

TI Designs: TIDEP-01006

Autonomous robotics reference design with Sitara™ processors and mmWave sensors using ROS



Description

This TI reference design showcases autonomous robotics with the Processor SDK Linux running on the Sitara™ AM57x processor, and the mmWave SDK running on the IWR6843ISK. This design demonstrates the functionality of an embedded robotic system where point-cloud data from the mmWave radar sensing is processed by the Sitara™ AM57x processor which runs Robot Operating System (ROS) and is the main processor for the overall system control. This design integrates enhanced processor SoCs, sensors, and builds embedded robotic systems upon widely adopted open source software components. It provides a great entry point to evaluate full autonomous robotics with Sitara processing and mmWave sensing, and enables rapid development of robotics applications by leveraging TI's hardware and software components.

Resources

TIDEP-01006	Design Folder
AM5728	Product Folder
AM57x	Product Overview
TMDSEVM572x	Tool Folder
IWR6843	Product Folder
IWR6843ISK	Tool Folder



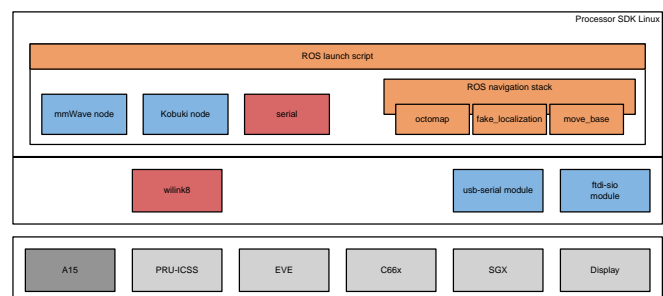
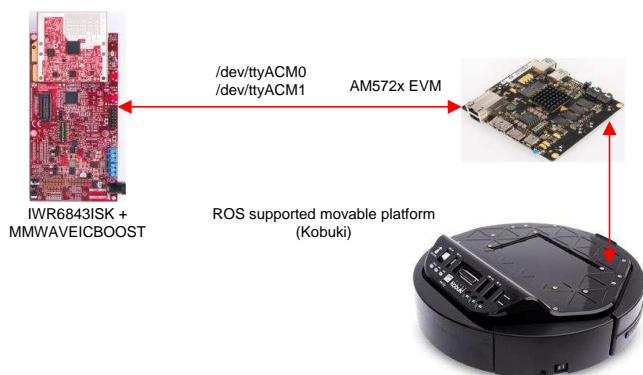
[ASK Our E2E™ Experts](#)

Features

- Embedded ROS on AM57x SoC
- mmWave radar sensing and seamlessly communicating with AM57x processor over ROS
- AM57x processor controls the mobile Kobuki platform to navigate and avoid obstacles sensed by mmWave sensors
- Real-time visualization of navigation on a remote Ubuntu Linux box with ROS Rviz
- Step-by-step guide to establish a fully-operational robotic system with sensory data and mobile base
- Open source ROS implementation enables easy adjustment/enhancement of existing modules and addition of new modules

Applications

- **Industrial:**
 - Robotics (Industrial/Logistics/Delivery/Service Robots and Forklifts), object detection and localization, sorting, quality control and inspection, and packaging
- **Smart cities:**
 - Security monitoring and road inspection
- **Drones:**
 - Obstacle avoidance, path planning, and flight control



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

Autonomous robotics has seen significant growth recently in various industrial applications, and is anticipated to further grow. The International Federation of Robotics (IFR) reported that global sales of industrial robots reached a new record of 387,000 units in 2017. That is an increase of 31 percent compared to the previous year [1]. It also estimates that more than 3 million industrial robots will be in use in factories around the world by 2020 [2].

This TI design showcases autonomous robotic system development leveraging hardware and software components from Texas Instruments (TI); specifically, the high-speed, real-time processing with the Sitara™ AM57x processors and accurate sensing with the mmWave radar sensors. This TI design demonstrates the functionality of an embedded robotic system, where point-cloud data from the mmWave sensing is communicated and processed by the Sitara™ AM57x processor. The Sitara™ AM57x processor runs the Robot Operating System (ROS) and is the main processor for overall system control.

With a collection of software libraries and packages, the ROS provides a flexible framework for creating robotic applications. Running ROS on top of Linux is a good starting point of developing robotics software. TI's Processor SDK Linux provides a fundamental software platform for development, deployment and execution of embedded applications on a TI processor running Linux. To enable creation of robotics software, Processor SDK Linux has included ROS (indigo LTS) through the meta-ros open embedded layer. Leveraging widely adopted open source (ROS and Linux) enables easy tuning, enhancement, and addition of robotics modules to meet requirements of various robotics applications.

It is frequently the case that autonomous robotics applications require sensors to enable interaction with the environment. Various types of sensors can be used, such as cameras, ultrasound, time-of-flight RGB-D cameras, and Lidar sensors. An important type of sensor is the mmWave sensor, which provides unprecedented accuracy with high spatial and velocity resolution on a single chip. For software, TI's mmWave SDK provides foundation components to facilitate end users to develop mmWave radar applications using TI's mmWave sensors.

This TI design builds a fully operational robot navigation system with embedding processing and modular ROS. The single ROS host runs on Sitara and acts as a broker for all the inter-node transactions. The point-cloud radar data from the mmWave node is seamlessly communicated to the AM57x processor for real-time processing. Meanwhile, the AM57x processor controls a Kobuki node, i.e., the mobile platform, to navigate and avoid obstacles detected by the mmWave sensor. The AM57x processor also communicates with a Ubuntu Linux box for real-time visualization of the navigation.

1.1 Key System Specifications

The key system devices used in this TI design include the AM57x SoC and the IWR6843 mmWave Sensor.

1.1.1 AM57x SoC

The AM57x is a highly-integrated, pin-compatible, scalable, Sitara™ class processor. It has single- or dual-core Arm® Cortex®-A15 cores. The AM57x is designed for embedded applications including Programmable Logic Controllers (PLCs), industrial network switches, industrial gateways for protocol translation, human machine interface (HMI), grid infrastructure protection and communications, and other industrial use applications. The device includes the following subsystems:

- Cortex®-A15 microprocessor unit (MPU) subsystem, including two Arm® Cortex®-A15 cores
- Two digital signal processor (DSP) C66x subsystems
- Two Embedded Vision Engine (EVE) subsystems
- Two Cortex-M4 subsystems, each including two Arm® Cortex®-M4 cores
- Two dual-core Programmable Real-Time Units for Industrial Communications (PRU-ICSS)
- Display subsystem (DSS)
- Video Processing subsystem (VPE)
- Video Input Capture (VIP)
- 3D-graphics processing unit (GPU) subsystem, including POWERVR™ SGX544 dual-core
- 2D-graphics accelerator (BB2D) subsystem, including Vivante™ GC320 core
- Real-time clock (RTC) and Debug subsystems
- The device provides a rich set of connectivity peripherals, including among others:
 - USB 3.0 and 2.0
 - SATA 2
 - PCIe Gen2 Gigabit Ethernet Switch subsystems
- The device integrates on-chip memory, external memory interfaces and memory management.
- The device includes support for functional safety system requirements:
 - Error Detection and Correction:
 - Parity bit per byte on C66x DSP
 - L1 program cache and Single-Error Correction Dual-Error Detection (SECEDED) on L2 memories
 - SECEDED on Large L3 memory
 - SECEDED on external DDR memory interface (EMIF1 only)
 - MMU/MPU:
 - MMU used for key masters (Cortex-A15 MPU, Cortex-M4 IPU, C66x DSP, EDMA)
 - Memory protection of C66x cores
 - MMU inside the Dynamic Memory Manage

The AM57x can meet the requirement of high data input/output (I/O) bandwidth, for communicating with the sensing component to retrieve the point cloud data. It also has the data processing power for processing the point cloud data in real time.

1.1.2 mmWave Sensor

The IWR6843 device is an integrated single-chip mmWave sensor based on FMCW radar technology capable of operation in the 60- to 64-GHz band. The device is built with a Texas Instruments (TI) low-power 45-nm RFCMOS process, and this solution enables unprecedented levels of integration in an extremely small form factor. The IWR6843 is an ideal solution for low-power, self-monitored, and accurate radar systems in industrial applications (such as, building automation, factory automation, drones, material handling, traffic monitoring, and surveillance).

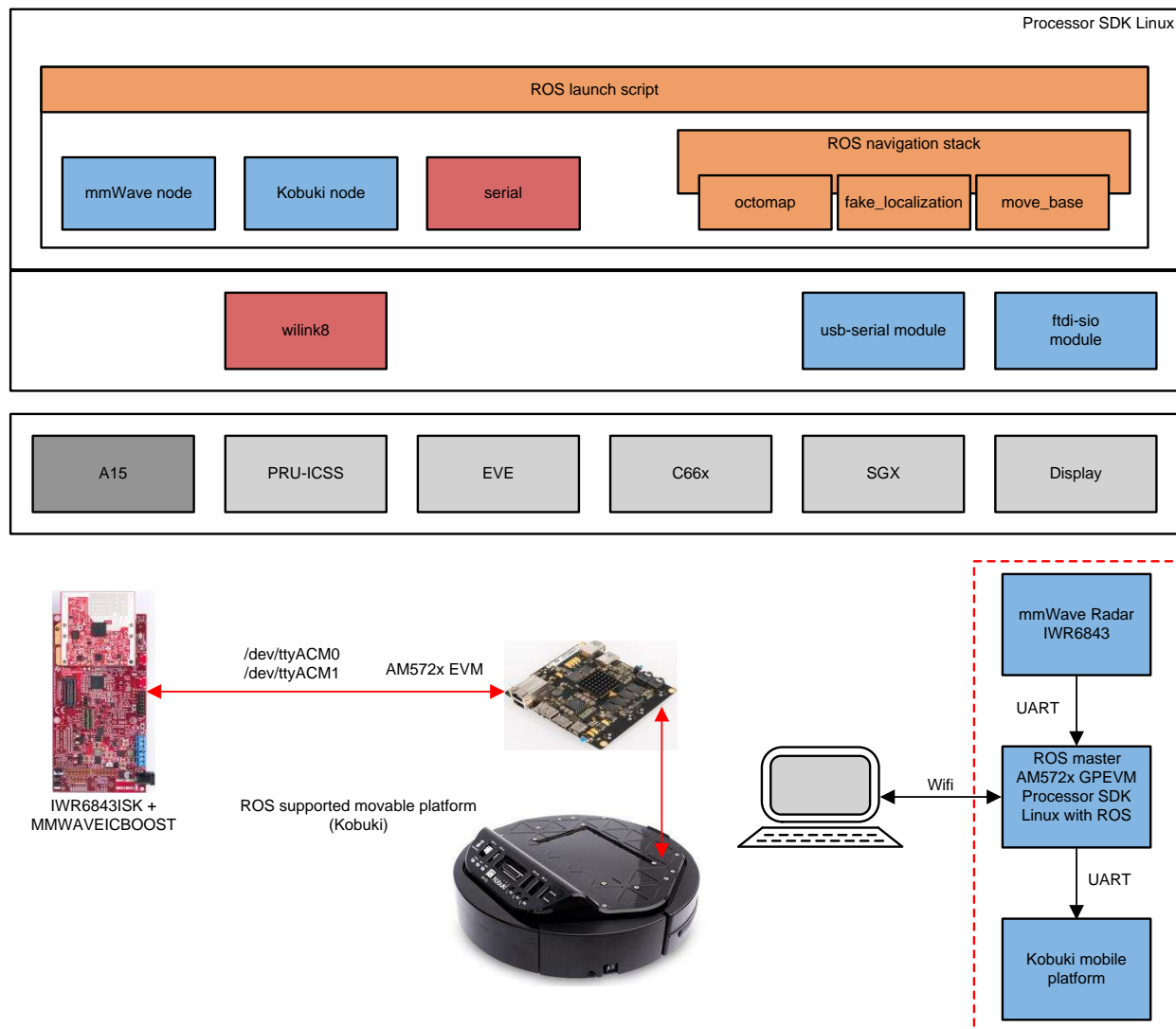
Key features of IWR6843:

