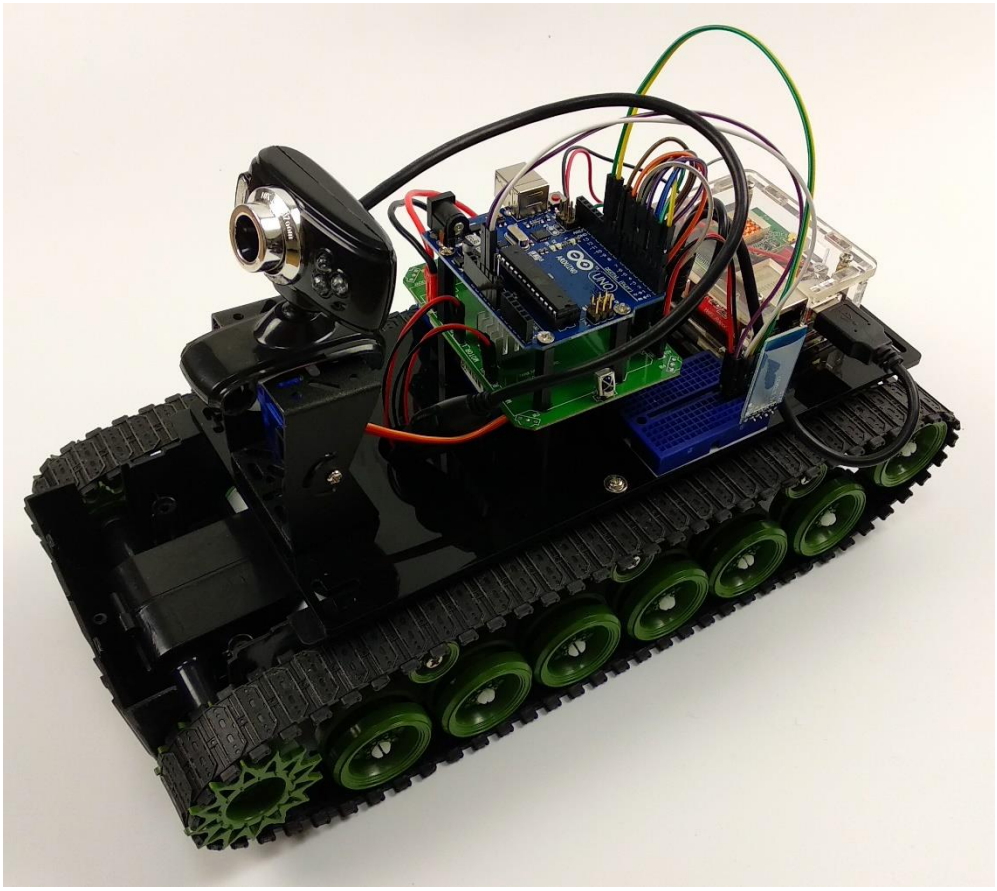
아두이노 로봇 탱크 키트

Arduino Robot Tank Kit



http://www.gameplusedu.com

http://www.gameplusbot.com

http://www.dronemaker.co.kr

아두이노 로봇 탱크 키트

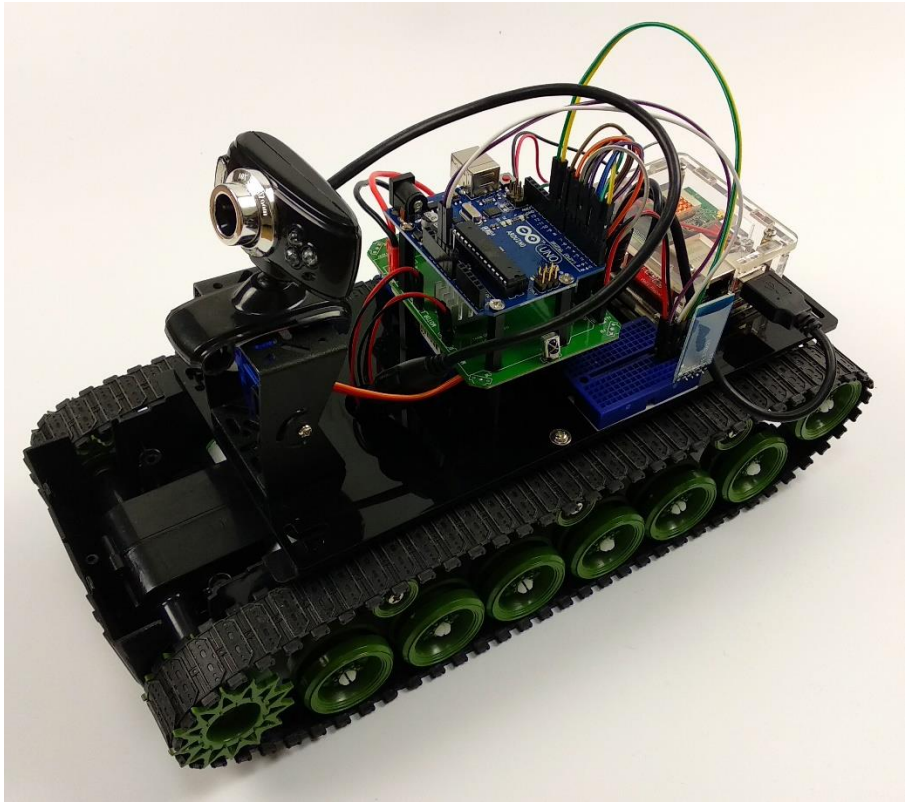
Arduino Robot Tank Kit

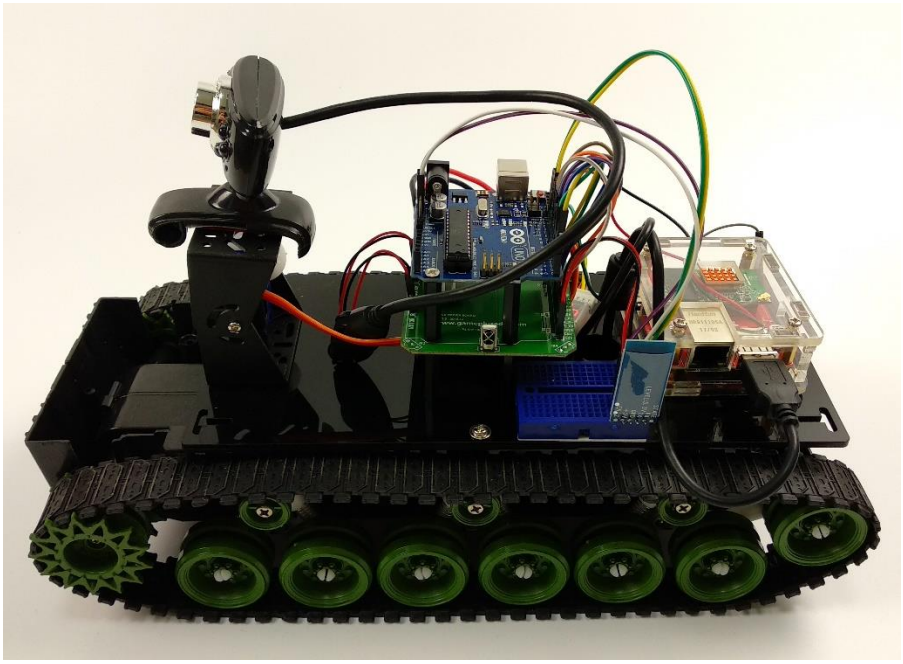
문서 버전 1.6

http://www.gameplusedu.com

http://www.gameplusbot.com

http://www.dronemaker.co.kr

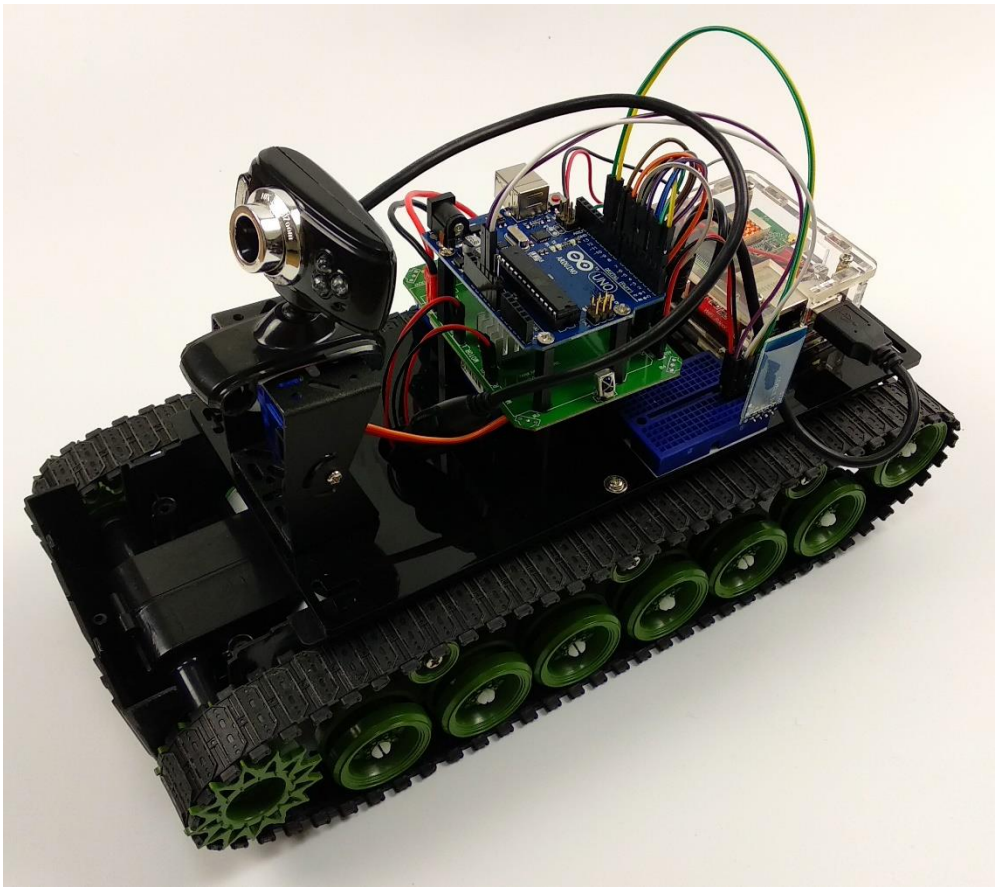




목차

1 소개

아두이노를 활용한 Arduino Robot Tank 키트입니다.



아두이노 보드를 사용하여 L293DD 제어보드를 통해 탱크를 조종합니다. 스마트폰을 이용하여 와이파이와 블루투스 통신으로 탱크를 조종할 수 있습니다. 리모트 컨트롤을 이용한 주행도 가능합니다.

아크릴 플레이트에 꼭 맞게 부품을 장착할 수 있고, 또한 지속적으로 업데이트되는 예제 코드와 부품 등을 추가하여 더 많은 기능을 구현할 수 있습니다.

처음 아두이노를 접하는 경우도 어렵지 않게 원리를 이해하고 조립할 수 있도록 설명하였으며 이해되지 않거나 수정할 부분이 있는 경우 본 도서의 기술지원 게시판, 이메일 등으로 연락주시기 바랍니다.

2 기초 기술 요구 사항

Arduino Robot Tank 운행에는 여러 종류의 하드웨어 모듈을 통합하여 사용합니다. DC 모터 제어, 와이파이 통신, 블루투스, 리모트 컨트롤 통신에 대한 이해가 필요합니다. 또한 운행을 하기 위해서는 하드웨어 모듈을 제어하기 위한 기초 수준의 C/C++ 프로그래밍 능력이 요구됩니다.

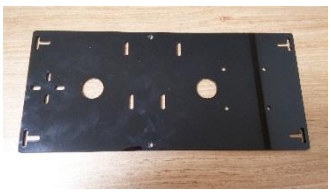
1) 아두이노 IDE 기본 사용 기술 습득
2) 아두이노 IDE 라이브러리 적용 방법
3) 아두이노 라이브러리 적용 및 사용에 대한 기본 이해
4) 조립에 필요한 부품 연결 및 하드웨어 구성 이해
5) 기초 C/C++프로그래밍 능력
6) DC 모터 제어, H-BRIDGE 제어 보드 컨트롤
7) 모듈 연결 및 응용
8) 원격 통신에 대한 이해
9) 시리얼 통신에 대한 기초 이해
10) 시스템에 소요되는 각종 부품들의 조합 및 연계 코드 구현

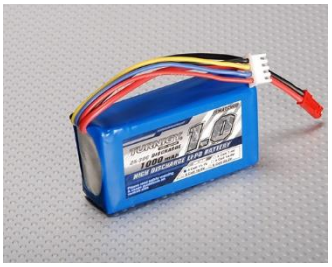
3 PACKAGE INCLUDED

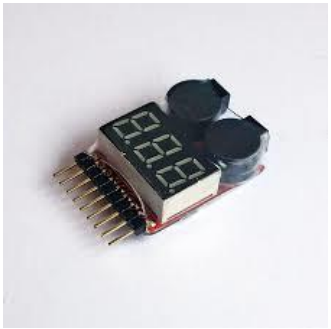
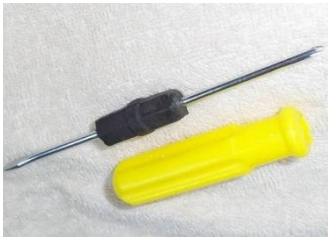
1 x 상판 아크릴 플레이트
3 x 제어보드 고정용 육각봉/너트 세트
3 x R3 고정 육각봉/볼트
3 x 거치대 아래쪽 고정용 볼트/너트
2 x 아크릴 플레이트 고정용 볼트

1 x 아두이노 우노 R3
1 x HC-06 블루투스 모듈
1 x 와이파이 모듈
1 x LiPo 배터리(3s 20-30C 1000mAh)
1 x 벨크로 테이프
1 x 서보 모터용 팬/틸트 거치대 세트
1 x 서보 모터 세트
1 x L293DD 제어보드
1 x 미니 브레드 보드
10 x 뒤퐁 M to M 케이블
20 x 뒤퐁 M to F 케이블
10 x 뒤퐁 F to F 케이블
1 x USB B-Type 케이블
1 x 리모트 컨트롤
1 x 배터리 충전기(Lipo Battery Charger B3 20W)
1 x 카메라
1 x 셀 체커
1 x 탱크 프레임 본체
1 x 드라이버
1 x 2.54 피치 40 핀 노멀 헤더핀

* 제품에 따라 구성품목이 다를 수 있음.

1 x 상판 아크릴 플레이트	**3 x 제어보드 고정용 육각봉/너트 세트**	**3 x R3 고정 육각봉/볼트**
		

3 x 거치대 아래쪽 고정용 볼트/너트	2 x 아크릴 플레이트 고정용 볼트	1 x 아두이노 우노 R3
		
1 x HC-06 블루투스 모듈	1 x 와이파이 모듈	1 x LiPo 배터리(3s 20-30C 1000mAh)
		
1 x 벨크로 테이프	1 x 서보 모터용 팬/틸트 거치대 세트	1 x 서보 모터 세트
		
1 x L293DD 제어보드	1 x 미니 브레드 보드	10 x 뒤퐁 M to M 케이블

		
20 x 뒤퐁 M to F 케이블	10 x 뒤퐁 F to F 케이블	1 x USB B-Type 케이블
		
1 x 리모트 컨트롤	1 x 배터리 충전기(Lipo Battery Charger B3 20W)	1 x 카메라
		
1 x 셀 체커	1 x 탱크 프레임 본체	1 x 드라이버
		

1 x 2.54 피치 40 핀 노멀 헤더핀		
		

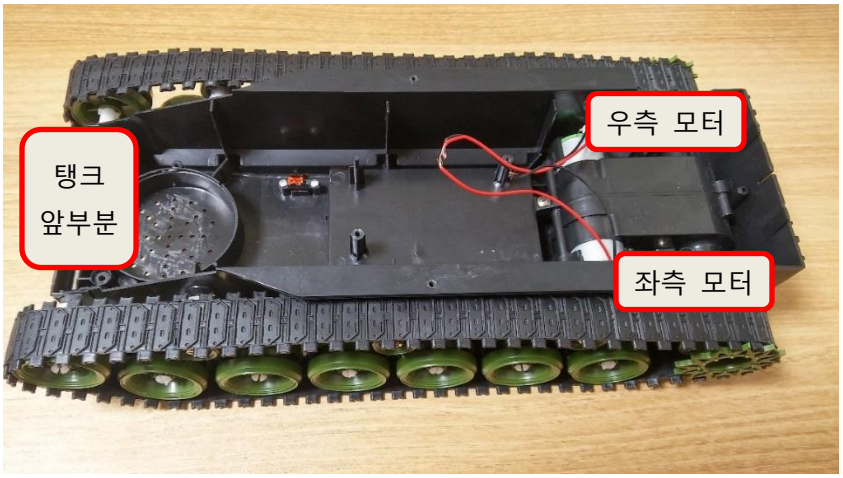
4 조립

탱크는 프레임 본체, 상판 아크릴 플레이트, 아두이노 우노 R3, 미니 브레드 보드, L293DD 제어보드, 배터리, 카메라, 와이파이 모듈, 블루투스 모듈 등으로 이루어져 있습니다.

4.1 하부 조립

탱크 하부는 프레임 본체로 이루어집니다.

4.1.1 프레임 본체



조립되어 동봉된 프레임 본체를 준비합니다.

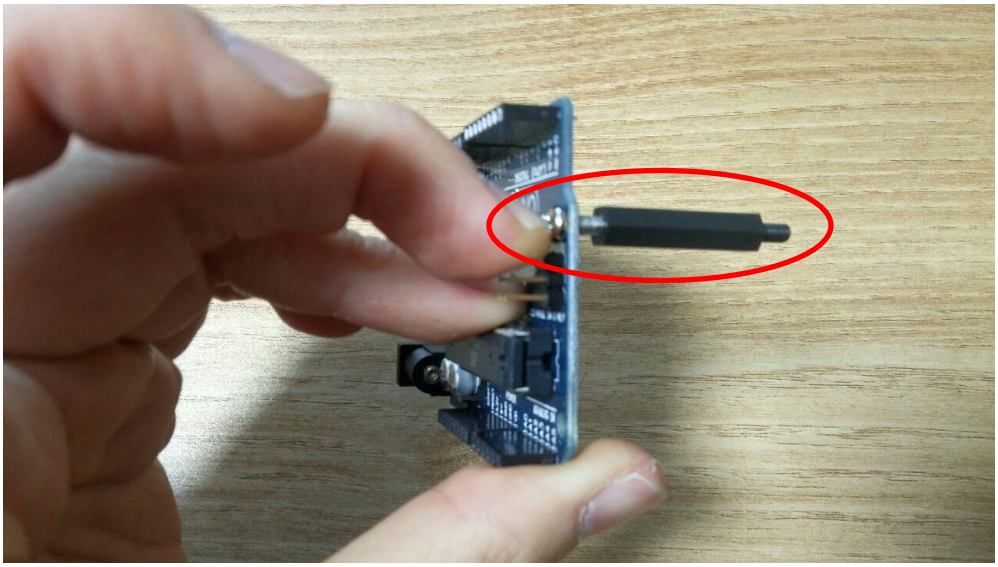
4.2 상부 조립

탱크 상부는 아두이노 보드, 미니 브레드 보드, L293DD 제어보드, 와이파이 모듈, 배터리, 블루투스 모듈 등으로 구성됩니다.

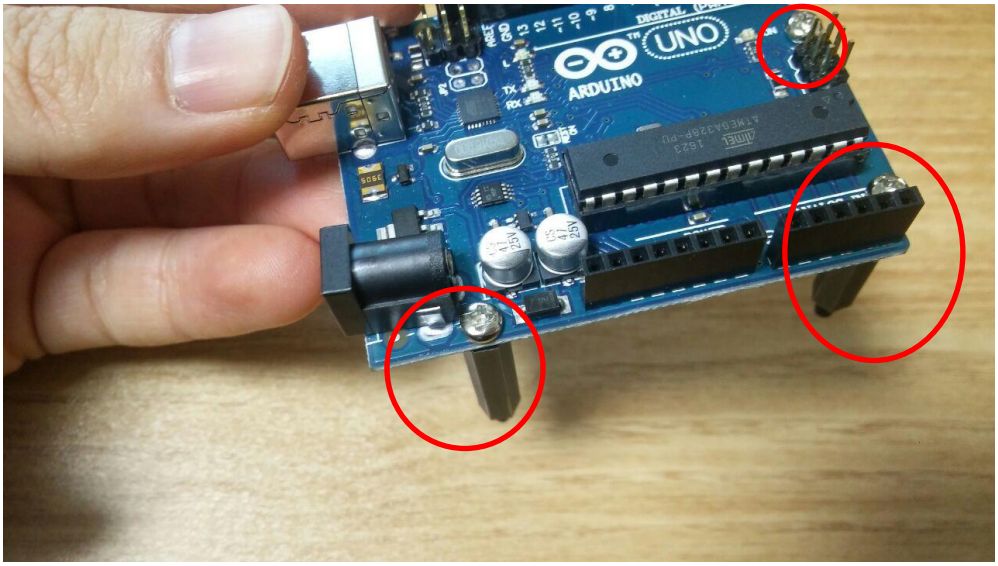
4.2.1 L293DD 제어보드, 아두이노 우노 R3 장착

아크릴 플레이트

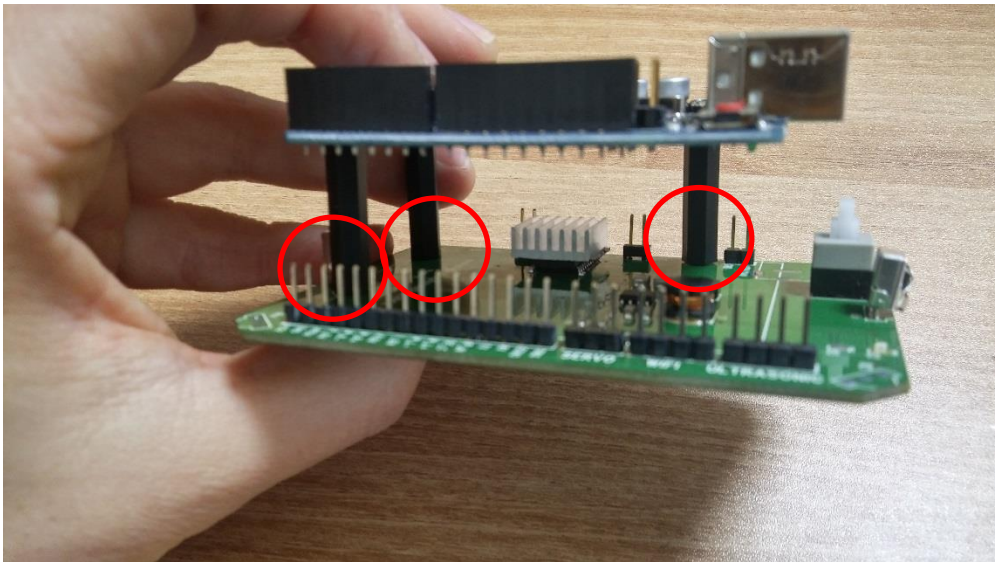
R3 와 R3 고정 육각봉/볼트



R3 의 구멍에 볼트를 넣고 볼트와 육각봉을 결합합니다.

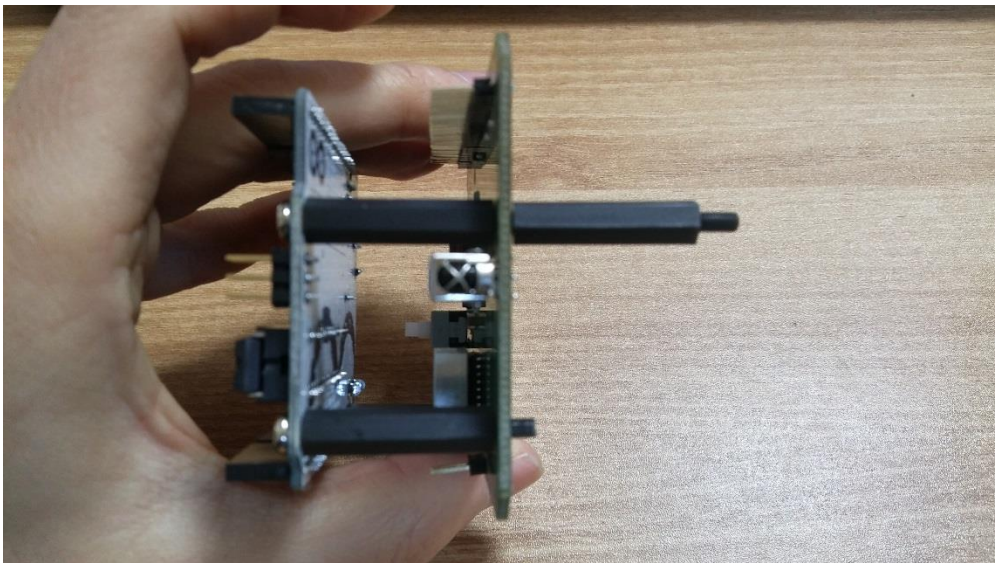


4 곳이 아닌 3 곳에 볼트/육각봉을 결합합니다.

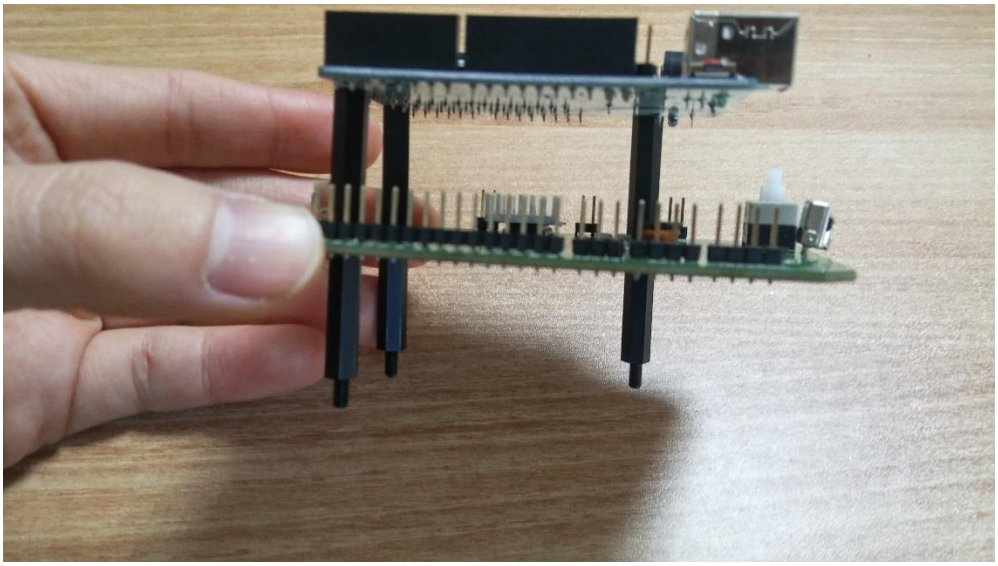


R3 의 육각봉을 L293DD 제어보드의 구멍에 넣어줍니다.

