EE478
PX4 Gazebo Simulation

(TA) Seongwoo Moon, Sungjae Min, Donghun Han
School of Electrical Engineering
KAIST

March 13, 2025

{seongwoo.moon,sungjae_min, donghun.han}@kaist.ac.kr

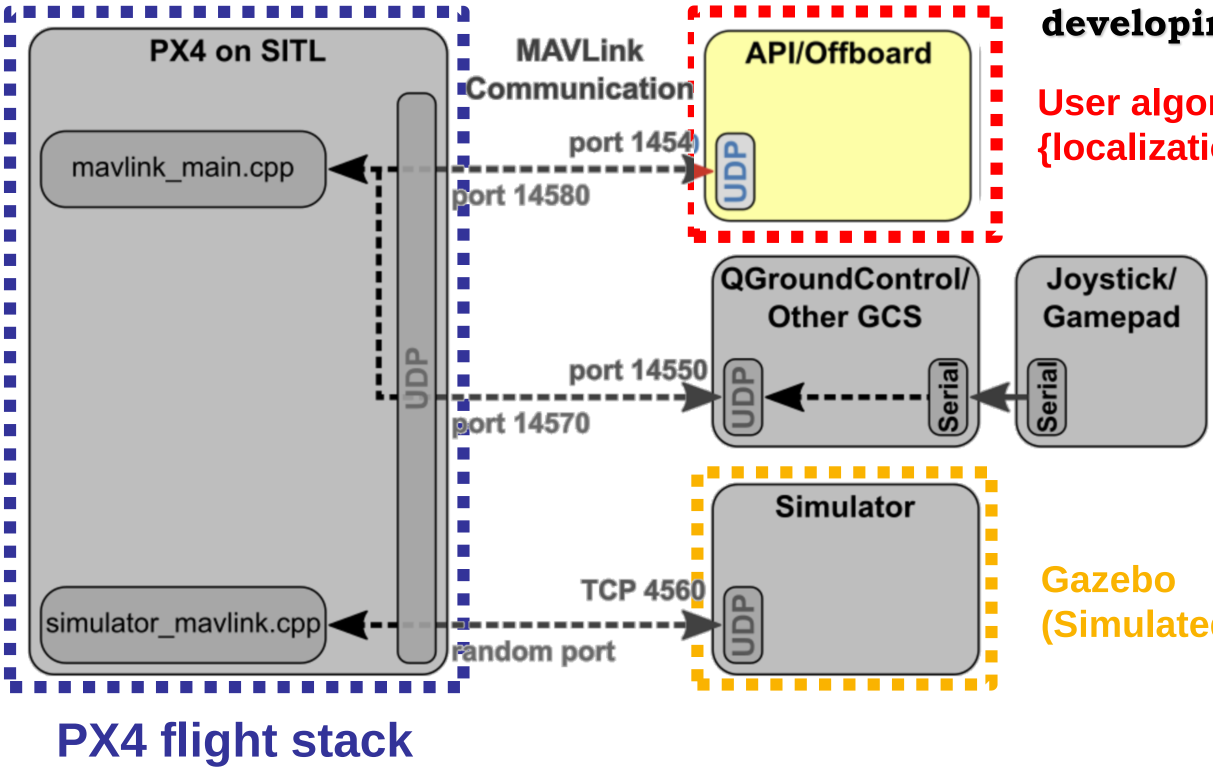
Contents

PX4 Gazebo Simulator
&
Controller Structure

PX4 Gazebo Simulator

❖ PX4 Software-in-the-loop(SITL) Simulation
- Test and debug the drone software stack in simulation
- PX4 supports **ROS** – **gazebo** simulation
- Essential before trying real flight!

We can focus on the developing algorithms!



User algorithms
{localization, perception,
planning, control}

Gazebo
(Simulated environment)

PX4 Gazebo Simulator with MAVROS

❖ MAVROS
- MAVLINK : Lightweight messaging protocol for drones
- MAVROS : MAVLINK extension for communicating with ROS.

