

# 임무 소프트웨어 Guide

대기환경 측정

msw\_sparrow\_air

# 목 차

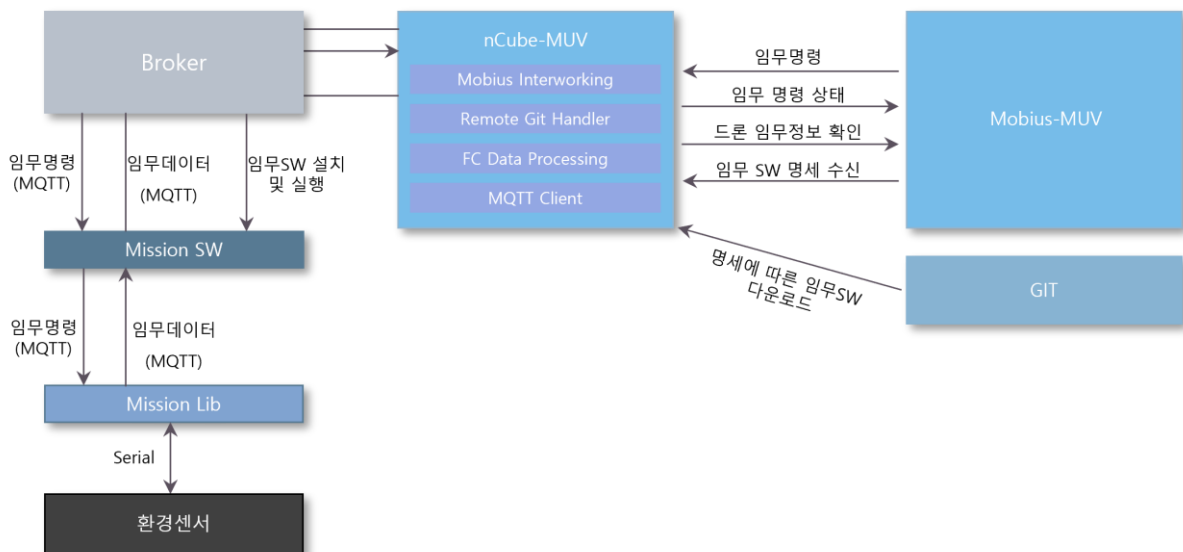
1	개요 .....	3
2	환경센서 .....	4
2.1	환경센서란? .....	4
2.2	환경센서 탑재 .....	4
3	임무Lib 및 임무SW 개발 .....	5
3.1	임무Lib, 임무SW와 개발도구 .....	5
3.2	임무Lib 개발 .....	5
3.3	임무SW 개발 .....	14

## 1 개요

이 문서는 중앙에서 임무 컴퓨터(Mission Computer, MC)를 통해 드론에 탑재된 환경센서를 이용하여 미세먼지와 같은 대기환경 측정과 에어펌프 제어를 위한 임무 SW를 개발하기 위한 방법에 대해 설명하는 문서이다. 여기서 설명하는 MC에 탑재된 임무 SW는 환경센서에서 생성된 환경센서 데이터를 Serial 인터페이스를 통해 수신하고 전달받은 데이터를 중앙으로 전송한다.

사용자는 MC를 드론에 탑재하여 아래의 아키텍처와 같이 nCube-MUV와 임무 SW를 동작하여 환경센서 데이터를 통해 대기환경 상태를 알 수 있다. Serial 통신을 통해 MC로 전달된 환경센서 데이터는 임무SW에서 센서별로 파싱하여 중앙으로 업로드 한다.

이를 위해 대기환경 측정 임무에 대해 설명하고 환경센서와 직접적으로 연결되어 데이터를 주고받는 임무라이브러리(Library, Lib), 임무 Lib를 포함한 형태로 nCube-MUV와 연동되어 외부와 통신하며 임무 Lib에서 데이터를 전달받는 임무 소프트웨어(Software, SW)에 대해 설명하고 동작하기 위한 방법을 설명한다.



< 대기환경 측정 임무 소프트웨어(msw\_sparrow\_air) 아키텍처 >

## 2 환경센서

### 2.1 환경센서란?

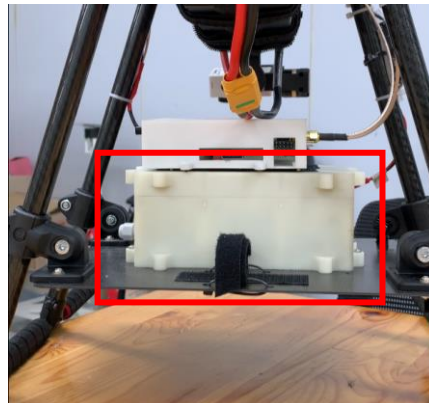
환경센서는 드론에 탑재되어 비행 중 대기환경을 측정하기 위한 임무장비로써 미세먼지, 일산화탄소, 이산화질소 등을 측정할 수 있는 센서를 포함하고 있고 에어펌프를 제어하여 보다 원활한 센서 측정이 가능하다. 환경센서는 Serial 인터페이스를 통해 MC와 통신하여 측정한 센서 데이터를 전달하고 에어펌프를 동작하기 위한 명령을 수신할 수 있다. 또한, I2C 인터페이스도 지원하여 Serial 인터페이스와 동일하게 사용 가능하다.



