Arduino Control Guide

ARDUINO

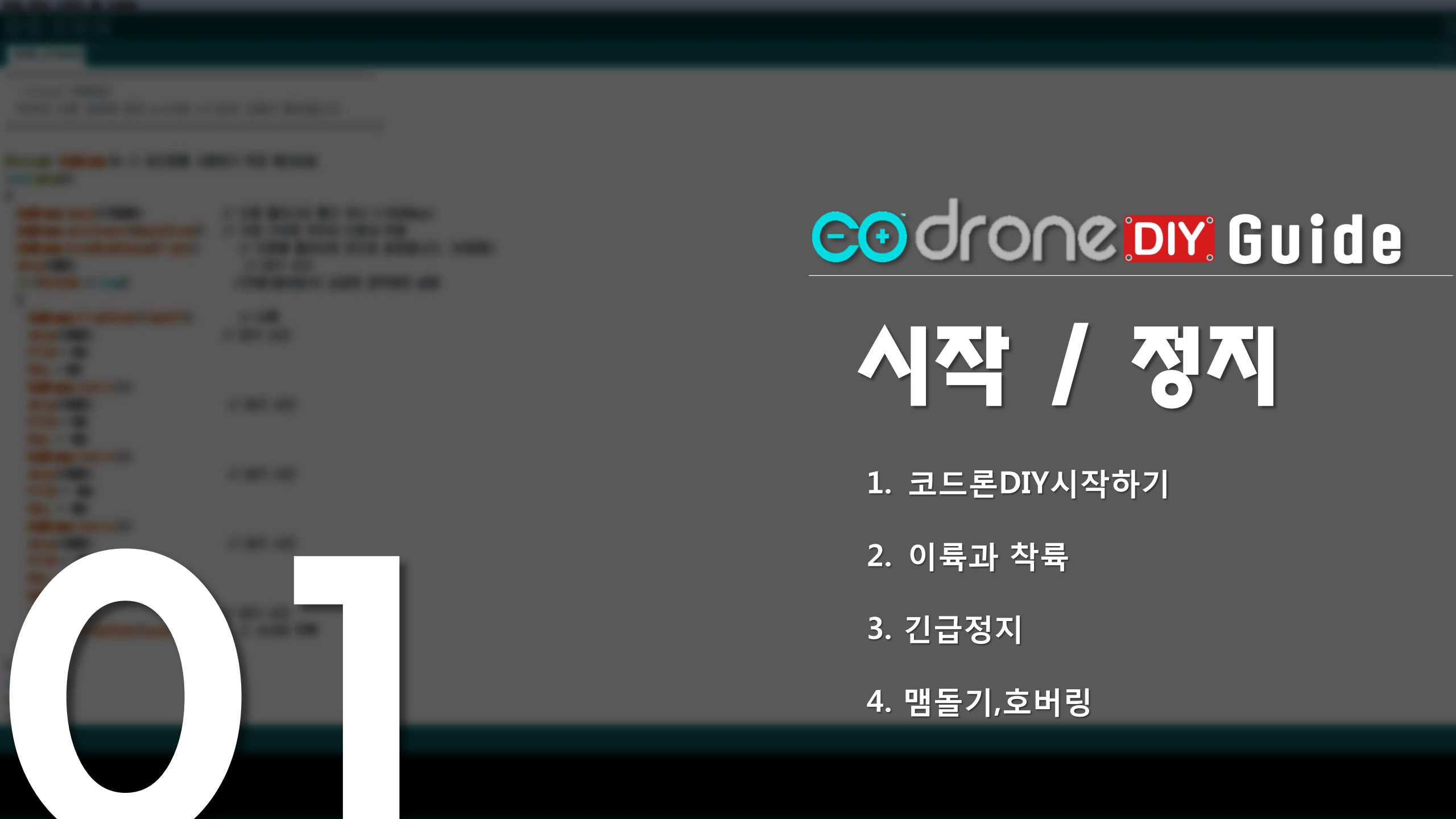
×

codrone DIY

ROBOLINK

INDEX

코드론DIY를 활용한 아두이노 기본 교육 가이드



edrone DIY Guide

시작 / 정지

01

1. 코드론DIY시작하기
2. 이륙과 착륙
3. 긴급정지
4. 맴돌기,호버링

1. Begin
(코드론 DIY 시작하기)

- 목표 코드론 DIY 시작하기에 대한 사용법
- 함수 begin ()
  Begin (지속시간)

설명

이 기능은 **코드론2를 사용하기 위한 비행 준비와 초기화를 진행하는 명령**입니다.
코드론2의 기능을 사용하기 위해서 꼭 필요한 함수입니다.

- 지속 시간을 사용하는 경우

지속 시간에 해당하는 **"초(seconds)"**를 입력하면 사용할 수 있습니다.

지속 시간 입력은 작성한 비행 프로그램에 영향을 주는 내용은 아닙니다.
지속시간은 안전을 이유로 제작된 기능입니다.

- **실수로 프로그램에 착륙이나 정지를 입력하지 않은 경우**
- **비행 시간을 너무 길게 입력한 경우**
- **무한히 반복하거나 비행하도록 작성한 경우**

이렇게 유저가 프로그램을 작성하다가 실수를 하는 경우가 있습니다.
이런 경우에 드론이 망가지거나 사고가 발생할 수 있습니다.

지속 시간을 입력하게 되면 프로그램을 실행한 이후부터 시간을 측정하여 그 시간이 지나면 자동으로 드론이 착륙하도록 합니다.

지속 시간 기능을 사용하여 **안전하게 비행**하도록 합니다.

2. Take Off And Land
(이륙과 착륙)

- **목표** 이륙과 착륙의 사용 방법
- **함수** Take off()
  land()
- **설명** 이륙은 드론이 상승하여 호버링을 시작하게 합니다.

  이륙 후에 드론은 다음 명령을 실행하기 전에 안정화를 위해 항상 5 초 동안의 대기 시간을 주어야 합니다.

  착륙시에 드론은 바닥을 인식하여 서서히 내려오게 되고 바닥에 가까워지면 멈추게 됩니다.

```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 10;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 10초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  CoDrone2.land();                  // 바닥으로 천천히 착륙
}

void loop()
{
}
```

3. EmergencyStop (긴급정지)

- **목표** 이륙과 착륙의 사용 방법
- **함수** Take off()
  land()
- **설명** 이 기능은 드론의 모든 명령을 중지하고 모든 모터를 정지 시켜서 드론을 정지시킵니다.

  비상 정지는 착륙과는 다르게 모터를 즉시 정지시킵니다.
  긴급 정지는 안전을 위한 중요한 기능입니다.