- MAVROS is essential to make the drone fly autonomously using ROS

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

Protocol for drone

MAVROS

Convert ROS message to MAVLINK protocol

ROS
Robot Operating System

Calculate and publish ROS message

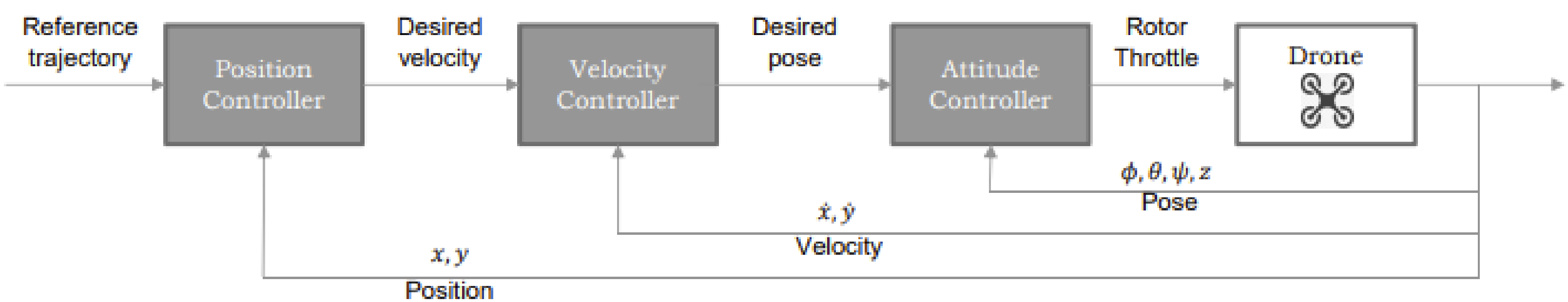
Controller Structure

❖ Cascade system

◆ Position Controller
- Controller for following a reference position using position feedback
- Easy to implement, fine tuning is uncomfortable

◆ Velocity Controller
- Controller for following a reference position using position feedback
- Easy to implement, can adjust velocity relatively easy

◆ Attitude Controller
- Controller for following a reference position using position feedback
- Freely design controller, but hard to implement



Overall cascade drone control architecture

Install PX4 Gazebo Simulator

Install Gazebo Simulator

❖ Install Dependencies for PX4 simulator

```
# Clone repository
git clone https://github.com/PX4/PX4-Autopilot.git
cd PX4-Autopilot
git checkout v1.14.0
git submodule update --init --recursive

# Install toolchain
bash ./Tools/setup/ubuntu.sh –no-nuttx
sudo reboot now
```

Github Link : https://github.com/PX4/PX4-Autopilot
Reference : https://docs.px4.io/v1.14/en/dev_setup/dev_env_linux_ubuntu.html

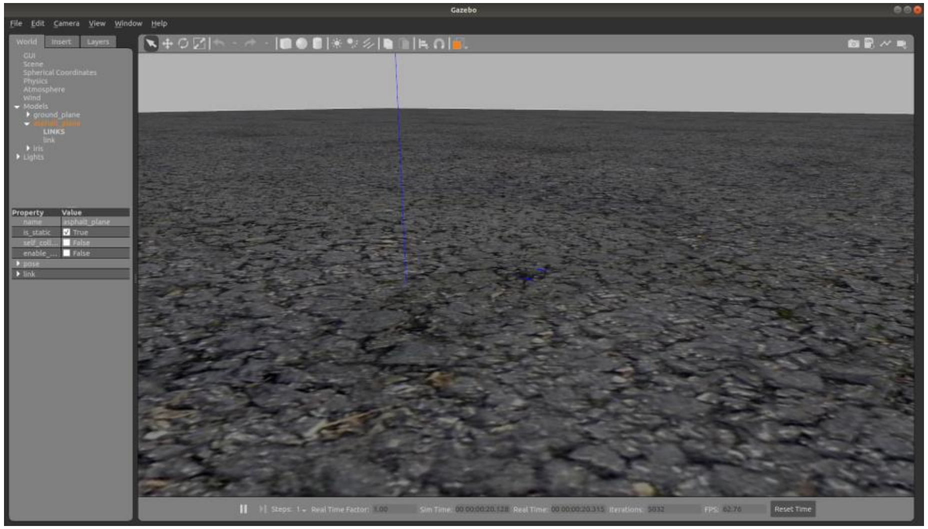
❖ I recommend you to search on the **google** first when error occurs, which will be faster than asking directly to TAs in solving problems

Install Gazebo Simulator

❖ Run PX4 simulator

```
cd <your PX4-Autopilot directory>
make px4_sitl_default gazebo-classic
```

When you can see the drone, move to the next step.



