src/sensor/src/CLidar.cpp

```cpp
#include "CLidar.h"

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

//---Constructor
CLidar::CLidar():nh_priv_("~")
{
  ROS_INFO("Lidar Node initalised");
    // Initialise subscriber
  laserScanSub = nh_.subscribe("scan", QSize, &CLidar::LidarScanMsgCallBack,
this);

  //ROS publisher to publish to a new topic
  lidarPub= nh_.advertise<std_msgs::Float64MultiArray>(TOPIC_NAME,QSize);

  // Populate Vector with default 0.0 lidar scan values
  for (int i = 0; i < LIDAR_DATA_SIZE; i++)
  {
    ScanData.push_back(0.0);
  }

  // Populate publishing message Float64MultiArray
  // https://answers.ros.org/question/226726/push-vector-into-multiarray-
message-and-publish-it/
  // set up dimensions
  msg2Pub.layout.dim.push_back(std_msgs::MultiArrayDimension());
  msg2Pub.layout.dim[0].size = ScanData.size();
  msg2Pub.layout.dim[0].stride = 1;
  msg2Pub.layout.dim[0].label = "x"; // or whatever name you typically use to
index vec1
  msg2Pub.data.clear();
  ROS_ASSERT(true);
}

//---Destructor
CLidar::~CLidar()
{
  ScanData.clear();
  ScanData.empty();
  ros::shutdown();
}

//---Call back function sub to LaserScan msg
void CLidar::LidarScanMsgCallBack(const sensor_msgs::LaserScan::ConstPtr &msg)
{
  // Read in range of lidar measurement at specified angles
  for (int num = 0; num < LIDAR_DATA_SIZE ; num++)
  {
    if (std::isinf(msg->ranges.at(SCAN_ANGLE[num])))
    {
      ScanData[num] = msg->range_max;
    }
    else
    {
      ScanData[num] = msg->ranges.at(SCAN_ANGLE[num]);
    }

    // Infinite range
    if( ScanData[num]==0.0)
    {
      ScanData[num] = msg->range_max;
    }
  }
}

//---Publshing function
void CLidar::FillPublishData()
{
  // copy in the data
  for (int i = 0; i < LIDAR_DATA_SIZE; i ++){
    msg2Pub.data.push_back(ScanData[i]);
  }

  // Publish
  lidarPub.publish(msg2Pub);

  // Clear data
  ScanData.clear();
  msg2Pub.data.clear();

}

//---------------------------------------------------------------
// CLidar NODE
int main(int argc, char* argv[])
{
  ros::init(argc, argv, "Lidar");
  CLidar Lidar;
  ros::Rate loop_rate(500);

  while(ros::ok)
  {
    Lidar.FillPublishData();

    // process callback for this node
    ros::spinOnce();
    loop_rate.sleep();
  }
  return 0;
}
```