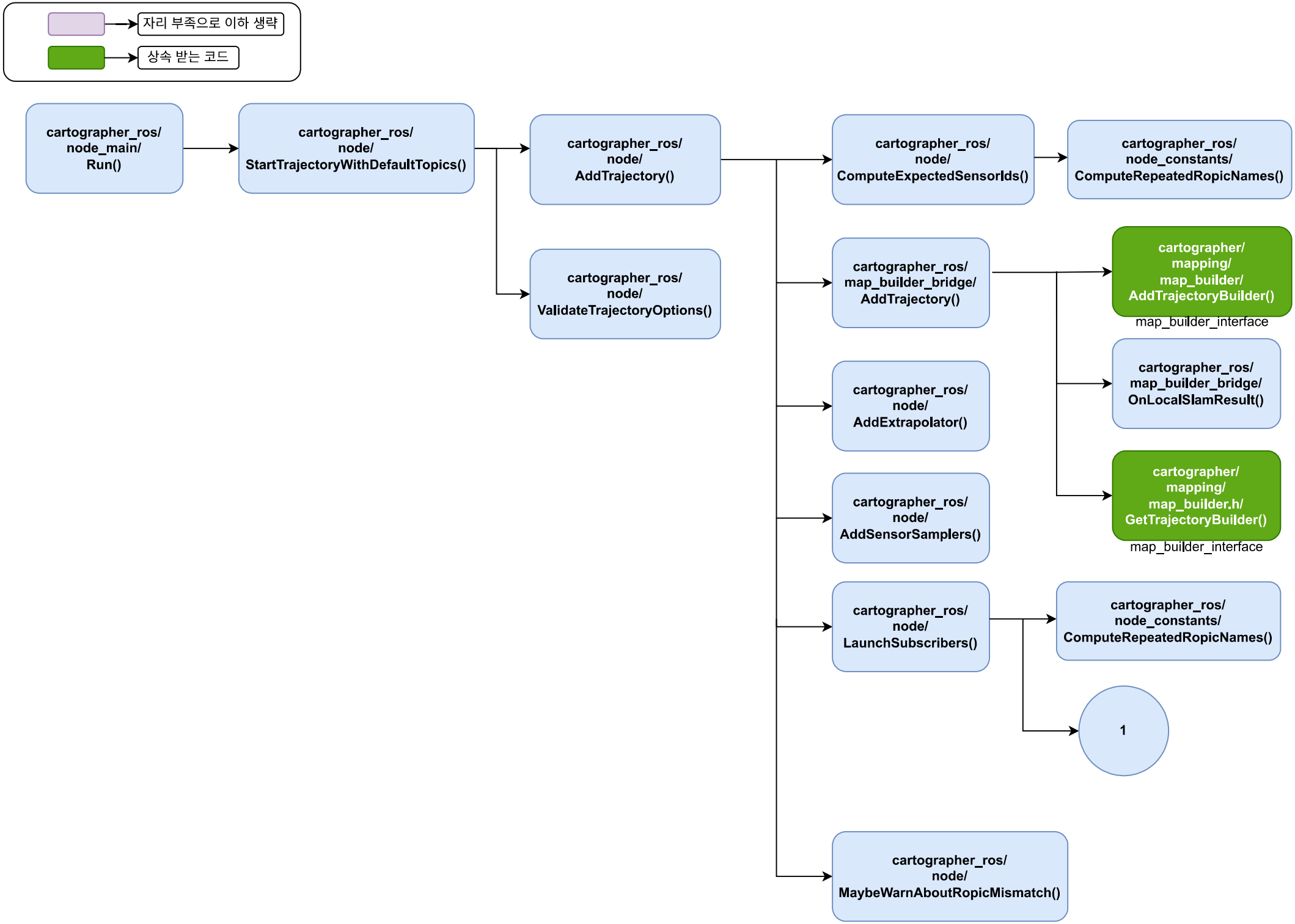