<환경센서>

### 2.2 환경센서 탑재

- 환경센서를 드론의 하단에 튼튼하게 고정시켜 장착한다.
- 환경센서의 Serial 포트를 MC의 Serial 포트(/dev/ttyUSB3)에 연결한다.



<드론에 환경센서가 장착된 모습>

### 3 임무Lib 및 임무SW 개발

#### 3.1 임무Lib, 임무SW와 개발도구

임무Lib는 임무 장비와 직접적으로 연결되어 임무 장비를 제어하는 소프트웨어이다. 필수 구현 요소로 임무 장비와 연결하기 위한 인터페이스(Serial, I2C, PWM 등) 연동, 내부적으로 임무SW와 통신하기 위한 MQTT 통신에 대한 내용이 있다. 또한 임무 장비, 사용자 또는 환경에 따라 데이터 모델 정의, 통신 모델 정의 등의 내용을 추가할 수 있다. 임무Lib를 임무SW에서 사용을 위해 개발자가 사전에 개발하여 응용프로그램 형태로 배포해야 한다.

임무SW는 임무Lib를 포함한 형태로 개발되며 외부와 통신하며 임무Lib를 제어하는 소프트웨어이다. 임무SW는 응용프로그램 형태의 임무Lib를 실행하기 위한 스크립트를 정의하는 내용이 필수이며, 또한 외부와 통신하기 위해 통신 모델을 정의하고 데이터 모델을 정의하는 내용을 필수로 한다.

개발자는 개발도구를 통해 임무SW를 좀 더 쉽게 개발하고 배포하여 드론에 탑재까지 가능하다. 개발도구는 블록코딩 형식으로 개발할 수 있으며 Github와 연동하여 이미 배포된 임무Lib와 임무SW를 재사용이 가능하고 개발된 임무SW를 배포할 수 있으며 드론과 임무에 대한 명세를 작성함으로써 자동으로 드론에 탑재와 동작이 가능하다.

#### 3.2 임무Lib 개발

환경센서 통신하기 위해 임무장비와 직접적으로 연결되어 제어가 가능한 임무Lib를 사전에 개발한다. 임무Lib는 MC에 탑재되어 환경센서와 직접적인 연결과 환경센서를 실질적으로 제어하는 역할을 수행한다. 임무Lib에는 아래의 목록이 필수적으로 구현되어야 한다. 개발된 임무Lib는 최종적으로 응용프로그램 형태로 뒤에 설명하는 임무SW에 포함된 형태로 동작한다.

- 임무Lib는 크게 4가지 부분으로 나뉜다.
  - ① MC와 내부적으로 데이터 통신을 위한 MQTT 통신 예시코드

```
def msw_mqtt_connect():
    global lib_topic
    global lib_mqtt_client
    global broker_ip
    global port

    lib_topic = ''

    lib_mqtt_client = mqtt.Client()
    lib_mqtt_client.on_connect = on_connect
    lib_mqtt_client.on_disconnect = on_disconnect
    lib_mqtt_client.on_subscribe = on_subscribe
    lib_mqtt_client.on_message = on_message
    lib_mqtt_client.connect(broker_ip, port)
    lib_mqtt_client.loop_start()
    return lib_mqtt_client

def on_connect(client, userdata, flags, rc):
    global control_topic
    global broker_ip
    print('[msw_mqtt_connect] connect to ', broker_ip)
    lib_mqtt_client.subscribe(control_topic, 0)
    print('[lib]control_topic\n', control_topic)

def on_disconnect(client, userdata, flags, rc=0):
    print(str(rc))

def on_subscribe(client, userdata, mid, granted_qos):
    print("subscribed: " + str(mid) + " " + str(granted_qos))

def on_message(client, userdata, msg):
    global missionPort
    global data_topic
    global control_topic

    message = str(msg.payload.decode("utf-8"))
    if message == 'G':
        on_receive_from_msw(message)

def on_receive_from_msw(str_message):
    global missionPort
```

```
global flag

if missionPort is not None:
    if missionPort.is_open:
        setcmd = b'G'
        print('setcmd: ', str_message)
        missionPort.write(setcmd)
```

② 환경센서와 직접적으로 연결하기 위한 인터페이스(Serial) 연동 예시코드

```
def missionPortOpening(missionPortNum, missionBaudrate):
    global airQ
    global missionPort

    if missionPort == None:
        try:
            missionPort = serial.Serial(missionPortNum, missionBaudrate, timeout=2)
            print('missionPort open. ' + missionPortNum + ' Data rate: ' + missionBaudrate)
            mission_thread = threading.Thread(
                target=missionPortData,
            )
            mission_thread.start()

        except TypeError as e:
            missionPortClose()
        else:
            if missionPort.is_open == False:
                missionPortOpen()
                send_data_to_msw(airQ)

def missionPortOpen():
    print('missionPort open!')
    missionPort.open()

def missionPortClose():
    global missionPort
    print('missionPort closed!')
    missionPort.close()

def missionPortError(err):
    print('[missionPort error]: ', err)
    os.kill(i_pid, signal.SIGKILL)

def airReqMessage():
    global missionPort
    if missionPort is not None:
        if missionPort.is_open:
            setcmd = b'I'
            missionPort.write(setcmd)
            flag = 0

def send_data_to_msw(obj_data):
    global lib_mqtt_client
    global data_topic
    lib_mqtt_client.publish(data_topic, obj_data)
```