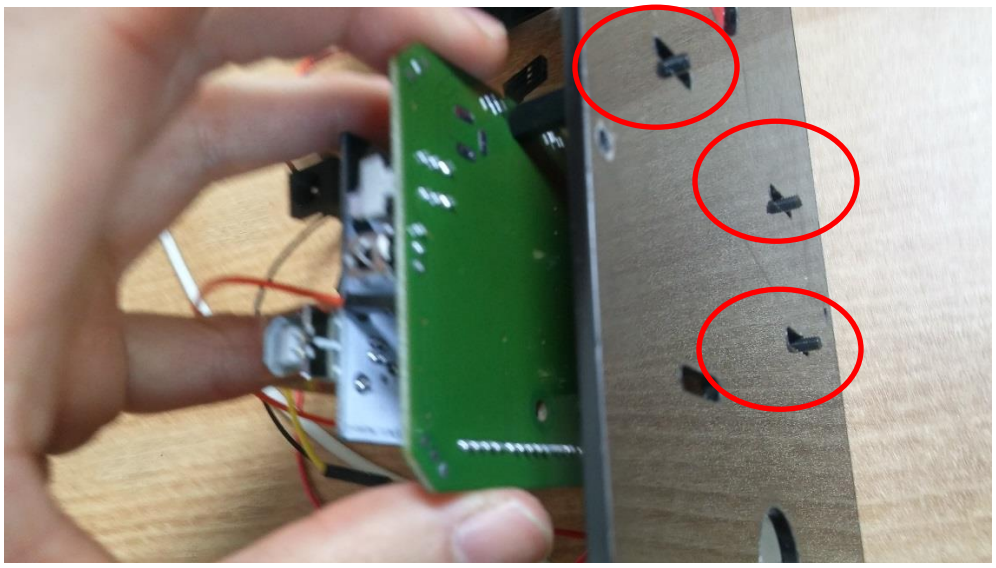
제어보드 고정용 육각봉/너트



제어보드 아래로 튀어나온 부분에 다시 육각봉을 결합합니다.

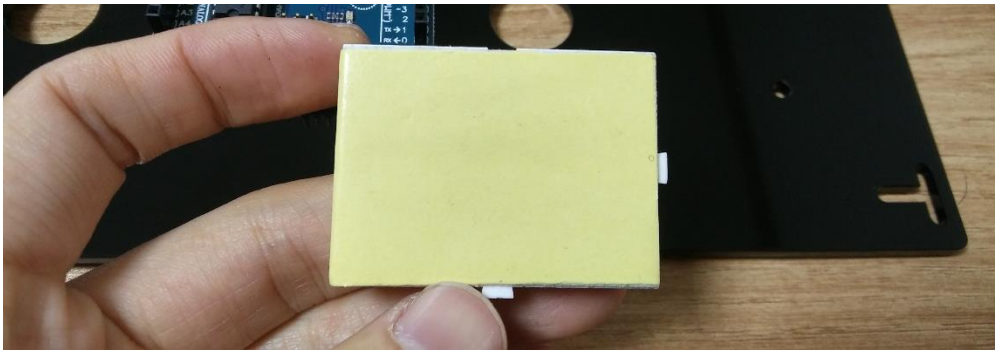


제어보드와 R3 가 육각봉으로 결합된 모습
*3 곳의 구멍을 이용하여 결합합니다.

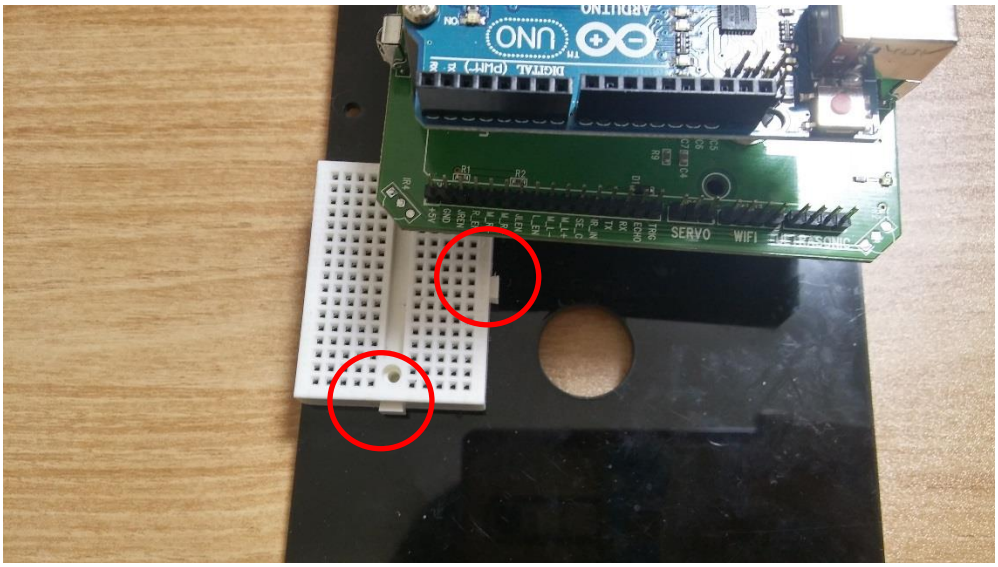


플레이트의 홀더에 육각봉을 넣어주고 너트로 조여줍니다. 3 곳 모두 조여줍니다.

4.2.2 배터리, 브레드 보드, 와아파이 모듈 장착

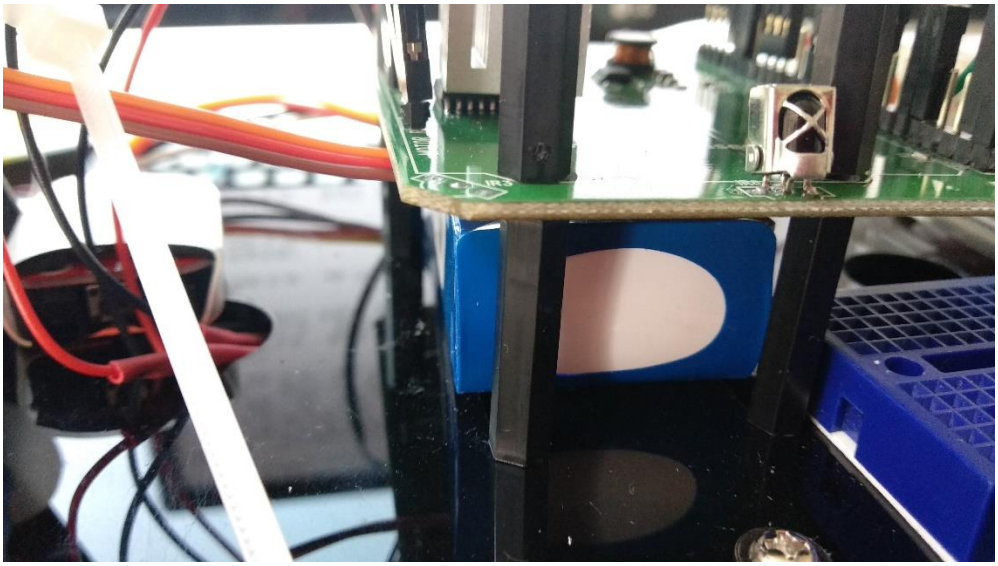


미니 브레드 보드를 준비합니다(브레드 보드 뒷면에 스티커가 있습니다)

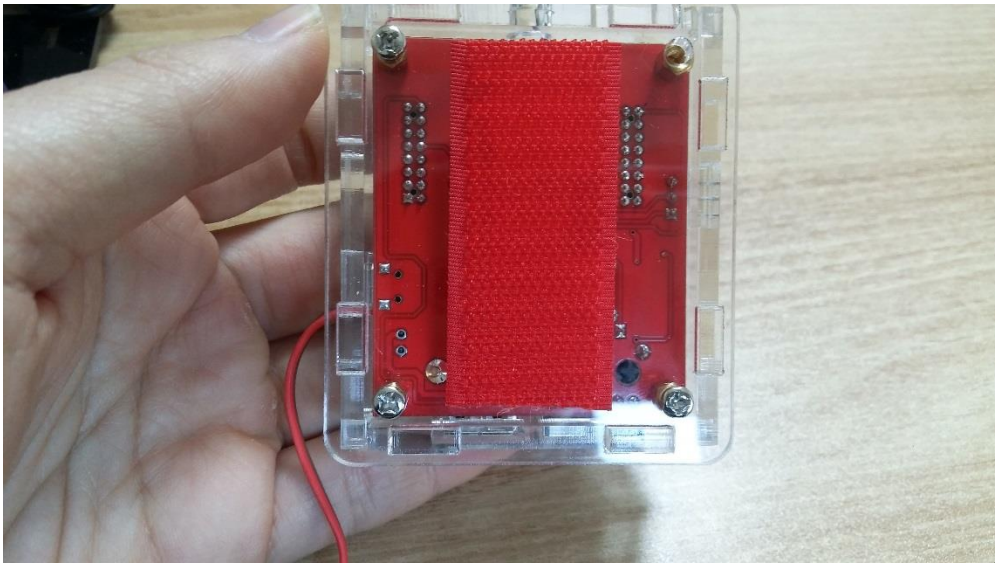


브레드 보드 뒷면을 스티커를 이용하여 위와 같은 위치에 미니 브레드 보드를 장착합니다.

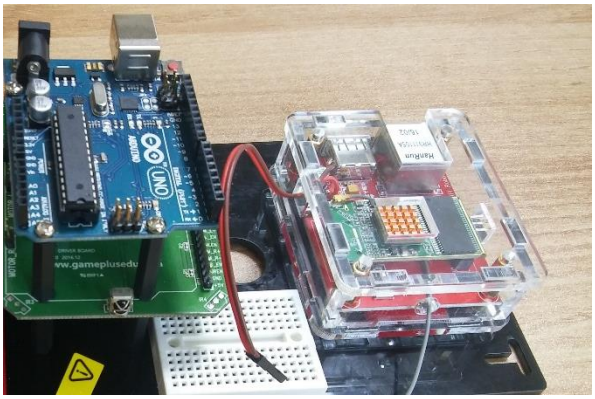
*위 그림처럼 브레드 보드의 튀어나온 홈이 오른쪽과 뒤쪽에 위치하도록 부착합니다.



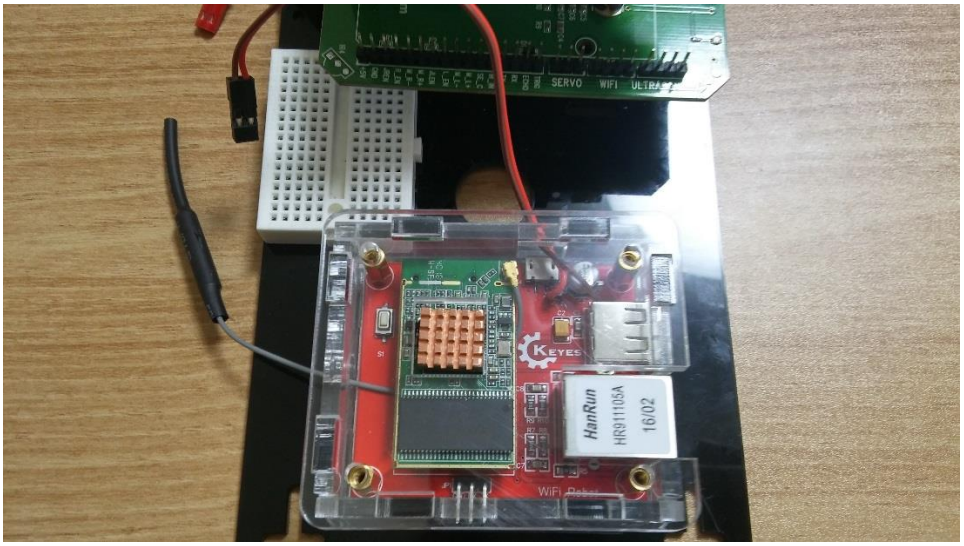
아크릴 플레이트와 배터리에 벨크로 테이프를 부착하고, 배터리를 위와 같이 제어보드 아래로 넣어 부착합니다.



와이파이 모듈 바닥에 밸크로 테이프를 부착합니다.



위와 같은 위치에 벨크로 테이프를 이용하여 와이파이 모듈을 부착합니다.

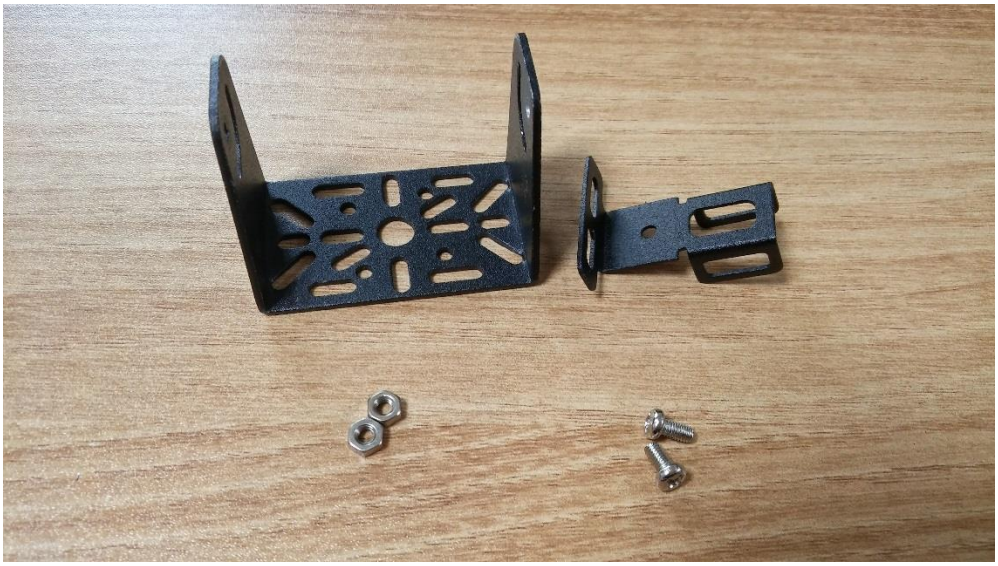


와이파이 모듈이 부착된 모습

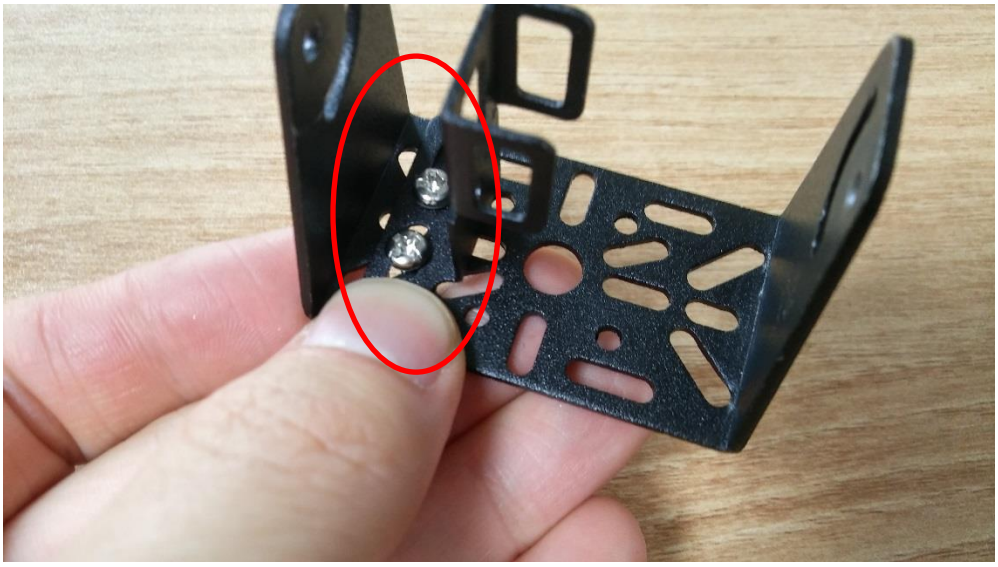
4.2.3 카메라 장착

카메라 고정에 쓰일 팬/틸트 거치대 부품입니다.

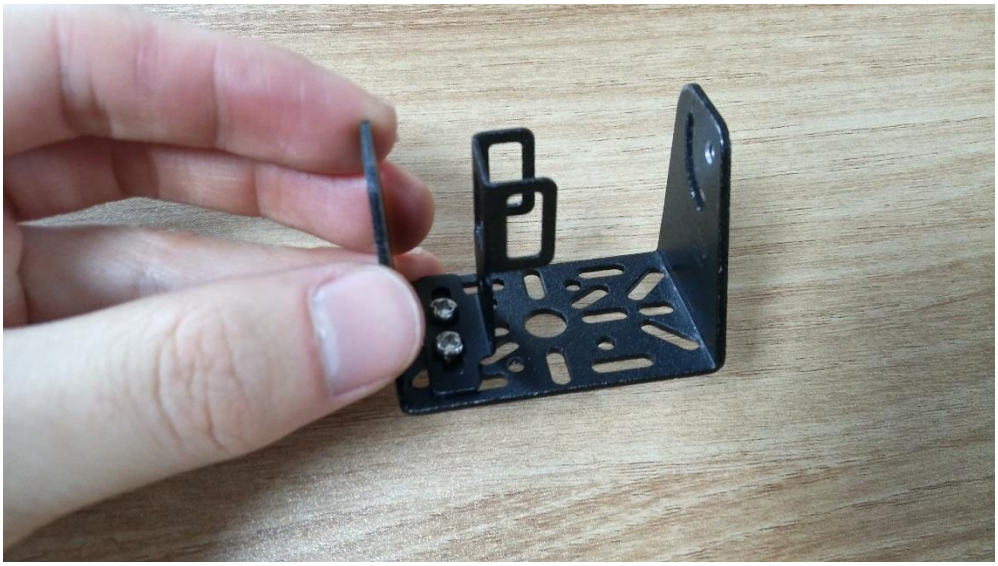
서보 모터와 부속품입니다.



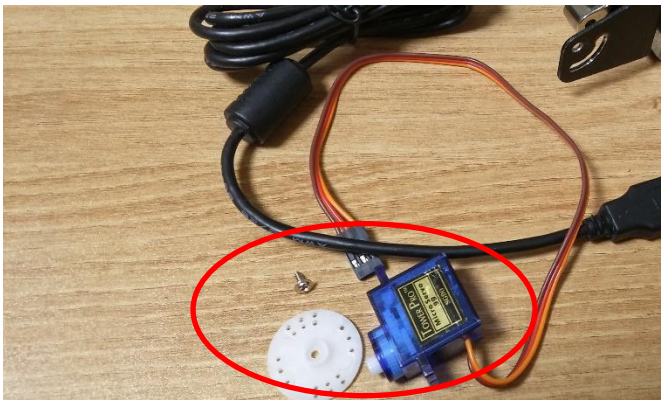
위와 같은 부품들을 준비합니다.



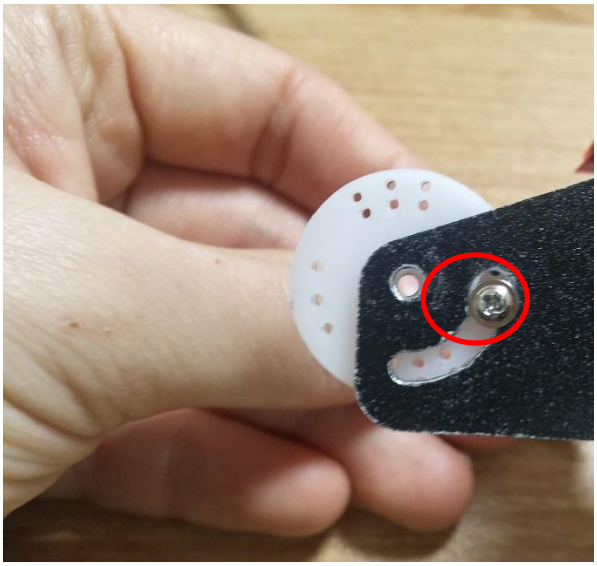
위 그림처럼 볼트와 너트를 이용하여 조립합니다.



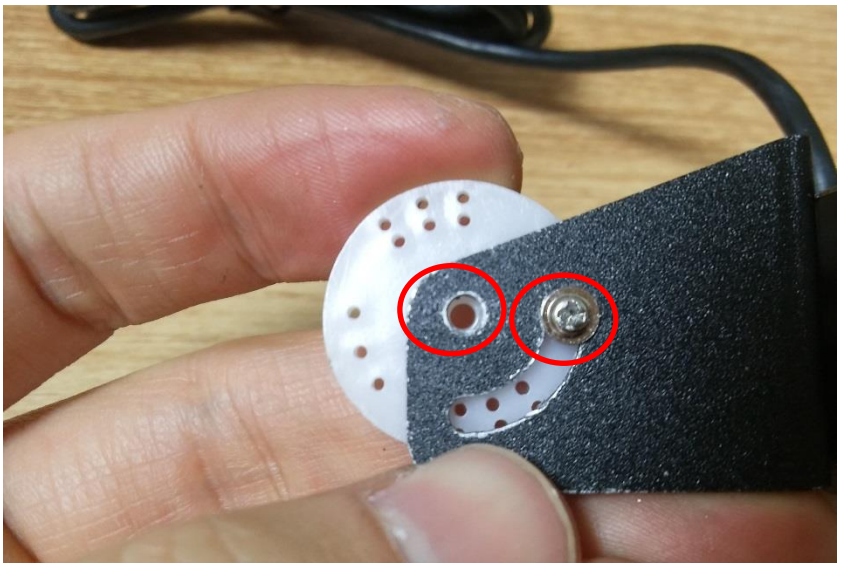
조립된 카메라 거치대 하판의 모습



완성된 거치대 상판과 서보모터, 서보모터 휠, 휠 고정용 볼트의 모습

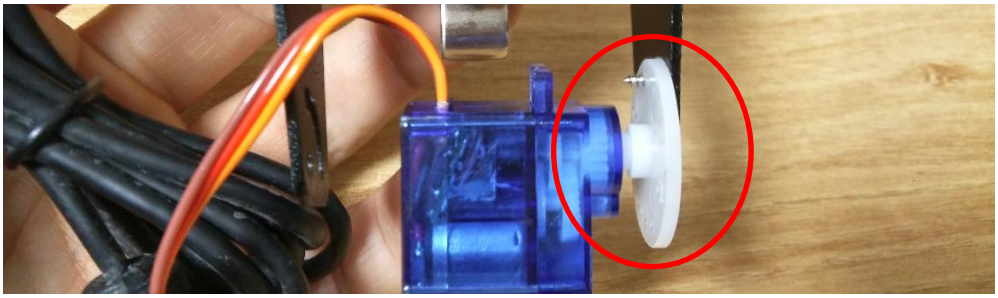


서보모터 휠을 위와 같이 지지대 상판에 위치시키고 제공된 볼트로 조여줍니다.

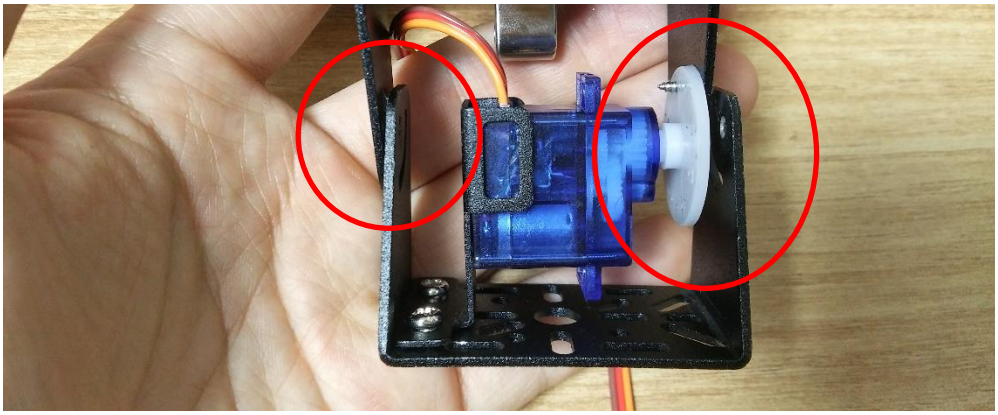


드라이버를 사용하여 휠과 지지대 상판을 단단히 고정시킵니다. 휠과 상판이 따로 움직여서는 안됩니다.

*휠의 가운데 구멍은 상판의 구멍과 일치해야 합니다.



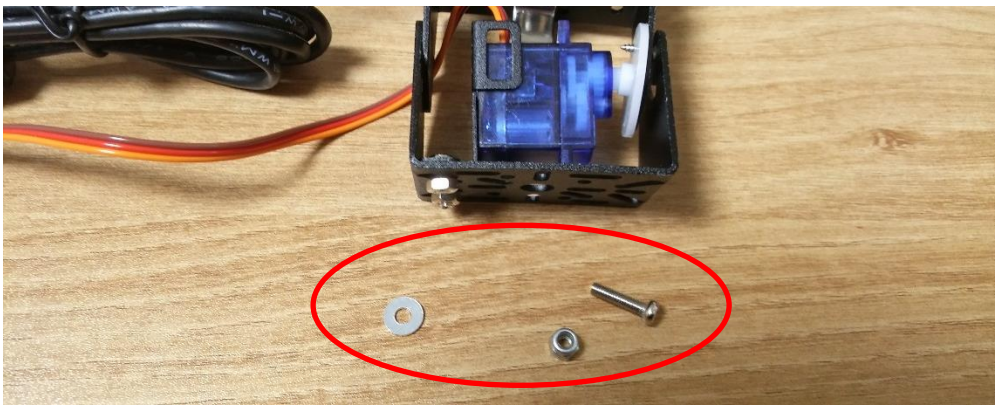
휠과 지지대 상판이 고정되었으면, 위와 같이 휠과 서보모터를 결합합니다.



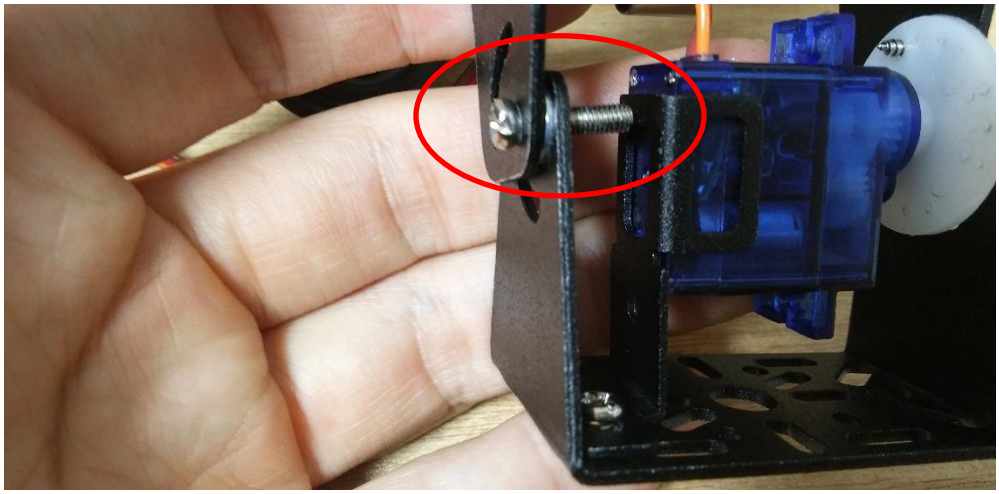
서보모터를 지지대에 맞게 끼워주고, 지지대 상판과 하판을 위 그림처럼 배치합니다.