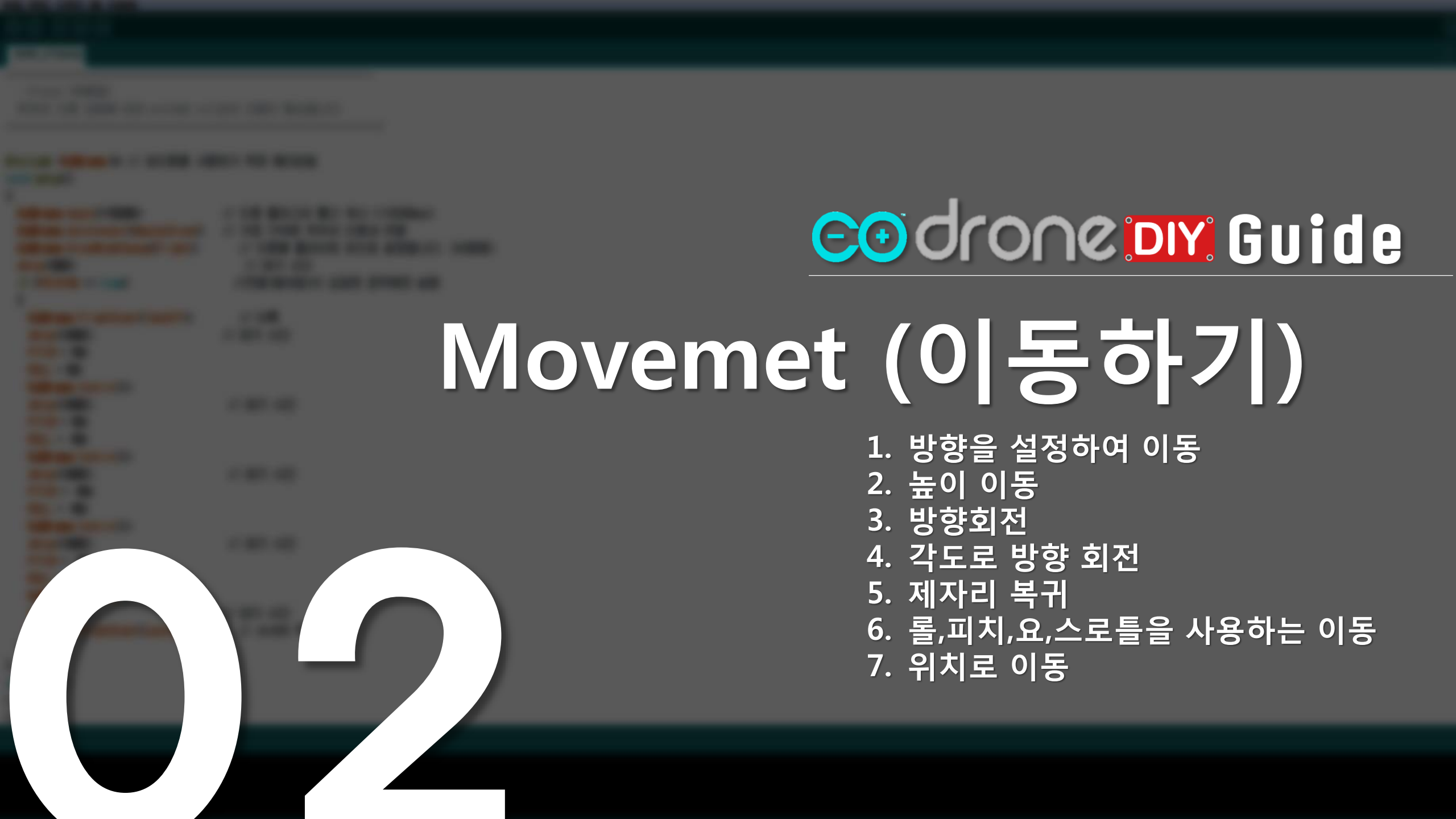
  이 예제에서는 모터가 바로 멈추는지 확인합니다.

```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 10;          // 비행 지속 시간을 10초로 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 10초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙

  delay(1000);                      // 1초간 기다리기

  // 이륙을 실행하였지만, 긴급 정지 명령이 바로 이어지므로
     멈추게 됩니다.
  CoDrone2.emergencyStop();      // 긴급 정지
}

void loop()
{
}
```

4. Hover
(맴돌기,호버링)

- **목표** 맴돌기에 대한 사용법
- **함수** hove()
- **설명** 이 기능을 사용하면 주어진 시간동안 드론의 고도와 위치를 유지합니다.
  이륙시에는 안정적인 동작을 위해 최소 5초 동안 hover 명령을 꼭 사용해야 합니다.
  (비슷한 명령으로는 delay()가 있습니다.)

```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 20;          // 비행 지속 시간을 20초로 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();     // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(10);     // 10초간 맴돌기 (호버링)

  CoDrone2.land();        // 바닥으로 천천히 착륙
}

void loop()
{
}
```



edrone DIY Guide

Movemet (이동하기)

1. 방향을 설정하여 이동
2. 높이 이동
3. 방향회전
4. 각도로 방향 회전
5. 제자리 복귀
6. 롤,피치,요,스로틀을 사용하는 이동
7. 위치로 이동

02

1. Movement
(이동하기)

01 드론을 이동시키는 명령입니다.

02 다양한 방식의 이동 방법을 배웁니다.

03 고도나,방향을 설정하거나 좌표를 입력하여 이동을 시킬수도 있습니다.

1. go
(방향을 설정하여 이동)

- **목표** 방향을 설정하여 이동에 대한 사용법
- **함수** go (방향, 출력, 지속시간)
- **설명** 쉬운 방식의 비행 이동입니다.
  입력한 방향으로 지속 시간과 힘에 따라 이동합니다.
  이 예제는 드론이 이륙하여 앞으로 전진한 후에 착륙합니다.

- FORWARD : 전진
- BACKWARD : 후진
- LEFT : 왼쪽 이동
- RIGHT : 오른쪽 이동
- UP : 상승
- DOWN : 하강

```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 10;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 10초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  int direction = FORWARD;
  int power = 30;
  float durationTime = 0.5;

  // 앞쪽 방향으로 30%의 힘으로 0.5초간 이동
  CoDrone2.go(direction, power, durationTime);

  CoDrone2.land();                  // 바닥으로 천천히 착륙
}

void loop()
{}
```

2. goToHeight
(높이 이동)

- **목표** 높이 이동에 대한 사용법
- **함수** goToHeight()
- **설명** 이 기능은 드론의 아래에 있는 바닥 혹은 물체를 기준으로 해당 높이로 이동하게 합니다.
  0.0 ~ 4.0 미터(m) 범위 내에서 작동합니다.
  바닥의 IR 센서를 사용하여 높이를 계속 확인 합니다.

  이 예제는 드론이 이륙한 후에, 높이를 변경하고 착륙하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 20;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();           // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(7);            // 7초간 맴돌기 (호버링)

  CoDrone2.goToHeight(1.5);     // 1.5m 높이로 이동
  CoDrone2.hover(3);            // 2초간 맴돌기 (호버링)

  CoDrone2.goToHeight(0.8);     // 0.8m 높이로 이동
  CoDrone2.hover(3);            // 2초간 맴돌기 (호버링)

  CoDrone2.land();              // 바닥으로 천천히 착륙
}

void loop()
{
}
```

3. turn
(방향 회전)

- **목표** 방향회전에 대한 사용법
- **함수** turn(회전방향, 지속시간, 출력)
- **설명** yaw를 움직이는 쉬운 방식의 방향 회전입니다. 입력한 방향으로 지속 시간과 힘에 따라서 회전합니다.

  - LEFT : 왼쪽 회전
  - RIGHT : 오른쪽 회전

  이 예제는 드론이 입력한 힘과 방향으로 회전하는 예제입니다

```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 10;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 10초 뒤에 착륙

  CoDrone2.takeoff();           // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);            // 5초간 맴돌기 (호버링)

  int direction = RIGHT;
  int power = 30;
  float duration = 3;

  // 오른쪽 방향으로 3초간 30%의 힘으로 회전
  CoDrone2.turn(direction, power, duration);

  CoDrone2.land();              // 바닥으로 천천히 착륙
}

void loop()
{
}
```

4. turnDegree
(각도로 방향 회전)

- **목표** 각도로 방향 회전에 대한 사용법
- **함수** turnDegree(회전방향, 각도, 지속시간)
- **설명** 입력한 방향의 각도로 드론을 회전시킵니다.
  입력한 각도와 지속시간에 따라 회전속도가 정해집니다.

  - LEFT : 왼쪽 회전
  - RIGHT : 오른쪽 회전

  이 예제는 드론이 입력한 방향으로 입력한 각도를 회전하는 예제입니다

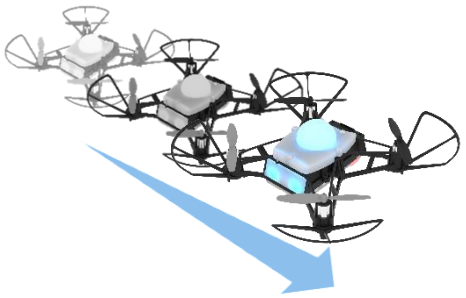
```
#include "CoDrone2.h"

void setup()
{
  int flightDuration = 10;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 10초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  int direction = RIGHT;
  float degree = 90;
  int time = 3;

