SEOULTECH HRR LAB
Humanoid Robot Research Lab

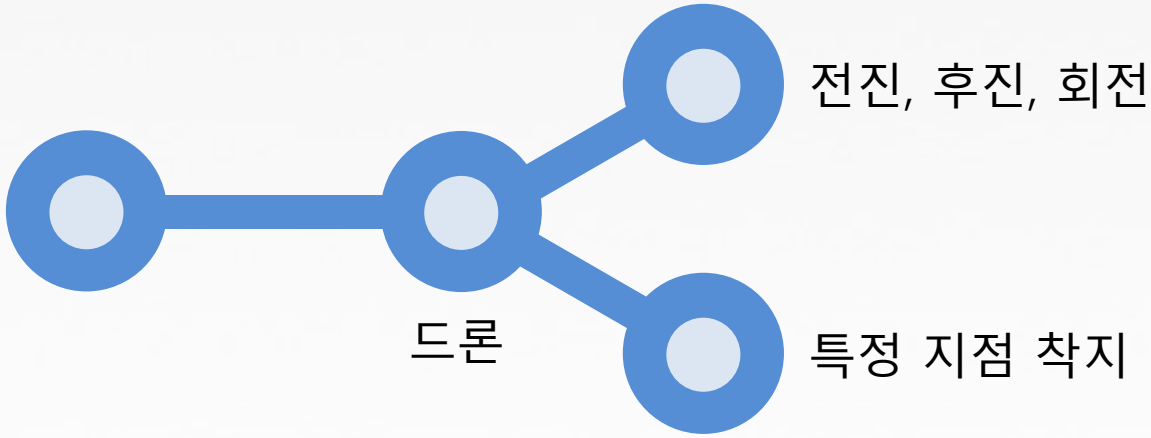
ESPERS
겨울방학 Git/Github 교육

강의자 : 김성욱

Github를 통한 협업

한륭이와 민우는 같이 드론을 만들기로 하였다. 한륭이는 드론의 전진, 후진, 요잉 등의 제어를 담당하기로 하였고, 민우는 드론에 있는 카메라로 특정 장소에 착지 가능하도록 영상 처리를 담당하기로 하였다.

	한륭	민우
13시	전진, 후진, 회전	특정 지점 착지
14시		전진, 후진, 회전 + 특정 지점 착지
15시	전진, 후진, 미세각도회전	전진, 후진, 미세각도회전 + 특정 지점 착지



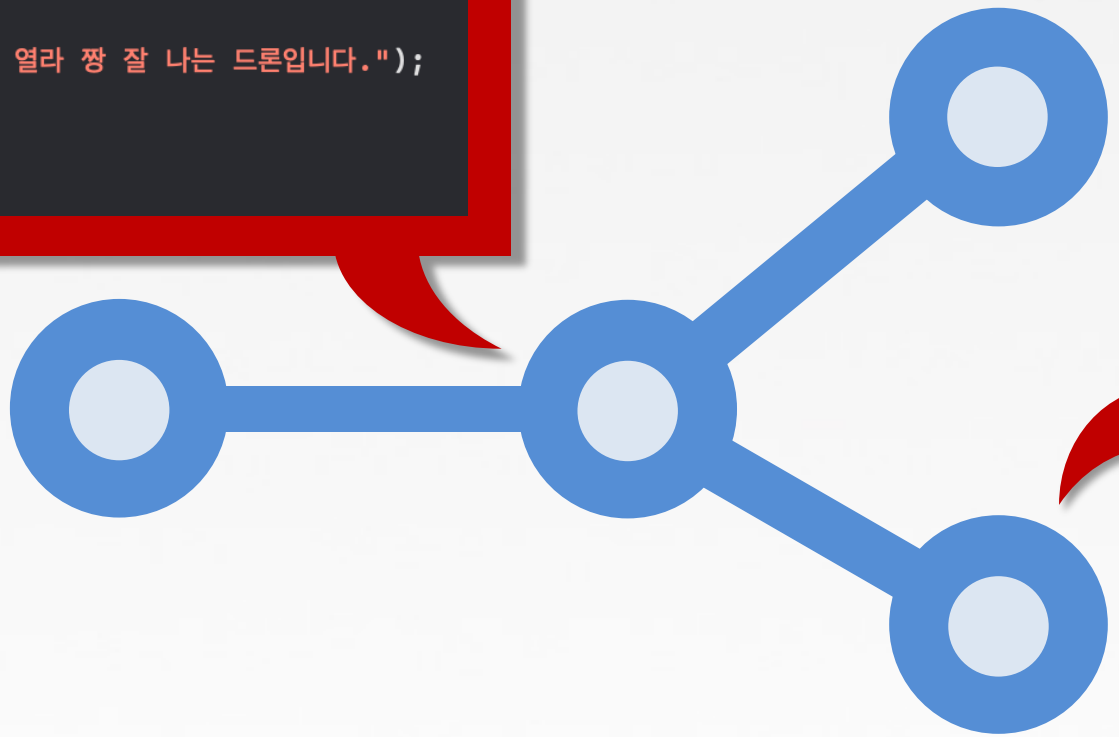
Github를 통한 협업

SEOULTECH HRR LAB Humanoid Robot Research Lab

```c
#include <stdio.h>

int main(void)
{
    printf("나는 열라 짱 잘 나는 드론입니다.");
    return 0;
}

```



```c
#include <stdio.h>

int main(void)
{
    printf("나는 열라 짱 잘 나는 드론입니다.");

    printf("영상 처리 착지");
    return 0;
}
```

```c
#include <stdio.h>

int main(void)
{
    printf("나는 열라 짱 잘 나는 드론입니다.");

    printf("전진");
    printf("후진");
    printf("회전");

    return 0;
}
```

Branch 관련 용어

[master],[master/origin] : 마스터 브랜치. Git이 제공하는 기본 브랜치를 의미.
[HEAD] : 현재 내가 보고있는 브랜치 (현재 파일의 상태는 HEAD가 가리키는 상태)
체크아웃(checkout) : 내가 보는 브랜치를 바꾸는 것.