### ③ 환경센서의 데이터를 파싱하기 위한 예시코드

```
def airQ_init():
    airQ['PM25'] = 0.0 # (ug/m3)
    airQ['PM10'] = 0.0 # (ug/m3)
    airQ['CO'] = 0.0 # (ppb)
    airQ['NO2'] = 0.0 # (ppb)
    airQ['O3_org'] = 0.0 # (org/ppb)
    airQ['O3_comp'] = 0.0 # (comp/ppb)
    airQ['SO2_org'] = 0.0 # (org/ppb)
    airQ['SO2_comp'] = 0.0 # (comp/ppb)
    airQ['T'] = 0.0 # (C)
    airQ['H'] = 0.0 # (%)
    airQ['NO2_OP1'] = 0 # (mV)
    airQ['NO2_OP2'] = 0 # (mV)
    airQ['O3_OP1'] = 0 # (mV)
    airQ['O3_OP2'] = 0 # (mV)
    airQ['CO_OP1'] = 0 # (mV)
    airQ['CO_OP2'] = 0 # (mV)
    airQ['SO2_OP1'] = 0 # (mV)
    airQ['SO2_OP2'] = 0 # (mV)

def missionPortOpen():
    print('missionPort open!')
    missionPort.open()

def missionPortClose():
    global missionPort
    print('missionPort closed!')
    missionPort.close()

def missionPortError(err):
    print('[missionPort error]:', err)
    os.kill(i_pid, signal.SIGKILL)

def airReqMessage():
    global missionPort

    if missionPort is not None:
        if missionPort.is_open:
            setcmd = b'I'
            missionPort.write(setcmd)
            flag = 0

def send_data_to_msw(obj_data):
    global lib_mqtt_client
```

```

global data_topic

lib_mqtt_client.publish(data_topic, obj_data)

def missionPortData():
    global missionPort
    global airQ
    global data_topic
    global control_topic
    global flag

    flag = 0
    airReqMessage()
    count = 0
    while True:
        missionStr = missionPort.readlines()
        try:
            if (not missionStr) or (missionStr[0] == b'\x00\n'):
                if not missionStr:
                    if count < 5:
                        count += 1
                        pass
                    else:
                        count = 0
                        airReqMessage()
                        flag = 0

                else:
                    airReqMessage()
                    flag = 0

            else:
                if flag == 0:
                    flag = 1
                    arrAIRQ = missionStr[3].decode("utf-8").replace(" ", "")
                    arrQValue = arrAIRQ.split(',')
                    airQ['PM25'] = float(arrQValue[0]) # (ug/m3)
                    airQ['PM10'] = float(arrQValue[1]) # (ug/m3)
                    airQ['CO'] = float(arrQValue[2]) # (ppb)
                    airQ['NO2'] = float(arrQValue[3]) # (ppb)
                    airQ['O3_org'] = float(arrQValue[4]) # (org/ppb)
                    airQ['O3_comp'] = float(arrQValue[5]) # (comp/ppb)
                    airQ['SO2_org'] = float(arrQValue[6]) # (org/ppb)
                    airQ['SO2_comp'] = float(arrQValue[7]) # (comp/ppb)
                    airQ['T'] = float(arrQValue[8]) # (C)
                    airQ['H'] = float(arrQValue[9]) # (%)
                    airQ['NO2_OP1'] = int(arrQValue[10]) # (mV)

```

```

        airQ['NO2_OP2'] = int(arrQValue[11]) # (mV)
        airQ['O3_OP1'] = int(arrQValue[12]) # (mV)
        airQ['O3_OP2'] = int(arrQValue[13]) # (mV)
        airQ['CO_OP1'] = int(arrQValue[14]) # (mV)
        airQ['CO_OP2'] = int(arrQValue[15]) # (mV)
        airQ['SO2_OP1'] = int(arrQValue[16]) # (mV)
        airQ['SO2_OP2'] = int(arrQValue[17]) # (mV)

        airQ = json.dumps(airQ)
        send_data_to_msw(airQ)
        airQ = json.loads(airQ)
    else:

        arrAIRQ = missionStr[0].decode("utf-8").replace(" ", "")
        arrQValue = arrAIRQ.split(',')
        airQ['PM25'] = float(arrQValue[0]) # (ug/m3)
        airQ['PM10'] = float(arrQValue[1]) # (ug/m3)
        airQ['CO'] = float(arrQValue[2]) # (ppb)
        airQ['NO2'] = float(arrQValue[3]) # (ppb)
        airQ['O3_org'] = float(arrQValue[4]) # (org/ppb)
        airQ['O3_comp'] = float(arrQValue[5]) # (comp/ppb)
        airQ['SO2_org'] = float(arrQValue[6]) # (org/ppb)
        airQ['SO2_comp'] = float(arrQValue[7]) # (comp/ppb)
        airQ['T'] = float(arrQValue[8]) # (C)
        airQ['H'] = float(arrQValue[9]) # (%)
        airQ['NO2_OP1'] = int(arrQValue[10]) # (mV)
        airQ['NO2_OP2'] = int(arrQValue[11]) # (mV)
        airQ['O3_OP1'] = int(arrQValue[12]) # (mV)
        airQ['O3_OP2'] = int(arrQValue[13]) # (mV)
        airQ['CO_OP1'] = int(arrQValue[14]) # (mV)
        airQ['CO_OP2'] = int(arrQValue[15]) # (mV)
        airQ['SO2_OP1'] = int(arrQValue[16]) # (mV)
        airQ['SO2_OP2'] = int(arrQValue[17]) # (mV)
        airQ = json.dumps(airQ)
        send_data_to_msw(airQ)
        airQ = json.loads(airQ)

except (ValueError, IndexError):
    airQ_init()
    airReqMessage()
    pass

except serial.SerialException as e:
    missionPortError(e)

