PX4-ROS2 기반 자율주행 UGV 플랫폼

Date : 2023. 11. 14

이동욱

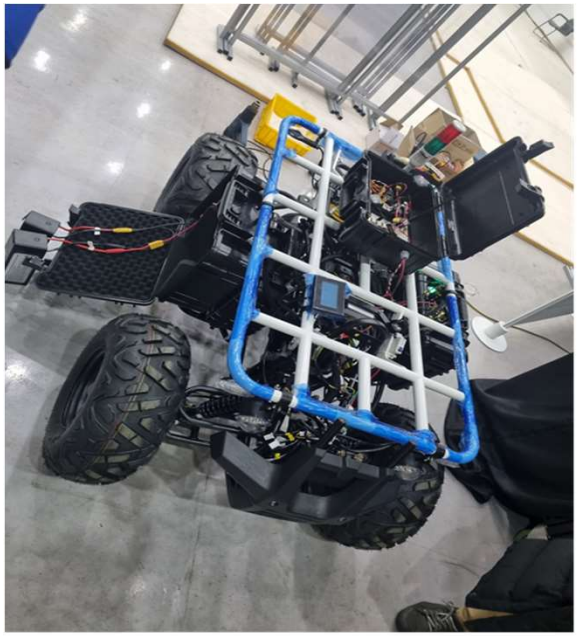
공주대학교 기계자동차공학부

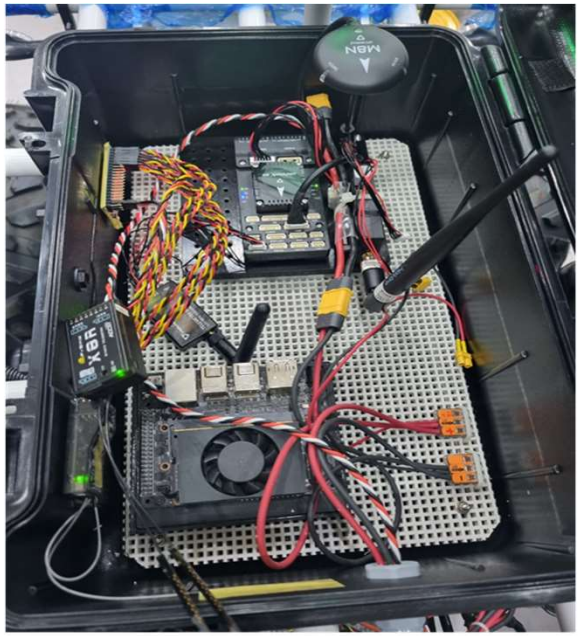
목차

1. 자율주행 UGV 플랫폼 소개

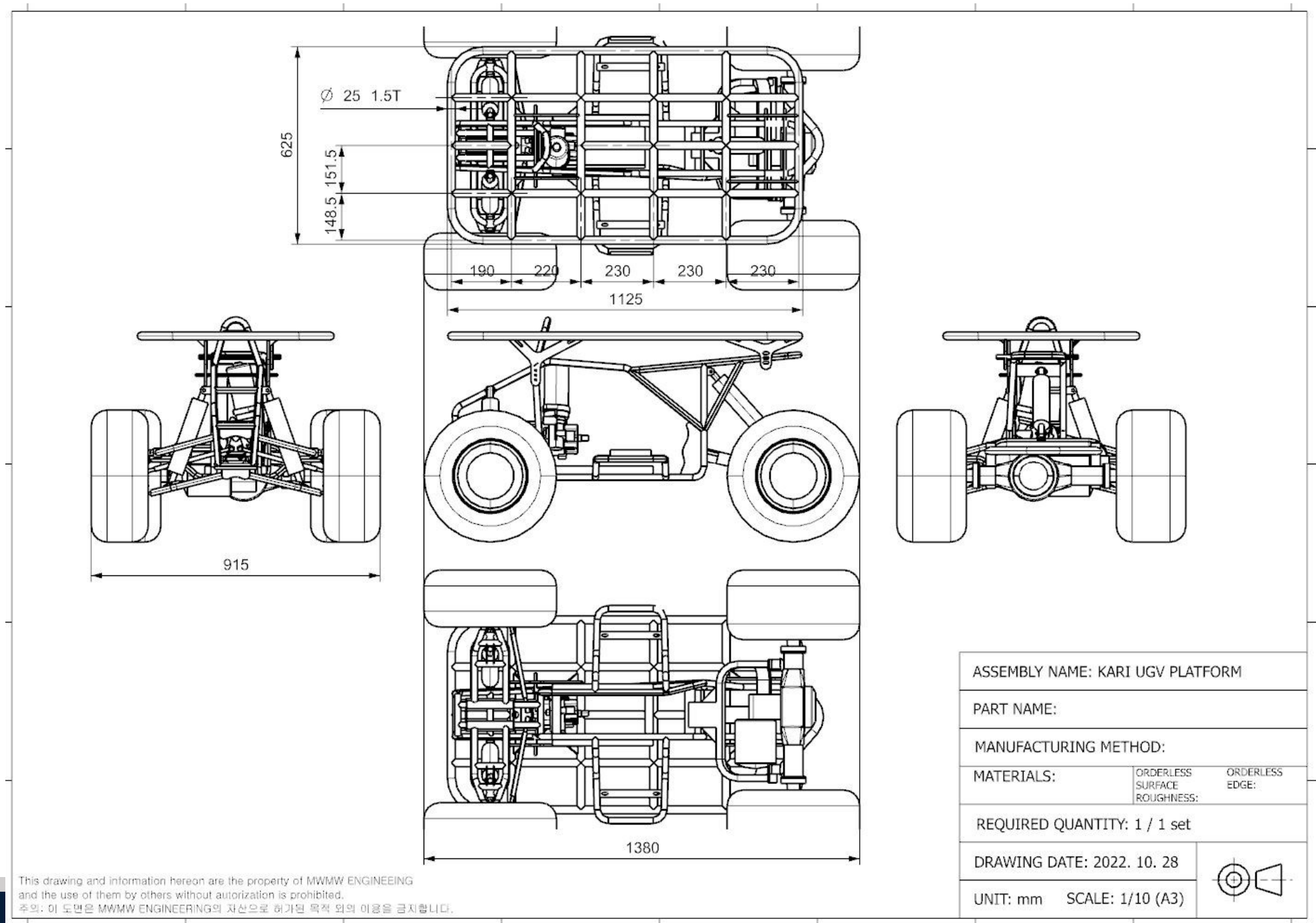
- 레저용 e-ATV(All Terrain Vehicle)을 자율주행 UGV(Unmanned Ground Vehicle)로 개조
- 서보 모터를 사용하여 전/후방 브레이크 레버 및 조향을 조정, 후륜에 모터를 이용하여 전/후진
- Control Interface Board, Pixhawk (Driving Controller), Jetson Nano (Mission Controller) 3개의 제어기로 구성되어 있음
- PX4(1.14.0) + ROS2(Humble) 기반 자율주행알고리즘 탑재
- 자율주행 기술 개발 및 실증을 위한 범용 플랫폼으로써 활용하고자 개발

