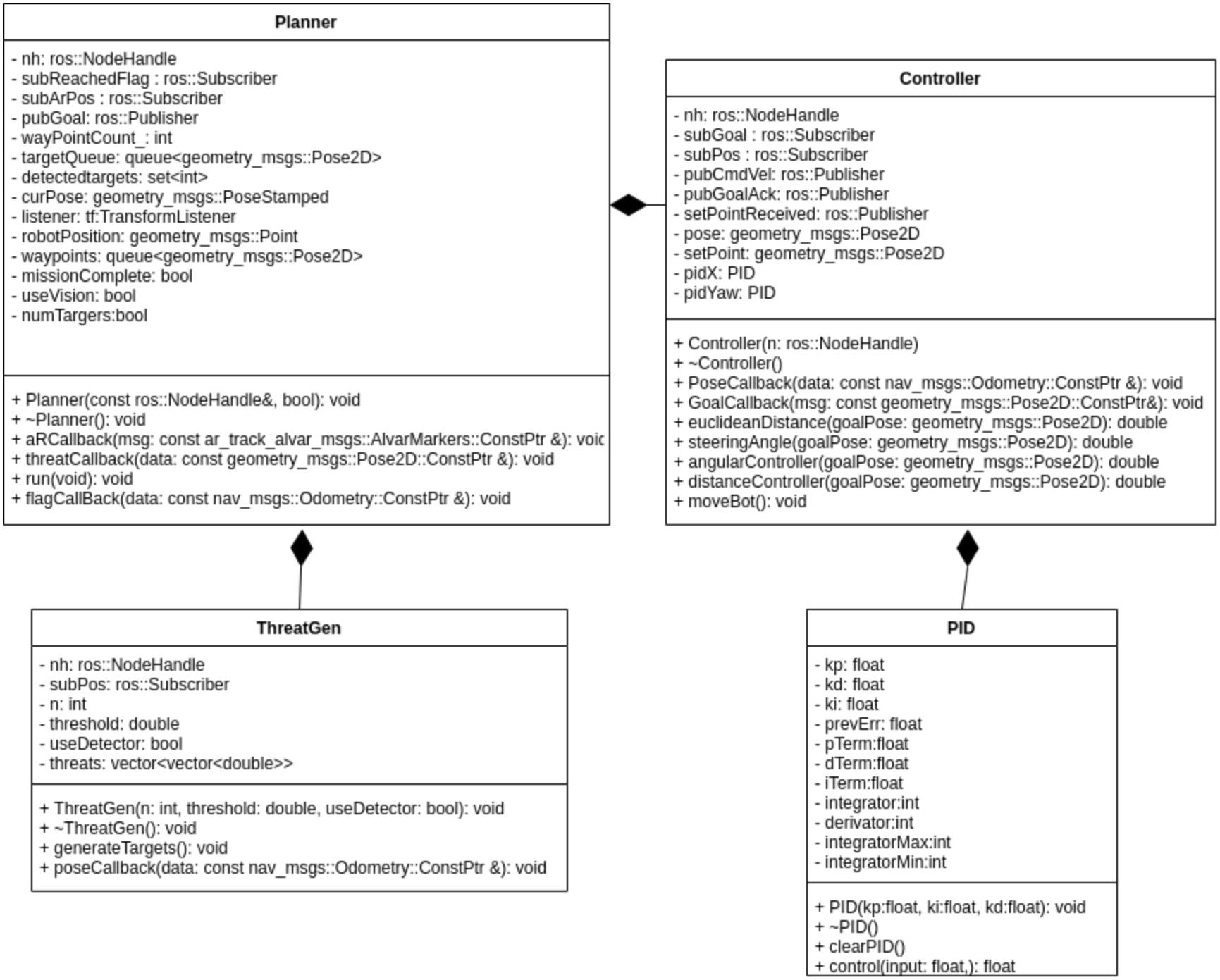
Planner
- nh: ros::NodeHandle
- subReachedFlag : ros::Subscriber
- subArPos : ros::Subscriber
- pubGoal: ros::Publisher
- wayPointCount_: int
- targetQueue: queue<geometry_msgs::Pose2D>
- detectedtargets: set<int>
- curPose: geometry_msgs::PoseStamped
- listener: tf:TransformListener
- robotPosition: geometry_msgs::Point
- waypoints: queue<geometry_msgs::Pose2D>
- missionComplete: bool
- useVision: bool
- numTargers:bool
+ Planner(const ros::NodeHandle&, bool): void
+ ~Planner(): void
+ aRCallback(msg: const ar_track_alvar_msgs::AlvarMarkers::ConstPtr &): voic
+ threatCallback(data: const geometry_msgs::Pose2D::ConstPtr &): void
+ run(void): void
+ flagCallBack(data: const nav_msgs::Odometry::ConstPtr &): void
Controller
- nh: ros::NodeHandle
- subGoal : ros::Subscriber
- subPos : ros::Subscriber
- pubCmdVel: ros::Publisher
- pubGoalAck: ros::Publisher
- setPointReceived: ros::Publisher
- pose: geometry_msgs::Pose2D
- setPoint: geometry_msgs::Pose2D
- pidX: PID
- pidYaw: PID
+ Controller(n: ros::NodeHandle)
+ ~Controller()
+ PoseCallback(data: const nav_msgs::Odometry::ConstPtr &): void
+ GoalCallback(msg: const geometry_msgs::Pose2D::ConstPtr&): void
+ euclideanDistance(goalPose: geometry_msgs::Pose2D): double
+ steeringAngle(goalPose: geometry_msgs::Pose2D): double
+ angularController(goalPose: geometry_msgs::Pose2D): double
+ distanceController(goalPose: geometry_msgs::Pose2D): double
+ moveBot(): void
ThreatGen
- nh: ros::NodeHandle
- subPos: ros::Subscriber
- n: int
- threshold: double
- useDetector: bool
- threats: vector<vector<double>>
+ ThreatGen(n: int, threshold: double, useDetector: bool): void
+ ~ThreatGen(): void
+ generateTargets(): void
+ poseCallback(data: const nav_msgs::Odometry::ConstPtr &): void
PID
- kp: float
- kd: float
- ki: float
- prevErr: float
- pTerm:float
- dTerm:float
- iTerm:float
- integrator:int
- derivator:int
- integratorMax:int
- integratorMin:int
+ PID(kp:float, ki:float, kd:float): void
+ ~PID()
+ clearPID()
+ control(input: float,): float