MOBILE ROBOTICS PLATFORM FOR EXPERIMENTS USING ROS: DEVELOMENT OF THE REAL – TIME BOARD

[1] career of Mechatronics Engineering, FICA, Technical University Northern, Av. 17 de Julio, Ibarra, Ecuador
e-mail: edwinlb20@gmail.com

Abstract

The field of robotics comprises a number of technologies and systems for its operation and control. The robotics branch that has evolved the most in recent years is mobile robotics. Mobile robots are machines capable of moving in any environment without being fixed to a single physical location. In addition, they perform complex movements that are performed in real time, occur with predictable planning and must be performed within a defined time frame.
At the Universidad Técnica del Norte, no mobile robotics tools have yet been developed to visualize, understand and apply the theories and fundamentals of robotics subjects and control systems. Based on the aforementioned aspects, the idea was born to realize this same project that was realized with educational aims to contribute to the area of Applied Sciences of the University. Connecting real-time integrated systems to ROS is a way to take advantage of the higher-level capabilities of this meta-operating system. For the development of this project, the mobile robot was implemented through the robotic platform (ROS), which provides libraries, visual software applications and communication tools. It was run on a computer with open source applications. For the part of the real time system a microcontroller and an ultrasonic sensor were used that are communicated by nodes to the computer.

Introduction

Robotics is experiencing explosive growth driven by advances in computing, sensors, electronics and software [1]. Robots are already revolutionizing the procedures that are used in medicine, agriculture, mining and transportation, by solving most of their control and automation needs. In this way, there are innumerable applications that day by day improve the daily life of society.
Currently there are several Frameworks of Development in Robotics such as Player / Stage / Gazebo, Yarp, Orocos, OpenRave among others, all open source, however, Robot Operating System (ROS) has managed to group the best features of all these projects giving a comprehensive and very uniform solution to the problem of development of robotic systems, is characterized by being a multi-language platform (c ++, python, java), peer2peer, tool-oriented, lightweight and open source (OpenSource)]. A system built using (ROS) provides the standard services of an operating system such as hardware abstraction, low-level device control, implementation of commonly used functionality, message passing between processes and packet maintenance [3]. It is based on a point-to-point architecture where processing takes place at nodes that can receive, deliver and multiplex messages from sensors, control, states, schedules and actuators, among others.
The realization of this project is based on a need which consists, at present, the Technical University of North and particularly the Faculty of Engineering in Applied Sciences, there are no mobile robots to be used as teaching and research equipment, a situation that repercussions on subjects related to robotics and control, making them purely theoretical.
Based on the described problem, we believe it imperative to implement a mobile robot for experiments in robotics that is used as teaching and research equipment in FICA laboratories**.**

Keywords
ROS (mobile robotic platform), real-time, ultrasonic sensor, distance,

Methodology

Before the development of the project begins with a small definition and with fundamental concepts for understanding the theme such as: real-time systems, ros and the tools that each of them gives us.

1. Real-time system

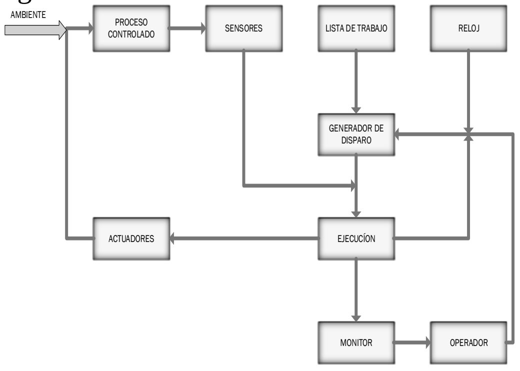
A system is the set of components that work together to achieve a common purpose. Real-time systems are systems based on a computer that must solve different aspects simultaneously, fast response, reacts to stimuli, failure in components or their connections and possible needs to adapt over time (while in operation) before the changes of requirement and circumstances.

A real-time system must meet three fundamental conditions:

• Interact with the real world.
• Emits correct answers.
• Complies with temporary restrictions.

