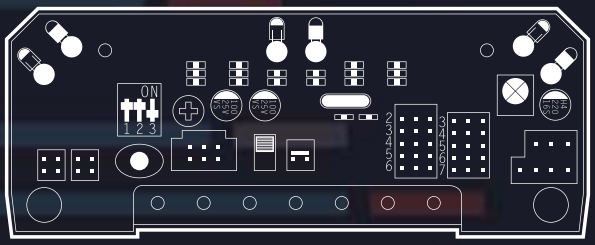
코드론
아두이노 프로그래밍

BUILD, CODE, PLAY



로보링크 주식회사

목 차

Drone Software

1. 왜 SW(소프트웨어) 코딩 교육이 중요할까?

과학 기술의 시대에 머릿속 무언가를 세상에 보여주는 단계에서 SW는 가장 기본적인 수단이 되며, 이것은 생각하며 일을 처리하는 컴퓨팅적 사고능력을 함양할 수 있다.

2. SW 드론 제어?

SW 코딩을 통한 알고리즘 능력은 전천후 서비스 로봇인 드론을 제어 가능하게 함으로써 드론을 활용한 다양한 아이디어 및 *HW 제어능력을 배양한다.

HW(하드웨어, Hardware) : 로봇의 구성물 중 전자 장치, 기계 장치와 같이 형체가 있어 눈에 보이는 것

목표 : SW 코딩 학습과 알고리즘을 통해 드론을 자유롭게 제어하는 능력 개발

- SW 개념과 이해
- 코딩의 시작 : Scratch
- 코딩의 제어 : Arduino
- 드론 제어 SW : Android / IOS / Drone Simulator
- 하드웨어 제어 SW : Rokit Simulator

1. 스마트 인벤터 보드(제어보드)와 코드론(CoDrone) 연결

robolinkSW.com의 아두이노 페이지에서 아두이노와 관련된 프로그램을 다운로드 한다.

1) 스마트 인벤터 보드에 BLE 보드를 연결하여 코드론과 페어링 및 구동을 준비한다.

필요 부품

스마트 인벤터 보드
(제어보드)

BLE 보드

4핀 케이블

마이크로 USB 케이블

2) 프로그래밍을 할 때에는 코드론의 상태와 호버링, 트림 등을 체크한다.

스마트 인벤터 보드에 전원 넣기

USB 케이블 - PC

건전지 케이스

*스마트 인벤터 보드에 건전지나 PC 전원을 이용하여 전원을 넣는다.

1. 스마트 인벤터 보드(제어보드)와 코드론(CoDrone) 연결

*스마트 인벤터 보드 연결하여 실행파일 넣기

프로그램 업로드

스마트 인벤터 보드 (제어보드)

1. 딥스위치 1번을 위로 올린다.
2. 리셋 버튼을 누른다.
3. 파란 LED가 깜빡이는지 확인한다.
4. 만약 파란 LED가 깜빡인다면 "대기 상태"이다. 이 때 스마트 인벤터 보드에 예제를 업로드 한다.

BLE 보드

5. 스위치를 눌러 모드를 전환한다.

업로드 모드 : 스위치를 한 번 누른다.

▶ 노란색 불빛이 깜박인다.

만약 불빛이 빨간색으로 바뀐다면 스위치를 다시 한 번 눌러서 노란색 불빛으로 바꿔준다.

빨간색 불빛은 "**페어링 대기 모드** (드론 연결 모드)"로 **로킷 브릭**이나 **드론 시뮬레이터**와 같이 PC로 조작하는 경우에 사용한다.

프로그램 실행

1. 업로드 완료 후 딥스위치 1번을 내린다. 

2. 리셋 버튼이나 On/Off 스위치로 스마트 인벤터 보드를 리부트하여 보드에 업로드한 프로그램을 실행한다.

2. 코드론을 위한 아두이노 프로그래밍

1) 기초 프로그래밍

<접속 명령>

페어링을 위한 명령

- CoDrone.AutoConnect(NearbyDrone);

<이동 명령>

각 이동 명령의 원하는 범위내의 값을 입력하고 CoDrone.Control();

어떤 상황에서든지 멈추고 싶을 때에는 CoDrone.FlightEvent(Stop);

이름	형식	범위	설명
ROLL	sbyte	-100 ~ 100	좌우 이동 (-)값은 **좌측**, (+)값은 **우측**으로 이동
PITCH	sbyte	-100 ~ 100	전후 이동 (-)값은 **후방**, (+)값은 **전방**으로 이동
YAW	sbyte	-100 ~ 100	좌우 회전 (-)값은 **반시계**, (+)값은 **시계**방향으로 이동
THROTTLE	sbyte	-100 ~ 100	상승·하강 (-)값은 **하강**, (+)값은 **상승**
EVENT	byte	0 ~ 255	이벤트

각 이동 명령의 원하는 범위내의 값을 입력하고 CoDrone.Control();

THROTTLE	**상승·하강**
ROLL	**좌 이동·우 이동**
PITCH	**전진·후진**
YAW	**좌회전·우회전**

2) 예제 프로그래밍

```
#include <CoDrone.h> // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                  // BLE보드의 통신 개시 (115200bps)

  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결

  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)
}

void loop()
{
  byte bt1 = digitalRead(11);      // ■ □ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);      // □ □ □ ■ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt8 = digitalRead(18);      // □ □ □ □ □ □ □ ■  밑면 적외선 센서를 입력으로 사용


   int analogValue0 = analogRead(A0);  // 자기가 원하는 것과 연결가능
   int analogValue1 = analogRead(A1);  // 자기가 원하는 것과 연결가능
   int analogValue2 = analogRead(A2);  // 자기가 원하는 것과 연결가능
   int analogValue3 = analogRead(A3);  // pitch         상 하
   int analogValue4 = analogRead(A4);  // yaw           좌 우
   int analogValue5 = analogRead(A5);  // throttle      위 아래
   int analogValue6 = analogRead(A6);  // roll          회전

   ////////////////////////////◆프로그램부분◆////////////////////////////////


   //이 부분에 본인이 원하는 드론 프로그래밍


   ////////////////////////////////////////////////////////////////////////////

}
```


3. 코드론을 위한 아두이노 프로그래밍 예제

◆프로그램부분◆ 에 대한 부분만 설명

페어링

1) 가장 가까이에 있는 코드론에 페어링 (NearbyDrone)

다운로드 후 실행하면 바로 페어링 구동

```
CoDrone.AutoConnect(NearbyDrone);    // 가장 가까운 위치의 드론과 연결
```

2) 최근에 페어링 한 코드론에 페어링 (ConnectedDrone)

```
CoDrone.AutoConnect(ConnectedDrone);  // 최근에 연결한 드론과 같은 어드레스 주소의
                                      드론을 연결
```

3) 코드론 주소로 페어링 (AddressInputDrone)

확인한 어드레스 주소를 입력할 시에 페어링이 된다

```
CoDrone.PrintDroneAddress();    // 최근에 연결한 드론의 어드레스 주소를 모니터로 출력한다.

byte droneAddress[6] = {0xFC, 0xA6, 0x61, 0x78, 0xD5, 0xA4};
// 어드레스 주소를 입력 - PrintDroneAddress() 명령으로 확인)

CoDrone.AutoConnect(AddressInputDrone, droneAddress);
// 입력한 어드레스 주소와 같은 드론과 연결
```

4) 코드론 주소 확인하기 (PrintDroneAddress)

```
CoDrone.PrintDroneAddress();    // 최근에 연결한 드론의 어드레스 주소를 시리얼 모니터로 출력
```

컨트롤

1) 이/착륙 (TakeOff-Landing)

다운로드 후 실행하면 바로 구동

```
void setup()
{
  CoDrone.begin(115200);                  // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);       // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);        // 드론을 플라이트 모드로 설정한다. (비행형)

  if (PAIRING == true)                    // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);         // 이륙

    delay(2000);                          // 대기 시간

    CoDrone.FlightEvent(Landing);         // 서서히 착륙
  }
}
```

2) 상승/하강 (ThrottleUp)

사용하는 드론의 상태에 따라 Throttle 과 delay 값을 수정한다

```
void setup()
{
  CoDrone.begin(115200);                  // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);       // 가장 가까운 드론과 연결
  CoDrone.DroneModeChange(Flight);        // 드론을 플라이트 모드로 설정한다. (비행형)

  if (PAIRING == true)                    // 연결(페어링)이 성공한 경우에만 실행
  {
    THROTTLE  = 100;                      // THROTTLE 값 입력 100 상승 -100 하강
    CoDrone.Control();                    // 조종값 전송

    delay(2000);                          // 대기 시간 입력

    CoDrone.FlightEvent(Stop);            // 정지
    }
  }
}
```


3) 전진/후진 (PitchUp)

사용하는 드론의 상태에 따라 Throttle 과 delay 값을 수정한다

```
void setup()
{
  CoDrone.begin(115200);                  // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);       // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);        // 드론을 플라이트 모드로 설정한다. (비행형)

  if (PAIRING == true)                    // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);         // 이륙

    delay(2000);                          // 대기 시간

    PITCH = 100;                          // PITCH 값 입력
    CoDrone.Control();                    // 조종값 전송

    delay(500);                           // 대기 시간

    CoDrone.FlightEvent(Landing);         // 서서히 착륙
  }
}
```

4) 좌 이동/우 이동 (RollUp)

사용하는 드론의 상태에 따라 Throttle 과 delay 값을 수정한다

```
void setup()
{
  CoDrone.begin(115200);                  // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);       // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);        // 드론을 플라이트 모드로 설정한다. (비행형)

  if (PAIRING == true)                    // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);         // 이륙

    delay(2000);                          // 대기 시간

    ROLL = 100;                           // Roll 값 입력
    CoDrone.Control();                    // 조종값 전송
```

```
    delay(1000);                    // 대기 시간

    CoDrone.FlightEvent(Landing);   // 서서히 착륙
  }
}
```

5) 좌회전/우회전 (YawUp)

사용하는 드론의 상태에 따라 Throttle 과 delay 값을 수정한다

시계 방향 (Yaw 양수) : 1 ~ 100
반시계 방향 (Yaw 음수) : -1 ~ -100

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);       // 이륙

    delay(2000);                        // 대기 시간

    YAW = 100;                          // Yaw 값 입력
    CoDrone.Control();                  // 조종값 전송

    delay(1000);                        // 대기 시간

    CoDrone.FlightEvent(Landing);       // 서서히 착륙
  }
}
```


6) 터틀턴 (Turn_Over)

코드론이 뒤집어져 있는 상태에서만 작동

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)

  delay(300);                           // 대기 시간

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TurnOver);      // 뒤집기

    delay(4000);                        // 실행 시간

    CoDrone.FlightEvent(Stop);          // 멈춤
  }
}
```

7) 비행 조종 (Controller_Flight)

아날로그 조이스틱으로 드론을 조종하는 예제
비행조종과 주행조종, 배틀 예제의 경우 스마트 보드를 조종기 형태로 조립해서 사용한다.

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)
}

void loop()
{
  byte bt1 = digitalRead(11);           // ■ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);           // □ □ □ ■ □ □ □  밑면 적외선 센서를 입력으로 사용
```

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)
}

void loop()
{
  byte bt1 = digitalRead(11);           // ■ □ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);           // □ □ □ ■ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt8 = digitalRead(18);           // □ □ □ □ □ □ □ ■  밑면 적외선 센서를 입력으로 사용

  if (bt1 && !bt4 && !bt8)              // 밑면 센서 가장 끝 11번 센서에 손을 대면 실행한다.
  {
    CoDrone.FlightEvent(Stop);          // 드론을 정지시킨다.
  }

  if (!bt1 && !bt4 && bt8)              // 밑면 센서 가장 끝 18번 센서에 손을 대면 실행한다.
  {
    CoDrone.FlightEvent(Landing);       // 드론을 착륙시킨다.
  }

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    YAW = -1 * CoDrone.AnalogScaleChange(analogRead(A3));
    // 아날로그 3번 핀의 값을 YAW 값으로 사용한다.          - 좌우회전

    THROTTLE  = CoDrone.AnalogScaleChange(analogRead(A4));
    // 아날로그 4번 핀의 값을 THROTTLE 값으로 사용한다.     - 승·하강

    ROLL = -1 * CoDrone.AnalogScaleChange(analogRead(A5));
    // 아날로그 5번 핀의 값을 ROLL 값으로 사용한다.         - 좌우이동

    PITCH = CoDrone.AnalogScaleChange(analogRead(A6));
    // 아날로그 6번 핀의 값을 PITCH 값으로 사용한다.        - 전·후진

    CoDrone.Control(SEND_INTERVAL);
```



```
    // 제어 신호를 보낸다. 통신이 안정하게 가도록 시간을 두고 보냄 (최소 50ms)
  }
}
```

8) 주행 조종 (Controller_drive)

주의! 바퀴 형태의 로버킷(드라이브 키트)을 별도로 장착해야 한다

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Drive);       // 드론을 드라이브 모드로 설정한다. (바퀴 주행형)
}

void loop()
{
  byte bt1 = digitalRead(11);           // ■ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);           // □ □ □ ■ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt8 = digitalRead(18);           // □ □ □ □ □ □ □ ■  밑면 적외선 센서를 입력으로 사용

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    THROTTLE  = CoDrone.AnalogScaleChange(analogRead(A4));
    // 아날로그 4번 핀의 값을 THROTTLE 값으로 사용한다.     - 전후 이동

    ROLL = -1 * CoDrone.AnalogScaleChange(analogRead(A5));
    // 아날로그 5번 핀의 값을 ROLL 값으로 사용한다.         - 좌우 이동

    CoDrone.Control(SEND_INTERVAL);
    // 제어 신호를 보낸다. 통신이 안정하게 가도록 시간을 두고 보냄(최소 50ms)
  }
}
```

9) 배틀 (Controller_Flight_Battle)

아날로그 조이스틱으로 드론을 조종하고 다른 드론들과 적외선을 발사하며 배틀할 수 있는 예제

```
#include <CoDrone.h>                        // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                    // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);         // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);          // 드론을 플라이트 모드로 설정 (비행형태)

  CoDrone.BattleBegin(FREE_PLAY);
  //팀 선택 : TEAM_RED/TEAM_BLUE/TEAM_GREEN/TEAM_YELLOW/FREE_PLAY
}

void loop()
{
  CoDrone.BattleReceive();                  // IR-Data 수신

  byte bt1 = digitalRead(11);               // ■ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);               // □ □ □ ■ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt8 = digitalRead(18);               // □ □ □ □ □ □ ■  밑면 적외선 센서를 입력으로 사용
  if (bt1 && !bt4 && !bt8)                  // 밑면 센서 가장 끝 11번 센서에 손을 대면 실행한다.
  {
    CoDrone.FlightEvent(Stop);              // 드론을 긴급 정지시킨다.
  }

  if (!bt1 && !bt4 && bt8)                  // 밑면 센서 가장 끝 18번 센서에 손을 대면 실행한다.
  {
    CoDrone.BattleShooting();               // 무기를 발사한다.
    CoDrone.ButtonPreesHoldWait(18);        // 버튼을 뗄때까지 기다린다. (연속적인 발사를 막는다.)
  }

  if (!bt1 && bt4 && !bt8)                  // 밑면 가운데 센서에 손을 대면 실행한다.
  {
    CoDrone.FlightEvent(Landing);           // 드론을 서서히 착륙 시킨다.
  }

  YAW = -1 * CoDrone.AnalogScaleChange(analogRead(A3));
  // 아날로그 3번 핀의 값을 YAW 값으로 사용한다.              - 좌우회전
```



```
  THROTTLE  = CoDrone.AnalogScaleChange(analogRead(A4));
  // 아날로그 4번 핀의 값을 THROTTLE 값으로 사용한다.         - 승·하강

  ROLL = -1 * CoDrone.AnalogScaleChange(analogRead(A5));
  // 아날로그 5번 핀의 값을 ROLL 값으로 사용한다.             - 좌우이동

  PITCH = CoDrone.AnalogScaleChange(analogRead(A6));
  // 아날로그 6번 핀의 값을 PITCH 값으로 사용한다.            - 전·후진

  CoDrone.Control(SEND_INTERVAL);
  // 제어 신호를 보낸다. 통신이 안정하게 가도록 시간을 두고 보냄 (최소 50ms)
}
```

10) 비행 조종 조이스틱 멈춤 (Controller_Flight_joyStop)

```
#include <CoDrone.h>                        // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                    // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);         // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Drive);           // 드론을 드라이브 모드로 설정한다. (바퀴 주행형)
}

void loop()
{
  byte bt1 = digitalRead(11);               // ■ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);               // □ □ □ ■ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt8 = digitalRead(18);               // □ □ □ □ □ □ ■  밑면 적외선 센서를 입력으로 사용

  if ((analogRead(A4) < 50) && (analogRead(A6) < 50))
  // 조이스틱 양쪽을 아래로 내리면 실행한다.
  {
    CoDrone.FlightEvent(Stop);              // 드론을 정지시킨다. (긴급 정지)
  // CoDrone.FlightEvent(Landing);          // 드론을 정지시킨다. (서서히 착륙)
  }

  if (PAIRING == true)                      // 연결(페어링)이 성공한 경우에만 실행
  {
    YAW = -1 * CoDrone.AnalogScaleChange(analogRead(A3));
    // 아날로그 3번 핀의 값을 YAW 값으로 사용한다.              - 좌우회전
```

```
    THROTTLE  = CoDrone.AnalogScaleChange(analogRead(A4));
    // 아날로그 4번 핀의 값을 THROTTLE 값으로 사용한다.  - 승·하강
    ROLL = -1 * CoDrone.AnalogScaleChange(analogRead(A5));
    // 아날로그 5번 핀의 값을 ROLL 값으로 사용한다.              - 좌우이동
    PITCH = CoDrone.AnalogScaleChange(analogRead(A6));
    // 아날로그 6번 핀의 값을 PITCH 값으로 사용한다.             - 전·후진
    CoDrone.Control(SEND_INTERVAL);
    // 제어 신호를 보낸다. 통신이  안정하게 가도록 시간을 두고 보냄(최소 50ms)
  }
}
```

LED 컨트롤

1) LED컨트롤 1 (LED_Color_01)

드론이 LED를 컨트롤 하는 예제

LedColor (Mode, Color, Time) ; 모드, 색상, 시간의 형식으로 입력한다.

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

byte modeTime = 7;                      // 모드 시간 변수
int delayTime = 1000;                   // 대기 시간 변수

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
}

void loop()
{
  CoDrone.LedColor(ArmDimming, Yellow, modeTime);
  // 노랑색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.

  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, Cyan, modeTime);
  // 하늘색으로 밝기 제어하여 천천히 깜빡이며 modeTime 따라 동작한다.

  delay(delayTime);                     // 대기 시간 입력
}
```


2) LED컨트롤 2 (LED_Color_02)

드론의 LED를 컨트롤 하는 예제

LedColor (Mode,R,G,B,Time) ; 모드, R,G,B, 시간의 형식으로 입력한다.

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

byte modeTime = 7;                      // 모드 시간 변수
int delayTime = 1000;                   // 대기 시간 변수

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
}

void loop()
{
  CoDrone.LedColor(ArmDimming, 255, 0, 0, modeTime);
  // 입력된 R,G,B 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, 0, 255, 0, modeTime);
  // 입력된 R,G,B 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, 0, 0, 255, modeTime);
  // 입력된 R,G,B 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, 0, 0, 0, modeTime);
  // 입력된 R,G,B 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력
}
```

3) LED컨트롤 3 (LED_Color_03)

드론의 LED를 컨트롤 하는 예제

LedColor (Mode,color[],Time) ; 모드, 색상 배열, 시간의 형식으로 입력한다.

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

byte modeTime = 7;                      // 모드 시간 변수
int delayTime = 1000;                   // 대기 시간 변수

byte color0[] = {255, 0, 0};            // color0 색상 배열 (R,G,B)
byte color1[] = {0, 255, 0};            // color1 색상 배열 (R,G,B)
byte color2[] = {0, 0, 255};            // color2 색상 배열 (R,G,B)
byte color3[] = {0, 0, 0};              // color3 색상 배열 (R,G,B)

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone); // 가장 가까운 위치의 드론과 연결
}

void loop()
{
  CoDrone.LedColor(ArmDimming, color0, modeTime);
  // color0에 입력된 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, color1, modeTime);
  // color1에 입력된 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, color2, modeTime);
  // color2에 입력된 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력

  CoDrone.LedColor(ArmDimming, color3, modeTime);
  // color3에 입력된 색으로 밝기 제어하여 천천히 깜빡이며  modeTime 따라 동작한다.
  delay(delayTime);                     // 대기 시간 입력
}
```


4) LED 기본값 설정 (LedColorDefault)

드론의 LED를 컨트롤 하는 예제 (전원이 꺼져도 유지된다)
코드론의 눈과 날개 : 두 군데의 LED를 동시에 설정할 수 있습니다.

LedColorDefault (Mode1,color[]1,Time1, Mode2,color[]2,Time2) : 모드1, 색상 배열1, 시간1, 모드2, 색상 배열2, 시간2의 형식으로 입력한다.

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

byte mode1 = ArmHold;                   // 모드1
byte color1[] = {0, 0, 255};            // color1 색상 배열 (R,G,B)
byte modeTime1 = 255;                   // 모드1 시간 변수

byte mode2 = EyeHold;                   // 모드2
byte color2[] = {255, 255, 0};          // color2 색상 배열 (R,G,B)
byte modeTime2 = 255;                   // 모드2 시간 변수

void setup()
{
  CoDrone.begin(115200);                // BLE 보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결

  CoDrone.LedColorDefault(mode1, color1, modeTime2, mode2, color2, modeTime2);
  // color에 입력된 색, mode와 modeTime에 따라 동작
}
```

상태 체크

1) 배터리 레벨체크 (LowBatteryCheck)

배터리 잔량에 따라 부저를 울리는 예제
배터리 값 : 0 ~ 100

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

byte level = 50;                        // 이곳에서 설정한 값이 배터리 기준 값이 됨

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 기능 개시
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
```

```
  CoDrone.LowBatteryCheck(level);
  // 만약 배터리가 입력한 기준 값 보다 작다면 부저를 울려서 알려줌
}
```

2) 드론 자세 체크 (Serial Print Attitude)

드론의 현재 자세 정보 (ROLL, PITCH, YAW) 값을 시리얼 모니터 창에 표시

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 기능 개시
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  delay(500);
}

void loop()
{
  AttitudeToSerialMonitor();            // 자세 시리얼 모니터
  delay(1000);
}

void AttitudeToSerialMonitor()
{
  CoDrone.Send_LinkModeBroadcast(LinkBroadcast_Active);  // 링크 모듈 모드를 엑티브로 변경

  CoDrone.Request_DroneAttitude();
  while (CoDrone.receiveAttitudeSuccess == 0)  // 받은 자세값이 성공적으로 체크되었을 때
  {
    CoDrone.Receive();
  }

  CoDrone.receiveAttitudeSuccess = 0;    //receiveAttitudeSuccess flag init

  CoDrone.Send_LinkModeBroadcast(LinkModeMute);  // 링크 모듈 모드를 뮤트로 변경
  delay(10);

  Serial.println("");
  Serial.println("--------- Now attitude -----------");
  Serial.print("ROLL₩t");
```



```
  Serial.println(AttitudeROLL);
  Serial.print("PITCH₩t");
  Serial.println(AttitudePITCH);
  Serial.print("YAW₩t");
  Serial.println(AttitudeYAW);
}
```

3) RSSI 신호 세기 (RSSI_Polling)

드론의 RSSI(신호세기) 값을 인벤터 보드의 LED 불빛으로 표시

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{

  CoDrone.begin(115200);                // BLE보드의 기능 개시
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DisplayRSSI();                // 연결된 드론의 RSSI(신호세기) 값을 LED
                                        불빛으로 표시
}
```

```
void loop()
{
}
```

4) 고도값 표시 (DisplayLEDRangeSensor) : 코드론 프로(CoDrone Pro) 전용

연결된 드론의 고도 값을 LED로 표시 (고도값은 mm)

```
#include <CoDrone.h>                        // 코드론을 사용하기 위한 헤더파일

int scale = 200;                            // LED 불빛 한개의 기준 고도 200mm
int firstLEDpin = 11;                       // 스마트 보드의 LED 핀 시작 번호
void setup()
{
  CoDrone.begin(115200);                    // BLE보드의 기능 개시
  CoDrone.AutoConnect(NearbyDrone);         // 가장 가까운 위치의 드론과 연결

  delay(500);

  for (int thisPin = 11; thisPin <= 18; thisPin++)
  {
    pinMode(thisPin, OUTPUT);
  }                                         // LED를 모두 출력을 설정

  for (int thisPin = 11; thisPin <= 18; thisPin++)
  {
    digitalWrite(thisPin, LOW);
  }                                         // LED를 모두 끄기로 설정
}

void loop()
{
  CoDrone.Request_Range();
  long oldTime = millis();
   while (CoDrone.receiveRangeSuccess == true)     // 받은 고도값이 성공적으로 체크되었
을 때
  {
    CoDrone.Receive();                      // 데이터 받기
    if (oldTime + 1000 < millis()) break;   // 대기 시간이 길어지면 빠져나옴
  }
```



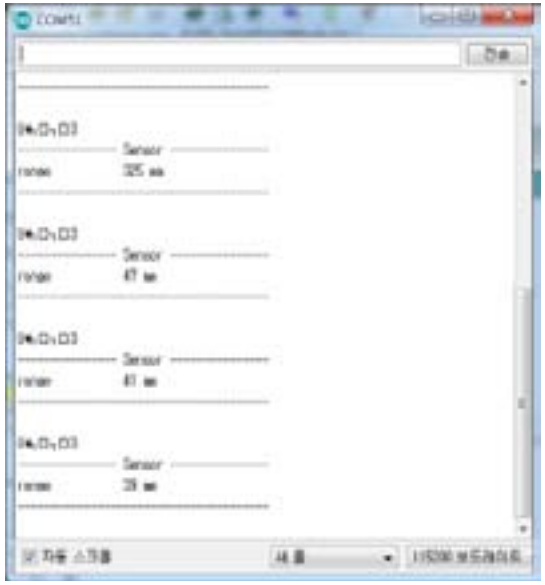
```
  if (CoDrone.receiveRangeSuccess == true)        // 고도값을 성공적으로 받았는지 확인
  {
    int _sensor = CoDrone.sensorRange[5] / scale; // 고도값을 설정한 스케일 값으로 변환
    if (_sensor < 0 )_sensor = 0;                 // LED 불빛의 최소값 체크
    if (_sensor > 7) _sensor = 7;                 // LED 불빛의 최대값 체크
    for (int thisPin = 11; thisPin <= 18; thisPin++)
    {
      digitalWrite(thisPin, LOW);
    }                                             // LED를 모두 끄기로 설정
    digitalWrite(firstLEDpin, HIGH);              // 첫번째 LED 핀을 켬
    for (int i = 1; i < _sensor ; i++)
    {
      digitalWrite(++firstLEDpin , HIGH);
    }                                             // 센서 값에 따라 LED 핀을 켬
  }
}
```

5) 고도값 표시 (SerialPrintRangeSensor) : 코드론 프로(CoDrone Pro) 전용

연결된 드론의 고도 값을 LED로 표시 (고도값은 mm)

```
#include <CoDrone.h>                        // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                    // BLE보드의 기능 개시
  CoDrone.AutoConnect(NearbyDrone);         // 가장 가까운 위치의 드론과 연결
  delay(500);
}
void loop()
{
  RangeSensorToSerialMonitor();             // 고도 값을 시리얼 모니터로 출력하는 함수 실행
}

void RangeSensorToSerialMonitor()           // 고도 값을 시리얼 모니터로 출력하는 함수
{
  CoDrone.Send_LinkModeBroadcast(LinkBroadcast_Active);  // 링크 모듈 모드를 엑티브로 변경
  delay(100);
  CoDrone.Request_Range();

  long oldTime = millis();
```

```
  while (CoDrone.receiveRangeSuccess == true)
  // 받은 고도값이 성공적으로 체크되었을 때
  {
    CoDrone.Receive();                    // 데이터 받기
    if (oldTime + 1000 < millis()) break;    // 대기 시간이 길어지면 빠져나옴
  }

  if (CoDrone.receiveRangeSuccess == true)  // 고도값을 성공적으로 받았는지 확인
  {
    CoDrone.Send_LinkModeBroadcast(LinkModeMute);   // 링크 모듈 모드를 뮤트로 변경
    delay(300);

    // ---------------------------- 센서값 출력 ---------------------------- //

    Serial.println("");
    Serial.println("---------------- Sensor ----------------");
    Serial.print("range tt");
    Serial.print(CoDrone.sensorRange[5]);
    Serial.println(" mm");
    Serial.println("---------------------------------------- ");
    delay(500);
  }
}
```



응용

1) 부메랑 턴 (V_Turn)

부메랑처럼 한바퀴를 돌고 다시 원점으로 돌아오는 예제
(사용하는 드론의 상태에 따라 Roll, Pitch 와 delay 값을 수정한다)

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)

  delay(300);                           // 대기 시간

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);       // 이륙

    delay(1000);                        // 대기 시간

    PITCH = 50;                         // PITCH 값 입력
    ROLL = 50;                          // ROLL 값 입력
    CoDrone.Control();                  // 입력된 값으로 드론 동작

    delay(1000);                        // 대기 시간

    PITCH = 50;                         // PITCH 값 입력
    ROLL = -50;                         // ROLL 값 입력
    CoDrone.Control();                  // 입력된 값으로 드론 동작

    delay(1000);                        // 대기 시간

    PITCH = -50;                        // PITCH 값 입력
    ROLL = -50;                         // ROLL 값 입력
    CoDrone.Control();                  // 입력된 값으로 드론 동작

    delay(1000);                        // 대기 시간
```

```
    PITCH = -50;                    // PITCH 값 입력
    ROLL = 50;                      // ROLL 값 입력
    CoDrone.Control();              // 입력된 값으로 드론 동작

    delay(1000);                    // 대기 시간

    CoDrone.FlightEvent(Landing);   // 서서히 착륙
  }
}
```

2) 원 돌기 (Circle_Turn)

드론의 전면이 중심을 바라보며 원을 한바퀴를 돌고 다시 원점으로 돌아오는 예제
(사용하는 드론의 상태에 따라 THROTTLE, PITCH, ROLL, YAW 와 delay 값을 수정한다)

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)
  delay(300);                           // 대기 시간
  if (PAIRING == true)
  {
    CoDrone.FlightEvent(TakeOff);       // 이륙
    delay(1000);                        // 대기 시간

    THROTTLE = 60;                      // THROTTLE 값 입력
                                        // 90도 반원 회전 4번 반복
    for(int i=0;i<4;i++)
    {
      THROTTLE  = THROTTLE  - 10;       // THROTTLE 값 입력
      YAW = -80;                        // YAW 값 입력
      ROLL = 50;                        // ROLL 값 입력
      CoDrone.Control();                // 조종값 전송
      delay(1500);                      // 대기 시간
    }
    CoDrone.FlightEvent(Stop);          // 멈춤
  }
}
```


3) 손바닥으로 점프 (hand_on)

드론이 짧게 이륙과 전진을 하여 사용자의 손에 떨어지는 예제
(사용하는 드론의 상태에 따라 PITCH 와 delay 값을 수정한다)

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);       // 이륙
    delay(1000);                        // 대기 시간

    PITCH = 30;                         // PITCH 값 입력
    CoDrone.Control();                  // 조종값 전송
    delay(300);                         // 대기 시간

    CoDrone.FlightEvent(Stop);          // 서서히 착륙
  }
}
```

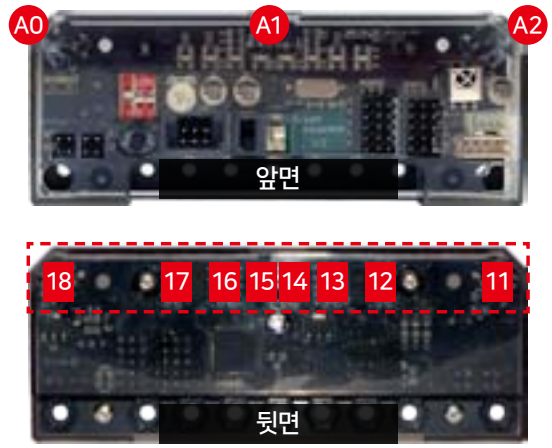
4) 장애물 점프 (hurdle_jump)

드론이 이륙과 전진을 하여 장애물을 넘어 착륙하는 예제
(사용하는 드론의 상태에 따라 THROTTLE, PITCH, ROLL 과 delay 값을 수정한다)

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)

  delay(400);                           // 대기 시간

  if (PAIRING == true)                  // 연결(페어링)이 성공한 경우에만 실행
  {
    CoDrone.FlightEvent(TakeOff);       // 이륙
    delay(1000);                        // 대기 시간
                                        // 상승
    THROTTLE = 20;                      // THROTTLE 값 입력
    CoDrone.Control();                  // 조종값 전송

    delay(500);                         // 대기 시간
                                        // 전진
    THROTTLE = 0;                       // THROTTLE 전송
    PITCH = 100;                        // PITCH 전송
    ROLL = -10;                         // ROLL 전송
    CoDrone.Control();                  // 조종값 전송

    delay(300);                         // 대기 시간
                                        // 제자리 비행
    THROTTLE = 0;                       // THROTTLE 전송
    PITCH = 0;                          // PITCH 전송
    ROLL = 0;                           // ROLL 전송
    CoDrone.Control();                  // 조종값 전송

    delay(500);                         // 대기 시간

    CoDrone.FlightEvent(Landing);       // 착륙
  }
}
```


5) 아날로그 센서 컨트롤 (Analog_Control)