1. 자율주행 UGV 플랫폼 소개

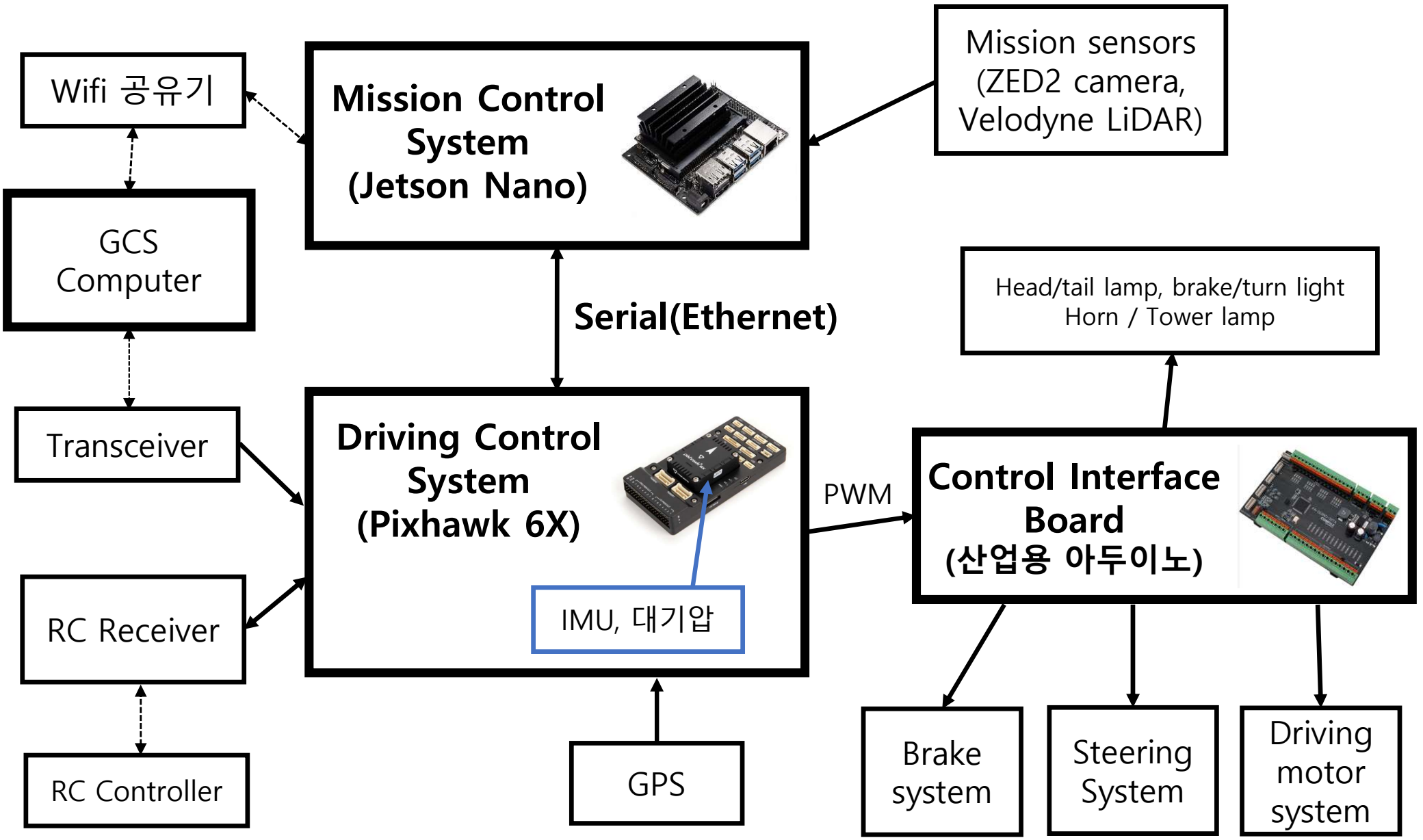




1. 자율주행 UGV 플랫폼 소개



2. 자율주행 플랫폼 구성 – 전장부



2. 자율주행 플랫폼 구성 - 제어기

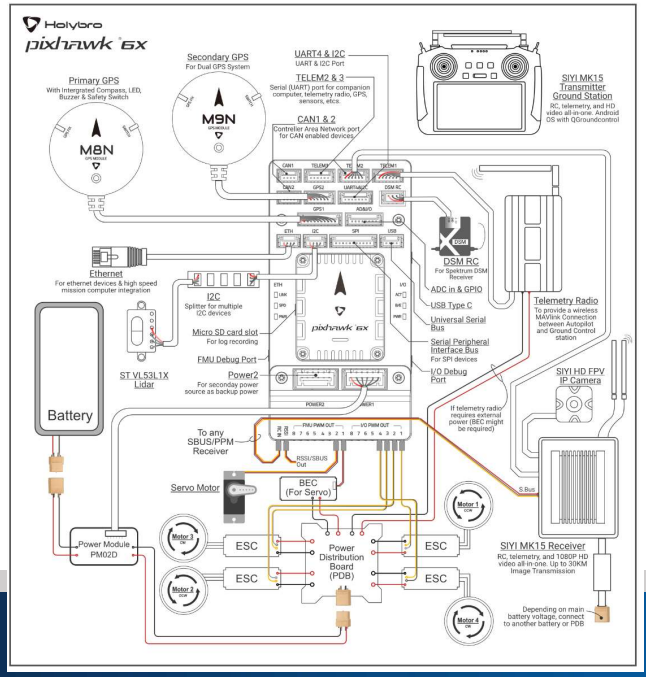
- Mission Control System, Driving Control System, Control Interface Board 가 제어임무를 분산하여 수행
- **Mission Control System**: 카메라, 라이다와 같은 정보를 받아서 주행 환경 인식(장애물, 주행가능영역, 표지판 등), 경로 생성 등 많은 계산량이 필요한 알고리즘의 연산 및 상위제어기로 써 동작
  - Jetson Nano + ROS2(Humble)

- **Driving Control System:** 안정적으로 제어 커맨드를 출력하고 UGV의 상태를 관리하는 중간 제어기 역할을 수행
  - GPS+Compass+IMU+Barometer 에 EKF를 적용하여 Localization 기능 제공
  - Pixhawk 6X + PX4(1.14.0)

- **Control Interface Board:** DCS에서 정해진 Steering, Motor, Brake 지령을 PWM 형태로 받아서 각 서브시스템에게 뿌려주는 인터페이스 보드 역할을 함
  - Light tower, 비상등, 방향지시등의 점멸 제어 + 비상정지 기능도 수행
  - 산업용 아두이노 보드

