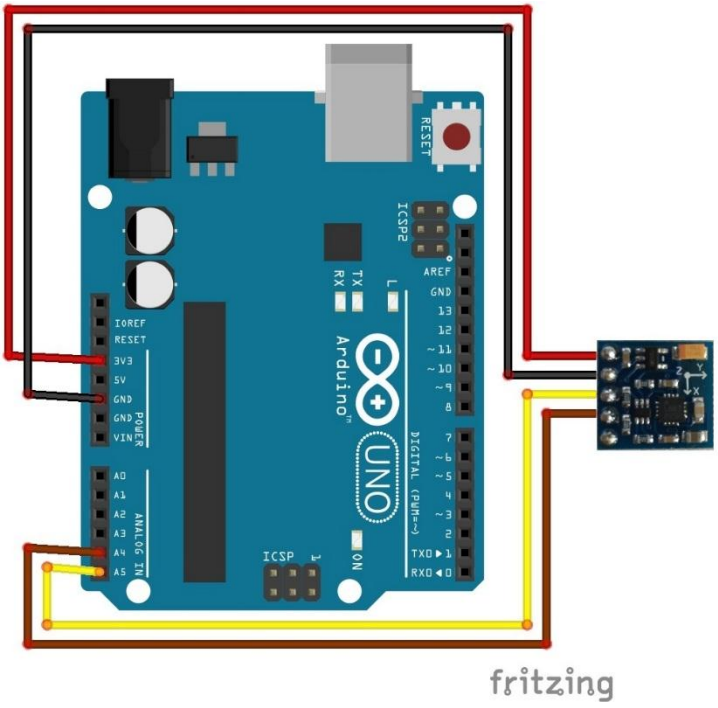


Fig. 16 HMC5883L 자기장 센서

I2C 통신을 하며 다음과 같이 연결하실 수 있습니다.



MPU6050		**Arduino**
VCC	<->	3V3
GND	<->	GND
SCL	<->	ANALOG IN A5
SDA	<->	ANALOG IN A4

참고로, SCL 과 SDA 는 보드마다 차이가 있습니다. 다음을 참고해주세요.

Board	**I2C / TWI pins**
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

```
#include <Wire.h> //I2C Arduino Library

#define address 0x1E //0011110b, I2C 7bit address of HMC5883

void setup(){
  //Initialize Serial and I2C communications
  Serial.begin(9600);
  Wire.begin();

  //Put the HMC5883 IC into the correct operating mode
  Wire.beginTransmission(address); //open communication with HMC5883
  Wire.write(0x02); //select mode register
  Wire.write(0x00); //continuous measurement mode
  Wire.endTransmission();
}

void loop(){

  int x,y,z; //triple axis data

  //Tell the HMC5883 where to begin reading data
```

```
  Wire.beginTransmission(address);
  Wire.write(0x03); //select register 3, X MSB register
  Wire.endTransmission();



  //Read data from each axis, 2 registers per axis
  Wire.requestFrom(address, 6);
  if(6<=Wire.available()){
    x = Wire.read()<<8; //X msb
    x |= Wire.read(); //X lsb
    z = Wire.read()<<8; //Z msb
    z |= Wire.read(); //Z lsb
    y = Wire.read()<<8; //Y msb
    y |= Wire.read(); //Y lsb
  }
  float rad = atan2(y, x) ;
  float degree = rad * 180 / PI ;
  //Print out values of each axis
  Serial.print("x: ");
  Serial.print(x);
  Serial.print("    y: ");
  Serial.print(y);
  Serial.print("    z: ");
  Serial.print(z);
  Serial.print("    deg :    ") ;
  Serial.println(degree) ;

  delay(250);
}
```

E. 조도센서 (Photocell)

준비물 : Uno R3, USB cable, 브레드보드, 조도센서, 10K 저항, 브레드보드 점퍼 와이어

배경 : 황화 카드뮴 셀(이하 CdS)은 저항의 일종이며 광전자를 이용한 반도체 효과를 이용하여 외부 빛의 조도에 의해 저항값이 결정됩니다. 빛이 강해지면 저항값이 약해집니다.

회로 : 밝기는 시리얼 모니터 기능에 의해 관찰될 것입니다. 아래와 같이 회로를 구성하고 코드를 업로드합니다. 코드가 업로드 되면 시리얼 모니터를 엽니다.

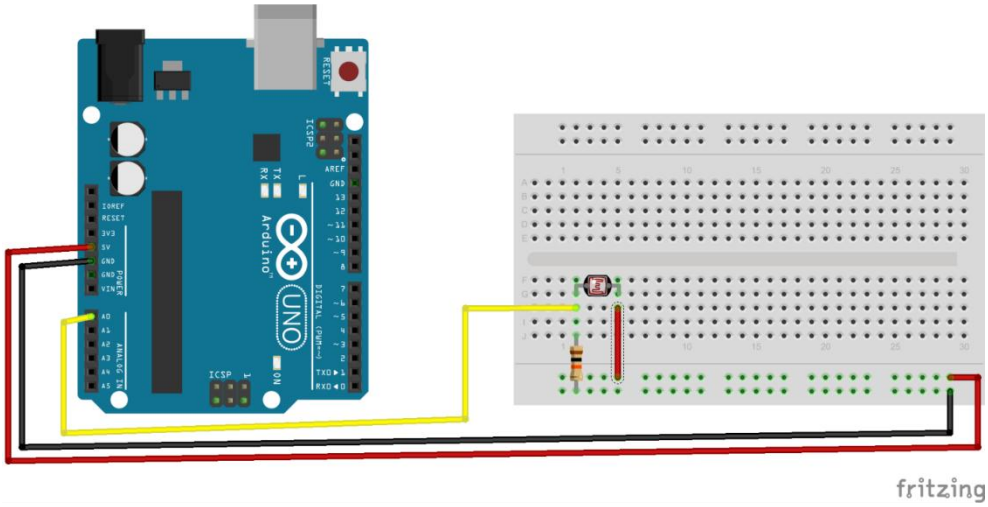


Fig 17 조도센서와 우노의 연결

코드:

```
int potpin=0;    //set analog interface #0 to connect photocell
int ledpin=13;
int val=0;    //define variable val
void setup()
{
   pinMode(ledpin,OUTPUT);//set digital interface #11 as output
   Serial.begin(9600);//set baud rate as 9600
}
void loop()
{
   val=analogRead(potpin);//read analog
   Serial.println(val);
   delay(10);//delay 0.01 s
}
```

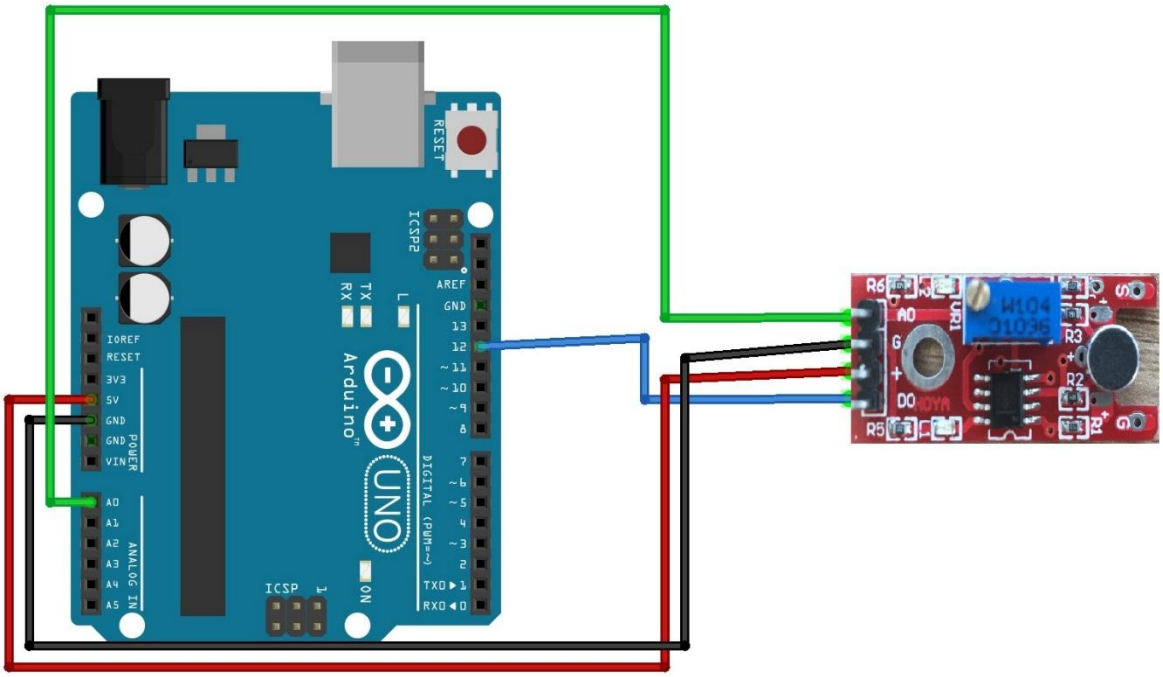
F. 사운드 센서

사운드 센서를 이용하여 소리의 세기를 인식할 수 있습니다. 아날로그와 디지털 값을 사용하여 소리의 세기 및 유무를 인식할 수 있는데, 파란색의 포텐쇼미터를 통해서 감도를 조정할 수 있습니다.

Fig. 18 사운드 센서 이미지

사운드 센서를 사용할 때는 포텐쇼미터를 이용하여 주변 환경 노이즈와 원하는 소리를 듣기 위한 감도 조정이 필요한데, 이를 위해서는 시리얼모니터링 혹은 내장된 LED를 통해서 그 변화를 알 수 있습니다. 예를 들어, 박수 소리를 인식하는지 아닌지 알기 위해서 다음처럼 연결할 수 있는데, 박수 소리가 없을 때는 LED에 불이 들어오지 않고, 박수 소리를 인식하면 센서의 LED에 빨간 불이 들어올 수 있도록 파란색 포텐쇼미터를 조정해줍니다. 혹은 아날로그값을 통해서 조정할 수도 있는데, 이 경우에는 void setup() 함수 안에 Serial.begin(9600);를 통해서 시리얼 모니터링을 할 수 있도록 활성화시킨 후, 메인 void loop() 함수 안에, int

ValA0 = analogRead(A0); Serial.println(ValA0);를 통해서 센서가 주변 노이즈와 박수 소리를 제대로 구별할 수 있는지 조정해줍니다.