General structure of a real-time system

The general structure of a real-time system that controls any process is shown in the figure.



Real-time system applications

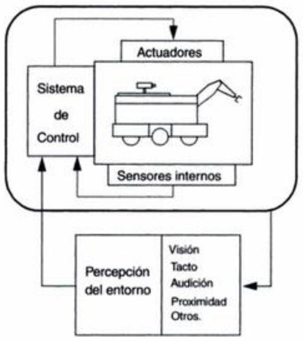
There are many applications where real-time systems are used:

- Defense systems
- Radar systems
- Process control
- Aviation air traffic control multimedia servers
- Comunication system
- Satellite systems
- Signal processing systems
- Autonomous navigation systems
- Data acquisition and control systems. Etc...

An important group of real-time systems are embedded systems. They are systems designed for a specific application. For example a mobile phone, abs control in a car, etc. However a personal computer would not be an embedded system since it was not designed to make a specific application. Thus the assertion that every embedded system is a real-time system is true, whereas the contrary statement is not. [4]

Real-time robots

A robot is a machine that performs productive work and imitation of movements and behaviors of living beings. Currently robots are engineering works that are composed of mechanical systems, actuators, sensors and control systems. The latter is important since it is in charge of obtaining the information from the outside by means of the sensors, interpreting the data and sending the information to the actuators. The figure details the robot its elements and the interaction with the environment.



In industry, robots are increasingly indispensable in production processes and with a growing population the demand for products requires them to be faster in collecting, interpreting and sending data to make them more efficient, in addition, they can to perform actions in response to eventualities that can arise in their environment in very short periods of time by them is necessary the implementation of systems in real time for the robot. [5]

The advantages of using real-time systems in robots are: its greater accuracy, security and adaptability to various contingencies of its environment and in each of the processes that are performing as it is continuously acquiring information from the external environment.

2. Robot Operating System (ROS)

The platform (ROS) is a framework for the development of control algorithms in robotics. It also has packages that include control libraries of devices for actuators, sensors, motors. It is open source, this license allows freedom for commercial use and for research, it is also currently

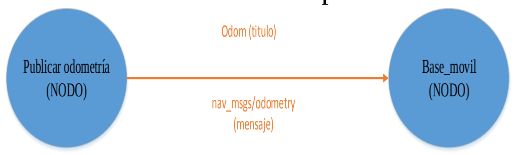
supported by Unix-Ubuntu and has experimental support for Mac, Fedora, Windows, OpenSuse among others.
The main advantage of this type of communication is the creation of large databases for free since all computers connected online can download files from other computers also connected.
(ROS) has two main parts for its smooth operation: the operating system core and the ros-pkg. The latter a group of open source license packages, developed by users that implement different functionalities such as mapping, planning, etc.

Main Features of the Robots Operating System (ROS)

The main characteristics of ROS are:
Free and Open Code.- (ROS) is free code under BSD license terms, contains freedom of use both commercial and research that means that you are available to the general public. [6]
"Peer to Peer".- It is a distributed system in which the different processes communicate with each other using a "peer to peer" topology. With this topology a new channel is used for the communication between two different processes, avoiding to use a central server to communicate all the processes. [6]
Multi-Language.- It can work in different programming languages such as: C ++, Phyton and Lisp. It also contains libraries in Java and Lua, in experimental phase.
Multi-Tools.- It contains a large number of tools, which allows us to perform different tasks, from browsing the files, also to modify the configuration parameters of a robot, process or driver, observing the topology of the running processes and performing communication between processes. [6]

Basic Concepts of the Robots Operating System (ROS)