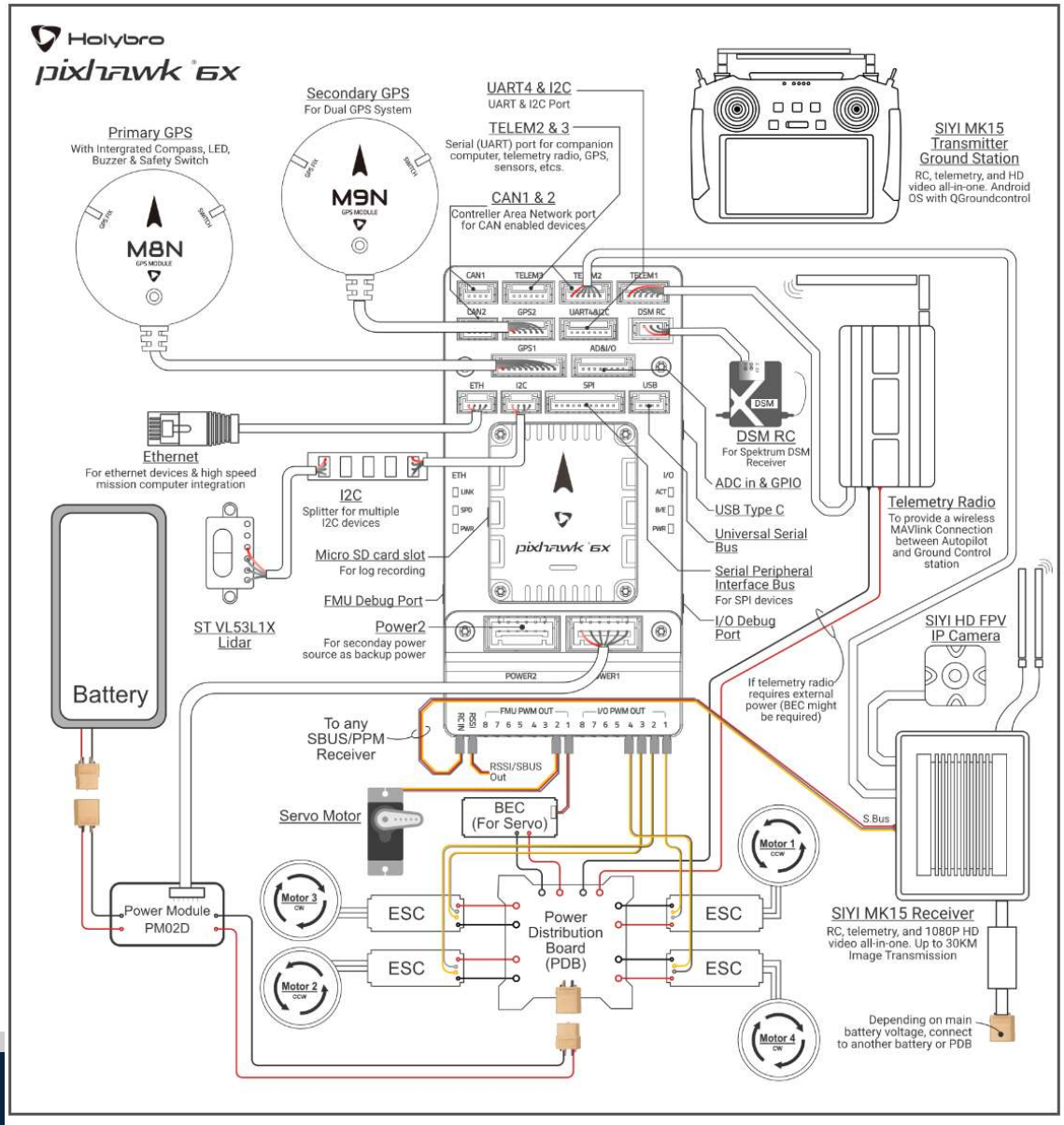
2. 자율주행 플랫폼 구성 - 제어기

- **PX4 (1.14.0) from Pixhawk 6X**
  - 신뢰성과 확장성을 가지고 자율주행 시스템을 제어할 수 있는 오픈소스 플랫폼
  - 주행제어를 위한 수많은 기능들 제공(다양한 주행 모드, 이상 신호 감지 등)
  - 다양한 주변기기와 결합하여 UGV의 자율주행에 필요한 각종 기본적인 기능들을 제공(RC, GPS, IMU, Localization, telemetry, PWM출력, 통신 등)
  - QGC(QGroundControl) 프로그램을 통해 UGV 모니터링 및 제어 가능
  - 내장된 IMU, Baro 와 외부의 GPS를 통해서 Localization 기능 제공
  - 1.14.0 버전 선정: DDS 통신을 기본으로 지원하는 최신 버전을 사용
    - Fast-RTPS → uXRCE-DDS 를 지원함으로써 ROS2와 호환성 개선



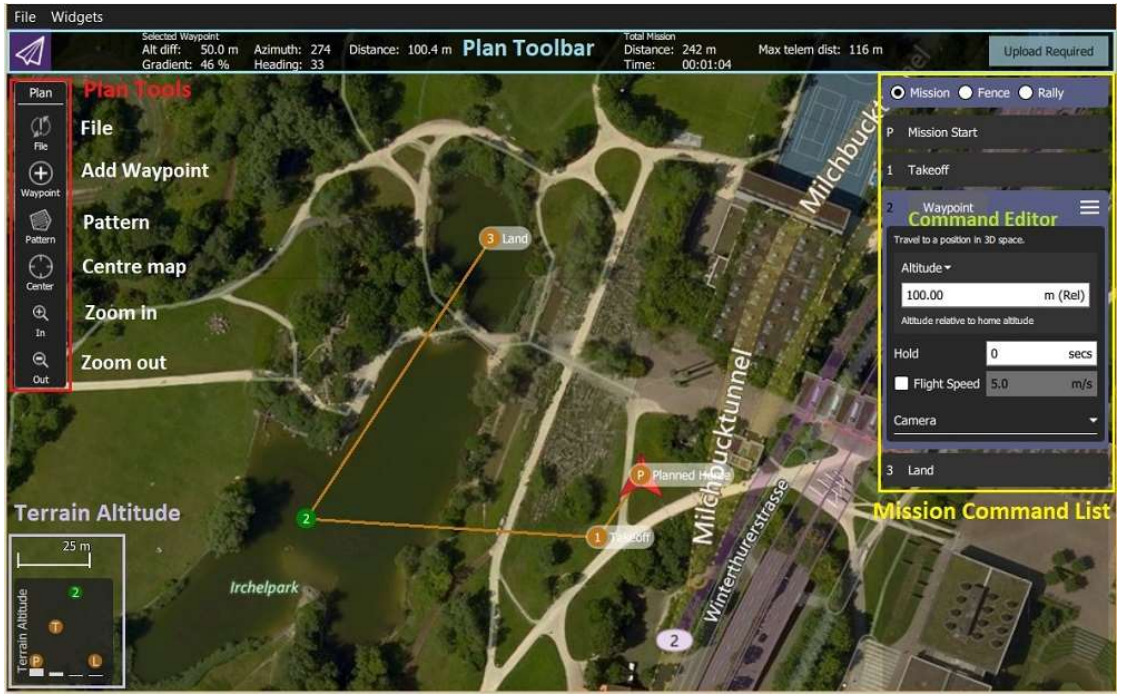
2. 자율주행 플랫폼 구성 - 제어기

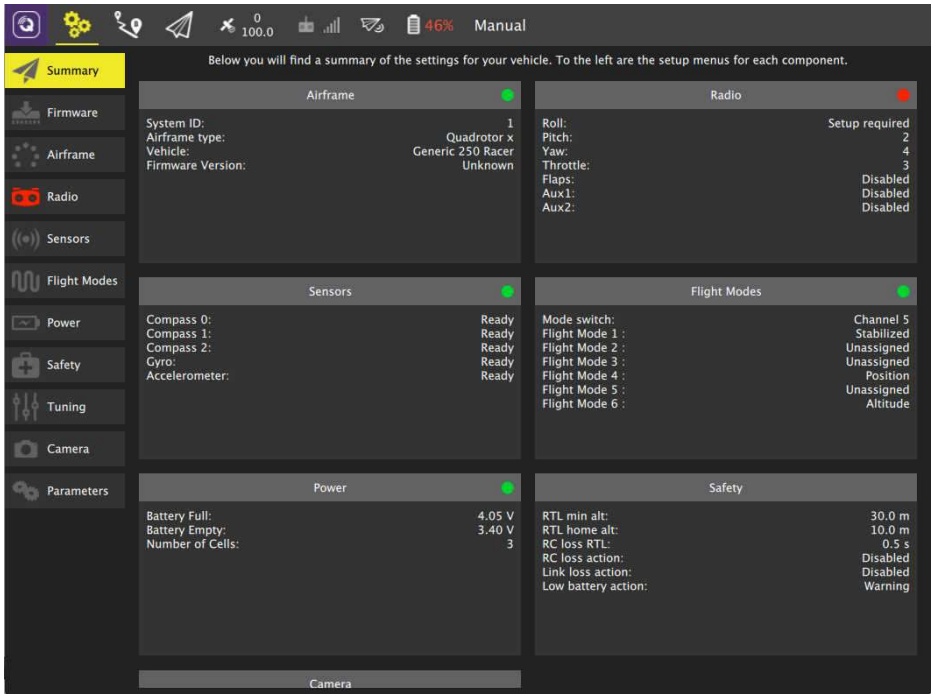
- **PX4 (1.14.0) from Pixhawk 6X**
  - **Pixhawk에 연결가능한 주변 기기 예**