```

#### ④ 임무 Lib가 시작되는 메인 코드 예시

```
def main():
    global lib_mqtt_client
    global control_topic
    global data_topic
    global broker_ip
    global port
    global lib

    my_lib_name = 'lib_sparrow_air'
    my_msw_name = 'msw'+ my_lib_name[3:] + '_' + 'msw'+ my_lib_name[3:]

    try:
        lib = dict()
        with open('./' + my_msw_name + '/' + my_lib_name + '.json', 'r') as f:
            lib = json.load(f)
            lib = json.loads(lib)

    except Exception as e:
        lib = dict()
        lib["name"] = my_lib_name
        lib["target"] = 'armv6'
        lib["description"] = "[name] [portnum] [baudrate]"
        lib["scripts"] = './' + my_lib_name + ' /dev/ttyUSB4 115200'
        lib["data"] = ['AIR']
        lib["control"] = ['Control_AIR']
        lib = json.dumps(lib, indent=4)
        lib = json.loads(lib)

        with open('./' + my_msw_name + '/' + my_lib_name + '.json', 'w', encoding='utf-
8') as json_file:
            json.dump(lib, json_file, indent=4)

    lib['serialPortNum'] = argv[1]
    lib['serialBaudrate'] = argv[2]

    control_topic = '/MUV/control/' + lib["name"] + '/' + lib["control"][0]
    data_topic = '/MUV/data/' + lib["name"] + '/' + lib["data"][0]

    msw_mqtt_connect()

    missionPortNum = lib["serialPortNum"]
    missionBaudrate = lib["serialBaudrate"]
    missionPortOpening(missionPortNum, missionBaudrate)

if __name__ == "__main__":
    main()
```

- 임무Lib는 임무SW와 연동을 위해 실행파일 형태로 배포한다. 파이썬으로 개발된 대기환경 측정 임무Lib는 아래의 명령어를 통해 응용프로그램 형태로 만들 수 있다.

```
$ python3 -m PyInstaller -F lib_sparrow_air.py
```

- Python으로 개발되어 응용프로그램 형태로 배포된 대기환경 측정 임무Lib는 아래 Github 주소에서 확인할 수 있다.

[https://github.com/loTKETI/lib\\_sparrow\\_air.git](https://github.com/loTKETI/lib_sparrow_air.git)

