SMELLBAD

# Galago

Project Description

powered by

**Robot Operating System**

## Ⅰ. 개발 배경 및 목적

기존의 국립공원 내 불법 구조물, 야생동물에 대한 조사를 실시할 경우 관련 담당자가 직접 탐방로를 따라 조사를 해야 하는 번거로움으로 인해 조사의 불편함이 증대되고 있음. 이에 드론을 활용하여 경로를 사전 지정 및 비행을 실시하여 불편함을 해소하고 지속적인 관리 / 감독이 이루어질 수 있도록 것에 대해 개발의 목적이 있음.

### 기존의 산림공원 조사 예시

불법구조물 감시활동


반달곰 생태계조사


산불감시활동


탐방로 안전관리

담당자가 직접 탐방로 조사를 해야 하므로 조사/감독 시 소요되는 시간의 증대

## II. 기대 효과

ROS를 활용한 Mission Planning 기능으로 조사/감시 활동의 시간을 최소화 하고 효율성을 증대함. 또한 이러한 자율주행 기술은 산림공원 뿐만 아니라 소방, 치안, 국토관리 등 주기적인 감시/지도 활동이 필요한 영역으로의 확대를 통해 4차산업혁명의 중심인 드론 산업의 부흥까지 이끌어 낼 수 있음.

고속도로 드론 단속

산림 병해충 감시 / 방제

건물의 기획, 설계, 시공, 유지관리하는 스마트 건설 기술

실종자 수색 드론 사용

**4차 산업혁명의 핵심인 국내 드론 산업의 육성**

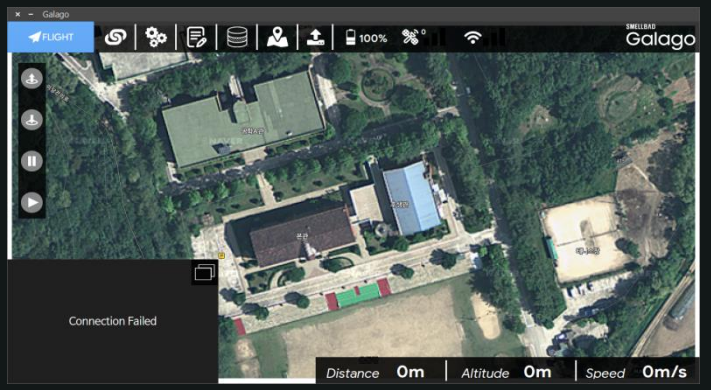
## II. 기대 효과

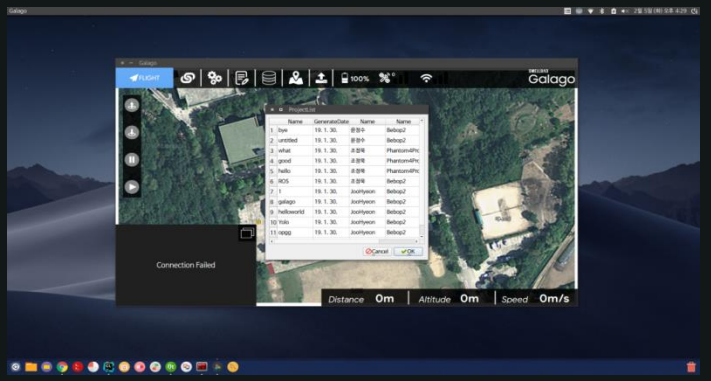
수업시간에 배운 일반 C++, My SQL, ROS 이외에도 JAVA script, STL, QT library, Open CV 등을 추가로 활용함으로써 개발자로서 다양한 라이브러리를 경험해보고 사용해볼 수 있도록 구현하고 ROS 뿐만 아니라 다양한 환경에서 활용할 수 있는 확장성을 고려한 개발을 통해 향후 시스템 개발 분야 취업 시 이러한 부분에 대해 고려할 수 있는 상황에 대해 간접적인 체험을 할 수 있음.


ROS를 활용한 미션 플래닝 프로젝트 " Galago "

## Ⅲ. Project 개요

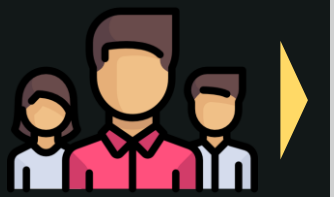
| | |
|---|---|
| 프로젝트명 | Smellbad Galago |
| 개발기간 | 2019. 01. 09 ~ 2019. 02. 15 (4주) |
| 개발범위 | • 드론 비행을 위한 Sub / Pub 코드 작성<br>• Waypoint 관리를 위한 Database 구축<br>• 사용자 경험(UX)를 고려한 전체 UI 디자인 |
| 의존성사항 | • bebop autonomy<br>• QT 5.12.0 이상<br>• open CV 4.0 이상<br>• mySQL 5.0 이상<br>• roscpp<br>• sensor_msgs<br>• image_transport<br>• cv_bridge |
| 업무분장 | • 윤정수 – 프로젝트 총괄 / UI 디자인<br>• 조정묵 – Waypoint 비행 코드 개발<br>• 이주현 – Waypoint 저장용 Database 구축 |




## Ⅳ. 시스템 구성도

Galago Mission Planner는 UI 영역 / Code 영역 / DB영역으로 계층화 하여 시스템을 개발하였으며 실질적인 무인비행체로의 비행기능은 ROS Platform을 활용하여 비행이 이루어질 수 있도록 함.