Gazebo Simulation

Drone in the simulation

Install MAVROS and QGrondControl

❖ Install MAVROS

```
sudo apt-get install ros-${ROS_DISTRO}-mavros ros-${ROS_DISTRO}-mavros-msgs  ros-${ROS_DISTRO}-mavros-extras
wget https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/install_geographiclib_datasets.sh
sudo bash ./install_geographiclib_datasets.sh
```

Reference : https://docs.px4.io/v1.14/en/ros/mavros_installation.html

❖ Install QGroundControl

Download QGC v4.2.8 from the link below

https://github.com/mavlink/qgroundcontrol/releases/

```
# Install dependencies
sudo usermod -a -G dialout $USER
sudo apt-get remove modemmanager -y
sudo apt install gstreamer1.0-plugins-bad gstreamer1.0-libav gstreamer1.0-gl -y
sudo apt install libfuse2 -y
sudo apt install libxcb-xinerama0 libxkbcommon-x11-0 libxcb-cursor-dev -y

# Reboot
sudo reboot now

# After reboot
sudo chmod +x <Path to downloaded QGC>/QGroundControl.AppImage
```

Install Gazebo Simulator

- Run PX4 simulator with MAVROS
  - {PX4_DIR} means the path of the directory PX4 is installed
  - Ex) /home/usrg/PX4-Autopilot
  - Please write **your own** PX4 installation path.

```
# Add to your ~/.bashrc file
# Careful with order
source ~/catkin_ws/devel/setup.bash
export PX4_DIR=<your px4 dir>
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:${PX4_DIR}:${PX4_DIR}/Tools/simulation/gazebo-classic/sitl_gazebo-classic
source ${PX4_DIR}/Tools/simulation/gazebo-classic/setup_gazebo.bash ${PX4_DIR} ${PX4_DIR}/build/px4_sitl_default

# Run with ROS
roslaunch px4 posix_sitl.launch
roslaunch mavros px4.launch
```

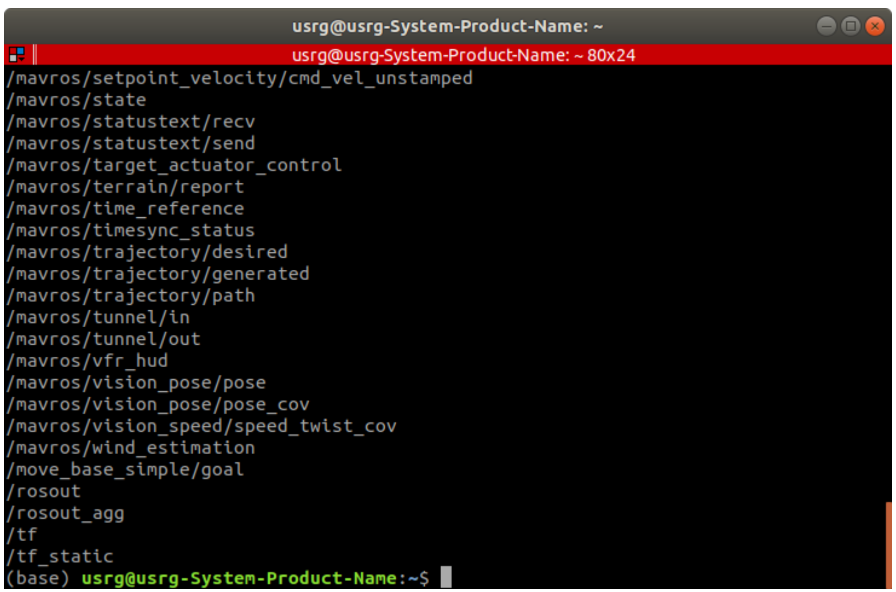
- Reference
- https://docs.px4.io/main/en/simulation/ros_interface.html