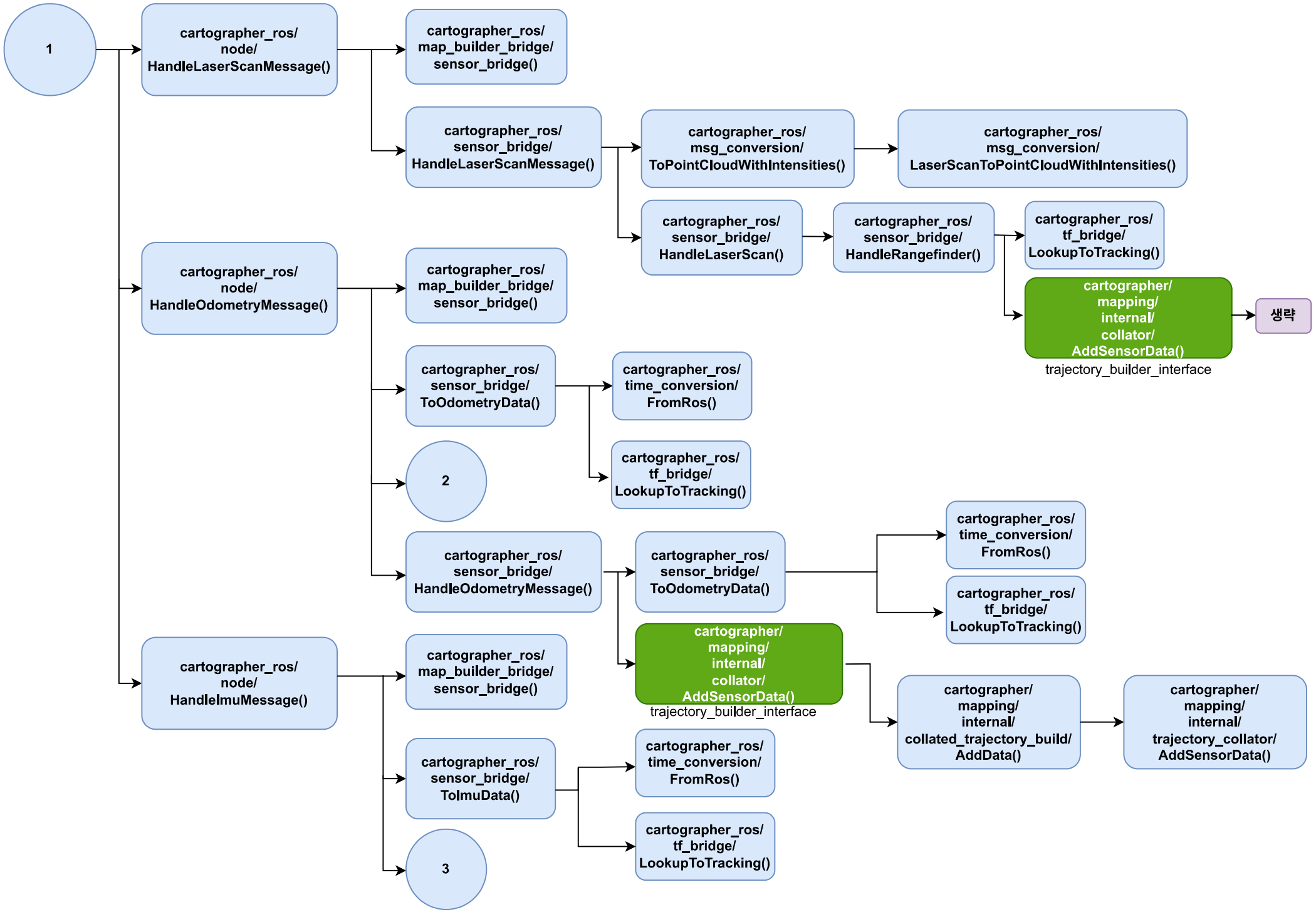
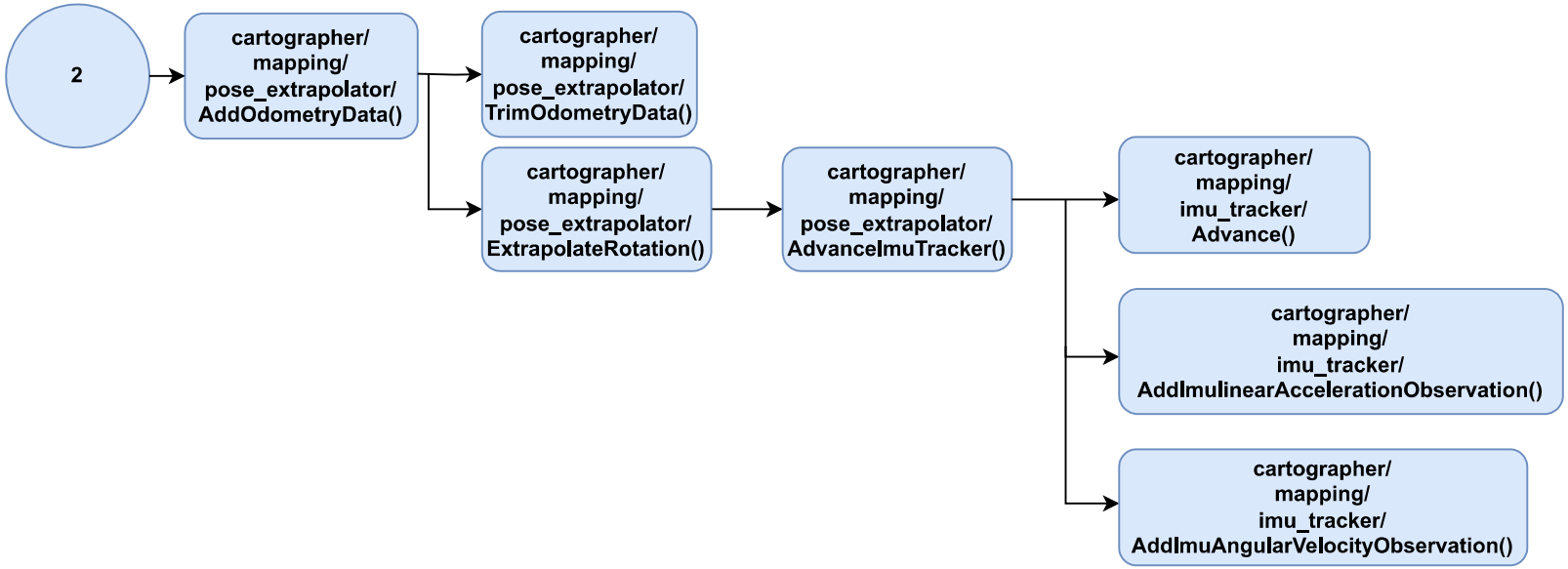
