Introduction to MAVROS

and Opensource UAVs

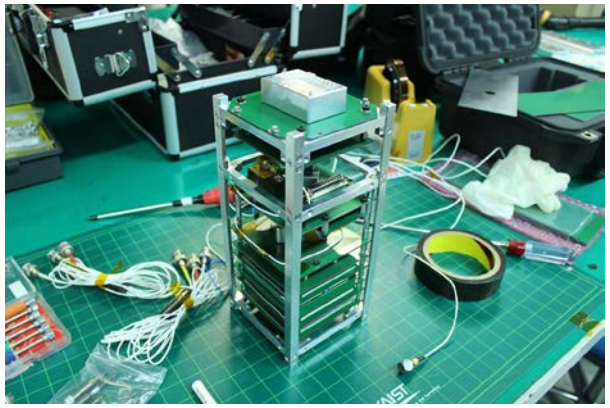
Jaeyoung Lim

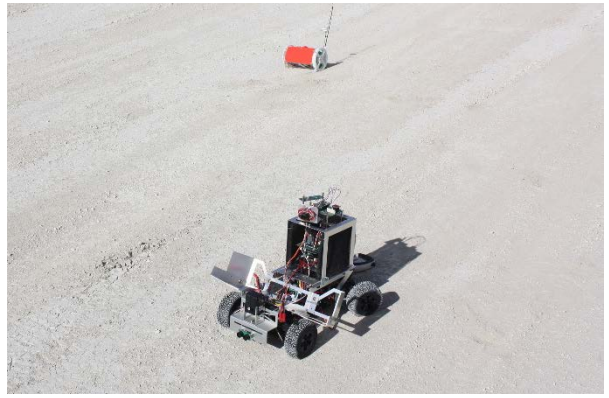
Program

수행일자	주요내용 및 세부목표	회차
2016. 6. 21	Introduction to MAVROS	1
2016. 6. 28	Gazebo를 사용한 무인기 SITL	2
2016. 7. 5	무인기 설계 방법론	3
2016. 7. 12	Pixhawk Controller를 사용한 쿼드콥터 제작	4
2016. 7. 19	MAVROS를 사용한 오토 호버링 / 설계 임무시행	5

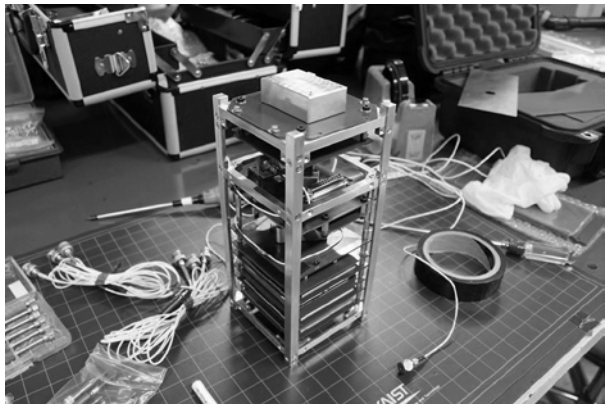
Context

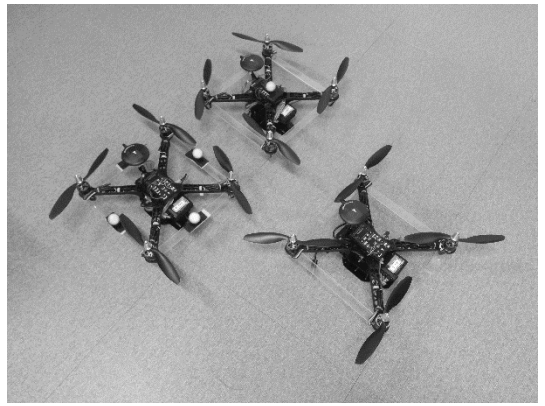
- PX4
  - Hardware
  - PX4 Flight Stack
  - Companion Computers
  - MAVLink
- MAVROS
  - ROS
  - MAVROS
  - SITL / HITL

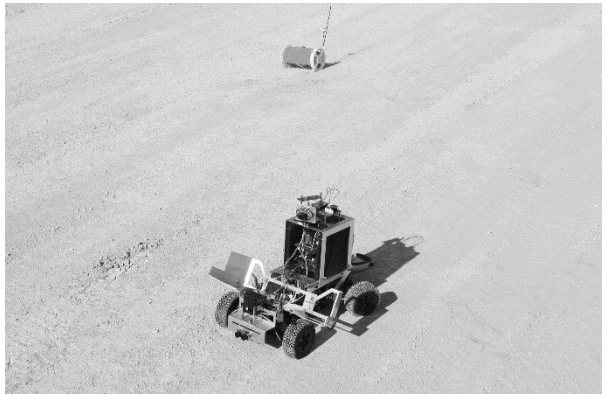



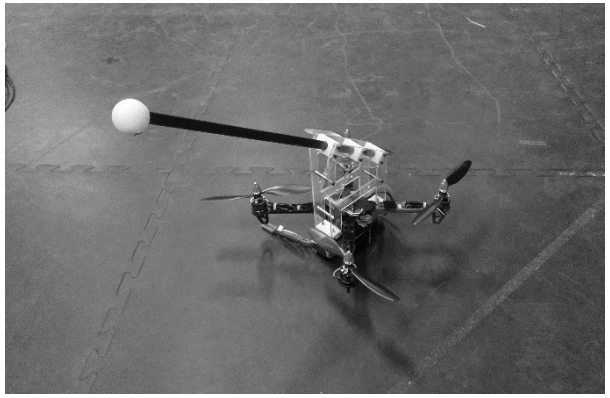



SEOUL NATIONAL UNIVERSITY
SNUSAT
SMALL SATELLITE TEAM







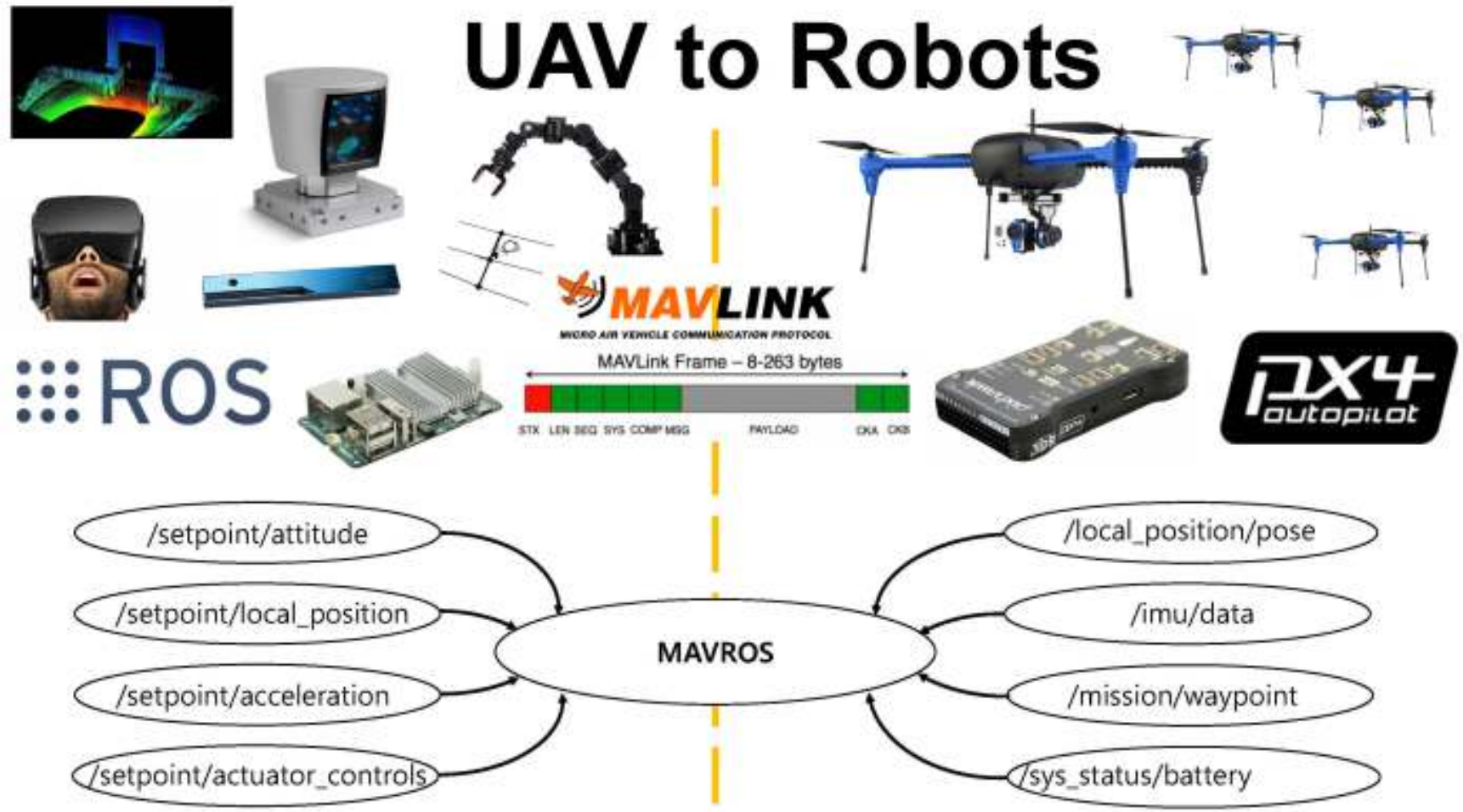



404Warehouse

Introduction

Introduction

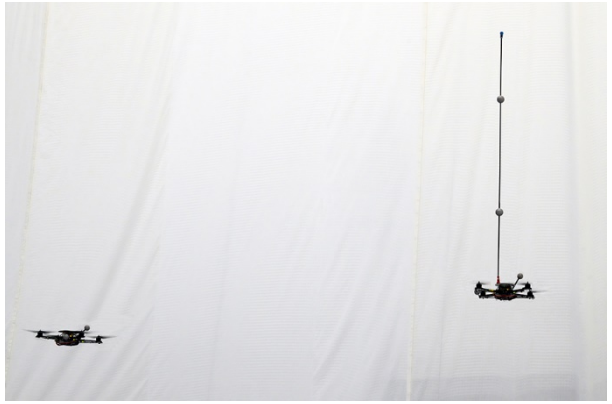
Introduction to MAVROS



ARDUPILOT (Not for this Seminar)

- Implemented as an application on PX4
- More Hobbyist oriented
- Arduino Compatible

PX4 Autopilot



ETH Zürich

Autonomous Systems Lab

3DR · QUALCOMM · ETH zürich · intel

CVG Computer Vision and Geometry Lab · Autonomous Systems Lab · IDSC · ROBOTICS & PERCEPTION GROUP

uaventure · AUAV · AIRDOG · Dronecode

LILY · wingtra · Prioria · Fotokite

fleye · SONY · ZMP

Applied Aeronautics · Purdue University · UAV Solutions

PX4 Autopilot

ETH Zürich

Autonomous Systems Lab

3DR · QUALCOMM · ETH zürich · intel

CVG Computer Vision and Geometry Lab · Autonomous Systems Lab · IDSC · ROBOTICS & PERCEPTION GROUP