- FMCW transceiver:
 - Integrated PLL, transmitter, receiver, baseband, and A2D
 - 60- to 64-GHz coverage with 4-GHz continuous bandwidth
 - Four receive channels
 - Three transmit channels
 - Ultra-accurate chirp engine based on Fractional-N PLL
 - TX power: 10 dBm
 - RX noise figure: 14 dB
 - Phase noise at 1 MHz: -92 dBc/Hz
- Built-in calibration and self-test:
 - Arm® Cortex®-R4F-Based Radio Control System (RCS)
 - Built-in firmware (ROM)
 - Self-calibrating system across frequency and temperature
- On-Chip programmable core for embedded-user application:
 - Integrated Arm® Cortex®-R4F Microcontroller clocked at 200 MHz
 - On-Chip Bootloader supports Autonomous mode (Loading User application from QSPI Flash memory)
- Integrated peripherals:
 - Internal Memories With ECC
 - Radar Hardware Accelerator (FFT, Filtering, and CFAR processing)
 - I2C
 - Two SPI ports
 - CAN-FD interface
 - Up to six general-purpose ADC ports

2 System Overview

2.1 Block Diagram

Figure 1. TIDEP-01006 Block Diagram



As shown in [Figure 1](#) above, this TI reference design consists of the following subsystems:

- An AM57x running Processor SDK Linux and ROS. The AM57x processor acts as the ROS master, runs the navigation stack, and interacts with the mmWave sensor (via the mmWave node) and the mobile platform (via the Kobuki node) for autonomous navigation. The AM57x processor also communicates with a Ubuntu Linux box which runs Rviz for visualizing the navigation.
- IWR6843 mmWave radar sensing
- Kobuki mobile platform
- Ubuntu Linux box for remote control and real-time visualization with ROS Rviz (Ubuntu Linux box is needed since Rviz requires an OpenGL desktop support, while Processor SDK Linux on the AM57x supports OpenGL ES 2.0 only).

2.2 Design Considerations

TI's Processor SDK Linux provides a fundamental software platform for developing embedded applications on a TI processor running Linux. To further enable the autonomous robotics, the Processor SDK Linux has included ROS, added and enhanced ROS packages to support navigation, and integrated the mmWave ROS packages to leverage mmWave sensors for real-time sensing with high accuracy.

2.2.1 Embedded ROS with Processor SDK Linux

To support the development of robotics applications, the Processor SDK Linux has included ROS (indigo distribution) through the meta-ros open embedded layer, starting from version 04.02.00.09. This is specified in the [processor-sdk oe-layersetup](#) config file, which sets up the meta layers for the Processor SDK Linux:

```
meta-ros,https://github.com/bmwcarit/meta-ros.git,master,e2566402ab108a19634354a934788109422cf409,layers=
```

meta-ros [3] provides a stable cross-compilation build system via the Yocto to establish the ROS indigo distribution in an embedded Linux system (for example, TI's Processor SDK Linux system). It provides many common ROS tools, libraries, and packages for developing and deploying robotics applications, including, but not limited to: catkin, control-msgs, hector-slam, kobuki, kobuki-core, kobuki-msgs, laser-geometry, navigation, octomap, roscpp-core, and so forth. The complete list can be found at the [git repo](#).

The target filesystem of the AM57x has indigo ROS installed at /opt/ros/indigo, similar to the indigo ROS installed on a Ubuntu Linux box.

```
root@am57xx-evm:/# ls -l /opt/ros/indigo/
drwxr-xr-x  2 root    root          4096 Oct  2 22:12 bin
drwxr-xr-x  4 root    root          4096 Sep 30 06:26 etc
drwxr-xr-x 141 root    root        20480 Oct  2 22:12 lib
-rw-r--r--  1 root    root           406 Sep 30 08:11 setup.bash
drwxr-xr-x 353 root    root       16384 Oct  2 22:12 share
```

2.2.2 Add and Enhance ROS Packages to Support Navigation in Processor SDK Linux

To support autonomous navigation with the Kobuki, several ROS modules are added or enhanced on top of the meta-ros layer in Processor SDK Linux. The corresponding Yocto recipes can be found from [recipes-ros](#) and listed here:

```
├─ hector-slam
│   ├── hector-map-tools_0.3.5.bb
│   ├── hector-nav-msgs_0.3.5.bb
│   └─ hector-trajectory-server_0.3.5.bb
├─ kobuki-driver
│   ├── files
│   │   └─ kobuki.conf
│   └─ kobuki-driver_0.6.5.bbappend
├─ navigation
│   ├── files
│   │   ├── 0001-navigation-rotate-recovery-mmwave-changes.patch
│   │   └─ 0002-navigation-move-base-mmwave-changes.patch
│   ├── move-base_1.12.14.bbappend
│   └─ rotate-recovery_1.12.14.bbappend
├─ octomap-server
│   └─ octomap-server_0.6.0.bb
```

Among these modules, the octomap-server and hector-slam modules are dependent ROS packages of the navigation but have not been included in the meta-ros. The kobuki-driver is adding the udev rule to insert the kernel modules needed to detect the Kobuki connection. The move-base and rotate-recovery modules under the navigation are enhanced to perform the initial rotation to clear the costmaps before the navigation. The move-base module is also taking the fixes and enhancements from the lab for autonomous robotics with ROS for mmWave [5].

2.2.3 Add ROS Packages to Enable mmWave in Processor SDK Linux

The ROS framework is mainly based on the publisher-subscriber model, and in some cases the server-client mode. The ROS framework and libraries support the point cloud, which is also the data output format of the mmWave sensors. In the ROS environment, the consumer of the point cloud is decoupled from the point cloud producer since the data format is well defined.

Due to this modular nature of ROS, it is easy to replace one sensor with another as long as the data format (in this case point cloud) is the same. Therefore, ROS is a good fit for developing robotics applications with the mmWave sensors.

TI has developed an mmWave ROS driver [4], and a lab for autonomous robotics with ROS for mmWave [5], with ROS running on a Ubuntu Linux box. In this TI design, the mmWave driver is added as a ROS package in Processor SDK Linux running on the AM57x. The ROS packages for autonomous robotics with mmWave are also added in Processor SDK Linux:

```
turtlebot_mmwave_launchers
ti_mmwave_rospkg
turtlebot_bringup
turtlebot_description
turtlebot_navigation
```

Below lists the Yocto recipes for integrating the mmWave driver and the autonomous robotics for mmWave in Processor SDK Linux:

```
|— mmwave-ros-autonomous-robotics_[version].bb
|— mmwave-ros-pkg-master_[version].bb
|— serial-ros_0.1.0.bb
```

Among the recipes above, the serial-ros is to build and install the serial driver for detecting the mmWave EVM. The mmwave-ros-pkg-master is to build the mmWave driver ROS package for interfacing with the mmWave EVM. In the lab for autonomous robotics with ROS for mmWave [5], turtlebot modules have been modified for achieving autonomous robotics with the mmWave sensor. In mmwave-ros-autonomous-robotics, these modified modules are reused with a few changes to set up the configuration and scripts to accommodate Processor SDK Linux.

2.2.4 Recompile and Deploy ROS Packages With Processor SDK Linux

The ROS indigo distribution and the ROS package required for navigation and mmWave sensing have been included in the Processor SDK Linux. Autonomous navigation can run out of box on the AM572x EVM as described in [Test Results](#).

When it is needed to modify an existing or add a new ROS package, follow the procedure below to recompile and deploy the modified/new ROS package with Processor SDK Linux.

First, refer to [Processor SDK Building The SDK](#) to set up the build environment.

Then, create/modify the Yocto recipe for the ROS package, using the hector-trajectory-server_0.3.5.bb and move-base_1.12.14.bbappend (described in [Section 2.2](#)) as the example for creating and modifying an ROS package.