  // 오른쪽 방향으로 90도 각도를 3초안에 회전
  CoDrone2.turnDegree(direction, degree, time);

  CoDrone2.land();                  // 바닥으로 천천히 착륙
}

void loop()
{
}
```

5. returnHome (제자리 복귀)

- **목표** 제자리 복귀 기능의 사용 방법
- **함수** returnHome()
- **설명** 이 기능은 드론이 이륙했던 위치로 이동하여 착륙하는 명령입니다.

  이 예제는 드론이 이륙한 후에 이동하고, 다시 제자리로 돌아와 착륙합니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();          // 비행을 위한 초기화와 준비

  CoDrone2.takeoff();        // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);         // 5초간 맴돌기 (호버링)

  CoDrone2.go(FORWARD, 1);  // 1초간 20%의 힘으로 전진

  CoDrone2.hover(2);         // 2초간 맴돌기 (호버링)

  CoDrone2.returnHome();     // 드론이 이륙한 지점으로 이동

  CoDrone2.hover(2);         // 2초간 맴돌기 (호버링)

  CoDrone2.land();           // 바닥으로 천천히 착륙
}

void loop()
{
}
```



H 리턴홈

5. move
(롤, 피치, 요, 스로틀을 사용하는 이동)

- **목표** Move의 사용 방법
- **함수** move(롤, 피치, 요, 스로틀)
  move(롤, 피치, 요 ,스로틀, 지속시간)
- **설명** 이 기능을 사용하여보다 복잡한 비행 동작을 만들 수 있습니다.
  롤, 피치, 요, 스로틀을 사용하여 드론을 제어합니다.
  지속시간을 입력하는 경우에는 해당하는 명령을 입력한 지속시간동안만 유지합니다.
  지속 시간이 지나면 롤, 피치, 요, 스로틀값이 0이 되며 호버링 상태를 유지하게 됩니다.

```
#include "CoDrone2.h“

int roll = 0;
int pitch = 0;
int yaw = 0;
int throttle = 0;
float durationTime = 0.0;

void setup(){
  int flightDuration = 20;          // 비행 지속 시간 설정
  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙
  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요
  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)
  //---------- 변수를 사용한 입력 방법 ----------------------------------//
  roll = 0;               // 롤 값을 0으로 설정      (롤 현재 상태 유지)
  pitch = 0;              // 피치 값을 0으로 설정    (피치 현재 상태 유지)
  yaw = 30;               // 요 값을 30으로 설정     (왼쪽으로 회전)
  throttle = 0;           // 스로틀 값을 0으로 설정  (스로틀 현재 상태 유지)
  durationTime = 1.0;     // 지속 시간 1초 입력      (1초간 명령 실행)
  // 1초동안 30만큼의 출력으로 왼쪽으로 회전
  CoDrone2.move(roll, pitch, yaw, throttle, durationTime);
  //----------- 지속 시간만큼 명령을 실행하는 경우 ----------------------------//
  // 1초동안 -30만큼의 힘으로 오른쪽으로 회전
  CoDrone2.move(0, 0, -30, 0, 1.0);   // 1초동안 30만큼의 힘으로 전진
  CoDrone2.move(0, 30, 0, 0, 1.0);   // 1초동안 30만큼의 힘으로 후진
  CoDrone2.move(0, -30, 0, 0, 1.0);
  //----------- 지속 시간이 없이 명령을 실행하는 경우 ---------------------------//
  // -30만큼의 힘으로 오른쪽으로 회전
 // (새로운 명령이 올때까지 계속해서 비행 - 무제한의 시간)
  CoDrone2.move(0, 0, -30, 0);
  delay(2000);    //2초 기다리기
  //-------------------------------------------------------------------------//
  // 이동을 멈추고 현재 상태 유지 (모든 값을 0으로 설정)
  CoDrone2.move(0, 0, 0, 0);
  CoDrone2.hover(1);  // 1초간 맴돌기 (호버링)
  CoDrone2.land();                  // 바닥으로 천천히 착륙}
void loop(){}
```

5. move
(롤, 피치, 요, 스로틀을 사용하는 이동)

- 지속시간: 비행의 지속시간, 지속 시간이 입력되지 않는 경우에는 입력한 값을 유지한채로 비행을 지속 합니다.

- roll (롤) : 좌우 이동의 출력이며 범위는 -100 에서 100 기본 값은 0이며, 양수(+)는 우이동, 음수(-)는 좌이동입니다.

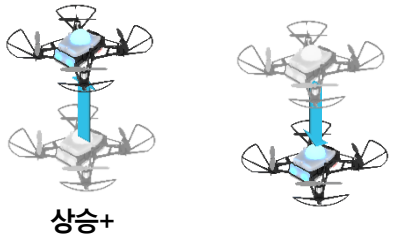
설명

- pitch (피치) : 앞뒤 이동의 출력이며 범위는 -100 에서 100 기본 값은 0이며, 양수(+)는 전진, 음수(-)는 후진입니다.

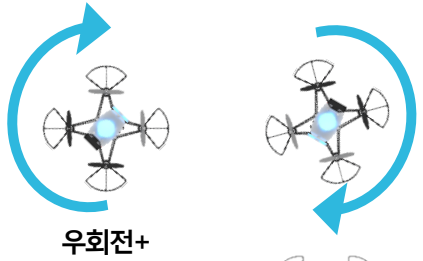
- yaw (요) : 좌우 회전의 출력이며 범위는 -100 에서 100 기본 값은 0이며, 양수(+)는 왼쪽, 음수(-)는 오른쪽 회전입니다.

throttle (스로틀) : 상승 하강의 출력이며 범위는 -100 에서 100 기본 값은 0이며, 양수(+)는 상승, 음수(-)는 하강입니다.

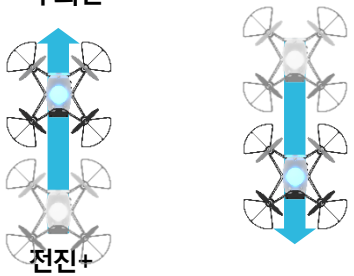
throttle : 상하 수직 이동



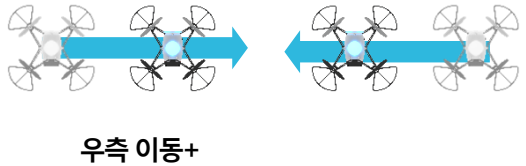
yaw : 좌회전, 우회전 이동



pitch : 전진, 후진 이동



roll : 좌측, 우측 이동



6. goPosition (위치로 이동)

- **목표** goPosition의 사용 방법
- **함수** 오른쪽 표와 같습니다.
- **설명** 해당하는 위치로 이동합니다.
  원하는 위치와 시간에 따라 드론의 움직임을 제어할 수 있습니다.

  입력한 이동 방향과 이동거리, 이동시간에 따라 이동합니다.
  입력한 회전 방향과 회전 방향, 회전 시간에 따라서 회전합니다.
  드론의 현재 기준으로 앞, 뒤, 왼쪽, 오른쪽, 위, 아래의 방향을 나타냅니다.

