

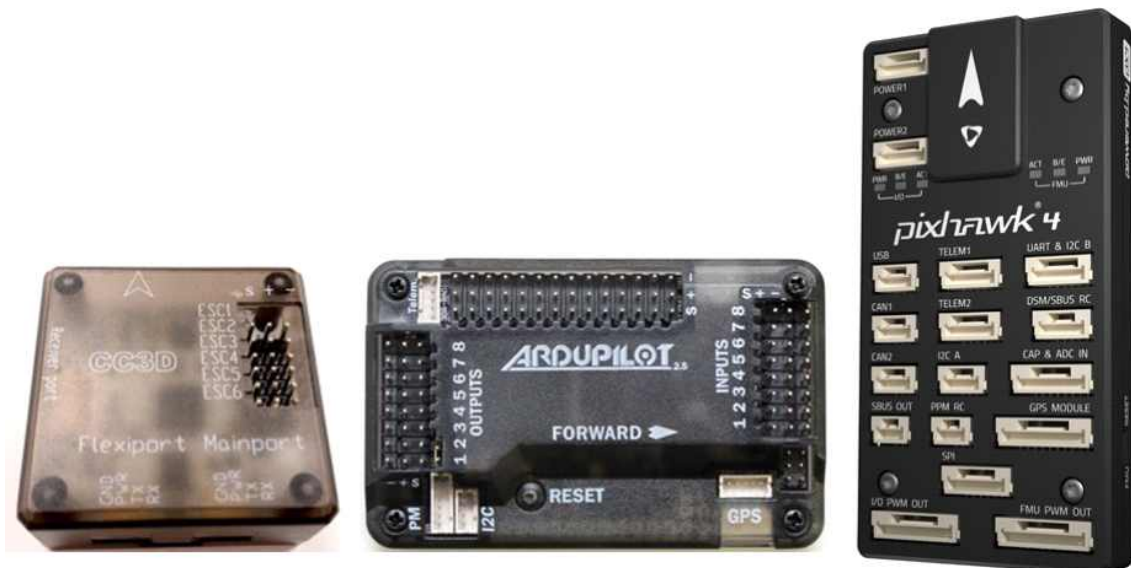
재미있는 아두이노 드론 만들기

아두콥터 V220 GPS

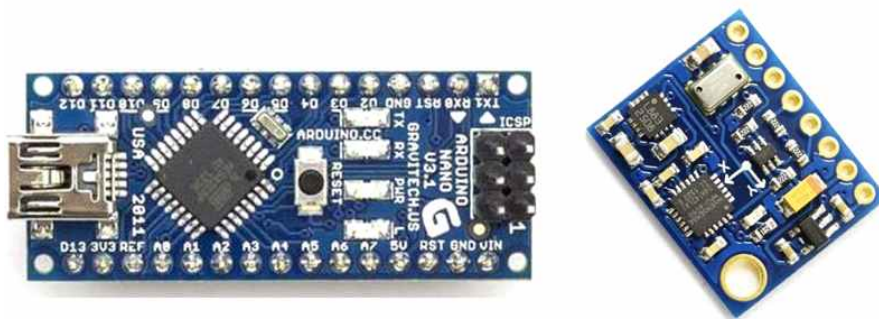


1. 아두이노 드론 개요

하늘을 나는 로봇 드론의 핵심인 비행제어장치 FC (Flight Controller)는 현재 CC3D, NAZE32, APM, PIXHAWK 등 많은 종류 들이 시중에 나와 있다. 이들 모두 소프트웨어와 하드웨어가 오픈 되어 공개 되어 있다. FC 제어 보드를 구매하여 오픈 소프트웨어를 FC 에 올리는 것만으로도 시중에서 판매되고 있는 완성품 드론 못지않게 성능과 안정성에 결코 뒤지지 않는다. 오히려 제작자의 창의성에 의해 많은 기능들을 추가하여 독특한 기능의 드론으로 거듭 날 수 있도록 개발이 가능하다. 자료들이 오픈되어 있어 나만의 드론을 만들 수 가 있다.



[그림1 오픈소스를 사용하는 비행제어장치]



[그림2. 아두이노 NANO 와 GY-86 센서]

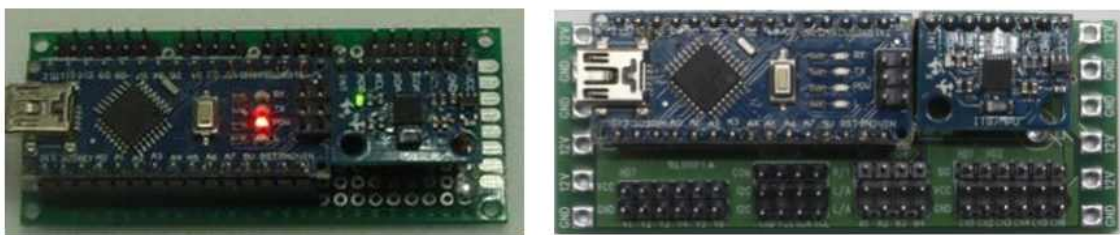
본 설명서는 아두이노를 사용한 드론 비행제어장치를 만드는 전 과정을 설명한다. 아두이노 보드 중 크기가 작으면서도 시리얼 통신 포트가 내장되어 있는 아두이노 Nano 보드와 아두이노 10축 비행제어어 센서 GY-86을 이용하여 250급의 쿼드콥터 드론을 만드는 것을 설명한다. 과정에 따라 납땜용인 만능기판을 사용할 수도 있다. 일반적으로 전용기판을 사용한다.

드론을 제어하기 위해서는 하드웨어뿐만 아니라 소프트웨어인 소스코드가 필요하다. 아두이노 보드에 최적화된 소스코드는 오픈 소스코드인 멀티위 (Multiwii)를 아두이노 Nano 보드에 다운로드하고 멀티위 컨피그 (Multiwii-Conf) 소프트웨어를 이용하여 컴퓨터에서 기체를 설정하는 방법에 대해 설명한다.

아두이노 NANO 보드에 멀티위 오픈소스코드를 올리고 멀티위 컨피그 GUI에서 간단한 설정만으로도 매우 안정적인 미니 드론을 만들 수 있다.

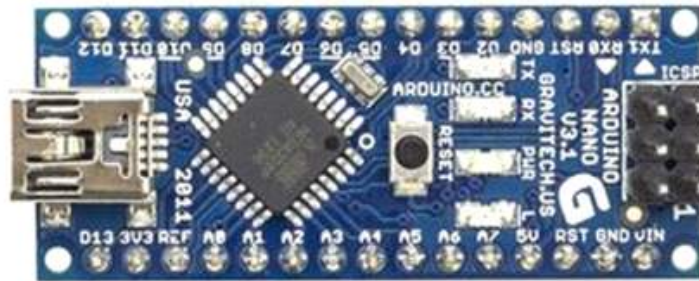
2. 아두이노 드론 준비하기

지금부터 아두이노 보드와 아두이노 센서를 사용한 아두이노 드론을 만들어 보자. 준비 사항으로는 아두이노 나노와 GY-521 또는 GY-80, GY-85, GY-86, GY-88만 있으면 완벽한 드론 비행장치를 만들 수 있다. GPS를 장착하여 AltHold, RTH(Return to home), Mission 기능 등을 추가하려면 Barometer가 내장된 GY-86센서 사용을 권장한다.

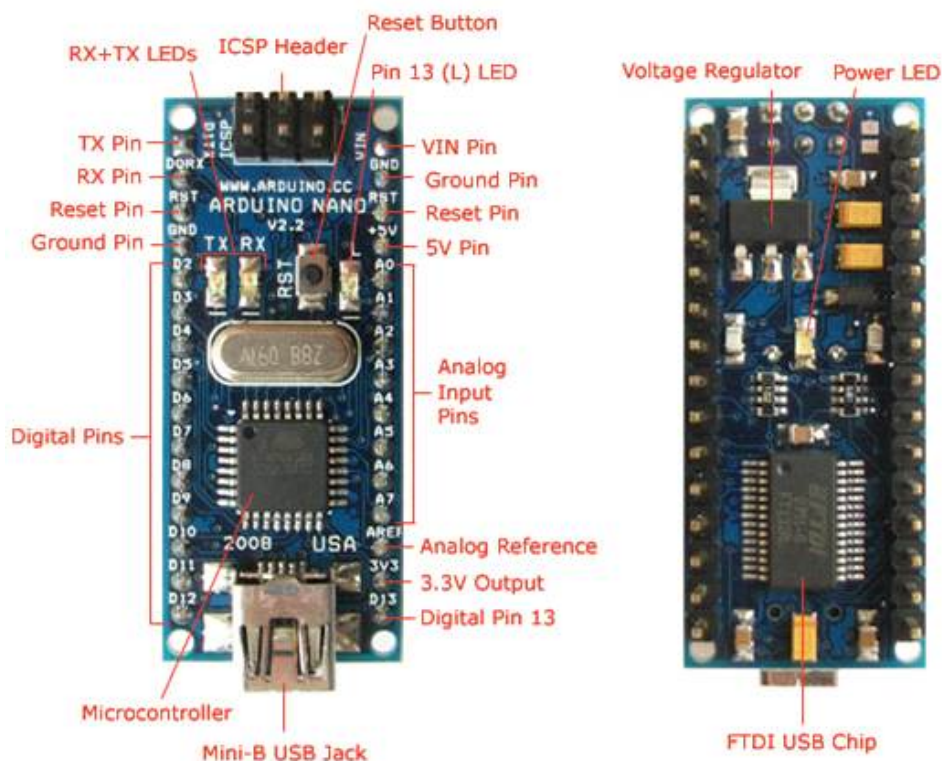


[그림3. 나노보드와 센서보드 조립]

(1) 아두이노 NANO V3 보드



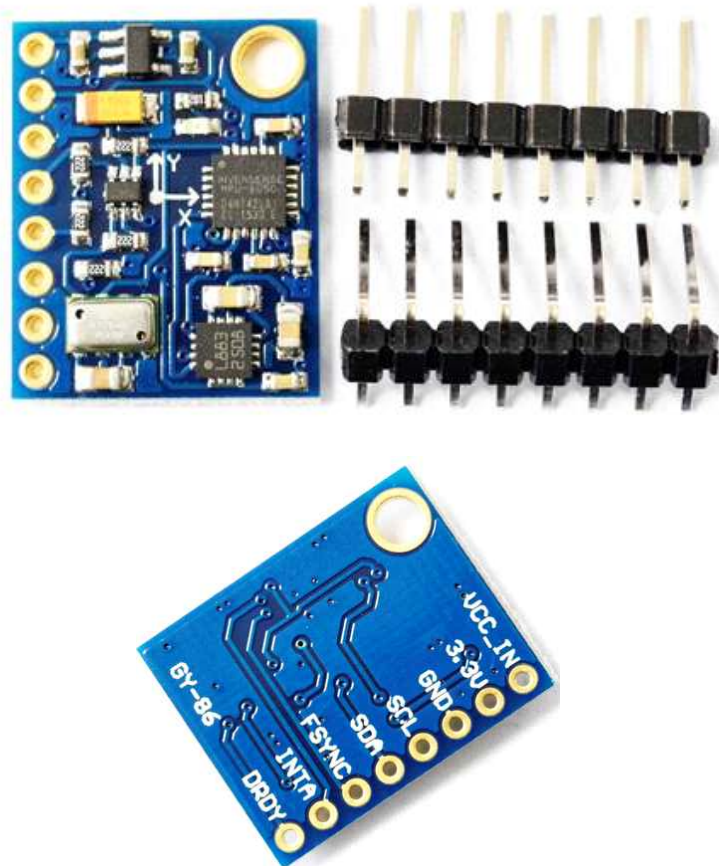
[그림4. 아두이노 NANO V3 보드]



- Microcontroller Atmel ATmega328
- Operating voltage is 5V
- Maximum supply voltage 6 to 20V
- Digital I/O pins 14 (of which 6 provide PWM output)
- DC current per I/O pin 40 mA
- DC current for 3.3V pin 50 mA
- Flash memory 16KB of which 2KB is used by Boot loader
- SRAM 1KB, EEPROM 512bytes
- Clock speed 16MHz

- USB Connector type Mini B male

(2) GY-86 비행제어 센서



[그림5. GY-86 통합형 드론제어 센서 모듈]

아두이노 GY-86 10축 비행제어센서 (자이로, 가속도, 지자기, 기압)로 하나의 모듈에 통합되어 있는 센서 보드이다.

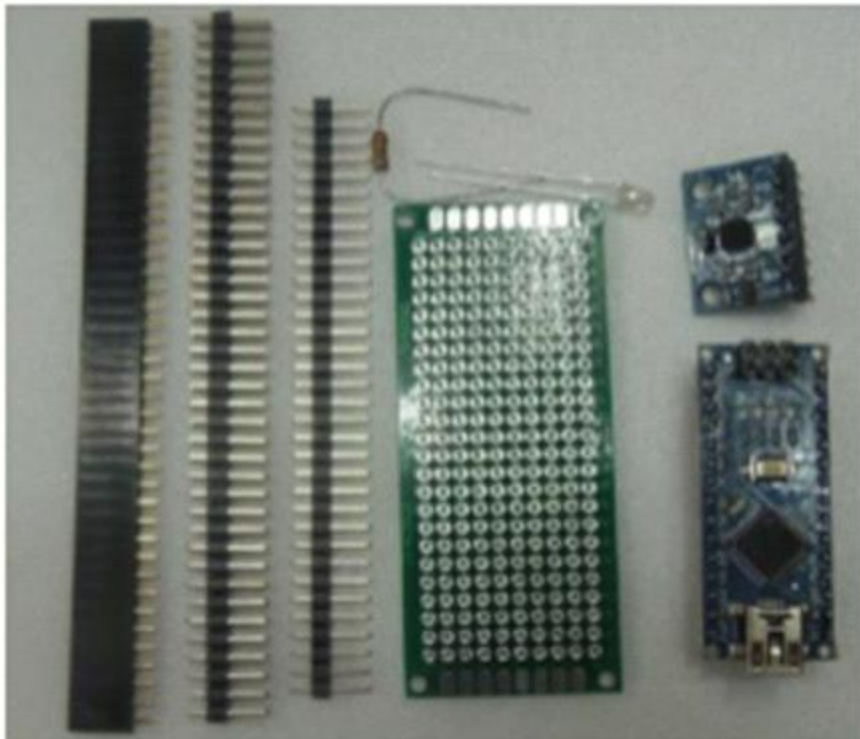
아두이노를 통해 10축 (3축 자이로, 3축 가속도, 3축 지자기, 1축 기압)을 측정할 수 있는 모듈이다. MPU6050 (3축 자이로, 3축 가속도), HMC5883L (3축 지자기), MS5611(1축 기압)을 하나의 모듈에 적용 하였다. 하나의 모듈에 여러 센서가 통합되어 센서 구현시 간편하게 마무리 할 수 있다. 자이로, 가속도, 지자기, 기압을 하나로 관리하므로 드론이나 무인항법장치등의 센서로 사용이 편히 하다는 장점을 가지고 있다.

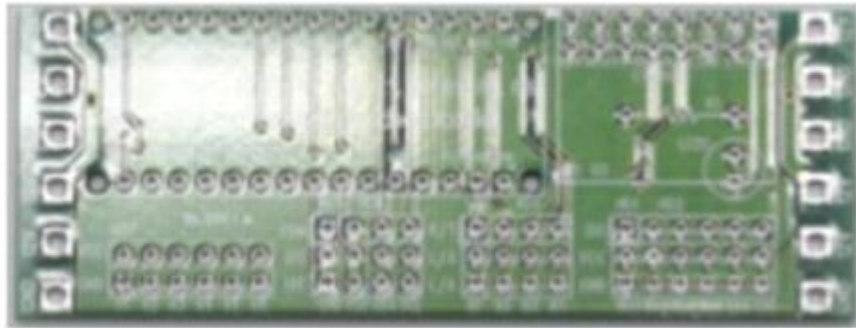
- 통신방식 : I2C
- 사용전압 : 3-5V

3. 부품 준비

(1) 아두이노 비행제어장치 일체형 보드 부품

- 가. Arduino Nano R3 보드
- 나. GY-521 가속센서 또는 GY-80, GY-85, **GY-86**, GY-88
- 다. Red LED 및 220Ω 저항
- 라. 만능기판 또는 전용기판
- 마. 2.54 Header Pin 및 Socket, 전선
- 바. 10cm F-F 케이블





[그림6. 드론 비행제어장치 및 파워보드 조립 준비]

(2) 드론 프레임



[그림7. 220급 안전 드론]

많은 종류의 드론들이 있다. 그 중에서도 200급부터 250급이 교육용 또는 레이싱 드론에 많이 적용되고 있다. 그림7.의 드론은 안전가드가 프레임 일체형으로 제작된 드론 프레임이다. 초보자도 쉽게 안전하게 드론을 배울 수 있다. 드론의 크기는 대각선의 두 모터 간 축간 거리로 한다. 이 프레임은 220급 드론이다.