*상판 왼쪽팔, 하판 왼쪽팔, 상판 오른쪽팔, 하판 오른쪽팔의 순서입니다.

*휠과 모터가 딱 들어맞도록 모터를 휠의 구멍에 밀어넣어 끼웁니다.

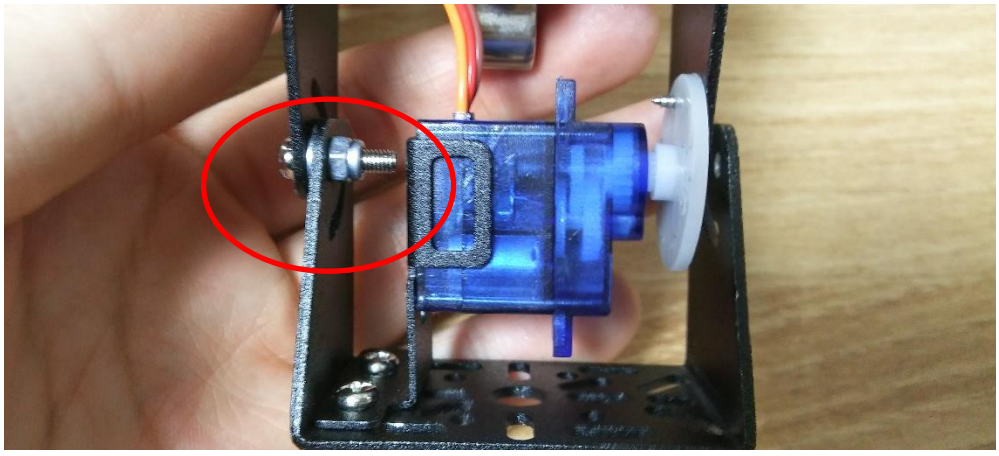


지지대의 왼쪽을 결합할 부품을 준비합니다.



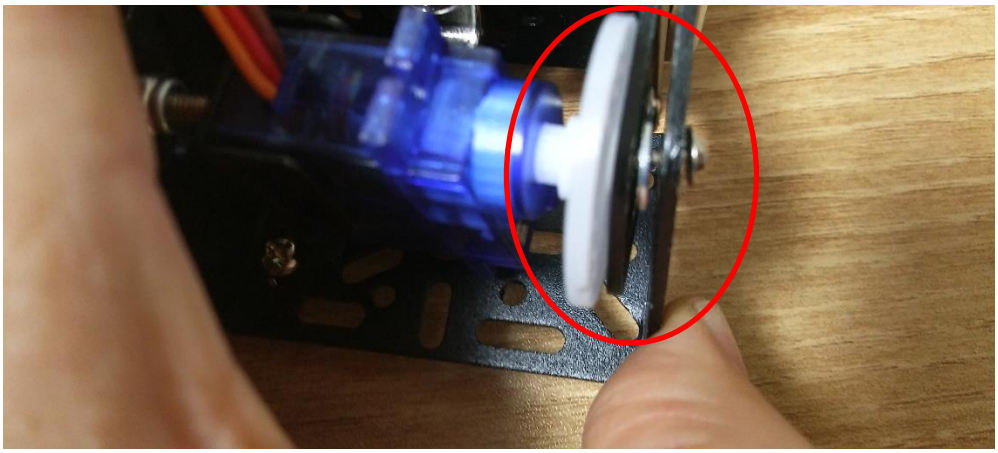
지지대 왼쪽 구멍에 볼트를 끼운 모습

*볼트, 상판, 와셔, 하판 순서입니다.

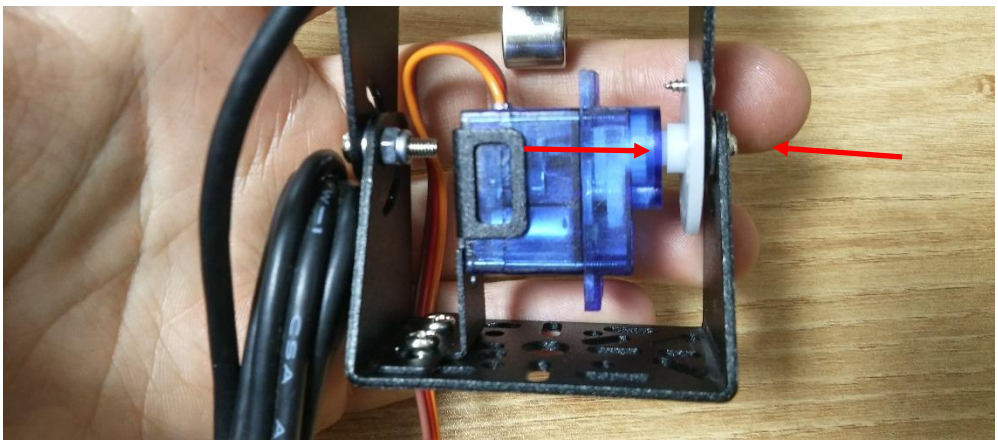


스패너 등의 공구를 이용하여 너트를 조여줍니다.

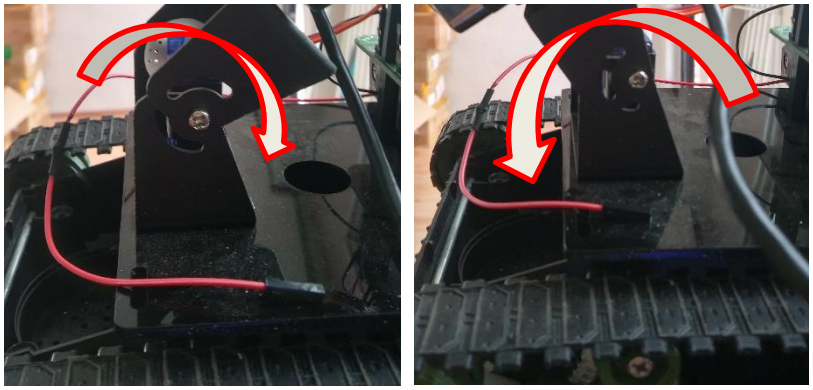
이제는 지지대의 오른쪽을 결합합니다. 결합 부품을 준비합니다.



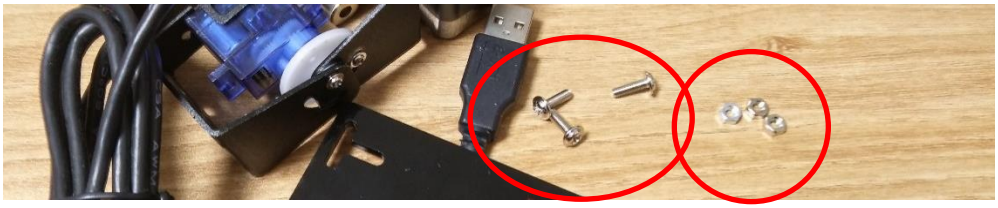
위와 같이 하판 구멍과 상판 구멍, 그리고 휠의 구멍을 결합합니다
*휠, 상판 구멍, 와셔, 하판 구멍, 볼트 순서입니다



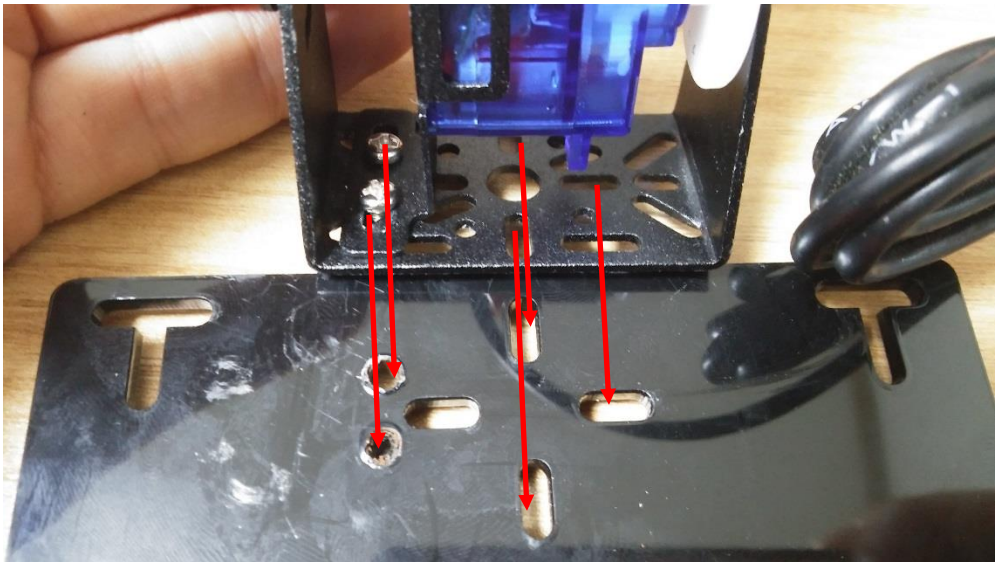
드라이버 등을 이용하여 볼트를 조여줍니다. 지지대 오른쪽과 서보모터 사이에 틈이 없도록 양쪽에서 눌러준 채 볼트로 조입니다.
* 하판을 손으로 잡고, 상판을 앞뒤로 젖혔을 때 모터와 맞물려 돌아갈 수 있도록 조여줍니다. 너무 세게 조이면 돌아가지 않으며, 더욱 약하게 조이면 지지대 자체가 흔들릴 수 있습니다.



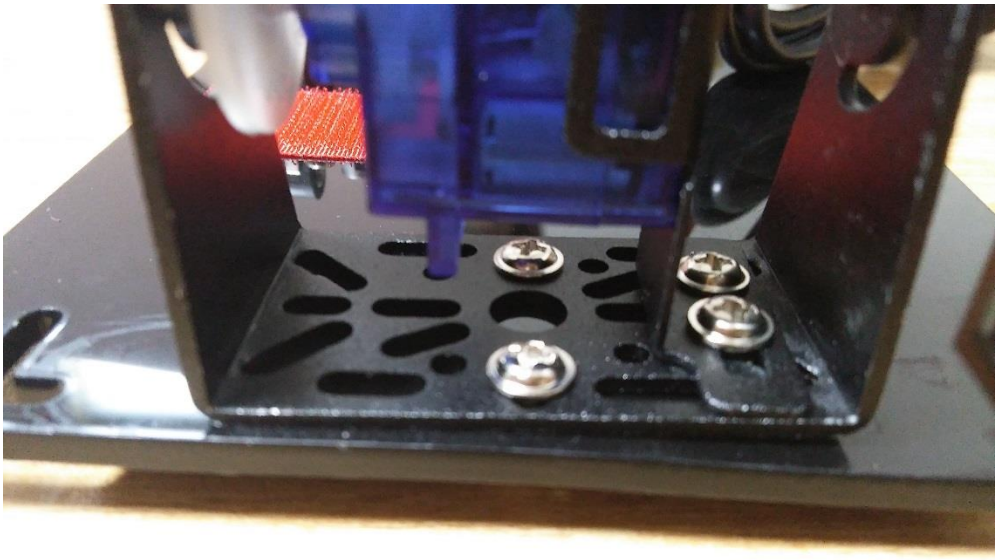
상판이 모터와 맞물려 앞,뒤로 돌아가는 모습



이제는 카메라 지지대와 아크릴 플레이트를 결합합니다. 위와 같은 부품을 준비합니다.



결합할 홀더의 위치

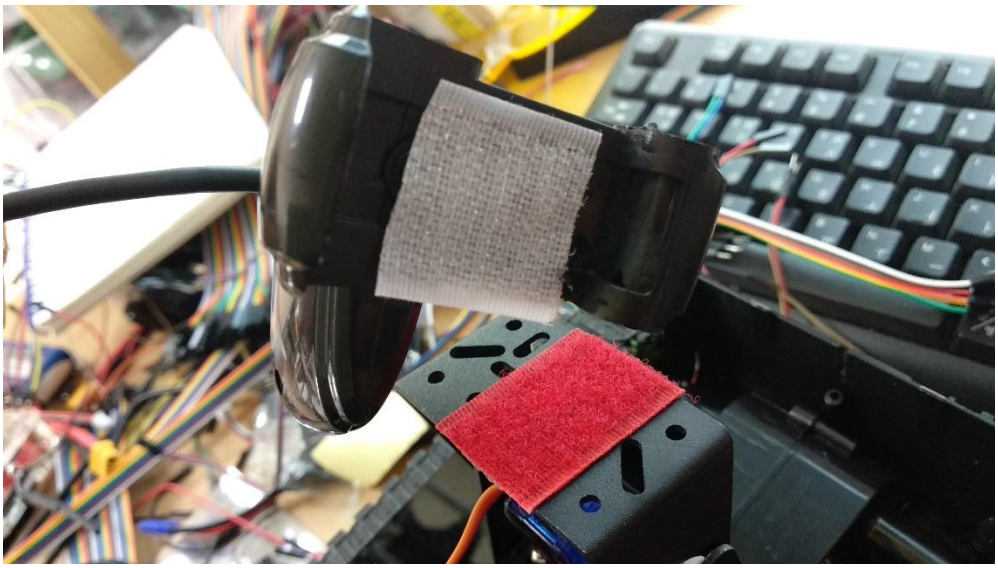


볼트로 지지대를 고정한 모습

*서보모터 지지대 볼트의 경우 너트를 풀어서 너트를 뒷면에 고정시키면 됩니다.

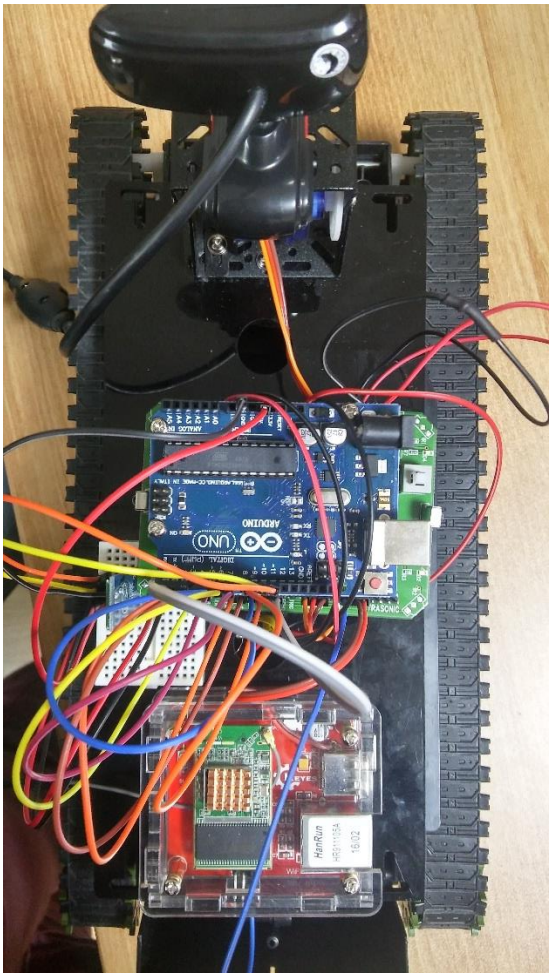
지지대와 결합된 아크릴 플레이트의 뒷면

카메라를 위와 같이 분리합니다.

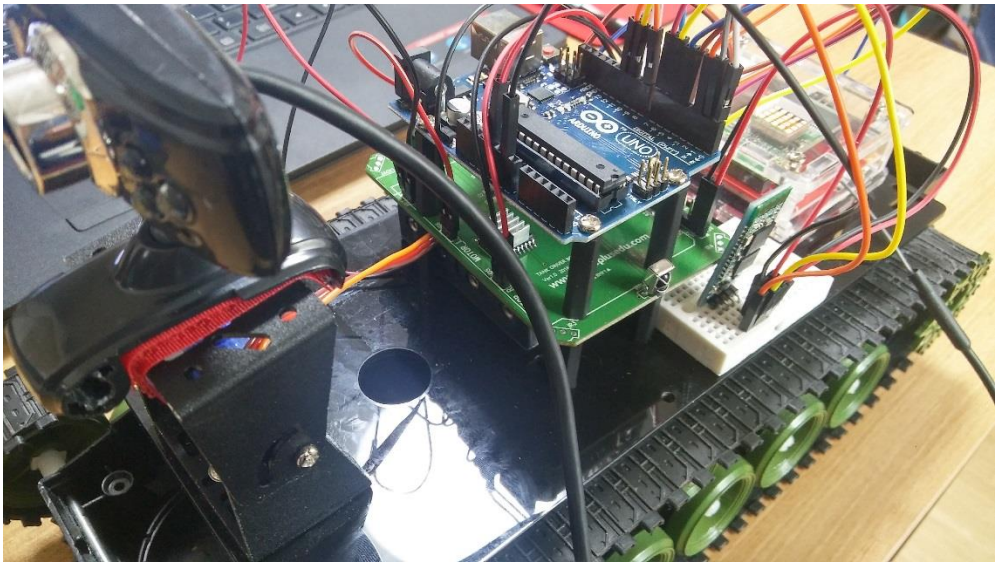


위와 같이 벨크로 테이프를 부착합니다.

카메라가 부착된 모습

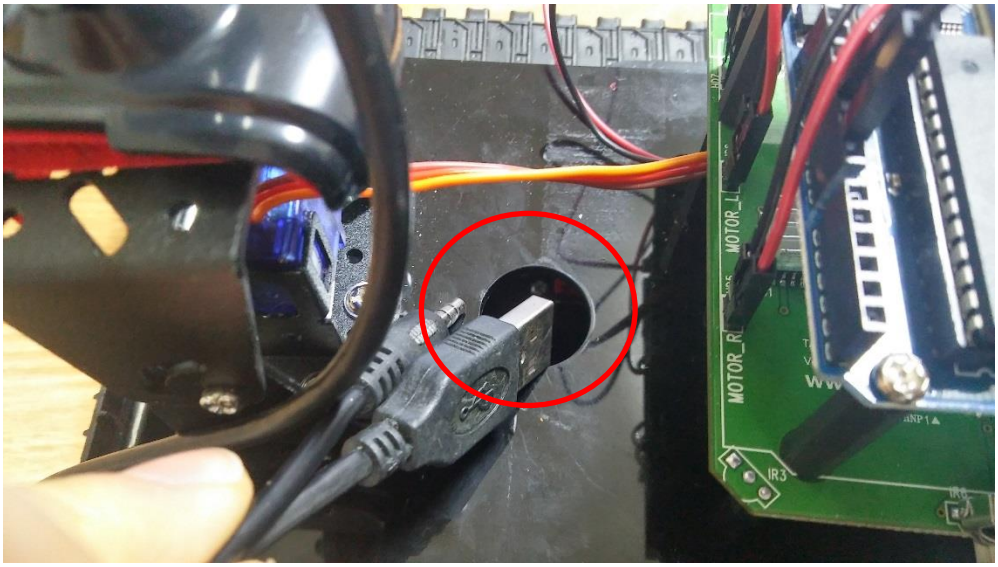


카메라 장착이 완료된 플레이트의 모습 1

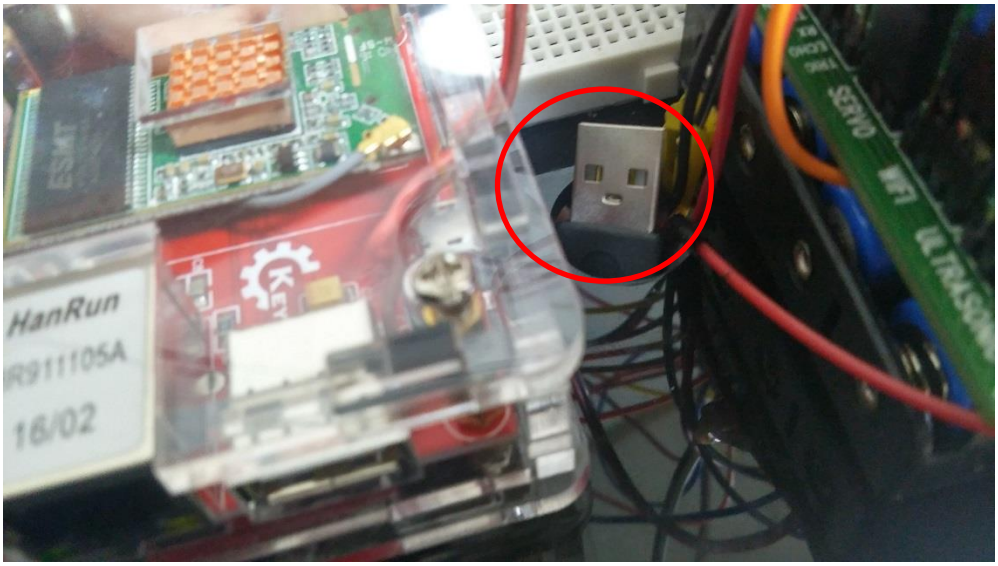


카메라 장착이 완료된 플레이트의 모습 2

4.2.4 프레임 본체와 아크릴 플레이트 결합

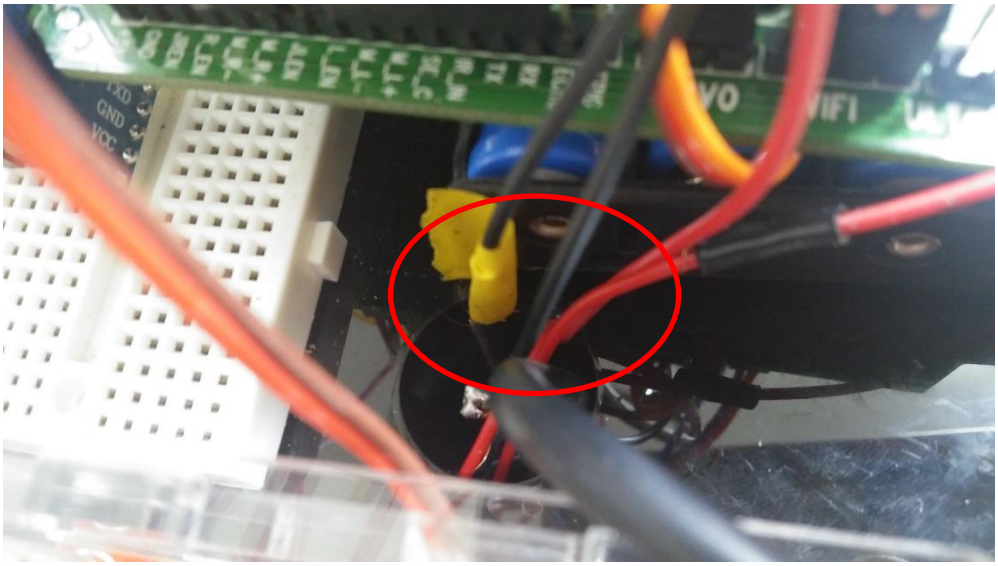


아크릴 플레이트를 프레임 본체 위에 올려주고, 플레이트의 구멍을 통해 카메라 선을 아래로 내립니다.

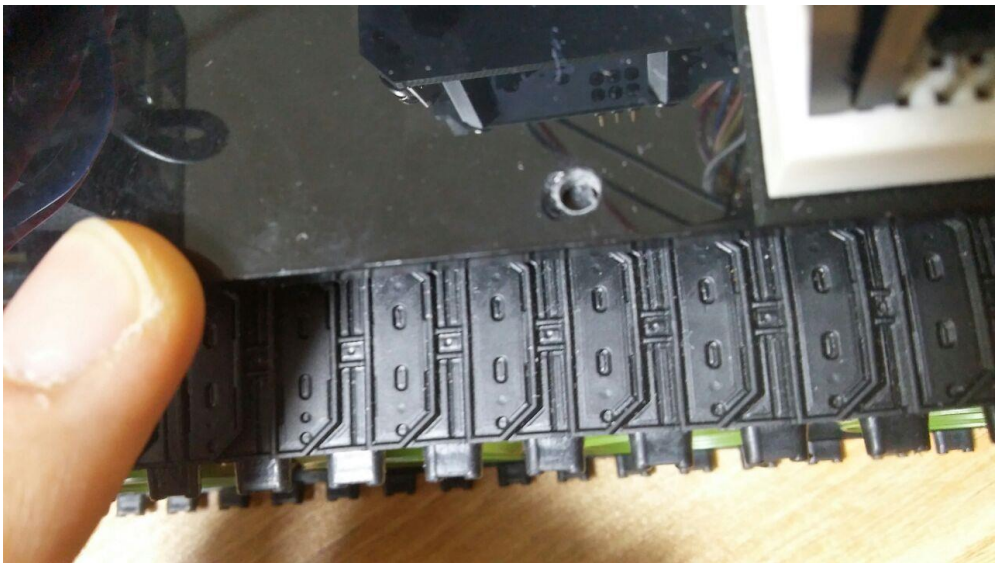


넣어진 카메라 선(USB)을 뒤쪽의 아크릴 구멍으로 올려줍니다.

카메라의 USB 케이블을 와이파이 모듈의 USB 단자에 꽂습니다.



DC 모터 선 네 가닥도 아크릴 뒤편 구멍으로 올려줍니다.

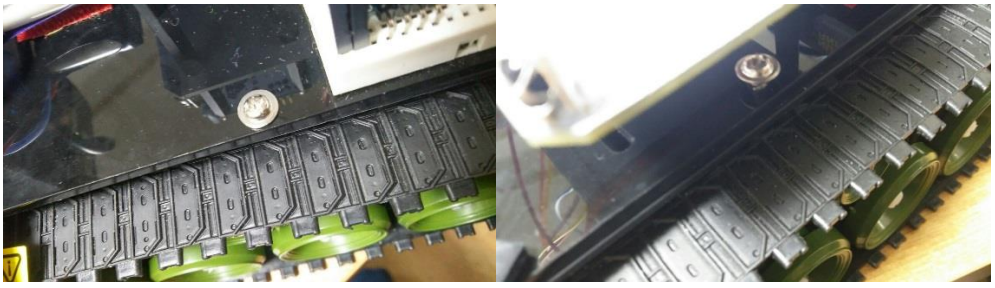


프레임 본체의 구멍과 플레이트의 구멍이 일치하도록 위치를 조정합니다

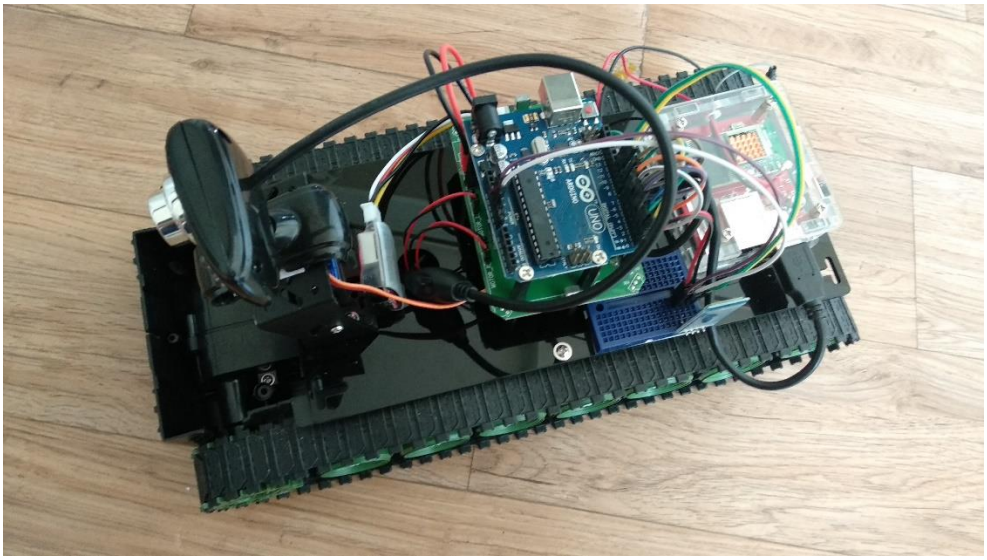
플레이트 고정용 나사를 준비합니다.