센서 핀	아두이노 핀
DO	D12
+	5V
G	GND
AO	A0

소스 코드

```
const int S = 12; // the number of the pushbutton pin
const int ledPin = 13; // the number of the LED pin
int sensorstate = 0; // variable for reading the pushbutton status
void setup() {
  //Serial.begin(9600); // 시리얼 모니터링을 하기 위해서
pinMode(ledPin, OUTPUT);
  pinMode(S, INPUT);
}
```

```
void loop(){
 sensorstate = digitalRead(S);
 // int ValA0 = analogRead(A0); // 아날로그값을 변수 ValA0에 저장
 // Serial.println(ValA0); // 값을 출력. 파란색 포텐쇼미터를 조정하여서 주변 노이즈와 박수 소리를 구별할 수 있는지 확인.
 if (sensorstate == HIGH) {
 // turn LED on:
 digitalWrite(ledPin, HIGH);
 }
 else {
 // turn LED off:
 digitalWrite(ledPin, LOW);
 }
}
```

G. 블루투스 통신 (HC-06)

블루투스 모듈 중에 저렴하고 사용하기 편리한 JY-MCU 블루투스 모듈을 이용하여 데이터를 전송 받을 수 있습니다. 하드웨어 TX와 RX를 사용할 것이기에 블루투스를 연결하기 전에 다음의 프로그램을 업로드합니다.

```
char val; // variable to receive data from the serial port
int ledpin = 8; // LED connected to pin 48 (on-board LED)

void setup() {

  pinMode(ledpin, OUTPUT);  // pin 48 (on-board LED) as OUTPUT
  Serial.begin(9600);          // start serial communication at 9600bps
}

void loop() {
```

```
  if( Serial.available() )       // if data is available to read
  {
    val = Serial.read();         // read it and store it in 'val'
  }
  if( val == 'H' )               // if 'H' was received
  {
    digitalWrite(ledpin, HIGH);  // turn ON the LED
  } else {
    digitalWrite(ledpin, LOW);   // otherwise turn it OFF
  }
  delay(100);                    // wait 100ms for next reading
}
```

업로드가 완료된 후, 다음처럼 아두이노와 연결합니다.

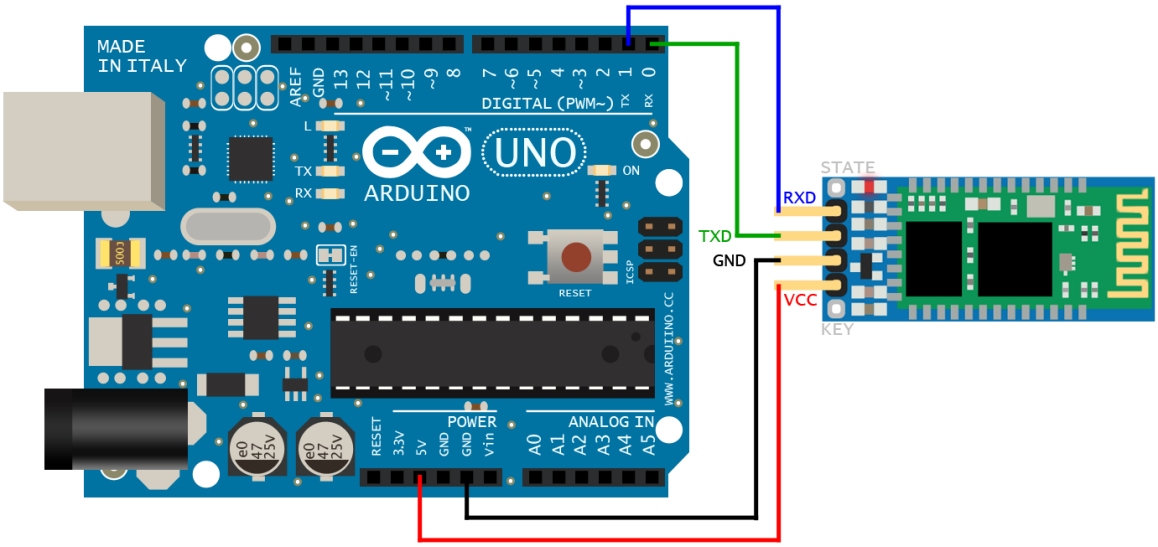


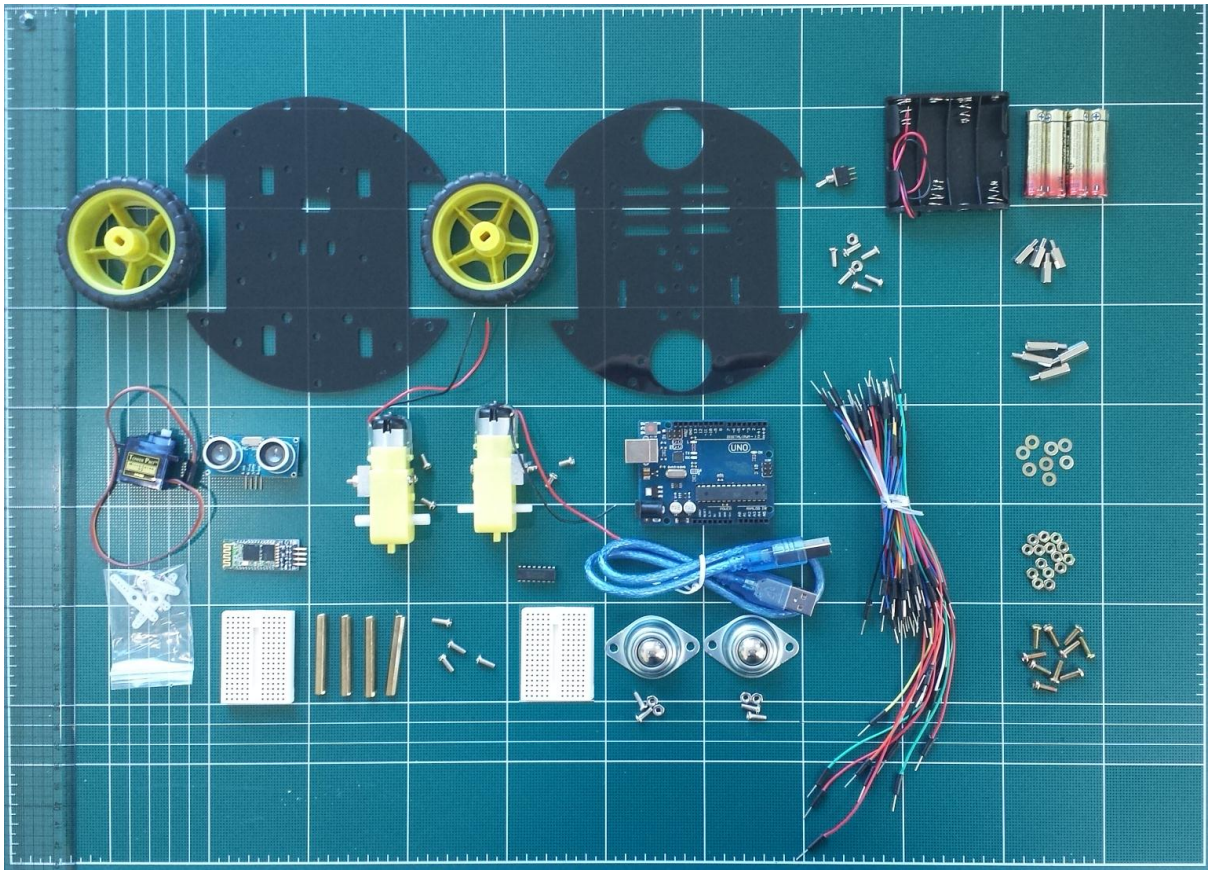
Fig. 19 블루투스와 우노의 연결

통신을 하기 위해서 컴퓨터에 블루투스 동글을 사용하거나 혹은 안드로이드 폰의 블루투스와 연결을 합니다. 장치를 추가하고 'HC-06'이라는 것을 찾고 비밀번호로 '1234'를 입력합니다.

4. MECHABOT(메카봇)의 소개 및 조립

A. MECHABOT(메카봇)의 소개

본 매뉴얼에서 소개할 모바일로봇 플랫폼은 메카봇(MECHABOT)으로 불립니다. 작은 사이즈로 두개의 바퀴를 이용하여 주행하는 로봇으로 특별한 납땜이 필요 없이 조립할 수 있습니다. 바퀴 2개에 볼 캐스터 2개를 사용하여 앞뒤 밸런스를 유지하였으며, 4개의 AAA 배터리를 통해 전원을 공급받고, 기본적으로 블루투스 모듈, 초음파 거리센서, 그리고 미니 서보 모터가 포함되어 있습니다. 메카봇의 컨트롤러는 우노 R3이며, SN754410 모터 드라이버를 사용하여 두 개의 DC 모터를 독립적으로 제어할 수 있습니다.



메카봇은 작은 사이즈의 플랫폼으로 막 로봇 공학을 배우거나, 모바일로봇 관련 연구를 하는 분에게도 좋은 플랫폼이 될 것입니다. 특히, 작은 사이즈를 통해서 Swarm Robotics 및 Cooperative Control 등 협업을 통한 로봇 시스템 설계 및 제어에 유용하게 사용될 수 있습니다. 메카봇 키트에 포함된 부품들은 다음과 같습니다.

1. **상/하부 덱:** 원형 모양의 검정색 덱은 모터, 배터리, 볼캐스터, 아두이노 등을 고정시킬 기본 섀시입니다. 직경 XXcm의 디스크 모양으로 바퀴가 들어갈 자리는 네모나게 파여 있습니다. 상부와 하부 덱은 30mm 서포터를 사용하여 고정되게 되어 있으며, 아크릴로 제작이 되었기 때문에 볼캐스터를 하부 덱과 연결할 때, 과도하게 스크류를 조여서 아크릴이 손상되지 않도록 주의하여야 합니다.

2. **두 개의 DC Geared 모터:** 두 개의 DC 모터는 노란색 하우징안에 기어와 함께 설치되어 있습니다. 기어를 통해서 큰 토크를 생성하며, 모바일로봇의 추진력을 제공합니다.

3. **고무 바퀴 2개:** 직경 XX의 두 개의 바퀴를 통해서 실내 혹은 실외에서 실습을 할 수 있습니다.

4. **볼캐스터 2개:** 볼캐스터는 로봇의 수평을 유지시키며, 이를 통해 균형을 유지하면서 이동을 하게 됩니다. 휠캐스터라고도 불립니다.

5. **초음파 거리 센서:** 초음파 거리 센서는 초음파를 이용하여 벽과의 거리 등을 감지하여 로봇이 외부 장애물을 감지하고 회피할 수 있는 동작을 할 수 있도록 도와줍니다.

