ExFacLab Documentation

SDVUNx: Software

This section contains a summary of the software used in every SDVUNx and some applications used to controlling and monitoring these robots.

Ubuntu: The OS of the brain

Every SDVUNx uses Ubuntu 18.04 as Operating System. Ubuntu is a Linux Distribution created and maintained by Canonical and has a server edition with stable and tested software packages: this server edition is used in SDVUN1, SDVUN2 and SDVUN3. Ubuntu is installed in the on board CPU: this CPU is an x86 processor in SDVUN1, SDUN2 and SDVUN3 and an ARM processor in SDVUN4. Every board contains USB ports, ethernet ports and Wifi cards and can communicate with other computers through these interfaces. USB port allows the communication with the main microcontroller and cameras, if exists. Figure 44 and Figure 45 contains sample pictures of used NUCs.

Fig. 44 Intel NUC: The On Board CPU of SDVUN1, SDVUN2 and SDVUN3

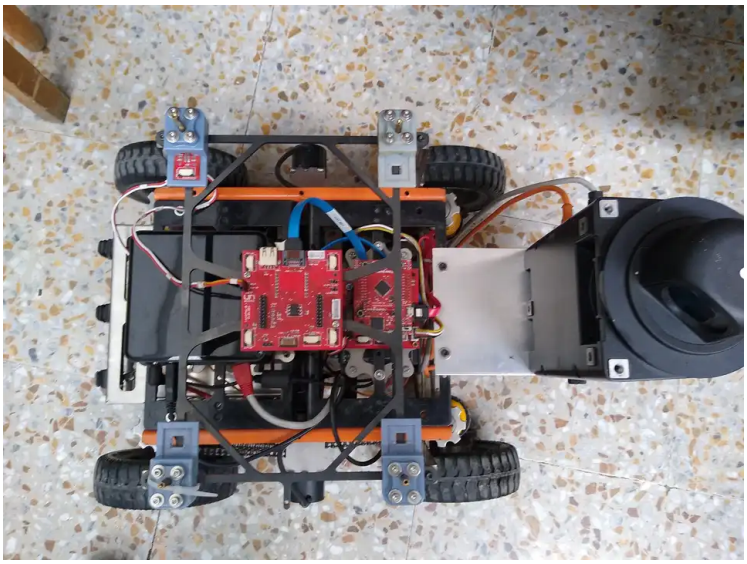

Fig. 45 Top view of the SDVUN1 with the Intel NUC in the left side

Generally its not necessary to use a display to configure Ubuntu, but sometimes, specially if network credentials changes, a display can be connected. For this case, a very simple desktop window manager is installed in Ubuntu: XFCE. This desktop environment is very similar to Windows and uses the sames paradigms like windowed programs, virtual desktops, task bars and menus. If you need more information about this environment, check this link

SSH

To stablish a connection with any of the SDVUNx mobile robots while it's moving, it's necessary to use SSH, and well known program that works in the Command Line Interface. This program allows to use a local computer to send commands to the SDVUNx. Also, SSH allows to use some programs with GUI, but you need to keep in mind that network resources of the WiFi is shared with other machines and users and using a GUI needs a high bandwidth consumption.

Fig. 46 SSH sample: connecting to SDVUN2

ROS: coordinator of the internal tasks

ROS is an Open Source robotics framework (also is called as Meta Operating System) that contains many tools to build and run programs that can control robots. Until now, ROS Melodic (a.k.a ROS 1) is used to maintain compatibility with developed software.

ROS works using “simple” programs that communicates with others using a communication channel with a well defined message structure. Every program is called *node* and the channels are called *topics*.

ROS provides the *Navigation Stack*, a set of nodes that allows to a mobile robot to navigate using odometrics, IMU data and laser scans. The Navigation Stack use all the sensors data to calculate it’s real pose inside a known environment. The Navigation Stack nodes are standard and all the developed code that belongs to ROS framework its mostly configuration files needed to adapt these nodes to the robot parameters.

Besides the standard nodes, other programs runs in ROS. These nodes works to communicate the *on board CPU* with the main microcontroller (TIVA), listen changes in PRIA and share its position to *SDV-Nav-Service* backend, mounted in the ExFacLab server.