**UI영역**

- Main UI
- New Project
- Open Project
- Waypoint List
- Take off
- Landing
- Start Mission

**Code영역**

- Take off
  Land
  Altitude
  Latitude
  Longitue
  Battery status
  GPS status
  Wifi status
- Move
- Waypoint Marking
  Get Now Position
- Insert Waypoint
  Update Waypoint
  Insert Project

**Data영역**

- Pilot Data
- Aircraft Data
- Project Data
- Waypoint Data

**구동영역**

- /Image_view
- /cmd_vel

## V. 세부내용 – UI 부문

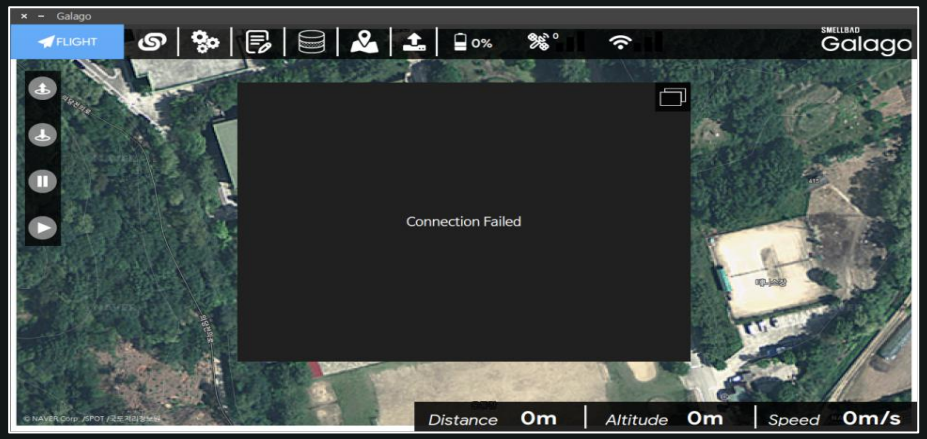
Galago의 UI는 실제 Mission Planner 프로그램들의 UI디자인을 기초로 하여 구성하였으며 실시간 드론의 위치정보 및 Mission Flight가 진행될 Point 지점이 표시되는 지도를 중심으로 상하로 소프트웨어 구동에 필요한 버튼 및 상태정보를 구현함으로서 해당 소프트웨어에서 모든 조작이 이루어질 수 있도록 구현하였음.

### Main UI 구성도

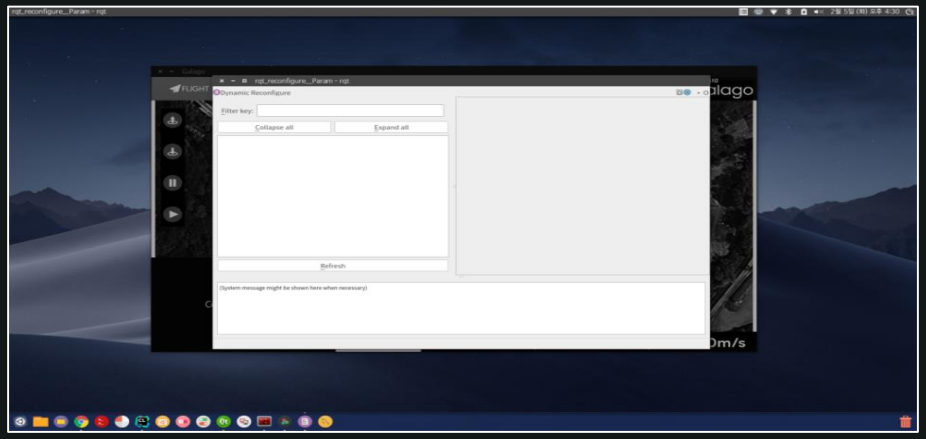
# V. 세부내용 – UI 부문

상단의 버튼 클릭 시 프로젝트의 생성 / 불러오기 / 프로젝트 내 WaypointList등이 표시될 수 있는 추가 UI를 구성하였으며, 드론 기기의 속도 / 고도 제한 설정은 rqt_reconfigure을 사용하여 UI를 구현하였음.