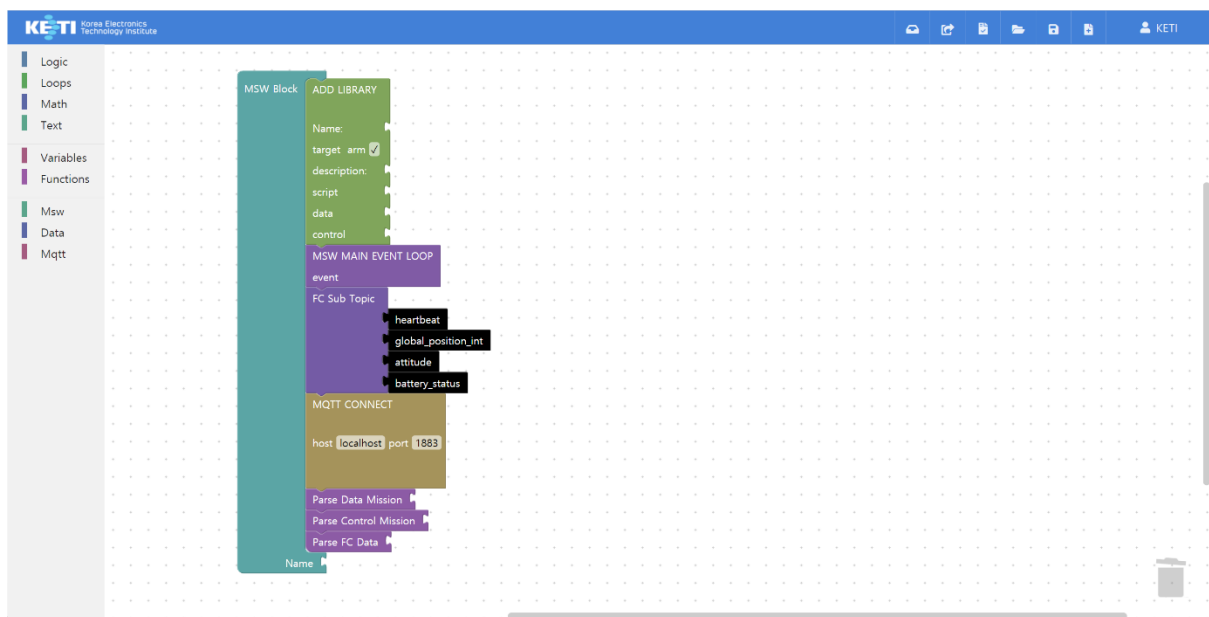
### 3.3 임무SW 개발

#### 3.3.1 MUV 개발도구

- 개발자가 임무SW를 손쉽게 개발하고 배포하여 MC에 자동으로 탑재까지 가능하도록 지원하는 블록코딩 형태의 웹 기반 지원도구이다.
- <http://203.253.128.177:7505> 주소로 접속하여 로그인하여 사용 가능하다.
- 코딩을 소스코드를 작성하는 형태가 아닌 블록코딩을 통해 쉽게 개발할 수 있으며 Github와 연동하여 업로드가 가능하고 approval이라는 명세를 입력함으로써 드론에 탑재가 가능하다.

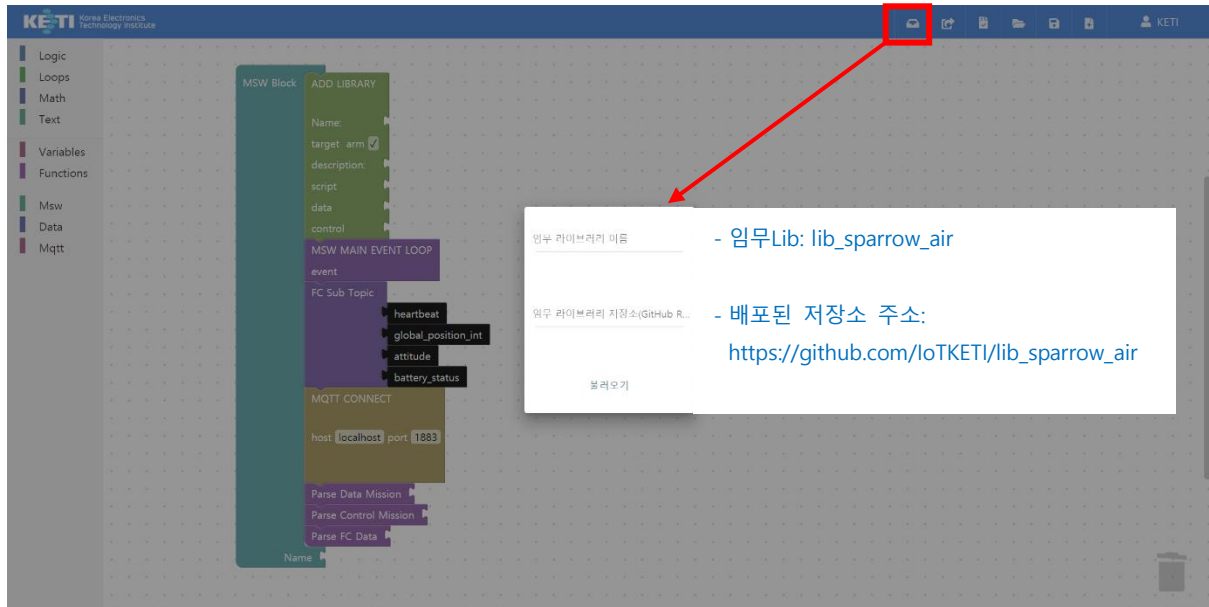
#### 3.3.2 임무 S/W 개발과정

- ① 아래 사진과 같이 임무SW의 기본이 되는 형태까지 블록으로 코딩한다. 이 구조는 임무SW도 동일한 구조를 가진다.