After the recipe is ready, follow [Processor SDK Yocto Recipes](#) to bitbake the recipe and install the newly built ROS package. For example, run the command below to build move-base.

```
MACHINE=am57xx-evm bitbake move-base
```

After the bitbake command above is successfully done, the newly built move-base .ipk files can be found in **./build/arago-tmp-external-linaro-toolchain/work/armv7ahf-neon-linux-gnueabi/move-base/[version]/deploy-ipks** directory. Copy these .ipk files to the target filesystem and run the command below on the target to install the .ipk files:

```
opkg install [package_ipk].ipk
```


2.3 Highlighted Products

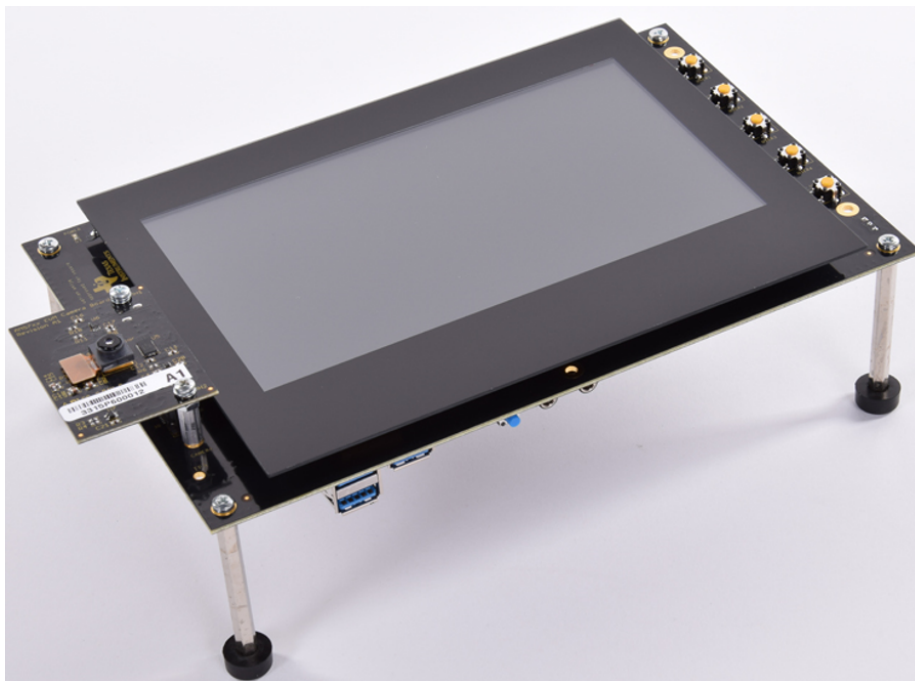
This TI design leverages the AM572x EVM as the main processor and the ROS master. The mmWave sensing is carried out with the IWR6843 Industrial Starter Kit (ISK) mounted on an MMWAVEICBOOST board. Both the AM572x EVM and the MMWAVEICBOOST board are assembled on a Kobuki mobile platform to form a fully operational robot navigation system.

2.3.1 AM572x EVM

The AM572x Evaluation Module (EVM) (see [Figure 2](#)) provides an affordable platform to quickly start evaluation of the Sitara™ Arm® Cortex®-A15 AM57x Processors (AM5728, AM5726, AM5718, AM5716) and accelerate development for HMI, machine vision, networking, medical imaging and many other industrial applications [6]. AM572x EVM is a development platform based on the dual Arm® Cortex®-A15, dual C66x DSP processor that is integrated with tons of connectivity (such as: PCIe, SATA, HDMI, USB 3.0/2.0, dual Gigabit Ethernet, and more).

The AM572x EVM also integrates video and 3D/2D graphics acceleration, as well as a dual-core Programmable Real-time Unit (PRU) and dual Arm® Cortex™-M4 cores.

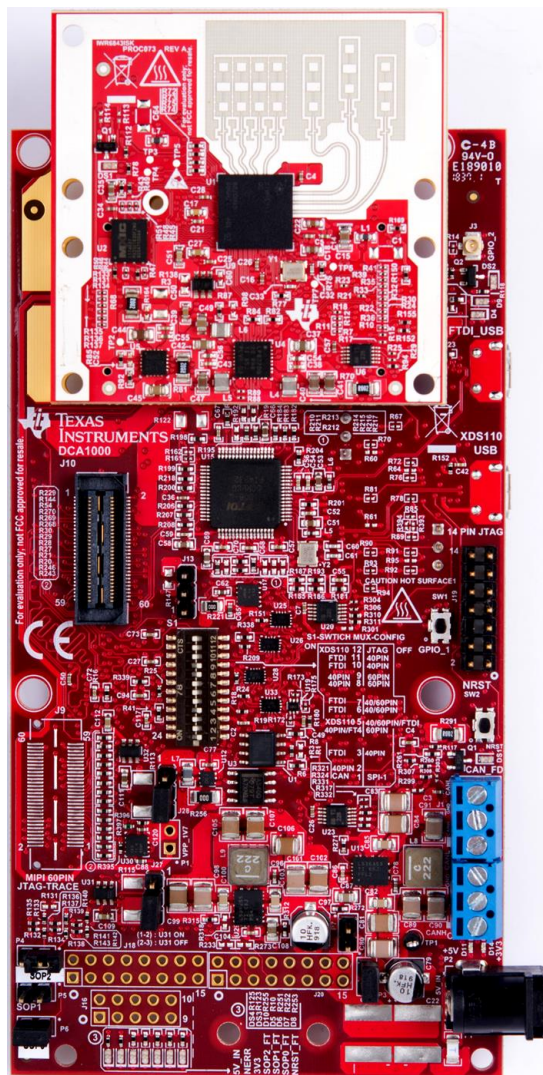
Figure 2. AM572x EVM



2.3.2 IWR6843ISK and MMWAVEICBOOST Boards

The IWR6843ISK and MMWAVEICBOOST are part of the mmWave EVM Hardware (see [Figure 3](#)). The IWR6843ISK is an easy-to-use evaluation board for the IWR6843 mmWave sensing device and, this board, contains a 60-GHz mmwave Radar transceiver in which antennas are etched and act as a Radar front-end board ([\[7\]](#)). The MMWAVEICBOOST is an add-on board used with the TI mmWave sensor Starter Kits to provide more interfaces and PC connectivity to the mmWave sensors. This carrier card enables debug, emulation, and direct interface to mmWave tools through USB connectivity ([\[8\]](#)). The IWR6843ISK and MMWAVEICBOOST contain everything required to start developing software for on-chip C67x DSP core and low-power ARM R4F controllers.

Figure 3. mmWave EVM: IWR6843ISK Mounted on MMWAVEICBOOST



2.3.3 Kobuki

The Kobuki (see [Figure 4](#)) is a low-cost mobile research base designed for education and research on state of art robotics. It is highly accurate odometry, amended by calibrated gyroscope, enables precise navigation [\[9\]](#).

The Kobuki is a mobile base that has sensors, motors, and power sources. The Kobuki also provides power supplies for an external computer as well as additional sensors and actuators. This TI design attaches the AM572x EVM to be the computational core and the IWR6843ISK (through the MMWAVEICBOOST board) as the 3-D sensor to the Kobuki, and makes it truly functional as a TurtleBot2 platform for autonomous robotics.

Figure 4. Kobuki



3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

This section provides the full list of hardware components required to build the autonomous navigation system. It also lists the software SDKs which are needed to flash the mmWave EVM and install on the AM572x EVM.

3.1.1 Hardware

The following hardware setup is needed to run the autonomous robotics demo:

- AM572x EVM with micro SD card
- mmWave EVM: IWR6843ISK and MMWAVEICBOOST Boards
- Kobuki with plate and standoff kit, as used by Turtlebot2
- Ubuntu Linux box for visualization
- TP-Link Wireless N Nano Router (TL-WR702N) with USB and Ethernet cables: follow the [client mode instructions](#) to configure the TP-Link Wireless N Nano Router
- LINKSYS EA3500 Wireless router
- USB 2.0 printer-style cable (A-Male to B-Male): used for connecting the AM572x EVM to Kobuki
- USB to microUSB cable: used for connecting the AM572x EVM to the mmWave EVM
- 2-pin miniFit JR connector/cable to go from Kobuki 12-V, 5-A output port to the AM572x EVM
- 12-V to 5-V DC-to-DC converter (must be able to output at least 2.5 A at 5 V) to allow the EVM to be powered from the Kobuki
- 2-pin miniFit JR connector/cable to go from Kobuki 12-V, 1.5-A output port to the 12-V input on the converter
- 2.1-mm barrel jack connector (center positive) with cable/wire to go from the 5-V output on the converter to the mmWave EVM
- Screws, washers, bolts, and nuts for mounting the AM572x EVM, mmWave sensor, and DC converter to the Kobuki platform

3.1.2 Software

TI mmWave EVM

Visit [CCS_UniFlash](#) page to install the UniFlash tool and flash the mmWave EVM with the Out-of-Box demo firmware from the [latest version of the mmWave SDK](#). The support for IWR6843 is provided starting from version 03.01.00.02.

AM572x EVM

The Autonomous robotics demo with mmWave is included in the AM57x Processor SDK for Linux starting from version 05.01.00.11. Download the [latest version](#), and refer to [Processor SDK Linux Getting Started Guide](#) to get started with the Processor SDK Linux, and create an SD card with PLSDK filesystem.

Ubuntu Linux Box with ROS Indigo LTS

Verification of this TI design is done with Ubuntu 14.04. For visualization of the ROS navigation, install the ROS indigo on the Ubuntu Linux box, which is preferred for compatibility reasons. Follow the [ROS indigo installation instructions](#):

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" >
/etc/apt/sources.list.d/ros-latest.list'
wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get install ros-indigo-desktop-full
```

There are a few additional ROS packages required on the Ubuntu Linux box. These additional ROS packages will be discussed in Software Setup for [Ubuntu Linux Box with ROS Indigo LTS](#).

3.2 Testing and Results

This section provides details of the hardware and software setup for running autonomous navigation. Step-by-step testing procedures and results are also provided.

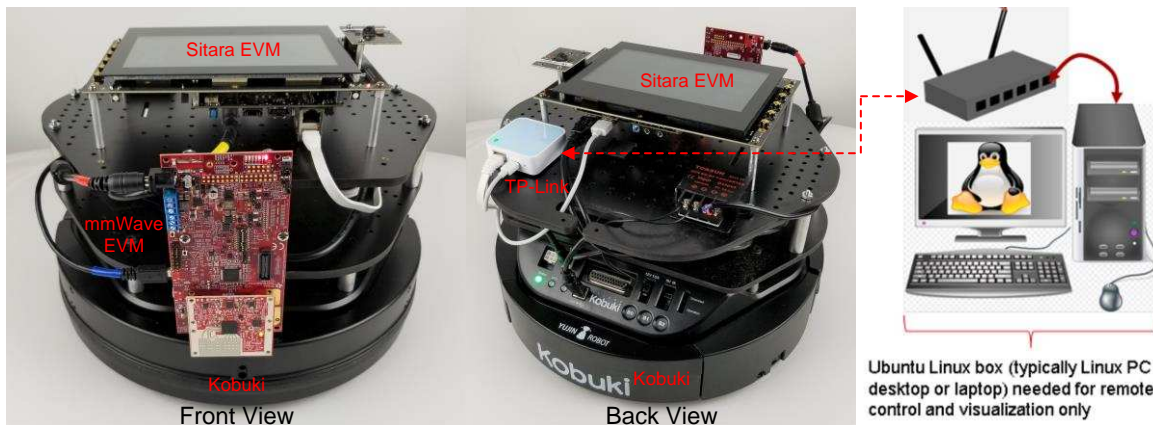
3.2.1 Test Setup

Hardware setup of this autonomous navigation system involves mounting the AM572x EVM and the mmWave EVM on the Kobuki. A few steps are also needed to further set up the software for the Ubuntu Linux box and the AM572x EVM.

3.2.1.1 Hardware Setup

Mounting and interconnection setup (see [Figure 5](#)):

- Mount the IWR6843ISK on the MMWAVEICBOOST Board to be the mmWave EVM
- Mount the mmWave EVM on the Kobuki
- Mount the AM572x EVM on the Kobuki
- Mount the 12-V to 5-V DC-to-DC converter on the Kobuki
- Mount the TP-Link on the Kobuki
- Connect the Kobuki to the AM572x EVM via the USB
- Connect the mmWave to the AM572x EVM via the USB
- Connect the TP-Link USB and the Ethernet cable to the AM572x EVM to make it WIFI enabled
- Power-on the mmWave EVM (make sure the SOP2 jumper on the mmWave EVM has been removed) with firmware flash done
- Power-on the AM572x EVM with the SD card (created with Processor SDK Linux filesystem) inserted
- Connect the Ubuntu Linux box to the wireless router (wired or wireless)

Figure 5. Hardware Mounting and Interconnection Setup


3.2.1.2 Software Setup

The software setup for the autonomous navigation includes several steps on the Ubuntu Linux Box to install the additional binary, plugin, and ROS packages. A few steps are also needed on the AM572x EVM to set up the environment and check the hardware connections.

3.2.1.2.1 Ubuntu Linux Box with ROS Indigo LTS

The ROS is a distributed system, meaning that it can communicate over a local network with other ROS components. For this demo, the Ubuntu Linux box and the AM572x EVM must be on the exact same network and must be able to ping each other by IP address. The Ubuntu Linux box must also be able to ssh into the AM572x EVM by IP address.

Install ssh on the Ubuntu Linux box using the following command:

```
$ sudo apt-get install ssh
```

In order to run Rviz for visualizing the autonomous navigation, it requires building and installing a ROS plugin of rviz_plugin_covariance.

First, follow the [ROS configuration instructions](#) to create a new ROS workspace:

```
source /opt/ros/indigo/setup.bash
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Second, clone and build the rviz_plugin_covariance:

```
cd ~/catkin_ws/src
git clone https://github.com/laas/rviz_plugin_covariance
cd ~/catkin_ws/
catkin_make
```

At the end of the catkin_make, there should be a line as shown below:

```
[100%] Built target rviz_plugin_covariance
```

After the plugin is successfully built, add the following lines to ~/.bashrc to set up the ROS variables on the Ubuntu Linux box for convenience. Please replace \$SITARA_IP_ADDR with the IP address found from running "ifconfig" on the AM572x EVM, and \$UBUNTU_IP_ADDR with the IP address found from running "ifconfig" on the Ubuntu Linux box.

```
source /opt/ros/indigo/setup.bash
source /<workspace_dir>/devel/setup.bash
export ROS_MASTER_URI=http://$SITARA_IP_ADDR:11311
export ROS_IP=$UBUNTU_IP_ADDR
```


The last step of the Ubuntu preparation is, copying the navigation demo packages from the AM57x target filesystem (located under `/opt/ros/indigo/share/`) to the Ubuntu Linux box (destination also `/opt/ros/indigo/share/`), including:

```
/opt/ros/indigo/share/turtlebot_description/
/opt/ros/indigo/share/turtlebot_bringup/
/opt/ros/indigo/share/turtlebot_mmwave_launchers/
/opt/ros/indigo/share/kobuki_description
```

3.2.1.2.2 AM572x EVM Running Processor SDK Linux

The ROS, mmWave ROS driver, as well as the autonomous navigation demo, have been included in the filesystem of the Processor SDK Linux, in the `/opt/ros/indigo` folder; therefore, no additional installation steps are required. Only setting up the configuration of the AM572x EVM and a few checks are needed.

First, modify `/opt/ros/indigo/setup.bash` to replace `$SITARA_IP_ADDR` with the IP address found from running "ifconfig" on the AM572x EVM:

```
root@am57xx-evm:/opt/ros/indigo# cat /opt/ros/indigo/setup.bash
export ROS_ROOT=/opt/ros/indigo
export PATH=$PATH:/opt/ros/indigo/bin
export LD_LIBRARY_PATH=/opt/ros/indigo/lib
export PYTHONPATH=/usr/lib/python3.5/site-packages:/opt/ros/indigo/lib/python3.5/site-packages
export ROS_MASTER_URI=http://$SITARA_IP_ADDR:11311
export ROS_IP=$SITARA_IP_ADDR
export CMAKE_PREFIX_PATH=/opt/ros/indigo
export ROS_PACKAGE_PATH=/opt/ros/indigo/share
touch /opt/ros/indigo/.catkin
```

The ROS packages of the Processor SDK Linux are compiled with Python3. As the default python setting in the Processor SDK is Python 2.7, update the symbolic links for python as stated below to run the ROS on the AM572x EVM:

```
cd /usr/bin
ln -sf python3 python.python
ln -sf python3-config python-config.python
```

Before running the demo, check that the AM572x EVM has been successfully connected to the Kobuki and the mmWave EVM. "dev/kobuki" shows the connection with the Kobuki, and "/dev/ttyACM*" shows the connection with the mmWave EVM.

```
root@am57xx-evm:/opt/ros/indigo# ls -l /dev/kobuki
lrwxrwxrwx    1 root    root          7 Sep 21 02:03 /dev/kobuki -> ttyUSB0

root@am57xx-evm:/opt/ros/indigo# ls -l /dev/ttyACM*
crw-rw----    1 root    dialout   166,    0 Sep 21 02:03 /dev/ttyACM0
crw-rw----    1 root    dialout   166,    1 Sep 21 02:03 /dev/ttyACM1
```

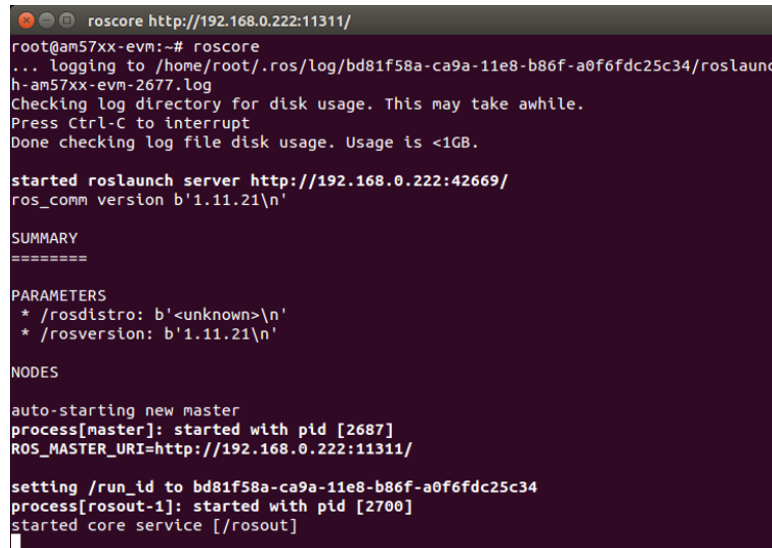
3.2.2 Test Results

Before running the navigation demo, close all the ROS terminal consoles for both the AM572x EVM and the Ubuntu Linux box. Then, follow the steps below to run the demo.

1. Open the first SSH terminal to the AM572x EVM and run `roscore` (see [Figure 6](#)).

```
source /opt/ros/indigo/setup.bash
roscore
```

Figure 6. AM572x EVM "roscore" Screen Capture



```
roscore http://192.168.0.222:11311/
root@am57xx-evm:~# roscore
... logging to /home/root/.ros/log/bd81f58a-ca9a-11e8-b86f-a0f6fdc25c34/roslaunch-
h-am57xx-evm-2677.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.222:42669/
ros_comm version b'1.11.21\n'

SUMMARY
=====

PARAMETERS
* /rostdistro: b'unknown>\n'
* /rosversion: b'1.11.21\n'

NODES

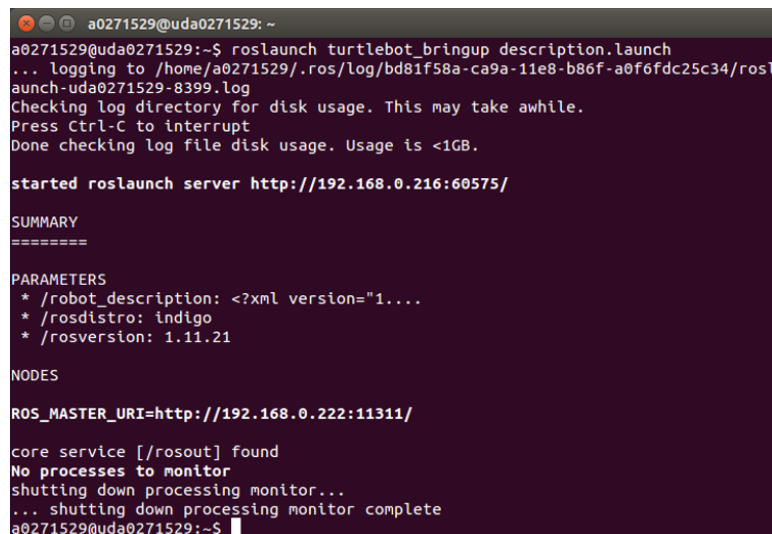
auto-starting new master
process[master]: started with pid [2687]
ROS_MASTER_URI=http://192.168.0.222:11311/

setting /run_id to bd81f58a-ca9a-11e8-b86f-a0f6fdc25c34
process[rosout-1]: started with pid [2700]
started core service [/rosout]
```

2. Open a terminal console on the Ubuntu Linux box, and launch the robot description (see [Figure 7](#)).

```
roslaunch turtlebot_bringup description.launch
```