  여러 축(x,y,z)의 이동 명령을 동시에 할 수 있습니다.
  이동 거리는 m 단위입니다.
  이동 시간이 함께 사용됩니다

```
// 한 축의 방향

goPosition(이동방향1, 이동거리1, 이동시간)

// 두축의 방향

goPosition(이동방향1, 이동거리1, 이동방향2, 이동거리2, 이동시간)

// 세축의 방향

goPosition(이동방향1, 이동거리1, 이동방향2, 이동거리2,
           이동방향3, 이동거리3, 이동시간)

// 회전

goPosition(회전방향, 회전각도, 회전시간)

// 회전 + 한축의 방향

goPosition(이동방향1, 이동거리1, 이동시간, 회전방향, 회전각도, 회전시간)

// 회전 +  두축의 방향

goPosition(이동방향1, 이동거리1, 이동방향2, 이동거리2,
           이동시간, 회전방향, 회전각도, 회전시간)

//회전 +  세축의 방향

goPosition(이동방향1, 이동거리1, 이동방향2, 이동거리2, 이동방향3,
           이동거리3, 이동시간, 회전방향, 회전각도, 회전시간)
```

6-1. goPosition
(이동방향 1, 이동거리1, 이동시간)

● 설명 이 예제는 드론이 앞과 뒤로 이동합니다.

```
#include "CoDrone2.h"

int direction = 0;
int distance = 0;
int time = 0;
```

```
void setup()
{
  int flightDuration = 20;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  // - 이동 방향과 거리, 시간 입력
  //---------------------------- 앞쪽으로 이동 ----------------------------//

  direction = FORWARD;  // 방향 : 전진
  distance = 1;         // 거리 : 1m
  time = 2;             // 시간 : 2초

  CoDrone2.goPositon(direction, distance, time);
// (FORWARD 1m) 를 2초 안에 이동
  CoDrone2.hover(1);                                // 1초간 맴돌기 (호버링)

  //---------------------------- 뒤쪽으로 이동 ----------------------------//

  CoDrone2.goPositon(BACKWARD, 1, 2);  // (BACKWARD 1m) 를 2초 안에 이동
  CoDrone2.hover(1);                   // 1초간 맴돌기 (호버링)


  CoDrone2.land();                  // 바닥으로 천천히 착륙
}
void loop()
{
}
```

6-2. goPosition
(이동방향1, 이동거리1, 이동방향2, 이동거리2 이동시간)

설명 이 예제는 드론이 앞과 오른쪽 대각선방향로 뒤와 왼쪽 대각선 방향으로 이동합니다.

```
#include "CoDrone2.h"

int direction1 = 0;
int distance1 = 0;

int direction2 = 0;
int distance2 = 0;

int time = 0;
```

```
void setup()
{
  int flightDuration = 20;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  // - 이동 방향과 거리, 시간 입력

  //------------ 앞과 오른쪽 대각선 방향으로 이동 ----------------------------//

  direction1 = FORWARD;  // 방향 : 전진
  distance1 = 1;         // 거리 : 1m

  direction2 = RIGHT;  // 방향 : 오른쪽
  distance2 = 1;         // 거리 : 1m

  time = 2;             // 시간 : 2초

  CoDrone2.goPositon(direction1, distance1, direction2, distance2, time);
// (FORWARD 1m) (RIGHT 1m) 를 2초 안에 이동
  CoDrone2.hover(1);                                // 1초간 맴돌기 (호버링)

  //---------------- 뒤와 왼쪽 대각선 방향으로 이동 ----------------------------//

  CoDrone2.goPositon(BACKWARD, 1, LEFT, 1, 2);
// (BACKWARD 1m) (LEFT 1m) 를 2초 안에 이동
  CoDrone2.hover(1);                  // 1초간 맴돌기 (호버링)

  CoDrone2.land();                    // 바닥으로 천천히 착륙
}
void loop()
{
}
```

6-3. goPosition

(이동방향1, 이동거리1, 이동방향2, 이동거리2, 이동방향3, 이동거리3, 이동시간)

설명 이 예제는 드론이 앞과 오른쪽 대각선방향으로 뒤와 왼쪽 대각선 방향으로 이동합니다.

```
#include "CoDrone2.h"

int direction1 = 0;
int distance1 = 0;

int direction2 = 0;
int distance2 = 0;

int time = 0;
```

```
void setup()
{
  int flightDuration = 20;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  // - 이동 방향과 거리, 시간 입력

  //--------------- 앞과 오른쪽 대각선 방향으로 이동 ---------------------------//

  direction1 = FORWARD;  // 방향 : 전진
  distance1 = 1;         // 거리 : 1m

  direction2 = RIGHT;  // 방향 : 오른쪽
  distance2 = 1;         // 거리 : 1m

  time = 2;             // 시간 : 2초

 CoDrone2.goPositon(direction1, distance1, direction2, distance2, time);
// (FORWARD 1m) (RIGHT 1m) 를 2초 안에 이동
  CoDrone2.hover(1);                                // 1초간 맴돌기 (호버링)

  //---------------- 뒤와 왼쪽 대각선 방향으로 이동 ---------------------------//

  CoDrone2.goPositon(FORWARD, 1, LEFT, 1, UP, 1, 3);
// 3초안에 1m 앞으로, 왼쪽으로 1m, 위로 1m 이동
  CoDrone2.hover(1);  // 1초간 맴돌기 (호버링)

  CoDrone2.land();                  // 바닥으로 천천히 착륙
}
void loop()
{
}
```

6-4. goPosition
(회전방향, 회전각도, 회전시간)

- **설명** 이 예제는 드론이 왼쪽과 오른쪽으로 회전합니다

```
#include "CoDrone2.h"

int direction1 = 0;
int angle = 0;
int time = 0;
```

```
void setup()
{
  int flightDuration = 20;          // 비행 지속 시간 설정

  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  // - 회전 방향과 각도, 시간 입력

  //----------------------------- 왼쪽으로 회전 ----------------------------//

  direction = LEFT_TURN;  // 방향 : 전진
  angle = 90;          // 거리 : 1m
  time = 2;             // 시간 : 2초

  CoDrone2.goPositon(direction, angle, time);
// (LEFT_TURN 90도) 를 2초 안에 이동
  CoDrone2.hover(1);                                // 1초간 맴돌기 (호버링)

  //----------------------------- 오른쪽으로 회전 ----------------------------//

  CoDrone2.goPositon(RIGHT_TURN, 90, 2);   // (RIGHT_TURN 90도) 를 2초 안에 회전
  CoDrone2.hover(1);                        // 1초간 맴돌기 (호버링)

  CoDrone2.hover(2);                        // 2초간 맴돌기 (호버링)

  CoDrone2.land();                         // 바닥으로 천천히 착륙
}

void loop()
{
}
```

6-5. goPosition

**(이동방향1, 이동거리1, 이동시간
회전방향, 회전각도, 회전시간)**

설명 이 예제는 드론이 왼쪽으로 회전하며 앞으로 이동하고, 오른쪽으로 회전하며 뒤로 이동합니다.

```
#include "CoDrone2.h"

int direction1 = 0; //방향 : 전진
int distance1 = 0;

int direction2 = 0;
int angle = 0;

int time1 = 0;
Int time2 = 0;
```

```
void setup()
{
  int flightDuration = 20;          // 비행 지속 시간 설정
  CoDrone2.begin(flightDuration);
// 비행을 위한 초기화와 준비 - 프로그램 시작 후 20초 뒤에 착륙

  CoDrone2.takeoff();               // 이륙 후 5초간의 대기 시간 필요

  CoDrone2.hover(5);                // 5초간 맴돌기 (호버링)

  // - 회전 방향과 각도, 시간 입력
  //---------------------------- 왼쪽으로 회전 ----------------------------//
  direction1 = FORWARD;  // 방향 : 전진
  distance = 1;          // 거리 : 1m
  time1 = 2;             // 시간 : 2초

  direction2 = LEFT_TURN;  // 방향 : 전진
  angle = 90;          // 거리 : 1m
  time2 = 2;             // 시간 : 2초

  CoDrone2.goPositon(direction1, distance, time1, direction2, angle, time2);
// (LEFT_TURN 90도) 를 2초 안에 이동

  CoDrone2.hover(1);                          // 1초간 맴돌기 (호버링)
  //---------------------------- 오른쪽으로 회전 ----------------------------//

  CoDrone2.goPositon(BACKWARD, 1, 2, RIGHT_TURN, 90, 2);
// (RIGHT_TURN 90도) 를 2초 안에 회전

  CoDrone2.hover(1);                    // 1초간 맴돌기 (호버링)

  CoDrone2.hover(2);                    // 2초간 맴돌기 (호버링)

  CoDrone2.land();                      // 바닥으로 천천히 착륙
}
void loop()
{
}
```

edrone DIY Guide

LED (드론의 LED 제어하기)

1. LED 색상지정
2. 이벤트 LED 색상 지정
3. 기본 LED 색상 지정

03

1. LED
(드론의 LED 제어하기)

01 드론의 LED를 제어합니다.