To obtain an adequate operation of (ROS) you must have knowledge of the fundamental elements of this platform such as: nodes, ROS Master, Messages and topics.
Nodes.- They are executables that communicate with other processes using topics or services. The use of nodes in (ROS) provides fault tolerance and separates the system code making it simpler. A packet can contain several nodes (each node has a unique name), each of them performs a certain action.
ROS Master.- Provides a register of the nodes that are running and allows communication between nodes. Without the master the nodes would not be able to find the other nodes, exchange messages or invoke services.
Messages.- The nodes communicate with one another by means of the message passage. Primitive message types ("integer", "floating point", "boolean", etc.) are supported and custom types can be created. [2]
Topics.- They are channels of communication to transmit data. Topics can be transmitted without direct communication between nodes, meaning the production and consumption of data is decoupled. Nodes can communicate with each other through topics, being able to act as publishers and as subscriber.
Publisher.- The node that acts as publisher is in charge of creating the topic for which it is going to spread certain messages. These messages can be visible by the nodes that are subscribed to this topic. [2]
Subscriber.- This node must be subscribed to the corresponding topics in order to access the messages that are published in them.
A node can post or subscribe to a message through a topic. Figure 6 shows an example of communication between two nodes by means of a topic. In this example, the "publisher odometry" node publishes a "nav_msgs / odometry" message in the topic "Odom" and the "Base_movil" node accesses this message by subscribing to this topic. The following figure details the communication between topics.



Real time in ROS

Despite the importance of reactivity and low latency in robot control (ROS), it is not a real-time operating system (RTOS), although it is possible to integrate (ROS) with real-time code. To do this, it must fulfill certain requirements for its proper functioning.
The requirements of a real-time system vary depending on the use case. In essence, the real-time requirement of a system has two components:

a. Latent state
- Update period (also known as deadline)
- Predictability

b. Fault mode
- How to react to a deadline [6]

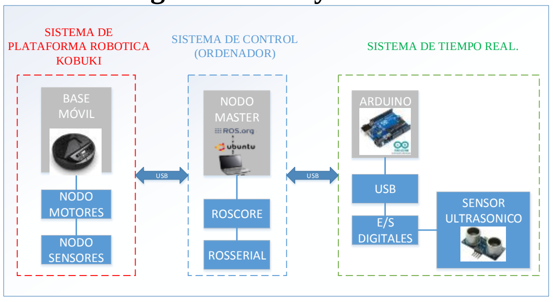
When referring to hard / soft / firmer systems in real time it generally refers to the failure mode. A hard real-time system treats a deadline as a system failure. A soft real-time system tries to meet deadlines, but does not fail when a deadline is missing. A firm real-time system discards calculations made for non-compliance with deadlines and may degrade your performance requirement in order to accommodate an unfulfilled deadline.
A real-time failure mode is often associated with high predictability. Critical security systems often require a real-time system with high predictability.

Solution to the problem raised

The Kobuki robotic platform has many sensors as they are: impact sensors are three that allows you in the instate to get to have contact with an object the robotic platform stops immediately, sensors are three levels of these sensors are located under the bumper that are infrared that serve to detect if it can follow by the direction anticipated otherwise it will take another route, sensor of fall of wheel are two are switches of two positions with return spring to its initial potion and are actuated by means of a lever. They are located on the inside of the wheels. However none of these sensors send a signal of the objects that surrounds it without colliding for this reason it has been convenient to implement a sensor that gives an early warning and so the operator makes avoidance decisions avoiding obstacles on either the robotic platform as in its environment without the need for the operator to be located near the robotic platform.
An ultrasonic sensor was selected as it provides a simple method of distance measurement. This sensor is perfect for any number of applications that require measurements between moving or stationary objects.
Consisting of ultrasonic transmitter, receiver and control circuits, when it is triggered, it sends a series of 40KHz ultrasonic pulses and receives an echo from an object. The distance between the unit and the object is calculated by measuring the travel time of the sound and its output as the width of a TTL pulse. [7] An ultrasonic sensor is shown in the following figure.

This sensor can be coupled to different applications such as: security systems, interactive animated exhibitions, parking attendance systems and robotic navigation.