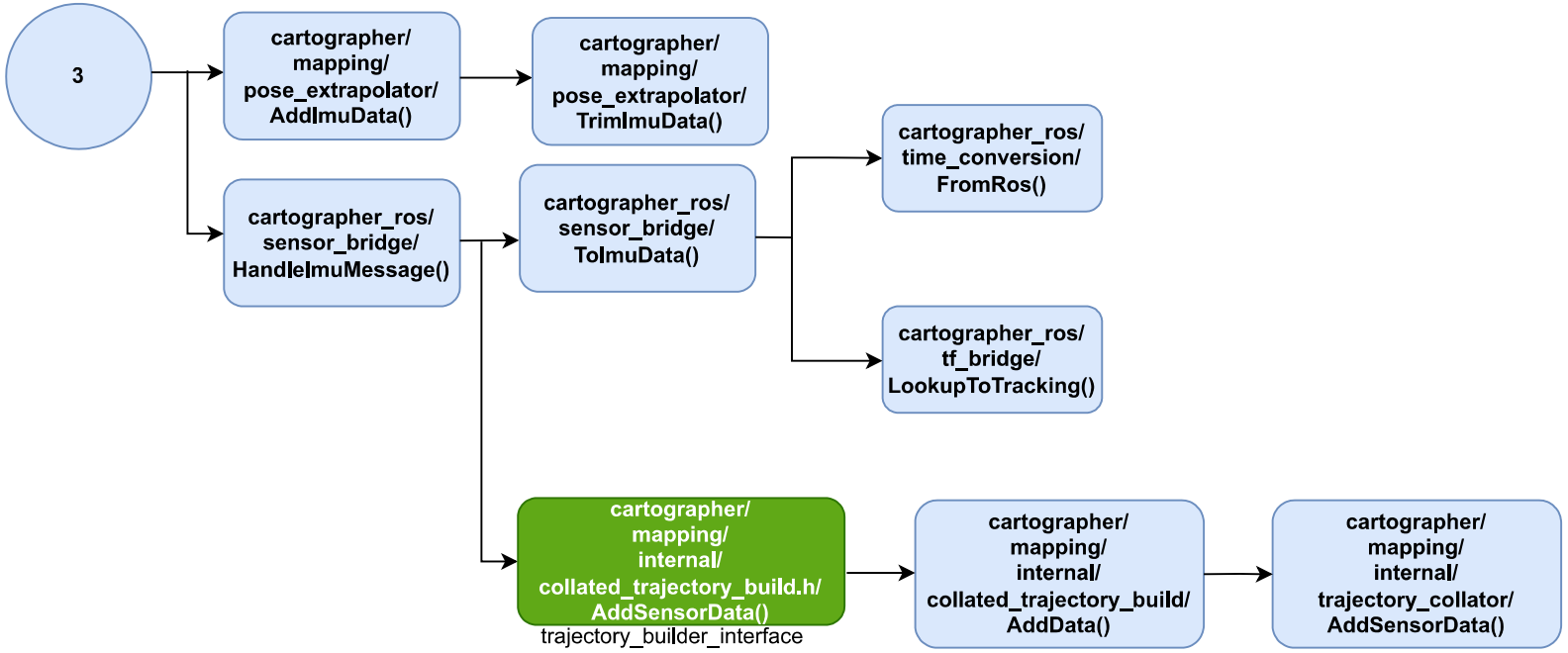