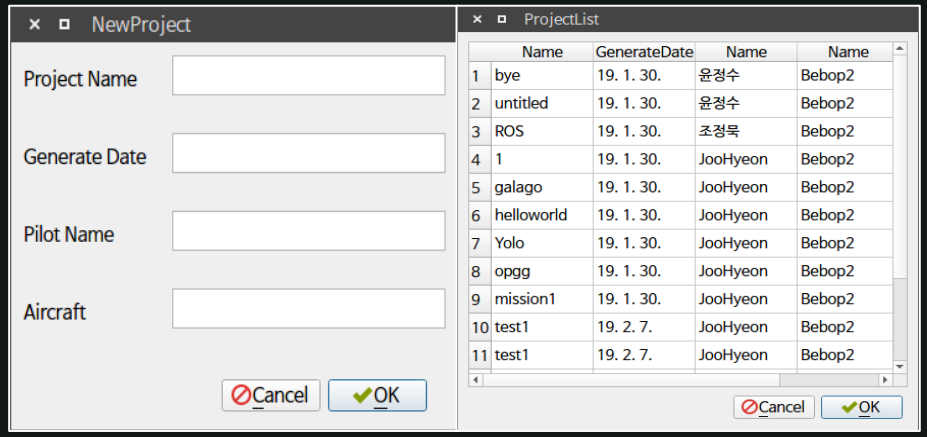
## FPV화면 확대 UI




## 기기 설정(rqt_reconfigure)

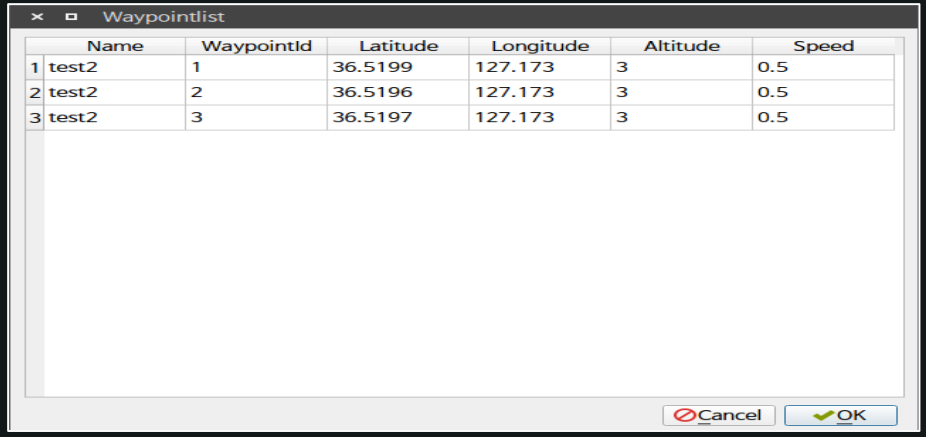


## 신규 프로젝트 생성 / 프로젝트 불러오기




| | Name | GenerateDate | Name | Name |
|---|---|---|---|---|
| 1 | bye | 19. 1. 30. | 윤정수 | Bebop2 |
| 2 | untitled | 19. 1. 30. | 윤정수 | Bebop2 |
| 3 | ROS | 19. 1. 30. | 조정묵 | Bebop2 |
| 4 | 1 | 19. 1. 30. | JooHyeon | Bebop2 |
| 5 | galago | 19. 1. 30. | JooHyeon | Bebop2 |
| 6 | helloworld | 19. 1. 30. | JooHyeon | Bebop2 |
| 7 | Yolo | 19. 1. 30. | JooHyeon | Bebop2 |
| 8 | opgg | 19. 1. 30. | JooHyeon | Bebop2 |
| 9 | mission1 | 19. 1. 30. | JooHyeon | Bebop2 |
| 10 | test1 | 19. 2. 7. | JooHyeon | Bebop2 |
| 11 | test1 | 19. 2. 7. | JooHyeon | Bebop2 |

## 프로젝트 Waypoint 리스트




| | Name | WaypointId | Latitude | Longitude | Altitude | Speed |
|---|---|---|---|---|---|---|
| 1 | test2 | 1 | 36.5199 | 127.173 | 3 | 0.5 |
| 2 | test2 | 2 | 36.5196 | 127.173 | 3 | 0.5 |
| 3 | test2 | 3 | 36.5197 | 127.173 | 3 | 0.5 |

# V. 세부내용 – UI 부문

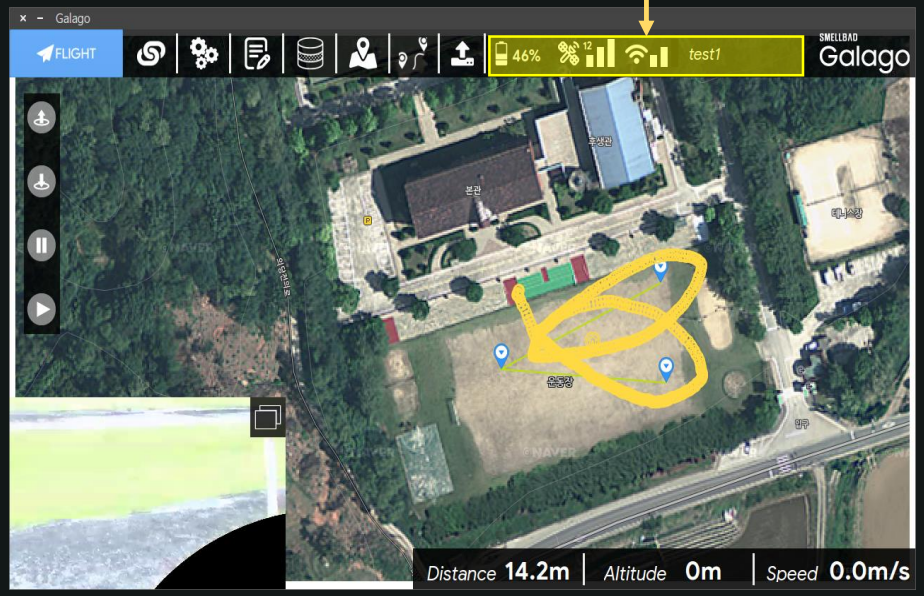
Galago Mission Planner는 UI 영역 / Code 영역 / DB영역으로 계층화 하여 시스템을 개발하였으며 실질적인 무인비행체로의 비행기능은 ROS Platform을 활용하여 비행이 이루어질 수 있도록 함.