드라이버를 이용하여 나사를 조여줍니다

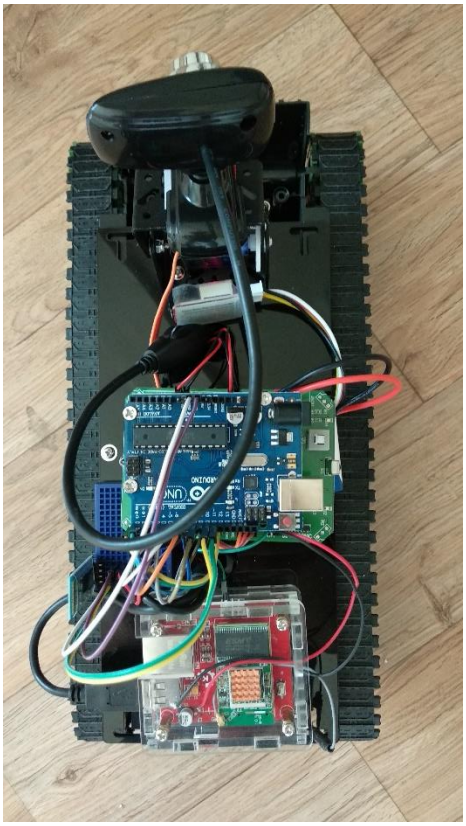
*플레이트가 움직이지 않도록 단단히 조여줍니다.



왼쪽, 오른쪽 홀더에 나사를 끼워 고정시킨 모습

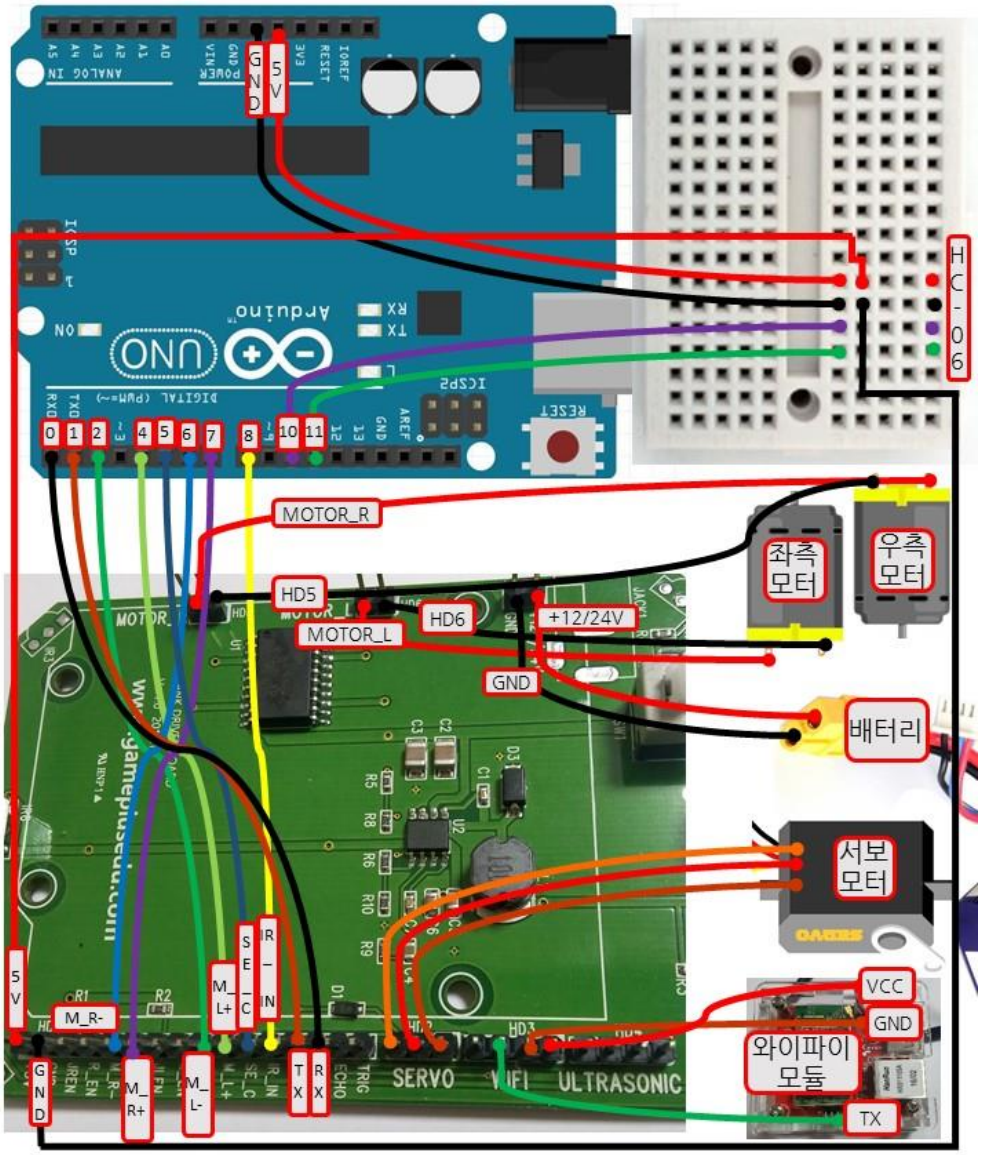


플레이트와 프레임 본체가 연결된 모습



플레이트와 프레임 본체가 연결된 모습

4.2.5 R3, 제어보드, 블루투스 모듈, 와이파이 모듈 결합



연결 회로도

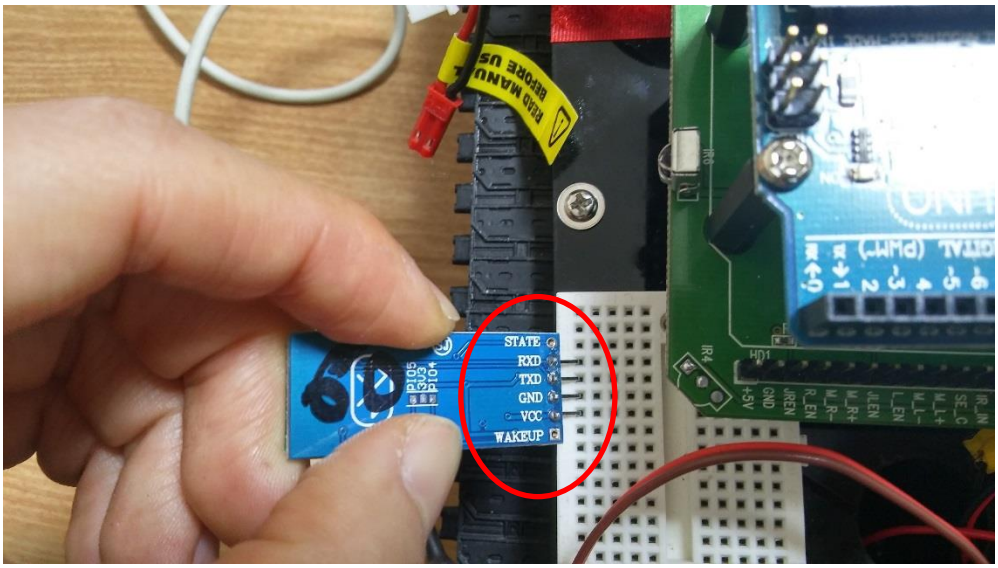
HC-06	브레드 보드	서보	L293DD 제어보드	아두이노 R3	배터리	와이파이 모듈	DC 모터
		주황색	왼쪽				
		빨강색	중간				
		갈색	오른쪽				
			MOTOR_L				좌측 모터 빨강선
			HD6				좌측 모터 검은선
			MOTOR_R				우측 모터 빨강선
			HD5				우측 모터 검은선
			WIFI 좌 2			TX	
			WIFI 좌 3			검은선	
			WIFI 좌 4			빨간선	
			+12/24V		빨간선		
			GND		검은선		
			M_L+	4			
			M_L-	2			
			M_R+	7			
			M_R-	6			
			IR_IN	8			
			SE_C	5			
			TX	1(TX->1)			
			RX	0(RX->0)			
VCC	J11						
GND	J12						
TXD	J13						
RXD	J14						
	F11			5V			
	F12			GND			
	F13			DIGITAL 10			
	F14			DIGITAL 11			
	G11		5V				
	G12		GND				

연결표

*브레드 보드의 구멍 위치는 보드 가장자리에 표기되어 있습니다.

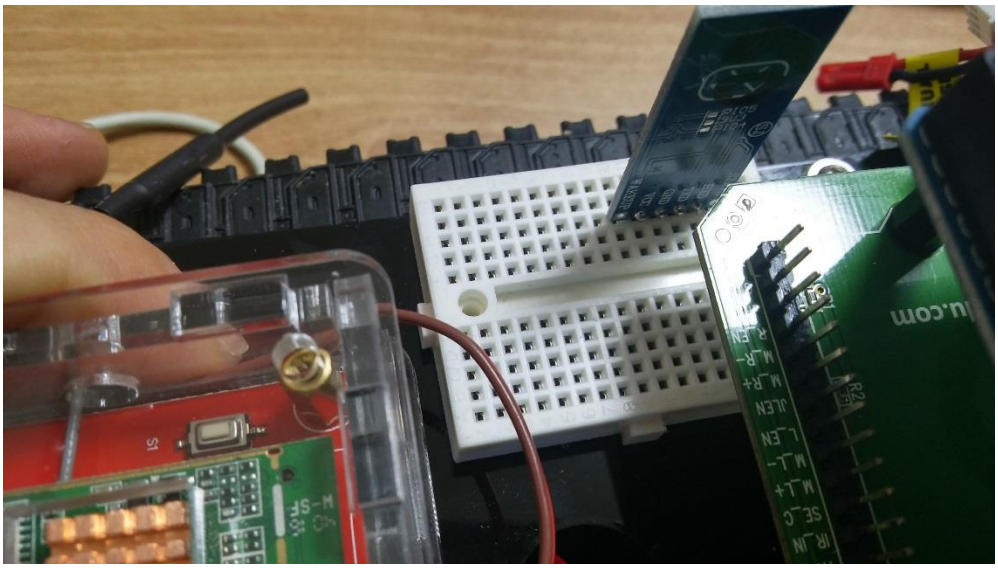
*탱크를 위에서 내려다봤을 때 왼쪽에 위치한 모터가 좌측 모터, 오른쪽에 위치한 모터가 우측 모터입니다.

*HC-06 와 브레드 보드의 연결 : HC-06 의 튀어나온 단자를 브레드 보드의 해당 구멍에 눌러 넣어줍니다.

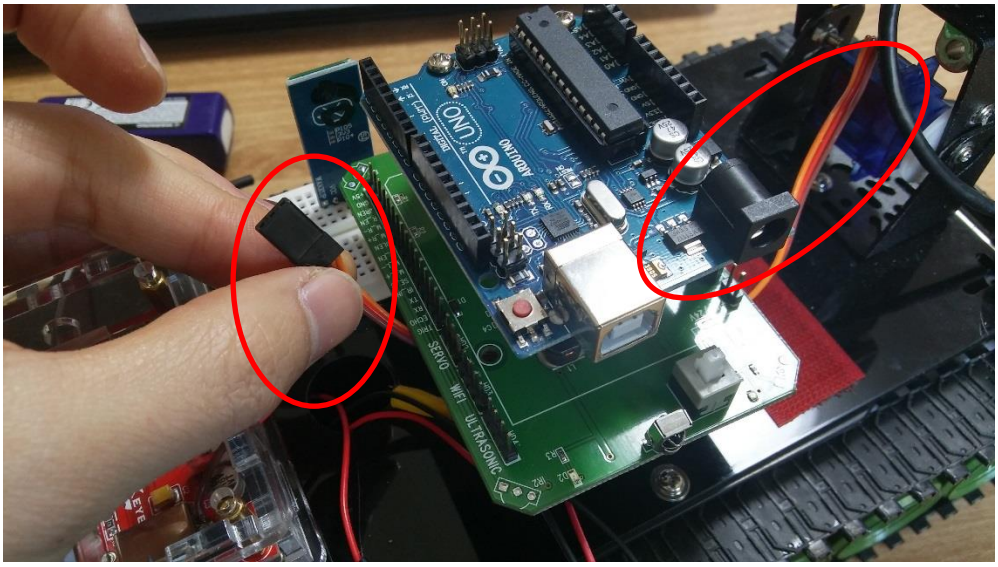


블루투스 모듈 HC-06 을 브레드 보드에 연결하는 모습

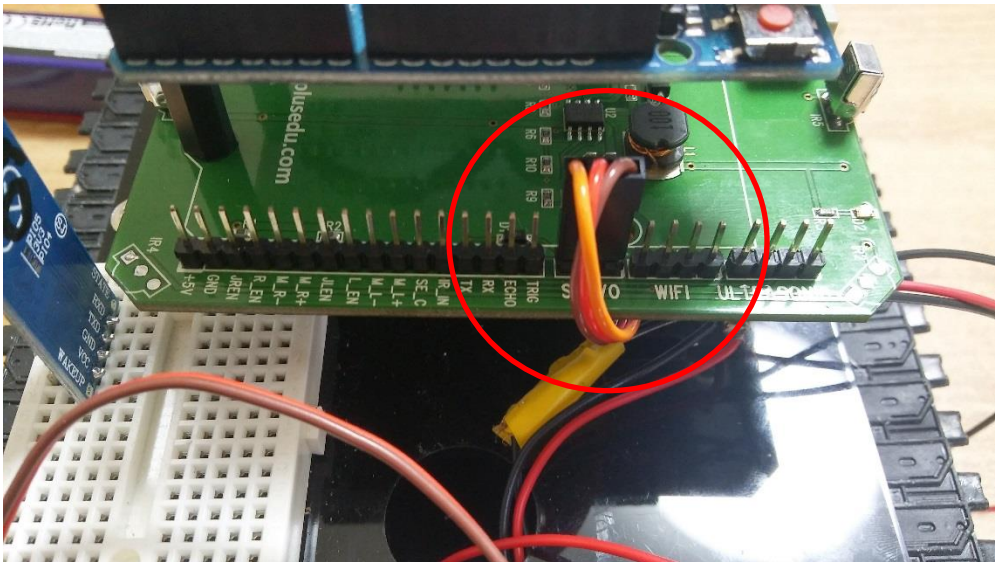
*브레드 보드의 구멍에 모듈의 튀어나온 단자를 넣어줍니다.



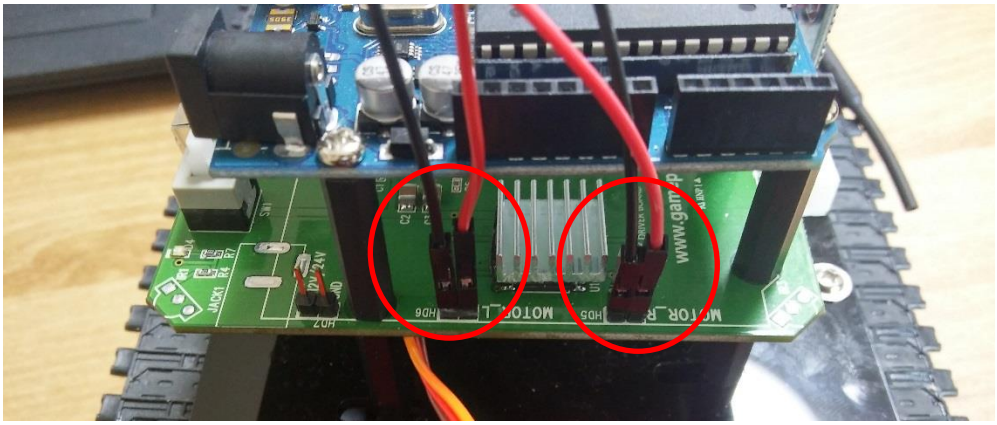
브레드 보드에 블루투스 모듈이 결합된 모습



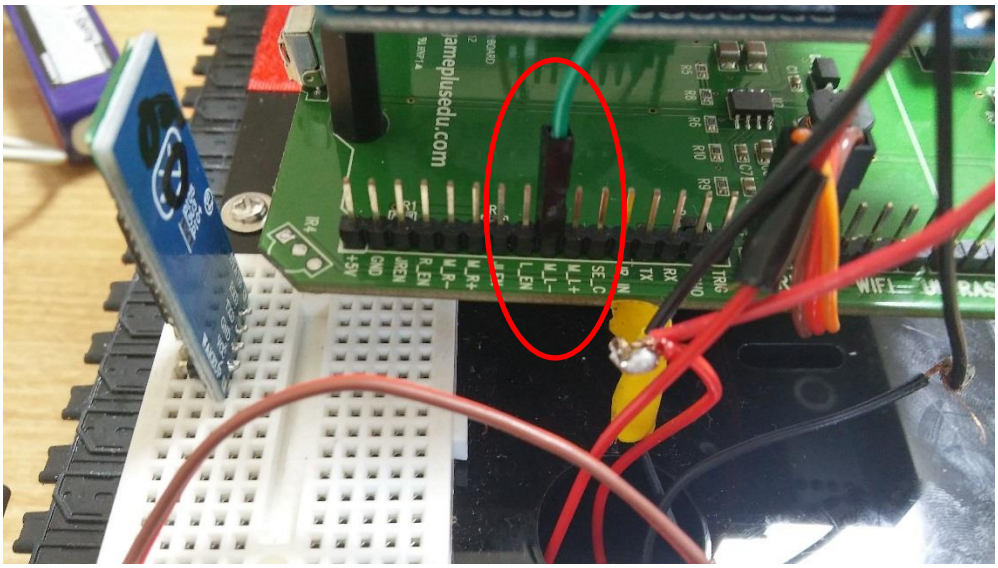
서보 모터의 선을 제어보드 밑으로 넣어 뒤쪽으로 빼줍니다.



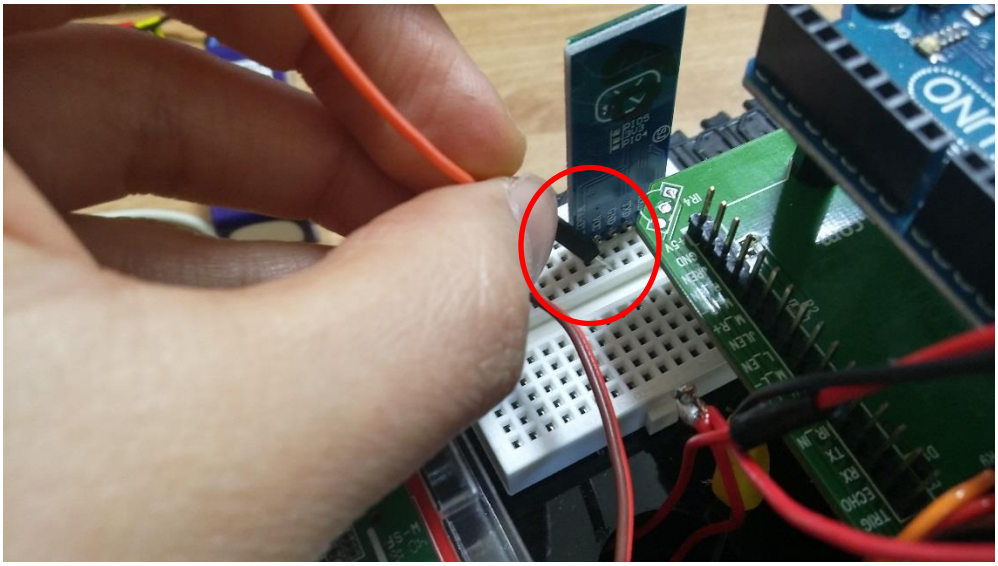
서보 모터에서 나온 선들을 제어보드에 꽂아준 모습



DC 모터에서 나온 선들을 제어보드에 꽂아준 모습

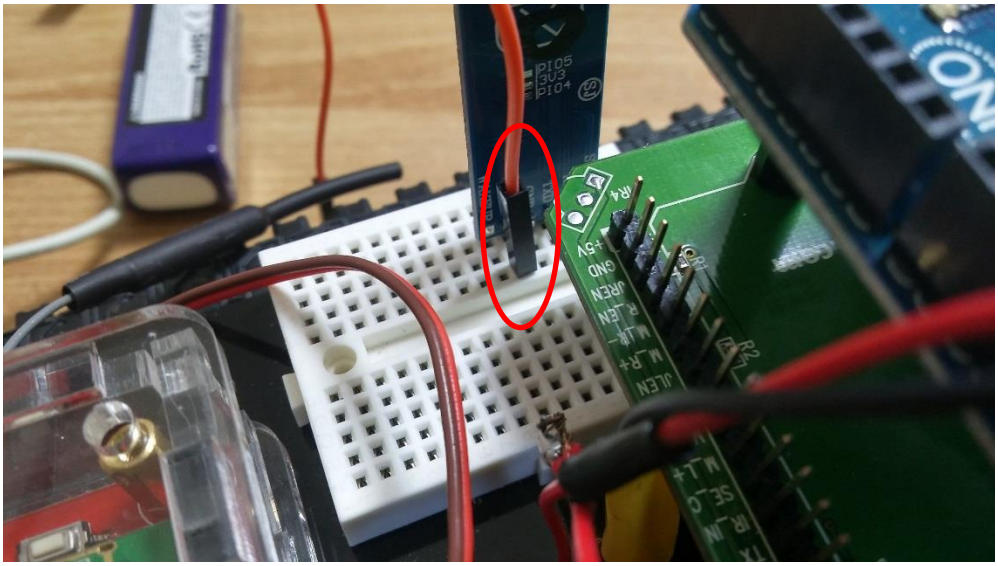


제어보드에 Female 케이블을 결합한 모습

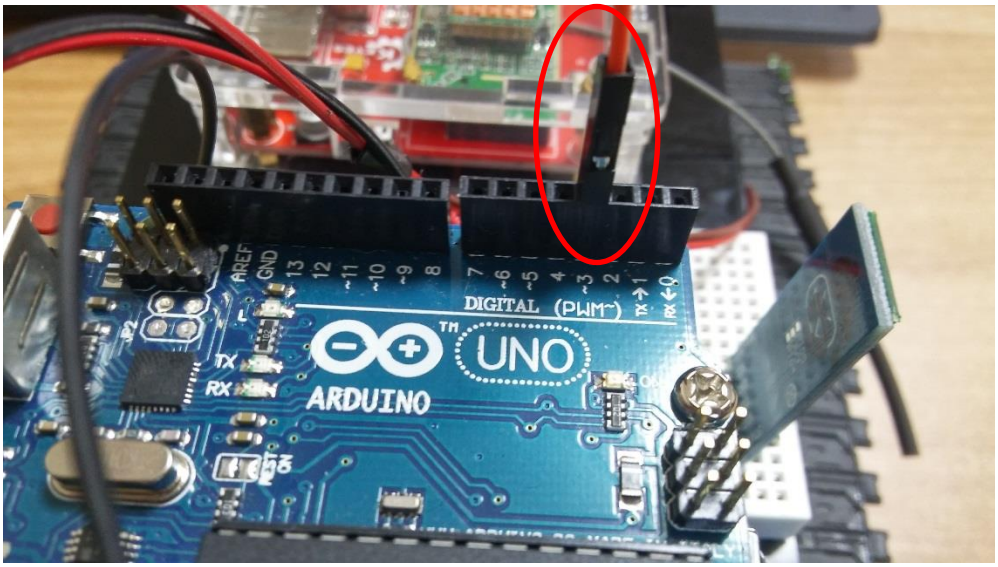


브레드 보드에 Male 케이블을 연결하는 사진

*브레드 보드의 구멍에 튀어나온 단자를 넣어줍니다.