02 다양한 방식으로 LED를 제어하는 법을 배워 봅니다.

1. setLED
(LED 색상 지정)

- **목표** LED 색상 지정의 사용법
- **함수** setLED(색상, 조명 패턴, 패턴 간격)
  setLED(빨강색, 녹색, 파란색, 조명 패턴, 패턴 간격)
- **설명** 이 기능은 LED 색상, 조명 패턴 및 조명 패턴 간격을 설정합니다. 빨강, 녹색 및 파랑 입력 값 또는 미리 정의 된 색상을 사용하여 색상을 설정할 수 있습니다.

조명 패턴

- ***BodyHold*** : 지정한 색상을 계속 켭니다. 간격 값이 클수록 밝기가 밝아집니다.

- ***BodyFlicker*** : 지정한 색상으로 깜빡입니다. 간격 값이 작을 수록 빠르게 깜빡입니다.

- ***BodyFlickerDouble*** : 지정한 색상으로 두 번씩 깜빡입니다. 간격 값이 작을 수록 빠르게 깜빡입니다.

- ***BodyDimming*** : 점점 밝아 졌다가 점점 어두워졌다를 반복합니다. 간격 값이 작을 수록 빠르게 변합니다.

- ***BodySunrise*** : 꺼진 상태에서 점점 밝아짐니다. 간격 값이 작을 수록 빠르게 변합니다.

- ***BodySunset*** : 켜진 상태에서 점점 어두워짐니다. 간격 값이 작을 수록 빠르게 변합니다.

프로그램이 실행되면 드론의 앞쪽에 장착된 LED의 색상과 패턴이 달라집니다. 현재 입력된 LED 상태를 계속 유지합니다.

```ino
#include "CoDrone2.h"
int colorR = 0;
int colorG = 0;
int colorB = 0;
void setup()
{
  CoDrone2.begin();     // Initialization and preparation for flight
  // - 정의된 색상을 사용하는 경우

  // 흰색으로 켜기
  CoDrone2.setLED(White, BodyHold, 100);
  delay(3000);
  // 녹색으로 깜빡이기
  CoDrone2.setLED(Green, BodyFlicker, 100);
  delay(3000);
  // 노란색으로 두번씩 깜빡이기
  CoDrone2.setLED(Yellow, BodyFlickerDouble, 100);
  delay(3000);
  // 보라색으로 점점 밝아졌다 어두워졌다 반복하기
  CoDrone2.setLED(Purple, BodyDimming, 2);
  delay(3000);
  // 파란색으로 꺼진 상태에서 점점 밝아지기
  CoDrone2.setLED(Blue, BodySunrise, 10);
  delay(3000);
  // 빨간색으로 켜진 상태에서 점점 어두워지기
  CoDrone2.setLED(Red, BodySunset, 10);
  delay(3000);
  // LED 검정색으로 (끄기)
  CoDrone2.setLED(Black, BodyHold, 100);
  delay(5000);
```

```ino
  // - R,G,B 색상을 사용하는 경우
  // 흰색으로 켜기
  colorR = 255;
  colorG = 255;
  colorB = 255;
  CoDrone2.setLED(colorR, colorG, colorB, BodyHold, 100);
  delay(3000);
  // 녹색으로 깜빡이기
  colorR = 0;
  colorG = 255;
  colorB = 0;
  CoDrone2.setLED(colorR, colorG, colorB, BodyFlicker, 100);
  delay(3000);
  // 노란색으로 두번씩 깜빡이기
  colorR = 255;
  colorG = 255;
  colorB = 0;
  CoDrone2.setLED(colorR, colorG, colorB, BodyFlickerDouble, 100);
  delay(3000);
  // 보라색으로 점점 밝아졌다 어두워졌다 반복하기
  colorR = 255;
  colorG = 0;
  colorB = 255;
  CoDrone2.setLED(colorR, colorG, colorB, BodyDimming, 2);
  delay(3000);
  // 파란색으로 꺼진 상태에서 점점 밝아지기
  colorR = 0;
  colorG = 0;
  colorB = 255;
  CoDrone2.setLED(colorR, colorG, colorB, BodySunrise, 10);
  delay(3000);
  // 빨간색으로 켜진 상태에서 점점 어두워지기
  colorR = 255;
  colorG = 0;
  colorB = 0;
  CoDrone2.setLED(colorR, colorG, colorB, BodySunset, 10);
  delay(3000);
  // LED 검정색으로 (끄기)
  colorR = 0;
  colorG = 0;
  colorB = 0;
  CoDrone2.setLED(colorR, colorG, colorB, BodyHold, 100);}
void loop(){}
```

2. setEventLED
(이벤트 LED 색상 지정)

- **목표** 이벤트 LED 색상 지정의 사용법

- **함수** setEventLED (색상, 조명 패턴, 패턴간격)
  setEventLED (R,G,B,조명패턴,패턴간격)

- **설명** 이 기능은 LED가 입력한 횟수만큼 작동하고 원래의 LED 상태로 돌아오게 됩니다.
  LED 색상, 조명 패턴 및 조명 패턴 간격을 설정합니다.
  빨강, 녹색 및 파랑 입력 값 또는 미리 정의 된 색상을 사용하여 색상을 설정할 수 있습니다

```
#include "CoDrone2.h"

int colorR = 0;
int colorG = 0;
int colorB = 0;
```

```
void setup()  {
  CoDrone2.begin();      // Initialization and preparation for flight
//------------------------------------------------------------ - 정의된 색상을 사용하는 경우
  // 흰색으로 켜기
  CoDrone2.setEventLED(White, BodyHold, 10, 100);
  delay(3000);
  // 녹색으로 깜빡이기
  CoDrone2.setEventLED(Green, BodyFlicker, 20, 10);
  delay(3000);
  // 노란색으로 두번씩 깜빡이기
  CoDrone2.setEventLED(Yellow, BodyFlickerDouble, 10, 10);
 delay(3000);
  // 보라색으로 점점 밝아졌다 어두워졌다 반복하기
  CoDrone2.setEventLED(Purple, BodyDimming, 1, 10);
  delay(5000);   //---------------------------------------- - R,G,B 색상을 사용하는 경우
  // 흰색으로 켜기
  colorR = 255;
  colorG = 255;
  colorB = 255;
  CoDrone2.setEventLED(colorR, colorG, colorB, BodyHold, 10, 100);
  delay(3000);
  // 녹색으로 깜빡이기
  colorR = 0;
  colorG = 255;
  colorB = 0;
  CoDrone2.setEventLED(colorR, colorG, colorB, BodyFlicker, 20, 10);
  delay(3000);
  // 노란색으로 두번씩 깜빡이기
  colorR = 255;
  colorG = 255;
  colorB = 0;
  CoDrone2.setEventLED(colorR, colorG, colorB, BodyFlickerDouble, 10, 10);
  delay(3000);
  // 보라색으로 점점 밝아졌다 어두워졌다 반복하기
  colorR = 255;
  colorG = 0;
  colorB = 255;
  CoDrone2.setEventLED(colorR, colorG, colorB, BodyDimming, 1, 10);
}
void loop()
{
}
```