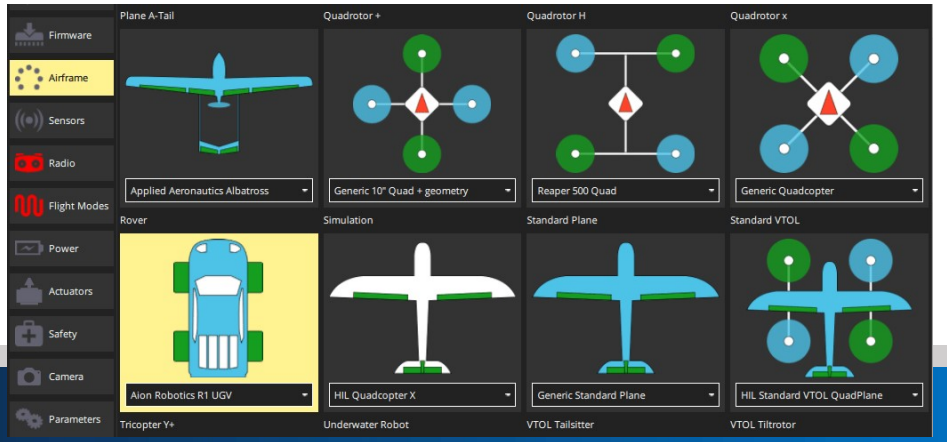


2. 자율주행 플랫폼 구성 - 제어기

- **PX4 (1.14.0) from Pixhawk 6X**
  - **QGroundControl 프로그램을 통한 UGV 모니터링, 관리 및 제어**
    - 다양한 종류의 기체 세팅 가능
    - 실시간 파라미터 튜닝 및 데이터 로깅







3. 자율주행 플랫폼 구성 - 제어기

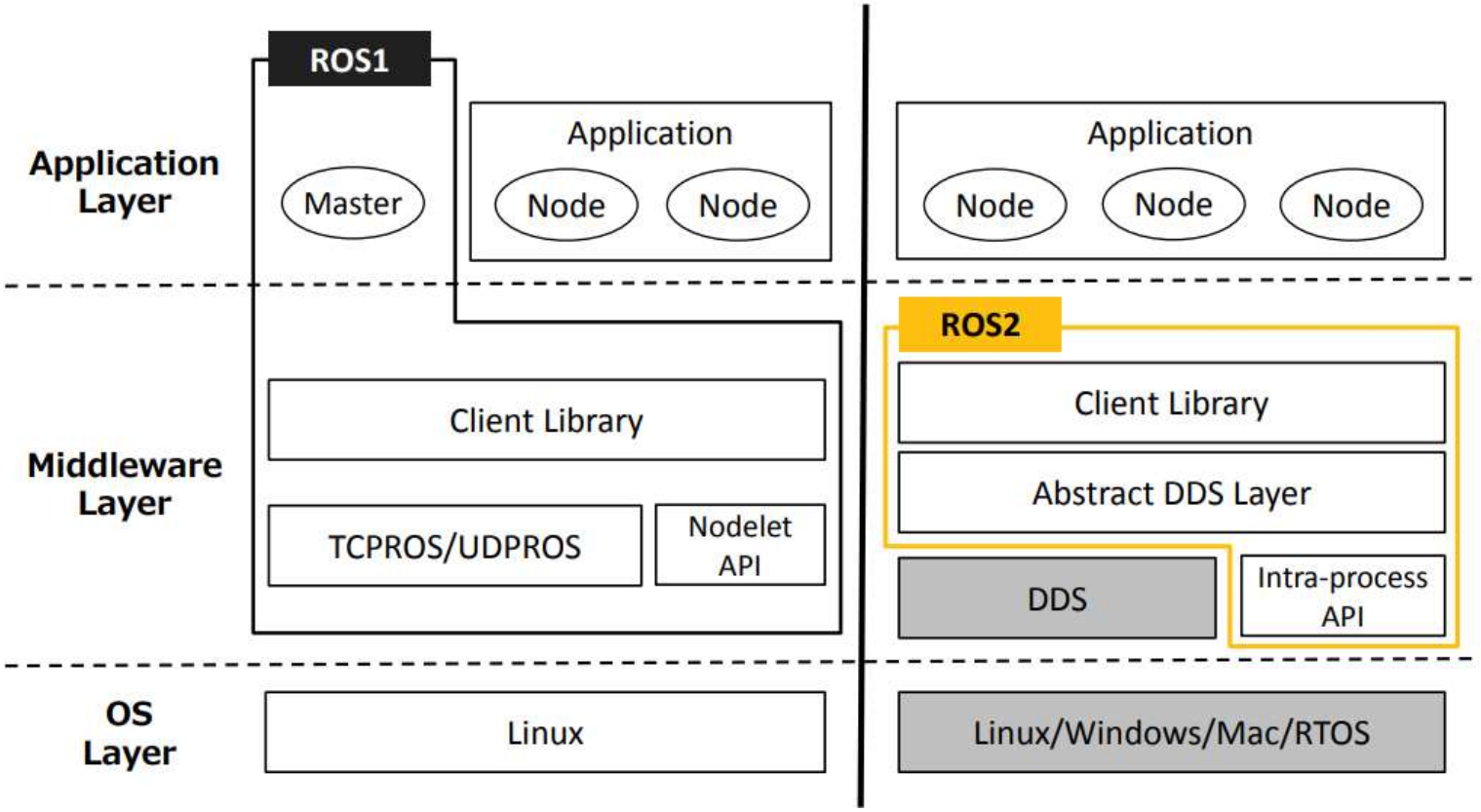
- **ROS2 (Humble) from Jetson Nano**
  - ROS(Robot Operating System)는 로봇 개발을 위한 라이브러리를 제공해주는 오픈소스 플랫폼
  - 자율주행 플랫폼의 다양한 제어 알고리즘을 개발할 수 있는 기반을 제공
  - ROS를 기반으로한 제어의 장점
    - 로봇관련 다양한 라이브러리 제공
    - 하드웨어에 종속되지 않기 때문에 프로그램 재사용성이 높음
    - 이기종 간에 통신 지원
    - 오픈소스 플랫폼
    - 활성화된 커뮤니티
    - 시각화(Rviz), 디버깅 툴
    - 동역학 시뮬레이션 환경(Gazebo)과의 연동

3. 자율주행 플랫폼 구성 - 제어기

- **ROS2 (Humble) from Jetson Nano** ROS 2
  - ROS1은 13번째 버전인 Noetic 을 마지막으로 릴리즈하여 2025년으로 공식적인 지원을 종료함
  - ROS2는 연구실에서의 연구/개발 용도로 만들어진 ROS1의 여러가지 한계점을 개선한 산업용을 목표로 하는 버전이라 할 수 있음
    1. 실시간성 보장
    2. 보안
       - ROS1은 ROS master 의 IP와 포트만 알면 해킹이 가능, ROS2는 DDS(Data Distribution Server)를 사용하여 이를 해결
    3. 다중도메인 지원(윈도우, 리눅스, Mac)
    4. 임베디드 컴퓨터에서도 사용가능
    5. 불안정한 네트워크 환경에서도 동작할 수 있음(Quality of Service)
    6. 이 이외에도 Lifecycle, 다양한 Build option, Multiple Node 같은 다양한 기능을 지원함
  - PX4와 호환되는 ROS2 버전은 Humble (on Ubuntu 22.04), Foxy (on Ubuntu20.04)
  - ROS1에 비해서 공유되는 오픈소스 패키지는 훨씬 적지만 앞으로 점점 더 커지고, 주도권을 가질 것으로 기대됨