Figure 7. Ubuntu "description.launch" Screen Capture



```
a0271529@uda0271529: ~
a0271529@uda0271529:~$ roslaunch turtlebot_bringup description.launch
... logging to /home/a0271529/.ros/log/bd81f58a-ca9a-11e8-b86f-a0f6fdc25c34/rosl
aunch-uda0271529-8399.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.216:60575/

SUMMARY
=====

PARAMETERS
* /robot_description: <?xml version="1....
* /rostdistro: indigo
* /rosversion: 1.11.21

NODES

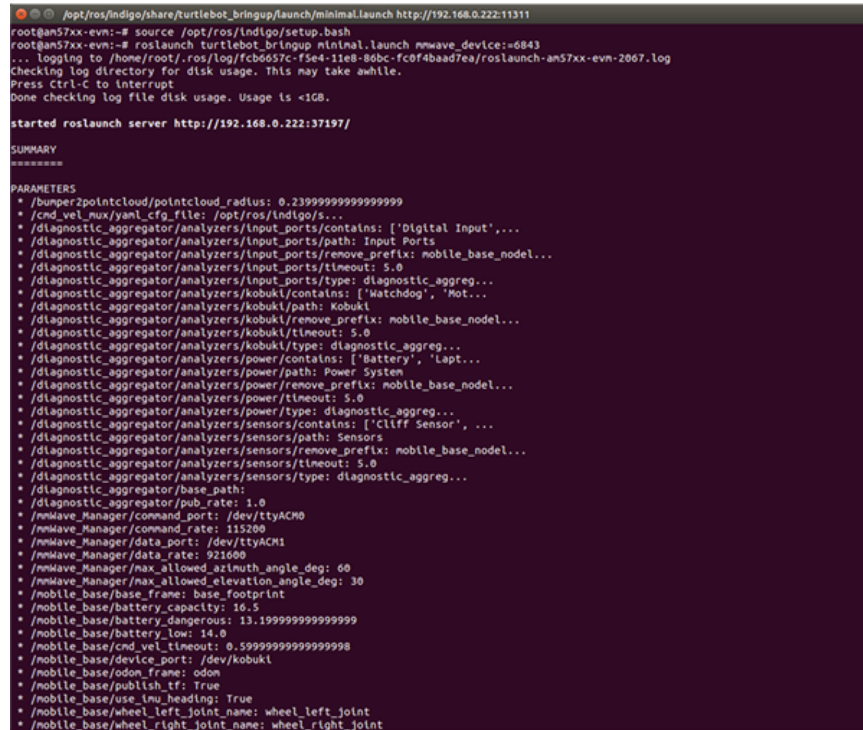
ROS_MASTER_URI=http://192.168.0.222:11311/

core service [/rosout] found
No processes to monitor
shutting down processing monitor...
... shutting down processing monitor complete
a0271529@uda0271529:~$
```


- Open the second SSH terminal to the AM572x EVM. Run the `minimal.launch` of `turtlebot_bringup` to set up the mmWave sensor and the Kobuki. Please set the `mmwave_device` to "6843" at the end of this command. [Section 3.2.2](#) shows the beginning of running `minimal.launch`, and [Figure 9](#) shows the end of `minimal.launch`.

```
source /opt/ros/indigo/setup.bash
roslaunch turtlebot_bringup minimal.launch mmwave_device:=6843
```

Figure 8. AM572x EVM "minimal.launch" Screen Capture - Beginning



```
root@am57xx-evm:~# source /opt/ros/indigo/setup.bash
root@am57xx-evm:~# roslaunch turtlebot_bringup minimal.launch mmwave_device:=6843
... logging to /home/root/.ros/log/fcb657c-f5e4-11e8-8d6c-fcf4baad7ea/roslaunch-am57xx-evm-2067.log
checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.222:37197/

SUMMARY
=====

PARAMETERS
* /bumper2pointcloud/pointcloud_radius: 0.23999999999999999
* /cmd_vel_mux/yaml_cfg_file: /opt/ros/indigo/s...
* /diagnostic_aggregator/analyzers/input_ports/contains: ['Digital Input',...
* /diagnostic_aggregator/analyzers/input_ports/path: Input Ports
* /diagnostic_aggregator/analyzers/input_ports/remove_prefix: mobile_base_model...
* /diagnostic_aggregator/analyzers/input_ports/timeout: 5.0
* /diagnostic_aggregator/analyzers/input_ports/type: diagnostic_aggreg...
* /diagnostic_aggregator/analyzers/kobuki/contains: ['Watchdog', 'Mot...
* /diagnostic_aggregator/analyzers/kobuki/path: Kobuki
* /diagnostic_aggregator/analyzers/kobuki/remove_prefix: mobile_base_model...
* /diagnostic_aggregator/analyzers/kobuki/timeout: 5.0
* /diagnostic_aggregator/analyzers/kobuki/type: diagnostic_aggreg...
* /diagnostic_aggregator/analyzers/power/contains: ['Battery', 'Lapt...
* /diagnostic_aggregator/analyzers/power/path: Power System
* /diagnostic_aggregator/analyzers/power/remove_prefix: mobile_base_model...
* /diagnostic_aggregator/analyzers/power/timeout: 5.0
* /diagnostic_aggregator/analyzers/power/type: diagnostic_aggreg...
* /diagnostic_aggregator/analyzers/sensors/contains: ['Cliff Sensor', ...
* /diagnostic_aggregator/analyzers/sensors/path: Sensors
* /diagnostic_aggregator/analyzers/sensors/remove_prefix: mobile_base_model...
* /diagnostic_aggregator/analyzers/sensors/timeout: 5.0
* /diagnostic_aggregator/analyzers/sensors/type: diagnostic_aggreg...
* /diagnostic_aggregator/base_path:
* /diagnostic_aggregator/pub_rate: 1.0
* /mmWave_Manager/command_port: /dev/ttyACM0
* /mmWave_Manager/command_rate: 115200
* /mmWave_Manager/data_port: /dev/ttyACM1
* /mmWave_Manager/data_rate: 921600
* /mmWave_Manager/max_allowed_azimuth_angle_deg: 60
* /mmWave_Manager/max_allowed_elevation_angle_deg: 30
* /mobile_base/base_frame: base_footprint
* /mobile_base/battery_capacity: 16.5
* /mobile_base/battery_dangerous: 13.199999999999999
* /mobile_base/battery_low: 14.0
* /mobile_base/cmd_vel_timeout: 0.59999999999999998
* /mobile_base/device_port: /dev/kobuki
* /mobile_base/odometry_frame: odom
* /mobile_base/publish_tf: True
* /mobile_base/use_imu_heading: True
* /mobile_base/wheel_left_joint_name: wheel_left_joint
* /mobile_base/wheel_right_joint_name: wheel_right_joint
```

Figure 9. AM572x EVM "minimal.launch" Screen Capture - End

```

/opt/ros/indigo/share/turtlebot_bringup/launch/minimal.launch http://192.168.0.222:11311
mmWaveDemo: />
[ INFO] [1543722671.311427349]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.312091354]: mmWaveQuickConfig: Sending command: 'calib0cRangeSig -1 0 -5 8 256'
[ INFO] [1543722671.319546205]: mmWaveConnSrv: Sending command to sensor: 'calib0cRangeSig -1 0 -5 8 256'
[ INFO] [1543722671.324284359]: mmWaveConnSrv: Received response from sensor: 'calib0cRangeSig -1 0 -5 8 256'
Done
mmWaveDemo: />
[ INFO] [1543722671.325751450]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.326325988]: mmWaveQuickConfig: Sending command: 'compRangeBlasAndRxChanPhase 0.0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0'
[ INFO] [1543722671.333245828]: mmWaveConnSrv: Sending command to sensor: 'compRangeBlasAndRxChanPhase 0.0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0'
[ INFO] [1543722671.342075375]: mmWaveConnSrv: Received response from sensor: 'compRangeBlasAndRxChanPhase 0.0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0'
Done
mmWaveDemo: />
[ INFO] [1543722671.343493503]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.344218346]: mmWaveQuickConfig: Sending command: 'measureRangeBlasAndRxChanPhase 0 1.5 0.2'
[ INFO] [1543722671.357234932]: mmWaveConnSrv: Sending command to sensor: 'measureRangeBlasAndRxChanPhase 0 1.5 0.2'
[ INFO] [1543722671.362906143]: mmWaveConnSrv: Received response from sensor: 'measureRangeBlasAndRxChanPhase 0 1.5 0.2'
Done
mmWaveDemo: />
[ INFO] [1543722671.364443018]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.365025527]: mmWaveQuickConfig: Sending command: 'aaafovCfg -1 -90 90 -90 90'
[ INFO] [1543722671.375828708]: mmWaveConnSrv: Sending command to sensor: 'aaafovCfg -1 -90 90 -90 90'
[ INFO] [1543722671.380277136]: mmWaveConnSrv: Received response from sensor: 'aaafovCfg -1 -90 90 -90 90'
Done
mmWaveDemo: />
[ INFO] [1543722671.382066102]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.382727239]: mmWaveQuickConfig: Sending command: 'cfarFovCfg -1 0 0 8.59'
[ INFO] [1543722671.391071714]: mmWaveConnSrv: Sending command to sensor: 'cfarFovCfg -1 0 0 8.59'
[ INFO] [1543722671.395410034]: mmWaveConnSrv: Received response from sensor: 'cfarFovCfg -1 0 0 8.59'
Done
mmWaveDemo: />
[ INFO] [1543722671.397554306]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.398351210]: mmWaveQuickConfig: Sending command: 'cfarFovCfg -1 1 -5.06 5.06'
[ INFO] [1543722671.412748511]: mmWaveConnSrv: Sending command to sensor: 'cfarFovCfg -1 1 -5.06 5.06'
[ INFO] [1543722671.417181666]: mmWaveConnSrv: Received response from sensor: 'cfarFovCfg -1 1 -5.06 5.06'
Done
mmWaveDemo: />
[ INFO] [1543722671.418734319]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.419482423]: mmWaveQuickConfig: Sending command: 'sensorStart'
[ INFO] [1543722671.428564591]: mmWaveConnSrv: Sending command to sensor: 'sensorStart'
[ INFO] [1543722671.435694108]: mmWaveConnSrv: Received response from sensor: 'sensorStart'
Done
mmWaveDemo: />
[ INFO] [1543722671.437078890]: mmWaveQuickConfig: Command successful (mmWave sensor responded with 'Done')
[ INFO] [1543722671.437283850]: mmWaveQuickConfig: mmWaveQuickConfig will now terminate. Done configuring mmWave device using config file: /opt/ros/indigo/share/ti_mmwave_ros/pkg/cfg/6843_3d_custom.cfg
[mmWaveQuickConfig-4] process has finished cleanly
log file: /home/root/.ros/log/fcb6657c-f5e4-11e8-86bc-fc0f4baad7ea/mmWaveQuickConfig-4*.log
[ERROR] [1543722731.684331238]: Kobuki : malformed sub-payload detected. [85][77][55 40 01 0f 80 79 00 00 00 83 AA 4C 9E 00 00 00 13 9f 00 ]
[ERROR] [1543722731.743101702]: Kobuki : malformed sub-payload detected. [73][170][00 AA 35 40 91 0f 0c 79 00 00 00 83 ]

```

After the Kobuki and the mmWave sensor are configured, you may see this periodic “Kobuki : malformed subpayload detected” error message. These error messages come from the Kobuki driver and do not affect the operation of the demo.

