Tekhnê
January - June 2017, Vol. 14, No. 1, pp. 61 – 72

ISSN 1692-8407

Performance evaluation of ROS on the Raspberry Pi platform as OS for small robots

Evaluación del desempeño de ROS sobre la plataforma Raspberry Pi como OS para pequeños robots

Andrés Moreno N.
Universidad Distrital Francisco José de Caldas
amorenon@correo.udistrital.edu.co

Daniel F. Páez C.
Universidad Distrital Francisco José de Caldas
dfpaezc@correo.udistrital.edu.co

This article presents the configuration and programming of the SERB robot for use in navigation applications, using as hardware support the Raspberry Pi Model B card, and as software the ROS OS Grovvy Galapagos. The designed robotic application uses different elements such as actuators (servomotors), sensors (proximity sensor), and an external control card (Arduino). We show the OS configuration on the platform and its performance with some basic navigation tasks.

Keywords: Path planning, ROS OS, Raspberry Pi, SERB robot

Este artículo presenta la configuración y programación del robot SERB para su uso en aplicaciones de navegación, utilizando como hardware soporte la tarjeta Raspberry Pi Modelo B, y como software el ROS OS Grovvy Galapagos. La aplicación robótica diseñada utiliza diferentes elementos como actuadores (servomotores), sensores (sensor de proximidad), y una tarjeta externa de control (Arduino). Se muestra la configuracion del OS sobre la plataforma y su desempeño con algunas tareas básicas de navegación.

Palabras clave: Planeación de rutas, ROS OS, Raspberry Pi, robot SERB

Article typology: Research

Date manuscript received: May 26, 2017
Date manuscript acceptance: June 30, 2017

Research funded by: Universidad Distrital Francisco José de Caldas.

Digital edition: http://revistas.udistrital.edu.co/ojs/index.php/tekhne/issue/view/798
How to cite: Moreno, A., Paez, D. (2017). *Performance evaluation of ROS on the Raspberry Pi platform as OS for small robots*. Tekhnê, 14(1), 61 -72.

Introduction

The present project was developed within the mobile robotics line of the ARMOS research group of the Universidad Distrital Francisco José de Caldas - Facultad Tecnológica. It corresponds to an evaluation of the feasibility of using ROS OS on the Raspberry Pi platform. This article, product of this study, specifies the different tools that were taken into account for the navigation, exploration, and displacement of the SERB Robot on a designed platform (Castiblanco & Martínez, 2014; Jacinto, Giral, & Martínez, 2016).

For the development of this application, we implemented as hardware support the Raspberry Pi card that for its low cost has become very popular in schools in England and some research centers worldwide. That is why studies have been conducted to take full advantage of the capacity of this card in various areas (Jacinto, Montiel, & Martínez, 2016).

To mention some relevant examples, we can talk about a study at the University of North Carolina, United States, in which data from the Raspberry Pi card was transmitted via Bluetooth to a touch screen. The Raspberry Pi was controlled by a modified version of the free software Debian Linux, which was optimized for ARM architectures. The screen had a graphical interface that could be controlled by users who could send data via a keyboard attached to the screen (Sundaram et al., 2013).

Another interesting case is the one of the University of Sao Paulo, where an open and low-cost code was presented for users who consume electronics. This study validated a generic *middleware* platform, to be tested on a Raspberry Pi model B card, where the results obtained show the study as an option or alternative for use as open source and low-cost (Calixto, Hira, Costa, & Deus, 2013).

A navigation system with 3-D audio systems for the visually impaired was developed at the University of New York City, USA. This consists of a special headset and microphone that are synchronized with a compass, gyroscope, and GPS system. Blind users can interact with this system, telling them the route and direction they want to go through the microphone. These commands are then processed by the Raspberry Pi card, which finally transmits to the user the correct route to follow (Jizhong, Ramdath, Iosilevish, Sigh, & Tsakas, 2013). This is a case of great interest in stand-alone navigation applications.