(3) 브러쉬리스 모터 (Brushless Motor)



[그림8. 브러쉬리스 모터]

MT2204 2300KV BLDC Motor 4 개를 사용한다. 아두콥터 드론은 쿼드콥터 이므로 4개의 모터를 사용하고 대각선으로 각각 동일하게 회전을 한다. 사용되는 모터의 종류는 회전방향에 따라 2가지가 있다. 그림8.에서 은색과 검정색이 2종류의 모터를 보여주고 있다.

(4) 변속기 (Electronic Speed Controller)



[그림9. 브러쉬리스 변속기]

4개의 모터의 속도를 제어하기 위해서 4개의 변속기를 사용한다. 변속기의 종류는 다양하게 많이 있다. 본 아두콥터에서 사용하는 변속기는 기본 변속기로 PWM 방식의 신호로 제어되는 변속기 이다.

(5) 프로펠러 (Propeller)



[그림10. 5045 프로펠러]

사용되는 프로펠러는 회전방향에 따라 2종류를 사용한다. 시계방향 2개, 반시계방향 2개를 사용한다.

(6) 조종기 및 수신기, 3S 11.1V 1500-2200mAh Li-Po Battery



6채널 이상의 조종기와 수신기를 사용한다. 배터리는 기체의 무게를 감안하여 1500mAh이상을 사용한다.

[그림11. 조종기, 수신기, 리튬폴리머 배터리]

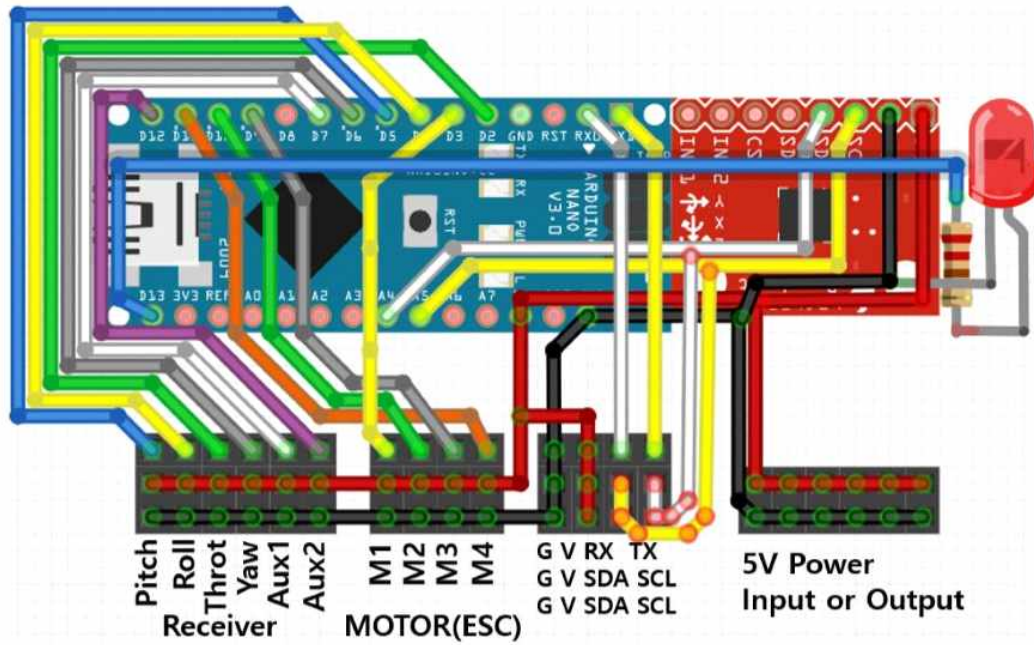
4. 아두이노 Nano 비행제어장치 (Flight Controller) 만들기

Arduino Nano 와 외부 Device 간의 인터페이스 역할을 하기 위한 Header 핀과의 결선표와 결선도이다. 수신기, 모터의 ESC, 시리얼통신, I2C 통신 및 5V 전원의 입출력을 위한 인터페이스 부분을 편리하게 하기 위해서 하나의 만능기판에 모든 부품들을 모아 장착하여 하나의 FC 로서의 기능을 하도록 한다. Nano 와 GY-521 도 header 소켓을 이용하여 탈장착이 가능 하도록 한다.

아두이노 나노와 외부 장치와의 결선표

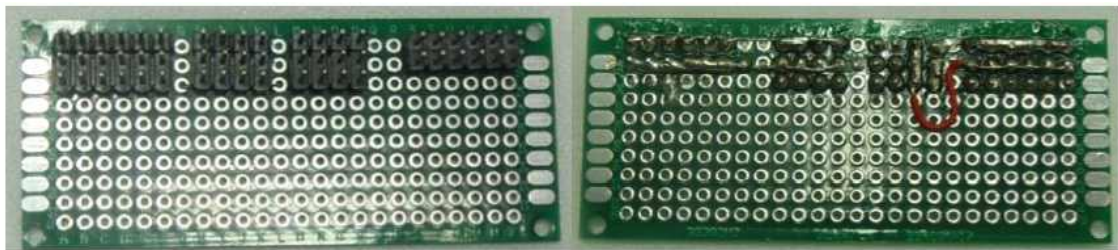
번호	기능	장치	아두이노 핀
1	Motor (ESC) Output	Motor1	D3
2		Motor2	D10
3		Motor3	D9
4		Motor4	D11
5	Receiver Input	Pitch	D5
6		Roll	D4
7		Throttle	D2
8		Yaw	D6
9		Aux1	D7
10		Aux2	D12
11	I2C	SDA	A4
12		SCL	A5
13	LED	LED	D13

3x7cm 240 홀 만능기판에 아래 사진과 같이 header pin 을 납땜하여 장착한다. 수신기, 모터와 ESC, GPS 나 Bluetooth 통신을 위한 Serial, LCD 등의 I2C 모듈을 위한 SCL(A4) 및 SDA(A5), 5V 전원의 입출력을 위해 인터페이스 포트로서 사용한다. 뒷면에 납땜하여 고정하고 전원라인과 그라운드 라인을 각각 아래 사진과 같이 동일 라인으로 결선한다.

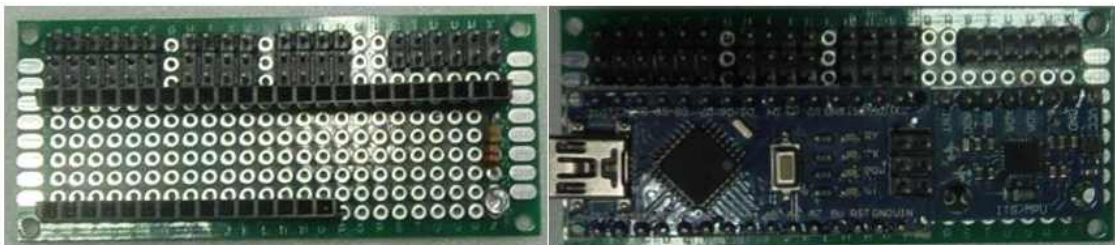


(외부 장치와 만능기판 아두이노 나노와의 결선도)

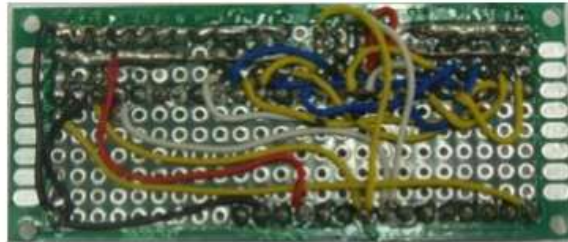
[그림12. 아두이노 보드와 센서모듈간의 결선도]



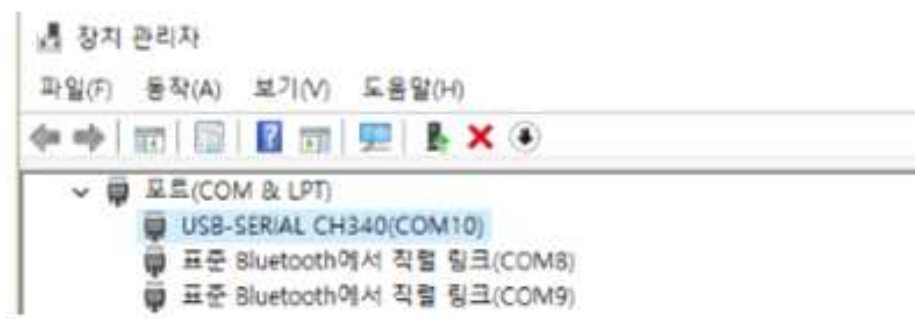
아두이노 나노와 GY-521 를 장착하기 위한 header 소켓을 나노와 GY-521 의 길이에 맞게 잘라서 만능기판에 장착 후 납땜한다. 또한 LED 와 저항도 납땜하여 고정한다.



결선표와 같이 만능기판의 뒷 판에 전선을 이용하여 결선한다. 쇼트가 발생하지 않도록 주의해서 납땜한다.



결선이 완료 되었으면 전원을 연결하여 Nano 와 GY-521 이 정상적으로 동작 하는지 확인 한다. PC 의 장치 관리자에서 나노의 시리얼포트가 정상적으로 인식되었는지 확인한다.

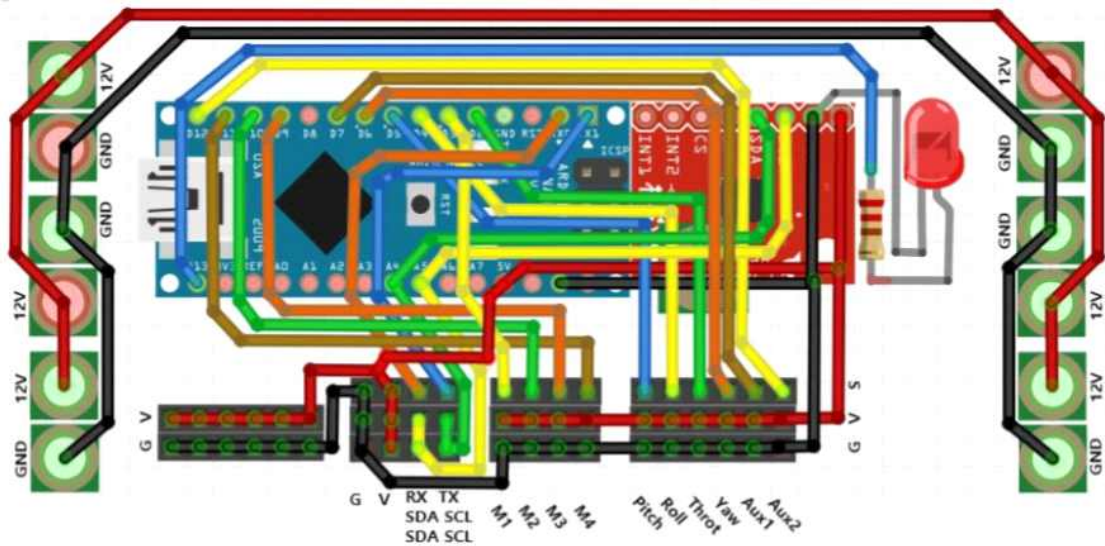


[그림13. 결선 후 확인]

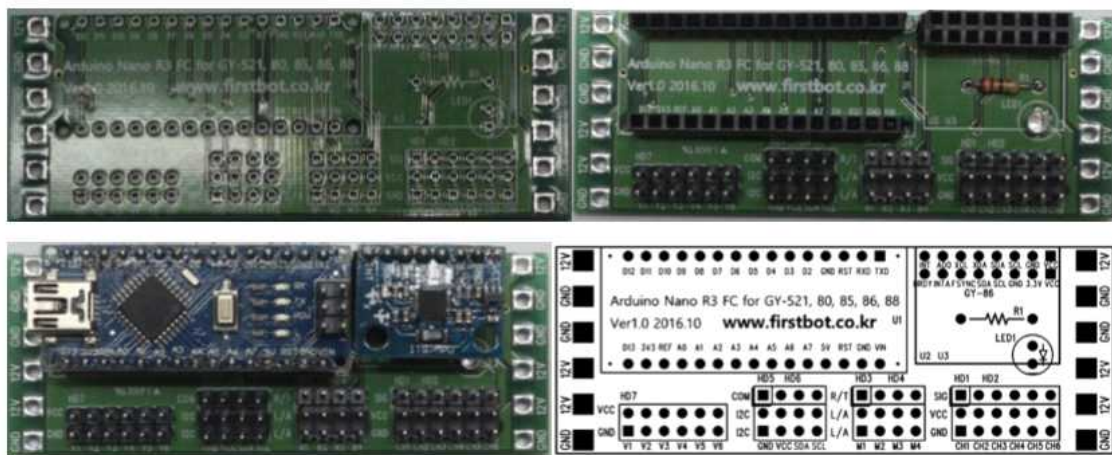
위와 같은 만능 기판의 불편 함을 줄이기 위해 회로 패턴을 모두 적용한 전용 PCB기판을 사 용해도 된다. 전용 PCB는 GY-521뿐만 아니라 80, 85, 86, 88도 사용 할 수 있으며 ESC 전원 공급을 하기 위한 별도의 Power 분배 패턴도 포함 하고 있다.

GY-521는 header핀에 장착하여도 괜찮으나 좀 더 안정성을 원한다면 header socket없이 직접 장착하여 납땜 하는 것이 좋다. 만약 GY-521이 아닌 센서(GY-86)들은 header socket의 2열에 삽입하여야 한다.

아래 결선도는 전용기판의 예이다. 만능기판과 전용기판의 핀헤더 및 핀소켓의 위치가 다르므로 이를 감안하여 본 설명서를 참조하기 바란다. 전용기판에는 12V 용 패턴이 준비되어 있어 별도의 ESC 공급용 파워분배 보드가 필요치 않다.



[그림14. 아두이노 NANO 전용기판의 결선 예]



5. MultiWii 아두이노 나노에 올리기

MultiWii와 MultiWiiConf GUI를 아래 사이트에서 다운로드 한다. MultiWii 는 아두이노 나노보드에 다운로드 할 FC 오픈소스이고, MultiWiiConf 는 기계 셋팅을 위한 GUI 이다.

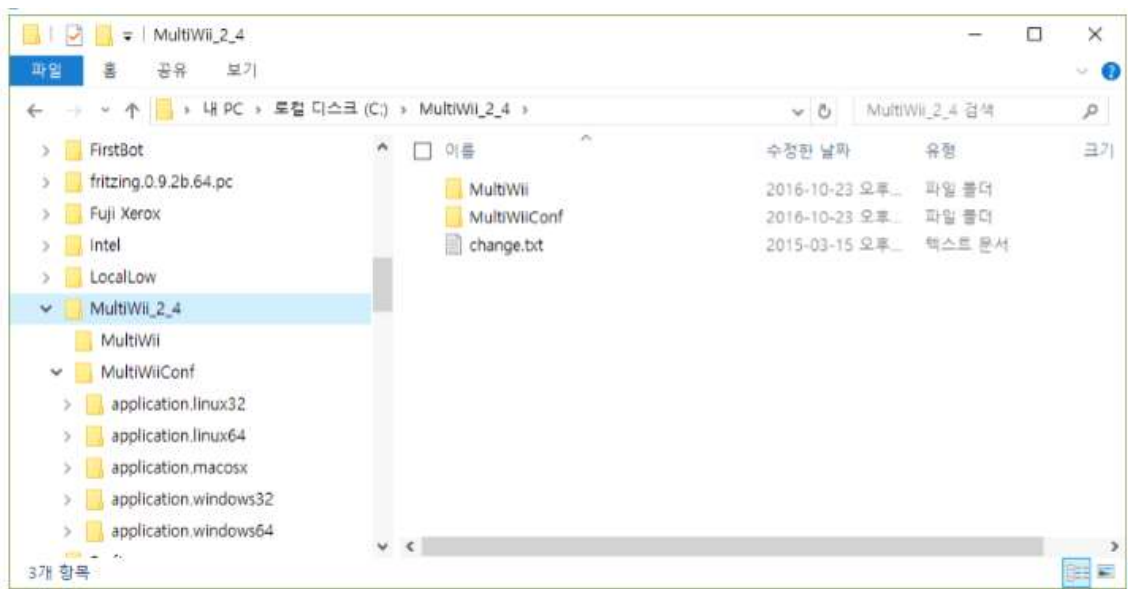
Multiwii Homepage : <http://www.multiwii.com>

Multiwii Download : <http://www.multiwii.com/software>

<https://code.google.com/archive/p/multiwii/>

<https://github.com/multiwii>

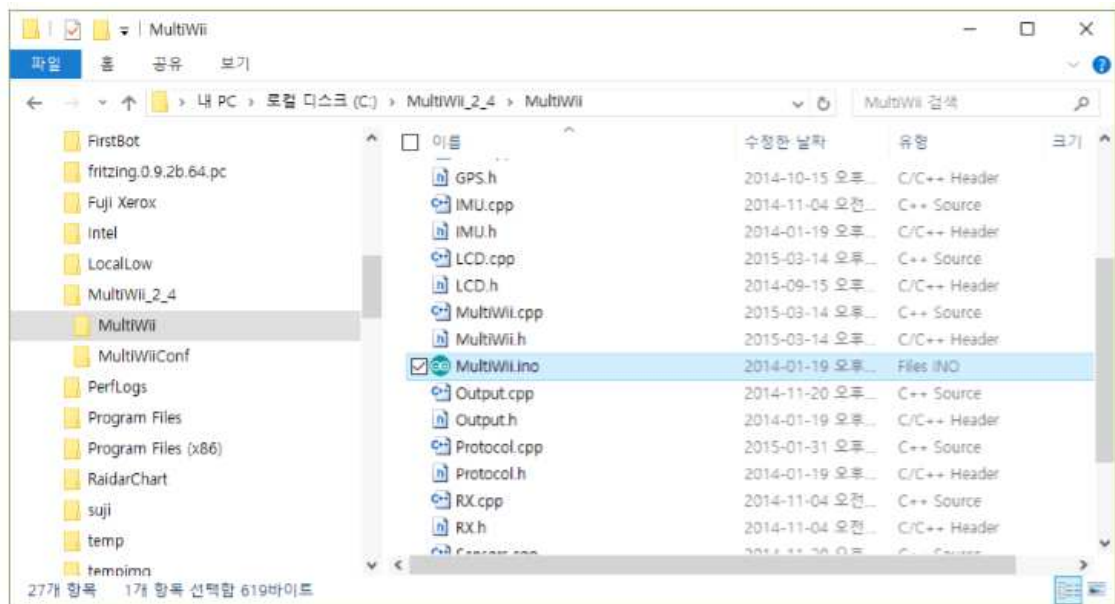
위 사이트에서 MultiWill_2_4.zip 파일을 다운로드 받아 압축을 풀면 아래와 사진과 같이 두 개의 상위 폴더(MultiWii, MultiWiiConf)가 있다.