4. Open the third SSH terminal to the AM572x EVM, and launch the radar navigation with the mmWave sensor. [Figure 10](#) shows the beginning of running radar_navigation.launch, and [Figure 11](#) shows the end of radar_navigation.launch.

```

source /opt/ros/indigo/setup.bash
roslaunch turtlebot_mmwave_launchers radar_navigation.launch

```

Figure 10. AM572x EVM "radar_navigation.launch" Screen Capture - Beginning

```

a0271529@uda0271529:~$ ssh root@192.168.0.222
root@an57xx-evm:~$ source /opt/ros/indigo/setup.bash
root@an57xx-evm:~$ roslaunch turtlebot_mmwave_launchers radar_navigation.launch
... logging to /home/root/.ros/log/bd81f58a-ca9a-11e8-b80f-a0f6dc25c34/roslaunch-an57xx-evm-4520.log
checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.222:38191/

SUMMARY
=====

PARAMETERS
* /fake_localization/use_map_topic: True
* /i_filt/filter_field_name: Intensity
* /i_filt/filter_limit_max: 100
* /i_filt/filter_limit_min: 28
* /i_filt/filter_limit_negative: False
* /move_base/DWAPlannerROS/acc_lin_theta: 2.0
* /move_base/DWAPlannerROS/acc_lin_x: 2.0
* /move_base/DWAPlannerROS/acc_lin_y: 0.0
* /move_base/DWAPlannerROS/forward_point_distance: 0.12
* /move_base/DWAPlannerROS/global_frame_id: odom
* /move_base/DWAPlannerROS/goal_distance_bias: 0.01
* /move_base/DWAPlannerROS/holonomic_robot: False
* /move_base/DWAPlannerROS/latch_xy_goal_tolerance: True
* /move_base/DWAPlannerROS/max_rot_vel: 2.0
* /move_base/DWAPlannerROS/max_scaling_factor: 0.20000000000000001
* /move_base/DWAPlannerROS/max_trans_vel: 0.5
* /move_base/DWAPlannerROS/max_vel_x: 0.20000000000000001
* /move_base/DWAPlannerROS/max_vel_y: 0.0
* /move_base/DWAPlannerROS/min_rot_vel: 0.40000000000000002
* /move_base/DWAPlannerROS/min_trans_vel: 0.10000000000000001
* /move_base/DWAPlannerROS/min_vel_x: 0.0
* /move_base/DWAPlannerROS/min_vel_y: 0.0
* /move_base/DWAPlannerROS/occdist_scale: 0.01
* /move_base/DWAPlannerROS/oscillation_reset_angle: 0.0
* /move_base/DWAPlannerROS/oscillation_reset_dist: 0.0
* /move_base/DWAPlannerROS/path_distance_bias: 32.0
* /move_base/DWAPlannerROS/publish_cost_grid_pc: True
* /move_base/DWAPlannerROS/publish_traj_pc: True
* /move_base/DWAPlannerROS/rot_stopped_vel: 0.40000000000000002
* /move_base/DWAPlannerROS/scaling_speed: 0.25
* /move_base/DWAPlannerROS/sin_time: 1.7
* /move_base/DWAPlannerROS/stop_time_buffer: 0.20000000000000001
* /move_base/DWAPlannerROS/trans_stopped_vel: 0.05000000000000003
* /move_base/DWAPlannerROS/vtheta_samples: 20
* /move_base/DWAPlannerROS/vx_samples: 3
* /move_base/DWAPlannerROS/vy_samples: 0
* /move_base/DWAPlannerROS/xy_goal_tolerance: 0.17000000000000001
* /move_base/DWAPlannerROS/yaw_goal_tolerance: 0.40000000000000002
* /move_base/GlobalPlanner/allow_unknown: True
* /move_base/GlobalPlanner/cost_factor: 3.0
* /move_base/GlobalPlanner/default_tolerance: 0.0
* /move_base/GlobalPlanner/lethal_cost: 253
* /move_base/GlobalPlanner/neutral_cost: 66

```

Figure 11. AM572x EVM "radar_navigation.launch" Screen Capture - End

```

z_filt (nodelet/nodelet)

ROS_MASTER_URI=http://192.168.0.222:11311

core service [/rosout] found
process[pcl_manager-1]: started with pid [4538]
process[x_filt-2]: started with pid [4539]
process[y_filt-3]: started with pid [4540]
process[z_filt-4]: started with pid [4541]
[ INFO] [1538963731.716847753]: Loading nodelet /x_filt of type pcl/PassThrough to manager pcl_manager with the following remappings:
[ INFO] [1538963731.716570076]: /x_filt/input -> /mmWaveDataHdl/RScan
[ INFO] [1538963731.716873775]: /x_filt/output -> /x_filt_out
process[i_filt-5]: started with pid [4548]
[ INFO] [1538963731.756673753]: waitForService: Service [/pcl_manager/load_nodelet] has not been advertised, waiting...
process[octomap_server-6]: started with pid [4556]
process[fake_localization-7]: started with pid [4564]
process[navigation_velocity_smoother-8]: started with pid [4576]
[ INFO] [1538963731.932698147]: Loading nodelet /y_filt of type pcl/PassThrough to manager pcl_manager with the following remappings:
[ INFO] [1538963731.933011089]: /y_filt/input -> /x_filt_out
[ INFO] [1538963731.933106574]: /y_filt/output -> /xy_filt_out
process[kobuki_safety_controller-9]: started with pid [4590]
[ INFO] [1538963732.031702419]: Loading nodelet /z_filt of type pcl/PassThrough to manager pcl_manager with the following remappings:
[ INFO] [1538963732.032017504]: /z_filt/input -> /xy_filt_out
[ INFO] [1538963732.032115918]: /z_filt/output -> /xyz_filt_out
process[move_base-10]: started with pid [4594]
[ INFO] [1538963732.103090637]: waitForService: Service [/pcl_manager/load_nodelet] has not been advertised, waiting...
process[voxel_grid_2_point_cloud-11]: started with pid [4607]
[ INFO] [1538963732.231257493]: Loading nodelet /i_filt of type pcl/PassThrough to manager pcl_manager with the following remappings:
[ INFO] [1538963732.231843256]: /i_filt/input -> /xyz_filt_out
[ INFO] [1538963732.232113120]: /i_filt/output -> /xyz_filt_out
[ INFO] [1538963732.246891386]: waitForService: Service [/pcl_manager/load_nodelet] has not been advertised, waiting...
[ INFO] [1538963732.379298015]: Initializing nodelet with 2 worker threads.
[ INFO] [1538963732.483849584]: waitForService: Service [/pcl_manager/load_nodelet] is now available.
[ INFO] [1538963732.516391826]: waitForService: Service [/pcl_manager/load_nodelet] is now available.
[ INFO] [1538963732.548013980]: waitForService: Service [/pcl_manager/load_nodelet] is now available.
[ INFO] [1538963734.532044263]: Using plugin "static_layer"
[ INFO] [1538963734.617682406]: Requesting the map...
[ INFO] [1538963735.095634648]: Resizing static layer to 38 X 29 at 0.050000 m/pix
[ INFO] [1538963735.195515722]: Received a 38 X 29 map at 0.050000 m/pix
[ INFO] [1538963735.246841589]: Using plugin "obstacle_layer"
[ INFO] [1538963735.270546980]: Subscribed to Topics: scan bump
[ INFO] [1538963735.861698539]: Using plugin "inflation_layer"
[ INFO] [1538963736.242833991]: Using plugin "static_layer"
[ INFO] [1538963736.265771943]: Requesting the map...
[ INFO] [1538963736.286704866]: Resizing static layer to 38 X 29 at 0.050000 m/pix
[ INFO] [1538963736.384361961]: Received a 38 X 29 map at 0.050000 m/pix
[ INFO] [1538963736.414709785]: Using plugin "obstacle_layer"
[ INFO] [1538963736.427042604]: Subscribed to Topics: scan bump
[ INFO] [1538963736.683044150]: Using plugin "inflation_layer"
[ INFO] [1538963737.070168693]: Created local_planner dwa_local_planner/DWAPlannerROS
[ INFO] [1538963737.114620938]: Sin period is set to 0.20
[ INFO] [1538963738.298215809]: odom received!

```

5. Open a terminal console on the Ubuntu Linux box, and launch the Rviz (navigation_visualization_2.rviz) for real-time visualization of the navigation. [Figure 12](#) shows the console output and [Figure 13](#) shows the Rviz display.

```
roslaunch rviz rviz -d
/opt/ros/indigo/share/turtlebot_mmmwave_launchers/launch/navigation_v
isualization_2.rviz
```

Figure 12. Ubuntu "navigation_v isualization_2.rviz" Screen Capture

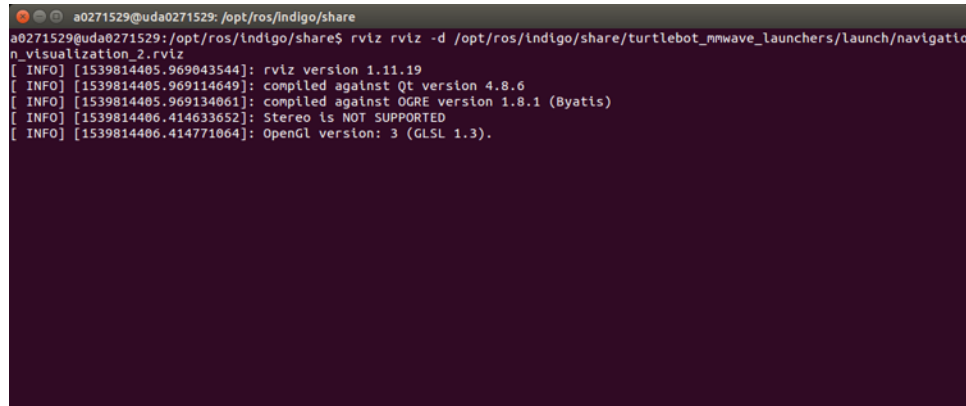
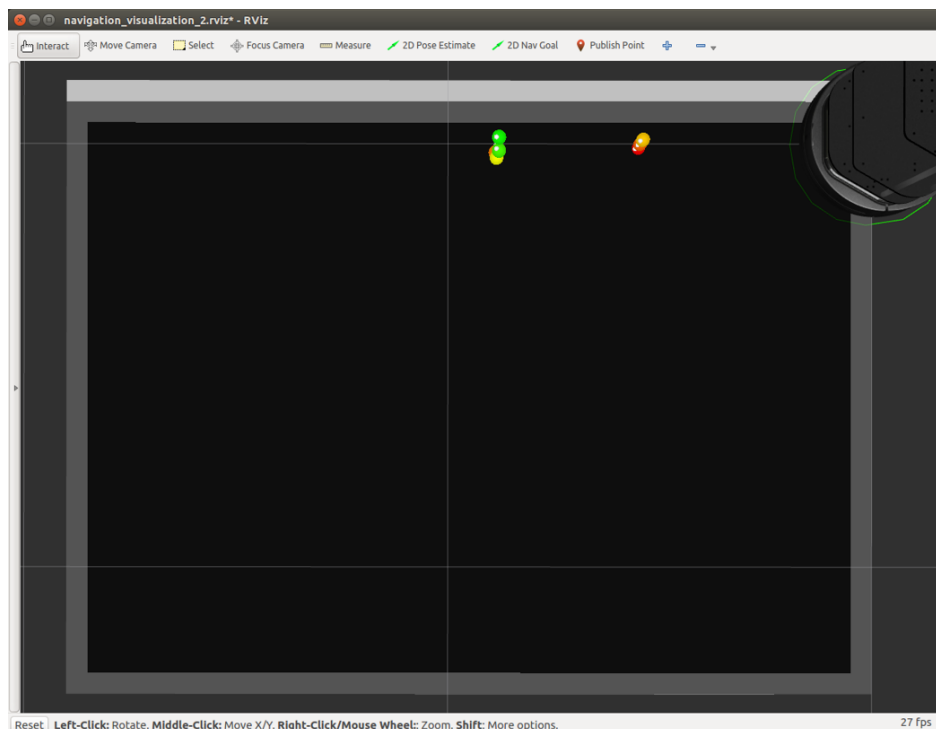


