On **Line 9**, we import the **NewPing** library that works with the HC-SR04 ultrasonic sensors. You can install the library using the **Manage Libraries** from Arduino IDE. In the search form type **newping**, and in the manager screen you will have the option to install the latest version of the library.

On **Line 10**, we import the **SimpleKalmanFilter** library to filter the ultrasonic sensor’s output. You can install the library using the **Manage Libraries** from Arduino IDE. In the search form type **simplekalman**, and in the manager screen you will have the option to install the latest version of the library.

**Lines 11-13**: we import ROS packages. **ros.h** is the standard library and is part of every ROS node that is running on Arduino. We need to include the **ros.h** header for any message used in the ROS node.  
The **ros/time.h** header represents the ROS clock time.  
The **sensor\_msgs/Range.h** is a message definition used to advertise a single range reading from the ultrasonic sensor valid along an arc at a distance measured.

**Lines 38-42**: create **newping** objects for all the sensors. The parameters of the object are the trigger and echo pins, and the maximum distance for the sensor.

**Line 55**: **NodeHandle** is an object which represents the ROS node. **ros::NodeHandle** will start the node on the Arduino board.

**Lines 58-69**: Loop through all the sensors, and when the sensor ping cycle is completed, add the results in **oneSensorCycle()**.

**Lines 72-75**: When a new ping is received, add the sensor distance to an array.

**Lines 85-91**: We filter the readings of the sensor. Store the readings different than 0, and if the sensor returns 0, we return the last valid output. In this way, we filter the false readings of the ultrasonic sensor.

**Lines 100-102**: the function to start counting the time using **millis()**. I prefer to use **millis()** instead of **delay()** from two reasons: **millis()** is more accurate than **delay()**, and is a non-blocking alternative of **delay()**.

**Lines 104-106**: Check if the time passed and return true.

**Lines 108-115**: write the settings of **Range** message object. The first parameter is the **radiation\_type** of the sensor – **ULTRASOUND**.  
The **frame\_id** is used to specify the point of reference for data contained in that message.  
The **field\_of\_view** represents the size of the arc that the distance reading is valid for in radians. HC-SR04 has a theoretical field of view of 30 degrees. In the previous post, I did tests to determine the most accurate area of the operating detection range. Since we aim to detect most of the objects in front of the robot accurately, we set a field of view of 15 degrees, which is about 0.26 radians.  
**Min** and **max** range are the values of the measurements that are considered valid.

**Lines 118-120**: We define three objects of type **Range**, and then we give the name of the topics – **range\_left**, **range\_center**, and **range\_right**.

**Lines 123-125**: We use the **advertise()** methods to create a **Publisher**, which is used to publish the ranges on topics.

**Lines 143-160**: We check the time and publish every 40 milliseconds. Read the value returned by the sensor. Then we use **nh.now()** to return the current time and then publish the range value.

**Line 161**: The ROS network monitors socket connections to push the messages from topics onto a queue. We send a new message and then use **ros::spinOnce()** to tell ROS that a new message arrives.