자리 부족으로 이하 생략
상속 받는 코드
cartographer_ros/
node_main/
Run()
cartographer_ros/
node/
StartTrajectoryWithDefaultTopics()
cartographer_ros/
node/
AddTrajectory()
cartographer_ros/
node/
ValidateTrajectoryOptions()
cartographer_ros/
node/
ComputeExpectedSensorIds()
cartographer_ros/
node_constants/
ComputeRepeatedRopicNames()
cartographer_ros/
map_builder_bridge/
AddTrajectory()
cartographer/
mapping/
map_builder/
AddTrajectoryBuilder()
map_builder_interface
cartographer_ros/
map_builder_bridge/
OnLocalSlamResult()
cartographer_ros/
node/
AddExtrapolator()
cartographer/
mapping/
map_builder.h/
GetTrajectoryBuilder()
map_builder_interface
cartographer_ros/
node/
AddSensorSamplers()
cartographer_ros/
node/
LaunchSubscribers()
cartographer_ros/
node_constants/
ComputeRepeatedRopicNames()
1
cartographer_ros/
node/
MaybeWarnAboutRopicMismatch()


1
cartographer_ros/ node/ HandleLaserScanMessage()
cartographer_ros/ map_builder_bridge/ sensor_bridge()
cartographer_ros/ sensor_bridge/ HandleLaserScanMessage()
cartographer_ros/ msg_conversion/ ToPointCloudWithIntensities()
cartographer_ros/ msg_conversion/ LaserScanToPointCloudWithIntensities()
cartographer_ros/ sensor_bridge/ HandleLaserScan()
cartographer_ros/ sensor_bridge/ HandleRangefinder()
cartographer_ros/ tf_bridge/ LookupToTracking()
cartographer/ mapping/ internal/ collator/ AddSensorData()
trajectory_builder_interface
생략
cartographer_ros/ node/ HandleOdometryMessage()
cartographer_ros/ map_builder_bridge/ sensor_bridge()
cartographer_ros/ sensor_bridge/ ToOdometryData()
cartographer_ros/ time_conversion/ FromRos()
cartographer_ros/ tf_bridge/ LookupToTracking()
2
cartographer_ros/ sensor_bridge/ HandleOdometryMessage()
cartographer_ros/ sensor_bridge/ ToOdometryData()
cartographer_ros/ time_conversion/ FromRos()
cartographer_ros/ tf_bridge/ LookupToTracking()
cartographer/ mapping/ internal/ collator/ AddSensorData()
trajectory_builder_interface
cartographer/ mapping/ internal/ collated_trajectory_build/ AddData()
cartographer/ mapping/ internal/ trajectory_collator/ AddSensorData()
cartographer_ros/ node/ HandleImuMessage()
cartographer_ros/ map_builder_bridge/ sensor_bridge()
cartographer_ros/ sensor_bridge/ ToImuData()
cartographer_ros/ time_conversion/ FromRos()
cartographer_ros/ tf_bridge/ LookupToTracking()
3