6. **AAA배터리 홀더와 4개의 AAA배터리:** 기본적으로 일반 AAA 배터리를 사용하여 로봇을 구동할 수 있으며, 필요에 따라서 충전 배터리를 통해 반영구적으로 로봇 실습을 할 수도 있습니다.

7. **SN754410 모터 드라이버:** SN754410 모터 드라이버는 기존의 L293 혹은

L298 모터 드라이버에 비해서 사이즈 측면에서 작고 저렴하지만, 본 로봇 플랫폼의 구동 모터 전력 및 토크에 적합하기 때문에 사용되었습니다.

8. **SG90 미니 서보 모터:** SG90 미니 서보 모터는 아두이노로부터 간단한 PWM 출력에 따라 0~180도까지 회전운동을 하도록 하며, 사용자가 미니 서보 모터에 독특한 디자인의 로봇 암 (Robot arm), 결합 링크 등을 통해 다른 로봇 및 환경과의 물리적 결합 및 접촉을 할 수 있습니다.

9. **HC-06 블루투스 슬레이브 모듈:** 기본적으로 블루투스 모듈을 통해서 모바일 폰과의 통신을 통한 무선 제어를 가능하게 하였습니다. 블루투스 모듈이 아닌 XBee, Wifi 등과의 통신도 별도의 통신 모듈을 통해 구현할 수 있습니다.

10. **2개의 미니 브레드보드:** 2개의 미니 브레드보드는 모터 드라이버, 블루투스 모듈, 초음파 센서, SG90 미니 서보 모터 등을 연결할 수 있습니다.

11. **65핀 점퍼와이어:** 65핀 점퍼와이어는 아두이노와 브레드보드 및 SG90과의 직접적 연결을 도우며, 납땜없이 모바일 로봇 키트를 돕도록 합니다.

12. **브레드보드용 3핀 스위치:** 브레드보드용 3핀 스위치를 통해서 전원 공급을 스위칭할 수 있습니다.

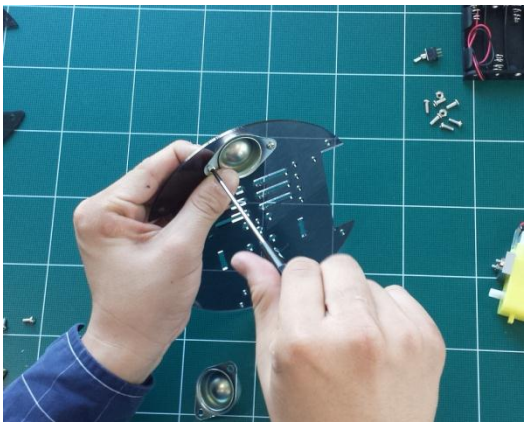
13. **우노 호환보드와 USB 케이블:** 우노 호환보드를 메인보드로 선택하여 본 튜토리얼을 진행하였으며, 이 외에도 다양한 아두이노 보드들을 사용하여 로봇을 제어하실 수 있습니다.

14. **서포터 및 볼트/너트**: 상부 및 하부 덱과의 결합 및 덱과 센서, 보드, 모터 등과의 결합을 위해 볼트/너트를 사용하고 있습니다.

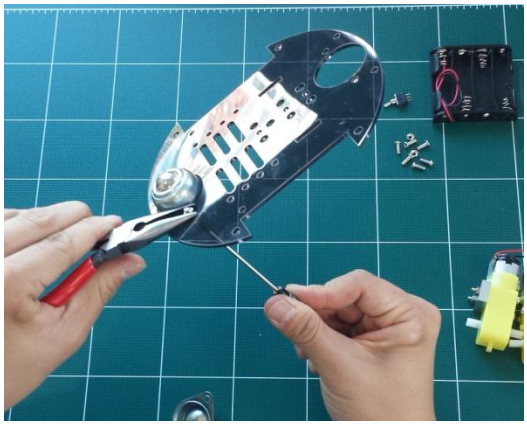
B. MECHABOT(메카봇)의 조립

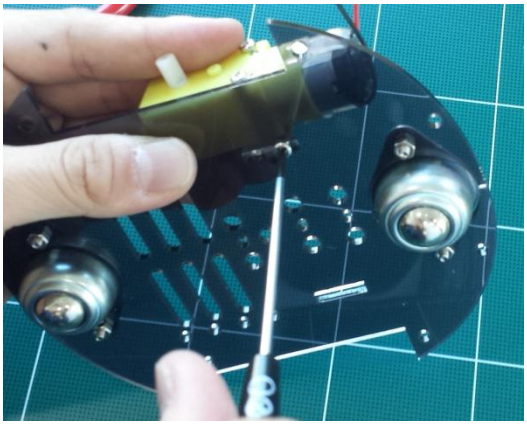
필요한 공구: 스크류 드라이버 (필수), 롱노우즈 (옵션), 스트리퍼 (옵션)

볕트와 너트의 체결을 위해서 스크류 드라이버는 필수로 필요합니다. 롱노우즈를 이용하여 볼트와 너트를 체결할 때, 너트가 헛도는 것을 막을 수 있지만, 필수적으로 필요한 것은 아닙니다. 마찬가지로 스트리퍼는 전선의 피복을 벗기기 위해서 사용되지만 일반 가위를 사용하여도 피복을 벗기실 수 있습니다.



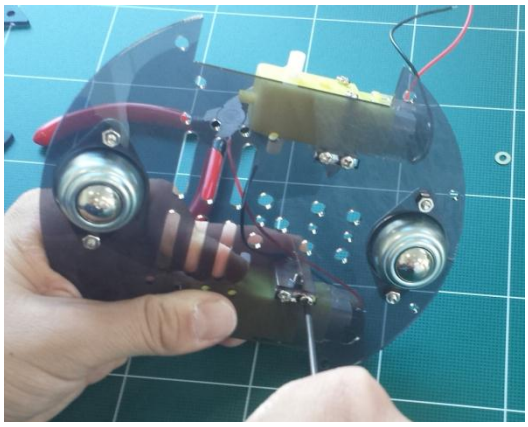
먼저, 하부 덱에 볼캐스터를 연결합니다. 볼캐스터를 하부 덱의 윗면에서 아랫면으로 집어넣고, 3mm(직경) x 6mm(길이)의 볼트를 이용하여 하부 덱에 고정시킵니다. 이 때, 볼캐스터가 제작과정에서 휘어져 있으면, 평평하게 만들어 준 후에 하부 덱에 고정시키도록 합니다. 볼캐스터가 약간 구부러져 있는 상황에서 하부덱에 볼트를 이용하여 세게 체결하게 되면 아크릴이 손상될 가능성이 있습니다.

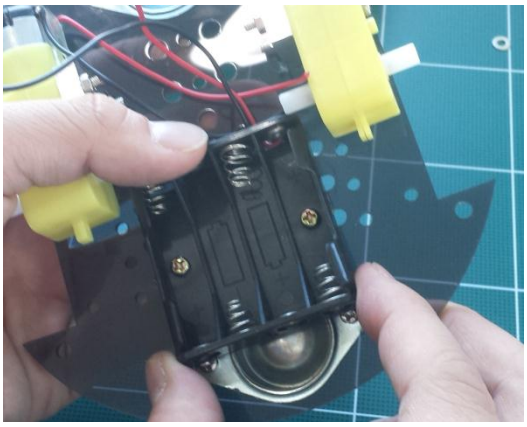




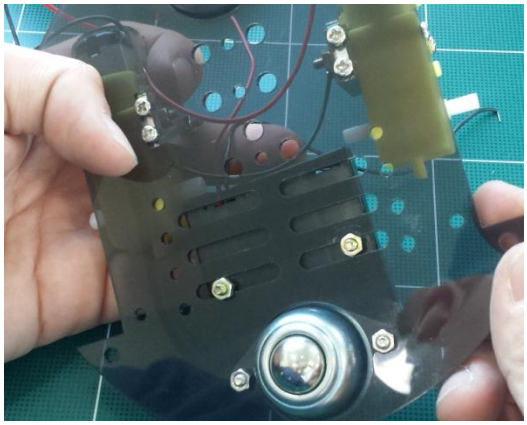
롱노우즈를 사용하여도 되고, 손으로 잡고 돌리셔도 무방합니다. 볼캐스터 두 개를 체결한 후에는 모터를 연결합니다. 위 그림과 같이 하부 덱의 윗쪽에서 아래쪽으로 배치하신 후에 사각형의 홈에 볼트 2개를 사용하여 연결합니다. 볕트를 모터의 알루미늄 프레임에 선가공된 구멍에 돌려서 체결합니다. 3mm(직경) x

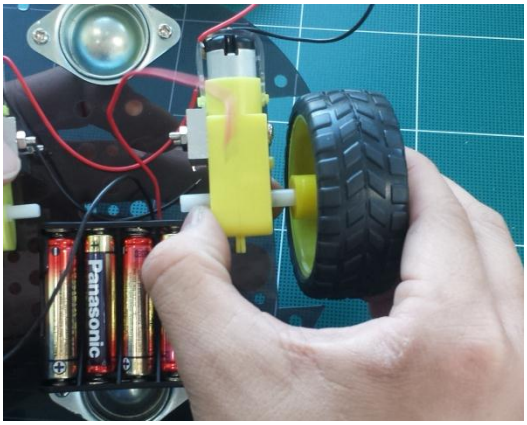
6mm (길이)의 볼트를 사용합니다.



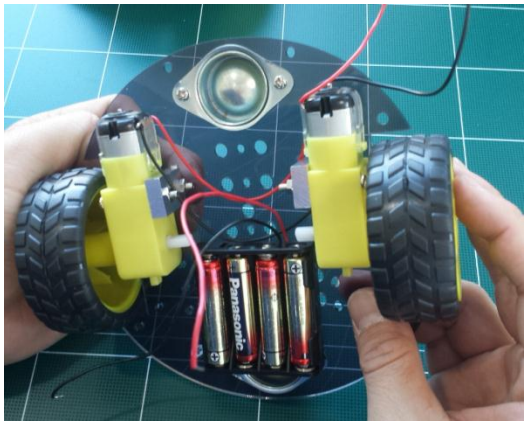


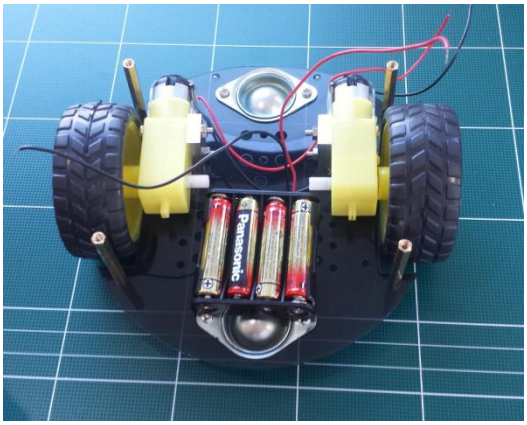
두 개의 모터를 하부 덱에 고정한 후에는 4AAA 배터리 홀더를 위 그림과 같이 연결합니다. 볼캐스터의 반쯤이 배터리 홀더에 의해 겹치게 되며 3mm x 8mm의 볼트와 3mm 너트를 사용하여 체결합니다.