3. setDefaultLED
(기본 LED 색상 지정)

- **목표** 기본 LED 색상 지정의 사용법

- **함수** setEventLED (색상, 조명 패턴, 패턴간격)
  setEventLED (R,G,B,조명패턴,패턴간격)

- **설명** 이 기능은 기본 LED 색뿐만 아니라 모드를 설정하므로 전원을 껐다가 다시 켠 후에도 해당 색과 조명 패턴을 유지합니다.
  빨강, 녹색 및 파랑 입력 값 또는 미리 정의 된 색상을 사용하여 색상을 설정할 수 있습니다.
  프로그램이 실행되면 입력한 LED 상태가 기본 LED 상태가 됩니다. 전원을 껏다가 켜도 유지됩니다

```
#include "CoDrone2.h"

int colorR = 0;
int colorG = 0;
int colorB = 0;

void setup()
{
  CoDrone2.begin();      // Initialization and preparation for flight

 // 전원을 켜면 빨간색이 켜진 상태가 됩니다.
  CoDrone2.setDefaultLED(Red, BodyHold, 100);

  /*
    colorR = 255;
    colorG = 0;
    colorB = 0;
    CoDrone2.setDefaultLED(colorR, colorG, colorB, BodyHold, 100);
  */
}
void loop()
{
}
```

edrone DIY Guide

Request (드론의 상태값 읽기)

03

1. State (상태)
2. Motion (모션센서)
3. Altitue (고도)
4. Position (위치)
5. Record And Setting (비행기록과 셋팅)

1. Request
(데이터 요청)

01 드론의 다양한 데이터 값을 확인합니다.

02 예제는 기본적으로 시리얼 모니터 창으로 확인하도록 되어있습니다.

1. getSystemMode
(드론의 시스템 동작 상태 값을 가져오기)

- **목표** 상태값 가져오기의 사용 방법
- **함수** getSystemMode()
- **설명** 현재 드론의 시스템 동작 상태 값을 가져옵니다. Boot, Start, Running, ReadyToReset, Error 의 상태를 가집니다.
프로그램입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.
드론의 시스템 동작 상태 값을 시리얼 모니터창에 출력하는 예제입니다

```
#include "CoDrone2.h"
void setup()  {
  CoDrone2.begin();              // 비행을 위한 초기화와 준비
}
void loop()  {
  // 드론의 동작 상태 값을 출력합니다.
  Serial.println("- State");
  Serial.print("getSystemMode₩t");
  byte mode = CoDrone2.getSystemMode();
// 드론의 시스템 동작 상태 값을 가져옵니다.
  Serial.println(mode, HEX);
// 드론의 시스템 동작 상태 값을 16진수로 출력합니다.
  displaySystemMode(mode);
// 드론의 시스템 동작 상태 값의 정의를 출력합니다.
  Serial.println();
  delay(1000);
}
// 드론의 시스템 동작 상태 값의 정의를 출력하는 함수입니다.
void displaySystemMode(byte mode)
{
  switch (mode)
  {
    case 0:
      Serial.println("NONE");    break;
    case 0x01:
      Serial.println("Boot");  break;
    case 0x02:
      Serial.println("Start");  break;
    case 0x03:
      Serial.println("Running"); break;
    case 0x04:
      Serial.println("ReadyToReset"); break;
    case 0x05:
      Serial.println("Error"); break;
  }
}
```

2. getFlightMode
(비행 제어기 동작 상태 값 가져오기)

- **목표** 비행 제어기 동작 상태 값 가져오기의 사용방법

- **함수** getFlightMode()

- **설명** 현재 드론의 비행 제어기 동작 상태 값을 가져옵니다.
  Ready, Start, Takeoff, Flight, Landing, Flip, Reverse, Stop, Accident, Error의 상태 값을 가집니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  비행 제어기 동작 상태 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"
void setup()  {
  CoDrone2.begin();              // 비행을 위한 초기화와 준비
}
void loop()  {
  // 비행 제어기 동작 상태 값을 출력합니다.
  Serial.println("- State");
  Serial.print("getFlightMode₩t");
  byte mode = CoDrone2.getFlightMode();
// 비행 제어기 동작 상태 값을 가져옵니다.
  Serial.println(mode, HEX);
// 비행 제어기 동작 상태 값을 16진수로 출력합니다.
  displayFlightMode(mode);
// 비행 제어기 동작 상태 값의 정의를 출력합니다.
  Serial.println();
  delay(1000);
}
// 비행 제어기 동작 상태 값의 정의를 출력하는 함수입니다.
void displayFlightMode(byte mode)  {
  switch (mode)  {
    case 0:
      Serial.println("NONE");    break;
    case 0x10:
      Serial.println("Ready");   break;
    case 0x11:
      Serial.println("Start");   break;
    case 0x12:
      Serial.println("TakeOff"); break;
    case 0x13:
      Serial.println("Flight");  break;
    case 0x14:
      Serial.println("Landing"); break;
    case 0x15:
      Serial.println("Flip");    break;
    case 0x16:
      Serial.println("Reverse"); break;
    case 0x20:
      Serial.println("Stop");    break;
    case 0x30:
      Serial.println("Accident"); break;
    case 0x31:
      Serial.println("Error");     break;
  } }
```

3. getFlightMode
(비행 제어 모드 값 가져오기)

- **목표** 비행 제어 모드 값 가져오기의 사용 방법

- **함수** getFlightControMode()

- **설명** 현재 드론의 비행 제어 모드 값을 가져옵니다.
다음과 같은 제어모드를 가지고 있습니다.
Attitude
(자세 - X,Y는 각도(deg)로 입력받음, Z,Yaw는 속도(m/s)로 입력 받음)
Position (위치 - X,Y,Z,Yaw는 속도(m/s)로 입력 받음)
Function (기능 - X,Y,Z,Yaw는 속도(m/s)로 입력 받음)

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  비행 제어 모드 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"
void setup()  {
  CoDrone2.begin();              // 비행을 위한 초기화와 준비
}
void loop()
{
  // 비행 제어 모드 값을 출력합니다.
  Serial.println("- State");
  Serial.print("getFlightControlMode₩t");

  byte mode = CoDrone2.getFlightControlMode();
// 비행 제어 모드 값을 가져옵니다.

  Serial.println(mode, HEX);
// 비행 제어 모드 값을 16진수로 출력합니다.

  displayFlightControlMode(mode);
// 비행 제어 모드 상태 값의 정의를 출력합니다.

  Serial.println();
  delay(1000);
}

// 비행 제어 모드 상태 값의 정의를 출력하는 함수입니다.
void displayFlightControlMode(byte mode)
{
  switch (mode)
  {
    case 0:
      Serial.println("NONE");    break;
    case 0x10:
      Serial.println("Attitude");  break;
    case 0x11:
      Serial.println("Position");  break;
    case 0x12:
      Serial.println("Function"); break;
  }
}
```

4. getMovementStatus (이동 상태 값을 가져오기)

- **목표** 비행 제어 모드 값 가져오기의 사용 방법
- **함수** getFlightControMode()
- **설명** 드론의 이동 상태값을 가져옵니다.
  Ready, Hovering, Moving, ReturnHome의 상태 값을 가집니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  이동 상태 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"
void setup()  {
  CoDrone2.begin();              // 비행을 위한 초기화와 준비
}
void loop()  {
  // 드론의 이동 상태 값 을 출력합니다.
  Serial.println("- State");
  Serial.print("getMovementStatus₩t");

 byte mode = CoDrone2.getFlightControlMode();
// 비행 제어 모드 값을 가져옵니다.

  Serial.println(mode, HEX);
// 비행 제어 모드 값을 16진수로 출력합니다.

  displayFlightControlMode(mode);
// 비행 제어 모드 상태 값의 정의를 출력합니다.