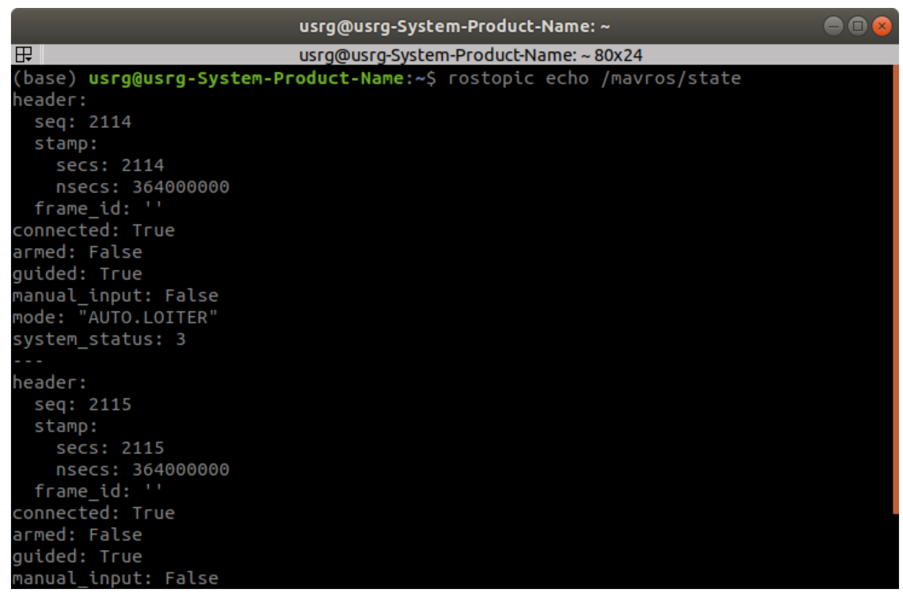
Install Gazebo Simulator

- Run PX4 simulator with MAVROS
  - To check whether it works, you should check two things
  - 1. rostopic list

    You should see the list of the topics from mavros
  - 2. rostopic echo /mavros/state