MultiWii 하위 폴더에서 MultiWii.ino 를 두 번 클릭하여 아두이노 스케치를 실행한다.

만약 아두이노 스케치가 설치되어 있지 않다면 아래 사이트에서 스케치를 다운로드 받아 PC 에 설치한다.

Arduino Sketch Download : <http://www.arduino.org/downloads>



스케치가 실행되면 기체 셋팅을 위해 상단의 config.h 탭을 누른다.



기체의 형태를 결정한다. #define QUADX 의 주석을 풀어 **QuadCopter** 로 정의 한다.

```

34  /***** The type of multicopter *****/
35  // #define GIMBAL
36  // #define BI
37  // #define TRI
38  // #define QUADP
39  #define QUADX
40  // #define Y4
41  // #define Y6
42  // #define HEX6
43  // #define HEX6X
44  // #define HEX6H // New Model
45  // #define OCTOX8
46  // #define OCTOFLATP
47  // #define OCTOFLATX
48  // #define FLYING_WING
49  // #define VTAIL4
50  // #define AIRPLANE
51  // #define SINGLECOPTER
52  // #define DUALCOPTER
53  // #define HELI_120_CCPM
54  // #define HELI_90_DEG

```

자이로 센서의 종류를 결정한다. #define GY_521 의 주석을 풀어 GY_86 로 선택한다.

```

132 // #define GY_80 // Chinese 10 DOF with L3G4200D ADXL345 HMC5883L BMP085, LLC
133 // #define GY_85 // Chinese 9 DOF with ITG3205 ADXL345 HMC5883L LLC
134 // #define GY_86 // Chinese 10 DOF with MPU6050 HMC5883L MS5611, LLC
135 // #define GY_88 // Chinese 10 DOF with MPU6050 HMC5883L BMP085, LLC
136 #define GY_521 // Chinese 6 DOF with MPU6050, LLC

```

무선 수신기에 대한 정의는 “def.h”에 있는데 AUX2는 옵션으로 설정 하도록 되어 있다

```

329 #define THROTTLEPIN 2
330 #define ROLLPIN 4
331 #define PITCHPIN 5
332 #define YAWPIN 6
333 #define AUX1PIN 7
334 #define AUX2PIN 0 // optional PIN 8 or PIN 12
335 #define AUX3PIN 1 // unused
336 #define AUX4PIN 3 // unused

```

AUX1 까지 활성화 되어 있는데 6 채널 조종기를 사용하고 있다면 멀티위 소스 “config.h” 에서 아래 같이 주석을 풀어서 AUX2 를 활성화 한다.

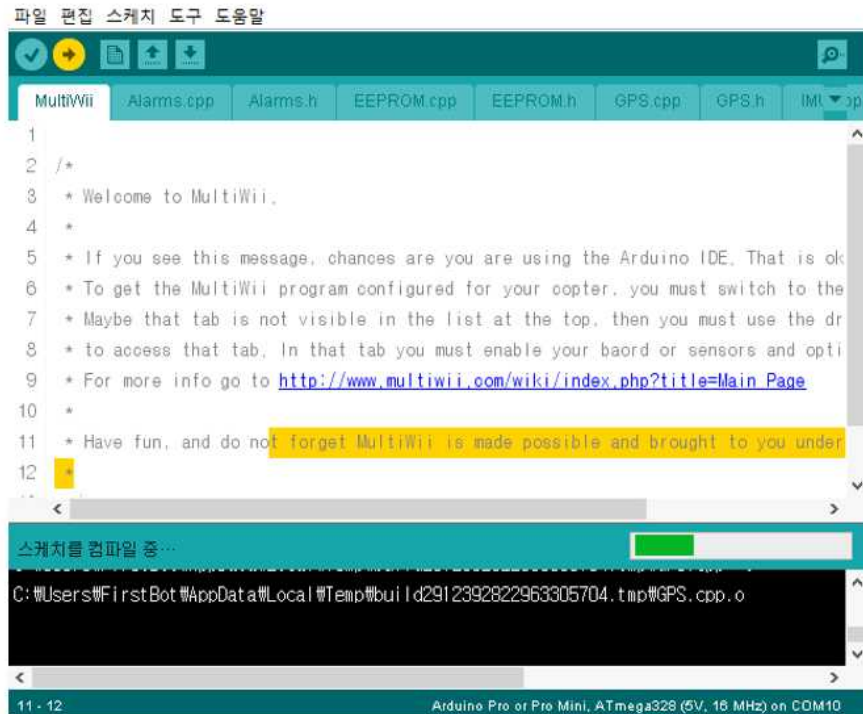
```

421 /***** Aux 2 Pin *****/
422 /* possibility to use PIN8 or PIN12 as the AUX2 RC input (only one, not both)
423    it deactivates in this case the POWER PIN (pin 12) or the BUZZER PIN (pin 8) */
424 // #define RCAUXPIN8
425 #define RCAUXPIN12

```

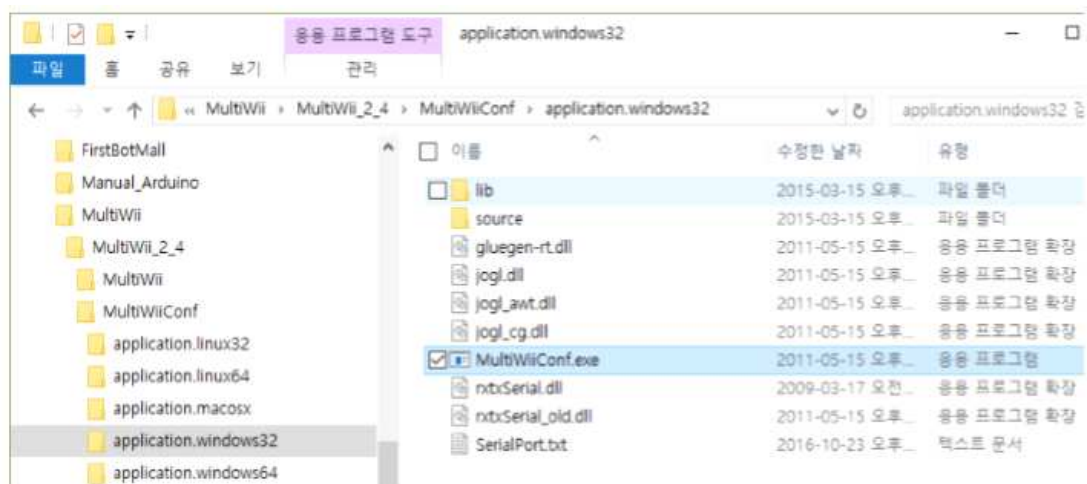


위와 같이 config.h 의 소스를 수정하고 Arduino Nano 의 시리얼 포트를 선택한다.



Upload 버튼을 눌러 MultiWill 소스를 아두이노 나노에 다운로드 한다.

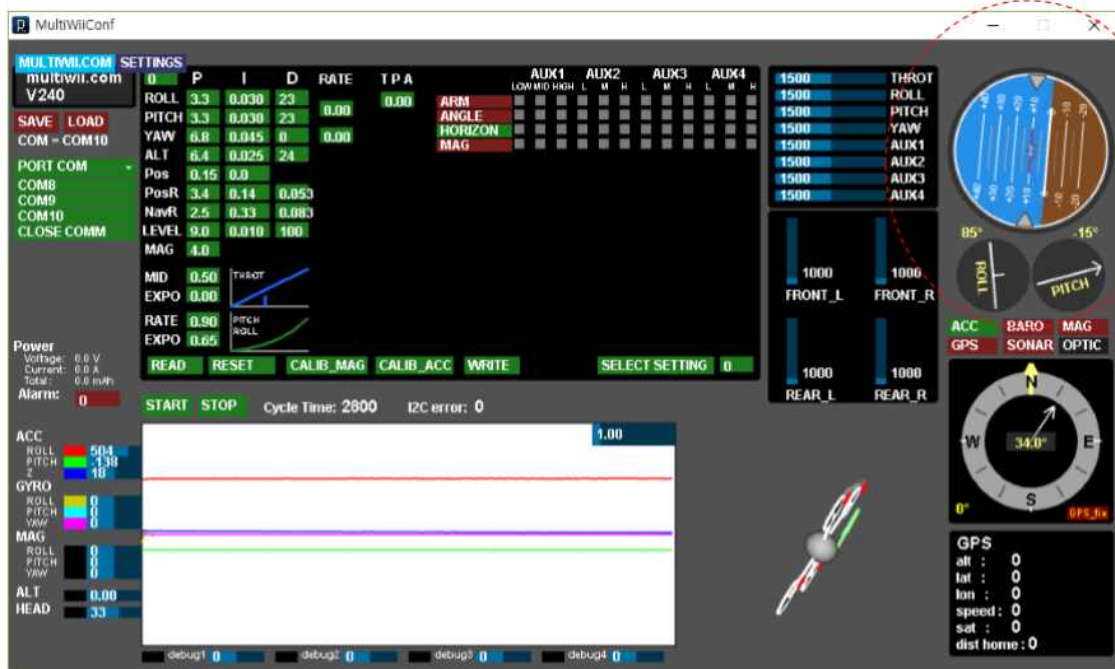
Quadcopter 의 기본적인 비행 셋팅을 위해 MultiWill 소스를 일부 수정하고 다운로드를 완료하였다. MultiWillConf GUI 를 실행하여 아두이노 나노 FC 보드의 이상 유무를 확인한다.



MultiWiiConf.exe 를 실행한다. 이 때 실행이 되지 않으면 오라클에서 제공하는 windows용 Java development platform이 설치 되어 있지 않아서 이다. 아래 링크에서 다운로드 받아 PC에 설치한다.

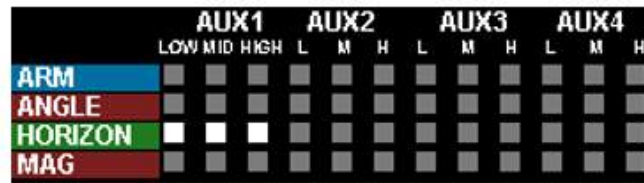
Windows용JavaDownload:https://www.java.com/ko/download/ie_manual.jsp?locale=ko

MultiWiiConf가 실행되면 Nano의 시리얼 포트를 선택한 후 Start 버튼을 누르면 현재 Nano FC의 정보를 읽어와 화면에 보여진다. GY-521이 보정이 되어 있지 않은 상태이므로 우측 화면의 수직수평계 및 Pitch 및 Roll 등의 각도가 0 이 아니다.



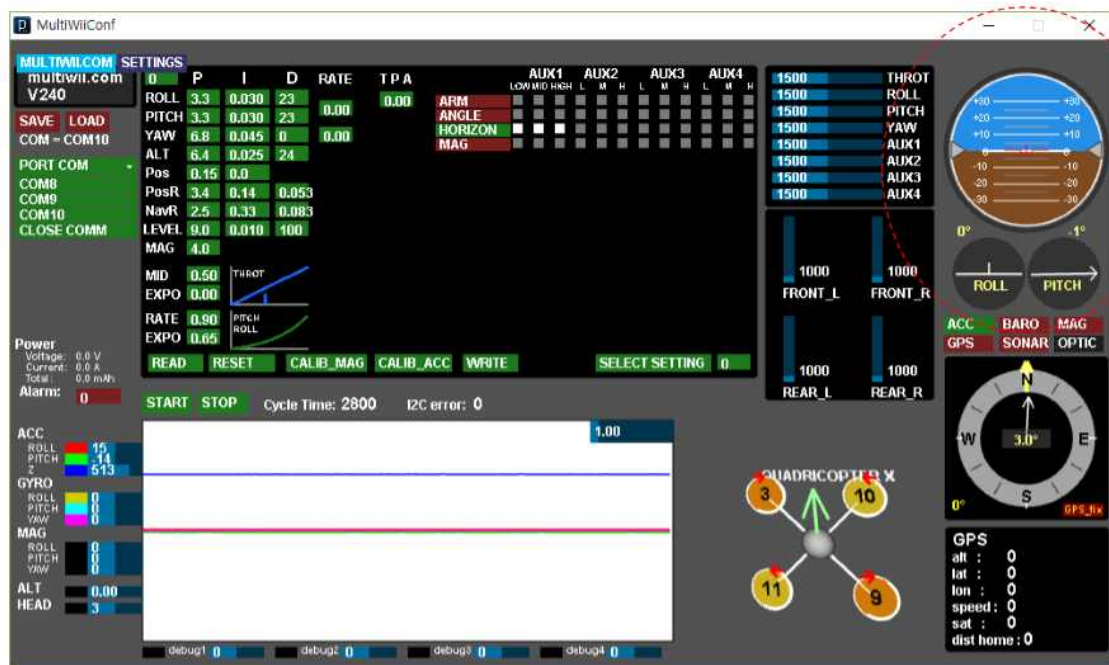
Nano FC를 수평 상태로 하고 **READ** **RESET** **CALIB_MAG** **CALIB_ACC** **WRITE** 에서 **CALIB_ACC** 를 눌러 가속(acceleration)센서를 보정(Calibration)한다. 이 때 수직수평계가 수직 및 수평이 되고, Roll, Pitch 의 각도가 0 도가 될 때까지 기다린다. (15초 정도).

그리고 **READ** **RESET** **CALIB_MAG** **CALIB_ACC** **WRITE** 의 Write 버튼을 눌러 저장한다.



조종기의 AUX1 을 이용하여 드론의 비행 모드를 결정하기 위해 사진의 HORIZON의 AUX1-LOW, MID, HIGH 체크 박스에 체크한 후 Write 버튼을 눌러 저장 한다. 조종기의 AUX1 의 스위치가 어디에 있든 비행 모드는 HORIZON 이다.