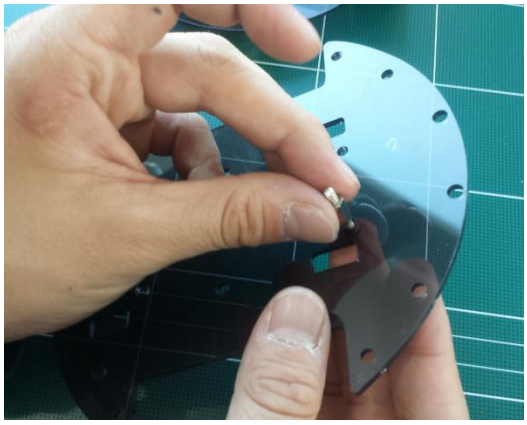


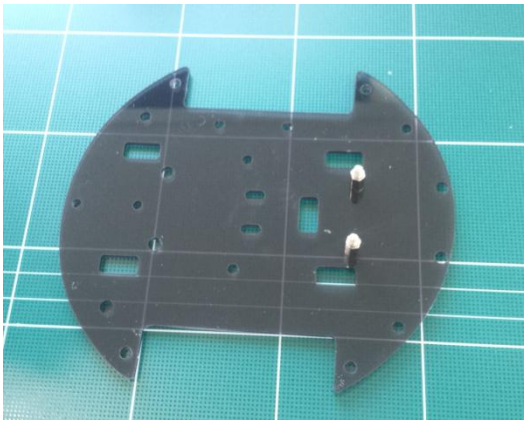
배터리 홀더를 하부 덱에 연결할 때, 위의 그림처럼 너트가 작게 느껴질 수 있습니다. 너트가 구멍에 빠져버리지 않도록 잘 연결하시길 바랍니다. 모터와 배터리 홀더를 연결하신 다음에는 바퀴를 끼웁니다.



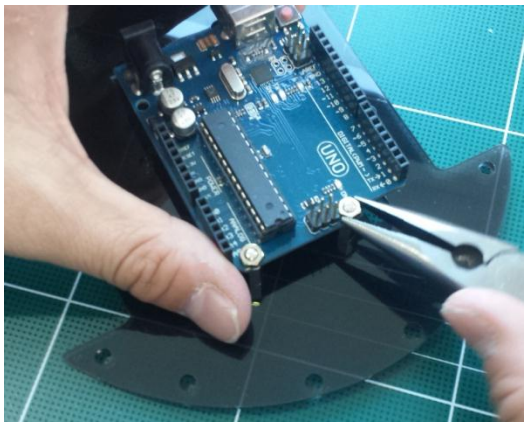


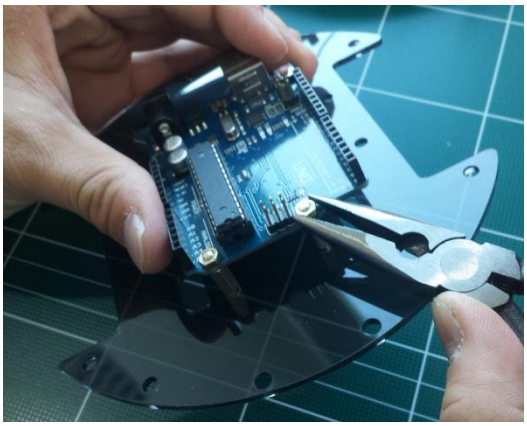
바퀴 두 개를 끼운 다음에는 XXmm 길이의 서포터를 3mm x 6mm 사이즈의 볼트를 사용하여 하부 덱에 고정시킵니다.



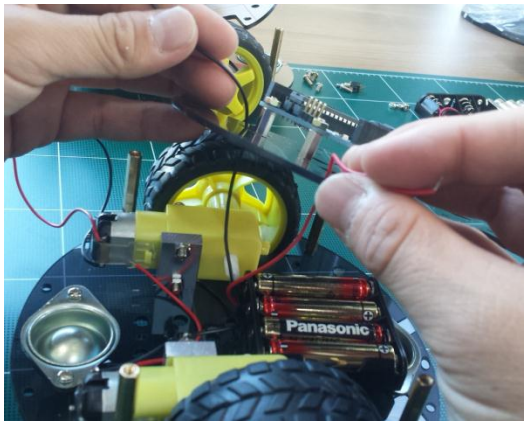


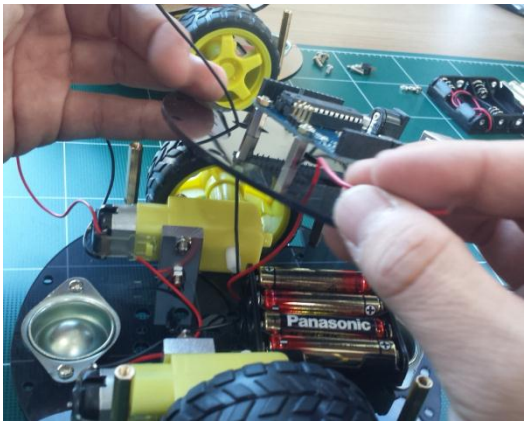
10mm 길이의 서포터를 상부 덱에 연결합니다. 이 때, 3mm x 6mm 길이의 볼트를 사용하여 서보터를 상부 덱에 고정시킵니다.



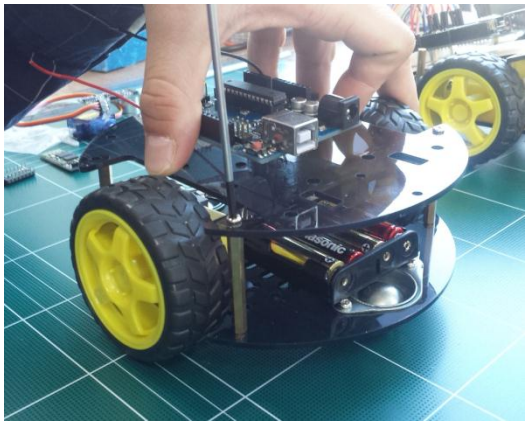


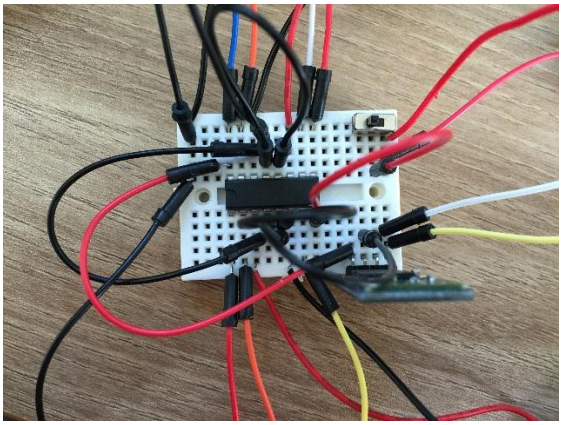
고정된 서포터에 우노 호환보드 너트를 사용하여 고정시킵니다.



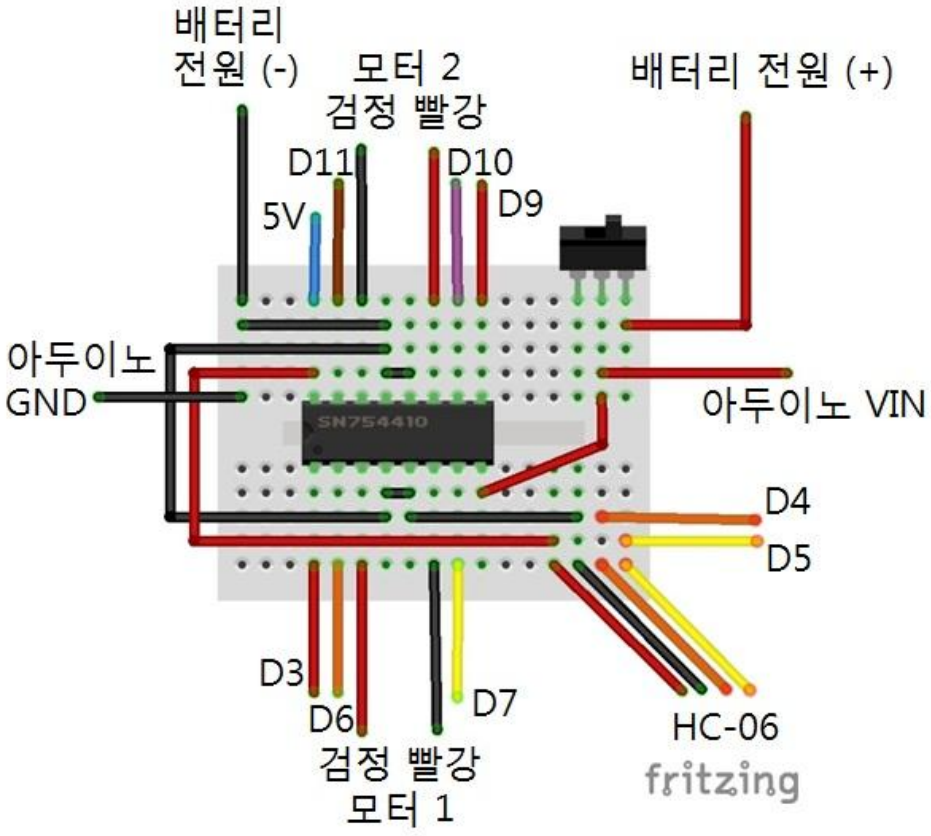


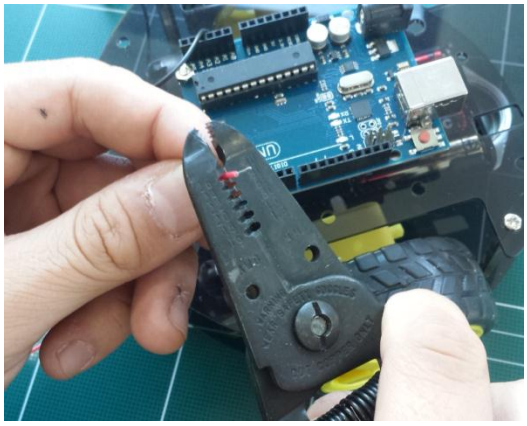
배터리 홀더로부터 나온 두 개의 전선 (빨간색과 검정색)을 상부 덱의 네모난 홈을 통해 미리 빼내어 줍니다.