## UI 구현 Code

```
void Galago::receiveGPS(int GPS)
{
    if(GPS > 10)
    {
        ui->gps_number->setText(QString::number(GPS));
        ui->gps_good->setStyleSheet("background-color: rgb(255, 255, 255)");
        ui->gps_normal->setStyleSheet("background-color: rgb(255, 255, 255)");
        ui->gps_weak->setStyleSheet("background-color: rgb(255, 255, 255)");
    }
    else if(GPS > 6)
    {
        ui->gps_number->setText(QString::number(GPS));
        ui->gps_good->setStyleSheet("background-color: rgb(0, 0, 0)");
        ui->gps_normal->setStyleSheet("background-color: rgb(255, 255, 255)");
        ui->gps_weak->setStyleSheet("background-color: rgb(255, 255, 255)");
    }
    else if(GPS > 2)
```

```
Galago::Galago(int argc, char** argv, QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::Galago),
    video(argc, argv),
    pub(argc, argv),
    PtoP(argc, argv)

    isBigView = false;
    isTakeoff = false;
    ui->setupUi(this);

    m_view = ui->webView;
    m_view->setGeometry(0,0,1280,700);
    m_view->setUrl(QUrl("file:///home/yjs/catkin_ws/src/galago/src/ui/map.html"));

    m_locInfo = new locationinfo(this);
    p_list = new ProjectList(this);
    p_new = new NewProject(this);
    dialog = new Dialog(this);
    w_list = new Waypointlist(this);

    video.init();
    connect(m_view->page()->mainFrame(),SIGNAL(javaScriptWindowObjectCleared()), this, SLOT(AddObject()));
    connect(&video, SIGNAL(sendFrame(QImage)), this, SLOT(receiveFrame(QImage)));
    connect(&video, SIGNAL(sendBatteryPercent(int)), this, SLOT(receivebattery(int)));
    connect(&video, SIGNAL(sendNumOfSatellite(int)), this, SLOT(receiveGPS(int)));
    connect(&video, SIGNAL(sendAltitude(double)), this, SLOT(receiveAltitude(double)));
    connect(&video, SIGNAL(sendWifi(int)), this, SLOT(receiveWifi(int)));
    connect(&video, SIGNAL(sendSpeed(double)), this, SLOT(receiveSpeed(double)));
    connect(&video, SIGNAL(sendLocation(double, double)), this, SLOT(receiveLocation(double, double)));

    connect(p_list, SIGNAL(sendProjectList(QString)), this, SLOT(receiveProjectName(QString)));
    connect(p_new, SIGNAL(sendProjectName(QString)), this, SLOT(receiveProjectName(QString)));
    connect(dialog, SIGNAL(sendWaypoint(double, double, double, double)), this, SLOT(receiveWaypoint(double, double, double, double)));
    connect(&pub, SIGNAL(sendSaveHome()), this, SLOT(receiveSavehome()));

    connect(ui->takeoff_button, SIGNAL(clicked()), &pub, SLOT(Takeoff()));
    connect(ui->landing_button, SIGNAL(clicked()), &pub, SLOT(Land()));
}
```

```
header:
  seq: 46
  stamp:
    secs: 1550195952
    nsecs: 304201140
  frame id: "base link"
rssi: -42
---
```



# V. 세부내용 – Code 부문

기체의 이륙지점 / 현재 위치정보를 지도에 표시하기 위해 Subscribe된 드론의 위치정보를 Javascript로 전달하여 지도상에 마커가 표시될 수 있도록 구현함.

## 드론의 위치정보 UI 구현 Code 흐름도

```
void Video::PositionCallback(const bebop_msgs::Ardrone3PilotingStatePositionChangedConstPtr& msg)
{
    double latitude = msg->latitude;
    double longitude = msg->longitude;

    Q_EMIT sendLocation(latitude, longitude);
}
```

```
    {
        if(latitude != 500 && longitude != 500)
        {
            QString Latitude = QString::number(latitude, 'g', 8);
            QString Longitude = QString::number(longitude, 'g', 9);
            m_view->page()->mainFrame()->evaluateJavaScript(QString("nowposition(%1,%2)").arg(Latitude).arg(Longitude));
        }
        calculateDistance(latitude, longitude);
    }
}
```

```
function nowposition(lat, lng)
{
    var position = new naver.maps.LatLng(lat, lng);

    var markerOptions = {
        position: position.destinationPoint(90, 15),
        map: map,
        icon: {
        url: '/home/yjs/catkin_ws/src/galago/src/icon/dot-and-circle-yellow.png',
        size: new naver.maps.Size(30, 30),
        origin: new naver.maps.Point(0, 0),
        anchor: new naver.maps.Point(25, 26)
        }
    };
    var marker = new naver.maps.Marker(markerOptions);
    map.panTo(position);
}
```

# V. 세부내용 – Code 부문

기기의 상태정보, Mission Flight 등 UI구현과 동시에 코드가 실행되어야 하는 상황인 경우 Qt 라이브러리에서 제공하는 Qthread 객체를 이용하여 Multi-Thread 시스템이 구현될 수 있도록 함.

## 기기 상태정보 Subscribe Thread

```
bool Video::init()
{
    ros::init(init_argc, init_argv, "image_view");
    if(!ros::master::check())
        return false;

    ros::start();
    ros::NodeHandle nh;

    image_transport::ImageTransport it(nh);
    sub = it.subscribe(cam_image_topic, 1, &Video::imageCallback, this);

    batsub = nh.subscribe(battery_topic, 1, &Video::batteryCallback, this);
    gpssub = nh.subscribe(gps_topic, 1, &Video::satelliteCallback, this);
    altitudesub = nh.subscribe(altitude_topic, 1, &Video::AltitudeCallback, this);
    wifisub = nh.subscribe(wifi_topic, 1, &Video::wifiCallback, this);
    speed = nh.subscribe(speed_topic, 1, &Video::speedCallback, this);
    location = nh.subscribe(loc_topic, 1, &Video::PositionCallback, this);
    start();

    return true;
}

void Video::run()
{
    ros::NodeHandle nh;

    image_transport::ImageTransport it(nh);
    sub = it.subscribe(cam_image_topic, 1, &Video::imageCallback, this);
    ros::spin();

    Q_EMIT rosShutdown();
}
```