2
cartographer/
mapping/
pose_extrapolator/
AddOdometryData()
cartographer/
mapping/
pose_extrapolator/
TrimOdometryData()
cartographer/
mapping/
pose_extrapolator/
ExtrapolateRotation()
cartographer/
mapping/
pose_extrapolator/
AdvanceImuTracker()
cartographer/
mapping/
imu_tracker/
Advance()
cartographer/
mapping/
imu_tracker/
AddImulinearAccelerationObservation()
cartographer/
mapping/
imu_tracker/
AddImuAngularVelocityObservation()


3
cartographer/
mapping/
pose_extrapolator/
AddImuData()
cartographer/
mapping/
pose_extrapolator/
TrimImuData()
cartographer_ros/
sensor_bridge/
HandleImuMessage()
cartographer_ros/
sensor_bridge/
ToImuData()
cartographer_ros/
time_conversion/
FromRos()
cartographer_ros/
tf_bridge/
LookupToTracking()
cartographer/
mapping/
internal/
collated_trajectory_build.h/
AddSensorData()
trajectory_builder_interface
cartographer/
mapping/
internal/
collated_trajectory_build/
AddData()
cartographer/
mapping/
internal/
trajectory_collator/
AddSensorData()

포인터에 값을 저장하는 경우

포인터로 부터 값을 가져오는 경우

<node.cpp>

cartographer_ros/
node/
AddSensorSamplers()

→ sensor_samplers_

```
sensor_samplers_.emplace(
    std::piecewise_construct, std::forward_as_tuple(trajectory_id),
    std::forward_as_tuple(
        options.rangefinder_sampling_ratio, options.odometry_sampling_ratio,
        options.fixed_frame_pose_sampling_ratio, options.imu_sampling_ratio,
        options.landmarks_sampling_ratio));
```

cartographer_ros/
node/
LaunchSubscriber()

→ subscriber_

```
subscribers_[trajectory_id].push_back(
    {SubscribeWithHandler<sensor_msgs::msg::LaserScan>(
        &Node::HandleLaserScanMessage, trajectory_id, topic, node_, this),
     topic}); //LM scan sub 하는 코드
```

cartographer_ros/
node/
AddExtrapolator()

→ extrapolators_

```
extrapolators_.emplace(
    std::piecewise_construct, std::forward_as_tuple(trajectory_id),
    std::forward_as_tuple(
        ::cartographer::common::FromSeconds(kExtrapolationEstimationTimeSec),
        gravity_time_constant));
```

cartographer_ros/
node/
Node()