General diagram of the system



Robotic Platform System (Kobuki)

IClebo Kobuki is a low-cost mobile research base designed for education and research on the state of art robotics. With continuous operation in mind, Kobuki provides power supplies for an external computer, as well as additional sensors and actuators. Its highly precise odometry, as amended by our gyroscope calibrated in the factory, allows accurate navigation.
The Kobuki is a mobile base. It has sensors, motors and power sources, however by itself, it can not do anything.
To be functional, it is necessary to build a platform above the command sheet. On the hardware side, this usually involves the addition of a notebook or a built-in board to be the computational core for your system and usually some extra sensors to make it really functional. On the software side, this involves building any proprietary software or integrating software from other groups with their own developmental sources.

Control System (Computer)
This system is composed of software as hardware. The software comprises of an Acer brand computer that meets the requirements necessary for optimal operation of the robot.
The hardware that will be used to carry out the project, is the operating system Ubuntu 14.04 LTE. For the good operation and development of the project the choice of the operating system is fundamental. Bearing in mind that (ROS) can also be installed on Windows, applications developed (Packets) must meet certain computational limitations. Also, it should be mentioned (ROS) is only fully functional on the Linux platform fundamentally for the Ubuntu distribution; with other distributions do not guarantee the correct operation of (ROS).
To conclude, Linux has been chosen, specifically the distribution Ubuntu 14.04 LTS, whose origin is based on Debian, in addition was chosen this version because it is LTS (Long Term Support), this means that it is a version of Ubuntu that will be supported and will be updated more time than a normal version.
Likewise, LTS versions are usually more stable and tested versions. In addition, the LTS versions of Ubuntu have support for 5 years and the normal versions have a support of 9 months, after that time will have to be updated in a relatively short period of time.
In the interior of the robotic operating system (ROS) two extremely important nodes are being executed for the communication between the computer and the system in real time these are: roscore and rosserial.

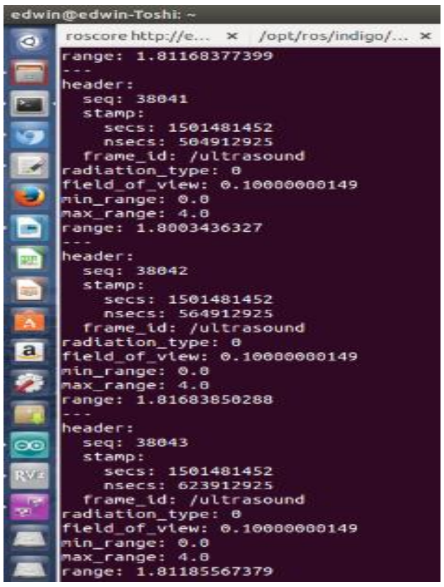
Roscore
It is a set of nodes and programs that is a precondition for the use of a system based on (ROS). A roscore must be executed in order that (ROS) can communicate with all the nodes. It starts with the roscore command.

Rosserial
ROS Serial is a point-to-point communications (ROS) over serial, mainly for the integration of low cost microcontrollers (Arduino) in (ROS). ROS series consists of libraries for use with Arduino, and nodes for the PC / Tablet side (currently in Python and Java). [9]

Real Time System
For the part of the system in real time was used the Arduino microcontroller and the ultrasonic sensor that are connected by means of cables, and the two elements is communicated to the computer by means of UBS.
Sampling frequency of the ultrasonic sensor is 1 samples per second.

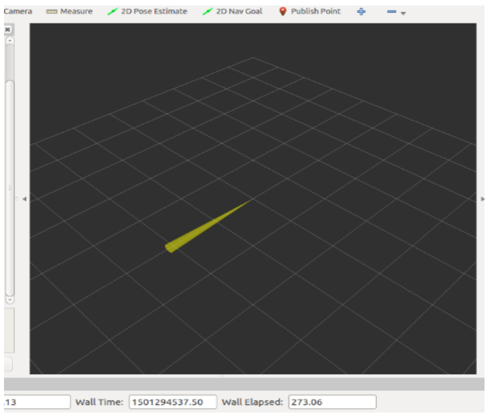
Testing, Results and Analysis
For the tests the following steps were performed:

a. Load the program that was made in the Arduino software to the Arduino board one.
b. It opens a new terminal in Ubuntu and runs roscore.
c. The code is executed in a new terminal
   roslaunch kobuki_node minimal.launch --screen to power the kobuki robot.
d. The code is entered in a new terminal
   roslaunch kobuki_node minimal.launch -screen that allows the robot to tele-operate.
e. Using the command ls / dev / tty and pressing the tab key twice, prints all the ports that are being used. You can check if the Arduino is connected to a computer.
f. Then the command is executed:
   rosrun rosserial_python serial_node.py / dev / ttyUSB0
   This command runs the rosserial client application that forwards its messages from the Arduino to the rest of (ROS).
g. Then the command is executed:
   rostopic list
   This command allows us to print a list of the topics to which you can subscribe.
h. Running the following code
   rostopic echo / ultrasound
   It prints the information of the ultrasonic subject that for the case is the Ultrasonic sensor towards the computer by the serial port, that is visualized in the terminal.



```
edwin@edwin-Toshi: ~
roscore http://e...  ×   /opt/ros/indigo/...  ×
range: 1.81168377399
---
header:
  seq: 38041
  stamp:
    secs: 1501481452
    nsecs: 504912925
  frame_id: /ultrasound
radiation_type: 0
field_of_view: 0.10000000149
min_range: 0.0
max_range: 4.0
range: 1.8003436327
---
header:
  seq: 38042
  stamp:
    secs: 1501481452
    nsecs: 564912925
  frame_id: /ultrasound
radiation_type: 0
field_of_view: 0.10000000149
min_range: 0.0
max_range: 4.0
range: 1.81683850288
---
header:
  seq: 38043
  stamp:
    secs: 1501481452
    nsecs: 623912925
  frame_id: /ultrasound
radiation_type: 0
field_of_view: 0.10000000149
min_range: 0.0
max_range: 4.0
range: 1.81185567379
```

i. To visualize in a graphical way can be done by executing the next catch.
rosrun rviz rviz
What the simulator opens and using a yellow beam represents the distances currently being sent by the sensor.



Analysis and Results

With this test it was verified that the sensor is sending data to the microcontroller, and that the microcontroller in turn is sending them to the computer.

Conclusions and Discussion

The Kobuki mobile robotic platform has the advantage of being an omnidirectional robot that allows to perform displacement and rotation movements facilitating the implementation of algorithms for teleoperation.

The configuration of the system was based on three fundamental segments that are: Kobuki robotic platform system, Control system, Real time system. Those who worked at par could solve the problem.

The implementation of the software with the hardware was done quickly thanks to the facilities provided by the Kobuki robotic platform.

Bibliography

[1] A. O. Baturone, Robótica: manipuladores y robots móviles. Marcombo, 2005.

[2] A. Koubaa, Robot Operating System (ROS): The Complete Reference. Springer, 2017.

[3] «ROS.org | Acerca de ROS». .

[4] Manuel Ortiz, «SISTEMAS INFORMÁTICOS EN TIEMPO REAL».

[5] A. Araújo, D. Portugal, M. S. Couceiro, J. Sales, y R. P. Rocha, «Desarrollo de un robot móvil compacto integrado en el middleware ROS», Rev. Iberoam. Automática E Informática Ind. RIAI, vol. 11, n.º 3, pp. 315-326, jul. 2014.

[6] «ROS/Introduction - ROS Wiki». [En línea]. Disponible en: http://wiki.ros.org/ROS/Introduction. [Accedido: 16-sep-2017].

[7] «Sensor de Distancia de Ultrasonido HC-SR04», Electronilab. .

[8] «The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu». [En línea]. Disponible en: https://www.ubuntu.com/. [Accedido: 16-sep-2017].

[9] «rosserial». [En línea]. Disponible en: http://library.isr.ist.utl.pt/docs/roswiki/rosserial.html. [Accedido: 13-sep-2017].