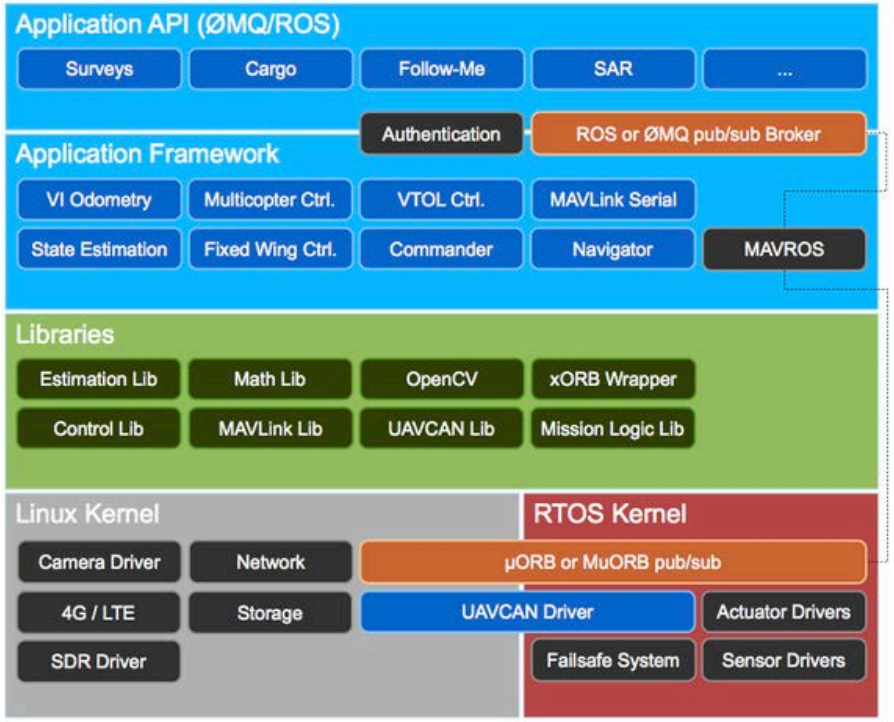
uaventure · AUAV · AIRDOG · Dronecode

LILY · wingtra · Prioria · Fotokite

fleye · SONY · ZMP

APPLIED AERONAUTICS · PURDUE UNIVERSITY · UAV SOLUTIONS

PX4 Autopilot

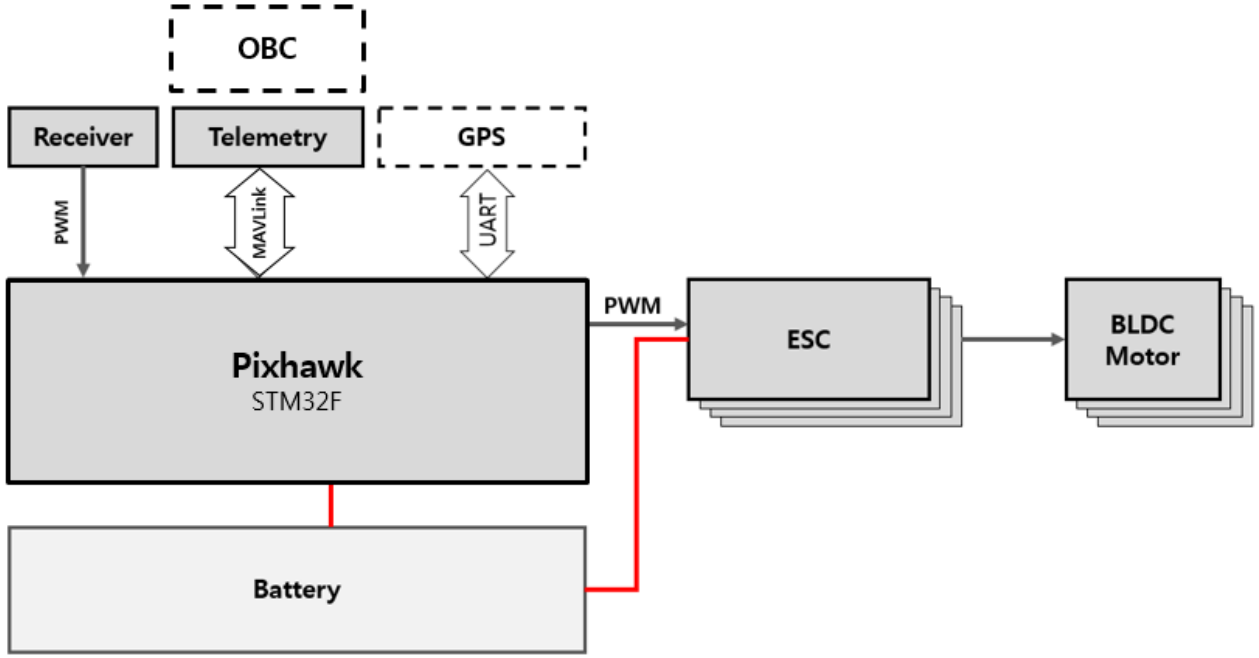


- Apps API: This interface is intended for app developers, e.g. using ROS or Drone API. This API is intended
- Applications Framework: This is the set of core default applications (nodes) which operate the core flight controls
- Libraries: This layer contains all system libraries and functionality for the core vehicle operation
- Operating System: The last layer provides hardware drivers, networking, UAVCAN and failsafe systems

Compatible with ROS

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

Opensource based Quadrotor



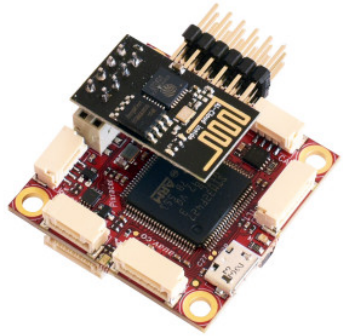
Major Performance Metrics

- Thrust to Weight Ratio
- Flight Time
- Payload

Flight Controller

Specs:
Size: **81x44x15mm**
Weight: **33.1g**
Microprocessor:
32-bit STM32F427 Cortex M4 core with FPU
168 MHz/256 KB RAM/2 MB Flash
32 bit STM32F103 failsafe co-processor
Sensors:
ST Micro L3GD20 3-axis 16-bit gyroscope
ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer
Invensense MPU 6000 3-axis accelerometer/gyroscope
MEAS MS5611 barometer

PX4 FMU



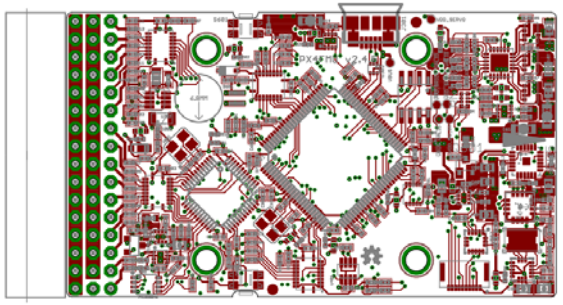
Pixracer

HKPilot32

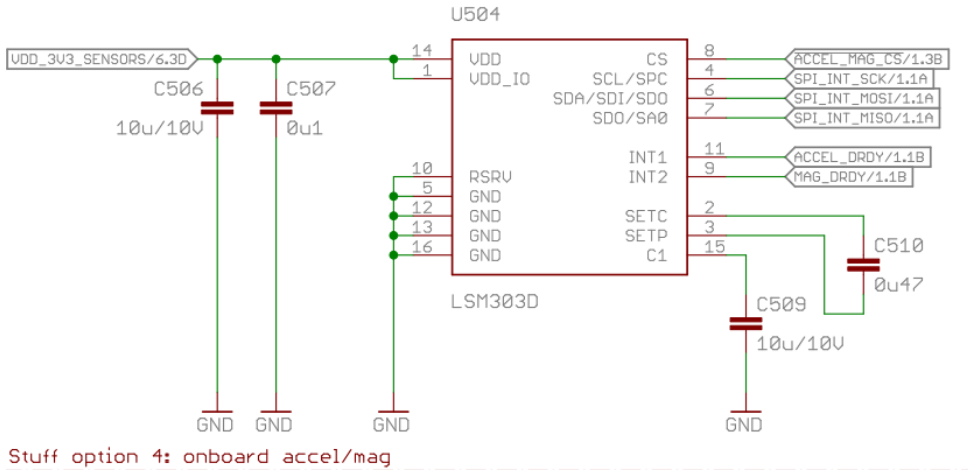
Snapdragon F.C.

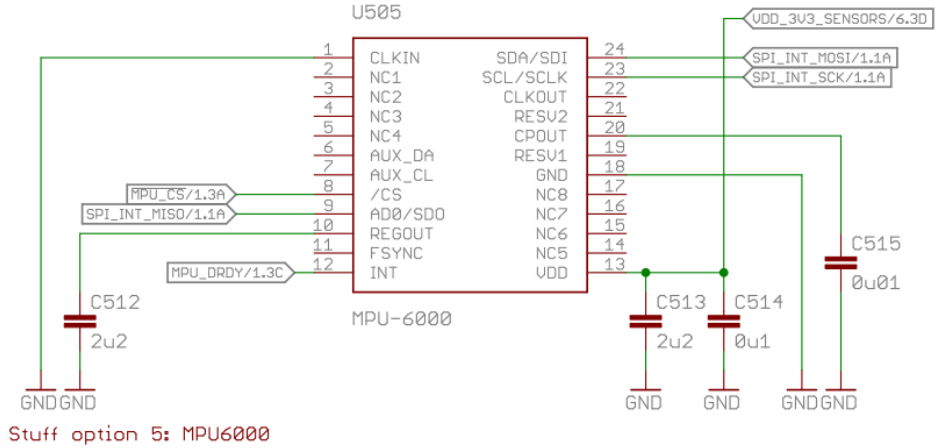
Raspberry Pi

Flight Controller



Sensors
ST Micro L3GD20 3-axis 16-bit gyroscope
ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer
Invensense MPU 6000 3-axis accelerometer/gyroscope
MEAS MS5611 barometer



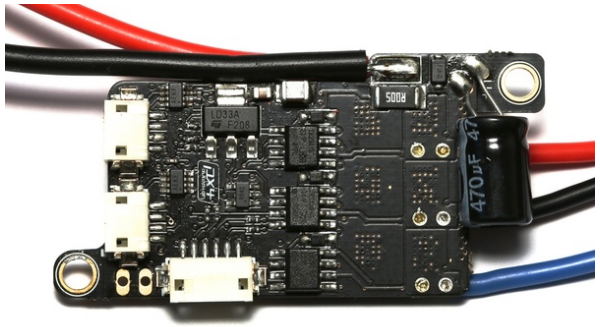


~v1.3.2 Only MPU 6000 is used
~v1.3.3 LSM303 is used with MPU 6000 but with no redundancy (no fault detection)

https://github.com/PX4/Hardware/blob/master/FMUv2/PX4FMUv2.4.5.pdf

Speed Controllers

PX4 ESC



Interfaces

- UAVCAN
- Command line interface (CLI) over UART (see the connector pinout below)
- PWM input

Output

- RPM Control
- Field Oriented Control (Direct Torque Control)

ESC

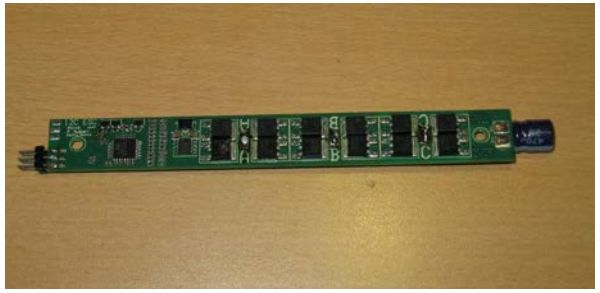
Interfaces

- PWM input(~400Hz)

