src/drive/include/TB3Drive.h

```
// Original class was turtlebot_drive taken from simulation example package
// Class TB3Drive is the main class that controls the bot based on
// lidar and odom data published by CLidar and CPose nodes.
// online- Authors: Taehun Lim (Darby). Modified by the team to fit with design

#ifndef TB3DRIVE_H_
#define TB3DRIVE_H_

#include <ros/ros.h>

#include <sensor_msgs/LaserScan.h>
#include <geometry_msgs/Twist.h>
#include <nav_msgs/Odometry.h>
#include <vector>
#include <std_msgs/Float64MultiArray.h>
#include <std_msgs/Float64.h>

#define DEG2RAD (M_PI / 180.0)
#define RAD2DEG (180.0 / M_PI)

// Lidar indexing
const int CENTER = 0;
const int LEFT   = 1;
const int RIGHT  =  2;

// Constant velocity to be used in calculating speed
const double LINEAR_VELOCITY  = 0.3;
const double ANGULAR_VELOCITY = 1.5;

// Bot states
const int GET_TB3_DIRECTION = 0;
const int TB3_DRIVE_FORWARD = 1;
const int TB3_RIGHT_TURN    = 2;
const int TB3_LEFT_TURN     = 3;
const int TB3_PID_LEFT =  4;
const int TB3_PID_RIGHT = 5;

// TB3DRive interface---------------------------------------------------------
// This class controls the robot based on the lidar readings. The class has the
// capabilities to transit states of the robot and compute linear and angular
// velocities and publish to cmd_vel to set velocity of the bot.

class TB3Drive
{
 public:
  TB3Drive();
  ~TB3Drive();
  bool controlLoop();

 private:
  // ROS NodeHandle
  ros::NodeHandle nh_;
  ros::NodeHandle nh_priv_;

  // ROS Topic Publishers
  ros::Publisher cmd_vel_pub_;
```

```cpp

  // ROS Topic Subscribers
  ros::Subscriber cLidarSub;
  ros::Subscriber cBotSub;

  // Variables - their names explain
  double escapeRange;
  double checkForwardDist;
  double checkSideDist;

  double forwardTarget;
  double forwardTargetTurn;
  double sideTarget;

  double maxTurnVel;

  double maxForwardVel;
  double minForwardVel;

  double angularVel;
  double linearVel;

  double turnKp;               // Proportional gain for angular velocity
  double forwardKp;            // Proportional gain for linear velocity

  double tb3Pose;             // Current Position form odometry - sent to by CPose
  double prevTB3pose;         // Previous Position from odometry

  int leftTurnFlag;

  std::vector<double>lidarData;

  // Function publishes to cmd_vel topic to control linear
  // and angular velocity of turtlebot.
  void updatecommandVelocity(double linear, double angular);

  // Callback functions receiving messages from CPose class and CLidar class
  void cLidarMsgCallBack(const std_msgs::Float64MultiArray::ConstPtr &msg);
  void cPoseMsgCallBack(const std_msgs::Float64::ConstPtr &msg);
};
#endif

```