## Mission Flight Thread

```
void PointToPoint::run()
{
    move();
}

bool PointToPoint::move()
{
    list<DNode>::iterator it = List.begin();
    while(it !=  List.end())
    {
        DNode waypoint = *it;

        double g_lati_radian = waypoint.Latitude * RADIAN;
        double g_long_radian = waypoint.Longitude * RADIAN;

        while (true) {
            try {
                ros::spinOnce();
            } catch (...) {
                ROS_ERROR("--- ERROR IN spin(), shutting down! ---");
                ros::shutdown();
            }
            float linXYZ_AngZ[4] = {0, 0, 0, 0};
            double c_lati_radian = this->c_lati * RADIAN;
            double c_long_radian = this->c_long * RADIAN;

            double dist_rad = acos(sin(c_lati_radian) * sin(g_lati_radian) + cos(c_lati_radian)
                                                                          * cos(g_lati_radian) * cos(c_long_radian - g_long_radian));

            double dist = dist_rad * ANGLE * METER;

            double radian = acos((sin(g_lati_radian) - sin(c_lati_radian) * cos(dist_rad))
                                 / (cos(c_lati_radian) * sin(dist_rad)));

            double c_bearing = this->c_yaw * ANGLE;
            if (c_bearing < 0)
                c_bearing = 360 - abs(c_bearing);

            double true_bearing = radian * ANGLE;
            if (waypoint.Longitude < this->c_long)
                true_bearing = 360 - true_bearing;
```

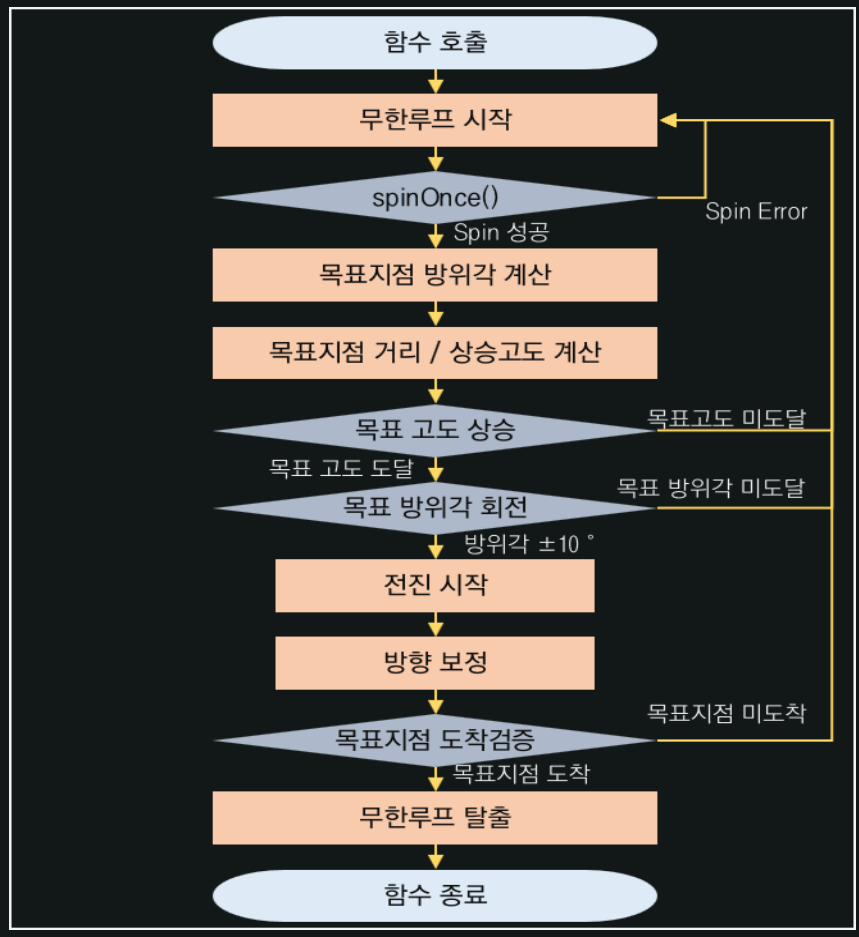
UI구현과 동시에 함수가 실행되는 Multi-Thread 구현

# V. 세부내용 – Code 부문

실제 비행에 필요한 명령을 내리는 함수는 ROS 패키지에 포함되어 있는 MAVLINK를 사용하는 것이 아닌 자체 알고리즘을 통한 구현으로 기체가 이동할 수 있도록 구현하였으며, 최종 목표지점 GPS 좌표 기준 3M 이내에 도달 시 다음 포인트로 기체가 이동할 수 있도록 함수를 구현함