Output

- RPM Control

I2C ESC



Interfaces

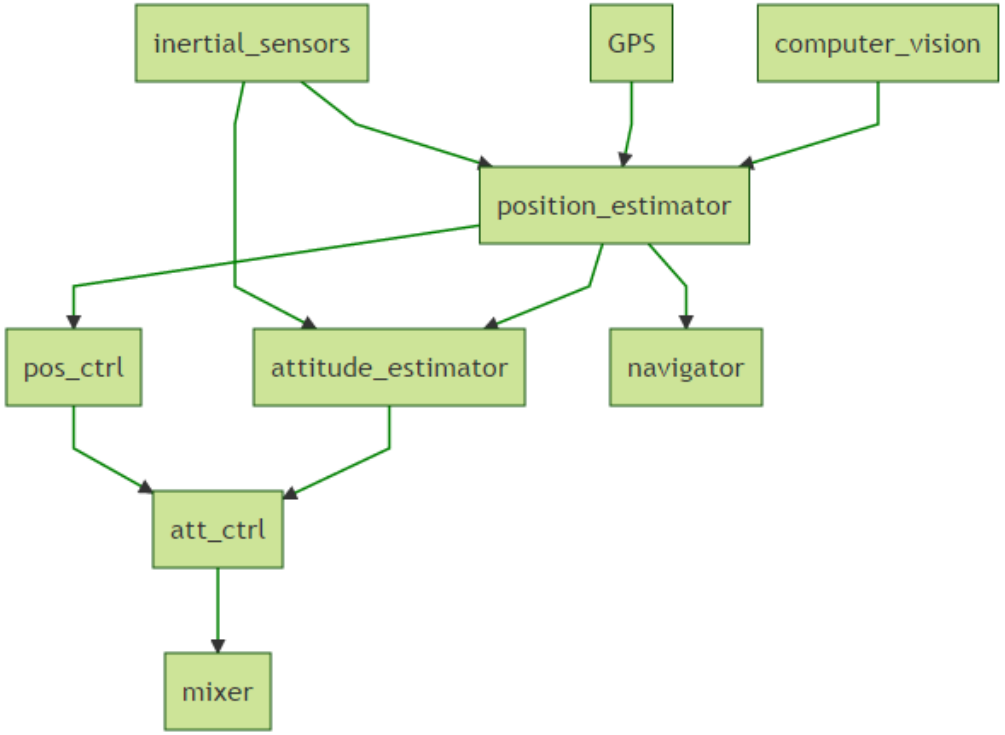
- I2C input

Output

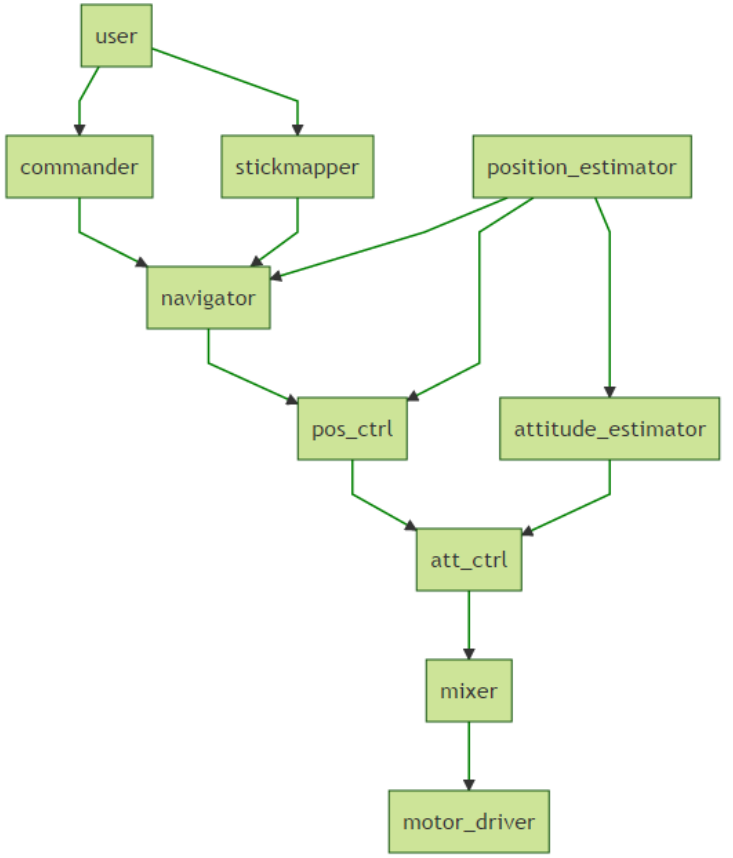
- RPM Control

PX4 Flight Stack

Highlevel Software Architecture



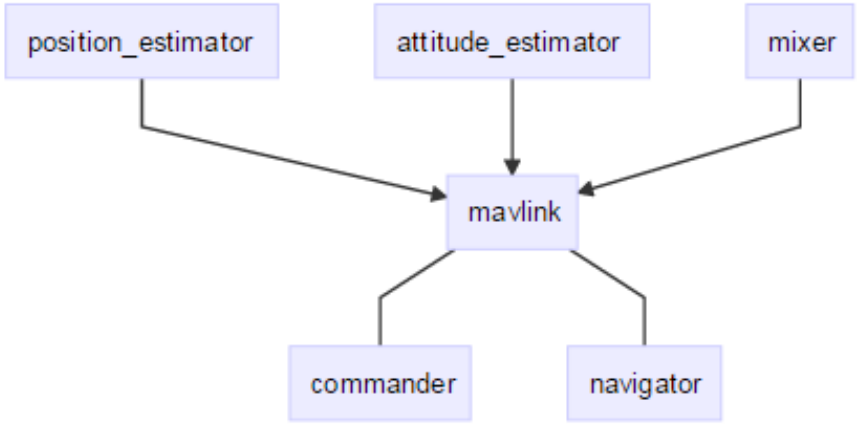
Estimation and Control Architecture



Architecture Overview

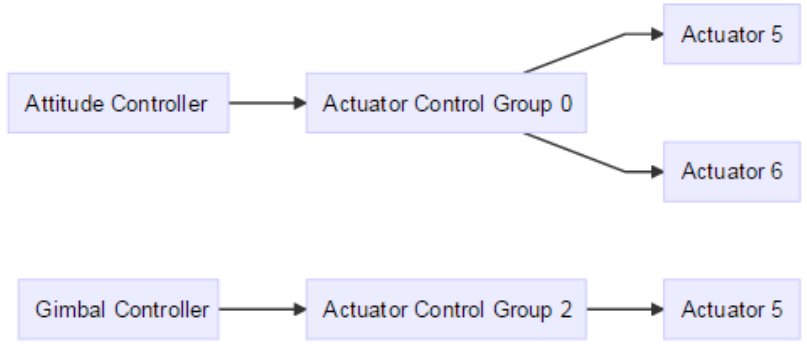
PX4 Flight Stack

Communication Architecture with the GCS



PX4 Flight Stack

Control Groups



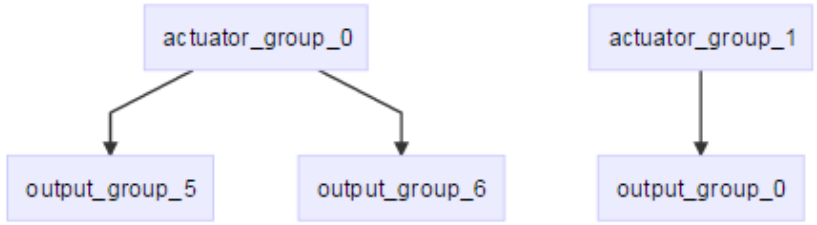
A particular controller sends a particular normalized force or torque demand (scaled from -1..+1) to the mixer, which then sets individual actuators accordingly. The output driver (e.g. UART, UAVCAN or PWM) then scales it to the actuators native units, e.g. a PWM value of 1300.

Control Group #0 (Flight Control)
- **0: roll (-1..1)**
- **1: pitch (-1..1)**
- **2: yaw (-1..1)**
- **3: throttle (0..1 normal range, -1..1**
- **4: flaps (-1..1)**
- **5: spoilers (-1..1)**
- **6: airbrakes (-1..1)**
- **7: landing gear (-1..1)**

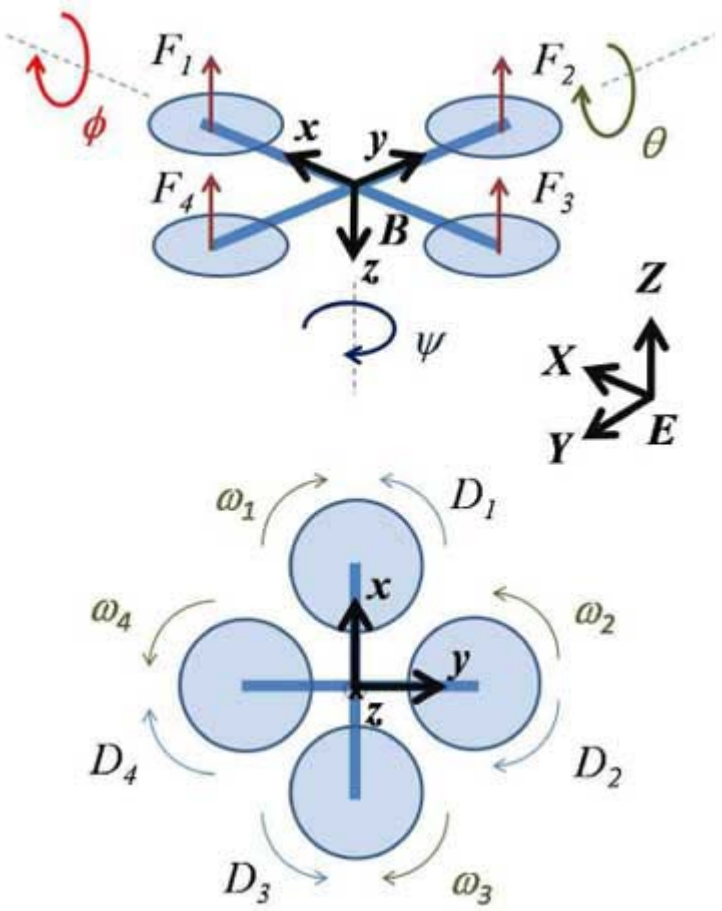
Control Group #1 (Flight Control VTOL/Alternate)
- **0: roll ALT (-1..1)**
- **1: pitch ALT (-1..1)**
- **2: yaw ALT (-1..1)**
- **3: throttle ALT (0..1 normal range, -1..1 for variable pitch / thrust reversers)**
- **4: reserved / aux0**
- **5: reserved / aux1**
- **6: reserved / aux2**
- **7: reserved / aux**

PX4 Flight Stack

Mixer

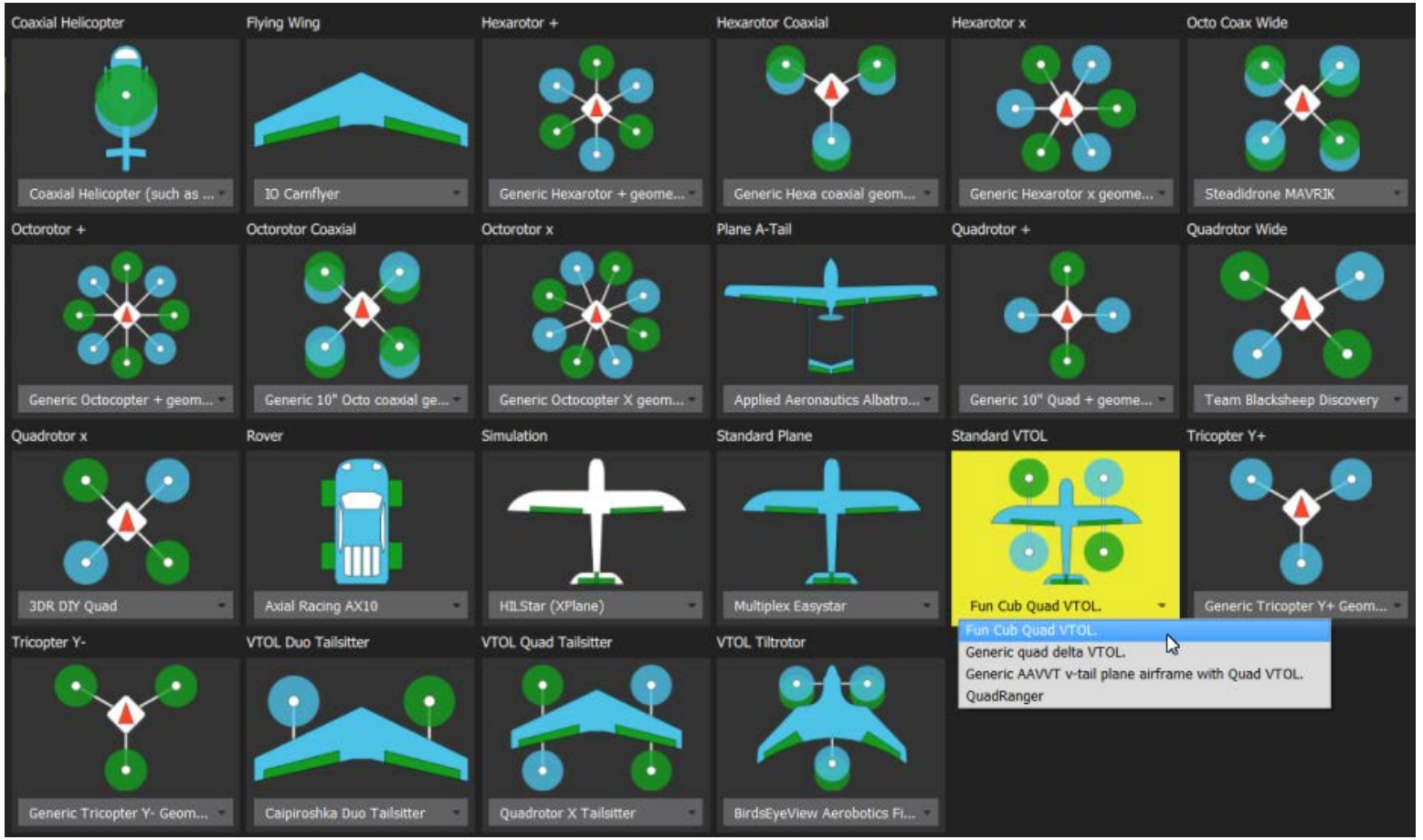



```
M: <control count>
O: <-ve scale> <+ve scale> <offset> <lower limit> <upper limit>
```