    You should see that the topic is actually publishing



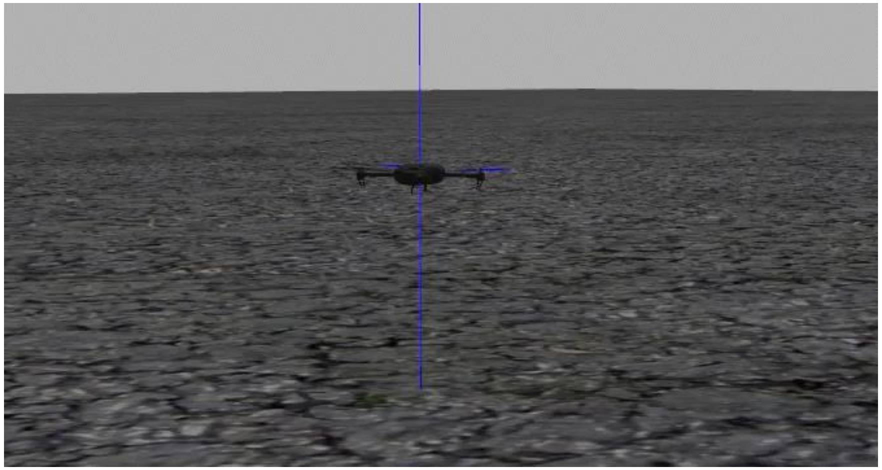


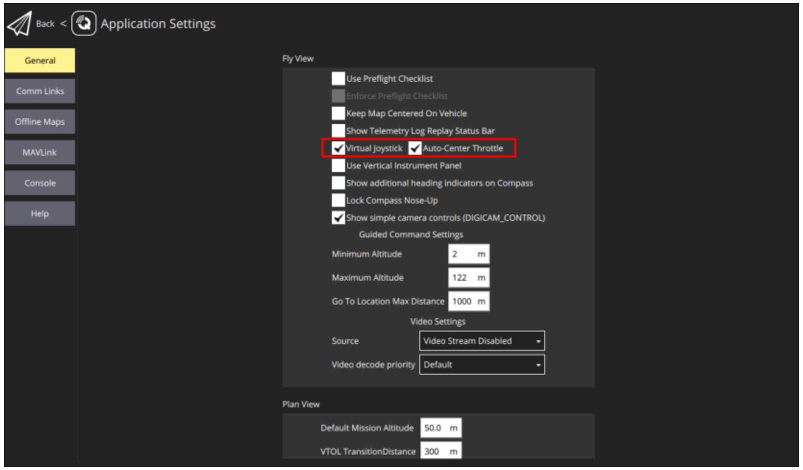
Control Drone in Gazebo

MAVROS Controller Interface

- Position controller
  - We can send position topics to the mavros node to make the drone fly
  - **Position topic**
  - /mavros/setpoint_position/local
    Send x,y,z position to the drone
  - The drone can fly by using simple rostopic pub command.
  - Topics should be published with fps larger than 2Hz

```
rostopic pub –r 20 /mavros/setpoint_position/local geometry_msgs/PoseStamped (skip)
```





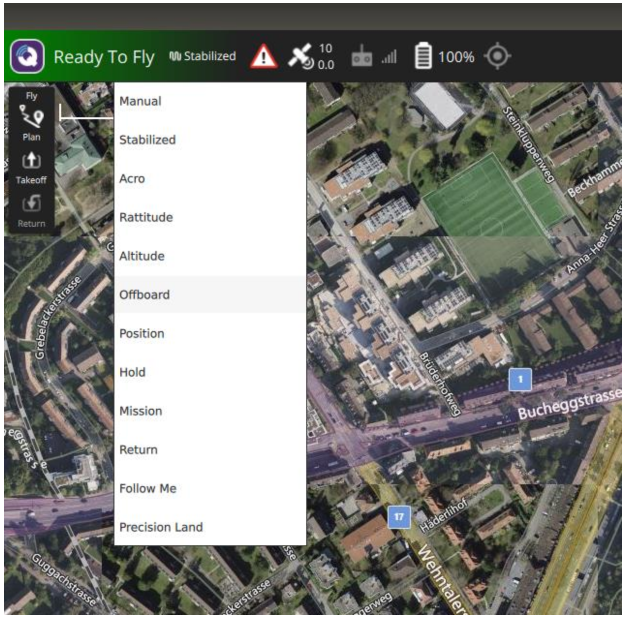
- Please run the QGroundControl, and set "Virtual Joystick".
- Reference : https://docs.qgroundcontrol.com/master/en/SettingsView/VirtualJoystick.html

MAVROS Controller Interface

- Position controller
  - After publishing topics, you should turn on the QGroundControl and manually change mode and arming.

```
# in the directory QGroundControl is installed
./QGroundControl.AppImage
```

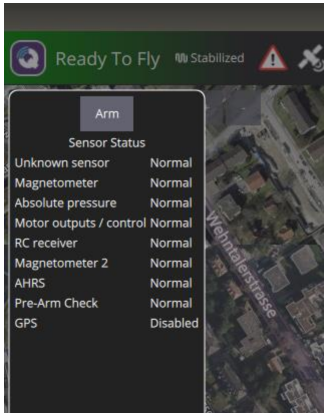
  - Change Mode
    Click "Stabilized" and Select "Offboard" Mode
    You should change this **after publishing the topic** since offboard mode requires topics which are published already!