Figure 13. Ubuntu "navigation_v isualization_2.rviz" Rviz Screen Capture



6. Start navigation on the AM572x EVM by setting the initial position and the goal to be reached

The demo uses a pre-defined size 4-by-6 feet map file, with three locations (a, b, and c for top, middle, and bottom, respectively) on each side (left and right). Open the fourth SSH terminal to the AM572x EVM, and set the initial position with the side (left or right) and the location (a, b, or c), and then, set the location (a, b, or c) on the other side as the goal, by running the scripts below (see [Figure 14](#)):

```
source /opt/ros/indigo/setup.bash
cd /opt/ros/indigo/share/turtlebot_mmwave_launchers/scripts/
./start_nav.sh
```

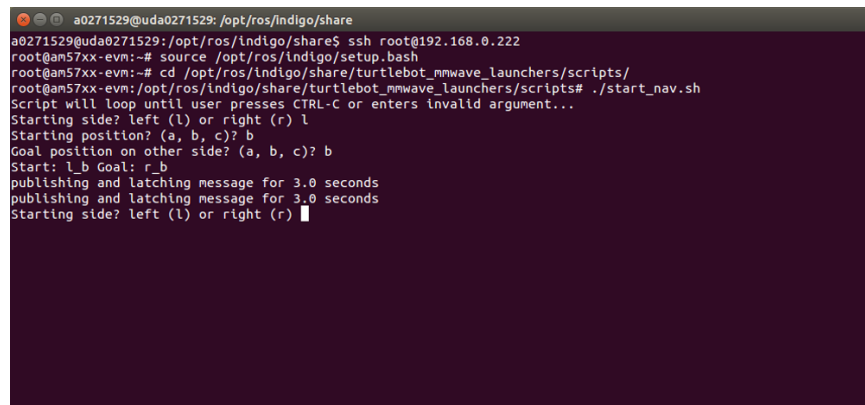
First, the Kobuki has the initial left-side middle-point B position set (see [Figure 15](#)).

Then, the Kobuki navigates to the right-side middle-point B goal (see [Figure 16](#)), following the shortest path available. After it reaches the goal, the Kobuki rotates and faces toward the initial position.

If there are objects between the initial position and the goal, the mmWave sensor detects them (shown as red/yellow/green circles in Rviz as in [Figure 17](#)). With the radar sensing data, a new path (the thin blue line connecting the initial position and the goal) is found in real-time to avoid the obstacles, demonstrating the "sense and avoid" capability of the autonomous navigation in this TI Design (see [Figure 17](#)).

To re-run the navigation demo, simply specify the starting side/position and the goal position from the console of `./start_nav.sh` (see [Figure 14](#)).

Figure 14. Sitara "start_nav.sh" Output Screen Capture



```
a0271529@uda0271529: /opt/ros/indigo/share
a0271529@uda0271529: /opt/ros/indigo/share$ ssh root@192.168.0.222
root@am57xx-evm:~# source /opt/ros/indigo/setup.bash
root@am57xx-evm:~# cd /opt/ros/indigo/share/turtlebot_mmwave_launchers/scripts/
root@am57xx-evm: /opt/ros/indigo/share/turtlebot_mmwave_launchers/scripts# ./start_nav.sh
Script will loop until user presses CTRL-C or enters invalid argument...
Starting side? left (l) or right (r) l
Starting position? (a, b, c)? b
Goal position on other side? (a, b, c)? r
Start: l b Goal: r b
publishing and latching message for 3.0 seconds
publishing and latching message for 3.0 seconds
Starting side? left (l) or right (r)
```

