Nama : Gusti Gratia Delpiera

NRP : 5026231097

Laporan Penugasan PX4 Simulation (Tugas 3)

1. Langkah pertama setelah membuat package dan menambahkan dependencies serta executable yang diperlukan pada file Cmakelist.txt dan package.xml adalah membuat file CPP di dalam folder src. Menambahkan header yang diperlukan, untuk membuat pola yang ditentukan disini saya menggunakan rclcpp, offboard_control_mode, trajectory_setpoint, vehicle_command, untuk include dari px4_msgs dan beberapa header dari standard library seperti iostream, chrono, dan cmath.

```
#include "rclcpp/rclcpp.hpp"
#include "px4_msgs/msg/vehicle_command.hpp"
#include "px4_msgs/msg/trajectory_setpoint.hpp"
#include "px4_msgs/msg/offboard_control_mode.hpp"
#include "px4_msgs/msg/vehicle_control_mode.hpp"
#include <chrono>
#include <iostream>
#include <cmath>
```

2. Langkah kedua membuat class Node, yang saya beri nama star_shape, untuk mempublish offboard control drone.

```
lass OffboardControl : public rclcpp::Node {

public:
  OffboardControl() : Node("star_shape") {

      };
  void arm();

  void land();
  void disarm();
private:
  void publish_offboard_control_mode();
  void publish_trajectory_setpoint(float x, float y, float z, float yaw);
  void publish_vehicle_command(uint16_t command, float param1 = 0.0, float
param2 = 0.0);

  };
```

Disini, saya membuat prototype dari empat buah method untuk public, yaitu untuk land, arm, dan disarm. Kemudian tiga buah method private untuk publish offboard control mode, trajectory setpoint, dan vehicle command (arm/disarm/land).

Kemudian saya menginisialisasi untuk publisher dengan create_publisher pada Node, dan juga offboard setpoint counter, sebagai counter untuk offboard mode.

```
trajectory_setpoint_publisher_ = this-
>create_publisher<px4_msgs::msg::TrajectorySetpoint>("fmu/in/trajectory_setpoi
nt", 10);
offboard_control_mode_publisher_ = this-
>create_publisher<px4_msgs::msg::OffboardControlMode>("fmu/in/offboard_control
_mode", 10);
vehicle_command_publisher_ = this-
>create_publisher<VehicleCommand>("/fmu/in/vehicle_command", 10);
```

Pada bagian private saya juga mendeklarasikan publisher dalam framework ROS2,

```
    rclcpp::Publisher<px4_msgs::msg::TrajectorySetpoint>::SharedPtr
trajectory_setpoint_publisher_;
    rclcpp::Publisher<px4_msgs::msg::OffboardControlMode>::SharedPtr
offboard_control_mode_publisher_;
    rclcpp::Publisher<VehicleCommand>::SharedPtr
vehicle_command_publisher_;//t
    rclcpp::TimerBase::SharedPtr timer_;
```

3. Sebelum masuk ke dalam lambda function, saya terlebih dahulu untuk membuat method arm, disarm, land, publish vehicle command, trajectory setpoint, dan offboard control mode.

```
void OffboardControl::arm()
{
  publish_vehicle_command(VehicleCommand::VEHICLE_CMD_COMPONENT_ARM_DISARM,
1.0);
  RCLCPP_INFO(this->get_logger(), "Arm command send");
}

void OffboardControl::land(){
  publish_vehicle_command(VehicleCommand::VEHICLE_CMD_NAV_VTOL_LAND, 0.0);
  RCLCPP_INFO(this->get_logger(), "Land command send");
}

void OffboardControl::disarm()
{
  publish_vehicle_command(VehicleCommand::VEHICLE_CMD_COMPONENT_ARM_DISARM,
0.0);
  RCLCPP_INFO(this->get_logger(), "Disarm command send");
}
```

Untuk tiga method disini (arm, disarm, land), saya menggunakan vehicle command yang tersedia pada px4_msgs, yang menggunakan mavlink messages sebagai protocol komunikasi.

```
void OffboardControl::publish_offboard_control_mode()
```

```
{
  OffboardControlMode msg{};
  msg.position = true;
  msg.velocity = false;
  msg.acceleration = false;
  msg.attitude = false;
  msg.body_rate = false;
  msg.timestamp = this->get_clock()->now().nanoseconds() / 1000;
  offboard_control_mode_publisher_->publish(msg);
}
```

Dan saya membuat publisher untuk offboard control, yang bernilai true untuk msg adalah msg.position. Karena kita akan mengubah posisi dari UAV.

```
void OffboardControl::publish_trajectory_setpoint(float x, float y, float z,
float yaw)
{
  TrajectorySetpoint msg{};
  msg.position = {x,y,z};
  msg.yaw = yaw;
  msg.velocity = {};
  msg.acceleration = {};
  msg.timestamp = this->get_clock()->now().nanoseconds() / 1000;
  trajectory_setpoint_publisher_->publish(msg);
}
```

Kemudian untuk publisher trajectory setpoint, saya mengambil 4 parameter, yaitu koordinat x,y,z, dan besar sudut yaw.

```
void OffboardControl::publish_vehicle_command(uint16_t command, float param1,
float param2)
{
  VehicleCommand msg{};
  msg.param1 = param1;
  msg.param2 = param2;
  msg.command = command;
  msg.target_system = 1;
  msg.target_component = 1;
  msg.source_system = 1;
  msg.source_component = 1;
  msg.from_external = true;
  msg.timestamp = this->get_clock()->now().nanoseconds() / 1000;
  vehicle_command_publisher_->publish(msg);
}
```

Selanjutnya publish vehicle command untuk mempublish vehicle cmd pada method2 sebelumnya seperti VEHICLE_CMD_COMPONENT_ARM_DISARM, dan VEHICLE_CMD_NAV_VTOL_LAND.