Tiva Firmware: The brute force

The Tiva-C Launchpad TM4C123 is the main microcontroller of the robot. It communicates with the *on board CPU* through an USB 2.0 port and receives control messages using serial port. The Firmware of the Tiva was developed using Energia IDE, an Arduino fork that makes the programming tasks easier.

The Tiva also communicates with other peripherals:

- Motor drivers using GPIO pins
- IMU, using I2C bus
- Button, buzzers, LEDs, ultrasound sensors and Flexiforce sensors, also using GPIO pins
- Sets drivers using GPIO pins (in SDVUN1, SDVUN2 and SDVUN3)
- Reads the motor encoders with GPIO pins (only in SDVUN4)
- Reads the Battery Monitors using I2C (only in SDVUN4)

Tiva don’t makes any superior task like calculate position or correction of pose: only moves the robot and reads the sensors. Its Firmware acts like an small Operating System because uses timers to reads every sensor, checks the peripherals and reports the information to the host. You can move the robot with ROS but without the Navigation Stack, sending simple speed messages to *sdv_serial* node. Figure 47 shows the SDVUN3 with its Tiva board.

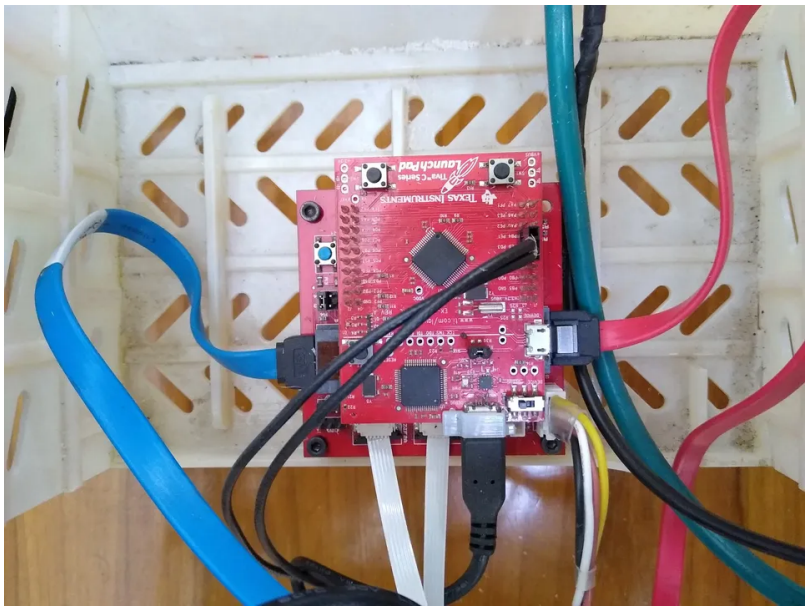

Fig. 47 Tiva inside the SDVUN3

PRIA: The messenger between robots

PRIA is a high level communication framework used to send tasks to multiple robots in the ExFacLab and is composed by a set of programs that runs in multiple devices. SDVUNx uses the *ros_coms* package to communicate with PRIA. This package is also used in other machines: it was developed to be agnostic and any ROS based robot can use it.

A remote Real Time Database mounted in Firebase is the central database of PRIA. All robots that needs to make collaborative tasks with others needs to read the remote database. For this reason, its necessary that any SDVUNx has internet connection.

To send commands through PRIA, you can use NODE-Red, an Open Source tool that allows to create collaborative robot tasks using nodes and links between these nodes.