→ map_builder_bridge_

```
map_builder_bridge_.reset(new cartographer_ros::MapBuilderBridge(node_options_, std::move(map_builder), tf_buffer.get()));
```

<sensor_bridge.cpp>

cartographer_ros/
sensor_bridge/
HandleOdometryMessage()

→ imu_tracker_

```
trajectory_builder_->AddSensorData(
    sensor_id,
    carto::sensor::OdometryData{odometry_data->time, odometry_data->pose});
```

cartographer_ros/
sensor_bridge/
HandleImuMessage()

→ trajectory_builder_

```
trajectory_builder_->AddSensorData(
    sensor_id,
    carto::sensor::ImuData{imu_data->time, imu_data->linear_acceleration,
                           imu_data->angular_velocity});
```

cartographer_ros/
sensor_bridge/
ToOdometryData()

→ extrapolators_

```
return absl::make_unique<carto::sensor::OdometryData>(
    carto::sensor::OdometryData{
        time, ToRigid3d(msg->pose.pose) * sensor_to_tracking->inverse()});
```

cartographer_ros/
sensor_bridge/
ToImuData()

→ extrapolators_

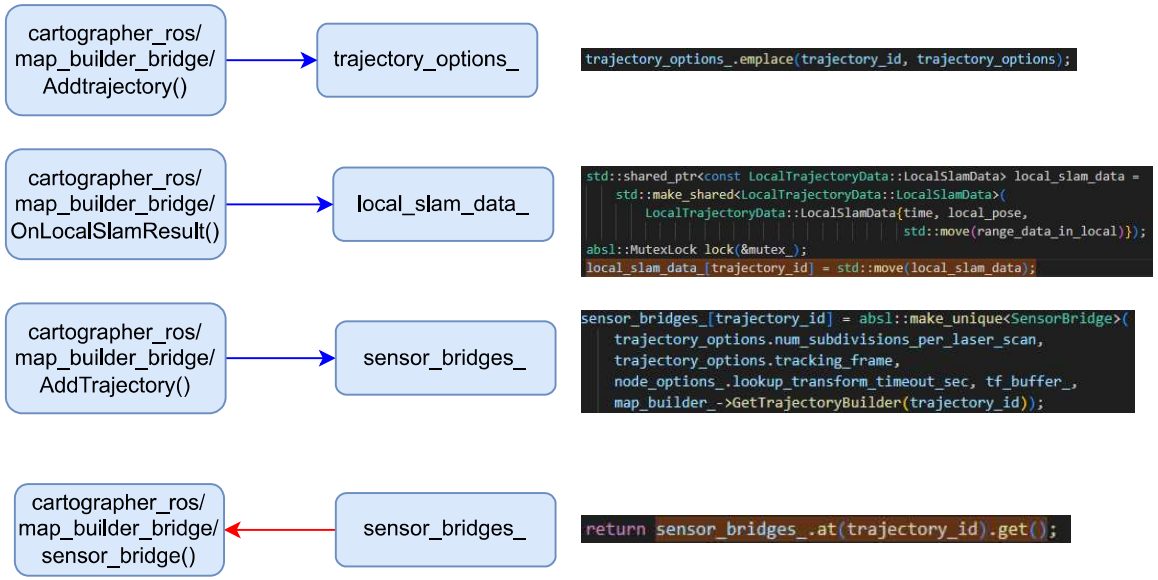
```
return absl::make_unique<carto::sensor::ImuData>(carto::sensor::ImuData{
    time, sensor_to_tracking->rotation() * ToEigen(msg->linear_acceleration),
    sensor_to_tracking->rotation() * ToEigen(msg->angular_velocity)});
```

cartographer_ros/
sensor_bridge/
HandleRangefinder()