4. Pada main program, digunakan untuk spin Node Offboard Control.

```
int main(int argc, char *argv[]) {
  std::cout << "Starting offboard control node..." << std::endl;
  setvbuf(stdout, NULL, _IONBF, BUFSIZ);
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<OffboardControl>());

  rclcpp::shutdown();
  return 0;
}
```

5. Sebelum membuat lambda function, saya membuat function untuk mengonversi derajat ke radian.

```
float rad(int degree){
  float result = M_PI * degree / 180;
  return result;
}
```

6. Terakhir untuk lambda function (timer_callback) :

```
offboard_setpoint_counter = 0;
      auto timer_callback = [this]() -> void {
        if(offboard_setpoint_counter==10){
          this-
>publish_vehicle_command(VehicleCommand::VEHICLE_CMD_DO_SET_MODE, 1, 6);
          this->arm();
        }

        if(offboard_setpoint_counter < 100){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(0.0,0.0,-5.0,0.0);
        }
        else if(offboard_setpoint_counter >= 100 && offboard_setpoint_counter
< 150){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(0.0,0.0,-5.0,-rad(18));
        }
        else if(offboard_setpoint_counter >= 150 && offboard_setpoint_counter
< 220){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(5.4,-1.8,-5.0,-rad(18));
        }
```

```
        else if(offboard_setpoint_counter >= 220 && offboard_setpoint_counter
< 250){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(5.4,-1.8,-5.0,rad(144));
        }
        else if(offboard_setpoint_counter >= 250 && offboard_setpoint_counter
< 320){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(2.0,2.8,-5.0,rad(144));
        }
        else if(offboard_setpoint_counter >= 320 && offboard_setpoint_counter
< 350){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(2.0,2.8,-5.0,-rad(90));
        }
        else if(offboard_setpoint_counter >= 350 && offboard_setpoint_counter
< 420){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(2.0,-2.8,-5.0,-rad(90));
        }
        else if(offboard_setpoint_counter >= 420 && offboard_setpoint_counter
< 450){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(2.0,-2.8,-5.0,rad(36));
        }
        else if(offboard_setpoint_counter >= 450 && offboard_setpoint_counter
< 520){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(5.4,1.8,-5.0,rad(36));
        }
        else if(offboard_setpoint_counter >= 520 && offboard_setpoint_counter
< 550){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(5.4,1.8,-5.0,-rad(144));
        }
        else if(offboard_setpoint_counter >= 550 && offboard_setpoint_counter
< 620){
          publish_offboard_control_mode();
          publish_trajectory_setpoint(0.0,0.0,-5.0,-rad(144));
        }
        else if (offboard_setpoint_counter == 630){
          this->land();
        }
        else if (offboard_setpoint_counter == 750){
          this->disarm();
        }
        if(offboard_setpoint_counter < 751){
          offboard_setpoint_counter++;
```

```
    }
};

timer_ = this->create_wall_timer(100ms, timer_callback);
```

a. Saya menginisiasi untuk setpoint counter bernilai 0 dan akan terus bertambah sebelum setpoint counter bernilai 750 (75 seconds).
b. Ketika setpoint counter bernilai 10 (detik pertama) maka publish set mode untuk offboard control, dan arm vehicle.
c. Drone takeoff dengan trajectory setpoint, dengan nilai z = -5.0 (5 m) selama 10 detik di titik (0,0) kemudian yaw ke arah -18 derajat.
d. Drone bergerak maju ke koordinat (5.4,-1.8) kemudian yaw ke arah 144 derajat.
e. Drone bergerak maju ke koordinat (2,2.8) kemudian yaw ke arah -90 derajat.
f. Drone bergerak maju ke koordinat (2,-2.8) kemudian yaw ke arah 36 derajat.
g. Drone bergerak maju ke koordinat (5.4,1.8) kemudian yaw ke arah -144 derajat.
h. Drone bergerak maju ke koordinat awal (0,0).
i. Drone mendarat, dan disarm.

Dengan demikian drone membentuk pola seperti Bintang, di sini saya menghitung untuk besar sudut dalam Bintang adalah 36 derajat :

1. Hitung besar sudut pusat pada titik tengah bintang. Dalam bintang beraturan, sudut pusat setiap segitiga dalam bintang adalah 360 derajat dibagi dengan jumlah sisi bintang. Jadi, dalam kasus ini, sudut pusat adalah 360 derajat dibagi dengan 5, yang sama dengan 72 derajat.
2. Setiap sudut di bagian tengah segitiga sama besar, sehingga sudut pada setiap sudut bintang adalah setengah dari sudut pusat. Dalam kasus ini, itu adalah 72 derajat dibagi 2, yang sama dengan 36 derajat.

   Jadi, sudut dalam bintang tersebut adalah 36 derajat.

   sehingga dapat disesuaikan untuk yaw angle drone Ketika membentuk pola Bintang.

LINK VIDEO LAPORAN :

https://drive.google.com/file/d/1ZwFbw48Z2T0Mog6nPrFQu2_J2JY5p4OW/view?usp=sharing