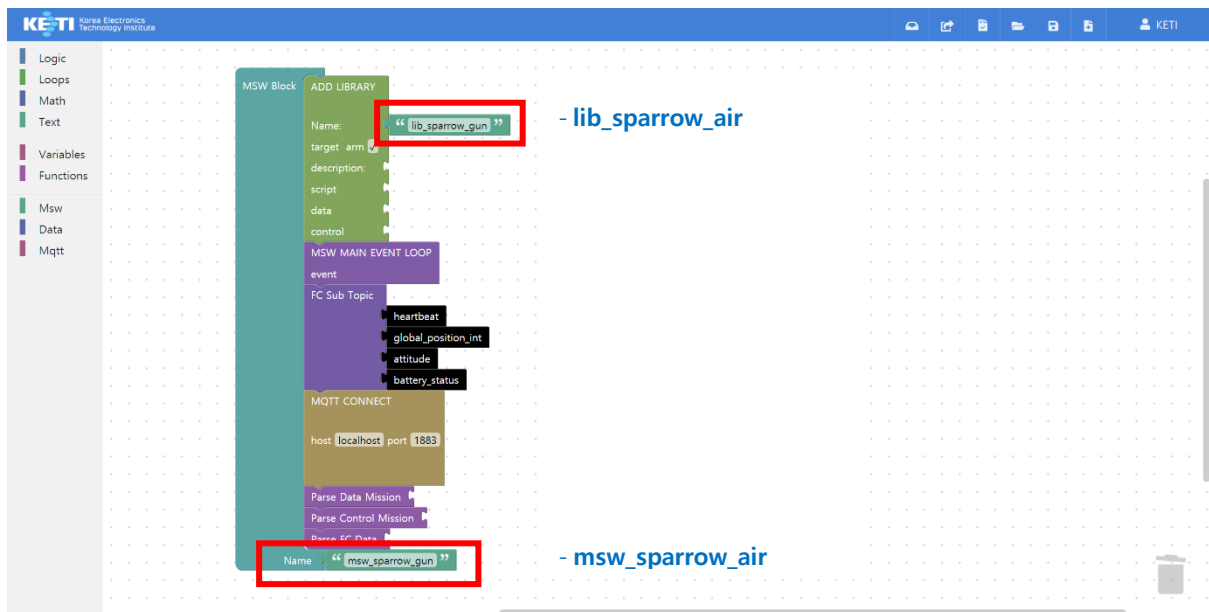
② 사전에 개발해서 배포한 임무Lib를 호출한다.

- ✓ 임무Lib 이름: 사전에 응용 프로그램 형태로 배포한 임무Lib의 이름
- ✓ 임무Lib 저장소: Github에 배포된 임무Lib를 가져오기 위한 주소