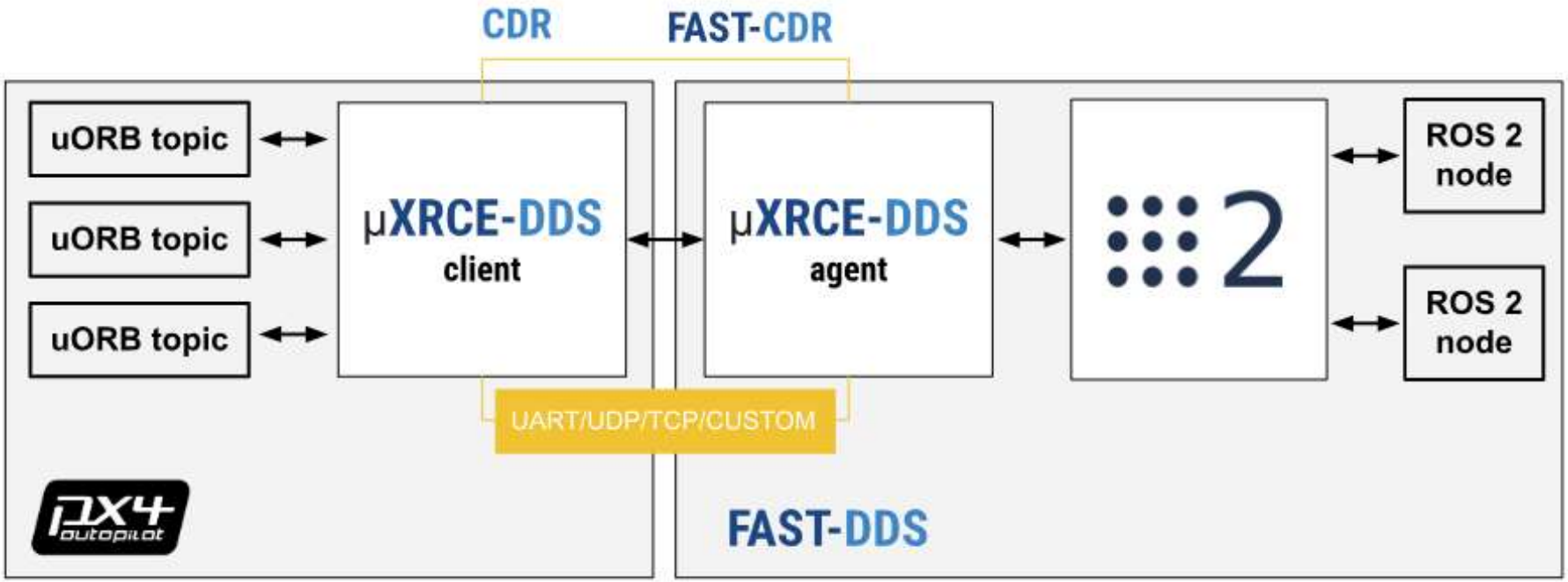
3. 자율주행 플랫폼 구성 - 제어기

- **ROS2 (Humble) from Jetson Nano**



3. 자율주행 플랫폼 구성 - 제어기

- **ROS2 (Humble) from Jetson Nano**
  - **uXRCE-DDS 기반 PX4-ROS2 데이터 통신**



3. 자율주행 플랫폼 구성 - 제어기

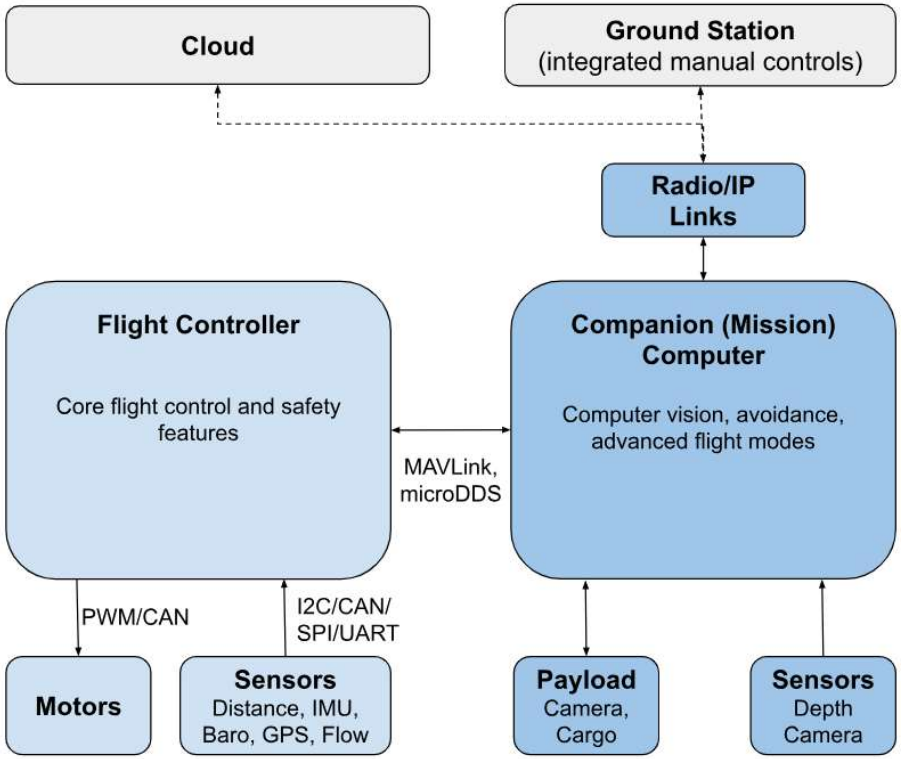
- **ROS2 (Humble) from Jetson Nano**
  - **Jetson Nano Vs. Intel NUC**
  - **Jetson Nano의 경우 자체 GPU를 가지고 있고 Nvidia에서 제공하는 JetPack SDK을 활용하여 Deep learning, Machine learning 기반 제어 알고리즘 개발에 특화**
    - **가격이 상대적으로 저렴하고 전력 소모도 매우 작음**
    - **카메라나 라이다의 데이터 처리 관련 SDK에서 CUDA를 필요로 하는 경우가 많음**
  - **Intel NUC는 소형 컴퓨터로써 Jetson Nano보다 더 강력한 CPU, RAM을 가지고 있기 때문에 GPU를 필요로 하지 않는 알고리즘에 한해서는 더 높은 성능을 제공할 수 있음**
    - **Jetson Nano는 Window를 지원하지 않기 때문에 Window를 기반으로 개발하는 경우에는 NUC를 사용**
    - **Jetson Nano는 Ubuntu 버전도 18.04 LTS(20.04로 버전 업 가능)로 고정되어 있기 때문에 상대적으로 호환성이 낮음**

3. 자율주행 플랫폼 구성 - 제어기

- **Jetson Nano에서 지원하는 Ubuntu 버전(18.04)와 ROS2 Humble이 적용가능한 Ubuntu 버전(Tier 1 22.04, Tier 3 20.04)이 맞지 않는 문제가 있음**
- **Jetson Nano의 버전을 18.04 → 20.04로 업그레이드 하여 사용할 필요가 있음**

3. 자율주행 플랫폼 구성 - GCS와 연결

- **Ground Control Station(GCS)가 Pixhawk와 Jetson Nano 중 어디와 통신을 주고 받는 것이 맞을까?**
- **GSC가 어디와 통신을 하는지에 따라 환경이 어떻게 달라질까?**



<PX4 공식 문서의 System with Companion computer 예시>

3. 자율주행 플랫폼 구성 - GCS와 연결

- **GCS가 Flight control system(Pixhawk)와 통신하는 경우**
  - GCS에서 Pixhawk에 줄 수 있는 명령이 한정되어 있음
  - Mission compute에 직접적인 명령을 줄 수 없기 때문에 미리 세팅된 기능 혹은 Pixhawk를 보조해주는 역할을 하게 됨
    - Mission compute가 할 수 있는 동작이 제한될 수 있음
  - 기본의 QGroundControl 인터페이스를 그대로 사용할 수 있음

- **GCS가 Mission control system(Jetson Nano)와 통신하는 경우**
  - Mission computer에서 환경인식부터 최종 제어입력 생성까지 모두 수행하는 PX4-offboard control 모드에서 동작
  - 다양하고 유연한 상위제어기 구성 가능
  - QGroundControl 와 ROS2 기반 플랫폼 사이 연결이 필요

3. 자율주행 플랫폼 구성 - GCS와 연결

- PX4 + Companion computer로 자율주행을 할 때 역할 분담?