Branch 관련 용어

병합(merge) : 두 브랜치를 합치는 것.
충돌(conflict) : 두 브랜치 사이의 내용이 달라, 서로 합칠 수 없는 것.
페치 (fetch) : 현재 원격 저장소(Github) 상의 상황만 가져오는 것.

```
#include <stdio.h>

int main(void)
{
    printf("나는 열라 짱 잘 나는 드론입니다.");

<<<<<<< HEAD
    printf("영상 처리 착지");
=======
    printf("전진");
    printf("후진");
    printf("미세 각도 회전");

>>>>>>> Ryung
    return 0;
}
```

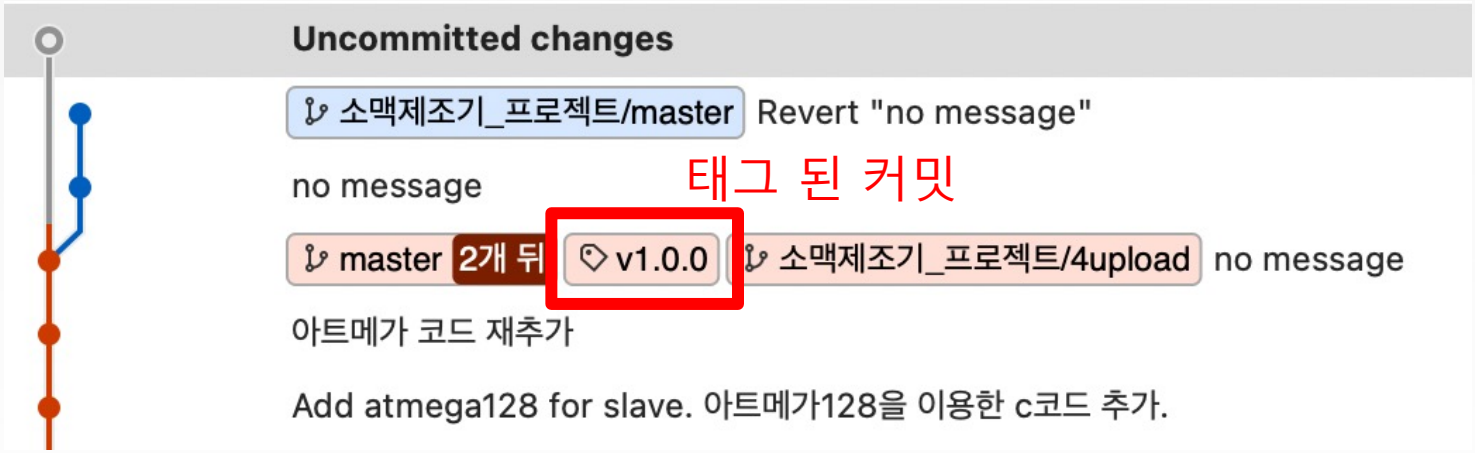
<<<<<<<<< HEAD
=========
>>>>>>>> 합쳐서 없앨 브랜치

위 부분을 지우고, 코드가 잘 작동하도록 수정!

현재 HEAD인 브랜치로 모든 다른 브랜치가 흡수되니
이 점 유의하자!!

Github를 통한 협업

풀 리퀘스트 (Pull Request) : 내가 올리는 코드를 레포지토리 관리자에게 적용해 달라고 요청하는 것.
릴리즈 (Release) : 현재 우리의 커밋을 특정 버전으로 취급하여, 배포하는 것.
태그 (tag) : 릴리즈 할 버전을 설정하는 것.



Github를 통한 협업

SEOULTECH HRR LAB Humanoid Robot Research Lab

<국룰 전략>

1. [master] 브랜치에는 직접 커밋을 올리지 않는다. (동시에 작업하다 꼬일 수 있으니)
2. 기능 개발을 하기 전에 [master] 브랜치를 기준으로 새로운 브랜치를 만든다.
3. 이 브랜치 이름은 [기능이름] 으로 정하고, 한 명만 커밋을 올린다.
4. [기능이름] 브랜치에서 기능 개발이 끝나면 [master] 브랜치에 이를 합친다.
5. 조장이 레포지토리를 맡고, 풀 리퀘스트가 올라 올 때마다 검수하여 확인한다.

Github를 통한 협업

포크 (Fork) : 다른 사람의 원격 저장소를 내 저장소로 저장하는 것.
주로, 회사나 큰 프로젝트 참여 등 대규모 프로젝트에 사용되며,
원본 제작에게 허락을 맡으면 내가 그 프로젝트에 기여할 수도 있다.