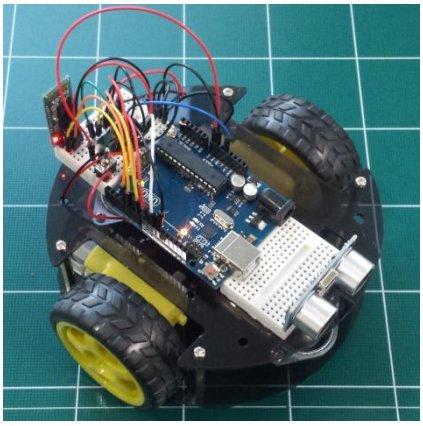




그리고 상부 덱과 하부 덱을 위와 같이 체결합니다. 서포터에 3mm x 6mm 볼트를 사용하여 체결할 수 있습니다. 그런 다음 미니 브레드보드 위에 SN754410 모터 드라이버 칩과 블루투스 슬레이브 모듈, 그리고 브레드보드용 3핀 스위치를 연결합니다. 아두이노와 점퍼와이어로 연결하는 법에 대해서는 다음의 회로도를 참고해주세요.







모터로부터 나온 두 개의 전선이 짧아서 브레드보드에 잘 연결되지 않는다면, 피복을 조금 벗겨내어 사용하실 수도 있습니다.

C. MECHABOT(메카봇)의 기본 구동 소스

메카봇의 기본 구동 소스는 블루투스로부터 값을 얻게 되면 조종되는 리모트 RC 카입니다. 소문자 a를 입력하면 왼쪽으로 회전하고, d를 입력하면 오른쪽, w를 입

력하면 전진, x를 입력하면 후진, 그리고 s를 입력하면 정지하는 동작입니다. 소프트웨어 시리얼을 통해서 블루투스가 아두이노에 연결되어 있는 상황에서도 프로그래밍을 할 수 있도록 하였습니다.

```
#include <SoftwareSerial.h> // 소프트웨어 시리얼
SoftwareSerial mySerial(4, 5); // RX, TX
int speedPinA= 3;
int speedPinB = 9;
int dir1PinA = 6;
int dir2PinA = 7;
int dir1PinB = 10;
int dir2PinB = 11;

byte incomingByte;

int speed_value_motorA;
int speed_value_motorB;
void setup() {
   mySerial.begin(9600);
  // set digital i/o pins as outputs:
  pinMode(speedPinA, OUTPUT);
  pinMode(speedPinB, OUTPUT);
  pinMode(dir1PinA, OUTPUT);
  pinMode(dir2PinA, OUTPUT);
  pinMode(dir1PinB, OUTPUT);
  pinMode(dir2PinB, OUTPUT);

  //Initial status
  digitalWrite(dir1PinA, LOW);
  digitalWrite(dir2PinA, LOW);
  digitalWrite(dir1PinB, LOW);
  digitalWrite(dir2PinB, LOW);
}
```

```
void loop() {
  // control the speed 0- 255
  speed_value_motorA = 255; // half speed
  speed_value_motorB = 255; // half speed
  analogWrite(speedPinA, speed_value_motorA);
  analogWrite(speedPinB, speed_value_motorB);



  if (mySerial.available() > 0) {
      incomingByte = mySerial.read();

      if (incomingByte == 'a') //left
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
      }
      if (incomingByte == 'd') //right
      {
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, LOW);
      }
      if (incomingByte == 's') //Stop
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, LOW);
```

```
      digitalWrite(dir2PinB, LOW);
      }
      if (incomingByte == 'w') //Forward
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, HIGH);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, HIGH);
      }
        if (incomingByte == 'x') //Back
      {
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
   }
  }
}
```

D. SoftwareSerial 사용하여 통신 및 업로드 충돌 피하기

이미 하드웨어적으로 사용할 수 있는 시리얼포트 0번과 1번 핀 외에 다른 디지털 핀을 이용한 시리얼통신을 할 경우, 소프트웨어 시리얼 (Software Serial)을 사용할 수 있습니다. 특히, 블루투스 및 XBee 등이 내장된 시리얼 포트에 물리적으로 연결되어 있는 상태라면, 아두이노에 업로드할 경우, 프로그래밍 업로드를 할 때 쓰는 통신과 사용되고 있는 시리얼 포트의 충돌로 인해서 에러를 발생하기 때문에 소프트웨어 시리얼통신 방법을 통해 충돌을 피할 수 있습니다.

사용방법은 간단합니다. 아두이노의 라이브러리를 사용하면 되는데, 이미 아두이노 소프트웨어에 내장되어 있는 라이브러리이기 때문에 별도로 다운로드 받으실 필요는 없습니다.

기본 구동 코드에서는 디지털 포트 4번과 5번을 블루투스 통신용 시리얼 포트로 설정하였는데, 그 방법은 다음과 같습니다.

1. 라이브러리 설정

```
#include <SoftwareSerial.h> // 소프트웨어 시리얼
```

2. 사용할 핀 결정

```
SoftwareSerial mySerial(4, 5); // RX, TX
```

3. setup에서 시리얼포트 활성화

```
mySerial.begin(9600);
```

4. 소프트웨어시리얼 통신 사용

```
if (mySerial.available() > 0) {

    incomingByte = mySerial.read();
```

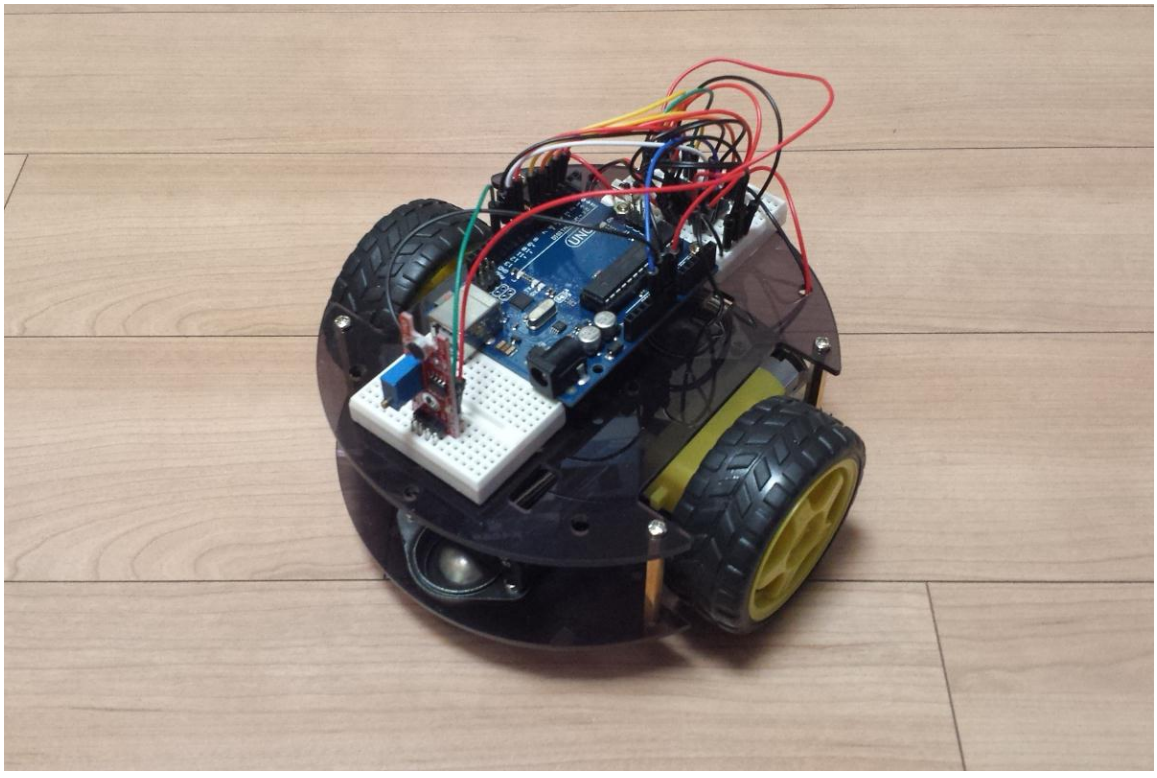
구동코드에서는 소프트웨어 시리얼을 통해서 값을 읽기만 하였는데, 출력을 하기 위해서는 mySerial.print 혹은 mySerial.write 등을 사용할 수도 있습니다.

5. MECHABOT(메카봇) 프로젝트

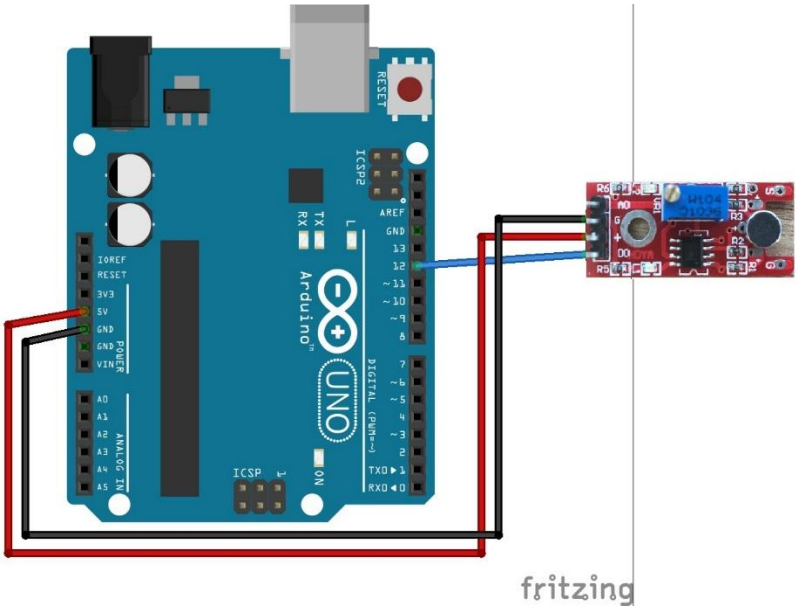
여기서부터 나오는 회로의 5V와 GND핀이 위쪽 브레드보드와 연결되어 있으므로 그쪽에서 병렬로 연결하여 사용하시면 됩니다.