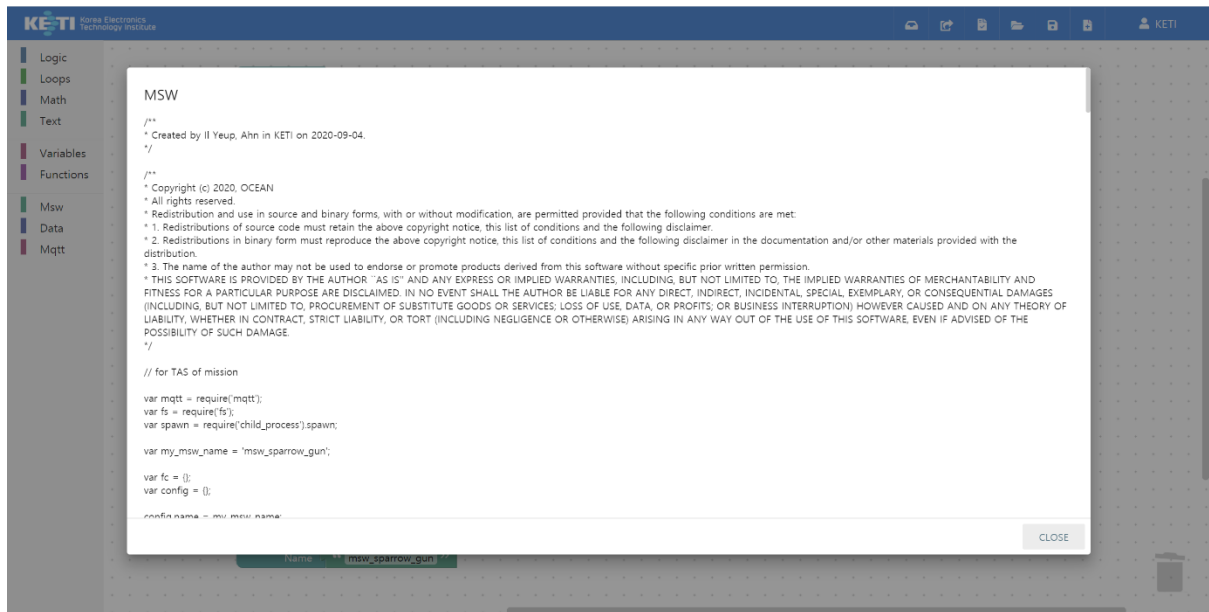


③ 생성된 임무Lib 블록 추가 및 임무SW의 이름 입력

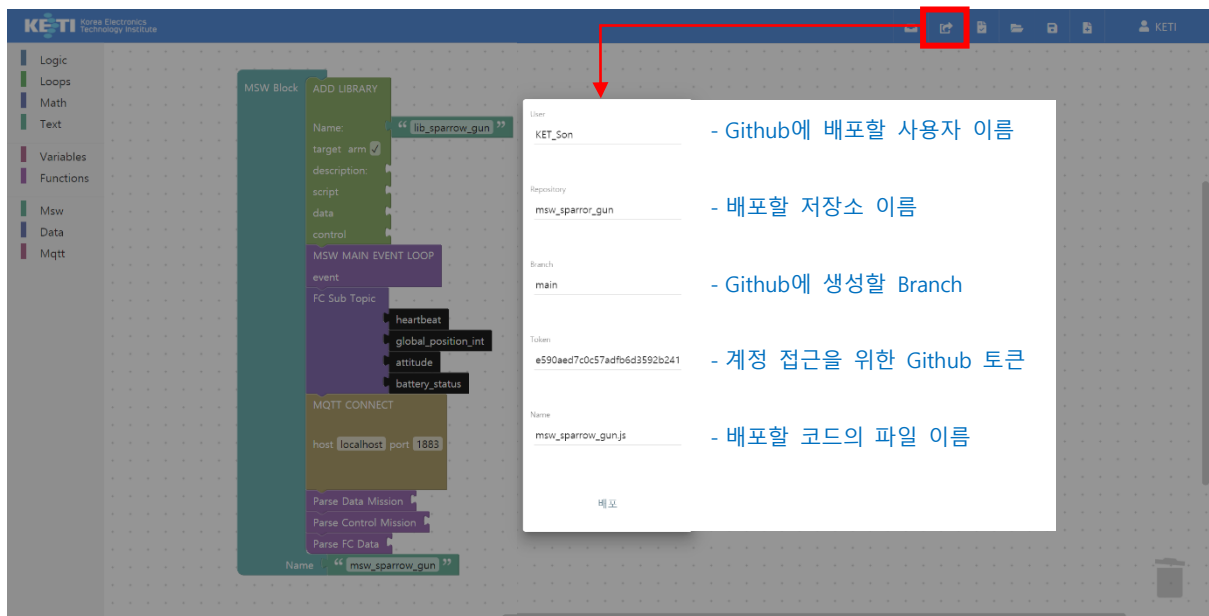
- ✓ lib\_sparrow\_gun 대신 lib\_sparrow\_air



④ 블록으로 코딩한 임무SW를 배포하기 전 소스코드 형태로 확인

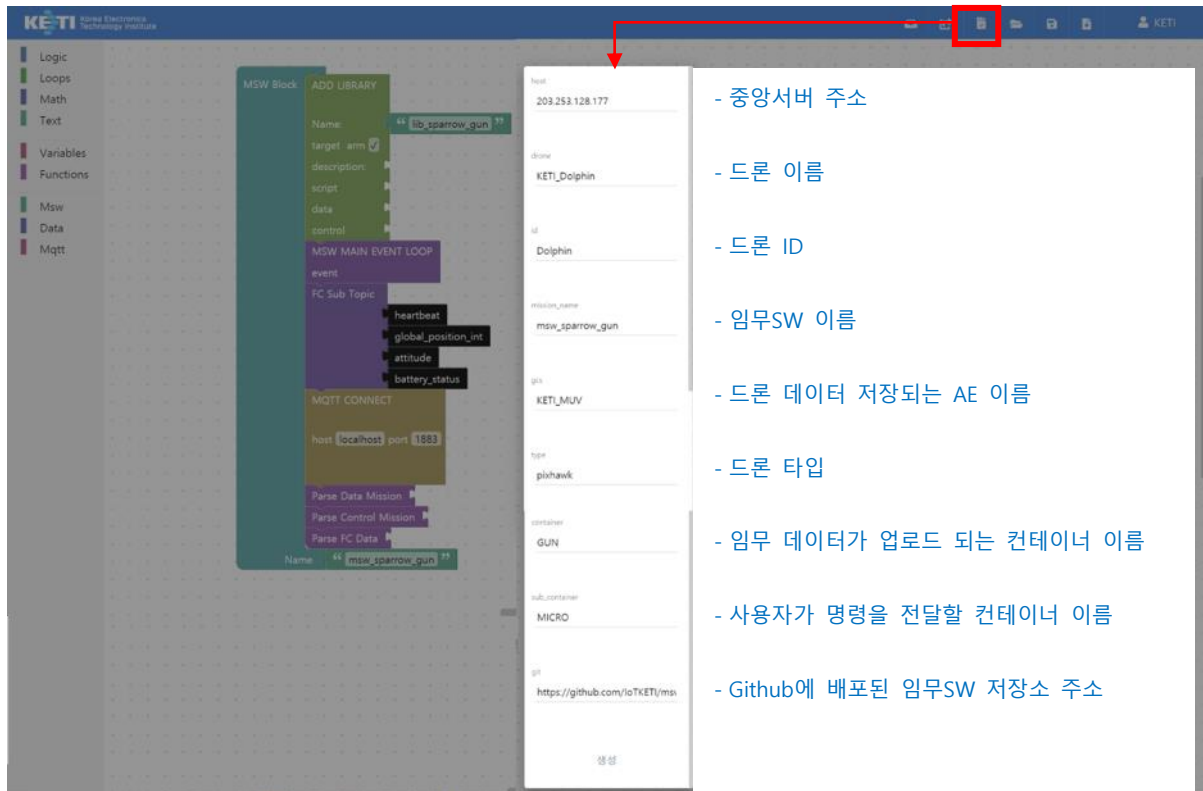


⑤ 임무SW 배포에 필요한 정보 입력하여 배포





## ⑥ 드론 탑재를 위한 드론 명세 작성



The screenshot shows the KETI nCube-MUV interface. On the left, there's a block-based programming area with 'MSW Block' and 'MSW MAIN EVENT LOOP' sections. On the right, there's a form for drone specifications. A red arrow points to the 'host' field.