## Point to Point Flight 코드 및 알고리즘 순서도

```
bool PointToPoint::Move(const double g_lati, const double g_long, const double g_alti, const double speed) {
    while (true) {
        try {
            ros::spinOnce();
        } catch (...) {
            ROS_ERROR("--- ERROR IN spin(), shutting down! ---");
            ros::shutdown();
        }

        float linXYZ_AngZ[3] = {0, 0, 0};
        double c_lati_radian = this->c_lati * (PI / 180);
        double c_long_radian = this->c_long * (PI / 180);
        double g_lati_radian = g_lati * (PI / 180);
        double g_long_radian = g_long * (PI / 180);

        double dist = ((((acos(sin(c_lati * PI / 180.0) * sin(g_lati * PI / 180.0) +
                      cos(c_lati * PI / 180.0) * cos(g_lati * PI / 180.0) * cos((c_long - g_long) *
                      PI / 180.0))) * 180.0 / PI) * 60.0 * 1.1515) * 1609.344) - 0.0;

        double dist_rad = acos(sin(c_lati_radian) * sin(g_lati_radian) + cos(c_lati_radian)
                          * cos(g_lati_radian) * cos(c_long_radian - g_long_radian));

        double radian = acos((sin(g_lati_radian) - sin(c_lati_radian) * cos(dist_rad))
                        / (cos(c_lati_radian) * sin(dist_rad)));

        double c_bearing = this->c_yaw * (180 / PI);
        if (c_bearing < 0)
            c_bearing = 360 - abs(c_bearing);

        double true_bearing = (acos((sin(g_lati_radian) - sin(c_lati_radian) * cos(dist_rad))
                              / (cos(c_lati_radian) * sin(dist_rad)))) * (180 / PI);
        if (g_long < this->c_long)
            true_bearing = 360 - true_bearing;

        if (true_bearing < c_bearing + 10 && true_bearing > c_bearing - 10 && dist > LIMITE_DIST)
            linXYZ_AngZ[0] = 0.5 * speed;
        else if (true_bearing < c_bearing + 30 && true_bearing > c_bearing - 30 && dist > LIMITE_DIST)
            linXYZ_AngZ[0] = 0;
        else linXYZ_AngZ[0] = 0.01;

        this->twist.linear.x = linXYZ_AngZ[0];
        this->twist.linear.z = linXYZ_AngZ[1];
        this->twist.angular.z = linXYZ_AngZ[2];
        this->ptop_pub.publish(twist);

        ROS_INFO("goal : %.2lf , Current : %.2lf, dist : %.2lf", true_bearing, c_bearing, dist);

        if (dist < LIMITE_DIST) break;
    }
}
```



## V. 세부내용 – Data 부문

Galago는 현장에서 직접 목표 Waypoint를 지정하고 비행 준비를 하여 비행을 하는 일련의 과정 중 목표 Waypoint를 비행지역 도착 전 사무실 같은 실내에서 사전에 지정하고 불러올 수 있도록 Database를 추가로 구축, 운영하여 사전 비행계획 수립 후 현장도착과 동시에 바로 Mission Flight가 이루어질 수 있도록 함.

### Database Table 구성도

**Pilot**

| Column 명 | 자료형 | NULL 허용 여부 | 비고 |
|---|---|---|---|
| Pilot Id | int | Not null | PK |
| Name | nvarchar(50) | Not null | |

**Project**

| Column 명 | 자료형 | NULL 허용 여부 | 비고 |
|---|---|---|---|
| Project Id | int | Not null | PK |
| Name | nvarchar(50) | Not null | |
| Generate Date | date | Not null | |
| Pilot Id | Int | Not null | |
| Aircraft Id | int | Not null | |

**Waypoint**

| Column 명 | 자료형 | NULL 허용 여부 | 비고 |
|---|---|---|---|
| Waypoint Id | int | Not null | PK |
| Project Id | int | Not null | PK |
| Latitude | int | Not null | |
| Longtitude | date | Not null | |
| Altitude | Int | Not null | |
| Aircraft Id | int | Not null | |

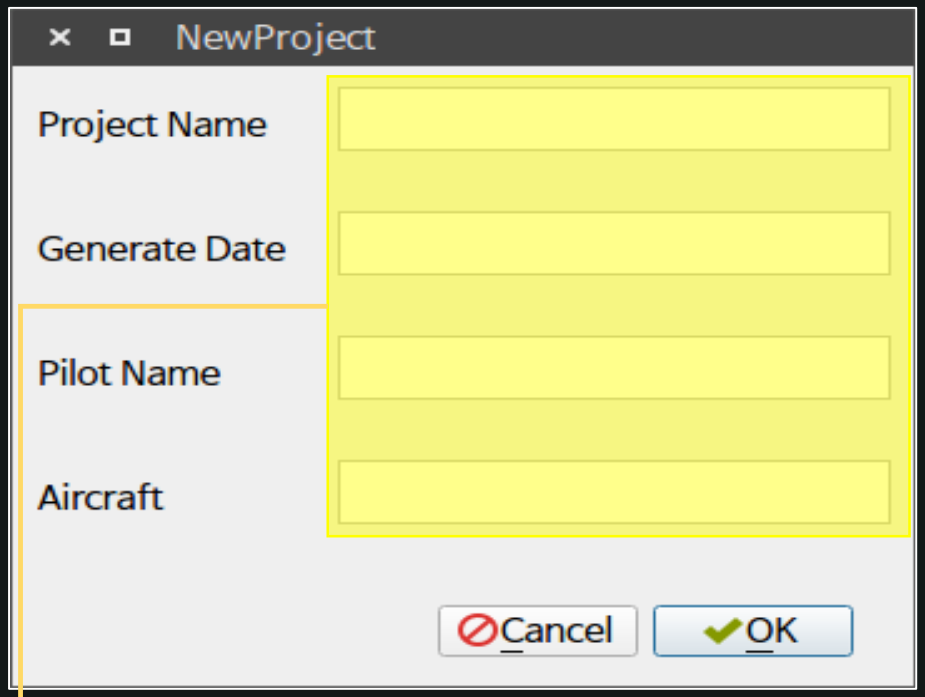
**Aircraft**

| Column 명 | 자료형 | NULL 허용 여부 | 비고 |
|---|---|---|---|
| Aircraft Id | int | Not null | PK |
| Name | nvarchar(50) | Not null | |

# V. 세부내용 – Data 부문

신규 프로젝트 생성 시 UI를 통해 값을 입력 받으며 입력 받은 값을 Database의 Stored Procedure 쿼리를 입력함으로써 Database에 순차적으로 정보가 입력될 수 있도록 하며, 프로젝트 또는 Waypoint를 불러오는 경우 UI 상단에 표시된 프로젝트명을 기준으로 하여 Procedure를 호출, 실행될 수 있도록 구현함.