PX4 Flight Stack

Supported Airframes

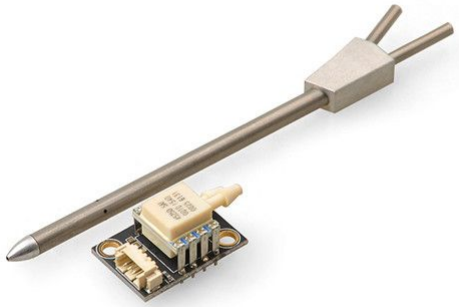


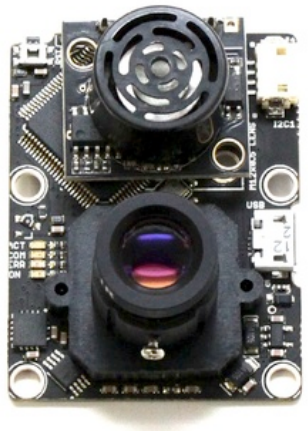
PX4 Flight Stack

VTOL

PX4 Flight Stack

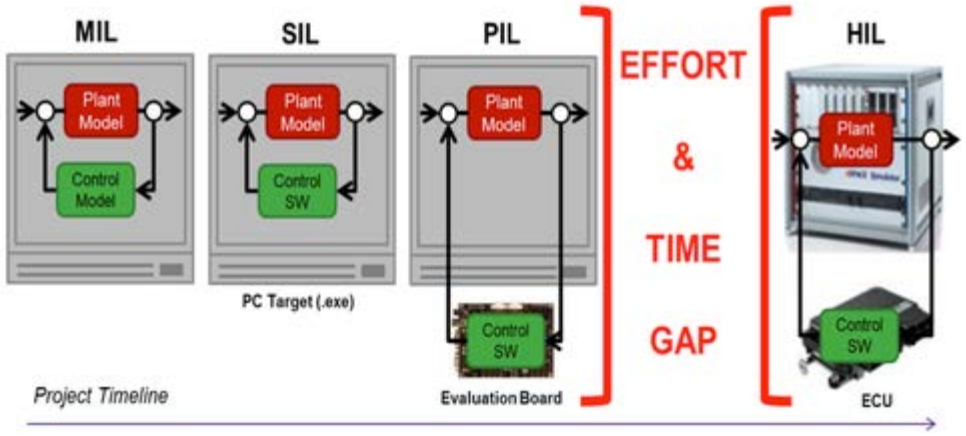
Supported Sensors

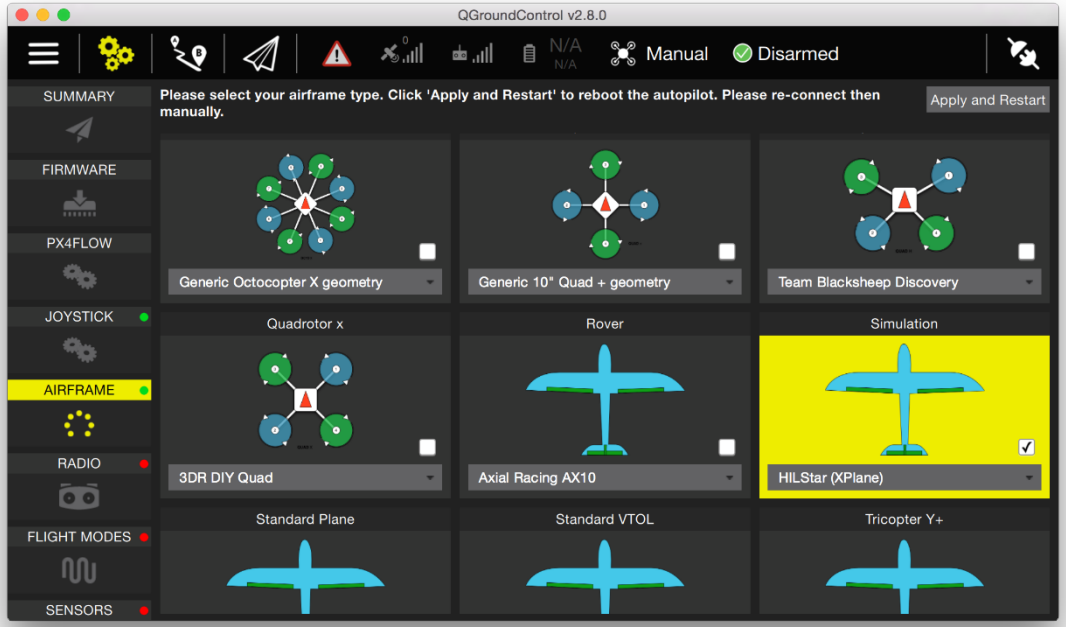




PX4 Flight Stack

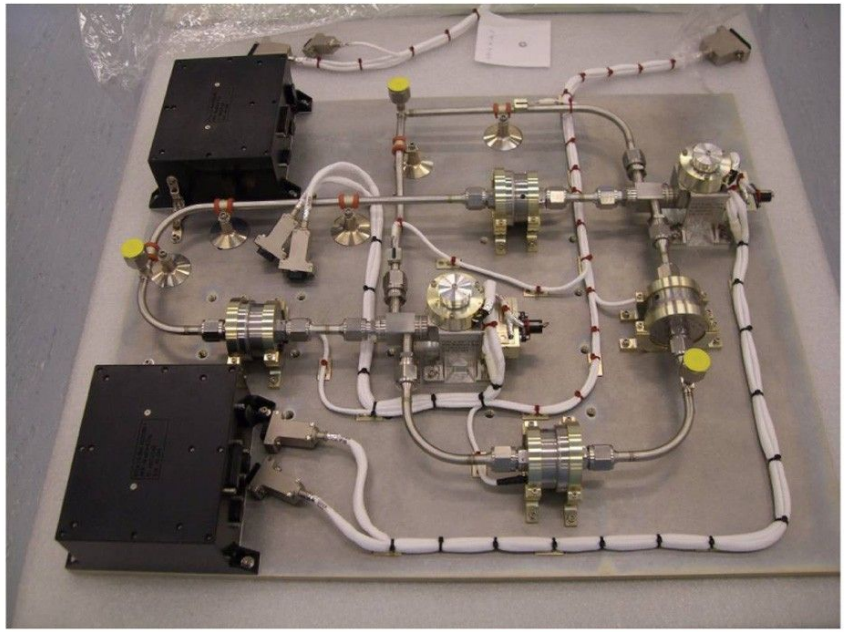
Hardware In the Loop(HITL)

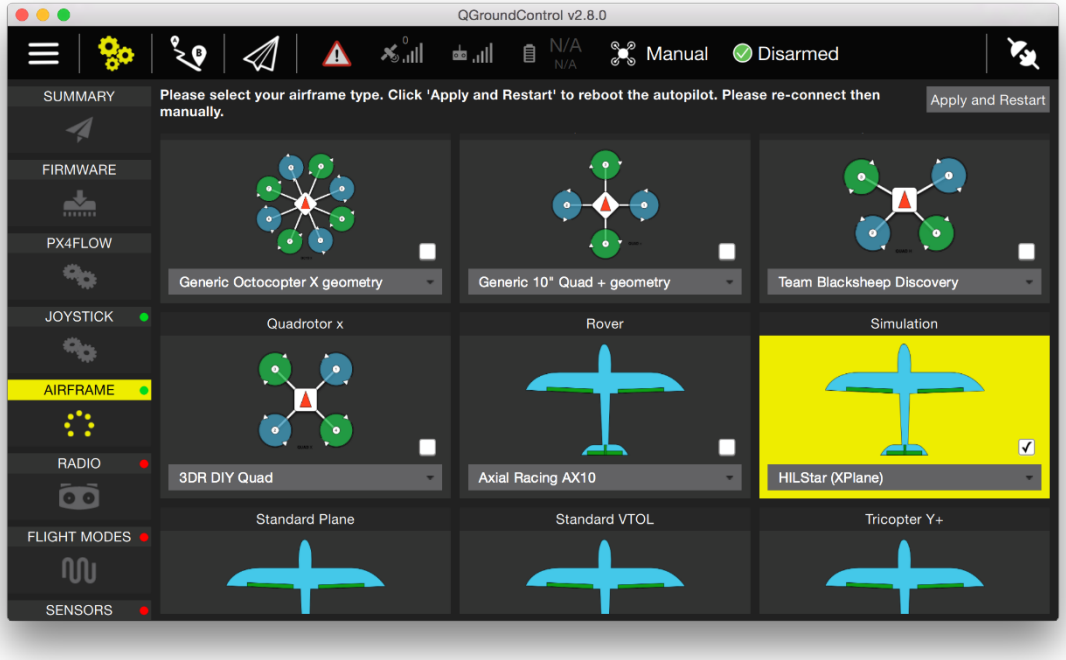




PX4 Flight Stack

Hardware In the Loop(HITL)

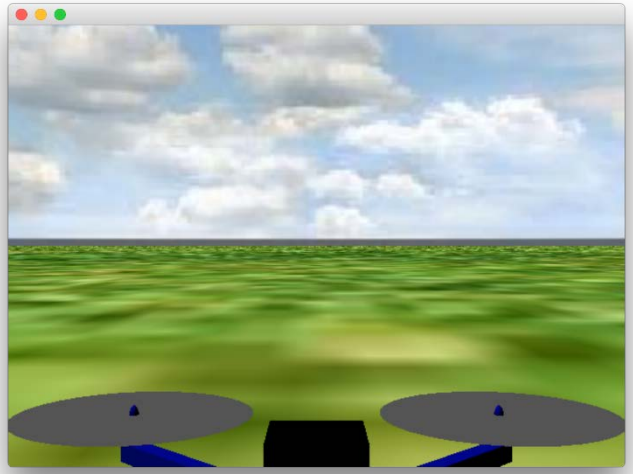




PX4 Flight Stack

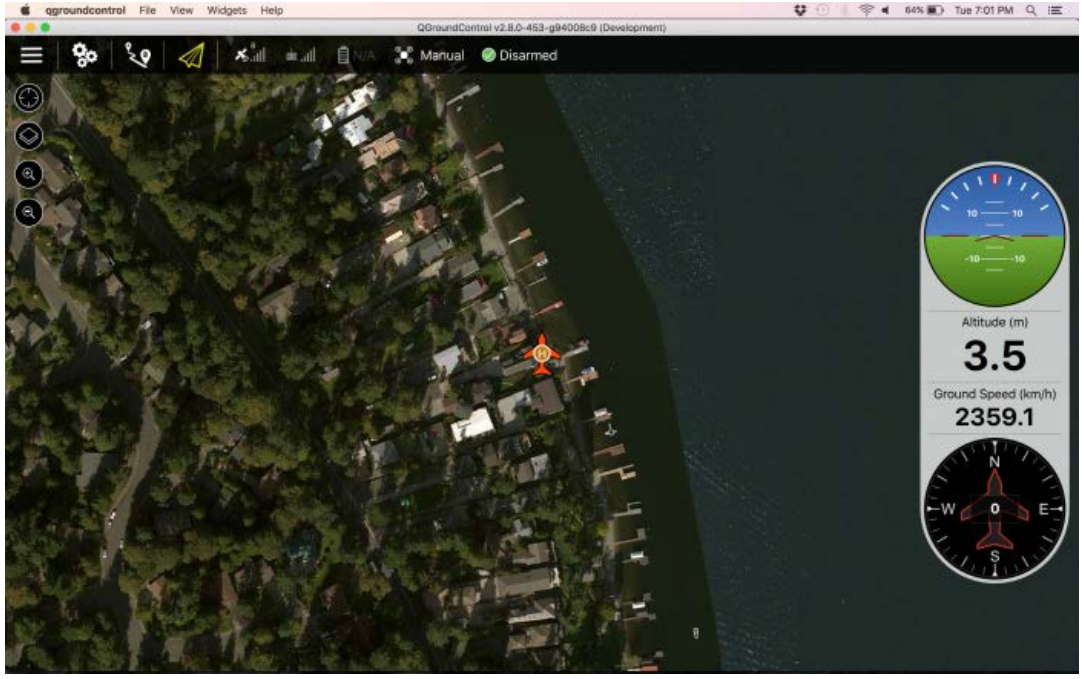
Software In The Loop(SITL)

GAZEBO



JMAVSIM

Ground Control Software



QGROUNDCONTROL

GROUND CONTROL STATION FOR SMALL AIR - LAND - WATER AUTONOMOUS UNMANNED SYSTEMS

- Open-source MAVLink Micro Air Vehicle Communication Protocol with lightweight serialization functions for microcontrollers
- 2/3D aerial maps (Google Earth support) with drag-and-drop waypoints
- In-flight manipulation of waypoints and onboard parameters (in EEPROM)
- Real-time plotting of sensor and telemetry data
- Logging and plotting of sensor logs
- Support for UDP, serial (radio modem) and mesh networks
- Supports multiple Autopilots (pxIMU, ArduPilotMega, SLUGS, MatrixPilot/UAVDevBoard, many more)
- MAVLink protocol supports up to 255 vehicles in parallel and project-specific custom messages can be added
- Head-up-display, support for digital video transmission/display

How can developers implement their applications on a UAV?

Companion Computer

Small Low Power

Odroid Xu4

Odroid C1+

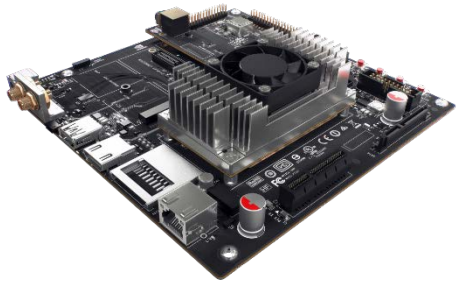
Raspberry Pi

Intel Edison

Large High Power

Intel NUC

Intel NUC



NVIDIA Jetson TK1

How can developers implement their applications on a UAV?

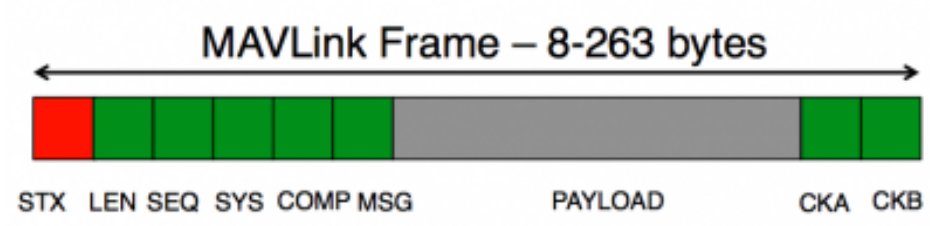
Support of a standardized interface compatible for variable hardwares

: DroneKit

: MAVROS

MAVLINK

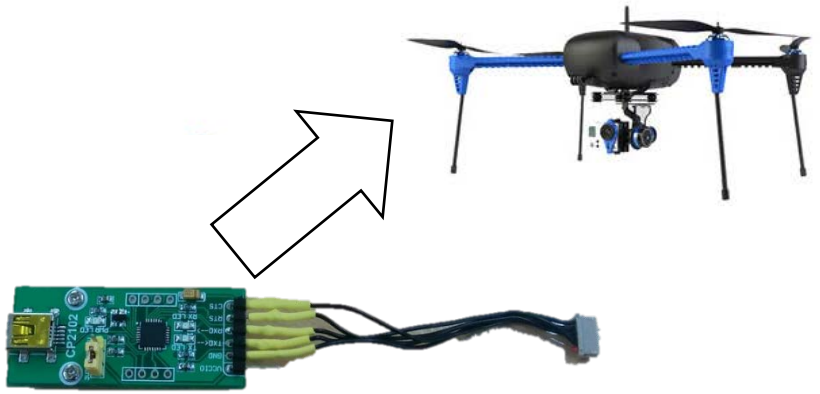
MICRO AIR VEHICLE COMMUNICATION PROTOCOL



Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload “means” and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