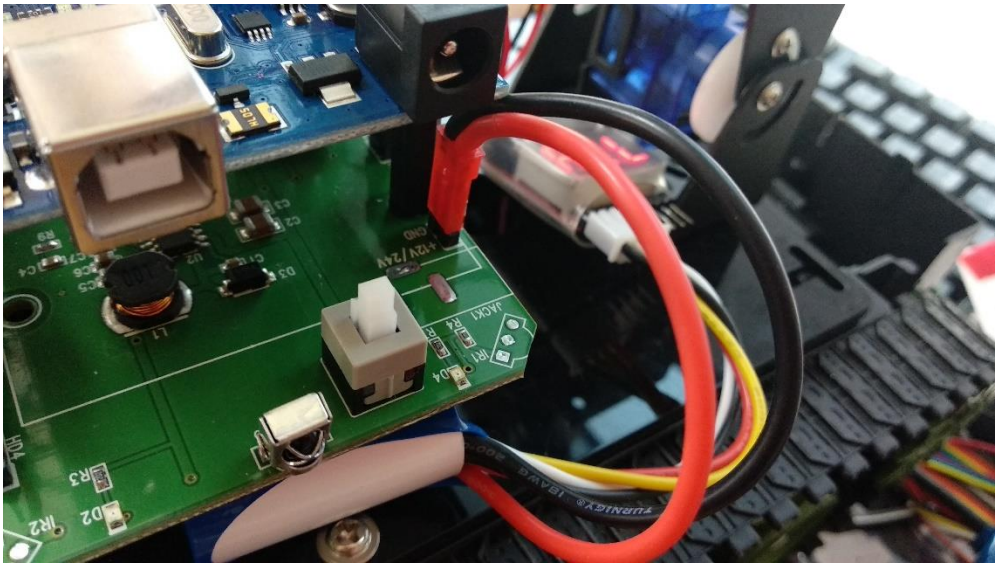


브레드 보드에 연결된 Male 케이블의 모습

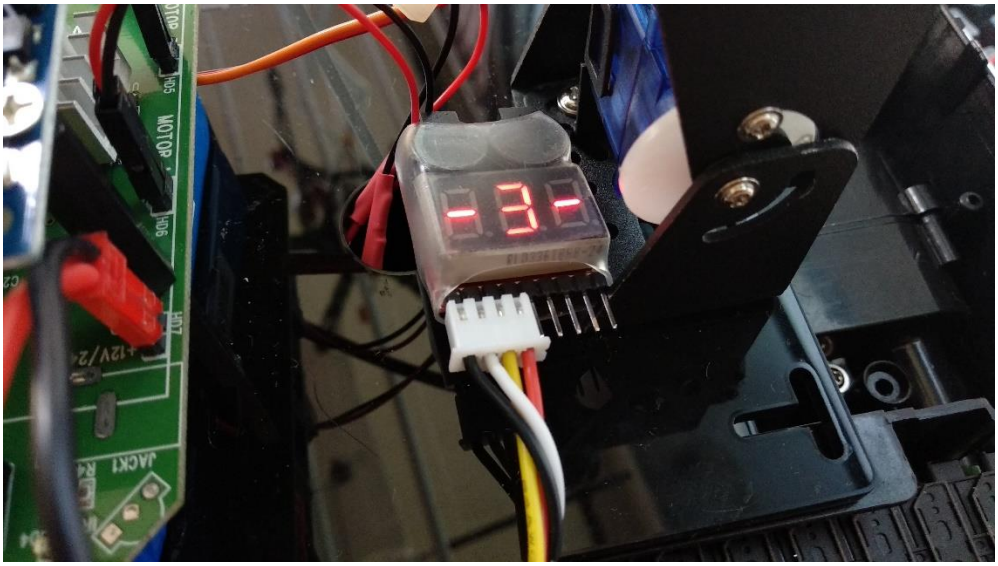


아두이노 우노 R3 에 Male 케이블을 연결하는 모습

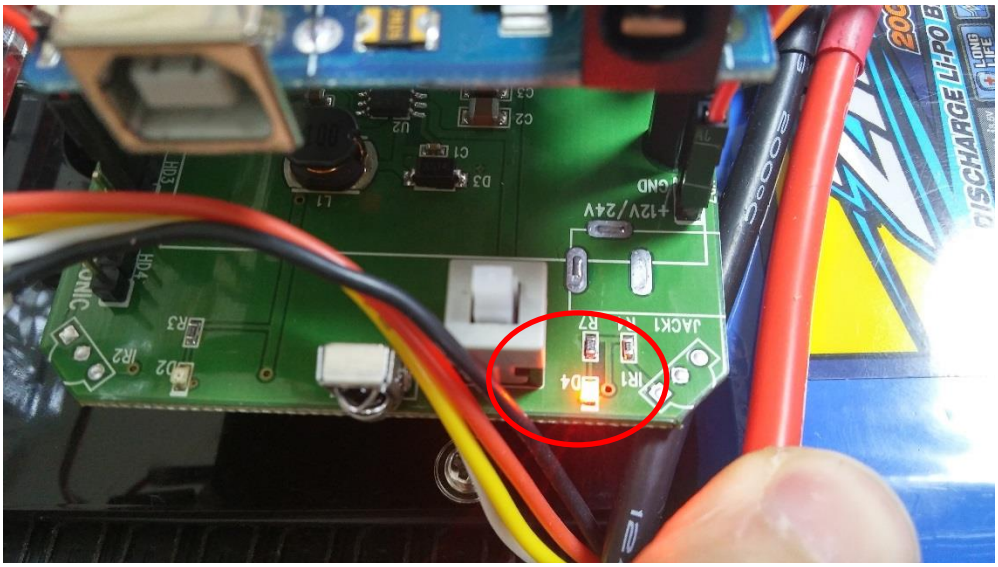
*R3 의 단자에 케이블의 튀어나온 단자를 넣어줍니다.



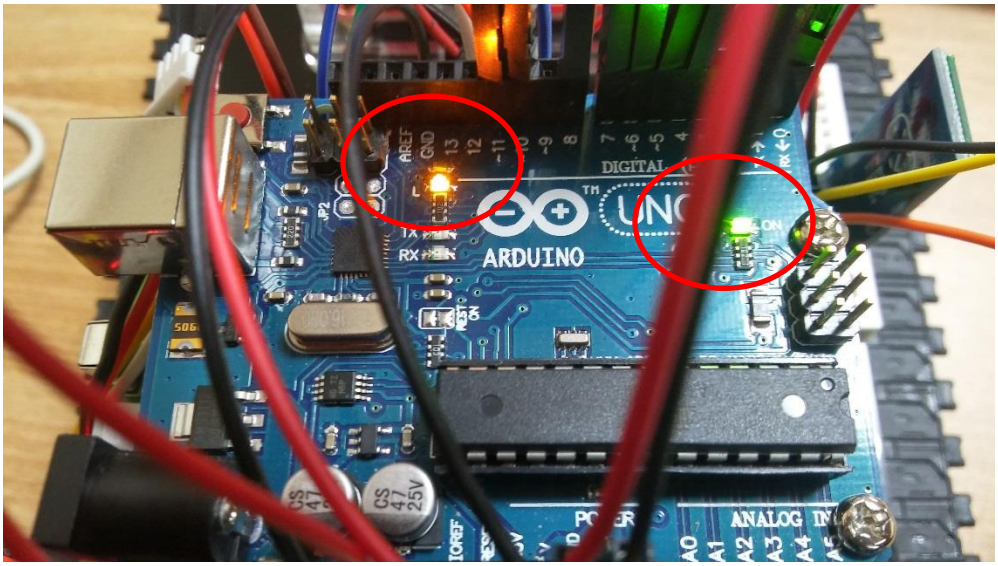
배터리의 전원선과 제어보드의 전원 단자를 연결한 모습



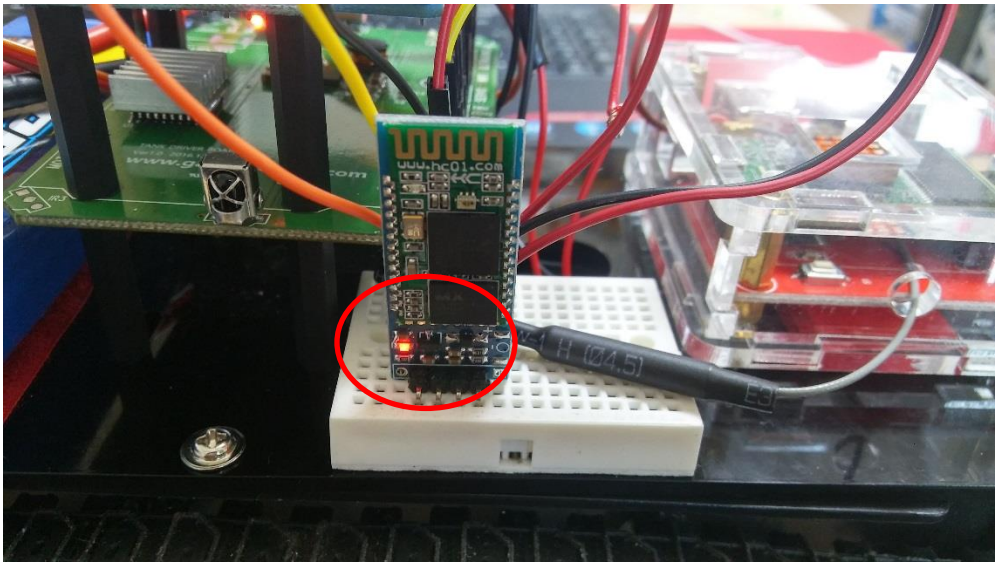
배터리와 셀체커를 연결한 모습



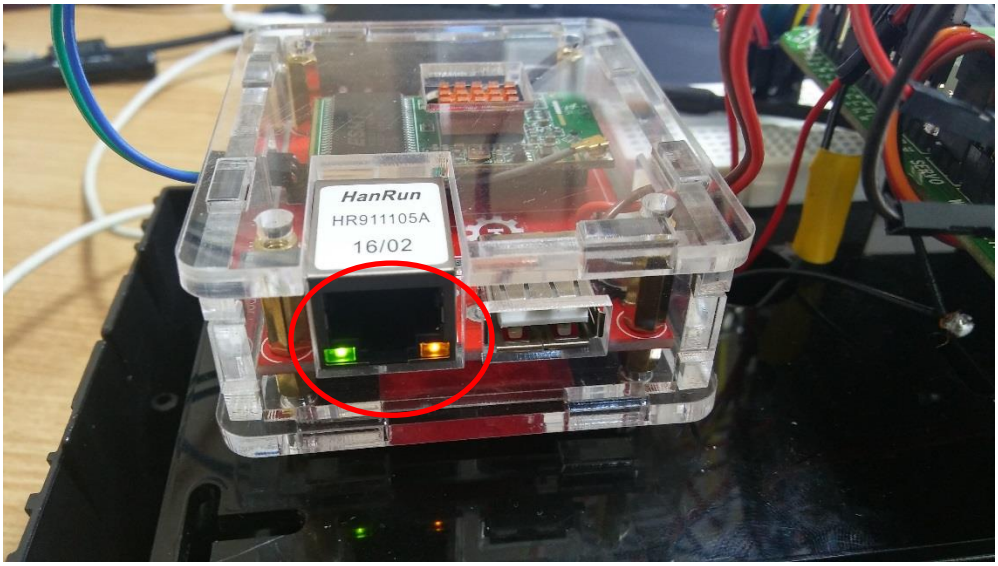
전원을 연결해주면(스위치를 누르면) 제어보드에서는 위와 같은 붉은빛이 들어옵니다.



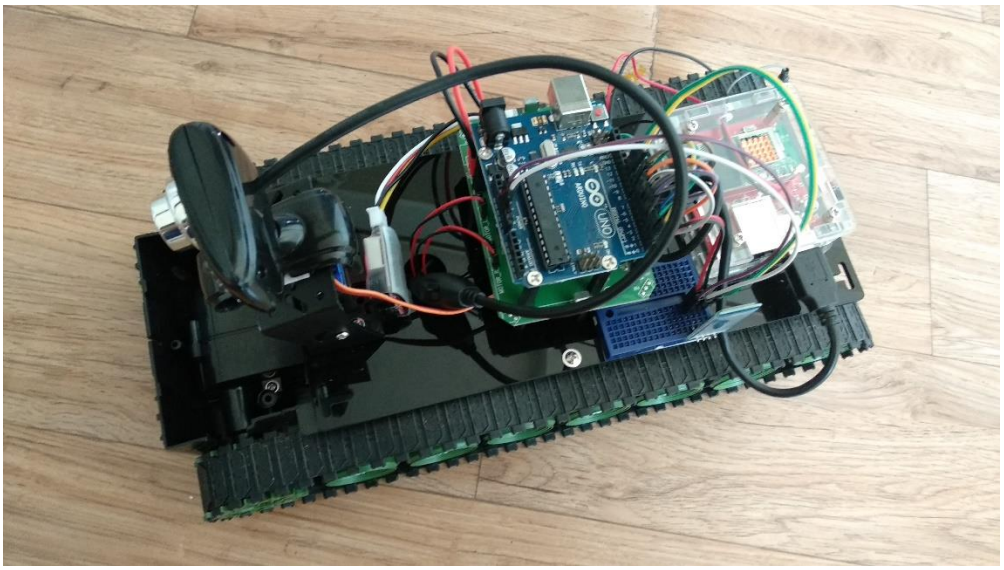
전원이 들어온 R3 의 모습(녹색불과 빨간불이 들어옵니다)



블루투스 모듈에 전원이 들어오면 붉은빛이 깜빡입니다.



와이파이 모듈에 불이 들어온 모습



조립이 완성된 모습

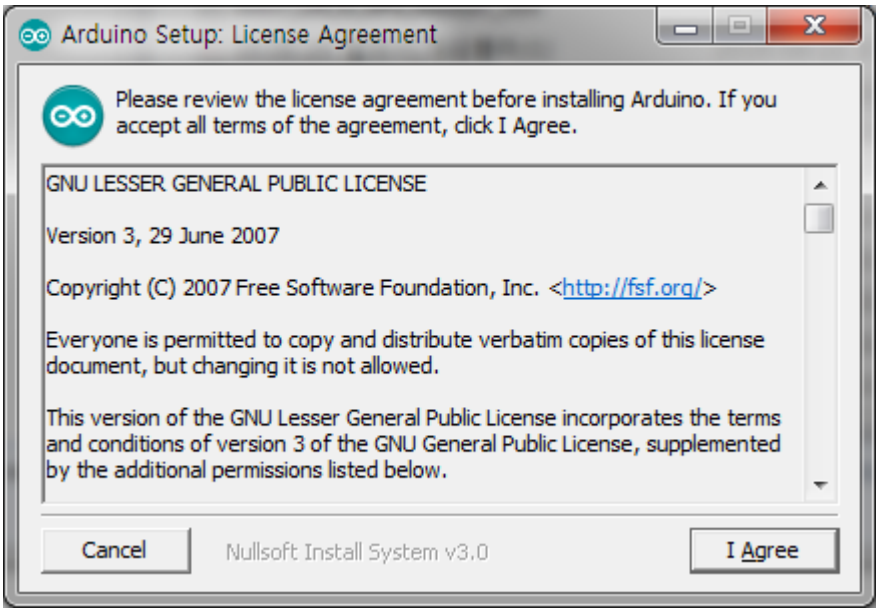
4.3 주행

실제 주행에 필요한 펌웨어와 스마트폰 앱으로 탱크를 테스트합니다.

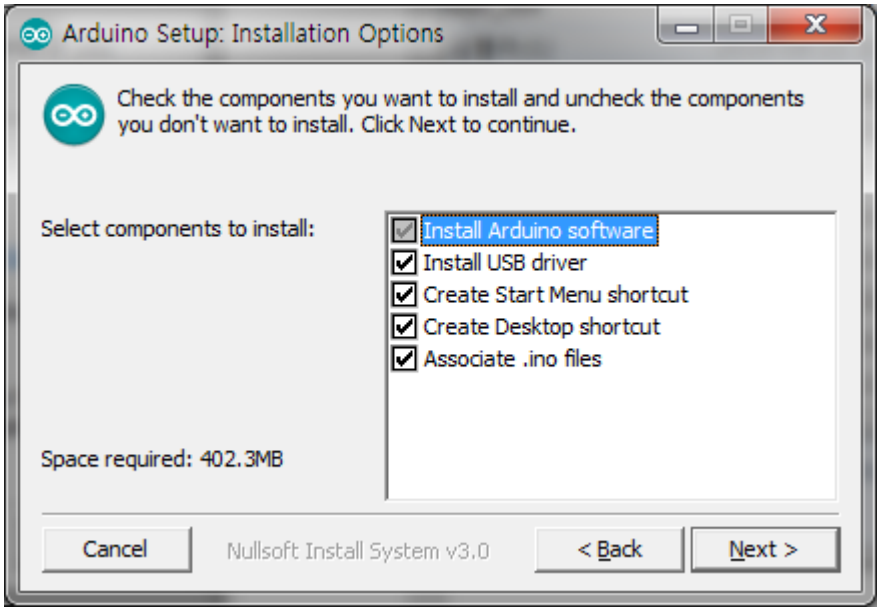
4.3.1 아두이노 소프트웨어(IDE) 설치

아래 주소로 접속하여 아두이노 소프트웨어를 다운로드합니다.

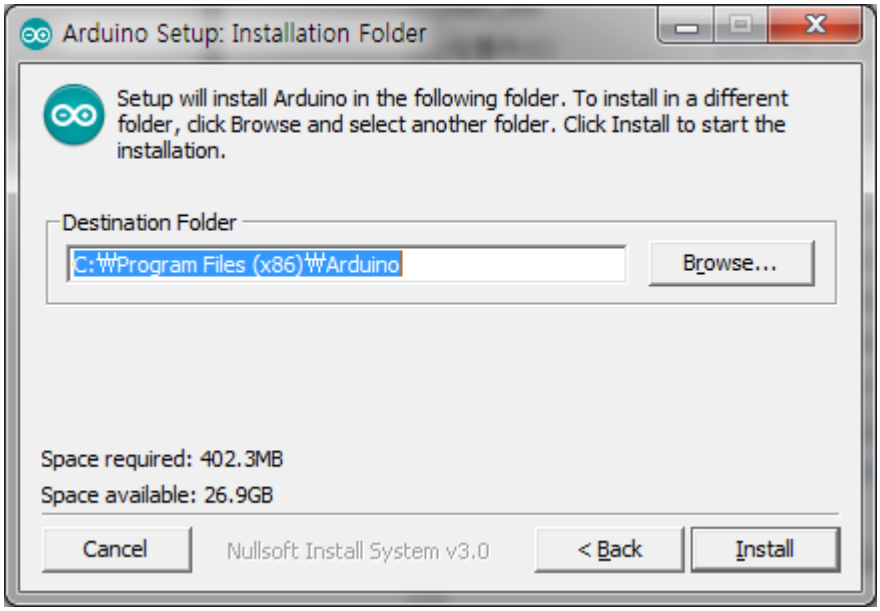
1. https://www.arduino.cc/en/Main/Software 접속
2. Windows Installer 클릭하여 다운로드



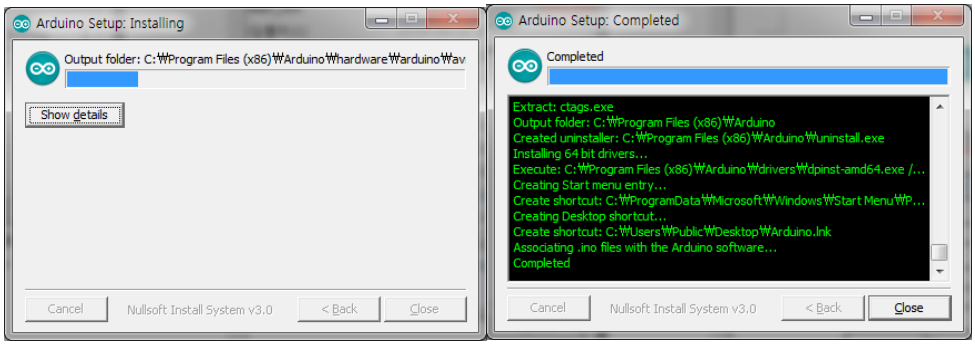
I Agree 를 클릭합니다



Next 를 클릭합니다

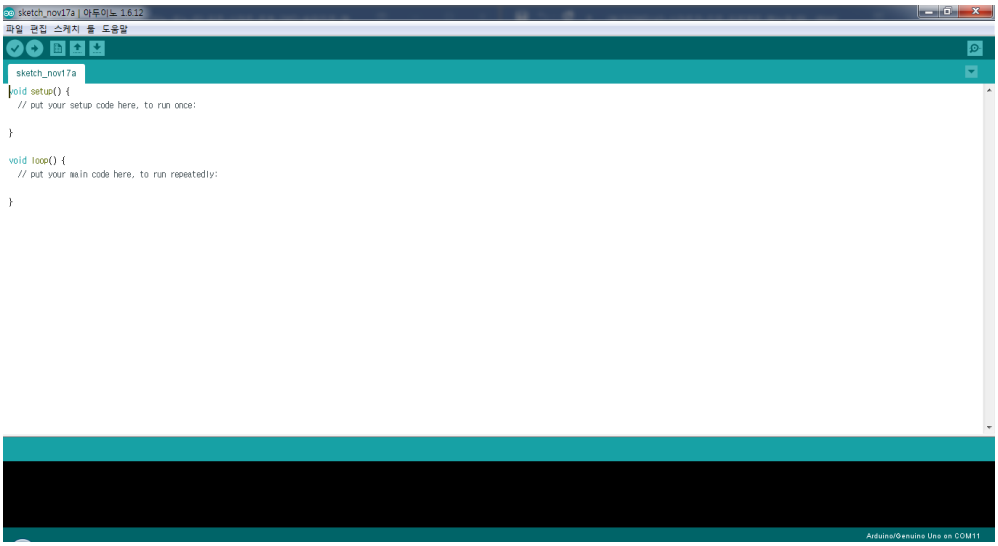


Install 을 클릭하여 설치를 시작합니다.



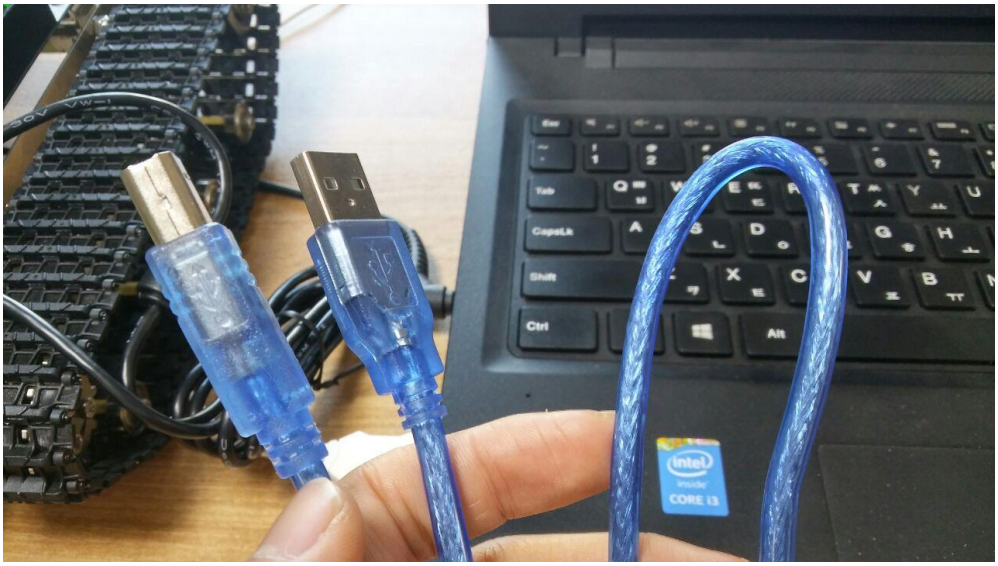
설치완료

Arduino 아이콘을 클릭하여 프로그램을 실행합니다

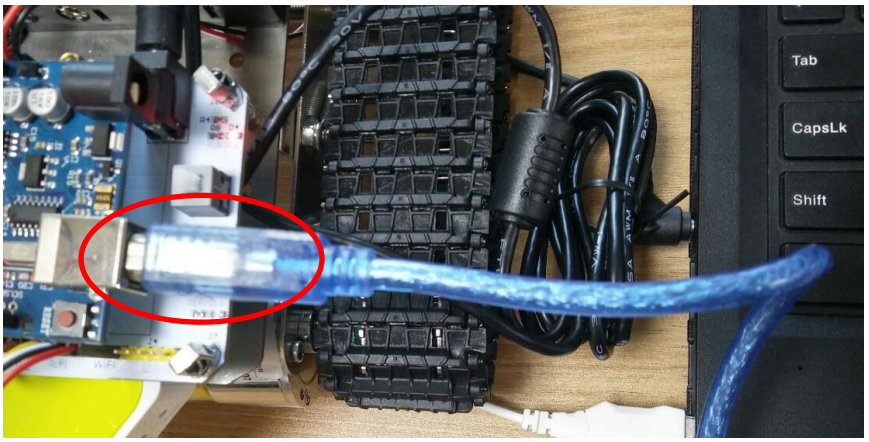


실행화면

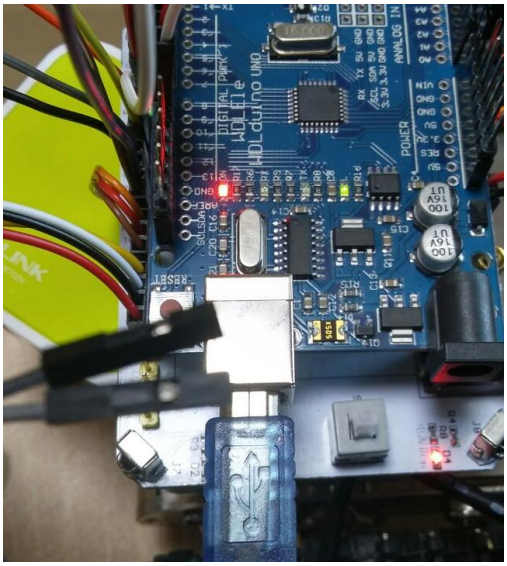
4.3.2 아두이노 IDE 연결



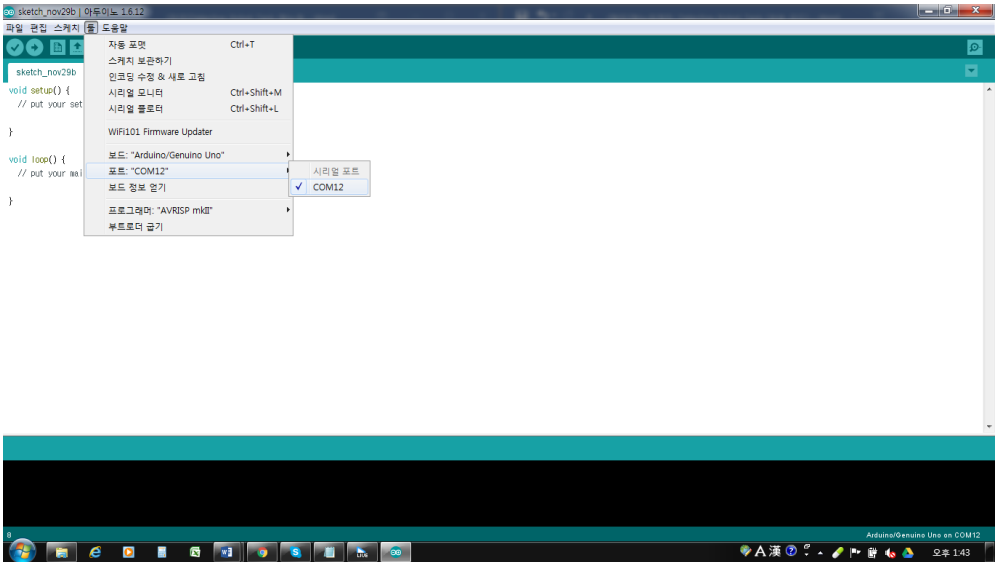
동봉된 USB 케이블



컴퓨터의 USB 단자와 아두이노 보드의 USB 단자를 연결합니다.



USB 케이블을 통한 업로드를 하기 위해 아두이노 우노 0 번&1 번 핀 케이블을 뽑아줍니다.



정상적으로 연결이 되었으면 IDE 에서 연결된 탱크를 확인할 수 있습니다.

*경우에 따라 장치 설치에 시간이 오래 걸릴 수 있습니다.