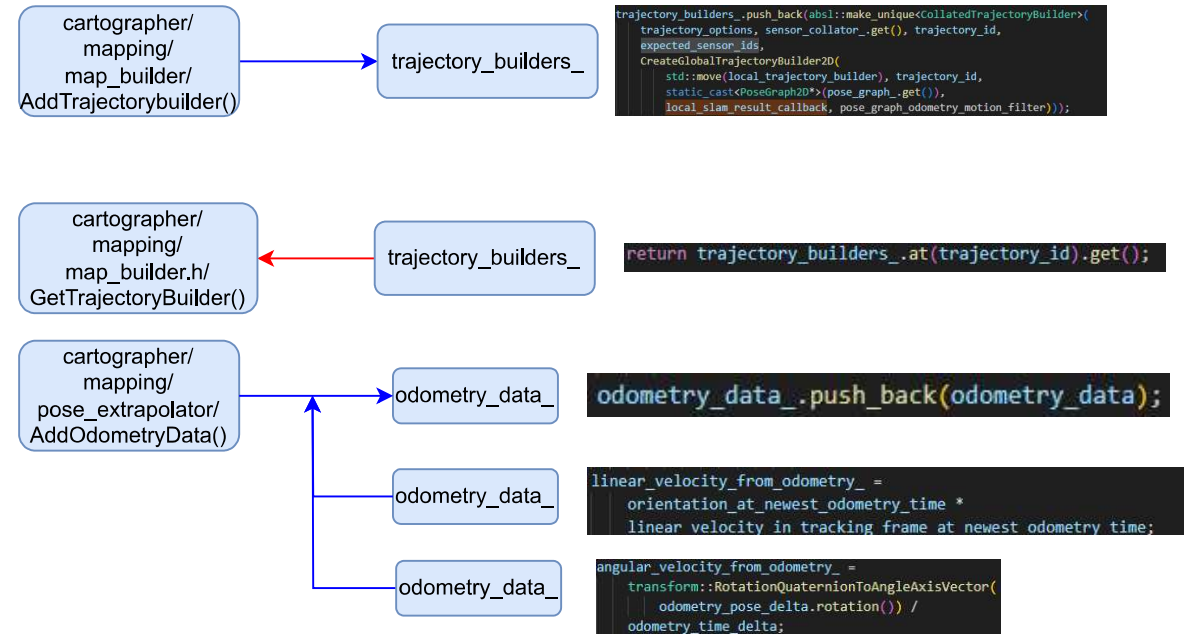
→ trajectory_builder_

```
trajectory_builder_->AddSensorData(
    sensor_id, carto::sensor::TimedPointCloudData{
                   time, sensor_to_tracking->translation().cast<float>(),
                   carto::sensor::TransformTimedPointCloud(
                       ranges, sensor_to_tracking->cast<float>()), intensities_});
```

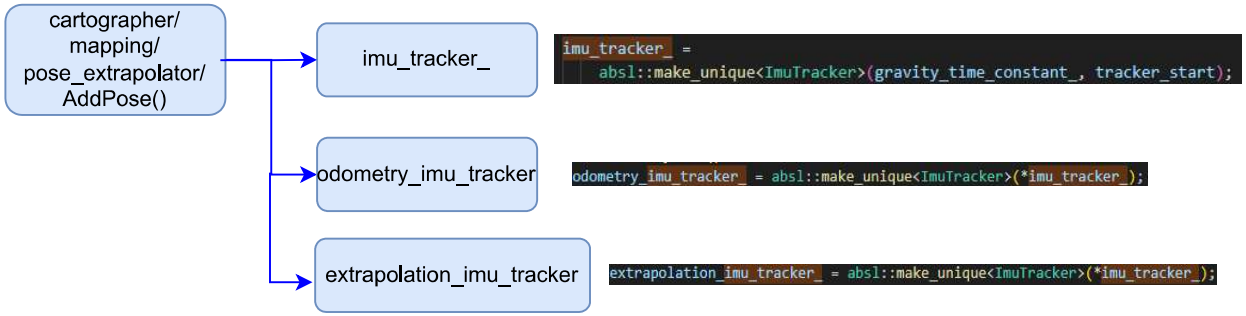
<map_builder_bridge.cpp>

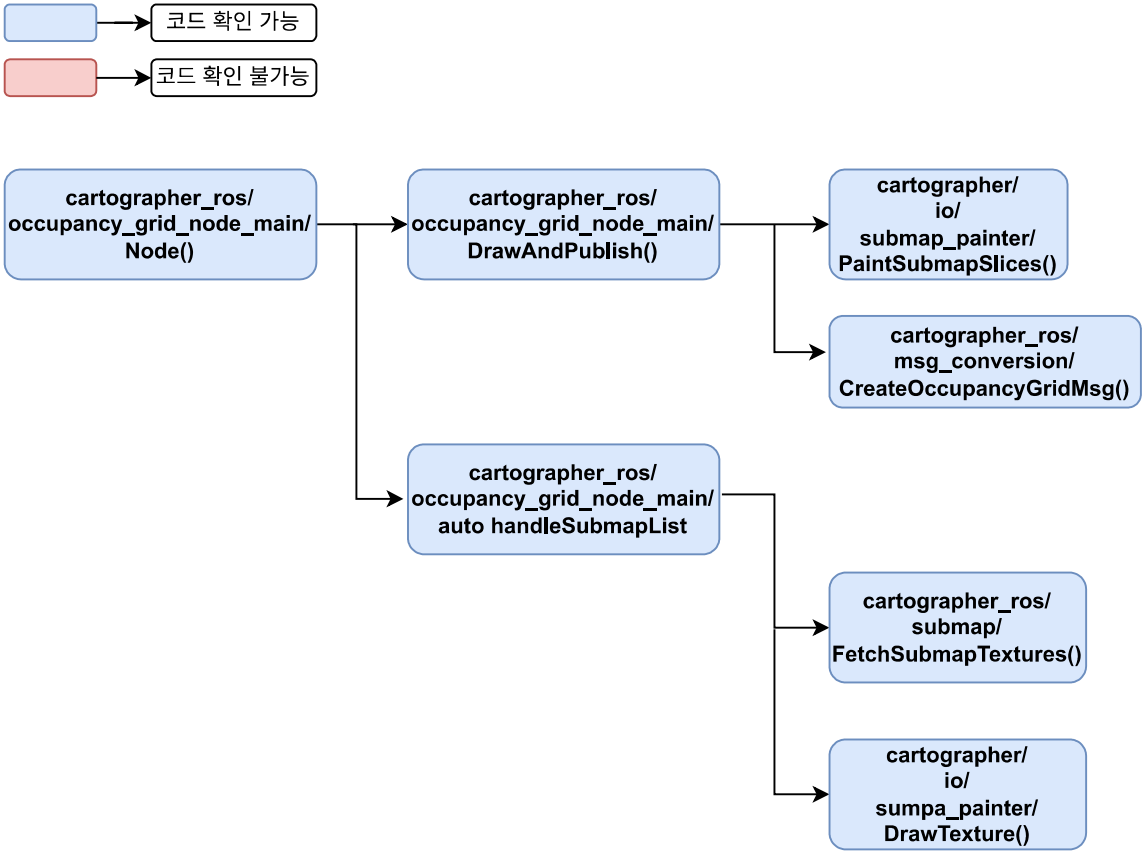


<map_builder.cc, .h>



<pose_extrapolator.cc>




코드 확인 가능
코드 확인 불가능
cartographer_ros/
occupancy_grid_node_main/
Node()
cartographer_ros/
occupancy_grid_node_main/
DrawAndPublish()
cartographer/
io/
submap_painter/
PaintSubmapSlices()
cartographer_ros/
msg_conversion/
CreateOccupancyGridMsg()
cartographer_ros/
occupancy_grid_node_main/
auto handleSubmapList
cartographer_ros/
submap/
FetchSubmapTextures()
cartographer/
io/
sumpa_painter/
DrawTexture()