OFFBOARD Control

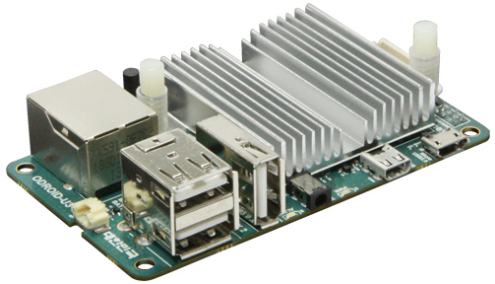
- **Modular approach**



Flight Controller

px4 autopilot

Companion Computer



MAVLINK

MICRO AIR VEHICLE COMMUNICATION PROTOCOL

MAVLINK

MICRO AIR VEHICLE COMMUNICATION PROTOCOL

Hacking

MAVLINK

MICRO AIR VEHICLE COMMUNICATION PROTOCOL

Original Code:

```
// decode the header
errcount = golay_decode(6, buf, gout);
if (gout[0] != netid[0] || gout[1] != netid[1]) {
// its not for our network ID
debug("netid %x %x\n",
(unsigned)gout[0],
(unsigned)gout[1]);
goto failed; }
```

Modified Code:

```
// decode the header
errcount = golay_decode(6, buf, gout);
if (gout[0] != netid[0] || gout[1] != netid[1]) {
// its not for our network ID
/* Modified by __int128 */
// Set our radio to use the captured packets NetID
param_set(PARAM_NETID, gout[0]))
// Save the value to flash
param_save();
// To read the new value we need to reboot. Rebooting
RSTSRC |= (1 4);
/* End of what was added by __int128*/ }
```

http://diydrones.com/profiles/blogs/hijacking-quadcopters-with-a-mavlink-exploit

DroneKit (Not for this Seminar)

YOUR AERIAL PLATFORM

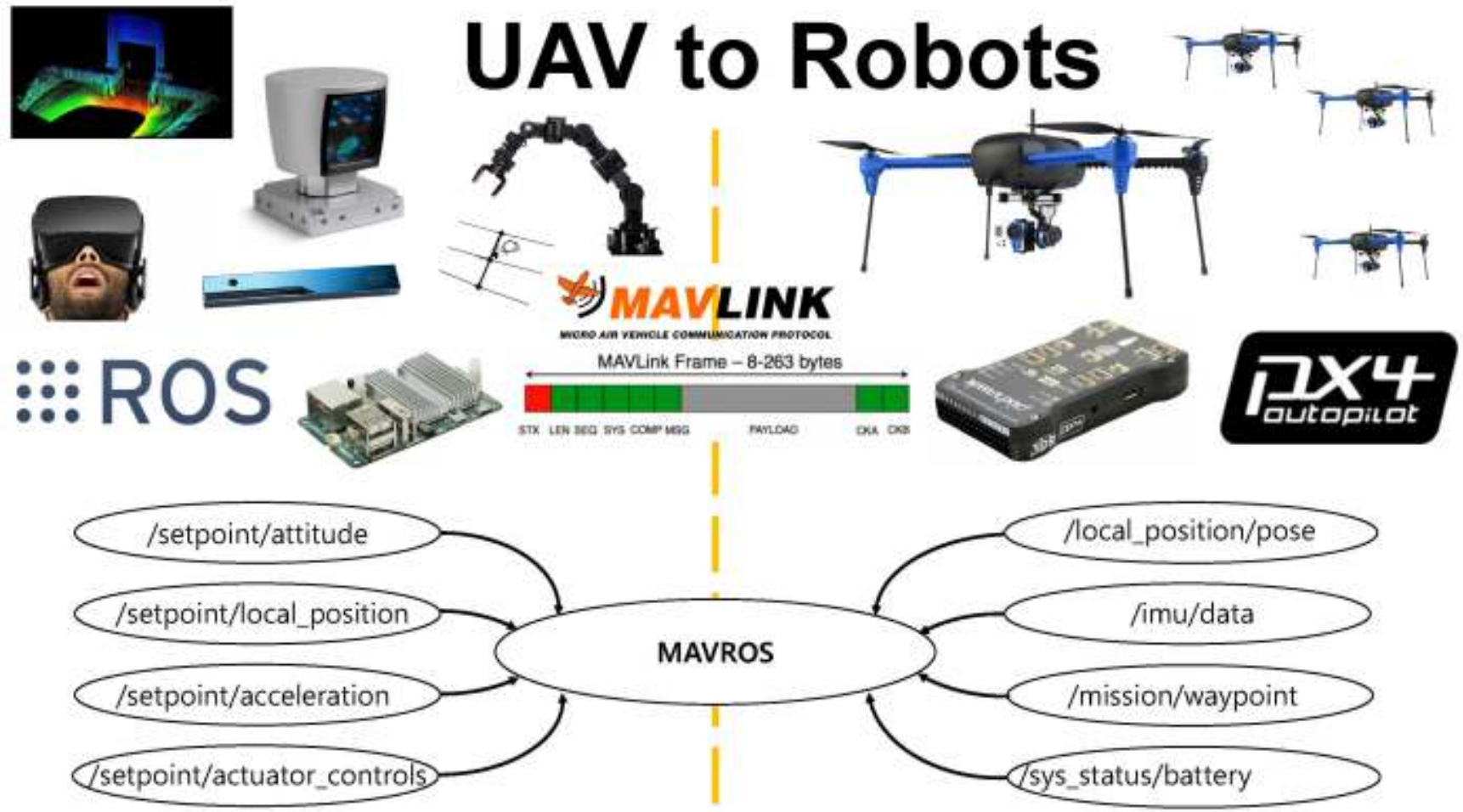
DroneKit offers an SDK and web API to easily develop apps for your drones

INTELLIGENT PATH PLANNING

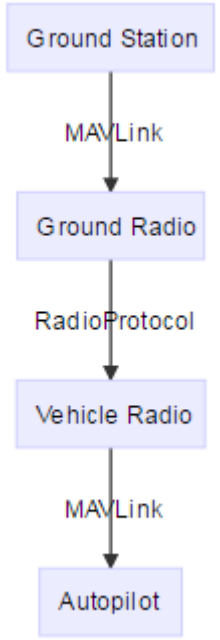
AUTONOMOUS FLIGHT

LIVE TELEMETRY

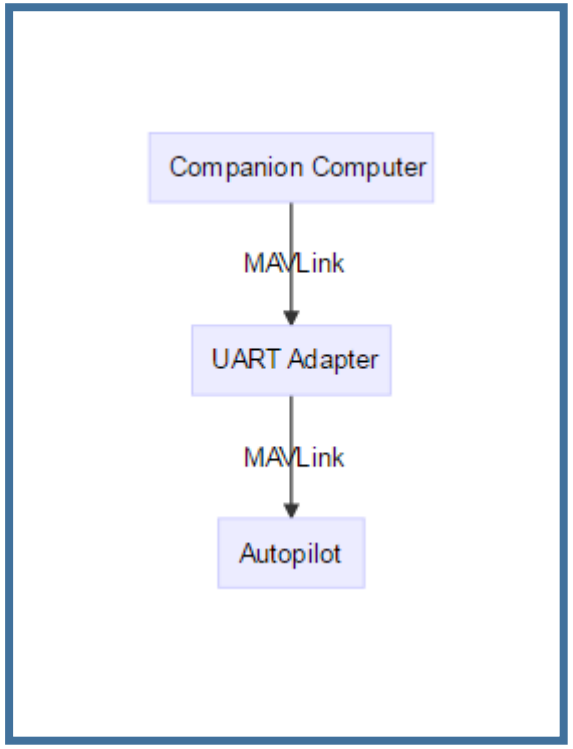
MAVROS


UAV to Robots
MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL
MAVLink Frame – 8-263 bytes
STX LEN SEQ SYS COMP MSG PAYLOAD CKA CKB
ROS
px4 autopilot
/setpoint/attitude
/setpoint/local_position
/setpoint/acceleration
/setpoint/actuator_controls
MAVROS
/local_position/pose
/imu/data
/mission/waypoint
/sys_status/battery

OFFBOARD Control



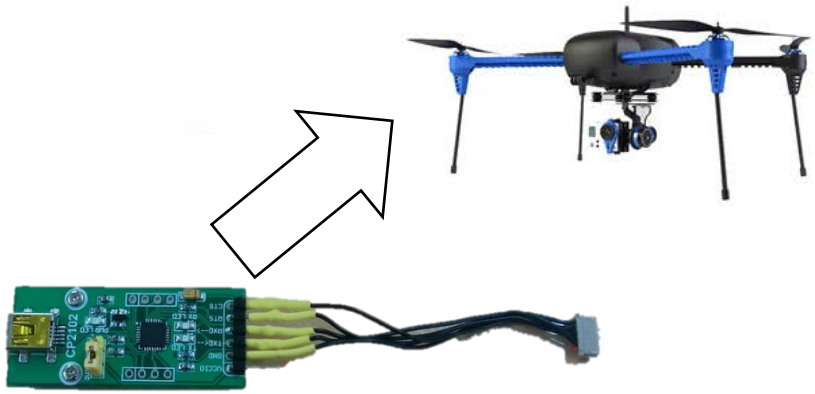
Serial radio Setup



Onboard processor

MAVROS/OFFBOARD Control

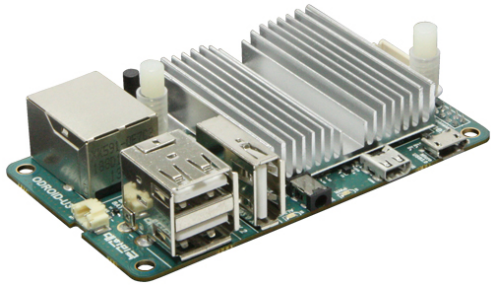
- **Modular approach**



Flight Controller

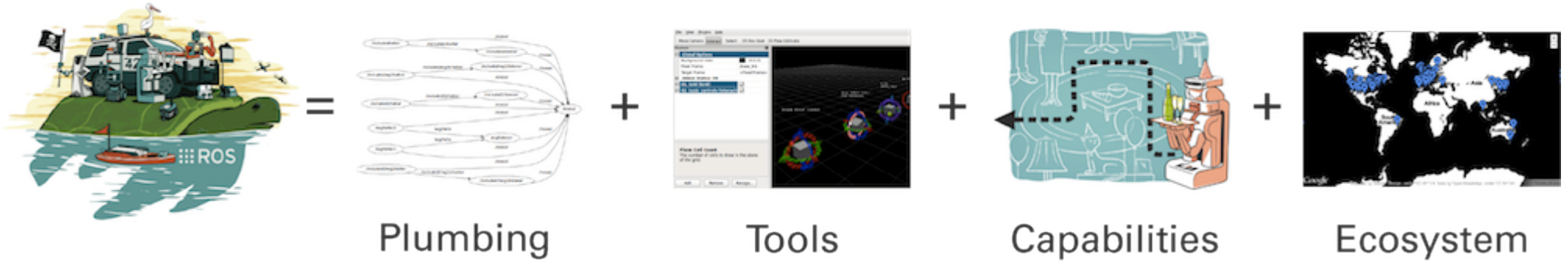
Companion Computer

ROS.org



MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

ROS

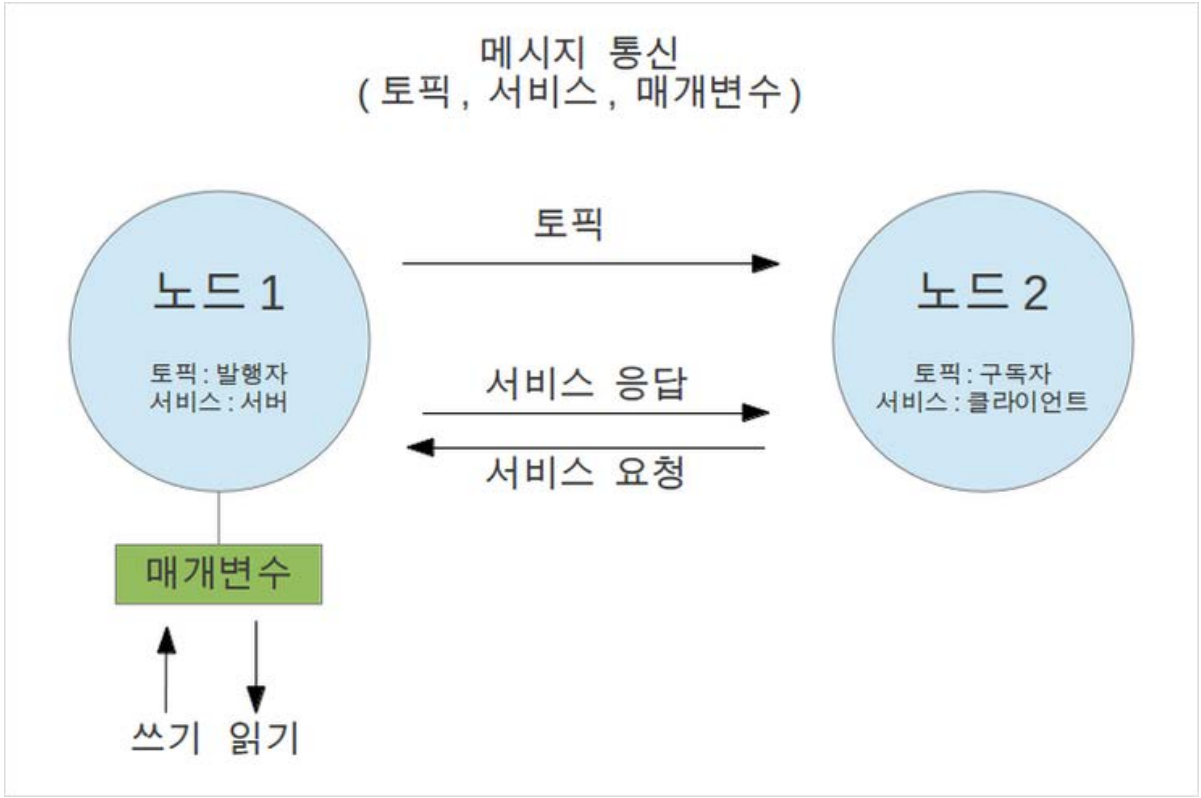


The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

- ***Peer to peer***
- ***Multi-lingual***
- ***Tools-based***
- ***Thin***
- ***Free and Opensource***

Quigley, Morgan, et al. "ROS: an open-source Robot Operating System."*ICRA workshop on open source software*. Vol. 3. No. 3.2. 2009.