(연결 선택 및 확인 : 툴 -> 포트 -> COM XX 체크)

4.3.3 와이파이 주행 코드

1. 다운로드 받은 통합 소스코드 파일의 압축을 해제합니다.
2. 해제한 폴더 안의 arduino_robot_tank_wifi.ino 파일을 아두이노 IDE 로 열고 탱크에 업로드합니다.

```
int Left_motor_P=2;
int Left_motor_N=4;

int Right_motor_P=7;
int Right_motor_N=6;

int IntSpeed = 200;

#define Max_deg   60
#define Min_deg   5
#define INI_deg   25

int servopin=5;
int myangle;
int pulsewidth;
int val;
unsigned char Continue_flag = 0;
unsigned char i=0;
unsigned char temp=0;
unsigned char deg = INI_deg;

int test0=0,test1=0,test2=0,test3=0,pre_test2=0;
bool BoolIsWifiConnected = false;

void Fast_TurnL(int g)
{
   digitalWrite(Right_motor_P,HIGH);
```

```
  digitalWrite(Right_motor_N,LOW);
  analogWrite(Right_motor_P,IntSpeed);
  analogWrite(Right_motor_N,0);

  digitalWrite(Left_motor_P,LOW);
  digitalWrite(Left_motor_N,HIGH);
  analogWrite(Left_motor_P,0);
  analogWrite(Left_motor_N,IntSpeed);
}

void Fast_TurnR(int a)
{
  digitalWrite(Right_motor_P,LOW);
  digitalWrite(Right_motor_N,HIGH);
  analogWrite(Right_motor_P,0);
  analogWrite(Right_motor_N,IntSpeed);

  digitalWrite(Left_motor_P,HIGH);
  digitalWrite(Left_motor_N,LOW);
  analogWrite(Left_motor_P,IntSpeed);
  analogWrite(Left_motor_N,0);
}

void advance(int d)
{
  digitalWrite(Right_motor_P,HIGH);
  digitalWrite(Right_motor_N,LOW);
  analogWrite(Right_motor_P,IntSpeed);
  analogWrite(Right_motor_N,0);

  digitalWrite(Left_motor_P,HIGH);
  digitalWrite(Left_motor_N,LOW);
  analogWrite(Left_motor_P,IntSpeed);
```

```
   analogWrite(Left_motor_N,0);
}

void Reverse(int e)
{
   digitalWrite(Right_motor_P,LOW);
   digitalWrite(Right_motor_N,HIGH);
   analogWrite(Right_motor_P,0);
   analogWrite(Right_motor_N,IntSpeed);

   digitalWrite(Left_motor_P,LOW);
   digitalWrite(Left_motor_N,HIGH);
   analogWrite(Left_motor_P,0);
   analogWrite(Left_motor_N,IntSpeed);
}

void Brake(int f)
{
   digitalWrite(Right_motor_P,LOW);
   digitalWrite(Right_motor_N,LOW);
   digitalWrite(Left_motor_P,LOW);
   digitalWrite(Left_motor_N,LOW);
}

void servopulse(int servopin,int myangle)
{
   pulsewidth=(myangle*11)+500;
   digitalWrite(servopin,HIGH);
   delayMicroseconds(pulsewidth);
   digitalWrite(servopin,LOW);
   delay(20-pulsewidth/1000);
}
```

```
void Set_deg(unsigned char deg_in)
{
   for(int i=0;i<=25;i++) {
      servopulse(servopin,deg_in);
   }
}

void Set_deg_fast(unsigned char deg_in)
{
   for(int i=0;i<=2;i++) {
      servopulse(servopin,deg_in);
   }
}

void Action(unsigned char ctrl_val)
{
   switch(ctrl_val)
   {
      case 'L': Fast_TurnL(1);break;
      case 'R': Fast_TurnR(1);break;
      case 'F': advance(1);break;
      case 'B': Reverse(1);break;
      case 'S': Brake(1);break;
      //servo
      case 'r': if(deg>=Min_deg) {deg -= 5; Set_deg_fast(deg);} Continue_flag =
1;break;
      case 'm': deg=INI_deg;Set_deg(deg);break;
      case 'l': if(deg<=Max_deg) {deg += 5; Set_deg_fast(deg);}Continue_flag =
2;break;
      case 's': Continue_flag = 0;break;
      default : break;
   }
}
```

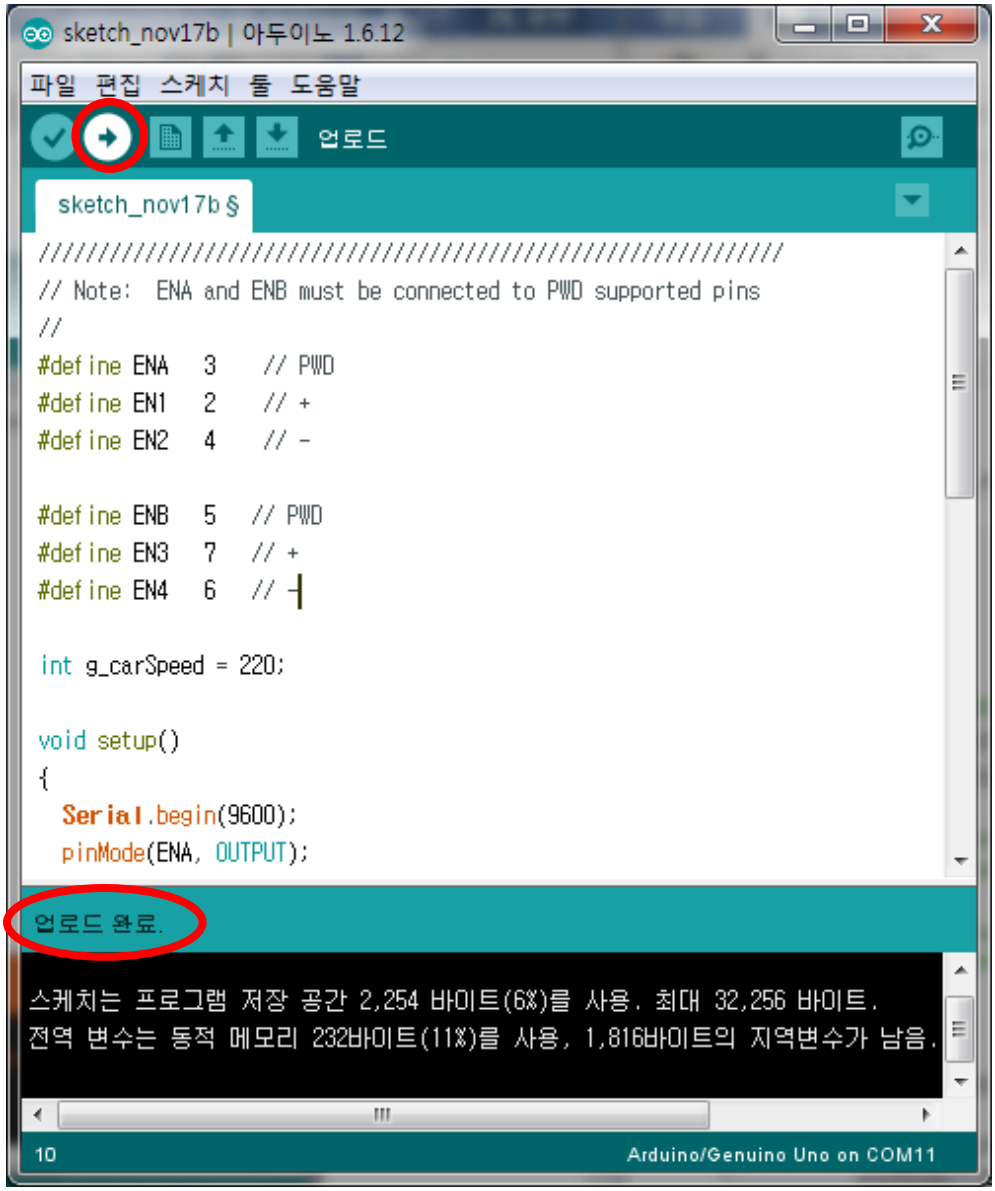
```
void Judge(void)
{
  if(Serial.available()>0)
  {
    if(BoolIsWifiConnected) {
      // 와이파이 앱이랑 연결되었을 때
      test1 = Serial.read();
      if((test0 == test1)||(test1 == 'S')||(test1 == 's'))
      {
        Action(test1);
      }
        else
      {
        test0 = test1;
      }
    }
    else {
      // 와이파이 앱이랑 연결되지 않았을 때
      test2 = Serial.read();
      if(test2 == '$') {
        if(pre_test2 == '$') {
          BoolIsWifiConnected = true;
        }
        else {
          pre_test2 = test2;
        }
      }
    }
  }

void setup()
```

```
{
   pinMode(Left_motor_P,OUTPUT);
   pinMode(Left_motor_N,OUTPUT);
   pinMode(Right_motor_P,OUTPUT);
   pinMode(Right_motor_N,OUTPUT);

   pinMode(servopin,OUTPUT);
   pinMode(13,OUTPUT);

   Set_deg(INI_deg);
   Brake(1);

   for(i=0;i<4 ;i++)
   {
      digitalWrite(13, LOW);
      delay(500);
      digitalWrite(13, HIGH);
      delay(500);
   }
   Serial.begin(9600);
   for(i=0;i<30 ;i++)
   {
      digitalWrite(13, LOW);
      delay(500);
      digitalWrite(13, HIGH);
      delay(500);
   }
   Serial.println("WIFI TANK");
}

void loop()
{
   Judge();
```

```
    switch(Continue_flag)
    {
       case 0:break;
       case 1:if(deg>=Min_deg) deg -= 5; Set_deg_fast(deg);break;
       case 2:if(deg<=Max_deg) deg += 5; Set_deg_fast(deg);break;
       default: break;
    }
}
```

4.3.4 코드 업로드, 시리얼 모니터

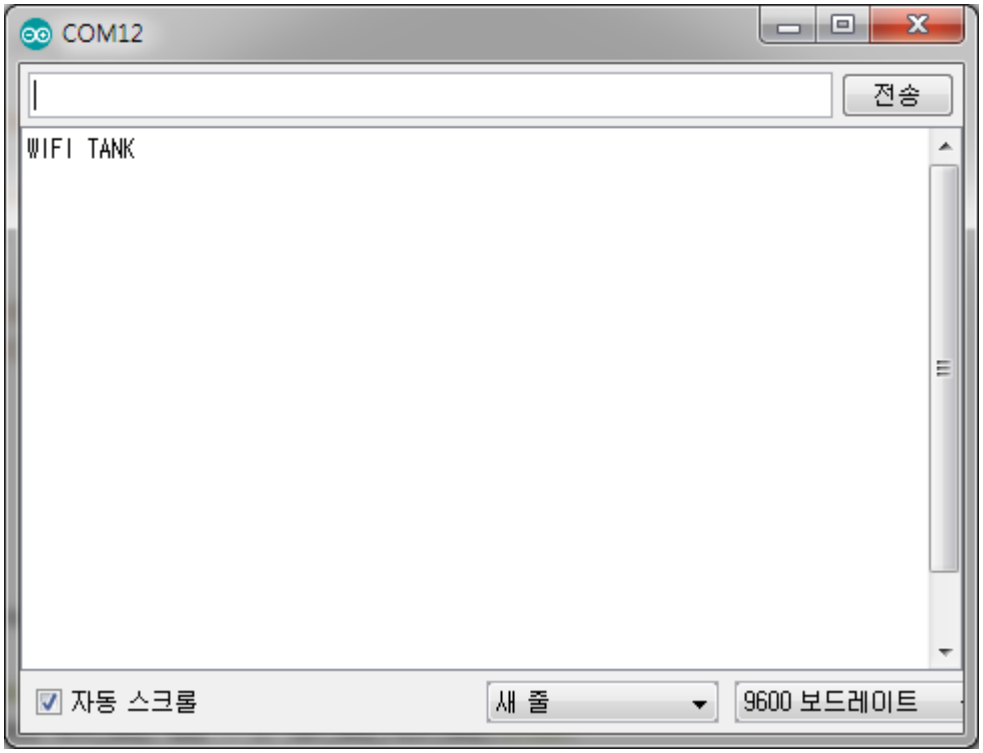
아두이노 IDE 에서 [파일] -> [새 파일] 후, 앞의 코드를 복사하여 붙여넣습니다. [파일] -> [저장] (Ctrl + s) 을 눌러 저장합니다. 그리고 코드를 업로드합니다.



[편집] 아래에 있는 화살표(업로드 버튼)을 눌러 탱크에 소스코드를 업로드 합니다(USB 가 연결되어 있어야 합니다)

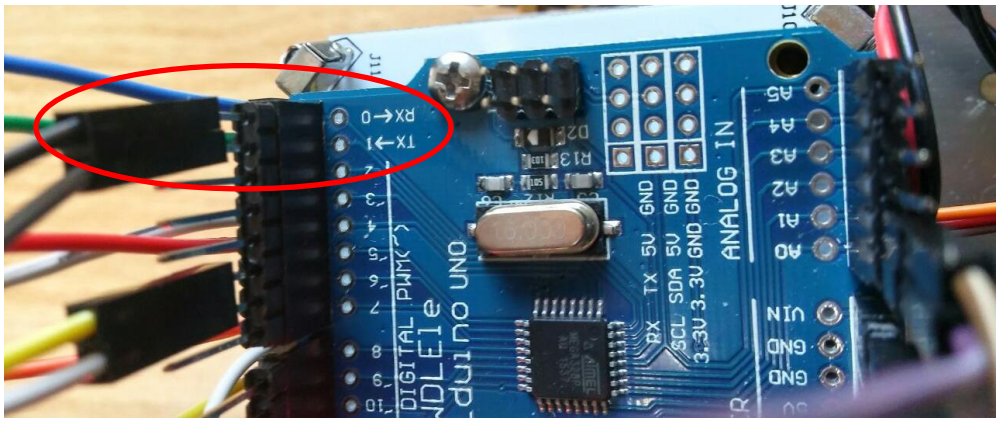
*위 그림의 소스코드 내용은 신경쓰지 않아도 됩니다.

오류없이 업로드가 되었다면 [툴] -> [시리얼 모니터] 를 통해 확인합니다.



업로드 완료 30 초 후 시리얼 모니터 창(위와 같이 뜬다면 정상입니다)

USB 케이블을 뽑아주고, 배터리의 전원을 연결해줍니다



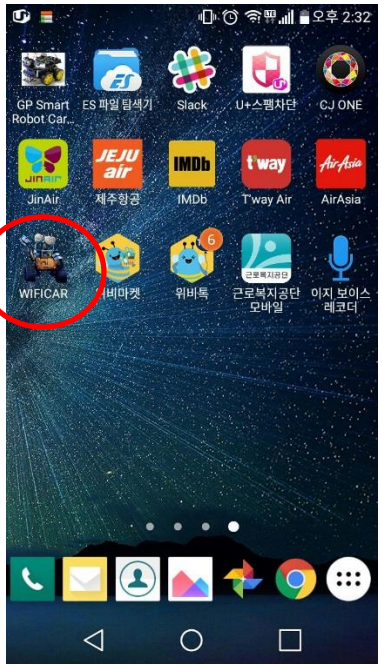
WIFI 통신을 위해 아두이노 우노의 0 번&1 번 핀을 다시 연결해줍니다.

4.3.5 스마트폰 앱과 와이파이 연동

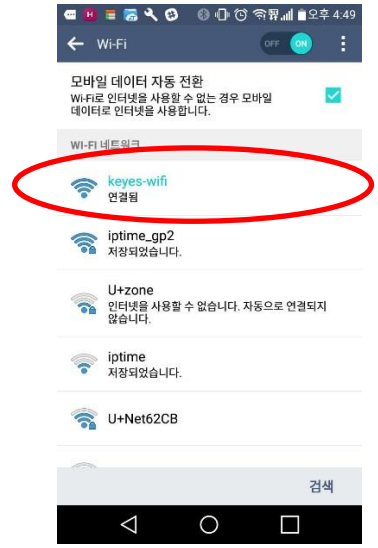
스마트폰으로 아래 링크에 접속하여 apk 파일을 다운로드 받아 앱을 설치합니다.

http://www.gameplusedu.com/gpshop/Arduino-Smart-Tank-KIT.apk

앱을 설치하면 아래와 같은 아이콘이 보입니다.



스마트폰에 나타는 앱 아이콘



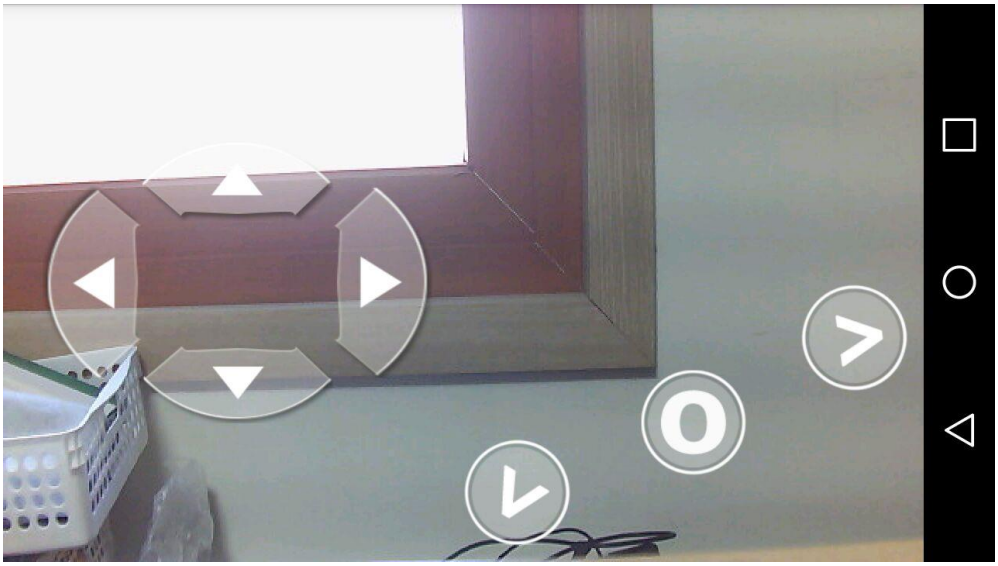
코드 업로드 후 30 초 정도가 지나면, 13 번 LED 의 깜빡임이 멈추고 계속 불이 들어와있는 상태가 되는데, 이 때 스마트폰의 WIFI 를 켜고, 위와 같이 와이파이 모듈에 연결합니다.

앱을 실행한 후 Settings 를 누릅니다.

셋팅 메뉴에서 "지원주소", "지원포트","영상주소" 를 위와 같이 수정하고 확인을 누른 후 앱의 시작화면으로 이동합니다.

앱을 실행합니다. 위와 같은 초기화면이 나타납니다.

"WIFI Tank" 버튼을 누릅니다.



위와 같은 화면이 나타납니다. 버튼을 활용하여 탱크를 조종할 수 있습니다.

4.3.6 와이파이 주행

(아두이노 보드 13 번 LED 의 깜빡임이 멈춘 후)

스마트폰 앱 조작에 따라 움직입니다.

4.3.7 리모트 컨트롤 라이브러리

IRremote.h 라는 라이브러리 헤더 파일 명칭이 아두이노의 로봇 IR 라이브러리와 중복되어 컴파일 오류가 나타나는 경우, 이를 해결하기 위해 아두이노 공개 소스 IR 라이브러리를 사용합니다.

1. 다운로드 받은 통합 소스코드 파일의 압축을 해제합니다.
2. 해제한 폴더 안의 arduino_robot_tank_rc.ino 파일을 아두이노 IDE 로 열고 탱크에 업로드합니다.

4.3.8 리모트 컨트롤 주행 코드

```
#include <stdlib.h>

#include "IRremote.h"
#include "TranslateIR.h"

int RECV_PIN = 8;
IRrecv irrecv(RECV_PIN);
decode_results results;

int Left_motor_P=2;
int Left_motor_N=4;

int Right_motor_P=7;
int Right_motor_N=6;

int IntSpeed = 200;

#define Max_deg   60
#define Min_deg   5
#define INI_deg   25
```

```
int servopin=5;
int myangle;
int pulsewidth;
int val;
unsigned char Continue_flag = 0;
unsigned char i=0;
unsigned char k=0;
unsigned char temp=0;
unsigned char deg = INI_deg;

int test0=0;
int test1=0;

void Fast_TurnL(int g)
{
   digitalWrite(Right_motor_P,HIGH);
   digitalWrite(Right_motor_N,LOW);
   analogWrite(Right_motor_P,IntSpeed);
   analogWrite(Right_motor_N,0);

   digitalWrite(Left_motor_P,LOW);
   digitalWrite(Left_motor_N,HIGH);
   analogWrite(Left_motor_P,0);
   analogWrite(Left_motor_N,IntSpeed);
}

void Fast_TurnR(int a)
{
   digitalWrite(Right_motor_P,LOW);
   digitalWrite(Right_motor_N,HIGH);
   analogWrite(Right_motor_P,0);
   analogWrite(Right_motor_N,IntSpeed);
```

```
  digitalWrite(Left_motor_P,HIGH);
  digitalWrite(Left_motor_N,LOW);
  analogWrite(Left_motor_P,IntSpeed);
  analogWrite(Left_motor_N,0);
}

void advance(int d)
{
  digitalWrite(Right_motor_P,HIGH);
  digitalWrite(Right_motor_N,LOW);
  analogWrite(Right_motor_P,IntSpeed);
  analogWrite(Right_motor_N,0);

  digitalWrite(Left_motor_P,HIGH);
  digitalWrite(Left_motor_N,LOW);
  analogWrite(Left_motor_P,IntSpeed);
  analogWrite(Left_motor_N,0);
}

void Reverse(int e)
{
  digitalWrite(Right_motor_P,LOW);
  digitalWrite(Right_motor_N,HIGH);
  analogWrite(Right_motor_P,0);
  analogWrite(Right_motor_N,IntSpeed);

  digitalWrite(Left_motor_P,LOW);
  digitalWrite(Left_motor_N,HIGH);
  analogWrite(Left_motor_P,0);
  analogWrite(Left_motor_N,IntSpeed);
}
```

```
void Brake(int f)
{
   digitalWrite(Right_motor_P,LOW);
   digitalWrite(Right_motor_N,LOW);
   digitalWrite(Left_motor_P,LOW);
   digitalWrite(Left_motor_N,LOW);
}

void servopulse(int servopin,int myangle)
{
   pulsewidth=(myangle*11)+500;
   digitalWrite(servopin,HIGH);
   delayMicroseconds(pulsewidth);
   digitalWrite(servopin,LOW);
   delay(20-pulsewidth/1000);
}

void Set_deg(unsigned char deg_in)
{
   for(int i=0;i<=25;i++) {
      servopulse(servopin,deg_in);
   }
}

void Set_deg_fast(unsigned char deg_in)
{
   for(int i=0;i<=2;i++) {
      servopulse(servopin,deg_in);
   }
}

void Action(char ctrl_val)
{
```

```
  switch(ctrl_val)
  {
    //move
    case 'L': Fast_TurnL(1);break;
    case 'R': Fast_TurnR(1);break;
    case 'F': advance(1);break;
    case 'B': Reverse(1);break;
    case '1': break;
    case '2': break;
    case '3': break;
    case '4': break;
    case 'S': Brake(1);break;
    //servo
    case 'r': if(deg>=Min_deg) {deg -= 5; Set_deg_fast(deg);} Continue_flag =
1;break;
    case 'm': deg=INI_deg;Set_deg(deg);break;
    case 'l': if(deg<=Max_deg) {deg += 5; Set_deg_fast(deg);}Continue_flag =
2;break;
    case 's': Continue_flag = 0;break;
    default: break;
  }
}

void translateIR() {
  for(k=0;k<sizeof(IR_RECEIVE_VALUE[CURRENT_IR_TYPE])/2;k++) {
    String temp = IR_RECEIVE_VALUE[CURRENT_IR_TYPE][k];
    if(temp.equalsIgnoreCase(String(results.value,HEX))) {
      test1 = *IR_RECEIVE_TRANSLATE[k];
    }
  }
}

void Judge(void)
```

```
{
   if(irrecv.decode(&results)) {
      if((test0 == test1)||(test1 == 'S')||(test1 == 's')) {
         translateIR();
         Action(test1);
         } else {
         test0 = test1;
      }
   irrecv.resume(); // receive the next value
   } else {
   }
}

void setup()
{
   pinMode(Left_motor_P,OUTPUT);
   pinMode(Left_motor_N,OUTPUT);
   pinMode(Right_motor_P,OUTPUT);
   pinMode(Right_motor_N,OUTPUT);

   pinMode(servopin,OUTPUT);

   Set_deg(INI_deg);
   Brake(1);

   for(i=0;i<4 ;i++)
   {
      digitalWrite(13, LOW);
      delay(500);
      digitalWrite(13, HIGH);
      delay(500);
   }
   Serial.begin(9600);
```

```
   Serial.print("REMOTE CONTROL TANK");
   irrecv.enableIRIn(); // Start the receiver
}

void loop()
{
      Judge();
}
```

4.3.9 리모트 컨트롤 주행

1. 앞의 코드를 USB 케이블을 이용하여 탱크에 업로드합니다(아두이노 우노의 0 번&1 번 핀을 뽑은 채 업로드합니다)

2. 업로드가 완료되었다면, USB 케이블을 제거합니다

3. 동봉된 리모트 컨트롤 버튼을 눌러 탱크를 조종합니다. 아두이노 소스를 수정하여 버튼에 해당하는 동작을 바꿀 수 있습니다.

동봉된 리모트 컨트롤. 앞의 소스 코드에 설정된 기본 값은 다음과 같습니다.