A. 박수 칠 때 떠나라

박수 칠 때 떠나라 프로젝트는 박수를 칠 때마다 로봇이 움직이는 데모입니다. 관건은 사람마다 박수 소리가 다르고, 주변 환경 소음이 다르기 때문에 사운드 센서에 내장된 파란색 포텐쇼미터 값을 조정하여 사용자의 박수 소리 및 주변 환경 소음을 구별할 수 있도록 하는 것입니다. 또한, 코드 내의 delay(1000)를 변경함으로써, 박수 소리를 인식했을 때 이동하는 거리를 변경할 수 있습니다. 사운드 센서의 디지털 출력 핀을 아두이노의 D12번에 연결하였습니다. 배터리가 아닌 USB로 전원을 공급하였을 경우와 배터리를 사용할 때의 센서에 대한 반응이 다르기 때문에 USB를 이용한 프로그래밍을 마치고, 배터리로 동작시킨 다음에 포텐쇼미터 튜닝 작업을 하시기를 추천드립니다.



부가 회로도



소스코드

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); // RX, TX
int speedPinA= 3;
int speedPinB = 9;
int dir1PinA = 6;
int dir2PinA = 7;
int dir1PinB = 10;
int dir2PinB = 11;

byte incomingByte;

int speed_value_motorA;
int speed_value_motorB;

// Sound Sensor
const int S = 12; // the number of the pushbutton pin
int sensorstate = 0; // variable for reading the pushbutton status
void setup() {
   mySerial.begin(9600);
```

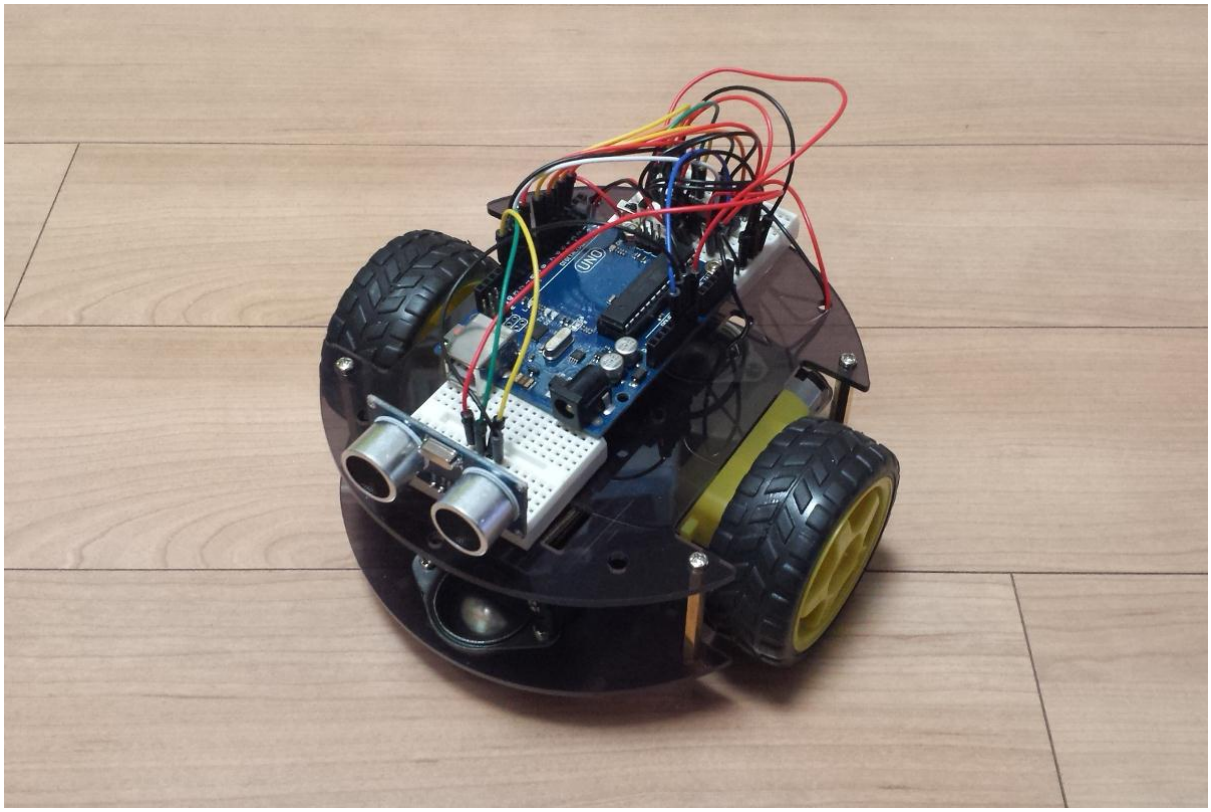
```
  Serial.begin(9600);
 // set digital i/o pins as outputs:
 pinMode(speedPinA, OUTPUT);
 pinMode(speedPinB, OUTPUT);
 pinMode(dir1PinA, OUTPUT);
 pinMode(dir2PinA, OUTPUT);
 pinMode(dir1PinB, OUTPUT);
 pinMode(dir2PinB, OUTPUT);

 //Initial status
 digitalWrite(dir1PinA, LOW);
 digitalWrite(dir2PinA, LOW);
 digitalWrite(dir1PinB, LOW);
 digitalWrite(dir2PinB, LOW);

 pinMode(S, INPUT);
}

void loop() {
 // control the speed 0- 255
 speed_value_motorA = 255; // half speed
 speed_value_motorB = 255; // half speed
 analogWrite(speedPinA, speed_value_motorA);
 analogWrite(speedPinB, speed_value_motorB);

 sensorstate = digitalRead(S);
 Serial.println(sensorstate);
 if (sensorstate == HIGH)
 {
     digitalWrite(dir1PinA, LOW);
     digitalWrite(dir2PinA, HIGH);
     digitalWrite(dir1PinB, LOW);
```

```
    digitalWrite(dir2PinB, HIGH);
    delay(1000);
}
else
{
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
}

if (mySerial.available() > 0) {
    incomingByte = mySerial.read();

    if (incomingByte == 'a') //left
    {
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, HIGH);
    digitalWrite(dir2PinB, LOW);
    }
    if (incomingByte == 'd') //right
    {
    digitalWrite(dir1PinA, HIGH);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
    }
    if (incomingByte == 's') //Stop
    {
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
```

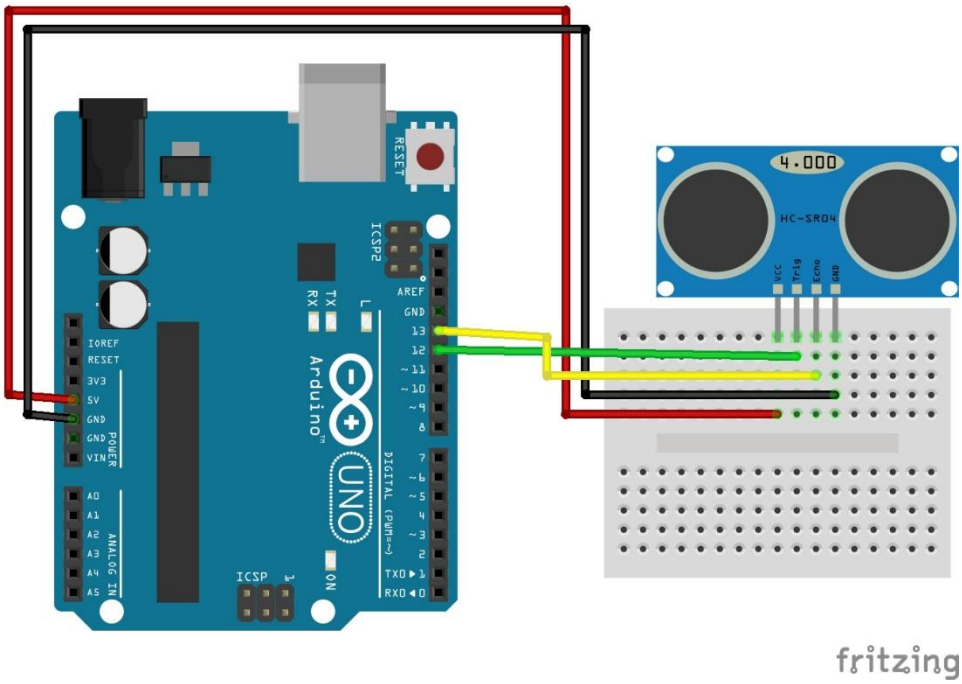
```
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, LOW);
      }
      if (incomingByte == 'w') //Forward
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, HIGH);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, HIGH);
      }
        if (incomingByte == 'x') //Back
      {
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
   }
  }
}
```

B. 초음파 거리 센서로 충돌 방지

초음파 거리 센서로 충돌 방지 데모는 모바일 로봇을 배울 때, 가장 많이 접하는 데모 중 하나로써, 가장 많이 사용하는 HC-SR04 초음파 모듈을 사용하였습니다. Trigger 핀은 아두이노의 D12에 Echo 핀은 아두이노의 D13번에 연결하였으며, 소스의 내부 코드를 변경하여 보다 복잡한 미로에서 충돌하지 않고 이동하는 로봇을 만들 수 있습니다.



부가 회로도


4.000
HC-SR04
Vcc
Trig
Echo
Gnd
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ON
ICSP
Arduino
UNO
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing

소스코드

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); // RX, TX
int speedPinA= 3;
int speedPinB = 9;
int dir1PinA = 6;
int dir2PinA = 7;
int dir1PinB = 10;
int dir2PinB = 11;

byte incomingByte;

int speed_value_motorA;
int speed_value_motorB;

// HC-SR04
const int TriggerPin = 12; //Trig pin
const int EchoPin = 13; //Echo pin
long Duration = 0;



void setup() {
   mySerial.begin(9600);
  // set digital i/o pins as outputs:
  pinMode(speedPinA, OUTPUT);
  pinMode(speedPinB, OUTPUT);
  pinMode(dir1PinA, OUTPUT);
  pinMode(dir2PinA, OUTPUT);
  pinMode(dir1PinB, OUTPUT);
  pinMode(dir2PinB, OUTPUT);

  //Initial status
```

```
  digitalWrite(dir1PinA, LOW);
  digitalWrite(dir2PinA, LOW);
  digitalWrite(dir1PinB, LOW);
  digitalWrite(dir2PinB, LOW);

  // HC-SR04
  pinMode(TriggerPin,OUTPUT); // Trigger is an output pin
  pinMode(EchoPin,INPUT); // Echo is an input pin
}

void loop() {
  // control the speed 0- 255
  speed_value_motorA = 255; // half speed
  speed_value_motorB = 255; // half speed
  analogWrite(speedPinA, speed_value_motorA);
  analogWrite(speedPinB, speed_value_motorB);

  digitalWrite(TriggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TriggerPin, HIGH); // Trigger pin to HIGH
  delayMicroseconds(10); // 10us high
  digitalWrite(TriggerPin, LOW); // Trigger pin to HIGH
  Duration = pulseIn(EchoPin,HIGH); // Waits for the echo pin to get high
  // returns the Duration in microseconds
  long Distance_mm = Distance(Duration); // Use function to calculate the
distance

  if (Distance_mm < 100) // 거리가 10cm 이내 일 때
  {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, LOW);
```

