src/sensor/include/CLidar.h

```
// This header file is for class CLidar which has the capability to
// read lidar data and publish it to TB3Drive class.
// The purpose of this is to isolate capability of reading in Lidar
// values from LaserScan msg so that any error in reading in lidar
// can be identified easily -- Follow OOP design

#ifndef CLIDAR_H_
#define CLIDAR_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>

const char TOPIC_NAME[] = "LIDAR";
const int LIDAR_DATA_SIZE = 3;
const int SCAN_ANGLE[] = {0, 45,315};

// Set queue size big to prevent loss if any
// delay occurs
//https://stackoverflow.com/questions/56444248/reason-to-set-queue-size-of-ros-publisher-or-subscriver-to-a-large-value
const int QSize = 1000;

/// @brief---
// CLidar interface-----------------------------------------------------
// This class is for storing the lidar data and publish to its own topic
// for drive class to listen to and get the data. It serves as a class
// that subscribes to the internal lidar scan topic and store it in a
// vector.
class CLidar{
  public:
    CLidar();
    ~CLidar();
    void LidarScanMsgCallBack(const sensor_msgs::LaserScan::ConstPtr &msg);
    void FillPublishData();

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

    // ROS Subscriber to listen in lidar
    ros::Subscriber laserScanSub;

    // ROS publisher to publish to a new topic
    ros::Publisher lidarPub;

        // Data for publishing
    std::vector<double> ScanData;
    std_msgs::Float64MultiArray msg2Pub;
};

```

```
56 #endif
```