Also at the Institute for Real-Time Learning Systems of the University of Siegen, Germany, an architecture was designed to link the programming language BML (Programming Language used in the systematization of robots in the military field) with the operating system ROS. It was demonstrated that it was possible to execute orders by means of BML that were processed by ROS and thus be able to assign specific tasks to a group of robots (Remmersmann, Tiderko, Langerwisch, Thamke, & Ax, 2012).

The emergence of the ROS operating system made possible a new low-cost navigation and programming technique on various standard robot platforms. Thanks to this, several studies and researches have been carried out on new methods of application of the ROS operating system in a variety of devices with innovative programming languages. Some of these studies are presented below:

- In the Trinity University, United States, a study in sixteen varieties of ROS OS was carried out in order to determine the most suitable in basic tasks of robotic movements, in order to be used in future robotic investigations for the same students of the Trinity University. The results obtained showed that the best ROS OS for robotic movements or displacements was the ROS Player/Stage, which is also easy to manipulate and navigate (Kerr & Nickels, 2012). The ARMOS research group has adopted Player/Stage as a simulation platform in its research projects.
- An architecture in BML *Behavior Markup Language*, which was based on Petri networks (useful in the design of hardware and software systems, for the specification, simulation and design of various engineering problems), was carried out at the Worcester Polytechnic Institute, United States, in order to ensure that errors are not generated that may appear in the robot control systems when synchronized with the operating system ROS (Holroyd & Rich, 2012).
- At Laval University, Canada, an open source library was created for academic use that is simple and flexible for communication between ROS and Matlab. This library allows easy integration between sensors and actuators of a robot with Matlab code. This library was also designed for a quick connection between a robot being run by ROS and users who handle Matlab code (Hold-Geoffroy, Gardner, Gagne, Latulippe, & Giguere, 2013).
- The University of Wisconsin-Madison presented the programming language *Robot Behavior Toolkit*, implemented in open source as an ROS module, which focused on an analysis of the behaviors of people in the community environment, so that later this type of behaviors or expressions can be reflected and simulated, in some movements of the robots, where they especially worked on the expressions of the gaze (Chien-Ming & Mutlu, 2012).

All this research points to the versatility of both hardware and software for professional training tasks in engineering and research. There are many more documented cases, however, those cited here are considered as fundamental concepts for the development of this project.

ISSN 1692-8407

Problem formulation

For applications in navigation of small robots with limited hardware/software resources, it is normal to use hardware platforms such as Raspberry Pi, Arduino, BeagleBoard, Mbed boards, etc. It is also common the low-level programming of these systems, which is perfect to perform tasks of medium complexity. However, when the task becomes complex, you want to evaluate many different strategies or you work with swarms of robots, the solution at this level becomes complicated.

In this sense, considering the limitations of the hardware, but at the same time the advantages provided by a development interface with ROS OS, the concerns arise, can you program an interface between the Raspberry Pi card model B and ROS OS? If there is an interface between the hardware (Raspberry Pi) and software (ROS OS), can you design a robotic application with acceptable performance? Does the robotic application between the two tools meet aspects such as the decrease in resources, the performance of the robot to perform the task (navigation), and compatibility?

Methodology

Setup of Raspberry PI platform

In the development of the robotic application in navigation, several elements were used. The first step consists of the initial configuration of the Raspberry Pi board, where an operating system compatible with Raspberry Pi is installed.

To download the operating system image an 8 Gb Micro SD is used, which will perform the ROM memory task on the board. The image that is downloaded and executed can be found at `www.raspberrypi.org`, where the NOOBS option is chosen. The main characteristic is the affinity with Archlinux, OpenELEC, Pidora, RISC OS, RaspBMC and Raspbian. To implement the operating system or image on the board it is necessary to choose the option shown in Fig. 1.

Figure 1. Image selection.