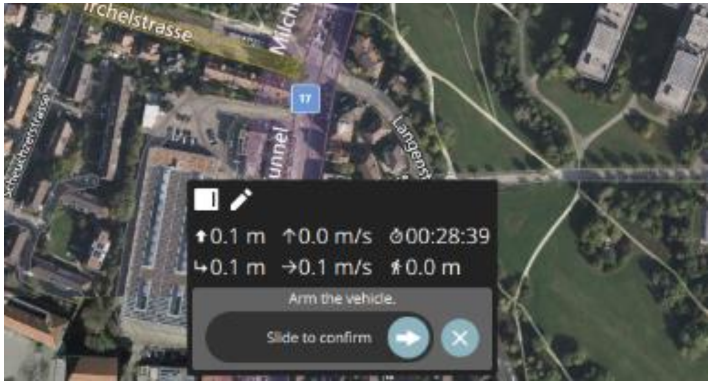


MAVROS Controller Interface

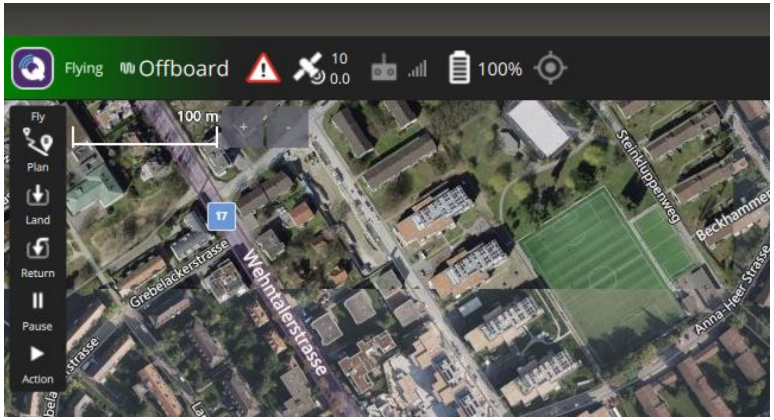
- Position controller
  - Arming

    Click “Ready To Fly” Button, and then click “Arm” Button.
    You can see Slide to confirm button, and slide the button to arm the drone.





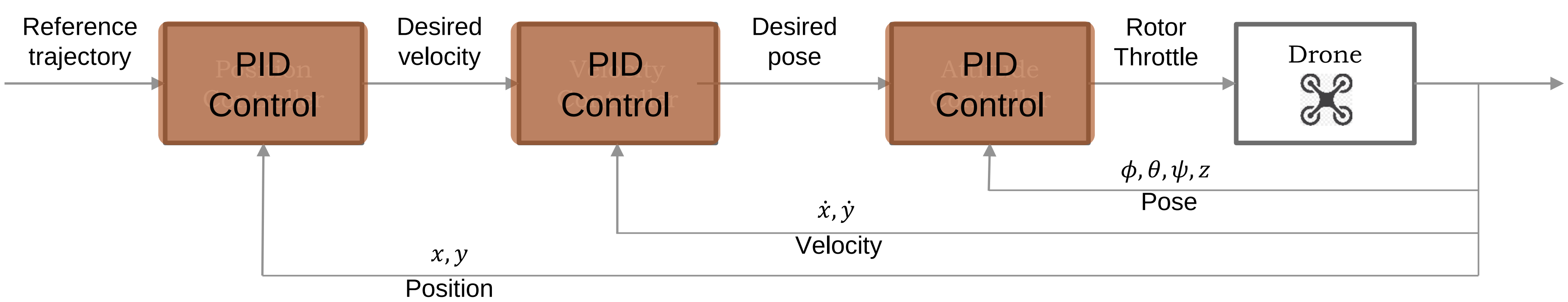
  - Result window



MAVROS Controller Interface

❖ Velocity controller
  - **Velocity topic**
  - /mavros/setpoint_position/local
    Send vx,vy,vz position and yaw direction to the drone

  - You should design your own PID Controller to make the drone fly



Overall cascade drone control architecture

MAVROS Controller Interface

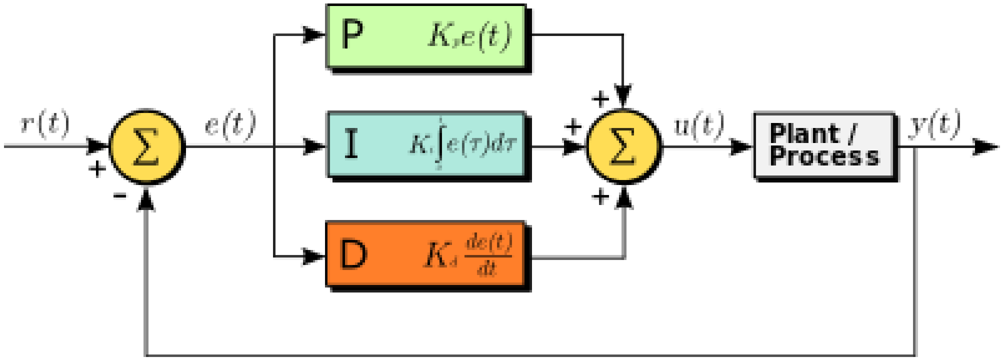
- MAVROS has lots of useful topics.
  - Subscribe
    /mavros/setpoint_position/local
    /mavros/setpoint_velocity/cmd_vel
  - Publish
    /mavros/state
    /mavros/local_position/pose
    /mavros/local_position/odom