ROS



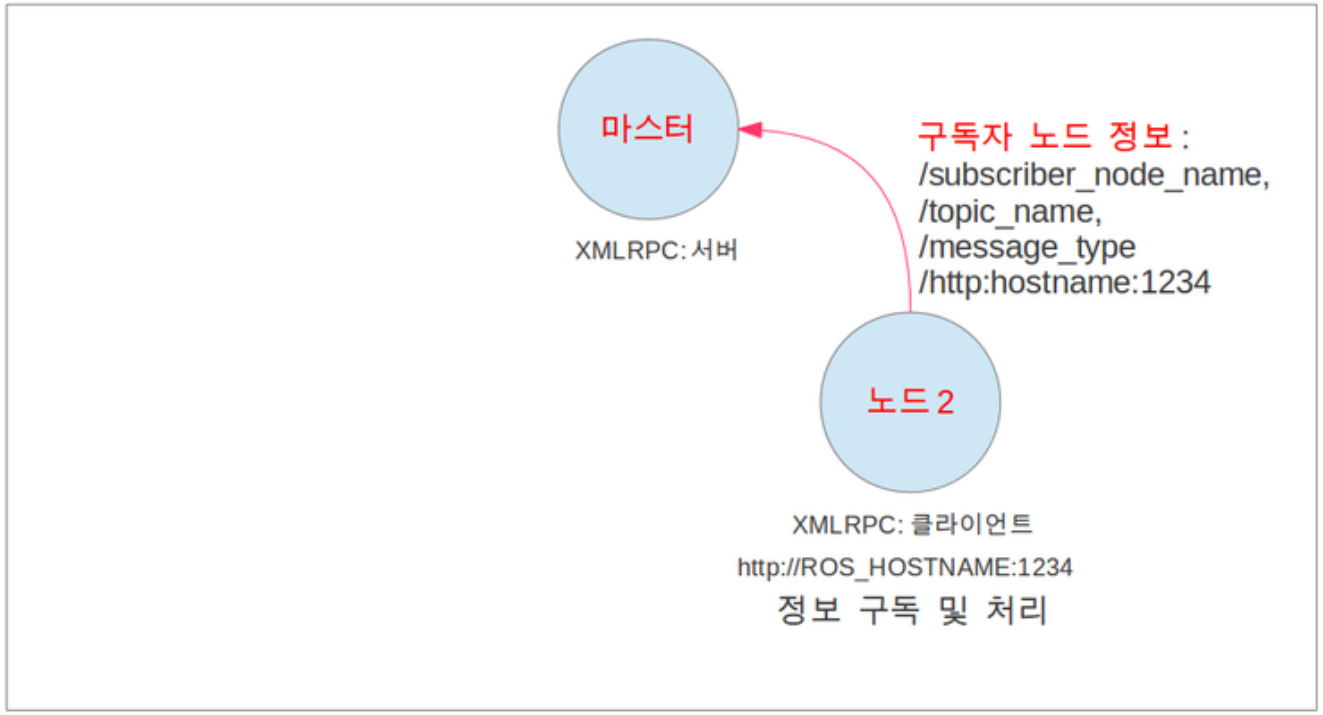
출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

ROS

출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

```
roscore
```

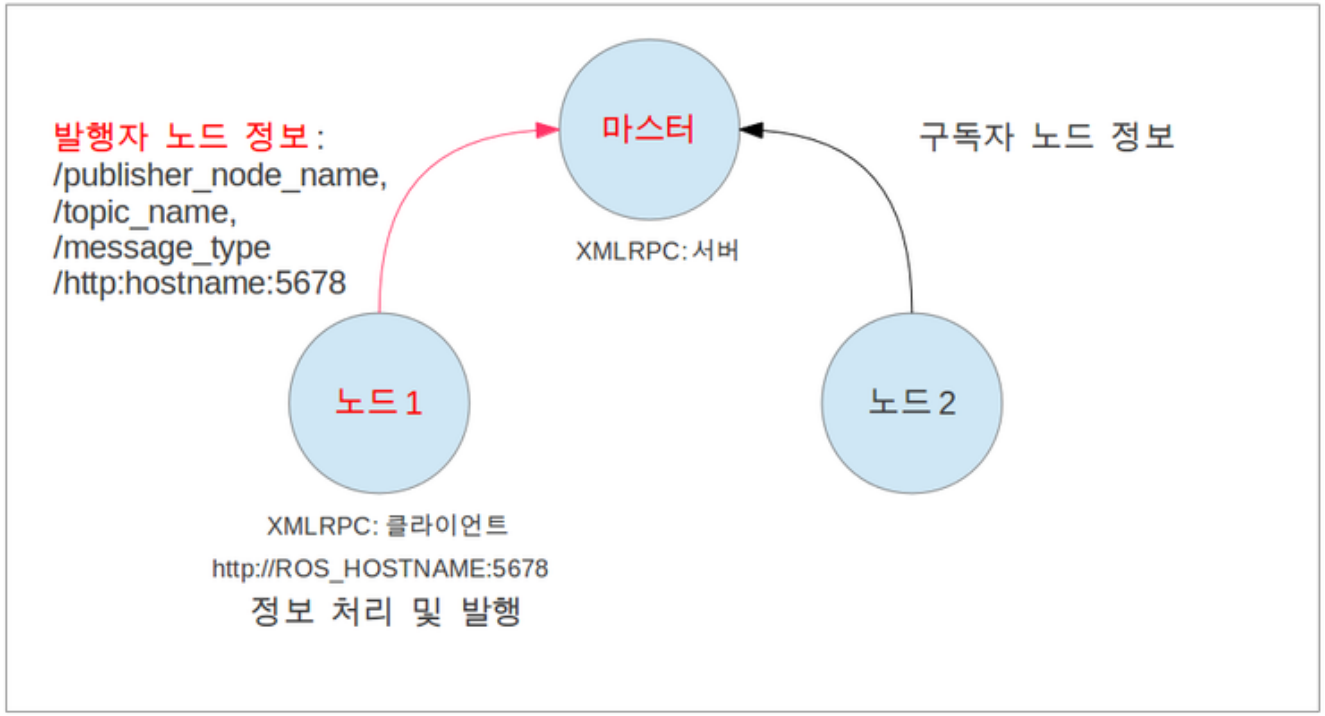
ROS



출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

```
rosrun <package name> <node name>
```

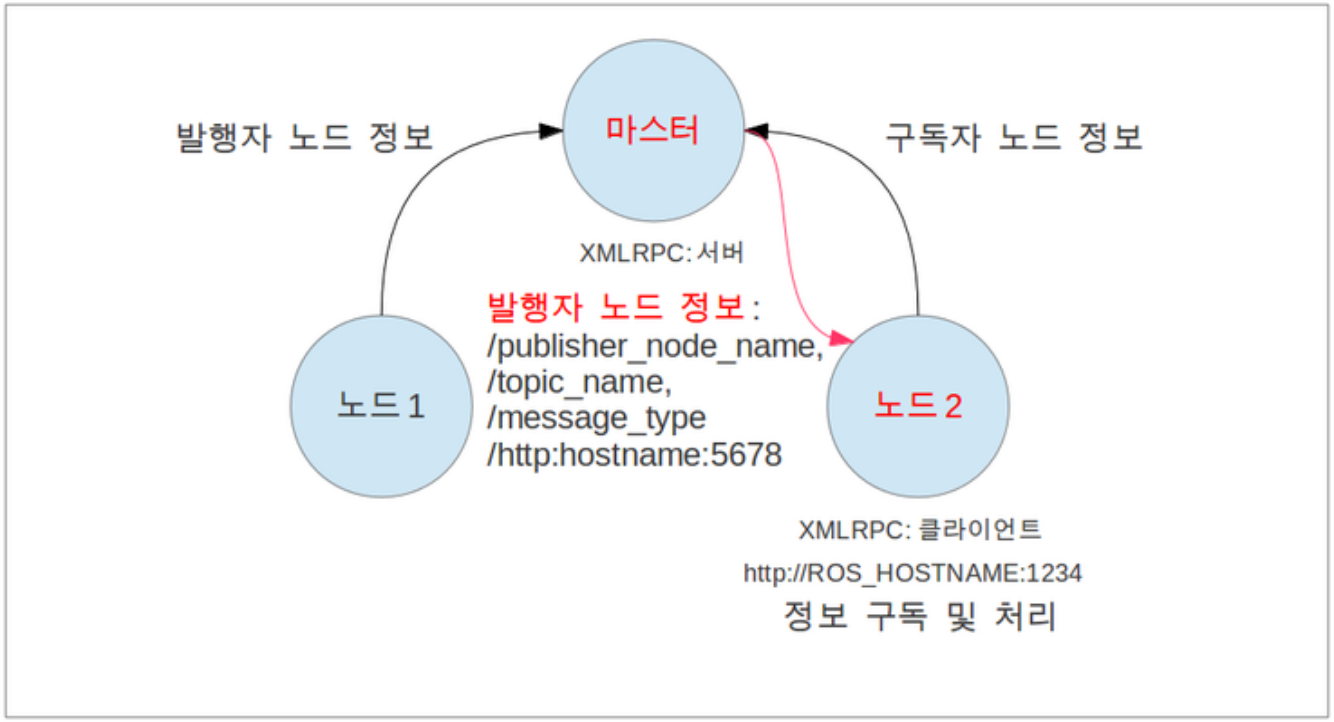
ROS



출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

```
rosrun <package name> <node name>
```

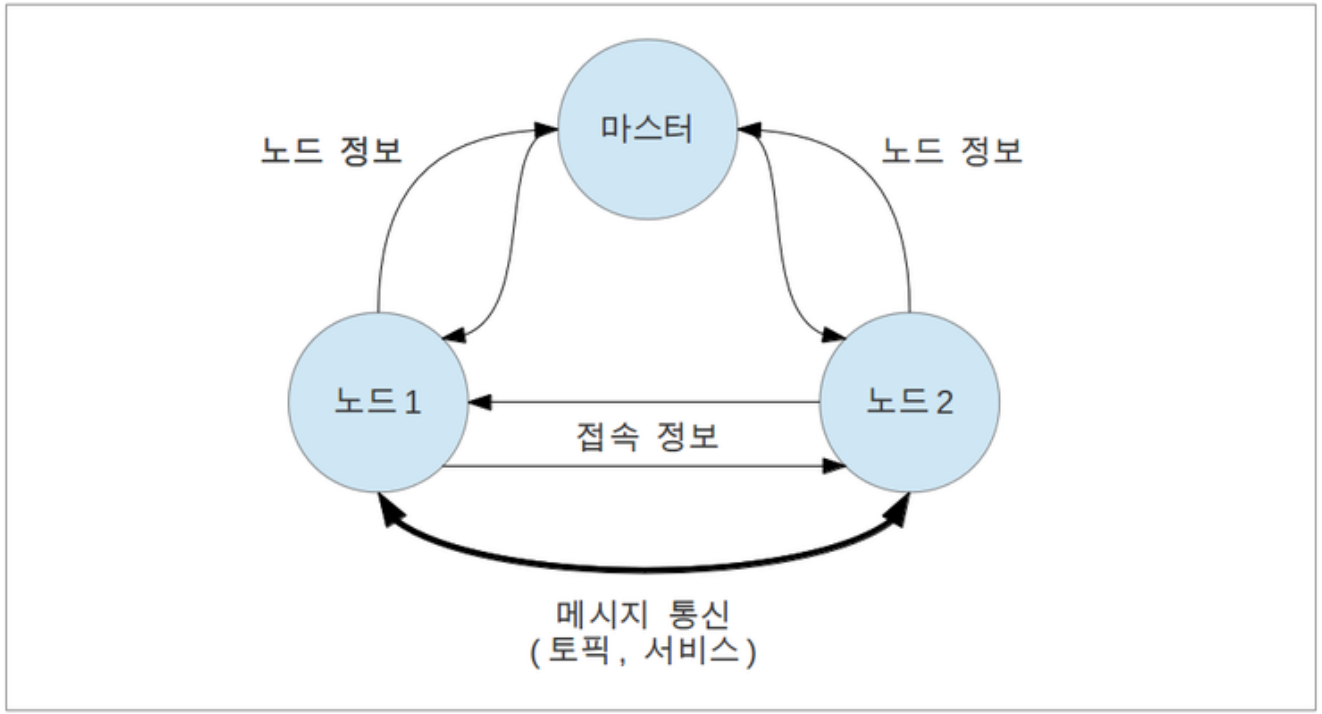
ROS



출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

```
rosrun <package name> <node name>
```