버튼	기능
CH	전진
|<<	좌회전
>||	우회전
+	후진
>>|	정지
0	카메라 각도 내리기
100+	카메라 각도 원위치

200+	카메라 각도 올리기

4.3.10 블루투스 주행 코드

1. 다운로드 받은 통합 소스코드 파일의 압축을 해제합니다.
2. 해제한 폴더 안의 arduino_robot_tank_bluetooth.ino 파일을 아두이노 IDE 로 열고 탱크에 업로드합니다.

```
#include <SoftwareSerial.h>
//////////////////////
//   <BT>          <UNO>
//    TX <-----> RX
//    RX <-----> TX
//////////////////////
SoftwareSerial btSerial(10, 11);   // RX, TX(UNO)

////////////////////////////////////////////////////////////
// Note:   ENA and ENB must be connected to PWD supported pins
//
#define Left_motor_P     2     // MOTOR_L+
#define Left_motor_N     4     // MOTOR_L-

#define Right_motor_P     7      // MOTOR_R+
#define Right_motor_N     6      // MOTOR_R-

//////////////////////
// Car direction
//
#define CAR_DIR_FW   0     // forward
#define CAR_DIR_BK   1     // backward
#define CAR_DIR_LT   2     // left turn
#define CAR_DIR_RT   3     // right turn
```

```
#define CAR_DIR_ST   4     // stop

/////////////////////////////////
// Default direction and speed
//
int g_carDirection = CAR_DIR_ST;
int g_carSpeed = 230; //

/////////////////////////////////////////////
// Note : confirm HIGH/LOW for correct movement
//
void car_forward()
{
   digitalWrite(Right_motor_P,HIGH);
   digitalWrite(Right_motor_N,LOW);
   analogWrite(Right_motor_P,255);
   analogWrite(Right_motor_N,0);

   digitalWrite(Left_motor_P,HIGH);
   digitalWrite(Left_motor_N,LOW);
   analogWrite(Left_motor_P,255);
   analogWrite(Left_motor_N,0);
}

void car_backward()
{
   digitalWrite(Right_motor_P,LOW);
   digitalWrite(Right_motor_N,HIGH);
   analogWrite(Right_motor_P,0);
   analogWrite(Right_motor_N,255);

   digitalWrite(Left_motor_P,LOW);
   digitalWrite(Left_motor_N,HIGH);
```

```
  analogWrite(Left_motor_P,0);
  analogWrite(Left_motor_N,255);
}

void car_right()
{
  digitalWrite(Right_motor_P,LOW);
  digitalWrite(Right_motor_N,HIGH);
  analogWrite(Right_motor_P,0);
  analogWrite(Right_motor_N,255);

  digitalWrite(Left_motor_P,HIGH);
  digitalWrite(Left_motor_N,LOW);
  analogWrite(Left_motor_P,255);
  analogWrite(Left_motor_N,0);
}

void car_left()
{
  digitalWrite(Right_motor_P,HIGH);
  digitalWrite(Right_motor_N,LOW);
  analogWrite(Right_motor_P,255);
  analogWrite(Right_motor_N,0);

  digitalWrite(Left_motor_P,LOW);
  digitalWrite(Left_motor_N,HIGH);
  analogWrite(Left_motor_P,0);
  analogWrite(Left_motor_N,255);
}

void car_stop()
{
  digitalWrite(Right_motor_P,LOW);
```

```
  digitalWrite(Right_motor_N,LOW);
  digitalWrite(Left_motor_P,LOW);
  digitalWrite(Left_motor_N,LOW);
}

////////////////////////////
// Execute car moving
//
void update_Car()
{
  Serial.println(g_carDirection);
  switch ( g_carDirection ) {
    case CAR_DIR_FW:
        car_forward();
        break;
    case CAR_DIR_BK:
        car_backward();
        break;
    case CAR_DIR_LT:
        car_left();
        break;
    case CAR_DIR_RT:
        car_right();
        break;
    case CAR_DIR_ST:
        car_stop();
        break;
    default :
        ;
  }
  return;
}
```

```
////////////////////////////////////
//   Class - Serial Protocol
//
class _CommProtocol
{
private:
   unsigned char protocolPool[28];
   int bufPoint;

public:
   _CommProtocol()
   {
   }

   void addPool(unsigned char cByte)
   {
      if (bufPoint < 28)
      {
         if (bufPoint == 0 and cByte != 0x0c)
            return;   // invalid code

         protocolPool[bufPoint++]=cByte;
      }
   }

   void clearPool()
   {
      bufPoint = 0;
      memset(protocolPool, 0x00, 28);
      Serial.println("clearPool");
   }

   bool isValidPool()
```

```
{
  if (bufPoint >= 28)
  {
    if (protocolPool[0] == 0x0c && protocolPool[14] == 0x0c)
    {
      return true;
    }
    else
    {
      clearPool();
      Serial.println("isValidPool 28 OVER");
    }
  }
  return false;
}

unsigned char getMotorLValue()
{
  unsigned char szProto[14];
  memcpy(szProto, protocolPool, 14);
  if (szProto[0] == 0x0C &&
    szProto[1] == 0x00 &&
    szProto[2] == 0x80 &&
    szProto[3] == 0x04 &&
    szProto[4] == 0x02)
  {
    unsigned char l = szProto[5];// -0x32;
    return l;
  }
  return 0x00;
}

unsigned char getMotorRValue()
```

```
  {
    unsigned char szProto[14];
    memcpy(szProto, &protocolPool[14], 14);
    if (szProto[0] == 0x0C &&
      szProto[1] == 0x00 &&
      szProto[2] == 0x80 &&
      szProto[3] == 0x04 &&
      szProto[4] == 0x01)
    {
      unsigned char l = szProto[5];// -0x32;
      return l;
    }
    return 0x00;
  }
};     // class(_CommProtocol)

////////////////////////////////////////////////
// Hint : Command codes come from keypad numbers
//
void controlByCommand(char doCommand)
{
  switch ( doCommand ) {
    case '+' :      // speed up
      g_carSpeed += 20;
      g_carSpeed = min(g_carSpeed, 255);
      break;
    case '-' :      // speed down
      g_carSpeed -= 20;
      g_carSpeed = max(g_carSpeed, 75);
      break;
    case '2' :      // forward
      g_carDirection = CAR_DIR_FW;
      break;
```

```
      case '5' :      // stop
        g_carDirection = CAR_DIR_ST;
        break;
      case '8' :      // backward
        g_carDirection = CAR_DIR_BK;
        break;
      case '4' :      // left
        g_carDirection = CAR_DIR_LT;
        break;
      case '6' :      // right
        g_carDirection = CAR_DIR_RT;
        break;
      default   :
        ;
    }
    return;
}

////////////////////////////////////////////////
// Create an instance of class(_CommProtocol)
//
_CommProtocol SerialCommData;

////////////////////////////////////////////////////////////
// Parse the serial input value and convert to MOVE command
//
void process_SerialCommModule()
{
    if (SerialCommData.isValidPool())
    {
      char motorLR[2];

      motorLR[0] = (char)SerialCommData.getMotorLValue();
```

```
    motorLR[1] = (char)SerialCommData.getMotorRValue();
    SerialCommData.clearPool();

    //
    Serial.print("Left [");
    Serial.print(motorLR[0],DEC);
    Serial.print("] Right [");
    Serial.print(motorLR[1],DEC);
    Serial.println("]");
    //

    char szCmdValue = '5';
    // set MOVE commands
    if (motorLR[0] == 0 && motorLR[1] == 0) {   // (0,0) stop
       szCmdValue = '5';
    }
    else
    {
       int nSpeed;
       nSpeed = max(abs(motorLR[0]), abs(motorLR[1]));

       // Set direction
       if (motorLR[0] > 0 && motorLR[1] > 0)     // (+,+) forward
       {
          szCmdValue = '2';
          g_carSpeed = 255.0f * ((float)nSpeed / 100.0f);
       }
       else if (motorLR[0] < 0 && motorLR[1] < 0)     // (-,-) backward
       {
          szCmdValue = '8';
          g_carSpeed = 255.0f * ((float)nSpeed / 100.0f);
       }
       else if (motorLR[0] < 0 && motorLR[1] > 0)     // (-,+) left turn
```

```
        {
          szCmdValue = '4';
          g_carSpeed = 255.0f * ((float)((float)nSpeed*1.66f) / 100.0f);
        }
        else if (motorLR[0] > 0 && motorLR[1] < 0)     // (+,-) right turn
        {
          szCmdValue = '6';
          g_carSpeed = 255.0f * ((float)((float)nSpeed*1.66f) / 100.0f);
        }
      }
      //
      Serial.print("speed ");
      Serial.print(g_carSpeed);
      Serial.print(" ");
      Serial.println(szCmdValue);
      //

      // Set the direction and speed with command
      controlByCommand(szCmdValue);
    }
}

void setup()
{
   Serial.begin(9600);       // PC serial monitor debugging
   btSerial.begin(9600);     // bluetooth serial connection

   //init car control board
   pinMode(Left_motor_P, OUTPUT);
   pinMode(Left_motor_N, OUTPUT);

   pinMode(Right_motor_P, OUTPUT);
   pinMode(Right_motor_N, OUTPUT);
```

```
  Serial.print("direction value ");
  Serial.println(g_carDirection);
  Serial.print("speed pwm value ");
  Serial.print(g_carSpeed);
  Serial.println("");
  //
}

void loop()
{

  if (btSerial.available()) {

    unsigned char cByte;

    cByte = btSerial.read();

    SerialCommData.addPool(cByte);   // store the serial input to Buffer

    process_SerialCommModule();       // parse and change the input value to
MOVE command

    update_Car();                          // execute car MOVE

  }

}
long microsecondsToCentimeters(long microseconds)
{
  return microseconds/29/2;
}
```

4.3.11 스마트폰 앱과 블루투스 연동

스마트폰을 이용하여 블루투스 모듈(HC-06)과 페어링합니다. 핀번호는 1234 입니다.

4.3.12 스마트폰 앱과 블루투스 연동

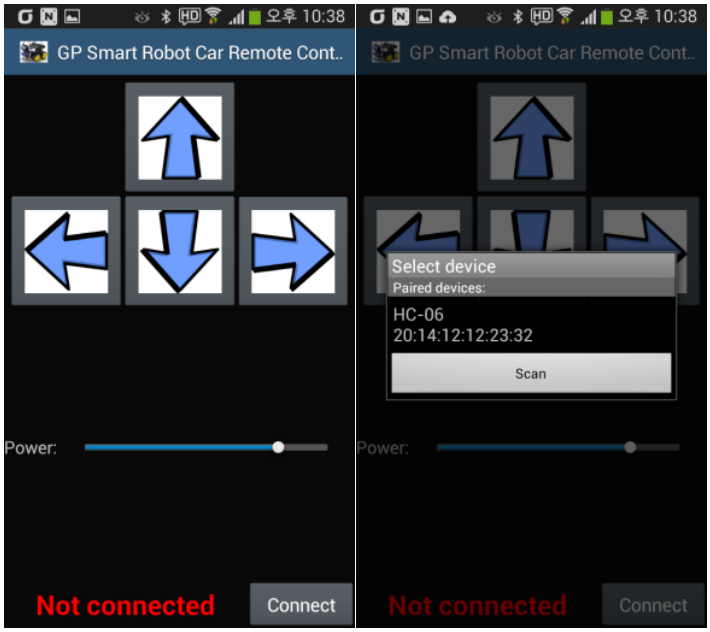
앞의 블루투스 제어 코드를 자동차에 업로드한 후 스마트폰으로 아래 링크로 접속하여 apk 파일을 다운로드 받아 앱을 설치합니다.

http://www.gameplusedu.com/pds/gpshop/arduino/robotics/kit_example/car_4wd/android_apk/GPSmartCarRemoteManager.apk

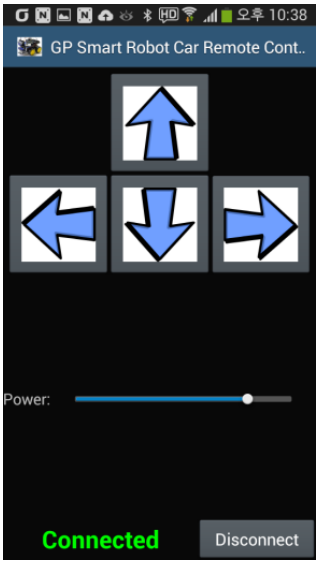
앱을 설치하면 아래와 같은 아이콘이 보입니다.

앱을 실행합니다

앱을 실행하면 아래와 같이 블루투스 모듈을 선택해 연결할 수 있고, 화살표 패드를 이용하여 자동차를 조작할 수 있습니다



4.3.13 블루투스 주행



정상적인 연결 화면

화살표 버튼을 누르게 되면 자동차는 화살표 방향으로 이동합니다. 버튼 터치 상태가 아닌 경우에는 정지 상태로 변경됩니다.

5 DC 모터

탱크를 운행하기 위해서는 모터 회전 및 제어를 해야 합니다. DC 모터만 제대로 동작시키면 탱크 관련 로봇은 쉽게 제작할 수 있습니다.

5.1 DC 모터 설명

DC 모터는 **(+)(-)** 연결해주면 회전하는 부품입니다. DC 모터는 **(+)(-)** 연결 극성 변경, 전압 조절에 의해 회전방향 및 속도를 변경할 수 있습니다.

Rated Voltage	DC 24V
Rated Current	0.33A
Rated Torque	3.12kg.cm
Output Speed	800RPM
Output Shaft Diameter	6mm / 5/16"
Gearbox Diameter	37mm / 1 1/2"
Total Length(Without Shaft)	57m / 2 3/16
Material	Metal, Electronic Parts

[참고용]

전압(V)의 크기에 따라 회전 속도의 변경이 가능합니다. 회전 속도 변경으로 운행속도 조절이 가능합니다. 물론 배터리와 DC 모터의 최대 RPM 을 기준으로 최고 속도이고 그 이하로 속도 조절이 가능합니다.

5.2 DC 모터 전원 연결 케이블 연결 주의 사항

DC 모터의 2 개 전원 연결선은 극성을 구분하지 않습니다. 무조건 가동 전압 충족 되면 돌아가는 부품입니다. 전원 연결선의 (+)(-)의 방향에 따라 회전 방향이 반전됩니다. 키트에 포함된 DC 모터의 입고 버전에 따라 검은색, 빨간색 전원 연결선의 위치가 다른 경우가 종종 있습니다.

그리고 DC 모터 연결된 검은색/빨간색(또는 검은색/흰색) 전선의 납땜 연결이 끊어질 수 있습니다. 약간의 주의가 필요합니다. 만약 끊어진 경우 인두기가 있다면 다시 연결하여 사용하면 됩니다. 또는 적절한 조치(절연 테이프로 임시로 붙여서 묶어주는 식)를 취하시기 바랍니다.

5.3 DC 모터 리드선 연결 시 리드선 주의 사항

부품 입고 순서 및 사정에 의해 DC 모터 리드선의 색상이 다를 경우에는 납땜이 가능하다면 전선을 분리, 일괄적으로 배선 순서를 다시 납땜해도 무방합니다. 그냥 사용하는 경우에는 DC 모터 전원 연결 배선할 때 리드선의 연결된 위치를 염두에 두시고 배선하기 바랍니다.

6 서보 모터

이 키트에서는 0 도에서 180 도까지 각도를 정밀하게 제어할 수 있는 장점을 가진 서보 모터를 이용하여 카메라의 각도를 조정하고 있습니다. 서보 모터에 대해 간단히 알아보도록 하겠습니다.

6.1 개요

"서보(Servo)" 란 "서보 메커니즘(Servo Mechnism)" 의 줄임말로서 일본 공업 규격(JIS)에서는 "물체의 위치, 방위, 자세 등을 제어량으로 하고 목표치의 임의 변화에 추종하도록 구성된 제어계" 라고 정의되어 있습니다.

또한 서보(Servo)는 Servant(하인)라는 단어에 기인하였다고 하는데, 이는 "주인에 충실한" 이라는 의미를 가지고 있습니다. 즉 서보 모터란 "주인의 명령에 충실하게 동작하는 모터" 를 나타내며 동작이란 위치, 속도 및 가속도의 3 요소를 말하지만 실제로 서보 모터의 적용에 있어서 위치 제어를 위한 적용처와 속도 제어를 위한 적용처의 두 가지로 나타납니다. 위치 제어에는 속도를 제어하여 위치를 추종하게 되고 속도 제어에는 순간 가속도를 제어하여 속도를 추종하게 됩니다.

서보 모터는 일반 모터와는 달리 빈번하게 변화하는 위치나 속도의 명령치에 대하여 신속하고 정확하게 추종할 수 있도록 설계된 모터를 의미합니다. 서보 모터는 급가속 및 급제동에 대하여 대응할 수 있는 구조를 가지고 있어야 하며, 따라서 서모 모터는 다음과 같은 두 가지의 조건을 만족해야 합니다.

▶ 큰 회전력(토크, Torque)을 가질 것
▶ 회전자(Rotor)의 관성 모멘트가 작을 것

위의 두가지 조건을 만족하는 것이 DC 모터입니다. 이 때문에 DC 모터는 오랫동안 서보 모터로 사용되어 왔고 일반 모터보다 큰 토크를 가지고 회전자 관성을 줄이기 위하여 회전자가 가늘고 긴 구조를 가지며 영구자석으로 여자속을 만들어 크기를 줄이고 작으면서도 큰 토크를 만들 수 있도록 하고 있습니다.

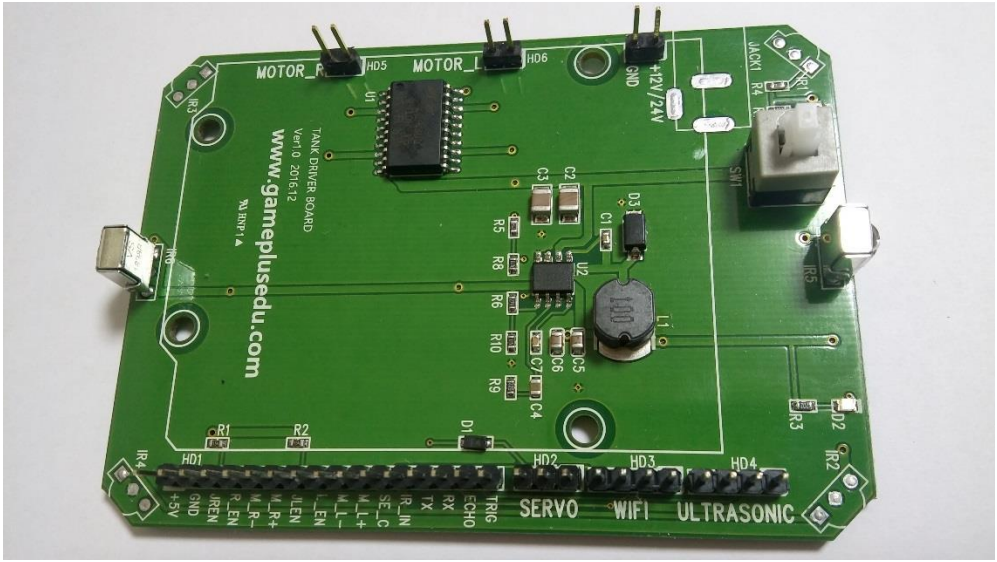
6.2 기술 사양

Size	23.2 × 12.5 × 22 mm
Weight	9 g
Digital	N
Free-run current @ 6V	200 mA
Stall current @ 6V	600 mA
Speed @ 6V	0.07 sec/60°2
Stall torque @ 6V	21 oz·in
Speed @ 4.8V	0.09 sec/60°3
Stall torque @ 4.8V	18 oz·in
Lead length	8 in

FEETECH FS90R Micro Continuous Rotation Servo 기술 사양

7 제어보드

DC 모터의 회전 및 속도 제어, 서보모터 제어, 와이파이 모듈 제어, 리모트 컨트롤 제어, 초음파 제어를 위해 사용합니다. 제어보드에 담겨있는 L293DD IC 는 표준 DTL 이나 TTL 로직 레벨을 받아들이며, DC 모터나 서보 모터를 제어하는데, 그리고 파워 트랜지스터를 스위칭하는데 사용됩니다. 이 키트에서 L293DD IC 는 DC 모터를 제어하는데 사용됩니다.



제어보드

L293DD IC

L293DD IC 는 8 개의 센터 핀을 가지고 있는 20 개의 납 표면에 구성되어 있는데, 이 납 표면들은 연결되어 있고 방열판으로 사용됩니다. 총 20 개의 제어 포트가 있으며, 회로의 내부 구성은 기본적인 H-Bridge 가 적용되어 있습니다.

L296DD IC

타입	H-Bridge
작동 공급 전압	4.5 V to 36 V
출력 전류	600 mA
작동 공급 전류	2 mA
최대 작동 온도	+ 150 C
장착 스타일	SMD/SMT

최소 작동 온도	- 40 C
출력 수	2
작동 온도 범위	- 40 C to + 150 C
단위 중량	266.700 mg

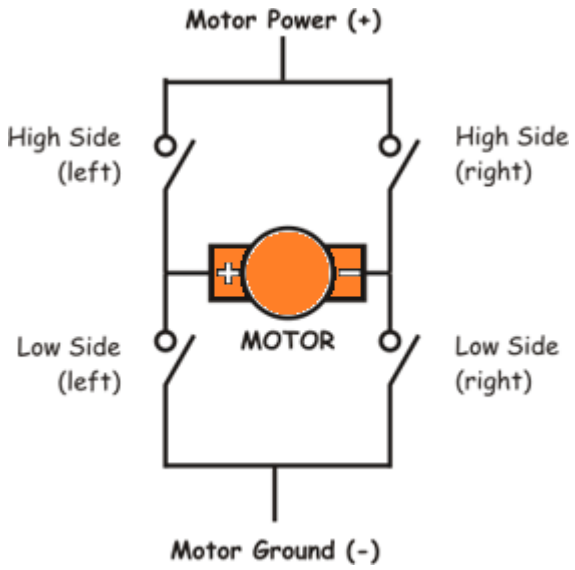
L293DD 기술 사양

IC(직접회로)란?

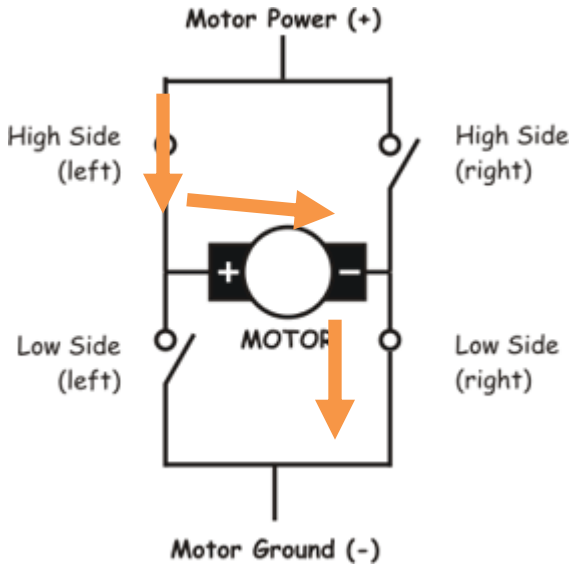
트랜지스터, 저항, 커패시터 등을 고밀도로 집적하여 회로 구성된 부품을 말합니다. 트랜지스터, 저항, 커패시터 등의 형태가 소규모, 소형화 형태로 된 것이 아닌 실리콘 기판 위에 인쇄 형태의 회로로 모두 집적되어 구성된 형태입니다.

7.1 H-BRIDGE 회로

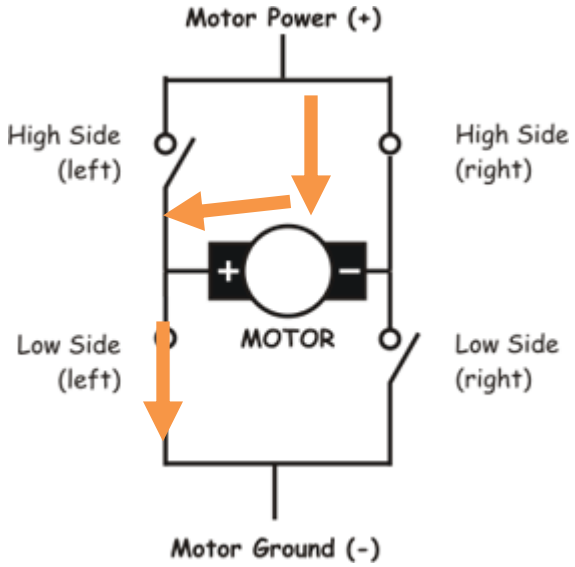
전류의 흐름을 정방향/역방향으로 변경해 주는 회로의 정식 명칭입니다. 알파벳 H 문자와 비슷합니다. DC 모터 연결 후 전류의 흐름을 바꾸어 주면 모터의 회전 방향이 바뀌게 됩니다.



H-Bridge 스위치의 중앙에는 DC 모터 같이 (+)(-) 극성으로 연결되어 있어야 합니다. 양쪽에 위치한 스위치의 열기/닫기 조합에 의해 전류의 흐름 방향이 반전 됩니다.

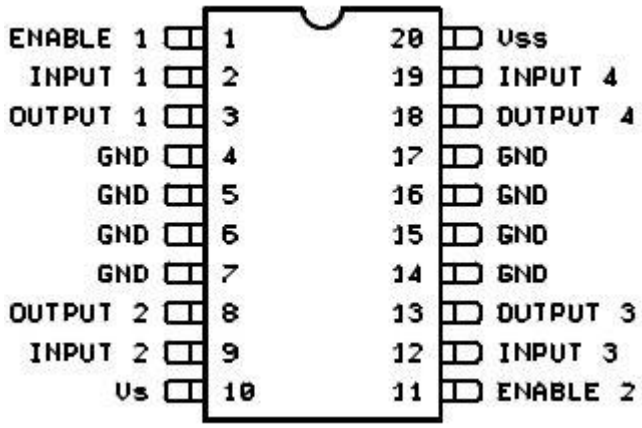


DC 모터 방향 변환 : 양극에서 음극으로 흐를 경우



DC 모터 방향 변환 : 음극에서 양극으로 흐를 경우

7.2 L293DD 포트



포트	설명
1	Enable 1 : HIGH 신호일 경우에는 왼쪽 부분의 IC 포트가 작동하도록 되어 있습니다. 반대로 LOW 신호일 경우에는 왼쪽 포트가 작동하지 않습니다. IC 포트 왼쪽 부분의 마스터 키 포트 역할을 담당하고 있습니다. INPUT1, OUTPUT1, INPUT2, OUTPUT2 사용 유무 포트입니다.
2	INPUT 1 : HIGH 신호가 입력 되면 OUTPUT 1 포트도 HIGH 신호가 됩니다
3	OUTPUT 1 : 모터의 리드선 연결 포트입니다.
4	GND
5	GND
6	GND
7	GND
8	OUTPUT 2 : 모터의 리드선 연결 포트입니다.
9	INPUT 2 : HIGH 신호가 입력 되면 OUTPUT 2 포트도 HIGH 신호가 됩니다
10	Vs : 모터 전원 공급 포트입니다. 모터에 맞게 적절한 전압 입력을 연결해 주어야 합니다.
20	Vss : IC 신호 제어 전원 입력 포트입니다. 5V 입력 해주도록 합니다. 용도에 따라 Vs 포트로 입력된 전원이 정제된 5V 출력 포트로도 사용되기도 합니다.
19	INPUT 4 : HIGH 신호가 입력 되면 OUTPUT 4 포트도 HIGH 신호가 됩니다.
18	OUTPUT 4 : 모터의 리드선 연결 포트입니다.

17	GND
16	GND
15	GND
14	GND
13	OUTPUT 3 : 모터의 리드선 연결 포트입니다.
12	INPUT 3 : HIGH 신호가 입력 되면 OUTPUT 3 포트도 HIGH 신호가 됩니다.
11	Enable 2 : HIGH 신호일 경우에는 오른쪽 부분의 IC 포트가 작동하도록 되어 있습니다. 반대로 LOW 신호일 경우에는 오른쪽 포트가 작동하지 않습니다. IC 포트의 오른쪽 부분의 마스터 키 포트 역할을 담당하고 있습니다. INPUT3, OUTPUT3, INPUT4, OUTPUT4 사용 유무 포트입니다.

Pin 1, 11	Pin 2, 12	Pin 9, 19	Function
High	Low	High	Turn clockwise
High	High	Low	Turn anti-clockwise
High	Low	Low	Stop
High	High	High	Stop
Low	Not applicable	Not applicable	Stop

DC 모터 회전 방향 설정

1 번 핀에 전압이 걸린 상태에 2 번 핀 LOW, 9 번 핀에 HIGH 설정할 경우 시계방향(clock wise)으로 회전합니다. 이처럼 각각의 핀에 High, Low 설정을 참조하여 DC 모터 회전 제어가 가능합니다. 11 번, 12 번, 19 번 핀도 동일하게 사용됩니다

7.3 제어보드 회로도

8 아두이노 보드

아두이노란?

아두이노는 오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러로 완성된 보드(상품)와 관련 개발 도구 및 환경을 말합니다. 2005 년 이탈리아에서 하드웨어에 익숙지 않은 학생들이 자신들의 디자인 작품을 손쉽게 제어할 수 있도록 하기 위해 고안된 아두이노는 처음에 AVR 을 기반으로 만들어졌으며, 아트멜 AVR 계열의 보드가 현재 가장 많이 판매되고 있습니다. ARM 계열의 Cortex-M0(Arduino M0 Pro)과 Cortex-M3(Arduino Due)를 이용한 제품도 존재합니다.

아두이노는 다수의 스위치나 센서로부터 값을 받아들여, LED 나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있습니다. 임베디드 시스템 중의 하나로 쉽게 개발할 수 있는 환경을 이용하여, 장치를 제어할 수 있습니다.

아두이노 통합 개발 환경(IDE)을 제공하며, 소프트웨어 개발과 실행코드 업로드도 제공합니다. 또한 어도비 플래시, 프로세싱, Max/MSP 와 같은 소프트웨어와 연동할 수 있습니다.