- Reference
- http://wiki.ros.org/mavros

PID Controller

❖ Proportional-Integral-Derivative Controller (PID Controller)

- PID Controller consists of **three terms**: proportional(P), integral(I) and derivative(D) term.
- Each term has a control gain: K_P gain, K_I gain, K_D gain.
- **P-term** is proportional to the error, $r(t) - y(t)$.
- **I-term** accounts for past error values and integrates them over time.
- **D-term** estimates the future trend of the error, based on its current rate of change.



$$u = K_P(v_d - v) + K_I \int_0^t (v_d - v)dt + K_D \frac{d(v_d - v)}{dt}$$

MAVROS velocity controller

❖ Code Explanation

The full code will be uploaded in the KLMS.

This part receives current position and prints it to the terminal

```
def pose_callback(msg):
    global current_pose
    current_pose = msg
    print("Pose Received")
    print("X : "+str(current_pose.pose.position.x)+", Y : "+str(current_pose.pose.position.y)+", Z : "+str(current_pose.pose.position.z))
```

Defines subscriber and publisher for current position and velocity

```
state_sub = rospy.Subscriber("mavros/local_position/pose", PoseStamped, callback = pose_callback)
local_vel_pub = rospy.Publisher("mavros/setpoint_velocity/cmd_vel", TwistStamped, queue_size=10)
```

In the main loop, you can calculate velocity command and publish it.

```
# main loop
while(not rospy.is_shutdown()):
    ##################################################
    # PID Controller
    cmd_velocity.twist.linear.x = 
    cmd_velocity.twist.linear.y = 
    cmd_velocity.twist.linear.z = 
    ##################################################

    local_vel_pub.publish(cmd_velocity)
    rate.sleep()
```

Assignment 1

Programming Assignment

- 1. Hover using Position controller by publishing topic in command line
  - Install PX4 simulation, MAVROS, and QGroundControl
  - Hover the drone using "rostopic pub" command

```
Ex) rostopic pub -r 20 /mavros/setpoint_position/local geometry_msgs/PoseStamped (skip)
```

  - Position should be (0,0,2), and rate should be 30Hz
  - Please submit

  1) terminal command you used
  2) image that drone is flying in the gazebo

Programming Assignment

❖ 2. Design PID Controller based on the example code.

❑ Everything is implemented except the PID controller part.

```
    ##################################################
    # PID Controller
    cmd_velocity.twist.linear.x = 
    cmd_velocity.twist.linear.y = 
    cmd_velocity.twist.linear.z = 
    ##################################################
```

❑ Write your own code to design.(You can use only P gain if you want)

❑ You can freely decide your goal point, fps, etc.

❑ Please submit

1) Source code
2) Screencapture of the terminal after entering the command "rostopic echo /mavros/setpoint_velocity/cmd_vel"

Programming Assignment

- 3. Implement waypoint tracking code
  - Make the drone fly along with (0,0,1), (1,0,1), (1,1,1), (0,1,0), and (0,0,1)
  - The trajectory will be similar with square.
  - You can modify the released controller code again, or make another code that will work based on the position topic.

  - Please submit

  1) Source code
  2) Rviz capture of the topic /mavros/local_position/odom

Programming Assignment

- Assignment should include
  - 1 PDF report that explains your implementation
  - Requirements of problem 1,2, and 3

Please submit the assignment in KLMS as a zip file (PDF + Requirements).
Name : student number_name_HW2.zip
Ex. 20250000_GildongHong_HW1.zip
Due date: 2025-03-28 (23:59, Friday)

Programming Assignment

- Notice
  - For simulation assignment, you can use N5 2354 experiment room.
  - Avaliable from Wednesday and Friday(13:00~20:00)
  - Please organize the equipment and components properly, and no food or drinks are allowed.
  - Send an email one of the TAs, if you want to use.

Q & A