  Serial.println();
  delay(1000);
}}
// 드론의 이동 상태 값의 정의를 출력하는 함수입니다.
void displayMovementStatus(byte mode)
{
  switch (mode)
  {
    case 0:
      Serial.println("NONE");    break;
    case 0x01:
      Serial.println("Ready");   break;
    case 0x02:
      Serial.println("Hovering");   break;
    case 0x03:
      Serial.println("Moving"); break;
    case 0x04:
      Serial.println("ReturnHome"); break;
  }
}
```

5. getHeadlessMode
(방위 기준 값을 가져오기)

- **목표** 방위 기준 값 가져오기의 사용 방법

- **함수** getHeadlessMode()

- **설명** 조종시 방향의 기준 값을 가져옵니다.
Headless는 사용자 중심 좌표로써 드론이 어느 방향을 바라보더라도 이륙 할때의 방향 또는 사용자가 지정한 방향을 기준으로 움직입니다.
Normal은 드론 중심 좌표로써 드론이 현재 향하는 방향을 기준으로 움직입니다. 기본 설정은 Normal 입니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  방위 기준 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"
void setup()
{
  CoDrone2.begin();             // 비행을 위한 초기화와 준비
}

void loop()
{
  // 방위 기준 값을 출력합니다.
  Serial.println("- State");
  Serial.print("getHeadlessMode₩t");

  byte mode = CoDrone2.getHeadlessMode();
// 드론의 방위 기준 값을 가져옵니다.

  Serial.println(mode, HEX);
// 드론의 방위 기준 값을 16진수로 출력합니다.

  displayHeadlessMode(mode);
// 드론의 방위 기준 값의 정의를 출력합니다.

  Serial.println();
  delay(1000);
}

// 드론의 방위 기준 값의 정의를 출력하는 함수입니다.
void displayHeadlessMode(byte mode)
{
  switch (mode)
  {
    case 0:
      Serial.println("NONE");    break;
    case 0x01:
      Serial.println("Headless");  break;
    case 0x02:
      Serial.println("Normal");  break;
  }
}
```

6. getSensorOrientationMode (센서 방향 값 가져오기)

- **목표** 센서 방향 값 가져오기의 사용 방법
- **함수** getSensorOrientationMode()
- **설명** 드론의 센서 방향 값을 가져옵니다.
  Normal, ReverseStart, Reversed 의 상태 값을 가집니다.
  드론이 뒤집혀 있는지 알수 있습니다.
  값이 1 인 경우 정상, 2인 경우 뒤집히기 시작, 3인 경우에 뒤집힌 상태를 뜻합니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  센서 방향 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"
void setup()
{
  CoDrone2.begin();             // 비행을 위한 초기화와 준비
}

void loop()
{
  // 드론의 센서 방향 값을 출력합니다.
  Serial.println("- State");
  Serial.print("getSensorOrientationMode₩t");

  byte mode = CoDrone2.getSensorOrientationMode();
// 드론의 센서 방향 값을 가져옵니다.

  Serial.println(mode, HEX);
// 드론의 센서 방향 값을 16진수로 출력합니다.

  displayHeadlessMode(mode);
// 드론의 센서 방향 값의 정의를 출력합니다.

  Serial.println();
  delay(1000);
}

// 드론의 센서 방향 값의 정의를 출력합니다.
void displaySensorOrientationMode(byte (byte mode)
{
  switch (mode)
  {
    case 0:
      Serial.println("NONE");    break;
    case 0x01:
      Serial.println("Normal ");  break;
    case 0x02:
      Serial.println("ReverseStart");  break;
    case 0x03:
      Serial.println(" Reversed ");  break;
  } }
```

7. getBatteryPercentage (베터리 잔량확인)

- **목표** 베터리 잔량 확인의 사용 방법
- **함수** getBatteryPercentage()
- **설명** 현재 드론의 배터리 잔량을 확인합니다.
  배터리양을 최저 0에서 최고 100의 값으로 나타냅니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  배터리 잔량 값을 시리얼 모니터창에 출력하는 예제입니다

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();             // 비행을 위한 초기화와 준비
}

void loop()
{
  // 드론의 배터리 잔량을 출력합니다.
  Serial.println("- State");
  Serial.print("getBatteryPercentage₩t");

  byte battery = CoDrone2.getBatteryPercentage();
// 드론의 배터리 잔량을 가져옵니다.

  Serial.println(battery);
// 드론의 배터리 잔량을 표시합니다.

  Serial.println();
  delay(1000);
}
```

8. getAngularSpeed
(자이로 센서 값 가져오기)

- **목표** 드론의 자이로 센서 값 가져오기의 사용 방법

- **함수** getAngularSpeed()

- **설명** 드론의 자이로 센서 값을 가져옵니다.
  롤, 피치, 요로 자이로 값을 표시합니다.
  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  자이로 센서의 롤, 피치, 요를 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();
}

void loop()
{
  gyrodata gyro;
  gyro = CoDrone2.getAngularSpeed();
// 수신된 드론의 자이로 센서 값을 저장

  // 드론의 자이로 센서 값을 출력합니다.
  Serial.println("- AngularSpeed");
  Serial.print("gyro roll : ₩t");
  Serial.println(gyro.roll);
  Serial.print("gyro pitch : ₩t");
  Serial.println(gyro.pitch);
  Serial.print("gyro yaw : ₩t");
  Serial.println(gyro.yaw);

  Serial.println();
  delay(500);
}
```

9. getAccelerometerd
(가속도 센서 값 가져오기)

- **목표** 드론의 가속도 센서 값 가져오기의 사용 방법

- **함수** getAccelerometer()

- **설명** 드론의 가속도 센서 값을 가져옵니다.
  x,y,z축의 값을 표시합니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  가속도 x, y, z 축의 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                          // 비행을 위한 초기화와 준비
}

void loop()
{
  acceldata accel;
  accel = CoDrone2.getAccelerometer();     // 수신된 가속도 센서 값을 저장

  // 드론의 가속도 센서 값을 출력합니다.
  Serial.println("- Accelerometer");
  Serial.print("accel x : ₩t");
  Serial.println(accel.x);
  Serial.print("accel y : ₩t");
  Serial.println(accel.y);
  Serial.print("accel z : ₩t");
  Serial.println(accel.z);

  Serial.println();
  delay(500);
}
```

10. getGyroAngles
(각도값 가져오기)

- **목표** 각도 값 가져오기의 사용 방법
- **함수** getGyroAngles()
- **설명** 드론의 각도 값을 가져옵니다.
  가속도와 자이로값으로 만들어진 각도 값입니다.
  롤, 피치, 요의 각도를 표시합니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 롤, 피치, 요 각도를 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                          // 비행을 위한 초기화와 준비
}

void loop()
{
  angledata angle;
  angle = CoDrone2.getGyroAngles();        // 수신된 드론의 각도 값을 저장

  // 드론의 각도 값을 출력합니다.
  Serial.println("- GyroAngles");
  Serial.print("angle roll : ₩t");
  Serial.println(angle.roll);
  Serial.print("angle pitch : ₩t");
  Serial.println(angle.pitch);
  Serial.print("angle yaw : ₩t");
  Serial.println(angle.yaw);