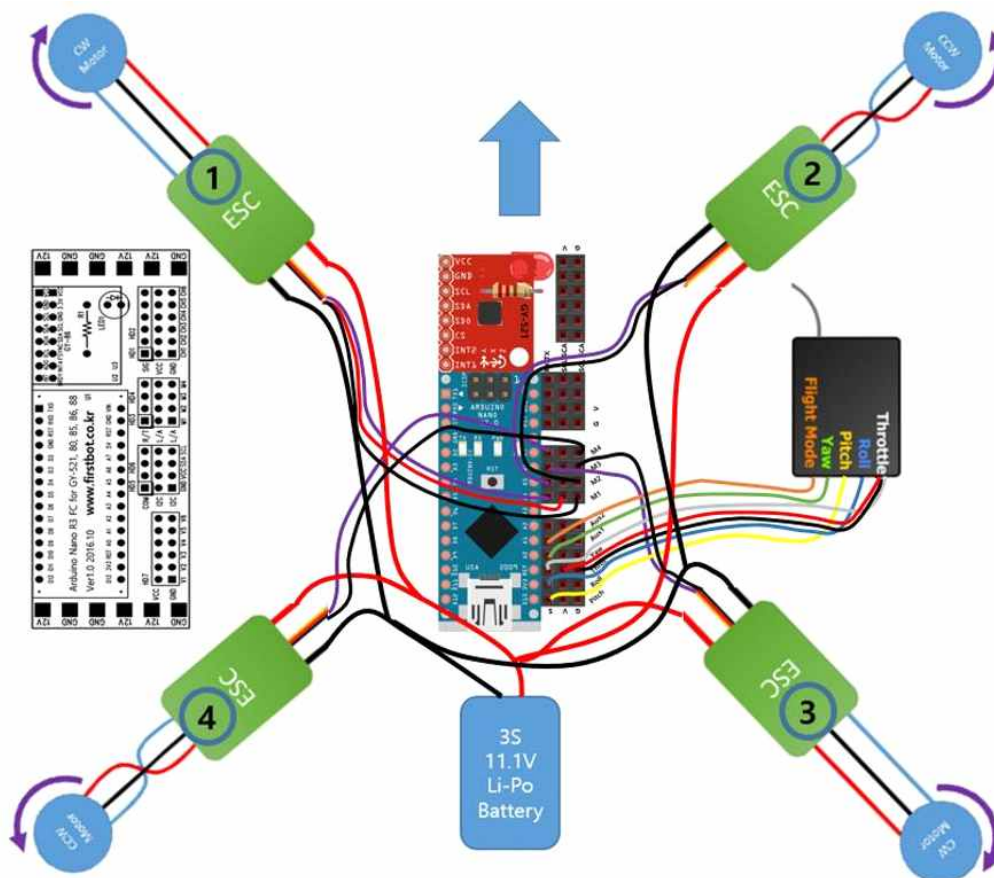
여기까지 셋팅 한 후의 사진이 아래와 같다면 Arduino Nano FC 는 정상적으로 만들어 진 것이다. 정확히 수직과 수평이 되었고 Roll 과 Pitch 의 각도가 0 도씩이면 정상이며 AUX1 에 의해 비행 모드가 HORIZON 으로 셋팅이 되었다.



Nano FC 가 정상적으로 동작하는지에 대해서만 확인하고 MultiWiiConf GUI 의 자세한 기능들에 대해선 뒤에서 다시 설명하겠다.

6. Nano FC, ESC, BLDC Motor, Receiver 및 Power 결선

Drone 220 Frame 의 중앙에 GY-86 이 위치 하도록 Nano FC 를 부착한다. 이 때 GY-86의 Y 화살표 표시가 앞 방향을 향하도록 해야 한다. 모터의 부착 순서와 모터와 ESC 의 결선을 눈여겨보자. 아래 사진에서 좌측 앞 방향과 우측 뒷 방향의 모터는 시계 방향으로 정회전 (CW)하므로 모터와 ESC 의 전선을 사진과 바로 결선하고, 우측 앞 방향과 좌측 뒷 방향의 모터는 시계 반대 방향으로 역회전(CCW) 하므로 가운데 선만 바로 연결하고 좌우 교차로 결선해야 한다. 하지만 조립이 완료 후에 회전 방향을 반드시 확인해서 원하는 방향과 반대일 경우는 다시 결선을 해야 한다. 4개 ESC의 BEC 5V 전원은 하나의 ESC에서만 전원선과 그라운드 선을 Nano FC의 Motor Interface 의 전원단에 연결한다. 나머지 ESC의 전원선은 연결하지 않고 그라운드 선만 연결하는 것을 권장한다. 그 이유는 ESC 마다의 출력전압의 차이가 있기 때문이다. 만능 보드로 구성된 FC 와 전용 보드 FC의 Pin Header 및 Socket 의 배치가 다르므로 결선 위치를 확인하고 주의해서 결선한다.



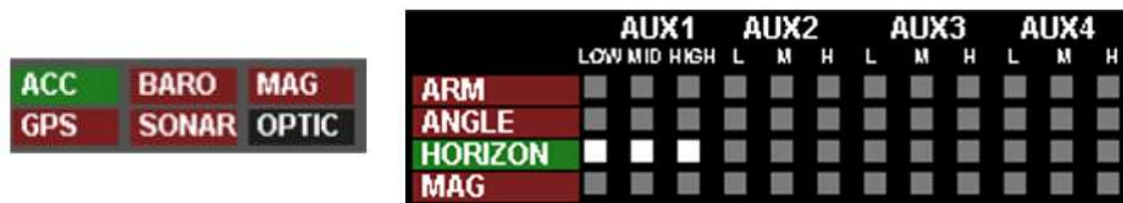
Nano FC와 Receiver의 연결은 듀폰 케이블로 연결한다. Nano FC Receiver 단의 전원선과 그라운드 선을 Receiver의 +/1 에 연결한다.

3S 11.1V Li-Po Battery 와의 결선은 전원 분배 보드를 이용하여 각 ESC 에 전원을 공급한다.

7. MultiWiiConf GUI 에서의 기체 셋팅

앞장에서 Nano FC 의 제작이 정상적으로 제작되었는지 확인하기 위해 잠깐 소개 했다. 이번에는 MultiWiiConf GUI 에서 기체를 셋팅하는 방법에 대해 알아 본다. 앞 장에서 설명한 부분만으로도 꽤 안정적인 비행이 가능하다.

(1) 비행모드 설정.



MultiWii 오픈소스에서 제공되는 비행 모드는 가속센서와 자이로 센서만 있는 경우, 아래 사진과 같은 비행 모드를 제공한다. 비행모드의 제공 여부는 자이로(gyro, 기울기) 센서, 가속(acceleration)센서, 지자계(magnetic)센서, GPS, 고도계, 기압계 등의 부착 여부에 따라서 달라진다. GY-86는 MPU6050칩을 사용한 10축 비행제어센서 모듈이다.

멀티위는 AUX1~AUX4 의 4 채널을 제공하여 여러 비행모드에 대응하는 키로서 사용 할 수 있으나, 본 매뉴얼에는 AUX1 만을 사용하여 HORIZON 비행 모드만을 AUX1의 LOW, MID, HIGH 에 동작(비행) 모드를 대응시키면 된다. 여기서는 AUX1 값이 LOW, MID, HIGH 모두 HORIZON 비행 모드로 할당 한다. 물론 AUX1-LOW는 ARM, AUX1-MID는 ANGLE, AUX1-HIGH를 각기 다른 방식의 비행 모드를 대응시킬 수 있다. 기본적인 비행 모드에 대한 설명은 아래와 같다.

ARM(ACRO) : 기본 비행 모드로 자이로 센서만 사용되며 자동수평 유지가 되지 않으므로 조종이 매우 어려워 초보자에게 적합하지 않다.

ANGLE : 자이로 센서와 가속센서 모두 사용되며 자동수평유지 비행 모드이다.

HORIZON : ANGLE 과 마찬가지로 자이로 센서와 가속센서 모두 사용되고 자동수평 유지 비행 모드이나 지정한 각도(Min/Max)를 넘어서면 ACRO 모드처럼 동작한다. 즉 조종기의 키를 천천히 움직이면 ANGLE 모드로 동작하고 빠르게 움직이면 ACRO 모드처럼 드론이 동작한다.

GY-521 센서 이용시 위와 같은 비행모드 밖에는 없다. 그러나 자이로, 가속, 지자계, 고도계, 기압계, GPS 등의 센서를 이용하면 아래와 같은 비행모드가 추가 될 수 있다.



예를 들어 GY-86 과 GPS 모듈을 장착하였다면 위의 그림과 같이 자이로 센서, 가속센서, 고도센서, 지자계 센서, GPS, 대기압(BARO) 센서가 활성화로 표시되고 가능한 비행 모드가 아래 사진과 같이 추가 된다

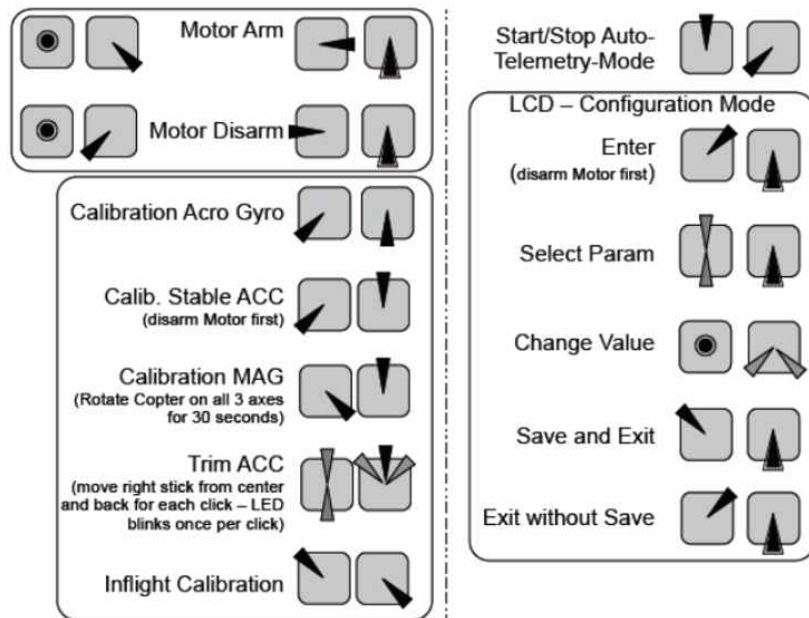
	AUX1			AUX2			AUX3			AUX4		
	LOW	MID	HIGH	L	M	H	L	M	H	L	M	H
ACC												
BARO												
MAG												
CAMSTAB												
CAMTRIG												
ARM												
GPS HOME												
GPS HOLD												
PASSTHRU												
HEADFREE												
BEEPER												
LEDMAX												
LLIGHTS												
HEADADJ												

문구에서 알 수 있듯이 GPS 모듈이 활성화 되어 있으면 GPS HOME, GPS HOLD, WAY Point (MISSION)등의 기능을, 지자계가 활성화 되어 있으면 HeadFree 등의 기능 및 비행 모드를 설정 할 수 있으며 대기압(BARO) 센서가 활성화 되어 있으면 고도를 유지 할 수 있는 기능을 추가 할 수 있다.

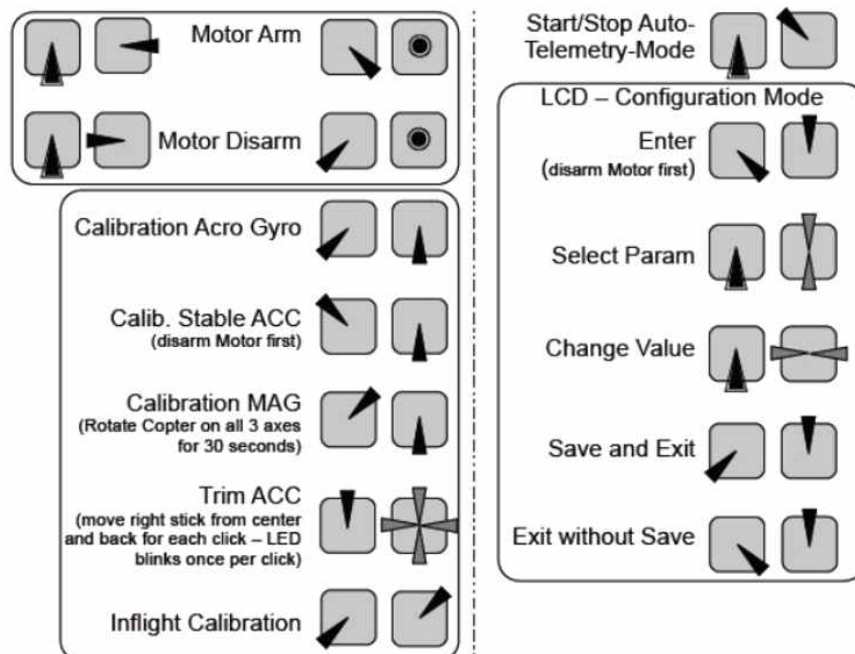
(2) 시동(Arming)

시동을 켜(Arm) 때는 조종기의 스로틀(Throttle) 키 값을 최하, 요(Yaw) 키 값을 최대로 해야 한다. 반대로 끌(Disarm) 때는 스로틀의 키 값을 최하로, 요 키 값을 최소로 하여 시동을 끈다. 그 외에 다른 기능들도 아래 그림과 같이 지원한다.

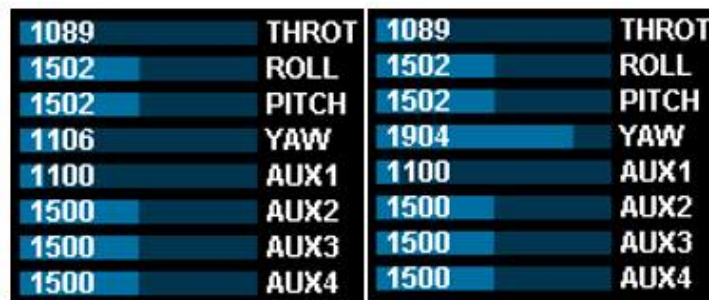
Mode 1



Mode 2



만약에 위와 같이 했는데도 시동이 걸리지 않는다면 아두이노 스케치의 Multiwii.h 파일에서 MINCHECK, MAXCHECK 값을 확인해야 한다. 아래 두 개의 그림에서 보면 조종기에서 수신된 요(YAW) 키 값의 최소가 1106 이고 최대값이 1904 이다.



Multiwii.h 에 키 값에 대한 최소 최대값이 1100, 1900 으로 디폴트 값으로 되어 있다. 이럴 경우 수신 받은 YAW 키 값이 최대 1904 이므로 시동(Arm)은 되지만, 최소값이 1106 이므로 시동이 꺼지지 않는다. 따라서 아래와 같이 Multiwii.h 의 다음 코드를 아래와 같이 수정한 후 컴파일과 Nano FC 에 다운로드를 하면 최소값이 1130 보다 작으므로 시동이 꺼진다.

```

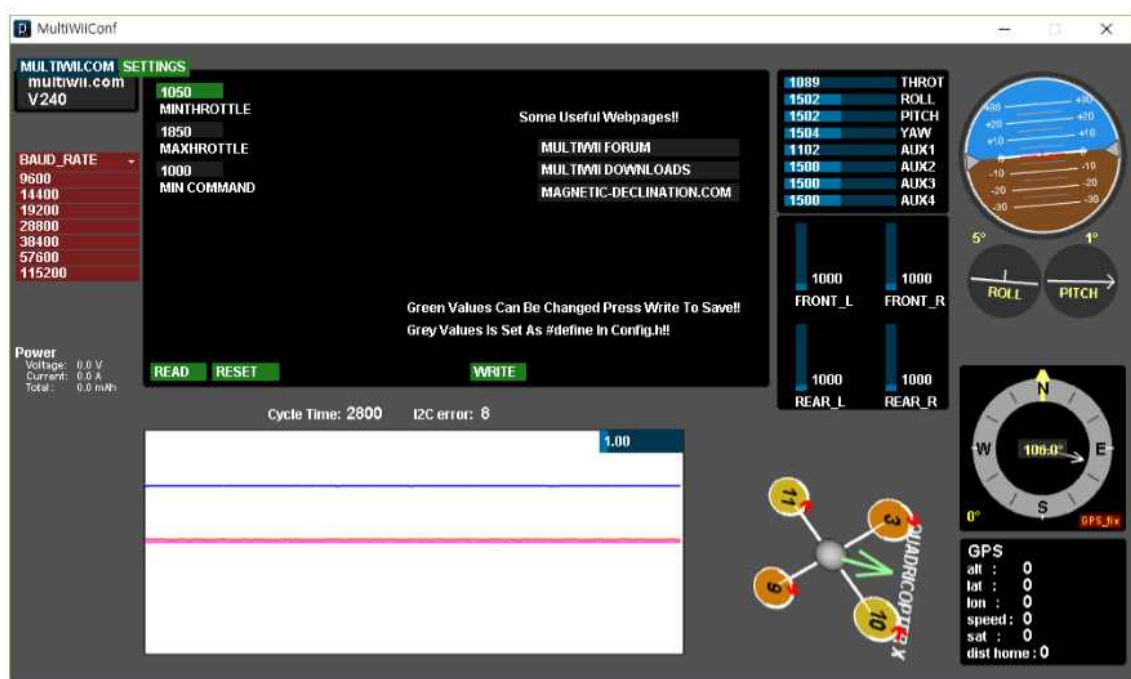
9 #define MINCHECK 1100      9 #define MINCHECK 1130
10 #define MAXCHECK 1900    10 #define MAXCHECK 1870

```

MultiWiiConf GUI 의 **MULTIWII.COM SETTINGS** 상단 탭 에서 SETTING을 누르면 아래의 사진과 같은 설정 화면이 나온다.