## 신규 프로젝트 생성 흐름도




```
void NewProject::on_buttonBox_accepted()
{
    QString ProjectName = ui->ProjectName->toPlainText();
    QString GenerateDate = ui->GenerateDate->toPlainText();
    QString PilotName = ui->PilotName->toPlainText();
    QString AircraftName = ui->AircraftName->toPlainText();
```

```
    queryModel = new QSqlQueryModel();
    queryModel->setQuery("call Galago.Insert_Project('"+ProjectName+"','"+GenerateDate+"','"+PilotName+"','"+AircraftName+"')");

    Q_EMIT sendProjectName(ProjectName);

    this->close();
}
```

Result Grid | Filter Rows: | Edit: | Export/Impo

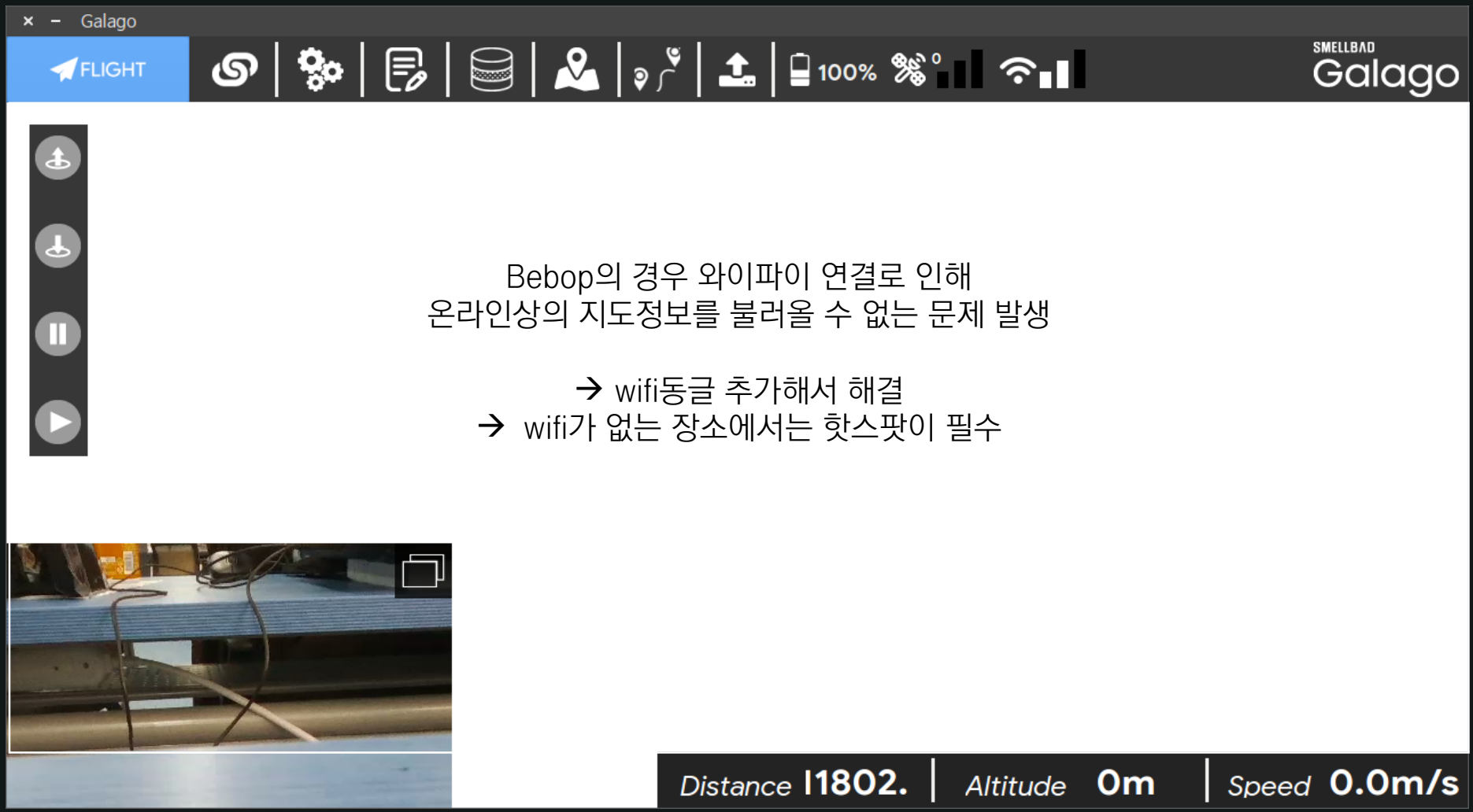
| # | ProjectId | Name | GenerateDate | PilotId | AircraftId |
|---|---|---|---|---|---|
| 1 | 1 | bye | 2019-01-30 | 1 | 1 |
| 2 | 2 | what | 2019-01-30 | 2 | 2 |
| 3 | 3 | good | 2019-01-30 | 2 | 2 |
| 4 | 4 | untitled | 2019-01-30 | 1 | 1 |
| 5 | 5 | hello | 2019-01-30 | 2 | 2 |
| 6 | 6 | ROS | 2019-01-30 | 4 | 1 |
| 7 | 7 | 1 | 2019-01-30 | 5 | 1 |
| 8 | 8 | galago | 2019-01-30 | 5 | 1 |
| 9 | 9 | helloworld | 2019-01-30 | 5 | 1 |
| 10 | 10 | Yolo | 2019-01-30 | 5 | 1 |
| 11 | 11 | opgg | 2019-01-30 | 5 | 1 |
| 12 | 12 | mission1 | 2019-01-30 | 5 | 1 |
| 13 | 13 | test1 | 2019-02-07 | 5 | 1 |
| 14 | 14 | test1 | 2019-02-07 | 5 | 1 |
| 15 | 15 | test2 | 2019-02-12 | 5 | 1 |
| 16 | 16 | Test3 | 2019-02-12 | 6 | 1 |