- 기본적으로 PX4로 주행을 하고 Companion Computer에서 경로상의 이상을 감지할 때 Offboard mode로 잠시 돌리고 장애물 회피를 수행하고 다시 주도권을 PX4로 넘기기?

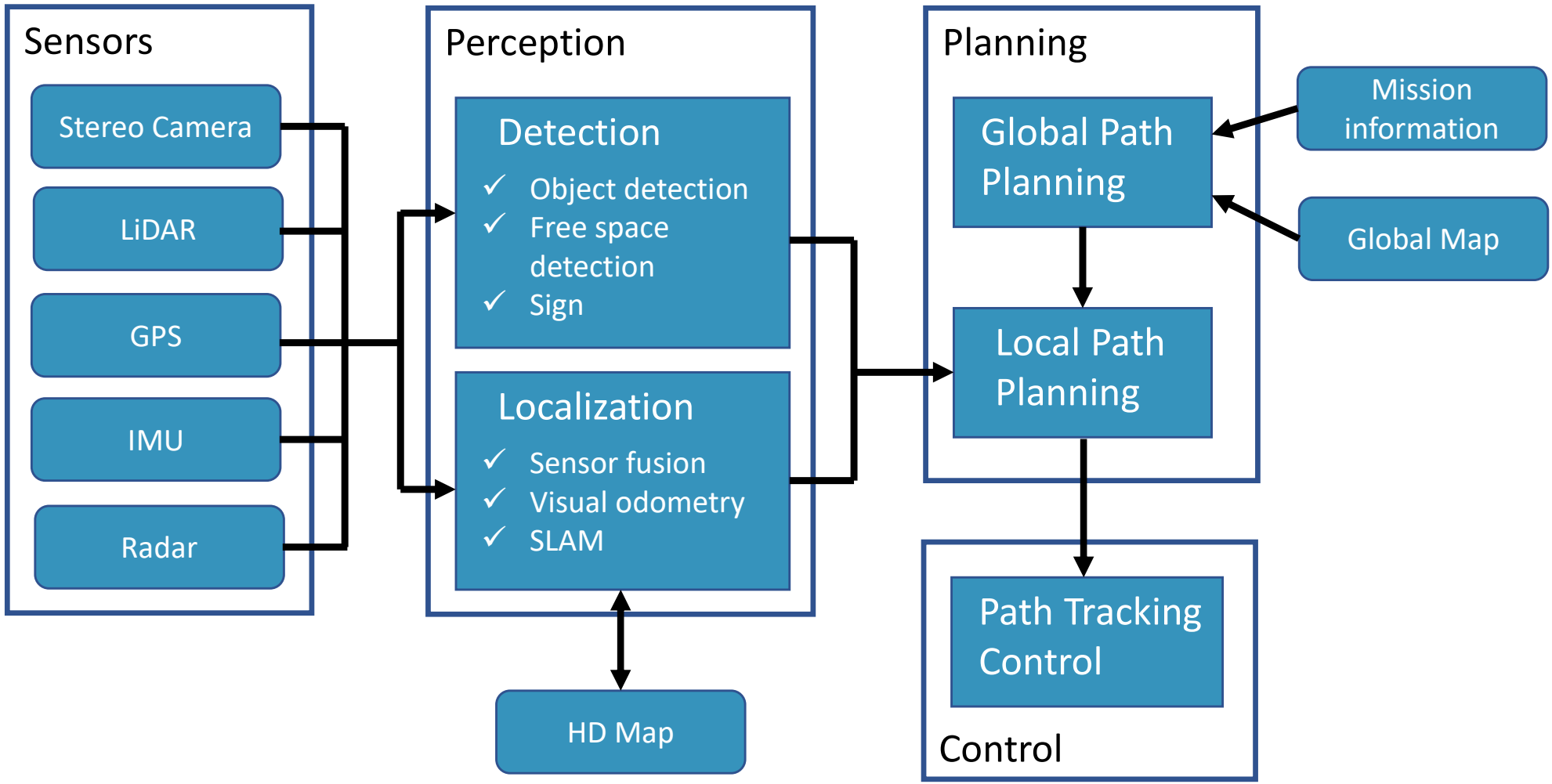
- Companion Computer에서 모든 제어(장애물 감지, 로컬 경로 생성, 경로 추종 커맨드 생성)을 다하여 PX4로 넘겨주면 PX4는 안정적으로 동작하는 모션 컨트롤러(+Localization)로써 동작하는가?

3. 자율주행 플랫폼 구성 - GCS와 연결

- 현재는 GCS가 DCS(Pixhawk)와 MCS(Jetson Nano)와 각각 telemetry와 Wifi를 통해서 동시에 병렬로 연결하도록 구성되어 있음
- GCS 와 Pixhawk는 mavlink를 기반으로 QGroundControl 인터페이스를 통해 시스템 모니터링 및 PX4 파라미터 실시간 수정 등의 작업을 수행
- GCS와 Jetson Nano는 Wifi를 통해 원격접속하여 terminal 환경에서 ROS2 기반 제어기에 대한 전반적인 명령을 내리고 모니터링을 수행