Figure 15. Ubuntu Rviz Screen Capture - Initial Left-Side Point B Position

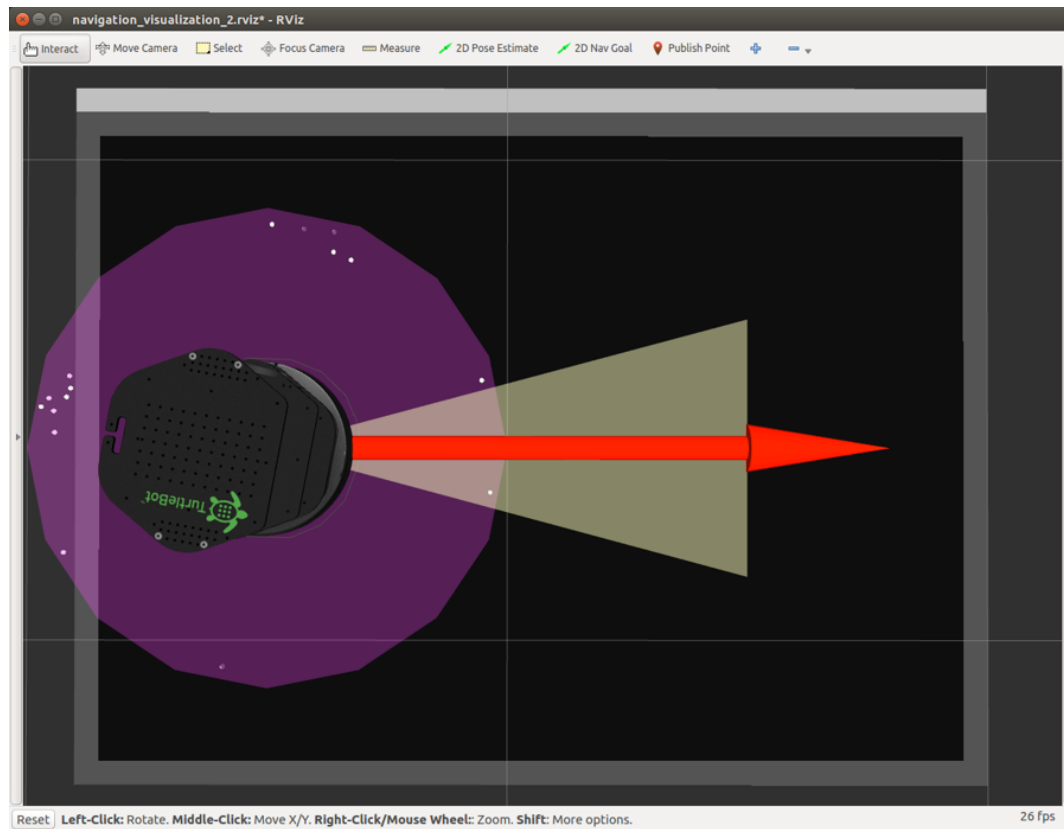


Figure 16. Ubuntu Rviz Screen Capture - Right-Side B Point Goal Reached

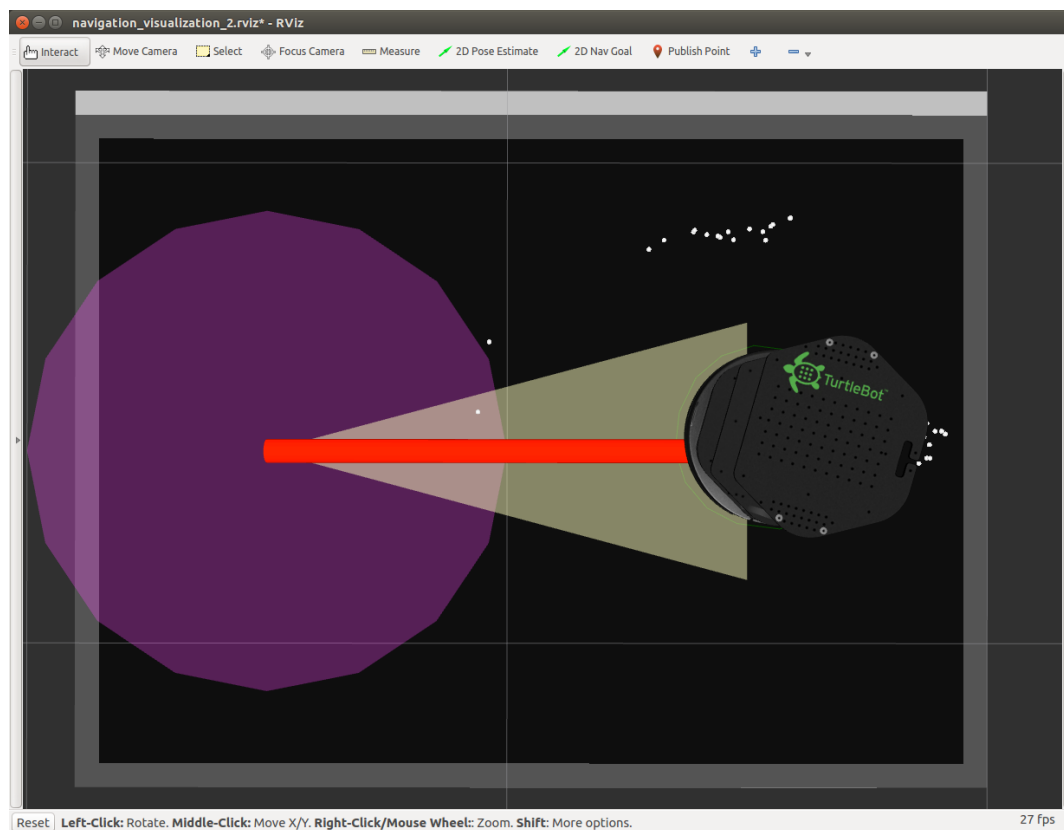
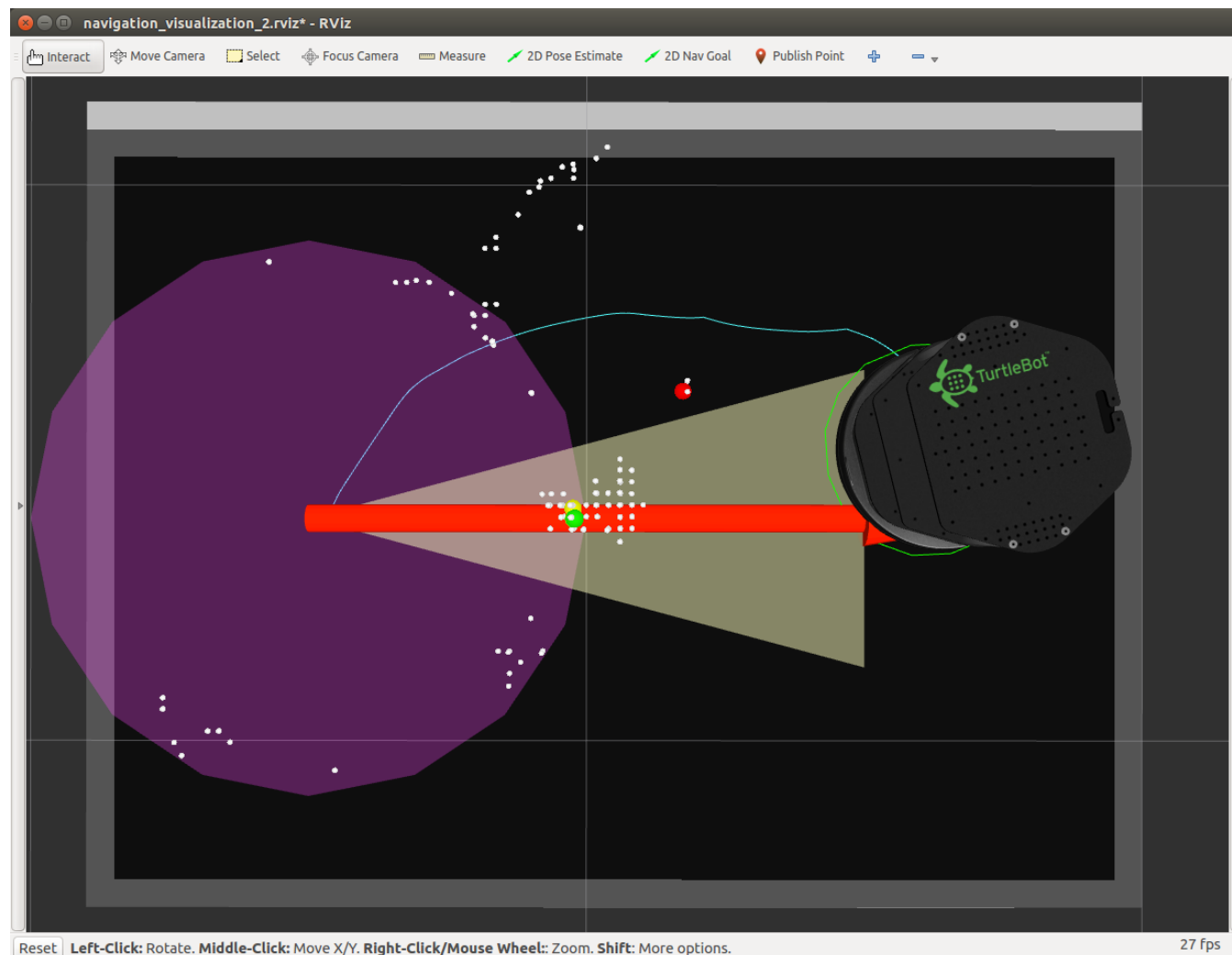


Figure 17. Ubuntu Rviz Screen Capture - Sense and Avoid


3.2.2.1 For IWR1443 User Support

Besides the IWR6843 (operating within 60- to 64-GHz, which is an open band for most industrial applications), the Processor SDK Linux also maintains support for the IWR1443 (operating within a 76- to 81-GHz band) for autonomous robotics.

To use the IWR1443 with this TI design, on the hardware side, use [IWR1443BOOST](#) as the mmWave EVM.

As for the software when using IWR1443, follow the same procedure described in [Section 3.2.2, Test Results](#) but for [step 3](#), set the mmwave_device to "1443" when running the minimal.launch command.

```
roslaunch turtlebot_bringup minimal.launch mmwave_device:=1443
```

4 Design Files

4.1 Schematics

To download the schematics for the AM572x, see the design files at [TIDEP-01006](#).

4.2 Bill of Materials

To download the bill of materials (BOM), see the design files at [TIDEP-01006](#).

5 Software Files

Download the [Processor SDK Linux for AM57x Sitara Processors](#)

Download the [mmWave SDK](#).

6 Related Documentation

1. [IFR Press Release, IFR CEO Roundtable, Munich, Jun 20, 2018](#)
2. [IFR Press Release, Frankfurt, May 30, 2018](#)
3. [Open embedded layer for ROS applications](#)
4. [Texas Instruments, mmWave sensors: industrial toolbox: labs: ROS point cloud visualizer](#)
5. [Texas Instruments, mmWave sensors: industrial toolbox: labs: autonomous robotics with ROS for mmWave](#)
6. [Texas Instruments, AM572x evaluation module](#)
7. [Texas Instruments, IWR6843 intelligent mmWave sensor standard antenna plug-in module](#)
8. [Texas Instruments, mmWave sensors carrier card platform](#)
9. [iClebo Kobuki](#)
10. [ROS and radar in processor SDK Linux](#)
11. [Texas Instruments, mmWave SDK](#)
12. [Texas Instruments, mmWave training series](#)

6.1 Trademarks

Sitara, E2E are trademarks of Texas Instruments.

Arm, Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

All other trademarks are the property of their respective owners.

6.2 Third-Party Products Disclaimer

TI'S PUBLICATION OF INFORMATION REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE AN ENDORSEMENT REGARDING THE SUITABILITY OF SUCH PRODUCTS OR SERVICES OR A WARRANTY, REPRESENTATION OR ENDORSEMENT OF SUCH PRODUCTS OR SERVICES, EITHER ALONE OR IN COMBINATION WITH ANY TI PRODUCT OR SERVICE.

7 Terminology

EVM	Evaluation Module
IFR	International Federation of Robotics
ISK	Industrial Starter Kit
LTS	Long Term Support
PLC	Programmable Logic Controller
PLSDK	Processor SDK for Linux
PRU	Programmable Real-Time Unit
ROS	Robotic Operating System
SDK	Software Development Kit
SoC	System on Chip
SSH	Secure Socket Shell

8 About the Author

Hongmei Gou is a software engineer with the Catalog Processor Group. She received the Ph.D. degree in electrical engineering from University of Maryland, College Park, in 2007. Her recent roles focus on Processor SDK Linux software packages for the Sitara devices, including robotics, machine vision, and industrial applications.

Djordje Senicic is a senior software engineer with the Catalog Processor Group. His recent roles focus on Processor Linux software packages for Sitara and Keystone devices, including deep learning, robotics, and machine vision applications.

Revision History

Changes from November 27, 2018 to January 28, 2019 (from * Revision (November 2018) to A Revision)	Page
• Global: Changed all "IWR1443" references to "IWR6843", where applicable	1
• Global: Changed all "IWR1443 EVM" references to "IWR6843ISK" , where applicable	1
• Global: Changed all "AM5" references to "AM572x", where applicable	1
• Global: Changed all "AM572x" references to "AM572x EVM", where applicable	1
• Global: Changed all "IWR1443BOOST" references to "IWR6843ISK + MMWAVEICBOOST", where applicable	1
• Global: Updated/Changed all figures to reflect the new IWR6843ISK + MMWAVEICBOOST"	1
• Resources: Changed the "IWR1443" Product Folder reference to "IWR6843" and updated the associated ulink.....	1
• Resources: Updated/Changed the "IWR1443BOOST" Tool Folder reference to "IWR6843ISK" and updated the associated ulink	1
• Section 1.1.2 (mmWave Sensor): Updated/Changed the section to reflect the key features for the IWR6843 mmWave sensor	4
• Section 2.2.2 (Add and Enhance ROS Packages ...): Added a sentence with move_base change details	6
• Section 2.2.3 (Add ROS Packages to Enable ...): Changed the Yocto recipe example to be a generalized [version]	7
• Section 3.2.2 (IWR6843ISK and MMWAVEICBOOST Boards): Updated/Changed this section to detail the IWR6843ISK and MMWAVEICBOOST boards within the mmWave EVM Hardware.	9
• Section 3.1.2 (Software/TI mmWave EVM): Updated/Changed paragraph to include the correct mmWave SDK version which supports IWR6843	11
• Section 3.1.2 (Software/Ubuntu Linux Box with ROS Indigo LTS): Added a sentence stating the Ubuntu version used for verifying this TI Design	11
• Section 3.2.1 (Test Setup): Updated/Changed paragraph to discuss the mounting of the IWR6843ISK and MMWAVEICBOOST boards to the Kobuki	11
• Section 3.2.1.1 (Hardware Setup): Updated/Changed the hardware setup to include the mounting of the IWR6843ISK and MMWAVEICBOOST boards to the Kobuki	11
• Section 3.2.2 (Test Results): Updated/Changed the code example command line to include the specific device (6843)	15
• Section 3.2.2: Updated/Changed the screen captures (beginning and end) to show usage with IWR6843	15
• Section 3.2.2.1 (For IWR1443 User Support): Added new subsection to support any IWR1443 users	21
• Section 4.1 (Schematics): Updated/Changed the specified ulink	22
• Section 6 (Related Documentation): Updated/Changed the "IWR1443" evaluation module reference to the mmWave sensor standard plug-in module for the "IWR6843" and updated the ulink	22
• Section 6: Updated/Changed the ulink for the TI mmWave ROS Driver	22
• Section 6: Updated/Changed the ulink for the TI Autonomous Robotics with ROS for mmWave	22
• Section 6: Added the mmWave sensors carrier card platform (MMWAVEICBOOST) title along with the ulink	22
• Section 7 (Terminology): Added the term "ISK" to this section	23
• Section 7: Updated/Changed the sequence order terms to be alphabetical	23

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated