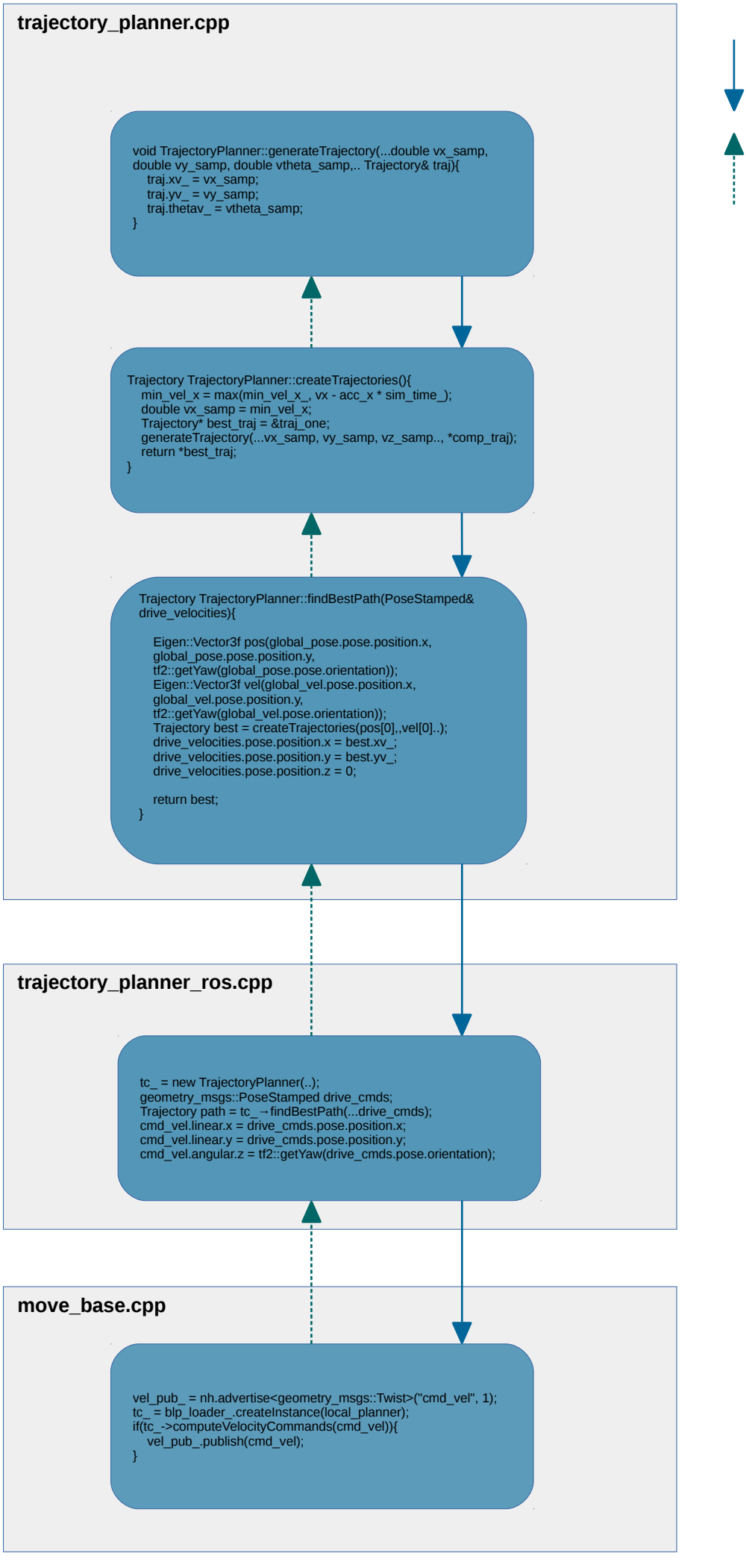
trajectory_planner.cpp
void TrajectoryPlanner::generateTrajectory(...double vx_samp,
double vy_samp, double vtheta_samp,.. Trajectory& traj){
traj.xv_ = vx_samp;
traj.yv_ = vy_samp;
traj.thetav_ = vtheta_samp;
}
Trajectory TrajectoryPlanner::createTrajectories(){
min_vel_x = max(min_vel_x_, vx - acc_x * sim_time_);
double vx_samp = min_vel_x;
Trajectory* best_traj = &traj_one;
generateTrajectory(...vx_samp, vy_samp, vz_samp.., *comp_traj);
return *best_traj;
}
Trajectory TrajectoryPlanner::findBestPath(PoseStamped&
drive_velocities){
Eigen::Vector3f pos(global_pose.pose.position.x,
global_pose.pose.position.y,
tf2::getYaw(global_pose.pose.orientation));
Eigen::Vector3f vel(global_vel.pose.position.x,
global_vel.pose.position.y,
tf2::getYaw(global_vel.pose.orientation));
Trajectory best = createTrajectories(pos[0],,vel[0]..);
drive_velocities.pose.position.x = best.xv_;
drive_velocities.pose.position.y = best.yv_;
drive_velocities.pose.position.z = 0;
return best;
}
trajectory_planner_ros.cpp
tc_ = new TrajectoryPlanner(..);
geometry_msgs::PoseStamped drive_cmds;
Trajectory path = tc_→findBestPath(...drive_cmds);
cmd_vel.linear.x = drive_cmds.pose.position.x;
cmd_vel.linear.y = drive_cmds.pose.position.y;
cmd_vel.angular.z = tf2::getYaw(drive_cmds.pose.orientation);
move_base.cpp
vel_pub_ = nh.advertise<geometry_msgs::Twist>("cmd_vel", 1);
tc_ = blp_loader_.createInstance(local_planner);
if(tc_->computeVelocityCommands(cmd_vel)){
vel_pub_.publish(cmd_vel);
}