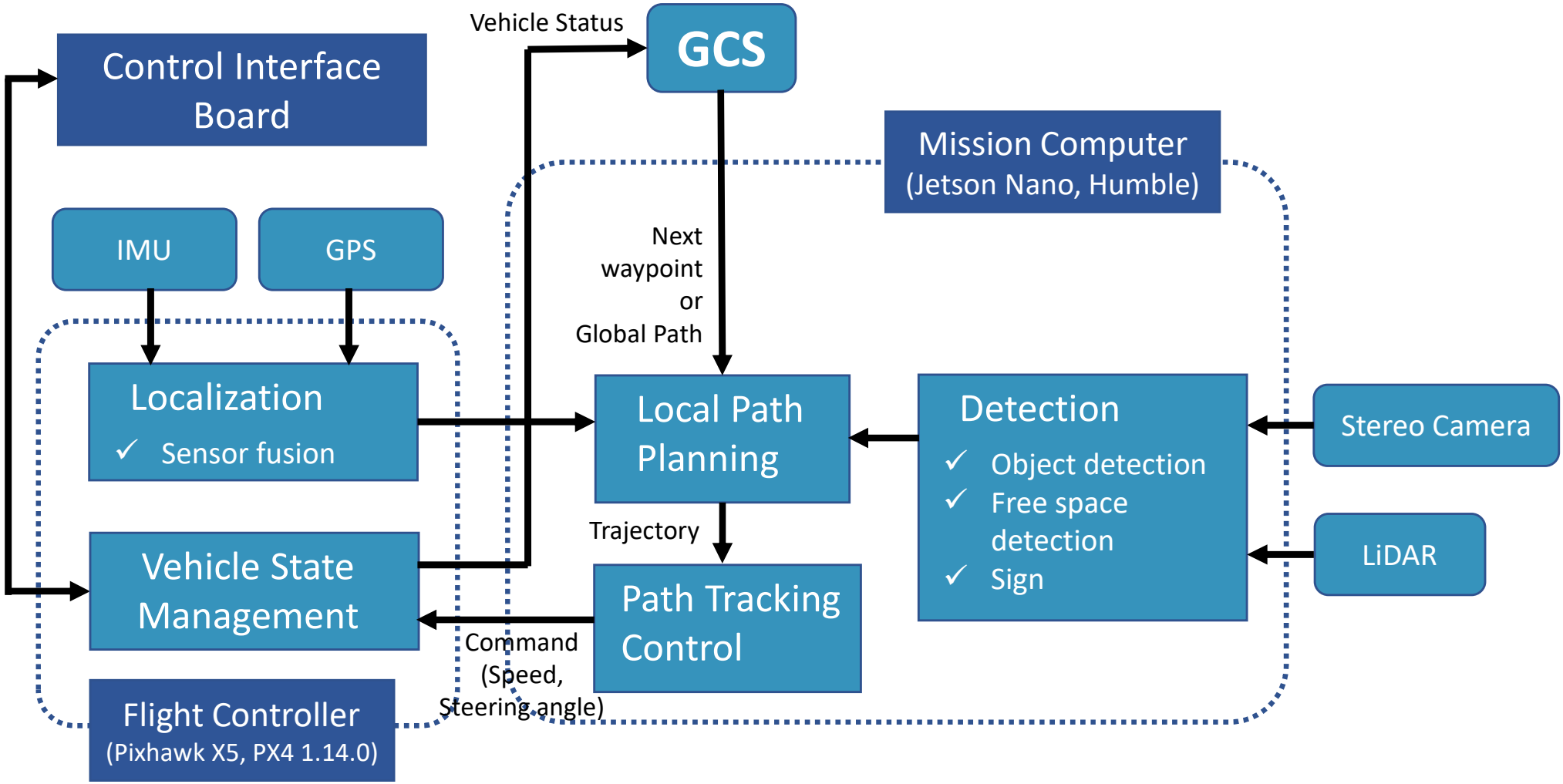
4. 자율주행 알고리즘 SW 구성

- 일반적인 ROS2 기반 자율주행 SW 알고리즘 구성



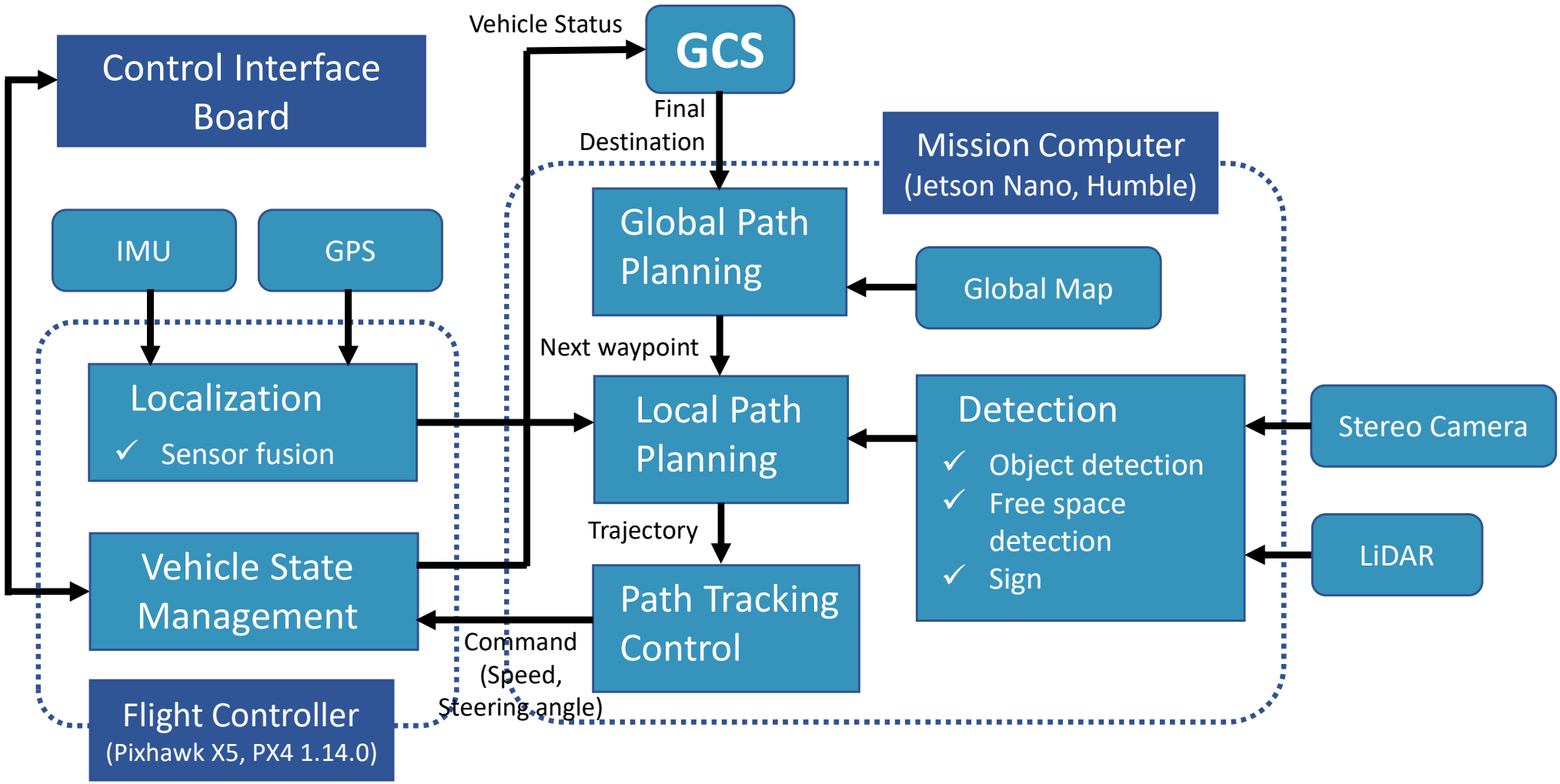
4. 자율주행 알고리즘 SW 구성

- **ROS2+PX4 기반 자율주행 SW 알고리즘 구성 - 1**



4. 자율주행 알고리즘 SW 구성

- **ROS2+PX4 기반 자율주행 SW 알고리즘 구성 - 2**



4. 자율주행 알고리즘 SW 구성

- **PX4의 Offboard Control mode에 따라 여러가지 형태의 상위제어기 설계 가능**

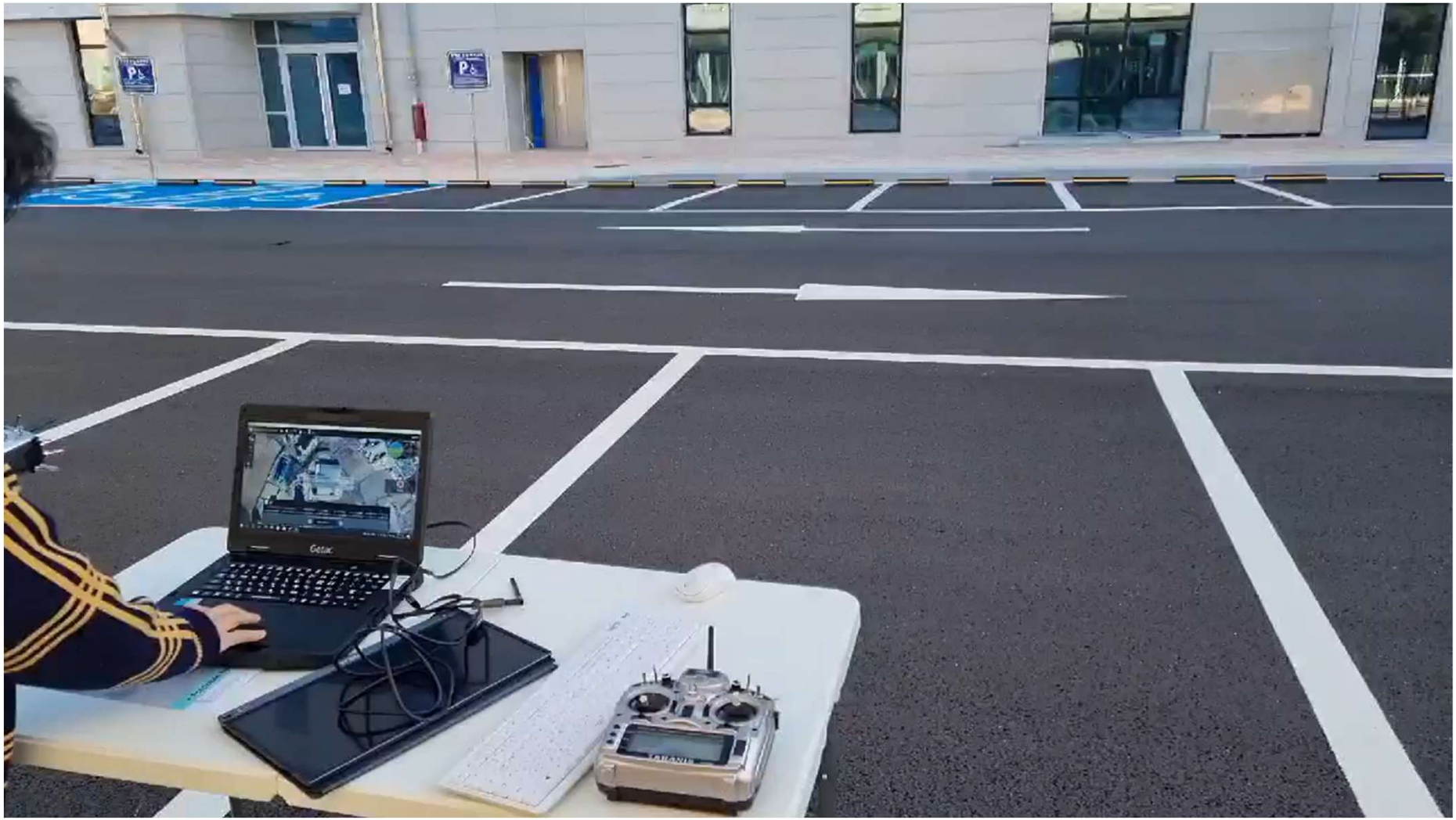
desired control quantity	position field	velocity field	acceleration field	attitude field	body_rate field	actuator field	required estimate	required message
position (NED)	✓	-	-	-	-	-	position	`TrajectorySetpoint`
velocity (NED)	✗	✓	-	-	-	-	velocity	`TrajectorySetpoint`
acceleration (NED)	✗	✗	✓	-	-	-	velocity	`TrajectorySetpoint`
attitude (FRD)	✗	✗	✗	✓	-	-	none	VehicleAttitudeSetpoint
body_rate (FRD)	✗	✗	✗	✗	✓	-	none	VehicleRatesSetpoint
thrust and torque (FRD)	✗	✗	✗	✗	✗	✓	none	VehicleThrustSetpoint and VehicleTorqueSetpoint

6. ROS2+PX4 아키텍쳐의 장점

1. GPU를 사용한 고연산을 필요로 하는 HW와 안정적으로 주행체를 제어하는 제어기의 역할을 나누는 것은 자율주행 로봇/차량에서 일반적인 접근법임
2. PX4에서 제공하는 주행제어를 위한 다양한 기본 기능들 활용 가능