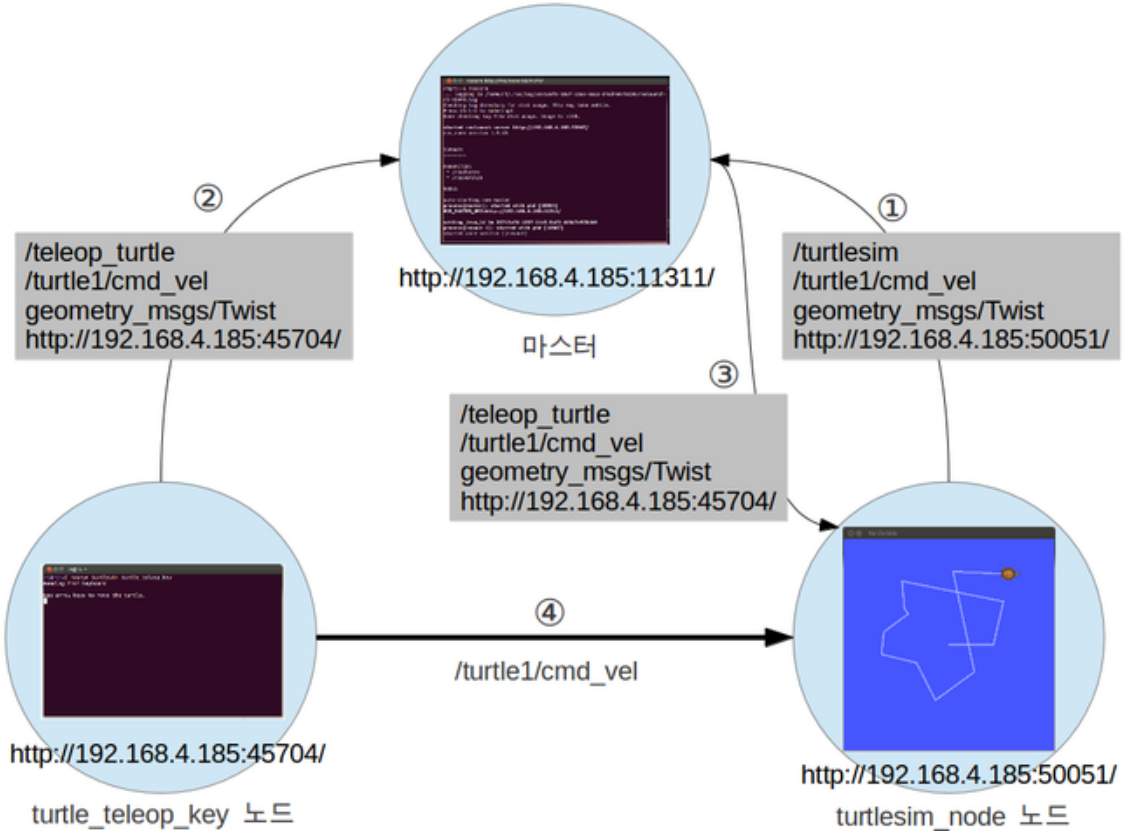
ROS



출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

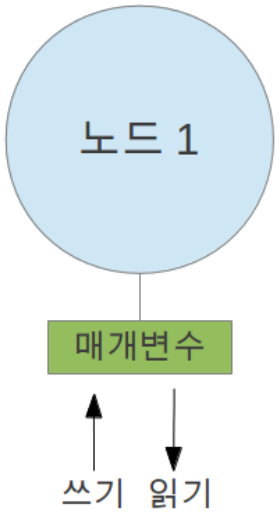
```
roslaunch  <package name> <launchfilename.launch>
```

ROS



출처: 표윤석, 로봇운영체제 강좌, 오로카 http://cafe.naver.com/openrt/3000

ROS-Messages



geometry_msgs/Twist Message

File: geometry_msgs/Twist.msg

Raw Message Definition

```
# This expresses velocity in free space broken into its linear and angular parts.
Vector3  linear
Vector3  angular
```

Compact Message Definition

```
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

autogenerated on Mon, 14 Mar 2016 10:24:57

ROS-Messages

geometry_msgs/Vector3 Message

File: geometry_msgs/Vector3.msg

Raw Message Definition

```
# This represents a vector in free space.
# It is only meant to represent a direction. Therefore, it does not
# make sense to apply a translation to it (e.g., when applying a
# generic rigid transformation to a Vector3, tf2 will only apply the
# rotation). If you want your data to be translatable too, use the
# geometry_msgs/Point message instead.

float64 x
float64 y
float64 z
```

Compact Message Definition

```
float64 x
float64 y
float64 z
```

autogenerated on Mon, 14 Mar 2016 10:24:57

MAVROS

Subsribed Topics

Position control

Accepts PoseStamped for position and yaw

•The Pose message position is sent to controllers

•Only yaw from orientation quaternion is sent

Topics :

•/mavros/setpoint/local_position

Attitude control

Accepts PoseStamped or PoseWithCovarianceStamped or TwistStamped for **angular velocity**(attitude rates) ,

Throttle range is 0.0 to 1.0

Topics :

•/mavros/setpoint/attitude (pose)

•/mavros/setpoint/att_throttle

•/mavros/setpoint/cmd_vel (twist)

Velocity control

Accepts TwistStamped for **linear velocities**

•The Twist message linear velocity is sent to controllers

•Angular velocity on Z (yaw rate) is sent

Topics :

•/mavros/setpoint/cmd_vel

Acceleration control

Accepts Vector3Stamped for the accelerations

•The XYZ vectors from message are sent to controllers. Whether targets are acceleration or force can be set in the plugin

Topics :

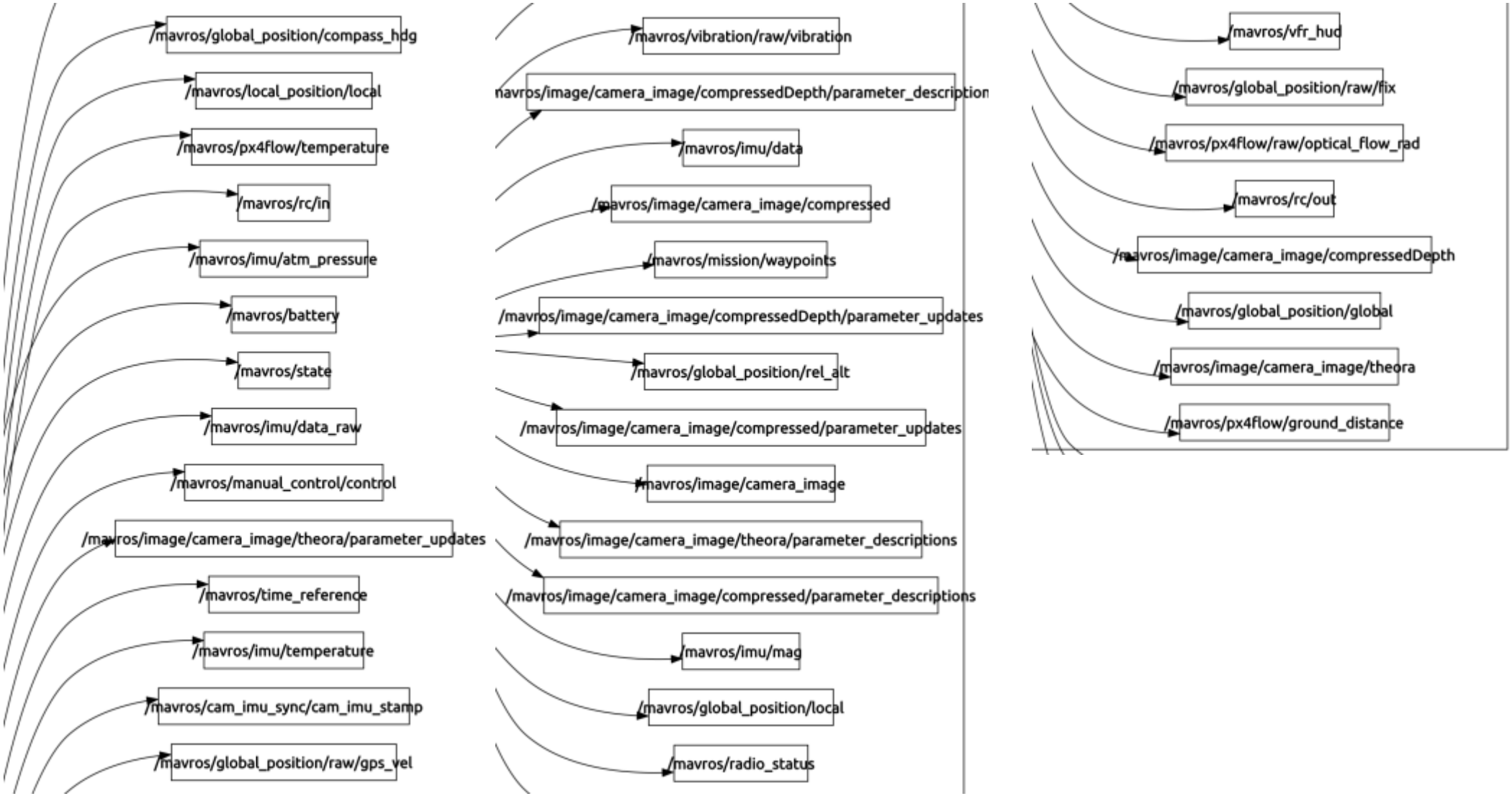
•/mavros/setpoint/accel

Acuator Control

Topics :

•/mavros/actuator_controls

Published Topics



MAVROS/ OFFBOARD Control

- Publish Node

```
#include <ros/ros.h>
#include <std_msgs/String.h>
#include <stdio.h>
#include "geometry_msgs/PoseStamped.h"
#include "geometry_msgs/Vector3Stamped.h"

int main(int argc, char **argv)
{
   ros::init(argc, argv, "pub_setpoints");
   ros::NodeHandle n;

   ros::Publisher chatter_pub = n.advertise<geometry_msgs::PoseStamped>("/mavros/setpo
local",100);
   ros::Rate loop_rate(100);
   ros::spinOnce();

   geometry_msgs::PoseStamped msg;
   int count = 1;

               //PositionReciever qp;
               //Body some_object;
               //qp.connect_to_server();


   while(ros::ok()){
           //some_object = qp.getStatus();
               // some_object.print();
               //printf("%f\n",some_object.position_x);
       msg.header.stamp = ros::Time::now();
       msg.header.seq=count;
       msg.header.frame_id = 1;
       msg.pose.position.x = 0.0;//0.001*some_object.position_x;
       msg.pose.position.y = 0.0;//0.001*some_object.position_y;
       msg.pose.position.z = 1.0;//0.001*some_object.position_z;
       msg.pose.orientation.x = 0;
       msg.pose.orientation.y = 0;
       msg.pose.orientation.z = 0;
       msg.pose.orientation.w = 1;

       chatter_pub.publish(msg);
       ros::spinOnce();
       count++;
       loop_rate.sleep();
   }


   return 0;
}
```

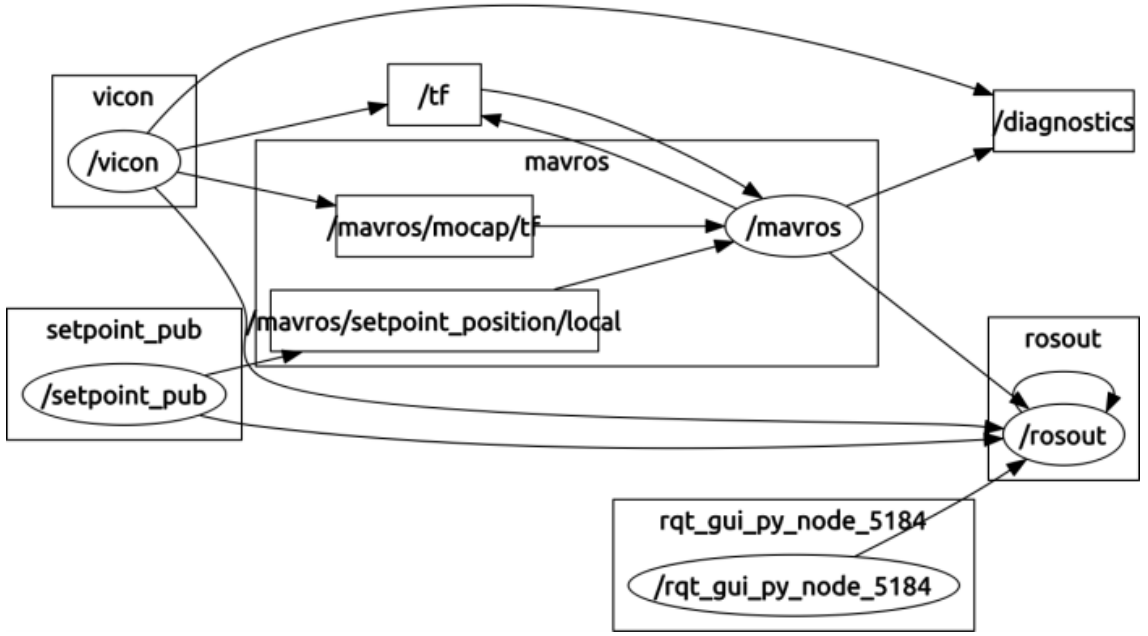
MAVROS/ OFFBOARD Control

- Launch File