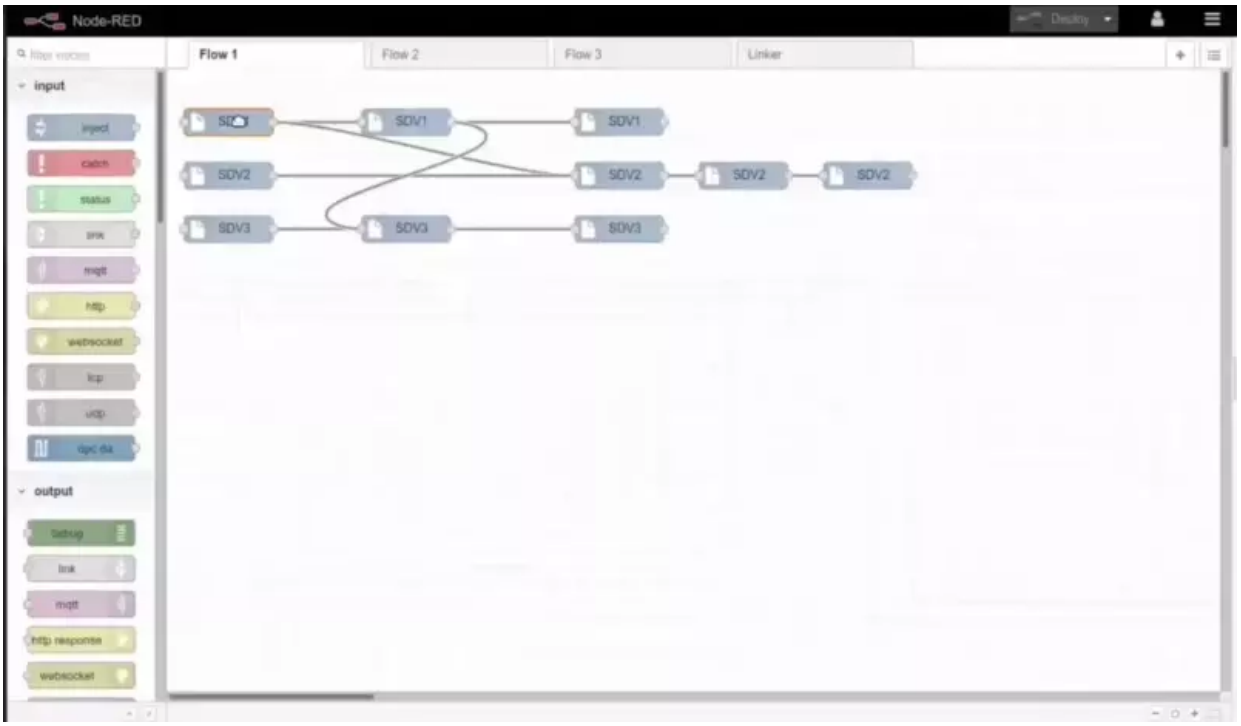

Fig. 48 Using Node-RED with three SDVUNx

SDV-Map-Viewer: a tool to control all the robots in one application

SDV-Map-Viewer is a Single Web Application that can send direct ROS position commands to any active SDVUNx. This application its mounted in the ExFacLab server and can be used in any external computer. To work, every SDVUNx shares its pose with a service called *Sdv-Nav-Service* that runs in the ExFacLab server and talks with the robots using a ROS bridge: this service shares this information with any connected client of *SDV-Map-Viewer* and sends commands to every SDVUNx using the same bridge.

This Web Application can works lonely, but to avoid the fragmentation of different projects, is embedded in *PRIA-Web-App*, another Single Web Application that uses Firebase as Authentication Service and can read the Firestore RTDB. Using these features, *SDV-Map-Viewer* is more secure and accessible.