   - 기본 주행제어, Airframe, 주행모드, 이상신호감지 등등
   - GPS+IMU+Kalman filter 의 Localization 모듈을 Pixhawk에서 제공받음
3. PX4에서 안정적으로 최종제어 입력값(속도, 조향)을 생성
   - ROS2에서 실시간성이 보장되도록 할 수 있지만 상위제어기에서 오류가 발생할 경우에도 안정적으로 동작이 가능하고, 신호 유실에 대해 비상 정지 상태 혹은 PX4 단독으로 주행제어를 할 수 있음
4. ROS2 환경을 기반으로 원하는 상위제어기를 자유롭게 설계 가능
5. QGroundControl을 이용한 관제 시스템 제공
6. 주행 플랫폼이 바뀌었을 때 ROS2는 그대로 두고 PX4만 수정하여 사용가능
7. 적용하고자 하는 상위 알고리즘이 바뀌었을 때 PX4는 그대로 두고 ROS2만 수정하여 사용가능

7. UGV 주행 실험 영상

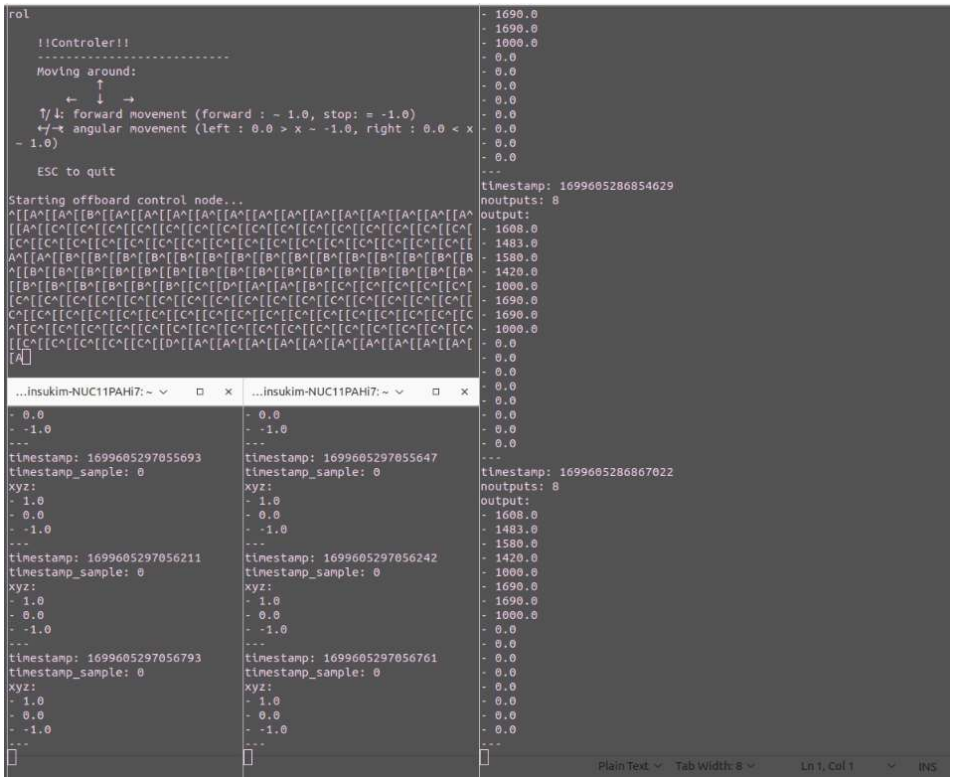
- **Mission mode 기반 동작 시험**



7. UGV 주행 실험 영상

- **Offboard mode(Actuator input) + Keyboard 입력 주행 실험**

Keyboard input



Actuator input (publish to PX4)

PWM output(subscribe from PX4)

8. 향후 개발 및 활용 계획

- UGV 자율주행 알고리즘 개발/검증용 범용 플랫폼으로써 개발해 나가고자 함

- 상위 제어기 개발
  - 다양한 시나리오에서 기술개발에 필요로 하는 기본 기능들을 구현
  - Ex: GPS 웨이포인트 기반 자율주행, 거리정보 기반 장애물 회피, Lane keeping 등

- 적용 방법?
  - 별도의 상위제어기 HW를 Jetson Nano에 연결하여 ROS끼리 통신을 통해 필요한 데이터를 주고 받기
  - Or 기존의 Jetson Nano에 해당 알고리즘 패키지를 적용
  - Docker?
  - ROS1에서 개발된 알고리즘의 경우 → ROS1 Bridge?

Thank You