Field	Value	Description
host	203.253.128.177	- 중앙서버 주소
drone	KETI_Dolphin	- 드론 이름
id	Dolphin	- 드론 ID
mission_name	msw_sparrow_gun	- 임무SW 이름
gcs	KETI_MUV	- 드론 데이터 저장되는 AE 이름
type	pixhawk	- 드론 타입
container	GUN	- 임무 데이터가 업로드 되는 컨테이너 이름
sub_container	MICRO	- 사용자가 명령을 전달할 컨테이너 이름
gti	https://github.com/loTKETI/msw	- Github에 배포된 임무SW 저장소 주소

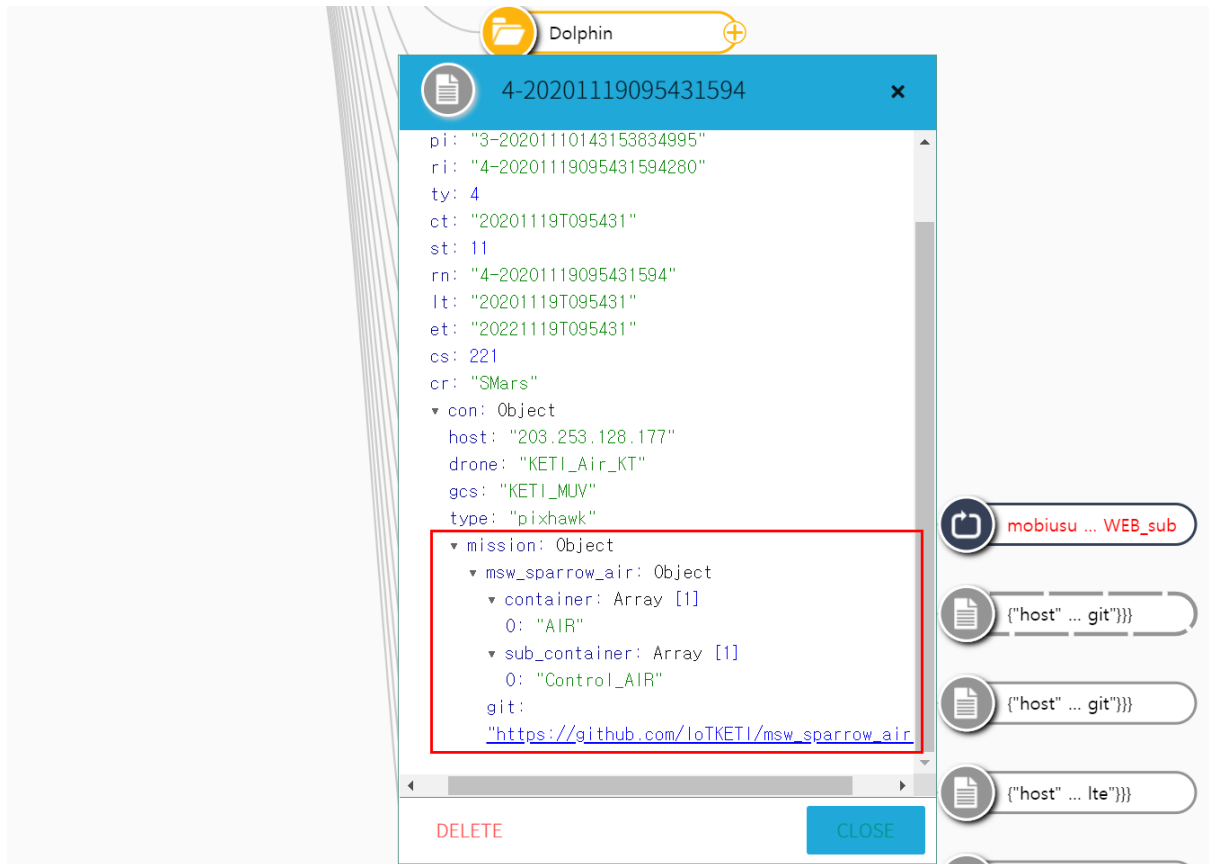
## ⑦ MC의 nCube-MUV 폴더에서 flight.json 정보 수정 (nCube-MUV 가이드 참고)

- ✓ flight에 ⑥의 과정에서 업로드한 명세의 드론 ID를 추가한다

```

{} flight.json U X
{} flight.json > ...
1  {
2      "approval_gcs": "MUV",
3      "flight": "Dolphin"
4  }
    
```

⑧ 명세 업로드 확인



⑨ 환경센서 에어펌프 제어 명령 전송

- ✓ /Mobius/{GCS 이름}/Mission\_Data/{드론 이름}/msw\_sparrow\_air/**Control\_AIR** 컨테이너 아래에 Content Instance를 생성하여 con값에 에어펌프 제어명령인 "G"를 입력한다.
- ✓ 환경센서가 제어명령을 전달받으면 환경센서 내의 에어펌프가 동작한다.