Project 1

# VI. 구동 영상

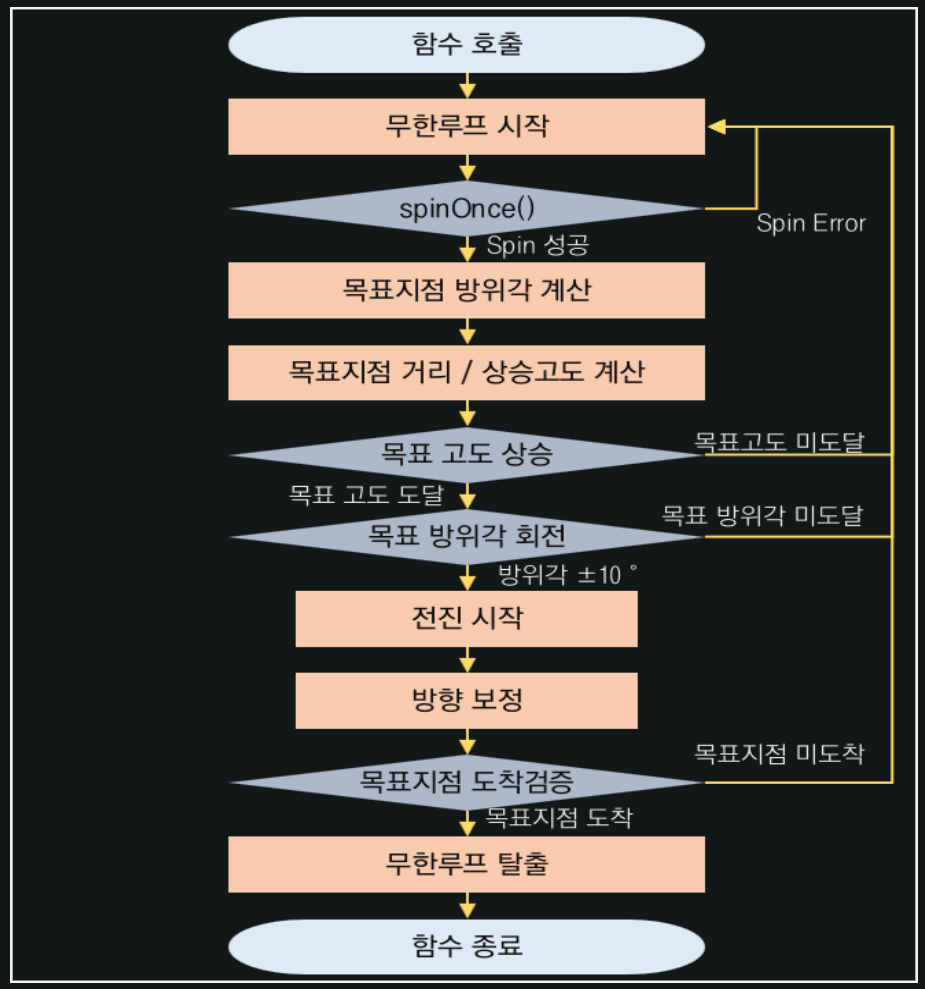
## Ⅶ. 개발 개선 사항 – 지도 호출을 위한 애로사항

현재 Galago에 구현되어 있는 지도 API는 인터넷에 연결되어 있어야만 표시가 가능한 지도로, Bebop을 위한 wifi와 지도 표현을 위한 인터넷 연결 두가지가 동시에 이루어져야 하는 애로사항이 발생하였음.



## Ⅶ. 개발 개선 사항 – 자체 알고리즘 구현에 따른 애로사항

MAVLINK 사용 시 보다 깔끔한 기체 이동이 가능하지만, GPS 신호 끊김 등 발생 시 기체가 예상할 수 없는 경로로 구현되는 경우가 많아 자체 알고리즘으로 code를 구현하였음. 그러나 해당 code의 경우 경로 수정을 위해 지속적인 기수 회전으로 간헐적으로 곡선 형태를 그리며 경로비행을 하는 경우 발생




## Ⅶ. 개발 개선 사항 – 자체 알고리즘 구현에 따른 애로사항

현재 Parrot社에서 제공하는 Mission Planning과 Galago의 경우 기수방향이 다음 목적지를 향하고 이동을 하고 있어 측면이동 등 사용자가 원하는 방향으로의 기수 이동이 불가능하다는 단점이 있음.

### MAVLINK를 통한 Mission Planning 시 기수방향

# 감사합니다

우린 이제 집에 Galago