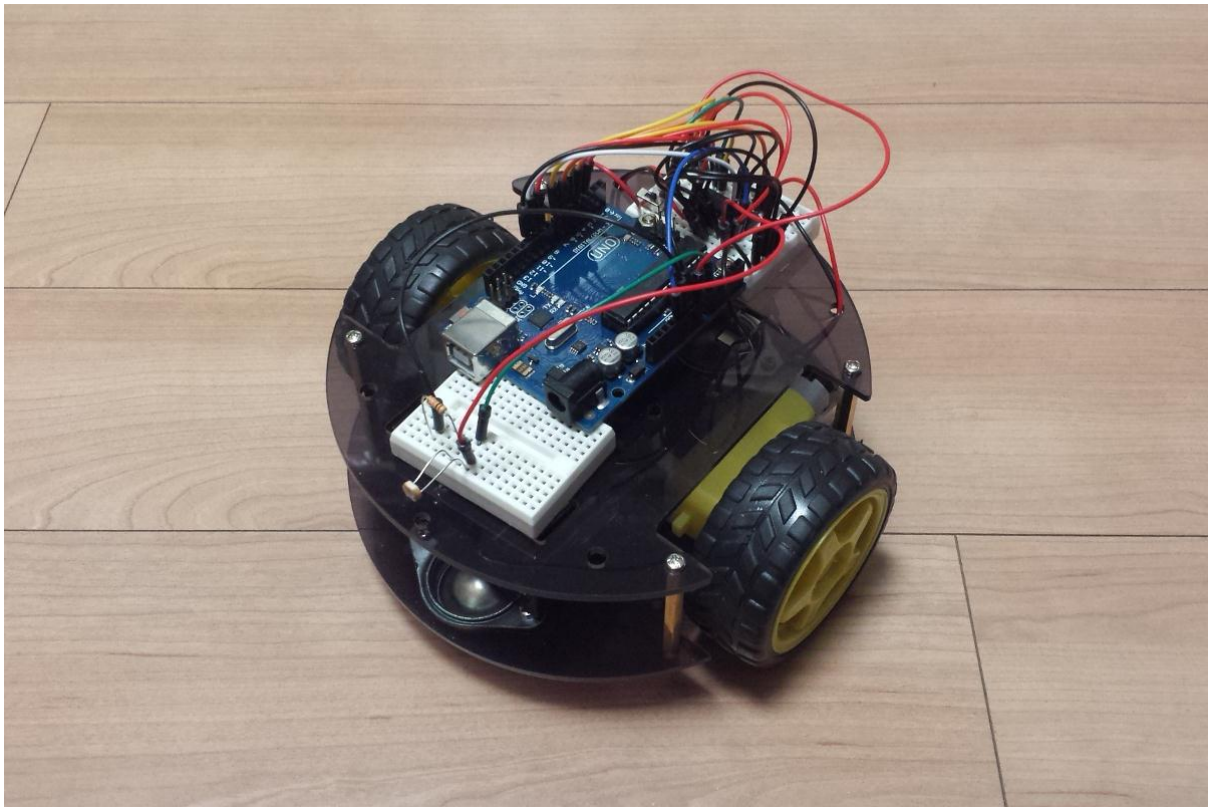
```
    digitalWrite(dir2PinB, LOW);
    delay(200); // 200ms 동안 정지하였다가
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, HIGH);
    digitalWrite(dir2PinB, LOW);
    delay(300); // 300ms 동안 왼쪽으로 회전합니다.
}
else // 그렇지 않을 경우에는 전진합니다.
{
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, HIGH);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, HIGH);
}

if (mySerial.available() > 0) {
    incomingByte = mySerial.read();

    if (incomingByte == 'a') //left
    {
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, HIGH);
    digitalWrite(dir2PinB, LOW);
    }
    if (incomingByte == 'd') //right
    {
    digitalWrite(dir1PinA, HIGH);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
```

```
      }
      if (incomingByte == 's') //Stop
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, LOW);
      }
      if (incomingByte == 'w') //Forward
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, HIGH);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, HIGH);
      }
        if (incomingByte == 'x') //Back
      {
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
   }
  }
}

long Distance(long time)
{
  // Calculates the Distance in mm
  // ((time)*(Speed of sound))/ toward and backward of object) * 10

  long DistanceCalc; // Calculation variable
  DistanceCalc = ((time /2.9) / 2); // Actual calculation in mm
```

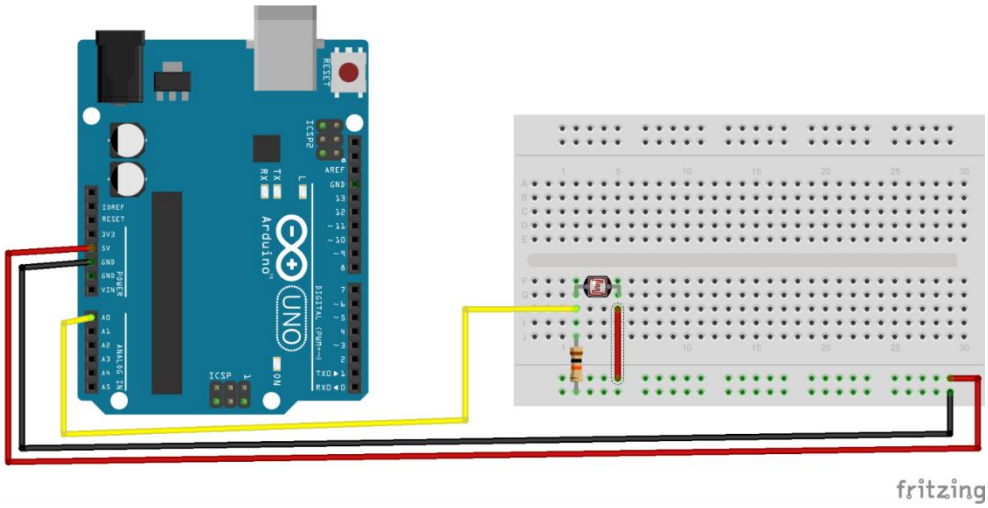
```
  //DistanceCalc = time / 74 / 2; // Actual calculation in inches
  return DistanceCalc; // return calculated value
}
```

C. 플래시로 조종하는 메카봇

플래시로 조종하는 메카봇 데모는 카메라의 LED 플래시가 로봇의 광센서를 비출 때 로봇이 따라오는 데모입니다. 광센서를 아두이노의 A0 (아날로그 0번 핀)에 연결한 후에 카메라의 플래시를 사용하여 데모를 즐겨보세요. 밝기를 1000으로 셋팅 하였지만, 이는 카메라의 플래시 혹은 주변 밝기에 따라서 조정하실 수 있습니다. 예를 들어 값을 100으로 한다면 주변의 밝기에 의해서 마냥 움직이는 로봇이 될 것이고, 너무 큰 값이라면 움직이지 않겠죠?



부가 회로도



소스코드

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); // RX, TX

int speedPinA= 3;
int speedPinB = 9;
int dir1PinA = 6;
int dir2PinA = 7;
int dir1PinB = 10;
int dir2PinB = 11;

byte incomingByte;

int speed_value_motorA;
int speed_value_motorB;



void setup() {
   mySerial.begin(9600);
```

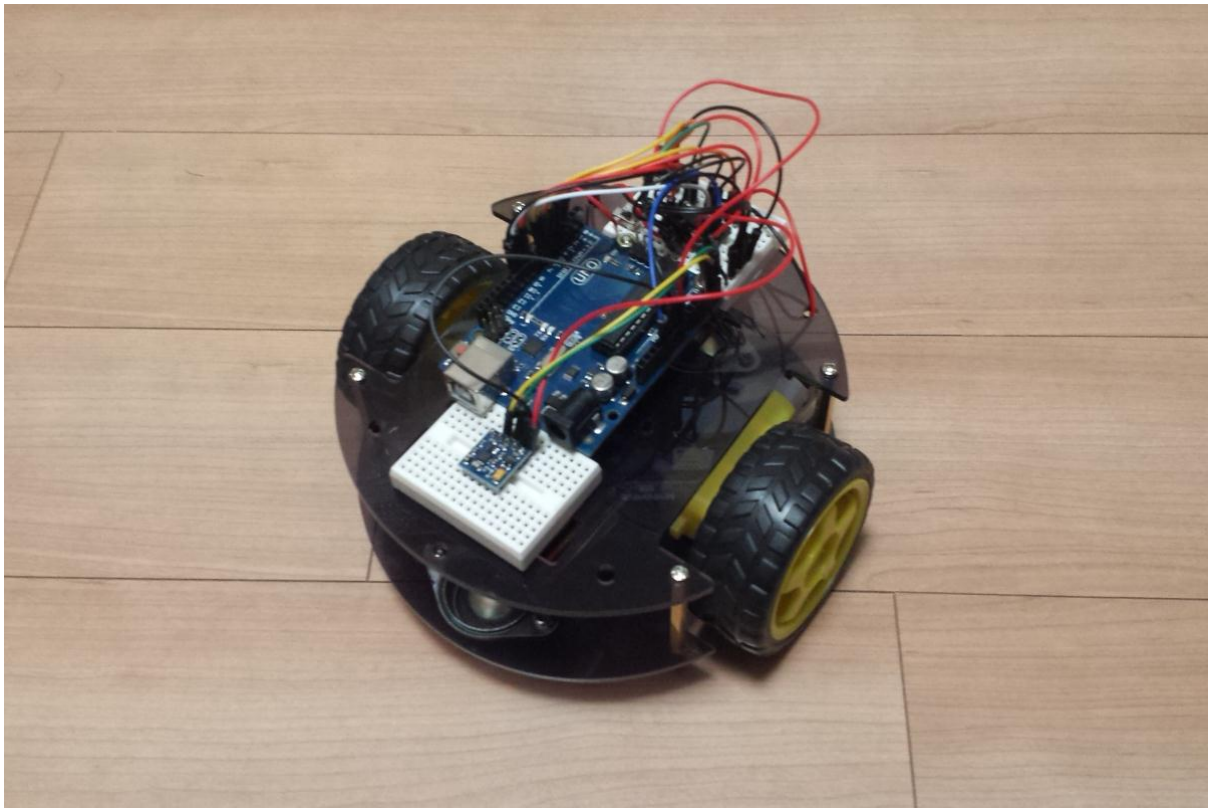
```
  Serial.begin(9600);
 // set digital i/o pins as outputs:
 pinMode(speedPinA, OUTPUT);
 pinMode(speedPinB, OUTPUT);
 pinMode(dir1PinA, OUTPUT);
 pinMode(dir2PinA, OUTPUT);
 pinMode(dir1PinB, OUTPUT);
 pinMode(dir2PinB, OUTPUT);