```
<launch>

<!--arg name="fcu_url" default="serial:///dev/ttyUSB0:921600" -->
<arg name="fcu_url" default="udp://:14540@192.168.1.36:14557"/>
<arg name="gcs_url" default="udp://:14556@192.168.150.2:14550" />
<arg name="tgt_system" default="1" />
<arg name="tgt_component" default="1" />


<node name="mavros" pkg="mavros" type="mavros_node" output="screen">
        <param name="fcu_url" value="$(arg fcu_url)" />
        <param name="gcs_url" value="$(arg gcs_url)" />
        <param name="target_system_id" value="$(arg tgt_system)" />
        <param name="target_component_id" value="$(arg tgt_component)" />

        <!--rosparam command="load" file="$(find mavros)/launch/px4_blacklist.yaml"-->

        <!-- enable heartbeat send and reduce timeout -->
        <param name="conn_heartbeat" value="5.0" />
        <param name="conn_timeout" value="5.0" />
        <!-- automatically start mavlink on USB -->
        <param name="startup_px4_usb_quirk" value="true" />
        <param name="mocap/use_tf" value="true"/>
        <param name="mocap/use_pose" value="false"/>
</node>

<node name="setpoint_pub" pkg="inrol_ros" type="pub_setpoints_pos" output="screen">

</node>

<node pkg="vicon_bridge" type="vicon_bridge" name="vicon" output="screen">
        <param name="stream_mode" value="ServerPush" type="str" />
        <!--param name="datastream_hostport" value="147.47.175.54:801" type="str" /-->
        <param name="datastream_hostport" value="147.46.175.54:801" type="str" />
        <param name="tf_ref_frame_id" value="/world" type="str" />
        <remap from="/vicon/quad/quad" to="/mavros/mocap/tf"/>
</node>


</launch>
```

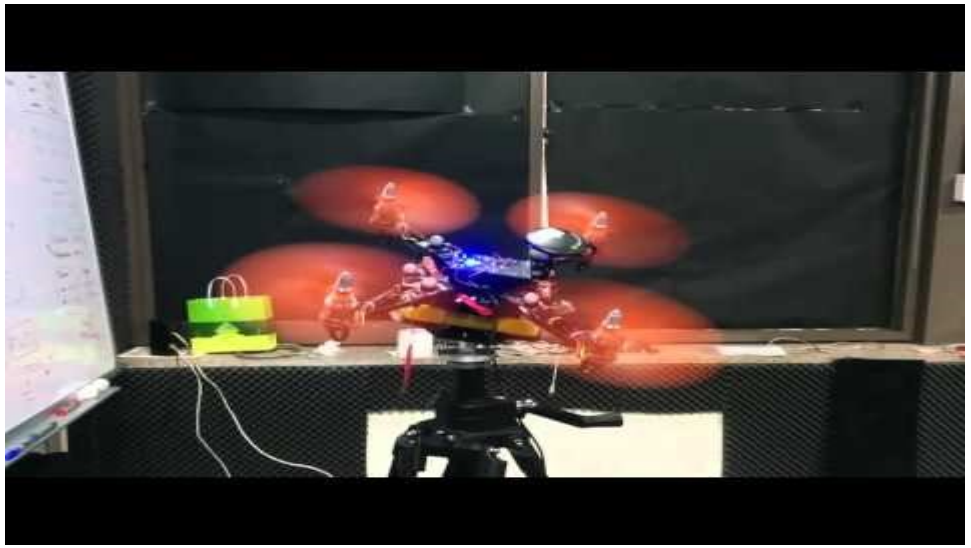
Offboard control

Setpoint Position Topic



Offboard control

Setpoint Attitude Topic



- **Attitude command on spherical joint**
- **Yaw drift in high angle attitude**
  - Due to compass bias induced by motor current

GAZEBO

Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. At your fingertips is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. Best of all, Gazebo is free with a vibrant community.

Features

Dynamics Simulation

Access multiple high-performance physics engines including ODE, Bullet, Simbody, and DART.

Advanced 3D Graphics

Utilizing OGRE, Gazebo provides realistic rendering of environments including high-quality lighting, shadows, and textures.

Sensors and Noise

Generate sensor data, optionally with noise, from laser range finders, 2D/3D cameras, Kinect style sensors, contact sensors, force-torque, and more.

Plugins

Develop custom plugins for robot, sensor, and environmental control. Plugins provide direct access to Gazebo's API.

Robot Models

Many robots are provided including PR2, Pioneer2 DX, iRobot Create, and TurtleBot. Or build your own using SDF.

TCP/IP Transport

Run simulation on remote servers, and interface to Gazebo through socket-based message passing using Google Protobufs.

Cloud Simulation

Use CloudSim to run Gazebo on Amazon, Softlayer, or your own OpenStack instance.

Command Line Tools

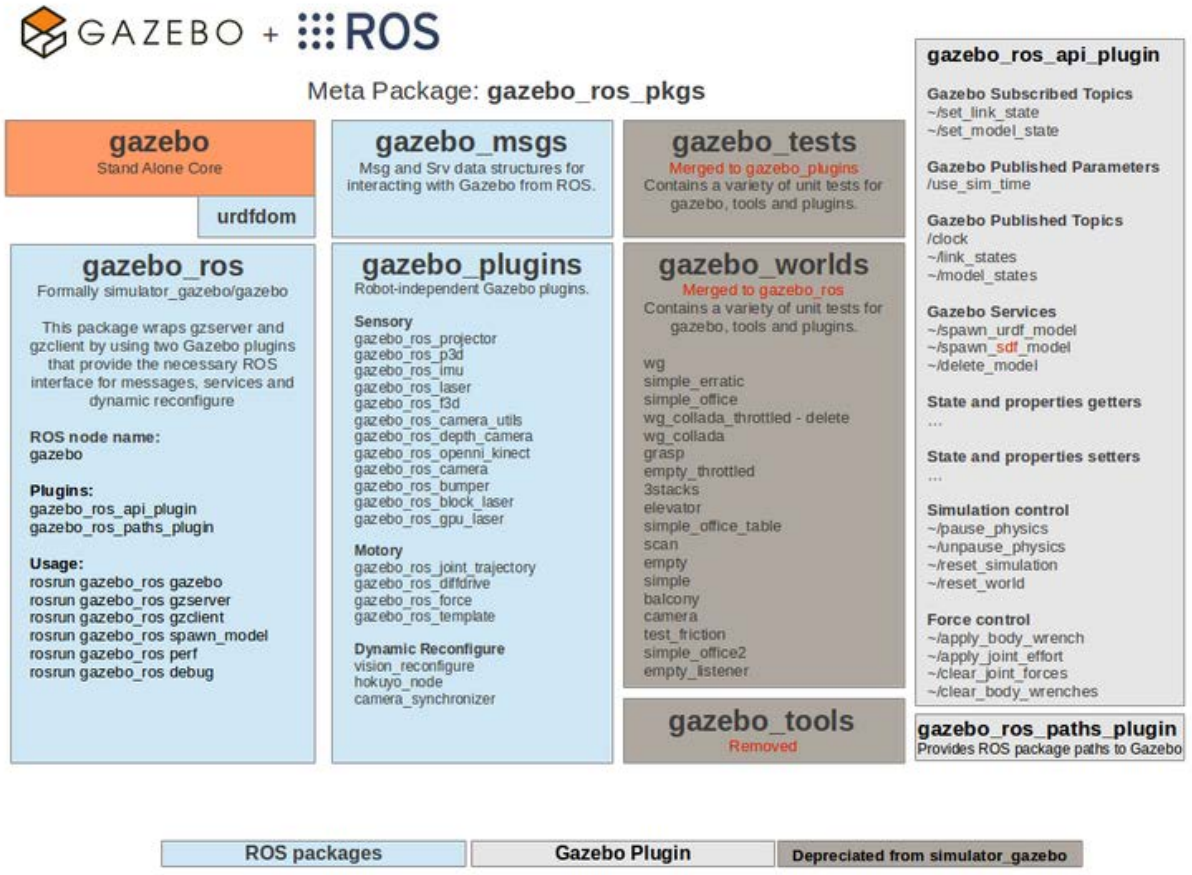
Extensive command line tools facilitate simulation introspection and control.

MAVROS/ OFFBOARD Control

- Launch File (SITL)

```
<launch>

<!--arg name="fcu_url" default="serial:///dev/ttyUSB0:921600" -->
<arg name="fcu_url" default="udp://:14540@192.168.1.36:14557"/>
<arg name="gcs_url" default="udp://:14556@192.168.150.2:14550" />
<arg name="tgt_system" default="1" />
<arg name="tgt_component" default="1" />


<node name="mavros" pkg="mavros" type="mavros_node" output="screen">
        <param name="fcu_url" value="$(arg fcu_url)" />
        <param name="gcs_url" value="$(arg gcs_url)" />
        <param name="target_system_id" value="$(arg tgt_system)" />
        <param name="target_component_id" value="$(arg tgt_component)" />

        <!--rosparam command="load" file="$(find mavros)/launch/px4_blacklist.yaml"-->

        <!-- enable heartbeat send and reduce timeout -->
        <param name="conn_heartbeat" value="5.0" />
        <param name="conn_timeout" value="5.0" />
        <!-- automatically start mavlink on USB -->
        <param name="startup_px4_usb_quirk" value="true" />
        <param name="mocap/use_tf" value="true"/>
        <param name="mocap/use_pose" value="false"/>
</node>

<node name="setpoint_pub" pkg="inrol_ros" type="pub_setpoints_pos" output="screen">

</node>

<node pkg="vicon_bridge" type="vicon_bridge" name="vicon" output="screen">
        <param name="stream_mode" value="ServerPush" type="str" />
        <!--param name="datastream_hostport" value="147.47.175.54:801" type="str" /-->
        <param name="datastream_hostport" value="147.46.175.54:801" type="str" />
        <param name="tf_ref_frame_id" value="/world" type="str" />
        <remap from="/vicon/quad/quad" to="/mavros/mocap/tf"/>
</node>


</launch>
```

SITL Gazebo


GAZEBO + ROS
Meta Package: gazebo_ros_pkgs
gazebo
Stand Alone Core
urdfdom
gazebo_msgs
Msg and Srv data structures for interacting with Gazebo from ROS.
gazebo_tests
Merged to gazebo_plugins
Contains a variety of unit tests for gazebo, tools and plugins.
gazebo_ros_api_plugin
Gazebo Subscribed Topics
~/set_link_state
~/set_model_state
Gazebo Published Parameters
/use_sim_time
Gazebo Published Topics
/clock
~/link_states
~/model_states
Gazebo Services
~/spawn_urdf_model
~/spawn_sdf_model
~/delete_model
State and properties getters
...
State and properties setters
...
Simulation control
~/pause_physics
~/unpause_physics
~/reset_simulation
~/reset_world
Force control
~/apply_body_wrench
~/apply_joint_effort
~/clear_joint_forces
~/clear_body_wrenches
gazebo_ros
Formally simulator_gazebo/gazebo
This package wraps gzserver and gzclient by using two Gazebo plugins that provide the necessary ROS interface for messages, services and dynamic reconfigure
ROS node name:
gazebo
Plugins:
gazebo_ros_api_plugin
gazebo_ros_paths_plugin
Usage:
rosrun gazebo_ros gazebo
rosrun gazebo_ros gzserver
rosrun gazebo_ros gzclient
rosrun gazebo_ros spawn_model
rosrun gazebo_ros perf
rosrun gazebo_ros debug
gazebo_plugins
Robot-independent Gazebo plugins.
Sensory
gazebo_ros_projector
gazebo_ros_p3d
gazebo_ros_imu
gazebo_ros_laser
gazebo_ros_f3d
gazebo_ros_camera_utils
gazebo_ros_depth_camera
gazebo_ros_openni_kinect
gazebo_ros_camera
gazebo_ros_bumper
gazebo_ros_block_laser
gazebo_ros_gpu_laser
Motory
gazebo_ros_joint_trajectory
gazebo_ros_diffdrive
gazebo_ros_force
gazebo_ros_template
Dynamic Reconfigure
vision_reconfigure
hokuyo_node
camera_synchronizer
gazebo_worlds
Merged to gazebo_ros
Contains a variety of unit tests for gazebo, tools and plugins.
wg
simple_erratic
simple_office
wg_collada_throttled - delete
wg_collada
grasp
empty_throttled
3stacks
elevator
simple_office_table
scan
empty
simple
balcony
camera
test_friction
simple_office2
empty_listener
gazebo_tools
Removed
gazebo_ros_paths_plugin
Provides ROS package paths to Gazebo
ROS packages
Gazebo Plugin
Depreciated from simulator_gazebo

ARDUPILOT (Not for this Seminar)

Standard Opensource Quadrotors

- Better flying qualities(Vibration, etc)
- Low Price (~600 usd)

3DRobotics IRIS

- Onboard Pixhawk
- Price : 599$

3DRobotics SOLO

- Onboard Pixhawk2
- Onboard Linux Computer (Operating ROS)
- Price : 990$

Standard Opensource Quadrotors

- Better flying qualities(Vibration, etc)
- Low Price (~600 usd)

Typhoon H 480

- Onboard Pixhawk2
- Onboard Linux Computer (Operating ROS)
- Price : ~1200$

3DRobotics SOLO

- Onboard Pixhawk2
- Onboard Linux Computer (Operating ROS)
- Price : 990$