아래의 사진에서 쓰로틀 키를 맨 아래에 두었을 때의 값이 1089 이다. Minthrottle 의 값은 1050 이다. 이 경우는 아무 문제없다. 그러나 Minthrottle 의 값이 1100 으로 설정 되어 있을 경우에는 매우 위험 할 수 있다. 시동을 켜자마자 모터가 회전이 될 수 있기 때문에 조종기에서 수신되는 쓰로틀의 값보다 작은 값으로 변경해 줘야 한다. 녹색 에디터 창에 마우스 커서를 대고 클릭 키를 누른 상태에서 값을 변경 한 후 WRITE 버튼을 눌러 저장한다.



(3) PID(Proportional Integral Derivative)의 설정

ROLL	3.3	0.030	23
PITCH	3.3	0.030	23
YAW	6.8	0.045	0

앞에서 설명한 것으로도 안정적인 비행이 가능하다. 하지만 본인의 취향에 맞는 비행 성능과 스타일을 찾고 싶다면 PID 값을 튜닝 할 수 있다. 유연한 비행 제어를 위해 보통 PID 제어를 제공한다. 모터의 종류, ESC의 종류, 프로펠라의 종류, 무게 등이 기체마다 모두 다르므로 PID의 설정은 기체마다 다르게 설정한다. 따라서 본인의 기체가 잘 반응하는 PID 값을 여러 차례 실험에 의해 찾아서 자신의 드론에 적용해야 한다. 그러나 MultiWii 오픈 소스를 적용한 220 급 Mini Drone에서는 기본 설정 값만으로도 꽤 안정적인 비행이 가능하다고 판단된다.

Proportional(비례) 제어

현재 값이 Reference와의 차이가 크면 크게 보상하고 작으면 작게 보상한다. 쉽게 말하면 목표치에서 많이 벗어나면 많이 보상하고 목표치에 가까우면 적게 보상한다. Drone에서 보면 균형(평형)을 유지하기 위해 평형에서 많이 벗어날수록 기울어진 측이 모터가 빠르게 도는 것을

의미한다. 물론 반대쪽 모터는 상대적으로 느리게 돌게 된다. 즉, Reference 와 차이에 Factor 를 곱하여 Reference 에 접근하는 방식으로 P 값이 너무 크다면 Reference 에 도달하지 못하고 기체는 불안해 질 수 있다. 또한 P 값이 너무 크면 기체 반응 속도가 너무 빨라 제어가 어렵고, 너무 작으면 안정적인 반면 반응 속도가 너무 느려지게 된다. 적당한 값으로 튜닝을 해야 한다.

Integral(적분) 제어


수평, 각도 0 에 맞추기 위해 기체가 흔들리게 되고 0 도에 수렴하지 못한 오차를 누적합산 하여 점차 오차를 줄여 나가는 방식으로 미세하게 기체를 수평으로 맞추어 나간다. 즉 한 쪽으로 기울어 졌을 때 반대쪽 오차를 만들어 반대쪽으로 기울게 만들고 이러한 행위를 수평이 될 까지 오차를 줄여가면서 하게 된다. 아무리 PI 값을 조정하여도 기체가 수평을 유지하지 못하고 호버링이 어렵다면 앞서 설명한 ACC Calibration 이 잘 못 되었을 가능성이 크므로 ACC Calibration 을 다시 실행한다.

Derivative(미분) 제어

상당히 빠르게 반응하는 factor 로 기체가 외부(강풍) 요인이나 빠르게 제어하는 경우 수평을 유지하기 위한 Reference 와의 차이를 빠르게 보상하는 역할을 한다.

PID 제어란 기체의 수평을 유지하기 위한 Reference 와의 차이를 빠르게 보정하기 위한 Factor 들이다. 따라서 정확한 Reference 를 있어야 하기 때문에 앞서 설명한 바와 같이 ACC Calibration 이 정확히 되어 있어야 한다.

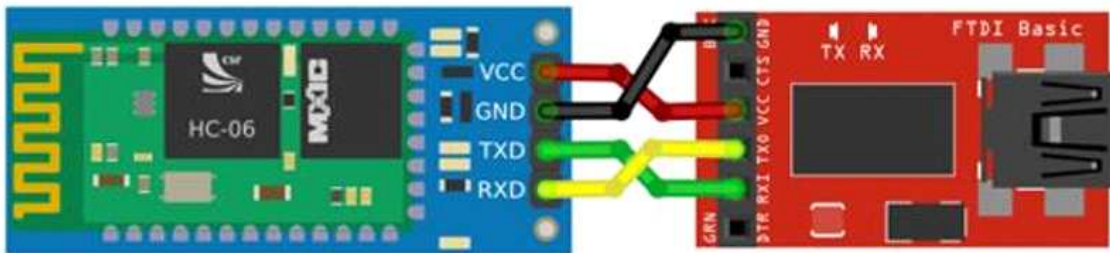
자신의 기체가 호버링이 잘 안되고 비행시 불안하다면 ROLL, PITCH, YAW 에 대한 값을 변경하여 자신의 기체에 적절한 값을 찾는다.

초보자라면 쓰로틀 커브  곡선을 와 같은 식으로 중간 출력 부분을 완만하게 급격하게 동작하지 않게 하는 것이 좋다.

TIP1. Bluetooth HC-06 Module 장착하기

MultiWiiConf 또는 안드로이드 앱인 “MultiWii Configuration Tool”, “EZ-GUI Ground Station” 등과 연동하기 위해 우선 baudrate 115200bps 로 설정한다. 기본 baudrate 는 9600bps, bluetooth name 은 HC-06 이다.

아래 그림과 같이 FT232RL 과 같은 USB to TTL Serial module 과 결선하고 PC 에서 시리얼 모니터를 이용하여 변경한다.



“AT” command 로 우선 통신 가능한지 확인한다.



“AT+NAMELibrePilot” command 로 LibrePilot 으로 블루투스명으로 변경한다.



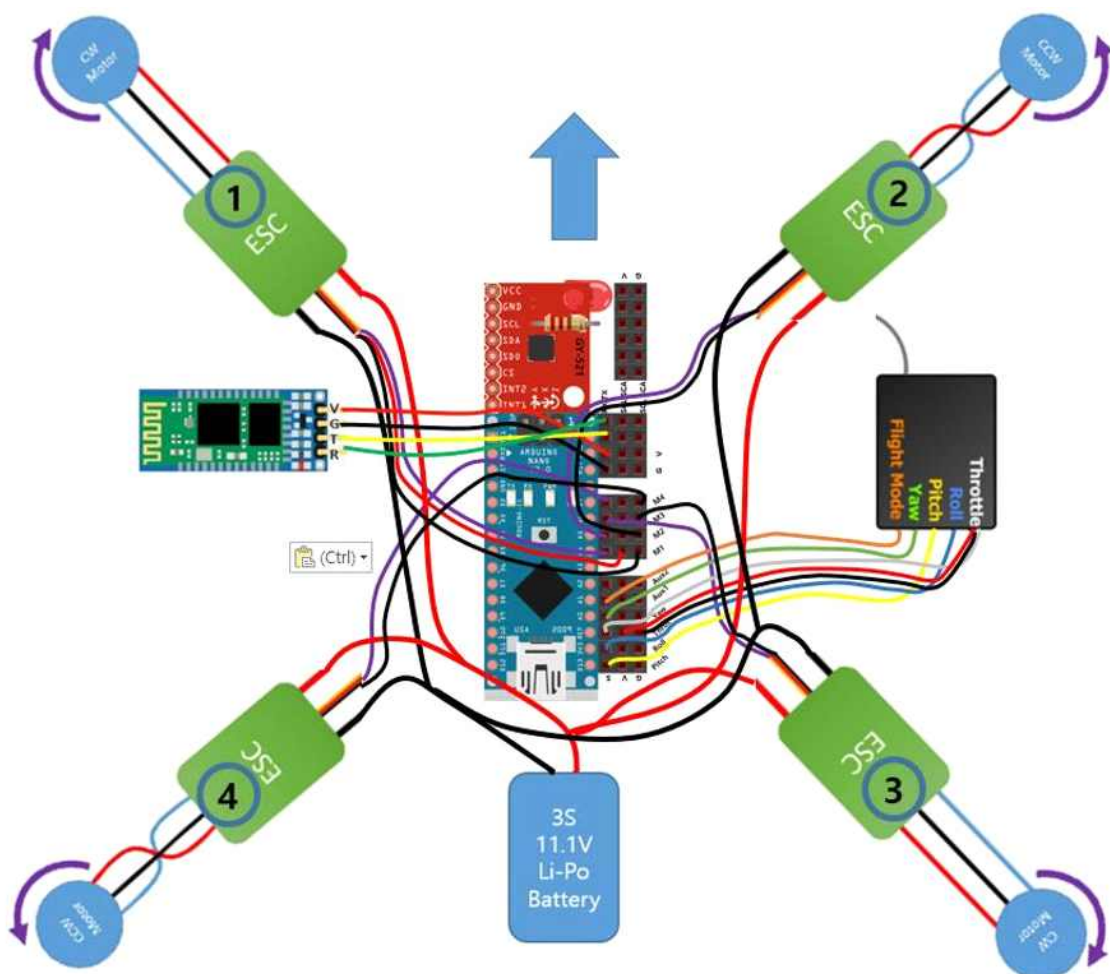
“AT+BAUD8” command 로 baudrate 를 115200bps 로 변경한다.

- 1-----1200
- 2-----2400
- 3-----4800
- 4-----9600 (Default)
- 5-----19200
- 6-----38400
- 7-----57600
- 8-----115200
- 9-----230400
- A-----460800
- B-----921600
- C-----1382400



HC-06 무선 시리얼 모듈을 아래와 같이 장착한다. HC-06 의 TX 와 RX 를 NANO 의 RX 및 TX 에 교차 하여 결선한다.

* 주의 할 점은 USB 케이블을 사용하여 NANO 와 시리얼 통신을 할 경우에는 HC-06 의 전원을 차단하여야 한다.



TIP2. GPS Module 장착하기

GPS Module 을 부착하여 GPS HOLD(고도유지) 및 GPS HOME(리턴 홈) 또는 MISSION(Way to point) 기능 등을 추가할 수 있다. Multiwii 에서는 NMEA, UBLOX, MTK protocol 을 지원하는 GPS 모듈들을 사용 할 수 있다. 9600bps 에서의 NMEA GPS 모듈을 Arduino D0(TX), D1(RX)를 이용하여 사용할 수 있지만 본 매뉴얼에서는 HC-06 Bluetooth module 을 telemetry 기능으로 사용하기 위해 D0, D1 을 사용하였기 때문에 I2C 포트를 이용하여 GPS 를 추가하는 것을 설명한다.



GPS 기능 구현에 사용된 GPS 모듈은 U-blox NEO-6M GPS 모듈로 5Hz update rate 를 가지면 설정값 저장을 위한 32K I2C EEPROM 을 내장하고 있다. 기본 Protocol 은 NMEA 이고 Baud rate 는 9600bps 이다.

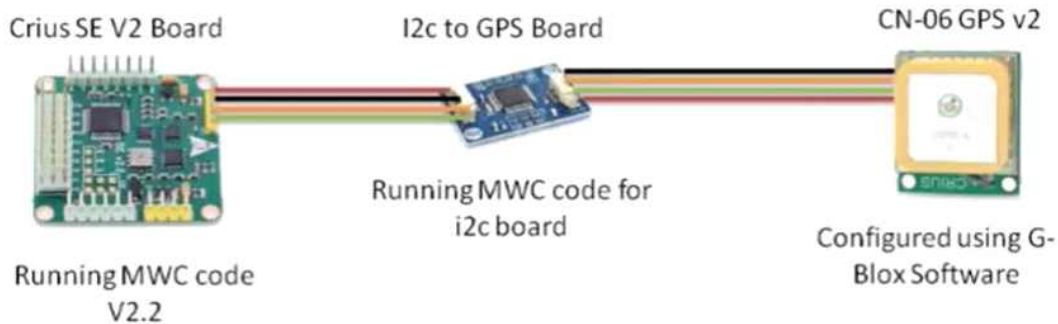
- U-blox NEO-6M GPS module
- 5Hz update rate
- 32k I2C EEPROM for save configuration
- Build in 25 x 25 x 4mm ceramic patch antenna
- LNA and SAW chip
- 3.3V LDO low noise regulator
- UART (TTL) port with EMI protection
- 3V lithium Rechargeable battery
- Power and fix indicator LEDs
- Reverse polarity protection for input power
- Standard NMEA protocol
- Baud rate : 9600
- Nav rate : 1Hz
- TIMEPULSE (fix LED) rate : 1Hz

아두이노 드론 만들기

GPS 모듈의 Serial 을 I2C 로 변환해 주는 모듈은 I2C-GPS NAV Module 을 사용한다.

- All GPS data available via the I2C bus to connect to 328P Multiwii FC
- A LED on board show GPS 3D Fix status
- ATmega 328P Microcontroller
- 2 Molex 1.25mm 4Pin socket for GPS receiver and FC
- 2 port for ISP and FTDI

MultiWii FC와 I2C 및 GPS Module 과의 전체적인 결선은 아래 그림과 같다.



1. GPS Module 의 9600bps NMEA Protocol 을 115200bps UBX protocol로 GPS 설정을 변경 한다. 그리고 I2C-GPS NAV 모듈은 아두이노 스케치를 이용하여 펌웨어를 변경 업로드 한다.

1.1 u-blox u-center GNSS Evaluation Software 를 다운로드 받아 PC 에 설치한다.

: <https://www.u-blox.com/en/product/u-center-windows>

 u-center for Windows, v8.23

1.2 I2C-GPS NAV code를 다운로드 받고 압축을 풀어 둔다. :

I2C_GPS_NAV-v2.1rc1.zip	V2.1rc1 initial release	Deprecated	Jul 13, 2012	28.95KB
---	-------------------------	------------	--------------	---------

<https://code.google.com/p/i2c-gps-nav/>

1.3 설치된 u-blox u-center Software 를 실행한다.

FTDI232R 모듈의 TX/RX에 GPS 모듈의 TX/RX를 서로 Cross(TX-RX, RX-TX)로 연결한 후 PC와 연결한 후 u-center Software 를 실행한다.

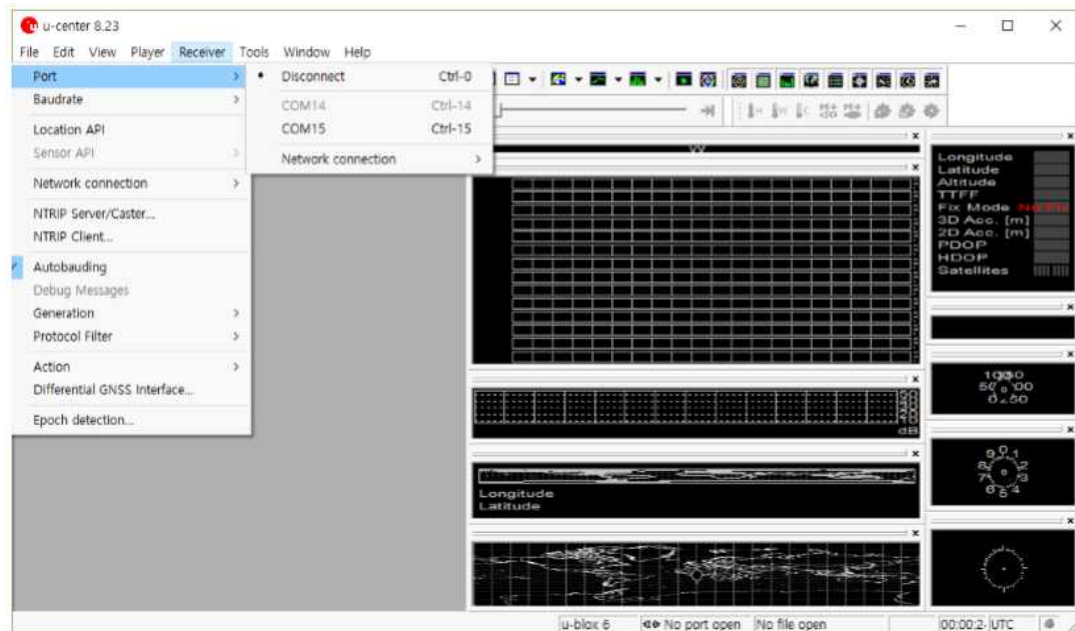
아두이노 드론 만들기



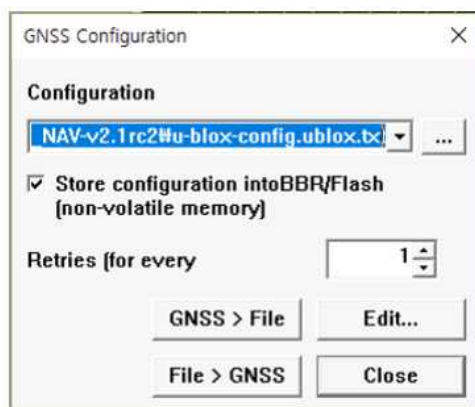
Key things to remember-

- Keep the polarity of the 5v power supply the same
- Swap over the TX and RX cables

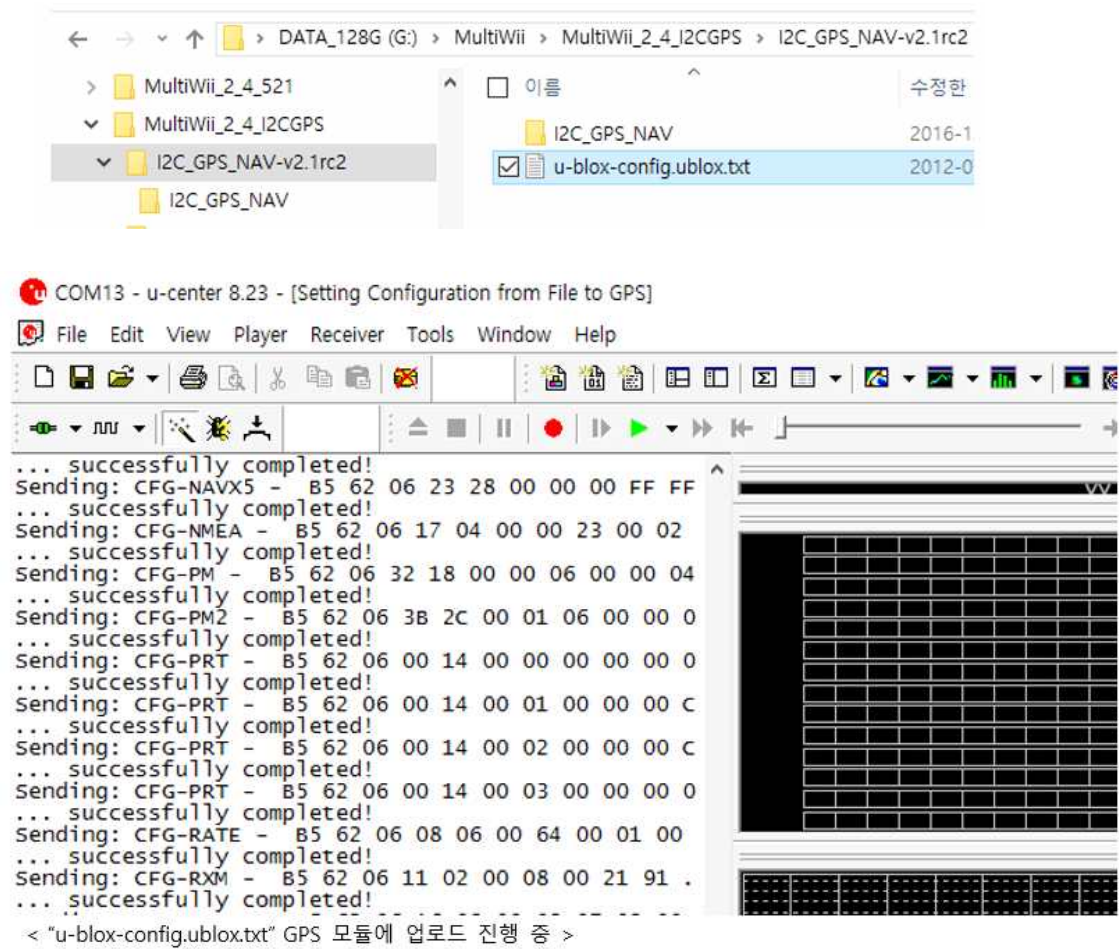
프로그램이 실행되면 FTDI232R과 연결된 COM 포트를 선택하고 Autobauding 도 선택한다.



Tools>GNSS Configuration 을 선택하면 아래 그림과 같이 팝업 창이 나타난다. 파일 선택 버튼을 눌러 다운로드 받아 압축을 푼 폴더(I2C-GPS NAV code)에서 “u-bloxconfig.ublox.txt”를 선택 한 후 “File>GNSS”를 버튼을 눌러 GPS모듈에 업로드를 실행한다.



아두이노 드론 만들기



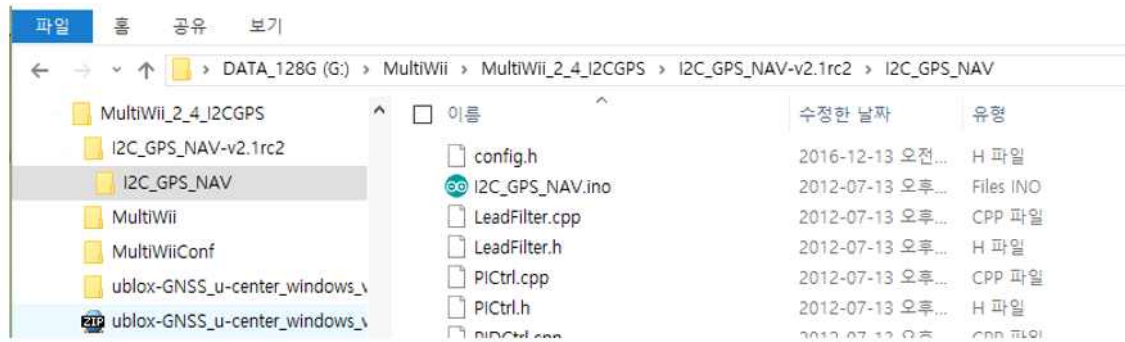
1.4 I2C-GPS NAV code의 수정 및 I2C 모듈에 업로딩

아래 그림과 같이 FTDI232R 의 DTR, RX, TX, VCC, GND 에 I2C 보드의 DTR, TX, RX, VCC, GND 를 연결 한 후 PC 에 연결한다. TX 와 RX 는 서로 Cross 로 연결 하여야 한다.

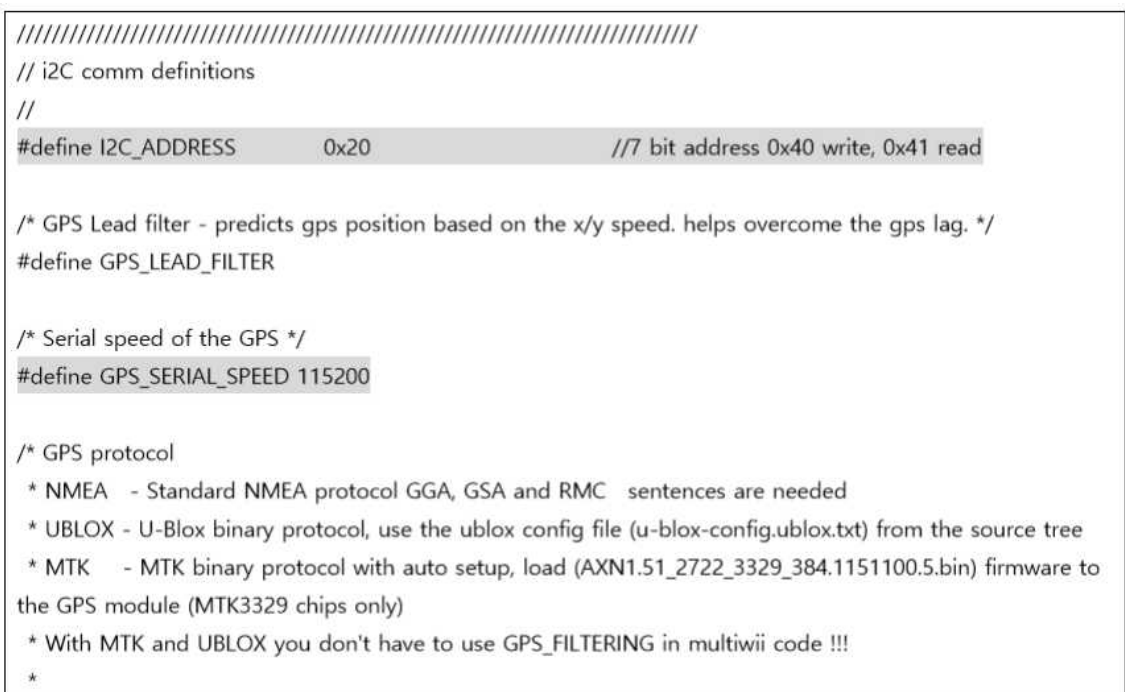


아두이노 드론 만들기

“I2C_GPS_NAV.ino”를 더블 클릭하여 아두이노 스케치를 실행한 후 코드를 수정한다.



Config.h 탭을 선택하여 다음과 같이 수정 및 확인한다.



아두이노 드론 만들기

```

*/

// #define NMEA
#define UBLOX
// #define MTK

// Default PID variables
////////////////////////////////////
// POSHOLD control gains
//
#define POSHOLD_P          .11
#define POSHOLD_I          0.0
#define POSHOLD_IMAX       20           // degrees

#define POSHOLD_RATE_P      1.4         //
#define POSHOLD_RATE_I      0.2         // Wind control
#define POSHOLD_RATE_D      0.010      // try 2 or 3 for POSHOLD_RATE 1
#define POSHOLD_RATE_IMAX   20         // degrees
////////////////////////////////////
// Navigation PID gains
//
#define NAV_P               1.4         //
#define NAV_I               0.20        // Wind control
#define NAV_D               0.006      //
#define NAV_IMAX            20         // degrees

////////////////////////////////////
// Navigation PID gains
//
#define NAV_P               1.4         //
#define NAV_I               0.20        // Wind control
#define NAV_D               0.006      //
#define NAV_IMAX            20         // degrees

////////////////////////////////////
// Navigational parameters and limiters initial values
//
#define CROSSTRACK_GAIN      1          // Weighting the cross track error
#define NAV_SPEED_MIN       100        // cm/sec minimum navigational speed when
NAV_SLOW_NAV is false
#define NAV_SPEED_MAX       300        // cm/sec maximum navigational speed
#define NAV_BANK_MAX        2500       // 20deg max banking when navigating (just for
security and testing)

////////////////////////////////////
// GPS data filtering - moving average filter vector length
//
#define GPS_FILTER_VECTOR_LENGTH 5

```

아두이노 드론 만들기

소스의 수정 및 확인이 끝났으면 컴파일 및 업로드를 실행 한다. 아두이노 스케치의 도구에서 보드는 NANO 보드를 프로세스는 ATmega328 을 선택하여 업로드시 정상적으로 업로드 되는 것을 확인하였다. [도구/보드 - Aduino Nano] [프로세스 - ATmega328] [프로그래머 - AVRISP mkII]

1.5 MultiWii 2.4 Source의 수정(config.h).

마지막으로 MultiWii 소스를 수정한다.



MultiWii 2.4 config.h 의 아래 항목의 찾아서 주석을 해제하여 MultiWii 에서 I2C GPS 를 지원할 수 있도록 한다.

...

```
#define GPS_BAUD 115200
```

...

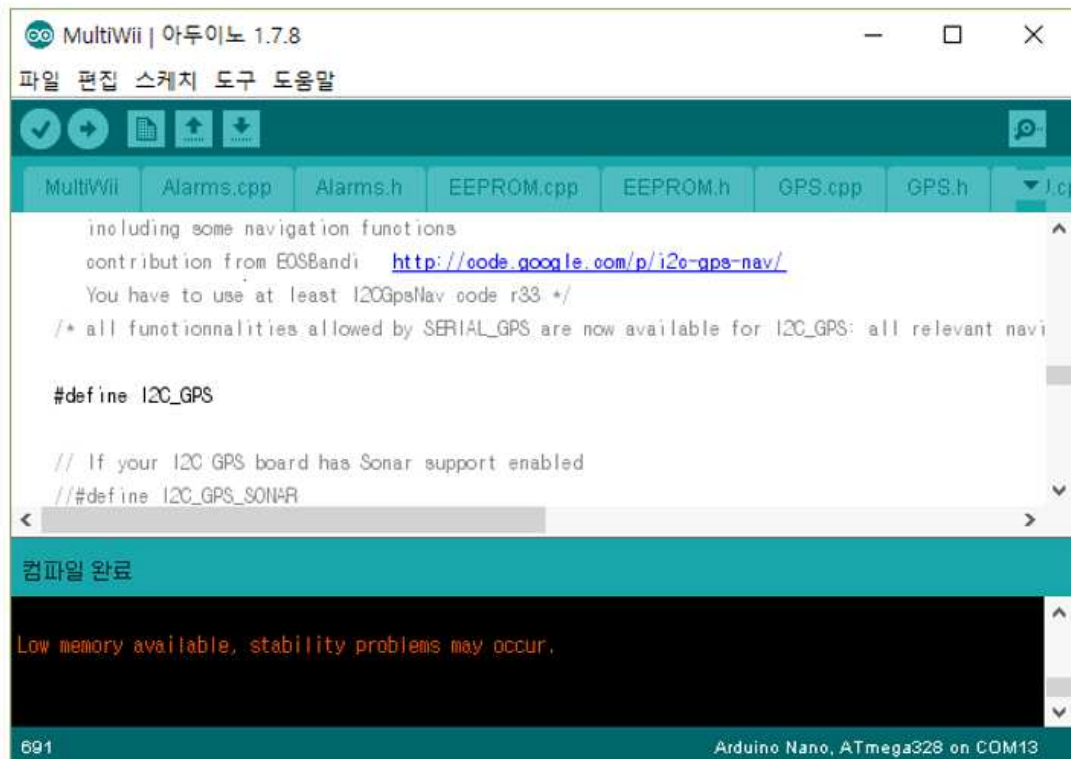
```
#define UBLOX
```

...

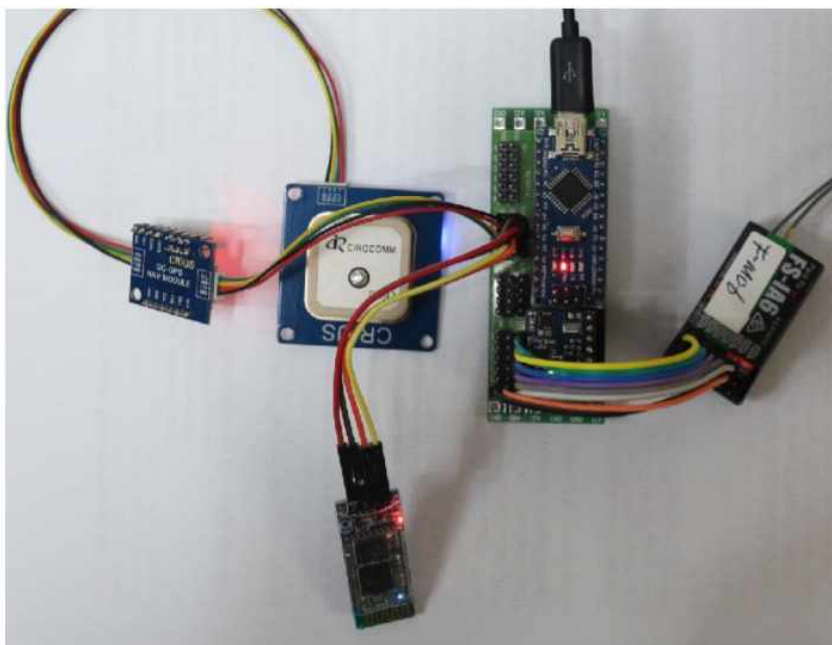
```
#define I2C_GPS
```

수정 후 컴파일을 하면 “Low memory available, stability problems may occur.” 메시지가 나오는데 무시한다.

아두이노 드론 만들기

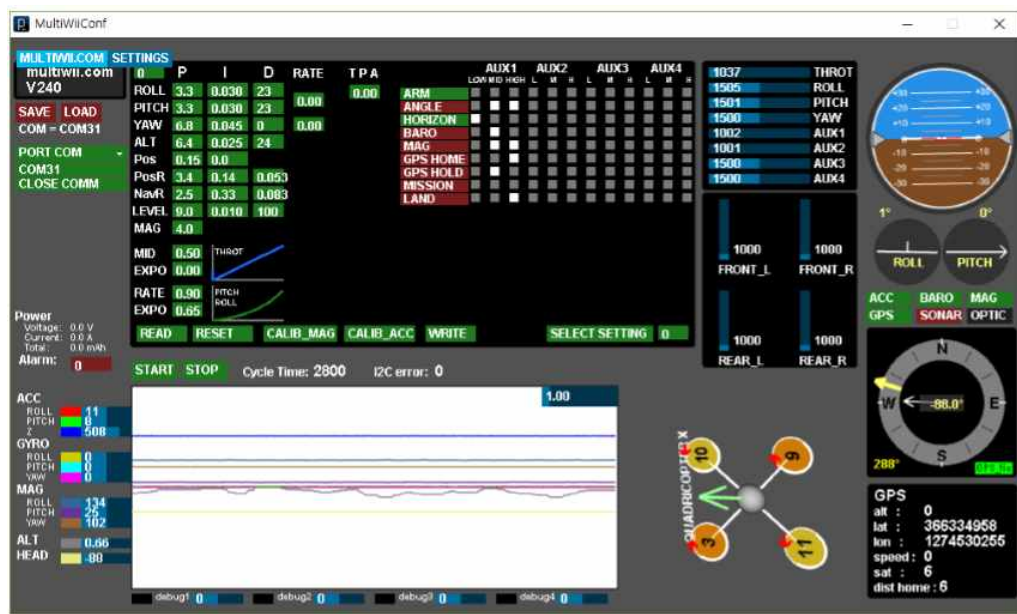


이로써 모든 준비는 완료 되었다. 아래 사진과 같이 Arduino NANO FC 보드에 각각 연결한다. I2C NAV 보드의 SDA, SCL 의 NANO FC 의 SDA, SCL 를 연결한다.



아래 사진과 같이 HC-06 Bluetooth 모듈에 의해 무선으로 MultiWiiConf 와 접속이 되고, GPS 와도 연결이 된 것을 확인 할 수 있다. AUX1 과 AUX 값에 적당한 비행 모드를 할당한 후 Write 버튼을 누른다.

아두이노 드론 만들기



ARM : 시동 여부를 표시

ANGLE : 조종기 스틱이 중앙에 있을 때 기체가 평형을 유지.

HORIZONE : ANGLE mode 와 Acro Mode 의 혼합으로 조종기 스틱이 최대치에서는 Acro mode 로 동작한다.

BARO : Barometer 센서를 사용한 고도유지를 위한 mode.

MAG : compass 센서를 이용한 RTH mode 에서 방향을 유지할 때 이용, GPS 선택시 동시에 선택하여야 한다.

GPS HOME : RTM mode 로 반드시 MAG 선택을 동시에 선택하여야 한다.

GPS HOLD : 비행중 위치를 고정할 때 사용 되며 BARO mode 와 MAG mode 를 동시에 선택

LAND : 자동착륙 기능으로 multiwii 소스의 수정이 필요하다.

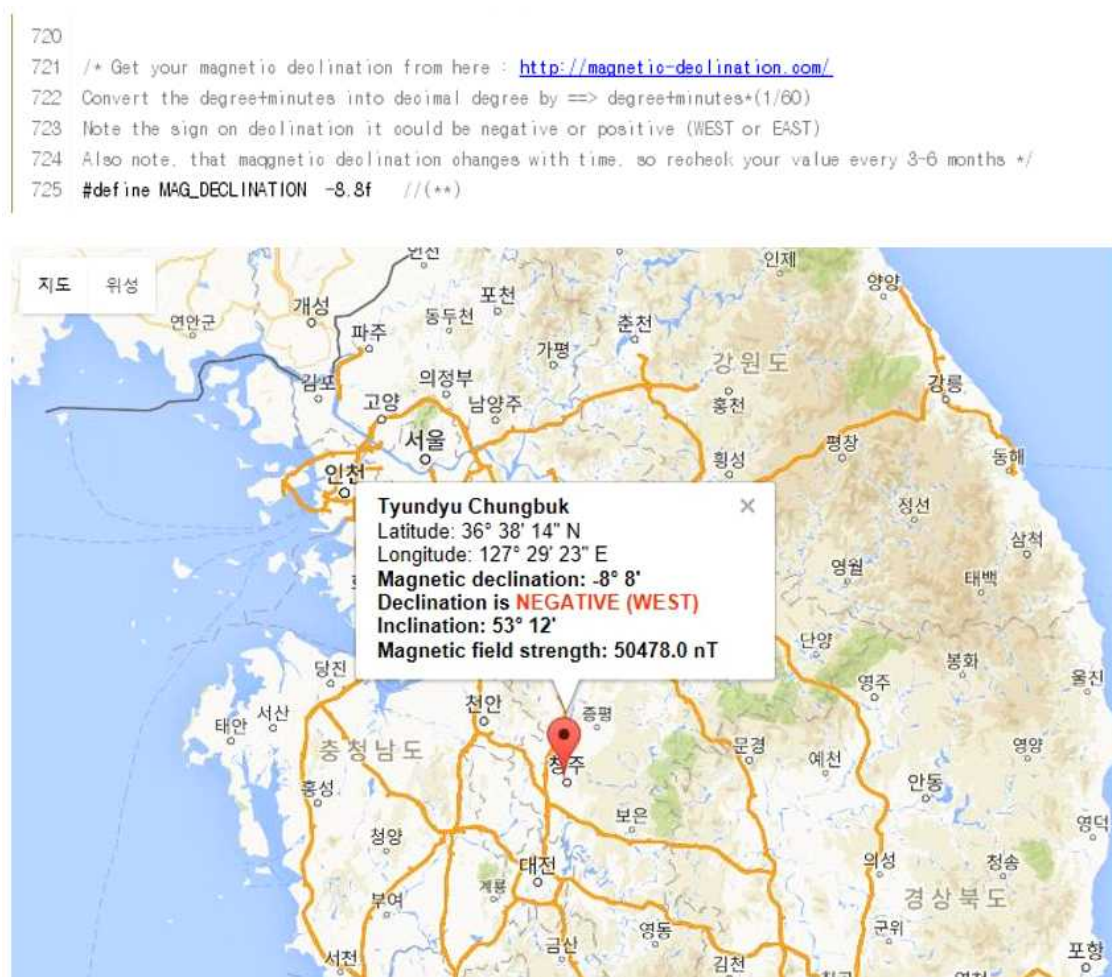
위와 같이 비행 모드를 조종기의 AUX1 에 의해 할 당 할 수 있다.

1.6 MultiWii 2.4 Source의 수정(config.h) for Auto Landing, failsafe 시 포함.

아두이노 드론 만들기

자동 착륙(Auto Landing)을 위해서는 MultiWii 의 소스 수정이 필요하다. 이 또한 Fail Safe 시에도 GPS Home & Auto Landing 이 자동으로 이루어진다. Fail Safety 의 경우는 송신기의 신호를 잃었을 때 또는 Batter fail safe 시의 경우이다.

○ 1 MultiWii 2.4 의 config.h 에서 #define MAG_DECLINATION x.xf 를 링크된 URL 을 클릭하여 현재 위치의 Magnetic declination 값을 확인하여 수정한다. Magnetic declination (방위각)은 항공기 내의 자기장이 지구의 자기장에 상호 작용하여 발생하는 compass 오차의 형태이며 이를 보상하는 것이다.(<http://magnetic-declination.com/>)



○ 2 #define FAILSAFE 를 주석을 해제하여 송신기로부터 신호가 수신 되지 않으면 비상 착륙 하도록 한다. FAILSAFE_DELAY 10 은 1 초를 의미하므로 1 초 이상 신호가 끊기면 작동이 된다. FAILSAFE_DETECT_THRSHOLD 985 는 수신 되는 Throttle 신호가 985 이하 이면 동작되며 Transmitter Failsafe 시의 Throttle value 은 Minthrottle 값에 200 을 더한 값으로 착륙시 모터가 회전한다. 이 또한 자신의 기체에 맞는 값으로 수정할 필요가 있다.

아두이노 드론 만들기

```

597     and motors is stopped. If RC pulse coming back before reached FAILSAFE_OFF_DELAY time, afte
598     #define FAILSAFE // uncomment to activate the failsafe functio
599     #define FAILSAFE_DELAY 10 // Guard time for failsafe activation after
600     #define FAILSAFE_OFF_DELAY 200 // Time for Landing before motors stop in O.
601     #define FAILSAFE_THROTTLE (MINTHROTTLE + 200) // (*) Throttle level used for landing - may
602
603     #define FAILSAFE_DETECT_TRESHOLD 985
604

```

③ AUX 박스를 체크하는 구문을 gps.cpp 파일에 추가한다.(350 line 주석처리, 351 line 추가)

```

348     GPS_adjust_heading():
349     if ((wp_distance <= GPS_conf.wp_radius) || check_missed_wp()) { //if yes switch to poshold mode
350         //if (mission_step.parameter1 == 0) NAV_state = NAV_STATE_HOLD_INFINIT;
351         if ( (mission_step.parameter1 == 0) && (!roOptions[BOXLAND]) ) { NAV_state = NAV_STATE_HOLD_INFINIT; }
352         else NAV_state = NAV_STATE_LAND_START; // if parameter 1 in RTH step is
353         if (GPS_conf.nav_rth_takeoff_heading) { magHold = nav_takeoff_bearing; }
354     }
355     break;

```

○ 4 multiwii.cpp 에 추가 코딩을 아래와 같이 failsafe flag 에 대한 사항을 추가 한다.

```

272
273 #if defined(FAILSAFE_LAND) || defined(FAILSAFE_RTH)
274     bool failsafe_nav = true;
275 #else
276     bool failsafe_nav = false;
277 #endif
278 #if defined(FAILSAFE_IGNORE_LAND)
279     bool failsafe_ignore = true;
280 #else
281     bool failsafe_ignore = false;
282 #endif

```

아두이노 드론 만들기

```

873 // Failsafe routine - added by MIS
874 #if defined(FAILSAFE)
875 /*
876 if ( failsafeCnt > (5*FAILSAFE_DELAY) && f.ARMED) { // Stabilize, and set Throttle to specifi
877     for(i=0; i<3; i++) rcData[i] = MIDRC; // after specified guard time after RC
878     rcData[THROTTLE] = conf.failsafe_throttle;
879     if (failsafeCnt > 5*(FAILSAFE_DELAY+FAILSAFE_OFF_DELAY)) { // Turn OFF motors after specified Time
880         go_disarm(); // This will prevent the copter to automatically rearm if failsafe shuts it down and pre
881         f.OK_TO_ARM = 0; // to restart accidentally by just reconnect to the tx - you will have to switch off fir
882     }
883     failsafeEvents++;
884 }
885 if ( failsafeCnt > (5*FAILSAFE_DELAY) && !f.ARMED) { //Turn of "Ok To arm to prevent the motors from spinnin
886     go_disarm(); // This will prevent the copter to automatically rearm if failsafe shuts it down and pre
887     f.OK_TO_ARM = 0; // to restart accidentally by just reconnect to the tx - you will have to switch off fir
888 }
889 failsafeCnt++;
890 */
891 if ( (GPS_numSat<5) || !f.GPS_FIX || !failsafe_nav ) {
892     if ( failsafeCnt > (5*FAILSAFE_DELAY) && f.ARMED){
893         for(i=0; i<3; i++) rcData[i] = MIDRC;
894         rcData[THROTTLE] = conf.failsafe_throttle;
895         if (failsafeCnt > 5*(FAILSAFE_DELAY+FAILSAFE_OFF_DELAY)){
896             go_disarm();
897             f.OK_TO_ARM = 0;
898         }
899         failsafeEvents++;
900     }
901     if ( failsafeCnt > (5*FAILSAFE_DELAY) && !f.ARMED){
902         go_disarm();
903         f.OK_TO_ARM=0;
904     }
905 }
906 failsafeCnt++;
907 #endif

```


아두이노 드론 만들기

```

1168 //NAV based failsafe
1169 #if defined(FAILSAFE) && (defined(FAILSAFE_RTH) || defined(FAILSAFE_LAND))
1170 if (f_GPS_FIX && GPS_numSat >= 5){
1171     if (failsafeCnt > (5*FAILSAFE_DELAY)&&f_ARMED){
1172         for(i=0; i<3; i++) roData[i] = MIDRC;
1173         if(!((NAV_state == NAV_STATE_LAND_IN_PROGRESS || NAV_state == NAV_STATE_LANDED) && failsafe_ignore)){
1174             roOptions[BOXLAND]=false;
1175             roOptions[BOXGPSHOME]=false;
1176             roOptions[BOXGPSHOLD]=false;
1177             #if defined(FAILSAFE_RTH)
1178                 roOptions[BOXLAND]=true;
1179             #endif
1180             #if defined(FAILSAFE_LAND)
1181                 roOptions[BOXLAND]=true;
1182                 failsafe_ignore=false;
1183             #endif
1184         }
1185         failsafeCnt=(5*FAILSAFE_DELAY);
1186     } else {
1187         #if defined(FAILSAFE_IGNORE_LAND)
1188             failsafe_ignore=true;
1189         #endif
1190     }
1191     failsafeCnt++;
1192 }
1193 #endif
1194
1195 //Generate a packed byte of all four GPS boxes.
1196 uint8_t gps_modes_ohck = (roOptions[BOXLAND]<< 3) + (roOptions[BOXGPSHOME]<< 2) + (roOptions[BOXGPSHOLD]<<1) +
1197

```

○ 3 FENCE_DISTANCE 의 설정 값 이상 이여도 RTM mode 로 동작된다. 본 매뉴얼에서는 600m 에서 300m 로 수정하였다. 또한 LAND_SPEED 는 100 으로 초당 50cm 로 착륙 속도를 설정한다.

```

766 //If FENCE_DISTANCE is larger than 0 then copter will switch to RTH when it farther from home
767 //than the defined number in meters
768 #define FENCE_DISTANCE    300
769
770 //This governs the descent speed during landing. 100 is equals approx 50cm/sec
771 #define LAND_SPEED        100


```

○ 4 MISSION 기능을 사용하기 위해서는 #define USE_MSP_WP 의 주석을 푼다. 하지만 아마도 메모리가 부족하여 컴파일이 되지 않을 것이다. 따라서 메모리를 확보 할 수 있는 소스코드의 정리가 되지 않았다면 MISSION 기능은 사용하지 못하므로 주석을 풀지 말자.

```

709
710 //Enables the MSP_WP command set , which is used by WinGUI for displaying an setting up navigation
711 #define USE_MSP_WP

```



스케치가 너무 큼; 이것을 줄이기 위해 다음을 참고하세요. <http://www.arduino.cc/en/Guide/Troubleshooting#size>

스케치는 프로그램 저장 공간 (101%) 중 31,096 바이트를 사용. 최대 30,720 바이트.

전역 변수는 (86%)의 동적 메모리중 1,766바이트를 사용, 282바이트의 지역변수가 남은. 최대는 2,048 바이트.

processing.app.debug.RunnerException: 스케치가 너무 큼; 이것을 줄이기 위해 다음을 참고하세요. <http://www.arduino.cc/en/Guide/Troubleshooting#size>

```

    at processing.app.debug.Compiler.size(Compiler.java:325)
    at processing.app.debug.Compiler.build(Compiler.java:117)
    at processing.app.Sketch.build(Sketch.java:1169)
    at processing.app.Sketch.build(Sketch.java:1142)
    at processing.app.Editor$BuildHandler.run(Editor.java:1977)
    at java.lang.Thread.run(Thread.java:745)

```

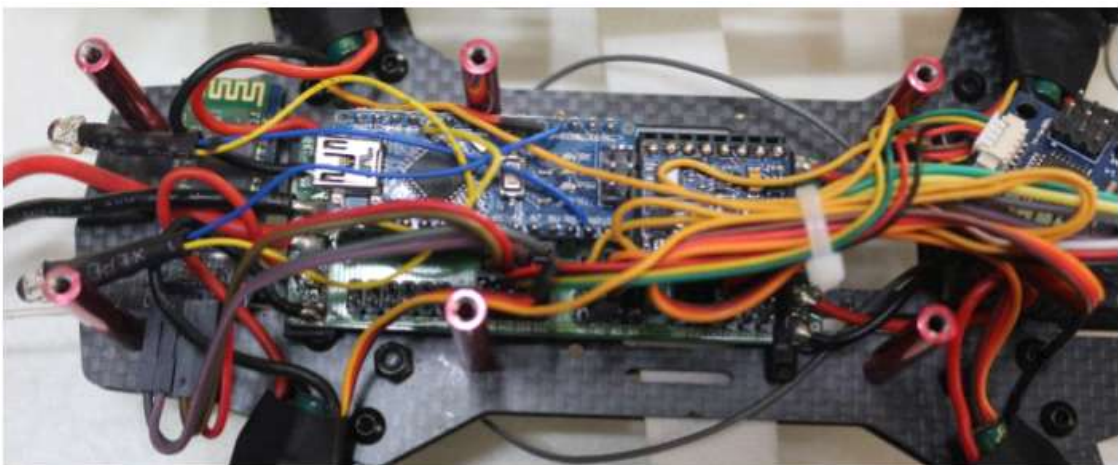
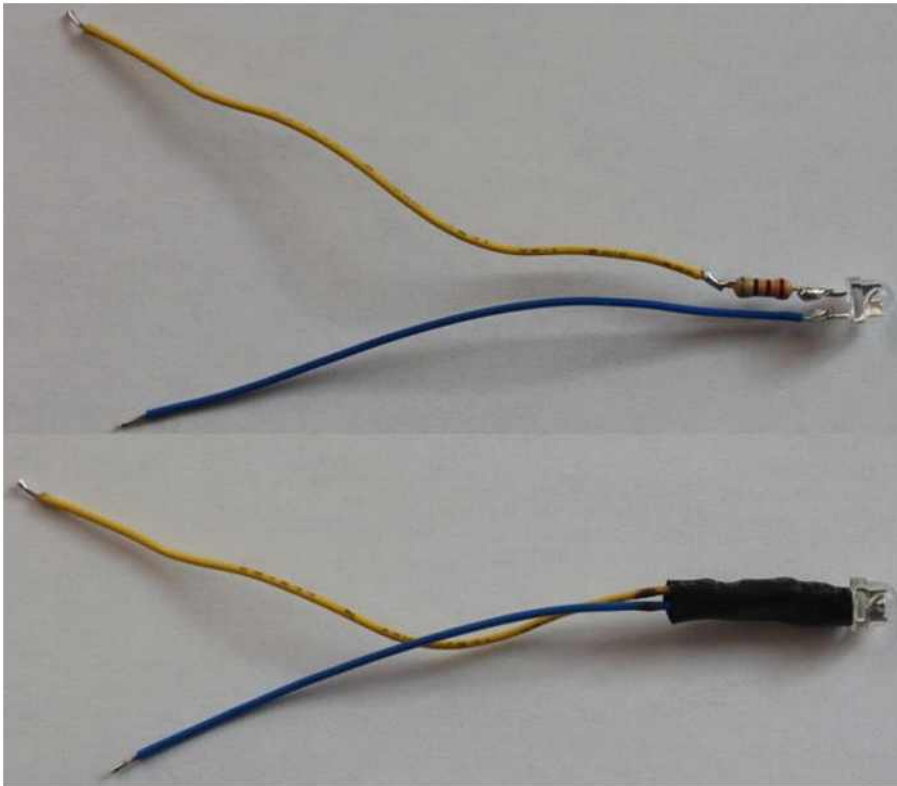
스케치가 너무 큼; 이것을 줄이기 위해 다음을 참고하세요. <http://www.arduino.cc/en/Guide/Troubleshooting#size>

따라서 필요하지 않는 기능들의 소스를 정리하지 않으면 Multiwii2.4 버전에서는 Failsafe 시의 RTH 또는 Auto Landing 기능과 동시에 사용할 수 없다. 물론 사용하지 않는 부분의 소스코드를 정리한다면 가능 할 것이다.

아두이노 드론 만들기

1.7 상태등(Status light) 장착하기(ARM, GPS)

GPS Mode 는 위성의 수가 6 개 이상 이어야 동작을 한다. GPS Module 자체에 부착된 LED 는 위성의 수가 그 이하여도 점멸한다. 따라서 GPS 가 완벽하게 활성화 되지 않은 상태에서 이륙을 한다면 잘못된 위치 정보를 가질 수 있다. 따라서 위성의 수가 6 개 이상인지를 확인 할 수 있는 LED 를 D8 에 부착한다. LED 및 220 Ω 저항으로 납땜 수 수축 튜브로 보호 한 후 Arudino NANO 보드의 GND 와 D8 에 납땜 한다

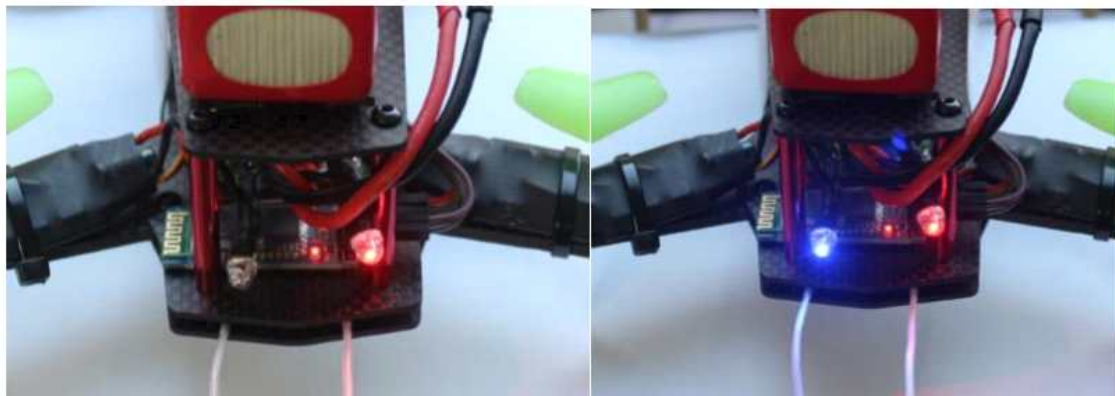


아두이노 드론 만들기

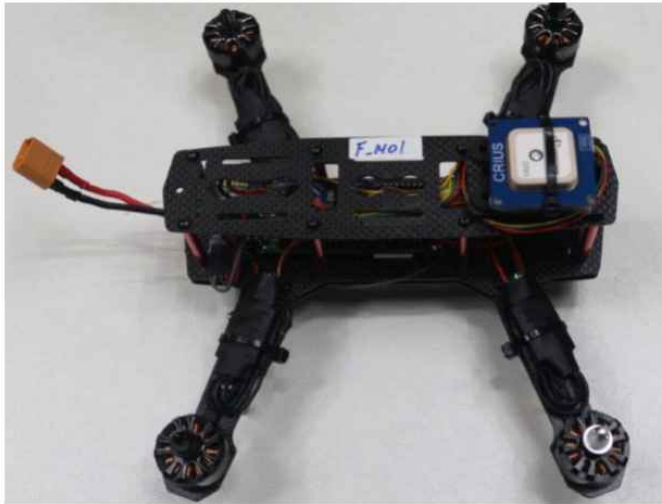
MultiWii 2.4 소스의 “Multiwii.cpp”의 Loop()함수 내에서 위성의 수가 6 개 이상일 때 LED가 켜지도록 아래와 같이 추가한다.

```
MultiWii | Alarms.cpp | Alarms.h | EEPROM.cpp | EEPROM.h | GPS.cpp | GPS.h | IMU.cpp | IMU.h
1135 // This handles the three roOptions boxes
1136 // unlike other parts of the multiwii code, it looks for changes and not based on flag settings
1137 // by this method a priority can be established between gps option
1138
1139 //Generate a packed byte of all four GPS boxes.
1140 uint8_t gps_modes_check = (roOptions[BOXLAND]<< 3) + (roOptions[BOXGPSHOME]<< 2) + (roOptions[BOXGPSHC
1141
1142 if(GPS_numSat > 5 ) { digitalWrite(8,HIGH); } else { digitalWrite(8,LOW); }
1143
1144 if (f.ARMED ) { //Check GPS status and armed
1145 //TODO: implement f.GPS_Trusted flag. idea from Dramida - Check for degraded HDOP and sudden speed
1146 if (f.GPS_FIX) {
1147 if (GPS_numSat > 5 ) {
1148 if (prv_gps_modes != gps_modes_check) { //Check for change since last
1149 NAV_error = NAV_ERROR_NONE;
1150 if (roOptions[BOXGPSHOME]) { // RTL has the priority over e
```

GPS Mode 를 사용할 시 배터리 인가 후 위성의 수가 6 개 이상이 되어서 LED 가 점등 된 후에 이륙을 한다면 좀 더 안정적인 GPS Hold 및 RTL 등이 가능하다. LED 모듈을 하나 더 준비하여 D13 에 연결하면 시동 여부를 알 수 있는 표시등으로 사용할 수 있다.



< 위성 6 개 이상이면 적색 LED, 시동시 청색 LED의 점등 >



TIP3. ESC Calibration

ESC 제조업체의 권장방법으로 Calibration 을 하는 것을 권장함. ESC 와 모터가 연결되어 있고 ESC 전원과 신호선이 수신기에 연결되어 있는 상태에서 조종기의 쓰로틀 스틱을 최대한 상태에서 ESC 에 전원을 인가하면 모터에서 평상시와는 다른 비프음이 들린다. 이 때 바로 조종기의 쓰로틀 스틱을 최소 위치에 놓으면 또 다른 비프음이 들리고 또다시 평상시의 노멀 스타트 비프음이 들리면 ESC Calibration(Max/Min Level)이 끝난 것이다. ESC 의 Calibration 방법은 제조사마다 다를 수 있으므로 제조사의 매뉴얼에 따른다.

TIP4. 모터가 출렁거려서 호버링이 불안하다면

이 경우는 모터에 의한 기체의 진동에 의해 자이로 센서가 영향을 받는 경우일 것이다. 이럴 경우에는 Config.h 에서 자이로의 필터값을 변경해 준다. 진동이 심할수록 낮은 주파수를 선택한다. 그래도 기체가 불안하다면 진동 요인을 제거해 주는 것이 중요하다. 보통은 20MHz 에서도 무난하나 최악의 경우 5HZ 그래도 안되면 진동의 원인을 제거한다.

```

512  /****** Lowpass filter for some gyros ******/
513  /* ITG3200 & ITG3205 Low pass filter setting. In case you cannot eliminate all vibrations to the Gyro, you can try
514     to decrease the LPF frequency, only one step per try. As soon as twitching gone, stick with that setting.
515     It will not help on feedback wobbles, so change only when copter is randomly twitching and all damping and
516     balancing options ran out. Uncomment only one option!
517     IMPORTANT! Change low pass filter setting changes PID behaviour, so retune your PID's after changing LPF.
518     available for ITG3050, ITG3200, MPU3050, MPU6050*/
519  //#define GYRO_LPF_250HZ // This is the default setting, no need to uncomment, just for reference
520  //#define GYRO_LPF_188HZ
521  //#define GYRO_LPF_98HZ
522  //#define GYRO_LPF_43HZ
523  //#define GYRO_LPF_20HZ
524  //#define GYRO_LPF_10HZ
525  #define GYRO_LPF_5HZ // Use this only in extreme cases, rather change motors and/or props — setting not available on ITG3200
    
```


< Mini Quadcopter 250size 의 Reference PID Value >

본 매뉴얼에서 사용된 Q250 Frame 과 ESC, Motor 등을 사용했을 시의 권장되는 PID 값이다. 이는 어디까지나 개인적인 비행 취향에 맞춰진 것이므로 본인의 취향에 맞는 PID 값을 찾을 것을 권장한다. 다만 Reference Value 로 활용하기 바란다.



< F330 Quadcopter 330size 의 Reference PID Value >

본 매뉴얼에서 사용된 F330 Frame 과 ESC, Motor 등을 사용했을 시의 권장되는 PID 값이다. 이는 어디까지나 개인적인 비행 취향에 맞춰진 것이므로 본인의 취향에 맞는 PID 값을 찾을 것을 권장한다. 다만 Reference Value 로 활용하기 바란다.



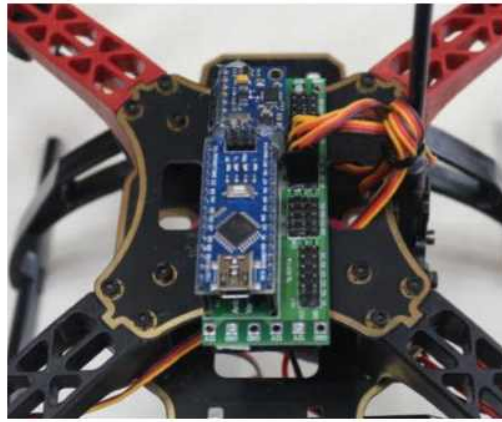
AUX1(CH5)은 조종기의 3 단 스위치를 CH5 에 할당하여 위와 같이 HORIZON, GPS HOLD, GPS HOME, MISSION 등의 비행모드를 제어할 수 있도록 한다. GPS HOME 과 LAND 를 동시에 수행하도록 하여 RTL(Return to launch) LAND 를 수행 할 수 있다.

아래의 사진은 F330 에 NANO FC 를 적용한 사진이므로 조립 시 참조한다.

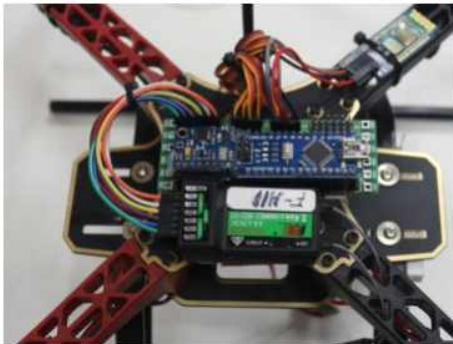


< ESC 의 전원과 XT Power 케이블을 PCB 의 +/-에 납땜한다 >

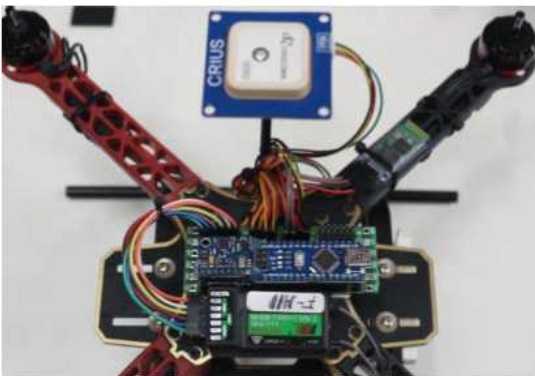
아두이노 드론 만들기



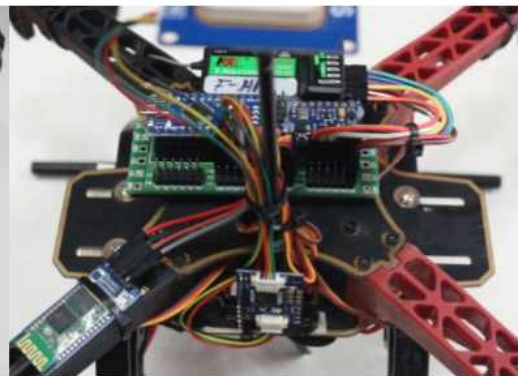
< 양면테이프를 이용하여 NANO FC 를 부착한다>



< 수신기의 CH1~CH5 를 NANO FC 와 결선한다. >



< I2C GPS 모듈을 사진과 같이 장착한다.>



< Barometer 에 빛과 먼지에 의한 오류를 방지하기 위해 검은색 스펀지로 GY-86 MCU 를 덮어 준다.>



- Revision History -

< Crius GPS Module 에러 발생시>

Crius GPS Module 이 “u-blox-config.ublox” GNSS 파일 업로드 에러가 발생하거나, GPS 신호를 잡지 못 할 경우에는 I2C_GPS_NAV.ino 에서 115200bps, ubx 가 아닌 9600bps, NMEA 프로토콜을 사용하자. 간혹 Crius GPS 모듈이 GNSS 파일이 업로드 되지 않거나, 얼마가지 않아 기본 프로토콜인 9600bps, NMEA 로 되돌아가는 현상이 발생하는 경우가 있다. 이럴 경우 기본 프로토콜을 그냥 사용하자.

```
4 // i2c comm definitions
5 //
6 #define I2C_ADDRESS      0x20           //7 bit address 0x40 write, 0x41 read
7
8 /* GPS Lead filter - predicts gps position based on the x/y speed, helps overcome the gps lag. */
9 #define GPS_LEAD_FILTER
10
11 /* Serial speed of the GPS */
12 #define GPS_SERIAL_SPEED 9600
13
14
15 /* GPS protocol
16 * NMEA - Standard NMEA protocol GGA, GSA and RMC sentences are needed
17 * UBLOX - U-Blox binary protocol. use the ublox config file (u-blox-config.ublox.txt) from the source tree
18 * MTK - MTK binary protocol with auto setup. load (AXN1.51_2722_3329_384.1151100.5.bin) firmware to the GP
19 * With MTK and UBLOX you don't have to use GPS_FILTERING in multiwii code !!!
20 *
21 */
22
23 #define NMEA
24 // #define UBLOX
25 // #define MTK
26
```

아두이노 드론 만들기

2016.10.24 - V1.0 : Nano FC 제작 및 MultiWill 및 MultiWillConf 기초 설명 초안 제작

2016.11.17 - V1.1 : Adding BT HC-06 모듈, NANO FC 전용 PCB 기판 내용 추가.

2016.12.16 - V1.2 : Adding I2C GPS 및 GPS Status LED 추가.

2017.08.09 - V1.3 : F330 PID 및 AUX Mode 설정 방법

2018.09.14 - V1.4 : Crius GPS Module 이 “u-blox-config.ublox” GNSS 파일 업로드
에러가 발생하거나, GPS 신호를 잡지 못 할 경우 해결 방법.