 //Initial status
 digitalWrite(dir1PinA, LOW);
 digitalWrite(dir2PinA, LOW);
 digitalWrite(dir1PinB, LOW);
 digitalWrite(dir2PinB, LOW);
}

void loop() {
 // control the speed 0- 255
 speed_value_motorA = 255; // half speed
 speed_value_motorB = 255; // half speed
 analogWrite(speedPinA, speed_value_motorA);
 analogWrite(speedPinB, speed_value_motorB);

 int brightness = analogRead(A0);
 if(brightness > 1000) // 밝기를 1000으로 셋업하였지만 이는 조정하실 수 있습
니다.
 {
     digitalWrite(dir1PinA, LOW);
     digitalWrite(dir2PinA, HIGH);
     digitalWrite(dir1PinB, LOW);
     digitalWrite(dir2PinB, HIGH);
     delay(1000);
```

```
}
else
{
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
}

if (mySerial.available() > 0) {
    incomingByte = mySerial.read();

    if (incomingByte == 'a') //left
    {
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, HIGH);
    digitalWrite(dir2PinB, LOW);
    }
    if (incomingByte == 'd') //right
    {
    digitalWrite(dir1PinA, HIGH);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
    }
    if (incomingByte == 's') //Stop
    {
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
```

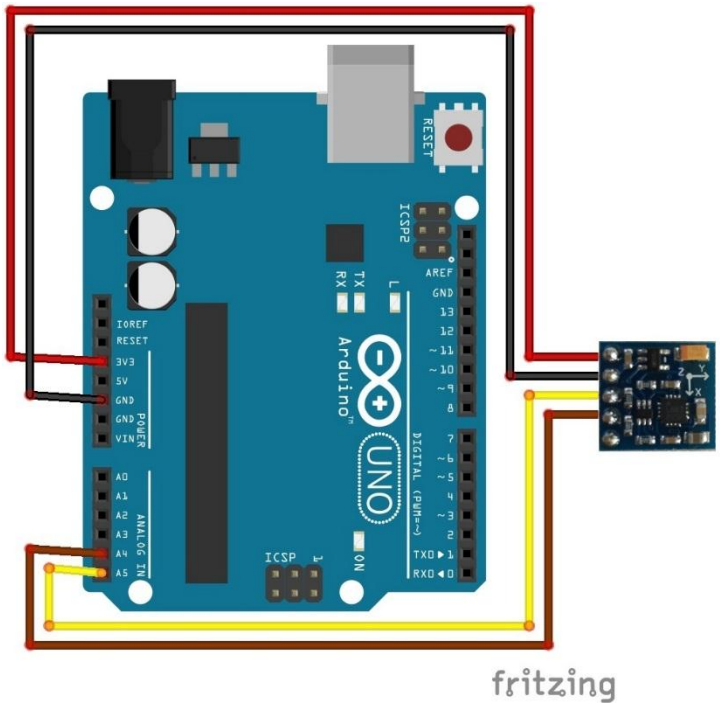
```
      }
      if (incomingByte == 'w') //Forward
      {
      digitalWrite(dir1PinA, LOW);
      digitalWrite(dir2PinA, HIGH);
      digitalWrite(dir1PinB, LOW);
      digitalWrite(dir2PinB, HIGH);
      }
        if (incomingByte == 'x') //Back
      {
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
   }
  }
}
```

D. 자기장 센서로 절대 방향 검출

자기장 센서로 절대 방향 검출하는 모바일 로봇 데모는 HMC5883L이라는 3축 자기장 센서를 사용하여 로봇의 절대적인 방향을 인식하고 이를 이용하여 회전하는 로봇을 보여줍니다. I2C 통신을 하기 때문에 우노와 사용할 때는 SDA 및 SCL핀을 우노의 A4와 A5에 연결하여 사용하실 수 있습니다. Wire라는 내장된 라이브러리를 불러서 I2C 통신을 할 수 있으며 출력되는 방위값을 시리얼모니터링 기능을 통해서 확인해보실 수 있습니다. Serial.println(heading*180/3.14);



부가 회로도


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX
RX
L
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
Arduino
UNO
DIGITAL (PWM=~)
TXD 1
RXD 0
ICSP
ON
fritzing

소스코드

```
#include <SoftwareSerial.h>
#include <Wire.h> //I2C Arduino Library
SoftwareSerial mySerial(4, 5); // RX, TX

int speedPinA= 3;
int speedPinB = 9;
int dir1PinA = 6;
int dir2PinA = 7;
int dir1PinB = 10;
int dir2PinB = 11;

byte incomingByte;

int speed_value_motorA;
int speed_value_motorB;

#define address 0x1E //0011110b, I2C 7bit address of HMC5883
double heading = 0;

void setup() {
   mySerial.begin(9600);
   Serial.begin(9600);
  // set digital i/o pins as outputs:
  pinMode(speedPinA, OUTPUT);
  pinMode(speedPinB, OUTPUT);
  pinMode(dir1PinA, OUTPUT);
  pinMode(dir2PinA, OUTPUT);
  pinMode(dir1PinB, OUTPUT);
  pinMode(dir2PinB, OUTPUT);

  //Initial status
```

```
  digitalWrite(dir1PinA, LOW);
  digitalWrite(dir2PinA, LOW);
  digitalWrite(dir1PinB, LOW);
  digitalWrite(dir2PinB, LOW);

    Wire.begin();

    //Put the HMC5883 IC into the correct operating mode
    Wire.beginTransmission(address); //open communication with HMC5883
    Wire.write(0x02); //select mode register
    Wire.write(0x00); //continuous measurement mode
    Wire.endTransmission();

}

void loop() {
  // control the speed 0- 255
  speed_value_motorA = 255; // half speed
  speed_value_motorB = 255; // half speed
  analogWrite(speedPinA, speed_value_motorA);
  analogWrite(speedPinB, speed_value_motorB);

    int x,y,z; //triple axis data

    //Tell the HMC5883L where to begin reading data
    Wire.beginTransmission(address);
    Wire.write(0x03); //select register 3, X MSB register
    Wire.endTransmission();
     //Read data from each axis, 2 registers per axis
    Wire.requestFrom(address, 6);
    if(6<=Wire.available()){
       x = Wire.read()<<8; //X msb
```

```
    x |= Wire.read(); //X lsb
    z = Wire.read()<<8; //Z msb
    z |= Wire.read(); //Z lsb
    y = Wire.read()<<8; //Y msb
    y |= Wire.read(); //Y lsb

    heading = atan2(y, x);
  }
  Serial.println(heading*180/3.14);
if (heading*180/3.14 > -140 && heading*180/3.14 < -30)
{
   digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, HIGH);
    digitalWrite(dir2PinB, LOW);
}
else if (heading*180/3.14 > 30 && heading*180/3.14 < 140)
{
  digitalWrite(dir1PinA, HIGH);
    digitalWrite(dir2PinA, LOW);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, LOW);
}
else if (180-abs(heading*180/3.14) < 30)
{
    digitalWrite(dir1PinA, LOW);
    digitalWrite(dir2PinA, HIGH);
    digitalWrite(dir1PinB, LOW);
    digitalWrite(dir2PinB, HIGH);
}

if (mySerial.available() > 0) {
```

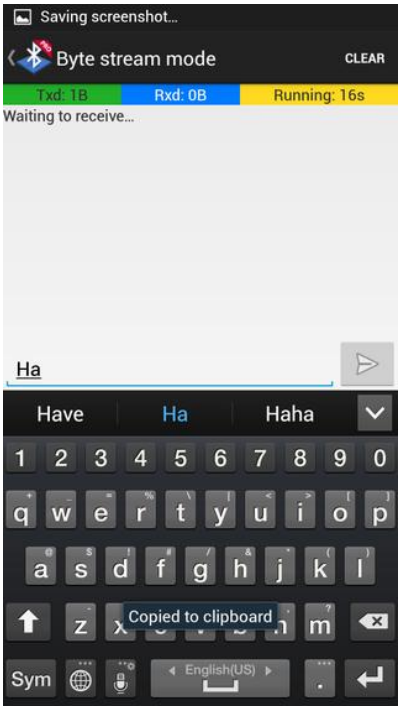
```
incomingByte = mySerial.read();

if (incomingByte == 'a') //left
{
digitalWrite(dir1PinA, LOW);
digitalWrite(dir2PinA, LOW);
digitalWrite(dir1PinB, HIGH);
digitalWrite(dir2PinB, LOW);
}
if (incomingByte == 'd') //right
{
digitalWrite(dir1PinA, HIGH);
digitalWrite(dir2PinA, LOW);
digitalWrite(dir1PinB, LOW);
digitalWrite(dir2PinB, LOW);
}
if (incomingByte == 's') //Stop
{
digitalWrite(dir1PinA, LOW);
digitalWrite(dir2PinA, LOW);
digitalWrite(dir1PinB, LOW);
digitalWrite(dir2PinB, LOW);
}
if (incomingByte == 'w') //Forward
{
digitalWrite(dir1PinA, LOW);
digitalWrite(dir2PinA, HIGH);
digitalWrite(dir1PinB, LOW);
digitalWrite(dir2PinB, HIGH);
}
  if (incomingByte == 'x') //Back
{
```

```
      digitalWrite(dir1PinA, HIGH);
      digitalWrite(dir2PinA, LOW);
      digitalWrite(dir1PinB, HIGH);
      digitalWrite(dir2PinB, LOW);
   }
 }
}
```

E. 블루투스 통신을 이용한 원격 조종

블루투스를 이용한 원격 조종을 위해서 스마트폰의 블루투스 기능을 사용하도록 하겠습니다. 블루투스 관련 앱을 다운로드 받아서 사용할 수 있고, 혹은 앱을 개발하여 사용할 수 있습니다. 먼저, 간단히 다운로드를 받은 후에 테스트를 해 보도록 합니다. 기본 구동 소스는 블루투스 통신을 사용하여 테스트해보실 수 있는데, 이를 위해서는 다양한 앱을 사용할 수 있으며 다음의 Bluetooth SPP Pro를 사용해 보았습니다.

https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE_SPP_PRO&hl=en



블루투스는 HC-06을 페어링하신 후에 1234라는 초기 암호를 입력하신 후에 사용하실 수 있습니다.