아두이노 우노

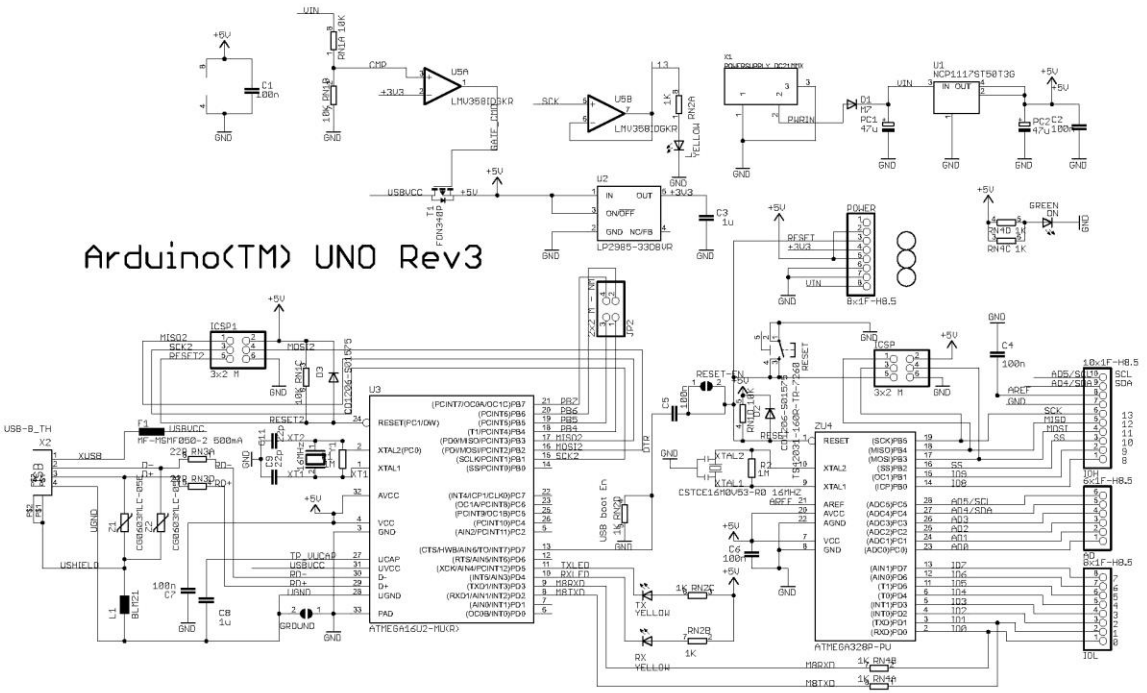
아두이노 우노는 ATmega328P 에 기반한 마이크로컨트롤러 보드입니다. 보드에는 14 개의 디지털 입력/출력 핀(이 중 6 개는 PWM 출력으로 사용될 수 있음), 6 개의 아날로그 입력, 16Mhz 석영 크리스탈, USB 연결, 파워 잭, ICSP 헤더와 리셋 버튼이 있습니다.

8.1 개요

아두이노 우노 R3 는 ATmega328P 에 기반한 마이크로컨트롤러 보드이며, 32KB 플래시 메모리 8 비트 마이크로컨트롤러와 2KB 램으로 구성되어 있습니다. USB 케이블로 컴퓨터와 연결할 수도 있고, AC-to-DC 어댑터나 배터리를 이용하여 전원을 공급할 수도 있습니다.

8.2 기술 사양

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB
Flash Memory for Bootloader	0.5 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz



8.3 아두이노 소프트웨어

오픈소스 아두이노 소프트웨어(IDE)는 코드 작성이 쉽고, 보드에 업로드하기도 용이합니다. Windows, Mac OS X, 그리고 Linux 에서 작동합니다. 환경은 자바로 써있고, 처리 소프트웨어와 다른 오픈소스 소프트웨어에 기반하고 있습니다. 또한 어떤 아두이노 보드와도 사용 가능합니다.

아두이노 소프트웨어는 코드 작성을 위한 텍스트 에디터, 메시지 영역, 텍스트 콘솔, 공통 기능과 메뉴 툴바로 이루어져 있습니다. 아두이노 하드웨어, Genuino 하드웨어와의 연결을 형성하고 프로그램을 업로드하고 통신이 가능하게 합니다.

스케치 작성

아두이노 소프트웨어를 이용하여 작성된 프로그램을 스케치(sketches)라고 부릅니다. 스케치는 텍스트 에디터에서 작성되고 .ino 라는 파일 확장자의 형태로 저장됩니다. 에디터에서는 자르기/붙여넣기 그리고 찾기/바꾸기가 가능합니다. 메시지 영역에서는 저장과 내보내기의 피드백이 가능하며 오류를 보여줍니다. 콘솔은 아두이노 소프트웨어가 출력한 텍스트(오류 메시지 등의 정보)를 보여줍니다. 우측 아래의 창에는 보드와 시리얼 포트가 표시됩니다. 툴바 버튼들을 통해 검증, 업로딩, 작성, 열기, 스케치 저장, 시리얼 모니터를 할 수 있습니다.

메뉴

[스케치(Sketch)]

- 확인/컴파일(Verify/Compile)
  - ■ 컴파일 과정에서의 오류를 확인합니다. 그리고 코드와 변수들이 사용하는 메모리 크기를 콘솔 영역에 보여줍니다.
- 업로드(Upload)
  - ■ 코드를 컴파일하고 포트를 이용하여 보드에 바이너리 파일을 업로드합니다.
- 프로그래머를 이용해 업로드(Upload Using Programmer)
  - ■ 보드의 부트로더에 덮어씁니다.
- 컴파일된 바이너리 보내기(Export Compiled Binary)
  - ■ 다른 툴을 이용하는 보드에 보내거나 아카이빙하기 위한 .hex 파일을 저장합니다.

- 스케치 폴더 보이기(Show Sketch Folder)
  - ■ 현재 스케치 폴더를 엽니다.
- 라이브러리 포함하기(Include Library)
  - ■ 코드 상단에 #include 문을 삽입하여 현재 스케치에 라이브러리를 추가합니다.
- 파일 추가...(Add File...)
  - ■ 스케치에 소스 파일을 추가합니다.

[툴(Tools)]

- 자동 포맷(Auto Format)
  - ■ 코드를 깔끔하게 정리합니다.
- 스케치 보관하기(Archive Sketch)
  - ■ .zip 형식으로 현재의 스케치를 아카이빙합니다. 아카이브는 현재 스케치와 같은 폴더에 생성됩니다.
- 인코딩 수정 & 새로고침(Fix Encoding & Reload)
  - ■ 에디터 문자표(char map)와 운영체제 문자표 사이의 불일치들을 수정합니다.
- 시리얼 모니터(Serial Monitor)
  - ■ 시리얼 모니터 창을 열고, 현재 선택된 포트에 연결된 보드와의 데이터 교환을 초기화합니다. 시리얼 포트 연결(opening)을 통한 리셋이 가능한 경우, 보드를 리셋(reset)합니다.
- 보드(Board)
  - ■ 이용하고 있는 보드를 선택합니다.
- 포트(Port)
  - ■ 포트를 선택합니다.
- 프로그래머(Programmer)
  - ■ 온보드 USB 시리얼 연결을 이용하지 않고 보드나 칩을 프로그래밍할 때, 하드웨어 프로그래머를 선택하기 위해 사용합니다. 보통은 이용하지 않지만, 새로운 마이크로컨트롤러에 부트로더를 구웠을 때 이 메뉴를 사용합니다.
- 부트로더 굽기(Burn Bootloader)
  - ■ 아두이노 보드 마이크로컨트롤러 부트로더를 구울 때 사용합니다. 보통은 필요하지 않지만, 새로운 ATmega 마이크로컨트롤러(부트로

더가 없는)를 구매했을 때 유용합니다. 부트로더를 굽기 전에 정확한 보드를 선택했는지를 확인하세요.

스케치북

아두이노 소프트웨어는 스케치들을 담아 놓는 곳인 스케치북이란 개념을 사용합니다. 스케치북의 스케치들은 [파일->스케치북]을 통해 열 수 있습니다. 처음 아두이노 소프트웨어를 실행했을 때는 자동으로 스체키북 폴더가 만들어집니다. 스케치북의 위치는 [파일->환경설정]에서 볼 수 있습니다.

시리얼 모니터

아두이노 보드가 보내오는 시리얼 데이터를 표시합니다. 보드로 데이터를 보내기 위해서는 텍스트를 입력하고 "전송" 버튼이나 엔터를 누르면 됩니다. 스케치의 Serial.begin 에 넘겨진 레이트와 일치하도록 보드 레이트를 선택할 수 있습니다. 윈도우, 맥, 리눅스에서는 시리얼 모니터를 통해 연결할 때 보드가 리셋됩니다.

9 스마트폰과 탱크의 와이파이 연동

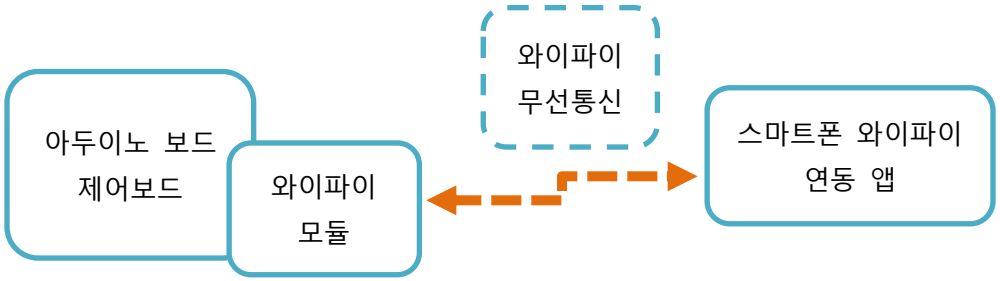
9.1 와이파이란

와이파이(Wi-Fi, WiFi)는 전자기기들이 무선랜(WLAN)에 연결할 수 있게 하는 기술로서, 주로 2.4 기가헤르츠(12 센티미터) UHF 및 5 기가헤르츠(6 센티미터) SHF ISM 무선 대역을 사용합니다. 무선랜은 일반적으로 암호로 보호되어 있지만, 대역 내에 위치한 어느 장치라도 무선랜 네트워크의 자원에 접근할 수 있도록 개방도 가능합니다.

와이파이 얼라이언스는 와이파이를 전기 전자 기술자 협회(IEEE) 802.11 표준에 기반한 모든 "무선 근거리 통신망"(WLAN) 제품으로 정의하고 있습니다. 와이파이 기술을 사용하는 장치에는 개인용 컴퓨터, 비디오 게임 콘솔, 스마트폰, 디지털 카메라, 태블릿 컴퓨터, 디지털 오디오 플레이어, 현대의 프린터가 포함됩니다.

와이파이 호환 장치들은 WLAN 네트워크와 무선 액세스 포인트를 통해 인터넷에 접속할 수 있습니다. 이러한 액세스 포인트(핫스팟)는 실내에서는 약 20 미터(66 피트)의 대역을, 실외에서는 이보다 더 큰 대역을 가집니다. 핫스팟 지원 범위는 무선파를 차단하는 벽이 있는 작은 방으로까지만 지원될 수 있고, 여러 액세스 포인트를 겹쳐 사용함으로써 수 제곱 킬로미터로까지 확대할 수 있습니다.

9.2 아두이노와 스마트폰의 와이파이 통신



와이파이 통신은 근거리 통신망에 사용되는 무선통신 기술입니다. 게다가 사용하기 편리하게 시리얼 통신이 가능한 현태로도 생산되므로, 시리얼 통신 가능한 MCU, PC 등의 대부분의 장치에서 사용할 수 있습니다. 시리얼 통신은 최소 송/수신 2 개의 포트만 있으면 사용가능한 구조입니다. 아두이노 보드도 안정적인 시리얼 통신이 가능한 기기이므로 와이파이 모듈을 사용하면, 안정적인 근거리 무선 시리얼 통신이 가능합니다.

9.3 와이파이 모듈

9.3.1 개요

이 키트에서는 탱크의 근거리 통신 제어를 담당할 부품으로 와이파이 모듈을 이용합니다.

와이파이 모듈

9.4 아두이노와 와이파이 모듈 연결

와이파이 모듈에 연결된 케이블 중 빨간선은 전원케이블이고 검은선(갈색선)은 접지 케이블입니다. 제어 보드와 와이파이 모듈로부터 나온 RX,TX 케이블이 연결되면, 제어 보드는 자신의 RX,TX 단자를 통해 아두이노 보드와 통신을 합니다. 물론 제어 보드의 RX,TX 단자와 아두이노 보드의 TX,RX 단자는 서로 케이블로 연결되어 있습니다. 와이파이 모듈은 스마트폰 앱으로부터 받은 값을 제어 보드를 통해 아두이노 보드로 전달합니다. 아두이노 보드는 수신된 값을 판단하여 제어 보드를 통해 DC 모터를 제어합니다.

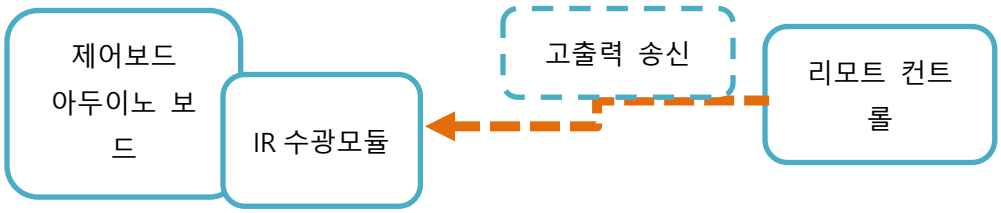
10 리모트 컨트롤과 탱크 연동

10.1 리모트 컨트롤이란

리모트 컨트롤 또는 원격 조정기(遠隔調整機)는 기계의 원격 운영에 쓰이는 전자 장치를 말합니다. 텔레비전, 라디오, 오디오 장비를 조정하는 데 주로 쓰이며 휴대용 오디오 기기를 조작하는 유선 장치도 리모컨이라고 불립니다. 1893 년 니콜라 테슬라가 리모컨의 첫 본보기 가운데 하나를 제공하였음을 테슬라 특허, 곧 미국 특허 613,809 에 서술되어 있습니다.

'리모컨' 이라는 낱말은 리모트 컨트롤(remote control)의 일본어식 줄임말입니다.

10.2 아두이노와 리모트 컨트롤의 통신



동봉된 리모트 컨트롤은 트랜지스터를 이용하여 고출력 송신을 이용합니다. 이렇게 송신된 신호는 제어보드에 장착되어 있는 IR 수광모듈에 감응하게 됩니다. IR 수광모듈은 다양한 파장의 빛에서 원하는 파장만 필터링할 수 있어, 리모트 컨트롤이 송신하는 파장을 수신하게 됩니다. 송신된 신호를 감지한 IR 수광모듈은 그 신호를 제어보드의 IR_IN 단자를 통해 아두이노 보드로 전달합니다(물론 제어보드의 IR_IN 단자와 아두이노 보드의 핀은 서로 연결되어 있어야 합니다). IR_IN 단자를 통해 신호를 전달받은 아두이노 보드는 프로그래밍에 기반하여 수신된 값을 기초로 DC 모터와 서보모터를 제어합니다.

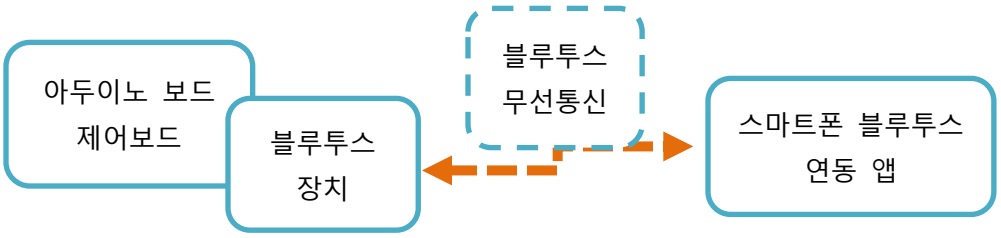
11 스마트폰과 탱크의 블루투스 연동

11.1 블루투스란

블루투스란 근거리 무선 통신 기술 규약을 의미합니다. 현재까지는 휴대용 기기, 근거리에서 사용되는 독립된 장치 등에 많이 사용됩니다. 블루투스의 출발은 휴대폰 제조사의 에릭슨, 노키아에서 개발된 만큼, 휴대용 기기에 많이 사용되고 있습니다. 블루투스의 출발은 소형 기기와의 통신을 위한 용도이므로 저전력을 사용하는 무선 통신 규약입니다. 스마트폰(휴대폰), 노트북, 태블릿 PC, 이어폰, 헤드폰, GPS 단말기, 키보드, 카메라, MP3 등에 사용됩니다.

블루투스의 사용 및 응용 범위는 계속 확대 추세이며, 현재의 블루투스는 4.0(~4.1)버전의 규약까지 나와있습니다. 블루투스 무선 통신 범위는 증가(50~100m)되면서 전력 소모는 줄어들어 더욱 더 효율적인 매체로 진보하는 중입니다. 차후, 다양한 형태의 블루투스 규약과 기기들이 등장할 것으로 예상됩니다.

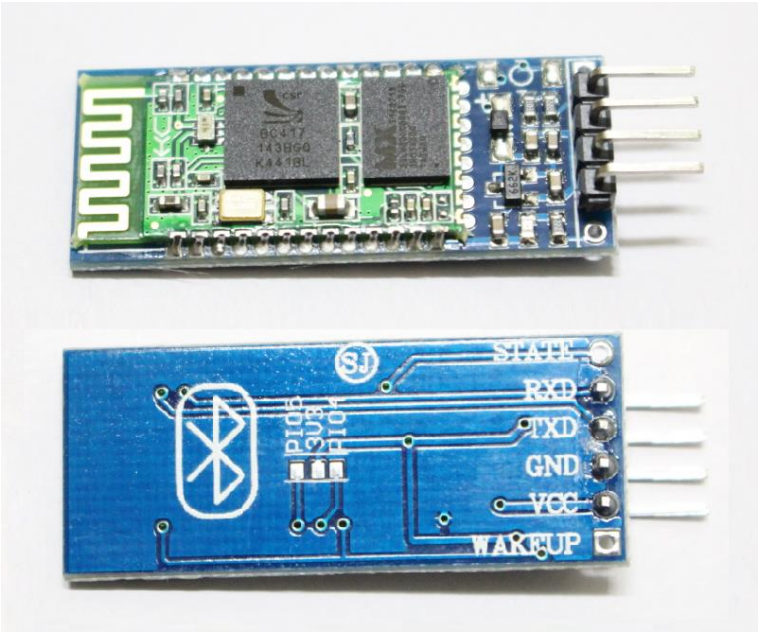
11.2 아두이노와 스마트폰의 블루투스 통신



블루투스 통신은 말 그대로 근거리 통신망에 사용되는 무선통신 규약입니다. 게다가 사용하기 편하게 시리얼 통신이 가능한 칩셋 구조로 생산되므로, 시리얼 통신 가능한 MCU, PC 등의 대부분의 장치에서 사용할 수 있습니다. 시리얼 통신은 최소 송/수신 2 개의 포트만 있으면 사용가능한 구조입니다. 아두이노 보드도 안정적인 시리얼 통신이 가능한 기기이므로 블루투스 모듈을 사용하면, 안정적인 근거리 무선 시리얼 통신이 가능합니다.

11.3 블루투스 HC-06 슬레이브 모듈

탱크의 근거리 통신 제어를 담당할 부품으로 블루투스 HC-06 슬레이브 모듈을 사용합니다. 아두이노 보드에 연결하여 사용가능한 블루투스 모듈은 HC-06 슬레이브 모듈과 HC-05 마스터 모듈이 있습니다. HC-05 블루투스 모듈은 마스터, 슬레이브 기능이 모두 지원됩니다.



블루투스 HC-06 슬레이브 모듈

HC-06 모듈은 아두이노 또는 MCU 보드를 사용하는 프로젝트에서 근거리 무선 통신에 많이 사용되며, 통신 거리는 10m 내외입니다. 좀 더 먼 거리를 사용하기 위해서는 블루투스 4.0 지원 모듈, RF 무선 통신 UART 모듈 등을 사용하기도 합니다.

기술 사양

- **2.4GHz 안테나 내장**
- **EDR 블루투스 2.0, 2Mbps - 3Mbps**
- **외부 8Mbit FLASH**
- **3.3V ~5V 사용 가능. (최대 7V)**
- **표준 HCI 포트 (UART)**
- **옵션 PIO 제어**
- **SMD 배치 프로세스로 모듈**
- **디지털 2.4GHz 무선 송신**
- **CSR BC04 블루투스 칩 기술**
- **블루투스 클래스 2 전력 레벨**
- **RoHS 규제 절차**
- **보관 온도: -40 +85 도, 작동 온도: -25 으로 75 도**

■ **외형 (27mm × 13mm × 2mm)**

본 키트에 사용되는 블루투스 모듈은 GND, VCC, RX, TX 핀과 아두이노 핀에 연결하여 사용하는 모듈입니다. 별도의 RX/TX 5V ←→ 3V3 쉬프트 다운 레벨 컨버터 저항 연결 구성을 할 필요는 없습니다.

11.4 아두이노와 블루투스 모듈 연결

HC-06 모듈(또는 HC-05 모듈)은 블루투스 무선 통신을 통해 시리얼 데이터를 주고 받을 수 있습니다. 블루투스 마스터/슬레이브 1:1 페어링이 된 상태인 경우 기기와의 통신 데이터는 모두 내부 시리얼 포트로 통신 가능하도록 구성되어 있습니다. 즉, HC-06(또는 HC-05) 모듈을 사용하기 위해서는 상대방 기기에서도 시리얼 포트 통신이 가능해야 합니다.

11.4.1 아두이노 하드웨어 시리얼 포트

아두이노에서의 시리얼 통신 자원은 포트 1 개가 있습니다. 참고로 아두이노 메가 2560 보드는 하드웨어 시리얼 포트가 4 개 있습니다. 아두이노 우노 R3 보드의 MCU 는 ATmega328p 를 사용합니다. ATmega328P MCU 에서는 하드웨어 시리얼 통신 포트가 1 개만 지원됩니다.

아두이노 우노 R3 보드에서는 D0, D1 포트 2 개를 하드웨어 시리얼 포트로 펌웨어 업로드 포트로 고정하여 사용하고 있습니다. USB 케이블 연결 후 아두이노 IDE 로 펌웨어 업로드 및 테스트 시 사용되고 있습니다.

블루투스 HC-06 모듈에는 RX (수신 포트), TX(송신 포트)가 있습니다. D0, D1 연결하여 사용할 수 있습니다. 블루투스 모듈의 RX/TX, D0, D1 연결일 때는 아두이노 IDE 스케치에서의 프로그램 업로드가 안됩니다. 펌웨어 업로드 후 단독으로

아두이노 사용할 경우에는 블루투스 모듈을 D0, D1 연결하여 사용할 수 있지만 다소 사용하기 불편합니다.

11.4.2 아두이노 소프트웨어 시리얼 통신

아두이노 IDE 스케치 기본 라이브러리 중에는 "소프트웨어 시리얼 라이브러리"가 있습니다. 소프트웨어적인 방식으로 시리얼 통신을 지원합니다. 아두이노 보드의 임의의 포트를 지정하여 수신/송신 포트로 사용할 수 있습니다. 수신(RX) D10, 송신(TX) D11 포트를 사용하는 경우에는 블루투스 모듈의 TX, RX 포트를 연결하면 됩니다.

11.4.3 하드웨어, 소프트웨어 시리얼 통신의 차이점

아두이노를 비롯한 MCU 보드에서의 시리얼 통신 자원을 하드웨어 포트, 소프트웨어 포트 방식으로 사용하면, 일반적인 시리얼 통신은 MCU 클럭 및 환경에 따라 최대 속도가 달라집니다. 그 외에 시리얼 FIFO 버퍼 용량 등이 있습니다.

>> 차이점
안정적인 시리얼 통신을 하기 위해서는 하드웨어 시리얼 포트를 사용 해야 합니다.

소프트웨어 시리얼 포트를 사용하는 경우에는 통신 처리를 하기 위해 다른 나머지 포트들이 제대로 신호 처리를 할 수 없는 경우가 대부분입니다. 반대로 다른 포트들이 신호 변경되는 동안에는 소프트웨어 시리얼 통신이 불가 합니다.

최근의 아두이노 소프트웨어 시리얼2 라이브러리를 사용하는 경우에는 하드웨어 시리얼 포트처럼 다른 포트에 영향을 받지 않고 사용가능 하다고 합니다. 필요한 경우에는 테스트해 보시기 바랍니다.

11.5 블루투스 시리얼 데이터 연동 기초

블루투스로 데이터를 주고 받기 위해서는 블루투스 기기(모듈) 페어링(Pairing)되어 있어야 합니다. 페어링이란 용어 그대로 짝짓기입니다. 블루투스 기기의 통신 기능은 모두 페어링 상태에서 작동이 가능하게 되어 있습니다.

스마트폰은 거의 모두 블루투스 마스터/슬레이브 기능을 지원하고 있습니다.

>> 블루투스 마스터 기능

마스터 기능은 쉽게 말하면 주변에 있는 블루투스 지원 기기들을 검색하여 페어링을 할 수 있는 기능입니다. 물론 반대로 슬레이브 기능도 가지고 있습니다. 즉, 다른 기기에서의 페어링 요청을 받아들여 사용될 수도 있습니다.

블루투스 마스터 모듈의 기본 기능 상태는 대부분 슬레이브 상태입니다. 마스터 기능으로 사용하기 위해서는 블루투스 모듈의 명령어 설정 모드로 진입하여 일련의 명령어를 입력하여 사용하게 됩니다.

>> 블루투스 슬레이브 기능

슬레이브 기능은 주변 블루투스의 요청에 의해 페어링이 될 수 있습니다. 스마트폰의 블루투스를 마스터로 사용하여 아두이노와 연결된 블루투스 모듈과 페어링을 합니다. 리모트 컨트롤 작동 또는 초음파 센싱 운행과 리모트 컨트롤 코드를 구현한 상태인 경우 거의 비슷하게 수정하여 변경할 수 있습니다.

블루투스 통신에 의한 운행 제어도 쉽게 적용시킬 수 있습니다. 시리얼 통신 기반으로 명령어 전송 및 처리 부분을 구현해 주면 됩니다.

12 다운로드 – 소스코드, 앱

12.1 소스코드

■ 로봇 탱크 소스코드 다운로드

arduino-robot-tank.zip

/arduino_robot_tank_rc : 리모트 컨트롤 주행 소스코드
/arduino_robot_tank_wifi : 와이파이 주행 소스코드
/arduino_robot_tank_bluetooth : 블루투스 주행 소스코드

12.2 안드로이드 앱

■ 스마트 탱크 와이파이 앱 : Arduino-Smart-Tank-KIT.apk

■ 스마트 탱크 블루투스 앱 : GPSmartCarRemoteManager.apk

감사합니다

http://www.gameplusedu.com

http://www.gameplusbot.com

http://www.dronemaker.co.kr