src/bot/src/CPose.cpp

```cpp
#include "CPose.h"

// Implementation file for class CLidar
// Functions :
//            - Constructor
//            - Destructor
//            - Call back function sub to odom msg
//            - Publshing function

//---Constructor
CPose::CPose():nh_priv_("~")
{
  ROS_INFO("Pose node initalised");
  // Subscribe to odometry topic
  odomSub = nh_.subscribe("odom", QSize, &CPose::odomMsgCallBack, this);

  //ROS publisher to publish to a new topic
  botPub = nh_.advertise<std_msgs::Float64>(topicName,QSize);

  // Current lienar and angular velocities
  curLinVel = 0.0;
  curAngVel= 0.0;

  // Pose data from odometry
  tb3Pose= 0.0;

  // Publish pose data
  PublishPose();

  ROS_ASSERT(true);
}

//--- Destructor
CPose::~CPose()
{
  ros::shutdown;
}

//--- Call back function sub to odom msg
void CPose::odomMsgCallBack(const nav_msgs::Odometry::ConstPtr &msg)
{
  // Compute current odometry
  double siny = 2.0 * (msg->pose.pose.orientation.w * msg->
pose.pose.orientation.z + msg->pose.pose.orientation.x * msg->
pose.pose.orientation.y);
    double cosy = 1.0 - 2.0 * (msg->pose.pose.orientation.y * msg->
pose.pose.orientation.y + msg->pose.pose.orientation.z * msg->
pose.pose.orientation.z);

    tb3Pose = atan2(siny, cosy);

  // Get current Twist data
  curLinVel = msg->twist.twist.linear.x;
  curAngVel = msg->twist.twist.angular.z;
}

//---Publshing function
```

```
54 void CPose::PublishPose()
55 {
56   msg.data = tb3Pose;
57   botPub.publish(msg);
58 }
59
60 //-------------------------------------------------------------
61 // CPose NODE
62 int main(int argc, char* argv[])
63 {
64   ros::init(argc, argv, "Pose_Node");
65   CPose bot;
66   ros::Rate loop_rate(125);
67
68   while(ros::ok)
69   {
70     bot.PublishPose();
71
72     // process callback for this node
73     ros::spinOnce();
74     loop_rate.sleep();
75   }
76
77   return 0;
78 }
```