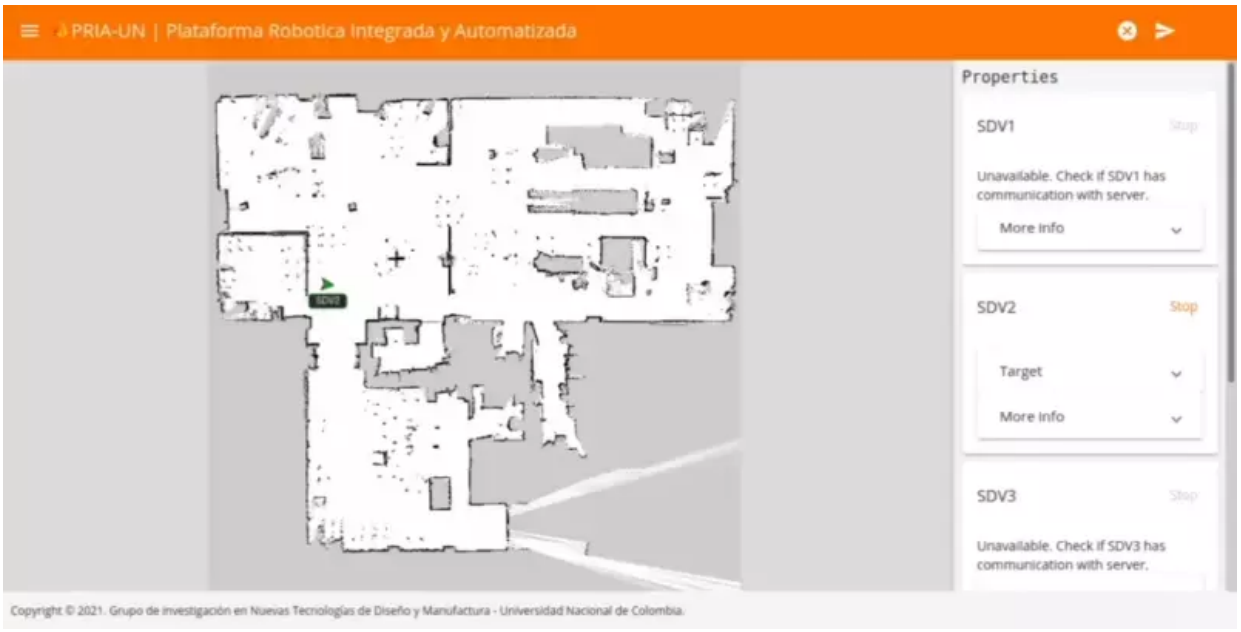

Fig. 49 SDV-Map-Viewer sending position commands to SDVUN2 while shows its current pose

SDVUNSIM: Simulating the ExFacLab

ROS is compatible with Gazebo, a robot simulation framework that uses a realistic physics engine for recreate robots in its environment. The *sdvun_sim* package is a set of ROS packages developed to simulate the SDVUNx and the ExFacLab. This simulation uses 3D models of every robot: this models contains approximated inertial properties of every robot. In the ROS side runs the Navigation Stack configured as is in the real robots. To get more information, see SDVUN_SIM.

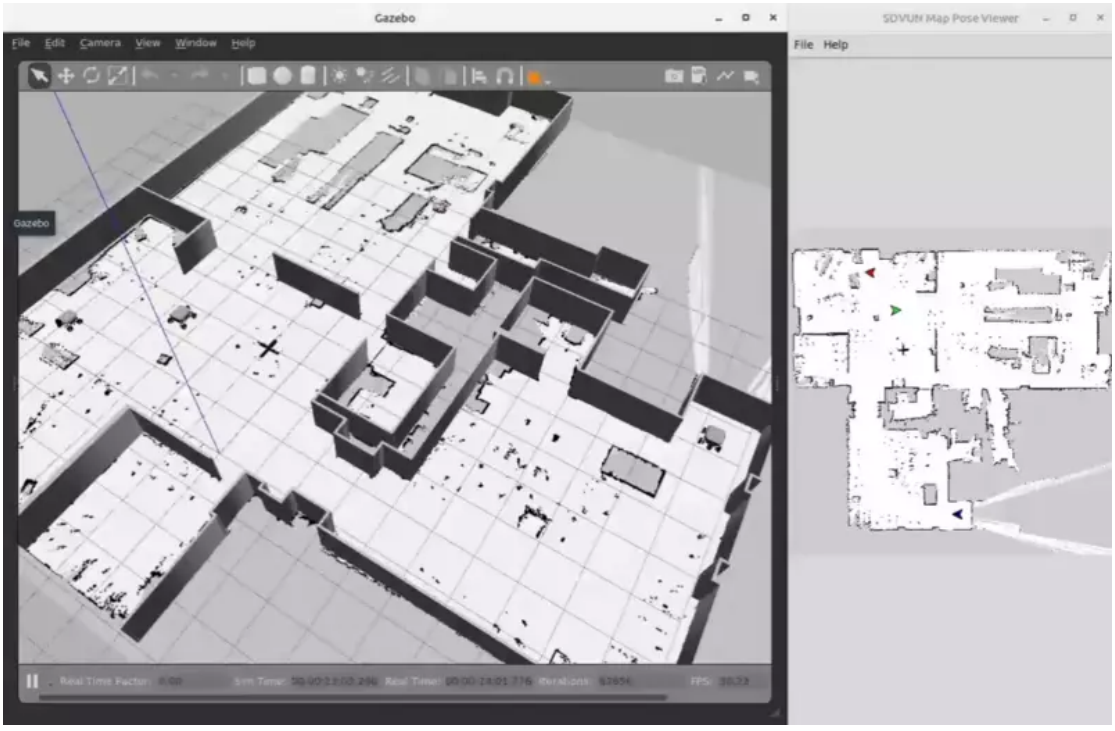

Fig. 50 sdvun_sim running three SDVUNx commanded with PRIA`