3개의 아날로그 센서로 드론을 조종하는 예제



- 인벤터 보드의 아날로그 센서 3개와 디지털센서 3개를 이용한 조종 예제
- bt4 (14,15)를 누르면 A1은 높이조절 A0, A2는 좌우 이동
- bt8 (18)을 누르면 A1은 높이조절 A0, A2는 전·후진
- bt1 (11)을 누르면 드론 정지

```
#include <CoDrone.h>                    // 코드론을 사용하기 위한 헤더파일

void setup()
{
  CoDrone.begin(115200);                // BLE보드의 통신 개시 (115200bps)
  CoDrone.AutoConnect(NearbyDrone);     // 가장 가까운 위치의 드론과 연결
  CoDrone.DroneModeChange(Flight);      // 드론을 플라이트 모드로 설정한다. (비행형)
}

void loop()
{
  byte bt1 = digitalRead(11);           // ■ □ □ □ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt4 = digitalRead(14);           // □ □ □ ■ □ □ □  밑면 적외선 센서를 입력으로 사용
  byte bt8 = digitalRead(18);           // □ □ □ □ □ □ ■  밑면 적외선 센서를 입력으로 사용

  int analogValue0 = analogRead(A0);    // 자기가 원하는 것과 연결가능
  int analogValue1 = analogRead(A1);    // 자기가 원하는 것과 연결가능
```

```
  int analogValue2 = analogRead(A2);      // 자기가 원하는 것과 연결가능

  if (bt1 && !bt4 && !bt8)                // 밑면 센서 가장 끝 11번 센서에 손을 대면 실행한다.
  {
    CoDrone.FlightEvent(Stop);            // 드론을 정지시킨다.
  }

  if (bt4)                                // 밑면 센서 중앙 14번 센서에 손을 대면 실행한다.
  {
    THROTTLE  = map(analogValue1, 0, 1023, -50, 200);  // 상승·하강
    ROLL = map(analogValue0, 0, 1023, 100, 0) + map(analogValue2, 0, 1023, -100, 0);
    //  좌우이동
    CoDrone.Control(SEND_INTERVAL);   // 제어 신호를 보낸다. 시간을 두고 보냄 (최소 50ms)
  }

  if (bt8)                                // 밑면 센서 가장 끝 18번 센서에 손을 대면 실행한다.
  {
    THROTTLE  = map(analogValue1, 0, 1023, -50, 200);  // 상승·하강
    PITCH = map(analogValue0, 0, 1023, -100, 0) + map(analogValue2, 0, 1023, 100, 0);
    // 전진·후진
    CoDrone.Control(SEND_INTERVAL);   // 제어 신호를 보낸다. 시간을 두고 보냄 (최소 50ms)
  }
}
```

6) 디지털 센서 컨트롤 (Button_Control)

7개의 디지털 센서로 드론을 조종하는 프로그램

```
#include <CoDrone.h>                      // 코드론을 사용하기 위한 헤더파일
int slowUp = 80;                          // 천천히 올라가는 정도의 호버링 값
int slowDown = -100;                      // 천천히 내려가는 정도의 호버링 값
int slowTime = 10;                        // 천천히 움직이는 속도 조절 : 커지면 천천히 올라감

void setup()
{
  CoDrone.begin(115200);

  CoDrone.AutoConnect(NearbyDrone);  // 가장 가까운 드론과 연결
  CoDrone.DroneModeChange(Flight);    // 드론을 플라이트 모드로 설정한다. (비행형)

  delay(300);
```



```
  // 밑면 센서 모두 입력으로 사용
  pinMode(11, INPUT);
  pinMode(12, INPUT);
  pinMode(13, INPUT);
  pinMode(14, INPUT);
  pinMode(15, INPUT);
  pinMode(16, INPUT);
  pinMode(17, INPUT);
  pinMode(18, INPUT);

  delay(100);
}

void loop()
{
  byte bt1 = digitalRead(11);
  byte bt2 = digitalRead(12);
  byte bt3 = digitalRead(13);
  byte bt4 = digitalRead(14);                 //14 & 15
  byte bt6 = digitalRead(16);
  byte bt7 = digitalRead(17);
  byte bt8 = digitalRead(18);

  //*************************** Slow Up ***************************//
  //■ □ □ □ □ □ □  1 번째 센서에 손대면 천천히 상승
  if (bt8 && !bt7 && !bt6 && !bt4 && !bt3 && !bt2 && !bt1)
  {
    THROTTLE = slowUp;
    CoDrone.Control();
    delay(slowTime);
    THROTTLE = 0;
    CoDrone.Control();
  }
  //*************************** Slow Down ***************************//
  //□ □ □  □ □ □  ■  7 번째 센서에 손대면 천천히 하강
  else if (!bt8 && !bt7 && !bt6 && !bt4 && !bt3 && !bt2 && bt1)
  {
    THROTTLE = slowDown;
    CoDrone.Control();
    delay(slowTime);
```

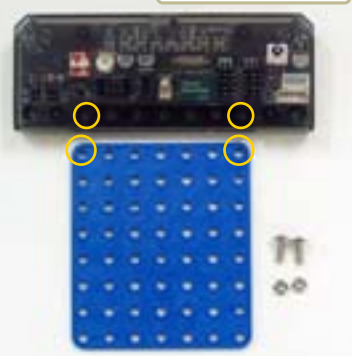
```
  THROTTLE = 0;
  CoDrone.Control();
}

//*************************** STOP ****************************//
//□ □ □ ■ □ □ □   밑   센서 4번째 감지시 Stop
else if (bt4)
{
  CoDrone.FlightEvent(Stop);
}
//*************************** Control ****************************//
//□ ■ □ □ □ □ □ 밑면 2 번째 센서 감지시 전진
//Forward
else if (!bt8 && bt7 && !bt6 && !bt4 && !bt3 && !bt2 && !bt1)
{
  PITCH = 40;
  CoDrone.Control();
}
// □ □ □ □ □ ■ □ 밑면 6 번째 센서 감지시 후진
// Back
else if (!bt8 && !bt7 && !bt6 && !bt4 && !bt3 && bt2 && !bt1)
{
  PITCH = -40;
  CoDrone.Control();
}
// □ □ ■ □ □ □ □ 밑면 3 번째 센서 감지시 왼쪽
// Left
else if (!bt8 && !bt7 && bt6 && !bt4 && !bt3 && !bt2 && !bt1)
{
  ROLL = -40;
  CoDrone.Control();
}
// □ □ □ □ ■ □ □   밑면 5 번째 센서 감지시 오른쪽
// Right
else if (!bt8 && !bt7 && !bt6 && !bt4 && bt3 && !bt2 && !bt1)
{
  ROLL = 40;
  CoDrone.Control();
}
  delay(10);
}
```


4. 코드론 컨트롤러(Controller) 조립

1) 제어보드(스마트 인벤터 보드)와 피트 조립하기

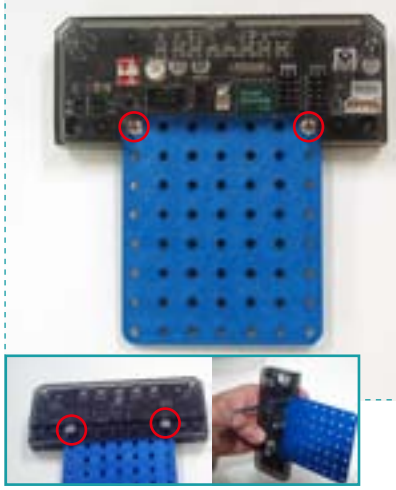
제어보드의 조립 부분에 맞게 피트를 올려준다.

