Report

Custom messages:

```
#goal
int32 input
nav_msgs/Path nav_path
---
#result
float32 travel_dist
---
#feedback
float32 delta_angles
```

“input” is assigned for recording LIDAR alarm, “nav_path” is to send poses;

“travel_dist” is for returning the final distance;

“delta_angles” is to compute the total angles the robot has been turned until present time;

Action_client:

I use all three goal callbacks:

the “result” gives back the final distance each time the goal accomplished.

```
void doneCb(const actionlib::SimpleClientGoalState& state,
        const rh_action_server::actmsgResultConstPtr& result) {
    ROS_INFO(" doneCb: server responded with state [%s]", state.toString().c_str());
    ROS_INFO("The final travel distance is %f",result->travel_dist);
    g_goal_active = false;
}
```

And feedback returns the accumulative angles of Gazebo:

```
void feedbackCb(const rh_action_server::actmsgFeedbackConstPtr& feedback) {
    delta_angle += feedback->delta_angles;
    ROS_INFO("Feedback: Gazebo already turned %f", delta_angle);
}
```

Active is just telling that goal is alive:

```
void activeCb()
{
  ROS_INFO("Goal is active");
  g_goal_active = true;
}
```

Then when receiving LIDAR:

```
  if (g_lidar_alarm) {
     ROS_INFO("LIDAR alarm received!");
     (*action_new).cancelGoal();
     goal.input = 1;
          }
     else{goal.input = 0;}
```

First cancel the current goal, then set the “goal.input = 1” so the server can recognize it can spin a certain angle.

Here I simply just let Gazebo to go straight then when it meets some obstacles, turn a certain angle, I guess that would be enough for Gazebo to run inside "Starting Pen" world:

```
        pose.position.x = 0.5; // say desired x-coord is 3
        pose.position.y = 0.0;
        pose.position.z = 0.0; // let's hope so!
        pose.orientation.x = 0.0; //always, for motion in horizontal plane
        pose.orientation.y = 0.0; // ditto
        pose.orientation.z = 0.0; // implies oriented at yaw=0, i.e. along x axis
        pose.orientation.w = 1.0; //sum of squares of all components of unit quaternion is 1
```

```
    pose_stamped.pose = pose;
    goal.nav_path.poses.push_back(pose_stamped);
```

Action server:

When receiving LIDAR alarms, alarm_sig will be set to 1. Let robot stop moving forward then spin 0.8 rad (or some other), at the same time give client some turning feedbacks:

```
alarm_sig = goal->input;
if(alarm_sig == 1){
  do_halt(); ///want every iter check if alarm
  do_spin(0.8);
  ROS_INFO("Gazebo is meeting something");
feedback_.delta_angles += 0.8;
as_.publishFeedback(feedback_);
  alarm_sig = 0;
}
```