  Serial.println();
  delay(500);
}
```

11. getRawMotion
(가곡되지 않은 모션 센서값 가져오기)

- **목표** 가공되지 않은 모션 센서값 가져오기에 대한 사용법
- **함수** getRawMotion()
- **설명** 가공되지 않은 모션 센서(가속도, 자이로) 자체의 값을 가져옵니다.
  드론에서는 센서에서 이렇게 입력된 값을 다양한 방법으로 계산하여 실제 자세를 제어합니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 가속도와 자이로 센서 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();   // 비행을 위한 초기화와 준비
}

void loop()
{
  rawmotion motion;
  motion = CoDrone2.getRawMotion();
// 가공되지 않은 모션 센서값을 저장

  // 모션 센서값 출력
  Serial.println("- RAW Motion");
  Serial.print("accel x : ₩t");
  Serial.println(motion.x);
  Serial.print("accel y : ₩t");
  Serial.println(motion.y);
  Serial.print("accel z : ₩t");
  Serial.println(motion.z);
  Serial.print("angle roll : ₩t");
  Serial.println(motion.roll);
  Serial.print("angle pitch : ₩t");
  Serial.println(motion.pitch);
  Serial.print("angle yaw : ₩t");
  Serial.println(motion.yaw);

  Serial.println();
  delay(500);
}
```

12. getDroneTemp
(드론 온도 값 가져오기)

- **목표** 드론 온도 값 가져오기의 사용 방법
- **함수** getDroneTemp()
- **설명** 드론의 온도 값을 가져옵니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 온도 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  // 드론의 온도를 출력합니다.
  Serial.println("- Temperature");
  Serial.print("getDroneTemp₩t");

  Serial.println(CoDrone2.getDroneTemp(), 5);
// 드론의 온도를 소숫점 5자리까지 표시합니다.

  Serial.println();
  delay(500);
}
```

13. getPressure
(드론 기압 값 가져오기)

- **목표** 드론 기압 값 가져오기의 사용 방법
- **함수** getDroneTemp()
- **설명** 드론의 기압 값을 가져옵니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 기압 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  // 드론의 기압을 출력합니다.
  Serial.println("- Pressure");
  Serial.print("getPressure₩t");

  Serial.println(CoDrone2.getPressure(), 2);
// 드론의 기압을 소숫점 2자리까지 표시합니다.

  Serial.println();
  delay(500);
}
```

14. getAltitude
(드론 고도 값 가져오기)

- **목표** 드론 고도 값 가져오기의 사용 방법
- **함수** getAltitude()
- **설명** 드론의 고도 값을 가져옵니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 고도 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  // 드론의 고도를 출력합니다.
  Serial.println("- Altitude");
  Serial.print("getAltitude₩t");

  Serial.println(CoDrone2.getAltitude(), 5);
// 드론의 고도를 소숫점 5자리까지 표시합니다.

  Serial.println();
  delay(500);
}
```

15. getHeight
(드론의 높이 센서 값 가져오기)

- **목표** 드론의 높이 값 가져오기의 사용 방법
- **함수** getHeight()
- **설명** 드론의 높이 값을 가져옵니다.
  바닥을 기준으로 드론 밑면과의 높이를 가져옵니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 높이 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  // 드론의 거리 센서를 출력합니다.
  Serial.println("- Height");
  Serial.print("getHeight₩t");

  Serial.println(CoDrone2.getHeight(), 3);
// 드론의 고도를 소숫점 3자리까지 표시합니다.

  Serial.println();
  delay(500);
}
```

16. getRange
(드론의 거리 센서 값 가져오기)

- **목표** 드론의 거리 센서 값 가져오기의 사용 방법
- **함수** getHeight()
- **설명** 드론의 거리 센서 값을 가져옵니다.

  bottom : 드론에 기본적으로 장착되어 있는 센서 (높이를 측정)
  front : 코드론2에 탑재되는 정면 감지용 센서
  left : 좌측 감지용 센서 (추가적으로 센서를 구매하여 장착)
  right : 우측 감지용 센서 (추가적으로 센서를 구매하여 장착)

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 거리 센서 값을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  rangeData range;
  range = CoDrone2.getRange();       // 수신된 드론의 각도 값을 저장

  Serial.println("- Range Sensor");

  Serial.print("front : ₩t");
  Serial.println(range.front);       // 정면 센서 감지 값을 표시합니다.

  Serial.print("left  : ₩t");
  Serial.println(range.left);        // 왼쪽 센서 감지 값을 표시합니다.

  Serial.print("right : ₩t");
  Serial.println(range.right);       // 오른쪽 센서 감지 값을 표시합니다.

  Serial.print("bottom : ₩t");
  Serial.println(range.bottom);      // 바닥 센서 감지 값을 표시합니다.

  delay(100);
}
```

17. getOptFlowPosition (옵티컬 플로우 값 가져오기)

- **목표** 옵티컬 플로우 값 가져오기의 사용 방법
- **함수** getOptFlowPosition()
- **설명** 드론의 옵티컬 플로우 값을 가져옵니다.
  x, y 값으로 표시됩니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  옵티컬 플로우 x,y 값을 시리얼 모니터창에 출력하는 예제입니다

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  optdata opt;
  opt = CoDrone2.getOptFlowPosition();
// 수신된 옵티컬 플로우 값을 저장

  // 옵티컬 플로우 값을 출력
  Serial.println("- Optical Flow");

  Serial.print("opt x : ₩t");                // 옵티컬 플로우 x 값
  Serial.println(opt.x);
  Serial.print("opt y : ₩t");                // 옵티컬 플로우 y 값
  Serial.println(opt.y);

  Serial.println();
  delay(500);
}
```

18. getPosition
(드론의 위치 가져오기)

- **목표** 옵티컬 플로우 값 가져오기의 사용 방법
- **함수** getOptFlowPosition()
- **설명** 드론의 옵티컬 플로우 값을 가져옵니다.
  x, y 값으로 표시됩니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  옵티컬 플로우 x,y 값을 시리얼 모니터창에 출력하는 예제입니다

```
#include "CoDrone2.h"

void setup()
{
  CoDrone2.begin();                    // 비행을 위한 초기화와 준비
}

void loop()
{
  optdata opt;
  opt = CoDrone2.getOptFlowPosition();
 // 수신된 옵티컬 플로우 값을 저장

  // 옵티컬 플로우 값을 출력
  Serial.println("- Optical Flow");

  Serial.print("opt x : ₩t");                // 옵티컬 플로우 x 값
  Serial.println(opt.x);
  Serial.print("opt y : ₩t");                // 옵티컬 플로우 y 값
  Serial.println(opt.y);

  Serial.println();
  delay(500);
}
```

19. getFlightRecord (드론의 비행 기록 가져오기)

- **목표** 드론의 비행 기록 가져오기의 사용 방법
- **함수** getFlightRecord()
- **설명** 드론의 비행 기록을 가져옵니다.
  드론의 비행 시간, 이륙 횟수, 착륙 횟수, 사고 횟수를 표시합니다.

  프로그램 입력이 완료되면 시리얼 모니터창을 열고 통신속도를 "57600bps"로 맞춥니다.

  드론의 비행 기록을 시리얼 모니터창에 출력하는 예제입니다.

```
#include "CoDrone2.h"
void setup()
{
  CoDrone2.begin();                          // 비행을 위한 초기화와 준비

  flightrecord record;
  record  = CoDrone2.getFlightRecord();   // 수신된 비행 기록 값을 저장

  // 비행 기록 값을 출력
  Serial.println("- FlightRecord");

  // 비행 시간  출력
  Serial.print("FlightTime₩t");
  Serial.print(record.hour);               // 드론의 비행시간 : 시간
  Serial.print(" hour ");
  Serial.print(record.minute);             // 드론의 비행시간 : 분
  Serial.print(" minute ");
  Serial.print(record.second, 3);          // 드론의 비행시간 : 초
  Serial.println(" second");

  // 이륙 횟수 출력
  Serial.print("countTakeOff₩t");
  Serial.println(record.takeoff);          // 드론의 이륙 횟수

  // 착륙 횟수 출력
  Serial.print("countLanding₩t");          // 드론의 착륙 횟수
  Serial.println(record.landing);

  // 사고 횟수 출력
  Serial.print("countAccident₩t");         // 드론의 사고 횟수
  Serial.println(record.accident);
}
void loop()
{
}
```