보드 조립 위치 :



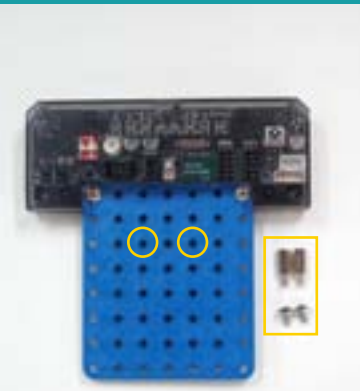
*보드 뒷면에 너트를 먼저 맞춰놓으면 앞면 볼트 체결이 더욱 수월하다!

피트의 양 모서리와 제어보드를 긴 볼트로 조립한다.

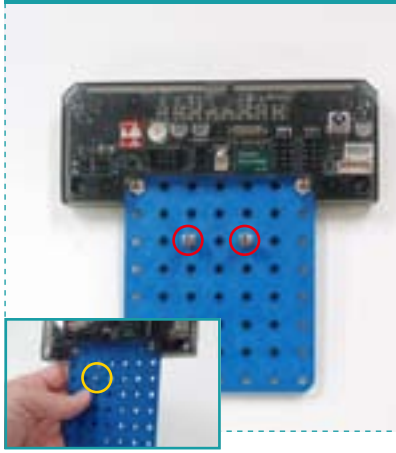


2) 피트에 BLE 보드 장착을 위한 지지대 조립하기

짧은 지지대와 일반 볼트를 각각 2개씩 준비한다.



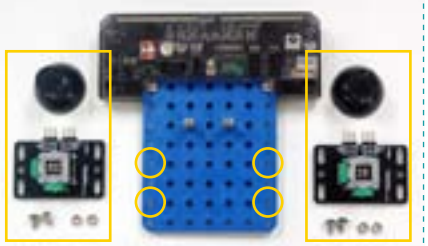
짧은 지지대의 꼬리부분이 위를 향하도록 아래 사진 위치에 조립한다.



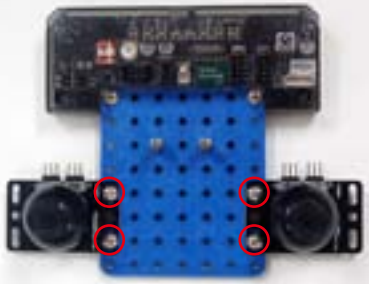
4. 코드론 컨트롤러(Controller) 조립

3) 조이스틱 모듈 조립하기

조이스틱 모듈 2개, 일반 볼트 4개,
너트 4개를 준비한다.



아래의 사진처럼 조이스틱 모듈을
피트에 조립한다.



4) 건전지 케이스 조립하기 (뒷면 조립)

3)번까지 조립한 컨트롤러와 건전지 케이스
1개, 긴 볼트 2개, 너트 2개를 준비한다.

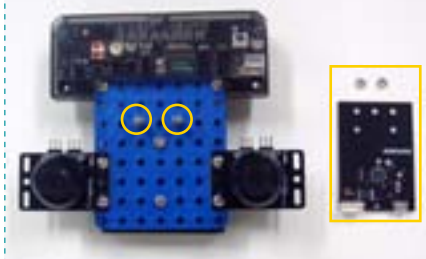
짧은 지지대의 꼬리부분이 위를
향하도록 아래 사진 위치에 조립한다.

코드론 소프트웨어 - Arduino (아두이노)

4. 코드론 컨트롤러(Controller) 조립

5) 건전지 케이스와 BLE 보드 조립하기

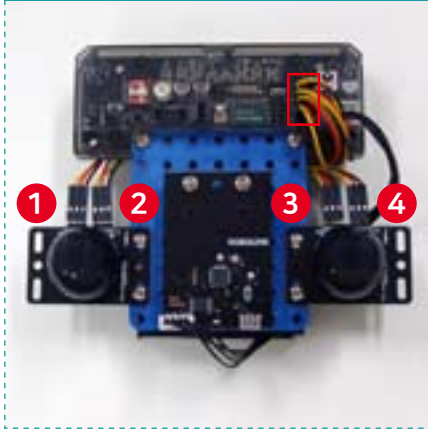
4)번까지 조립한 컨트롤러와 BLE 보드
1개, 너트 2개를 준비한다.

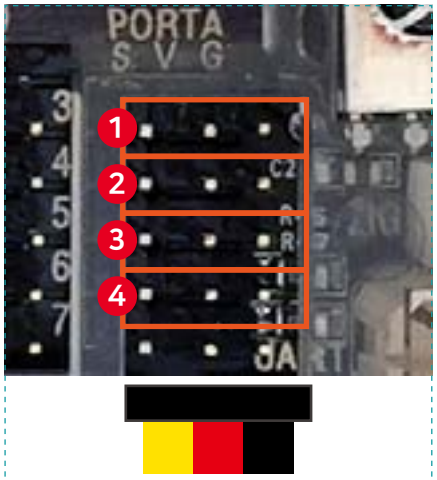


1)번에서 조립한 지지대 위에
BLE 보드를 올려 너트로 조립한다.

6) 제어보드(스마트 인벤터 보드) 배선하기

1, 2, 3, 4 번호에 맞게 순서대로
연결한다.





전선 방향 - 노랑 : S / 빨강 : V / 검정 : G

MEMO

코드론 소프트웨어 - Arduino (아두이노)

MEMO

RobolinkSW.com

- 온라인 사이트를 통해 매뉴얼과 프로그램 다운로드, 교육 자료 등을 확인해주세요.
- 로보링크 SW / Scratch / Arduino 등이 오픈 소스로 제공됩니다.