Raspbian image

The next step is to insert the memory into the Raspberry Pi, where they also connect the different peripherals that are: mouse, keyboard (which are connected by multiple USB inputs), screen that is connected through HDMI cable, where the card is powered through a cell phone charger of 5 V and 0.7 A, as shown in Fig. 2.

Figure 2. Development board connection.

Raspbian setup

Once the installation script is executed, the initial Raspbian menu will be accessed, where nine different characteristics or specifications for the correct functioning of the image on the card are observed. The menu is shown in Fig. 3.

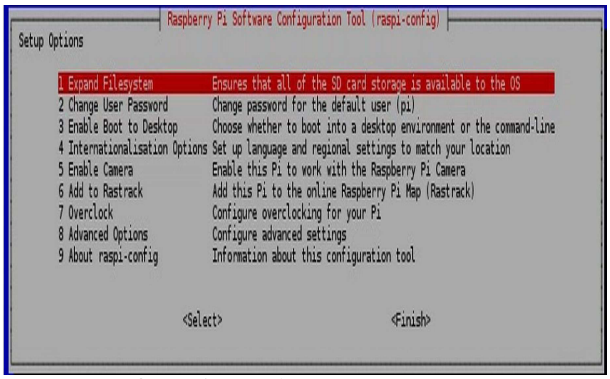

Figure 3. Configuration tool.

- The first option is to expand the memory of the Micro SD. And for the case of the running project is not necessary, because the NOOBS image is already predetermined to fully occupy the memory space.
- In the second option, it is possible to change the password to enter the system.
- The third option determines if it is required to access the graphic mode of the system or directly to the command line, in this case, it was determined to access the graphic mode which is done through the command `startx` (Xserver).

• The fourth option consists of specifying the time zone, language, and keyboard layout. I1 specifies the language, where it is recommended to select UTF-8 which corresponds to Colombia (coding that allows ñ and accents). Option I2 specifies the Colombia - Bogota zone, the keyboard configuration is the standard so option A3 *"PC indicates generic keyboard with 105 keys (intl)"*.

• The fifth option is to activate `Ctrl + Alt + Backspace` to close the graphics mode, where the default setting is chosen.

The above specifications are sufficient to enter the Raspbian graphics mode. The last request to enter the system is to type the username and password, codes specified below:

```
Usuario: pi
Contrasena: raspberry
startx
```

Wi-Fi connectivity

For easy control of the Raspberry Pi board, it must be connected to the Internet, and for this, there are two ways to connect it by UTP cable or Wi-Fi antenna. For the present project, the Internet configuration is done through Wi-Fi antenna. Below is a description of the necessary specifications for its configuration.

• Before starting Raspbian, the Wi-Fi antenna is connected to the Raspberry Pi board.

• When accessing the graphical interface, the Wi-Fi Config icon is accessed. The Wi-Fi device is recognized by the active USB option.

• When clicking on the scan option, whose function is to identify the different available networks (the Wi-Fi network in which the configuration is going to be made must be active or with the option visible), we verify the options and choose the appropriate network, in which the key is entered.

• The following window shows the data of USB port, Wi-Fi network and the IP address handled by our Modem or cell phone. This information is necessary for connection and control of the Raspberry Pi board remotely, as shown in Fig. 4.

Remote connection using SSH

As its name indicates is the option to operate remotely the Raspberry Pi board, this means that the connection of the different peripherals are not necessary. Because the programming will be carried out by means of a computer connected to the same Wi-Fi network with which the card works, this configuration is shown below:

• The following command is entered in the terminal window:

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

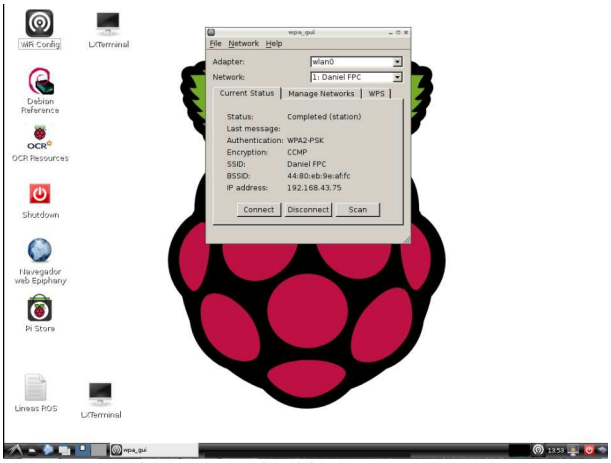

Figure 4. Configuration of Wifi on Raspberry Pi.

• The service is started with the following command:

```
sudo /etc/init.d/ssh start
```

• And in order for the execution of the previous command to be carried out immediately when starting to work with the board, the following command is executed:

```
sudo update-rc.d ssh defaults
```

The following will be the installation of the SSH client in Windows, for this, you must download and install the PuTTY program that is used to enter the command terminal remotely. PuTTY is a free licensed SSH, Telnet, rlogin, and TCP raw client. The download is located in the following link `http://cplus.about.com/`, where different options are found, but due to its ease of work, the installer version is the best option. This can work in a way compatible with the Raspberry Pi board because it contains an even higher level of security for the proper management of programming.

Once downloaded and installed on Windows, you can access the Putty program to establish the connection with the Raspberry Pi board and then enter the Host Name option, where you specify the IP address to which the Raspberry Pi board is connected, Port 22 is specified, you assign a name to the connection in the Saved Sessions field and click on Saved, this way you will not have to specify the IP address and name when you start working with Raspberry Pi remotely. Thus, when the board is turned on, the previously established specifications are loaded by selecting the name of our IP and loading the option by clicking on load and clicking on open as shown in Fig. 5.

By means of the remote connection with VNC, it is possible to access the desktop or graphic mode of the board remotely. The procedure is as follows:

• Enter to the terminal window the following command, which consists of the installation of the VCN server on the Raspberry Pi board. This will load the remote desktop.

ISSN 1692-8407

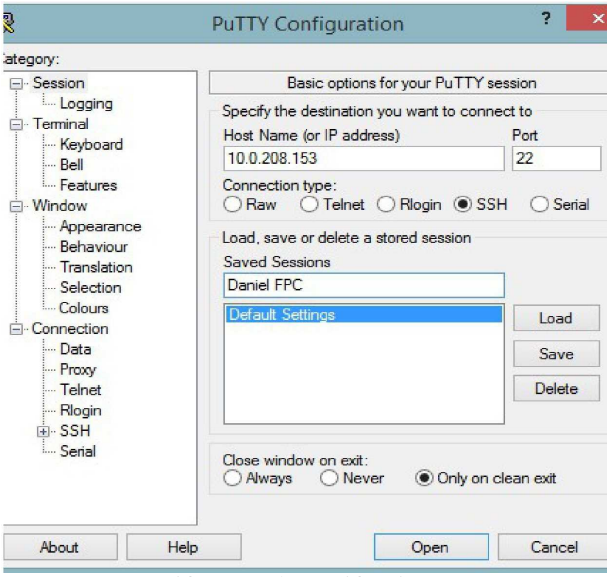

Figure 5. Putty Wifi network specifications.

This command can be executed using the PuTTY application (Fig. 6).

```
sudo apt-get install tightvncserver
```

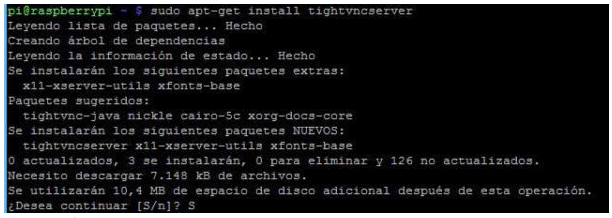

Figure 6. VNC server.

- When the installation of the VNC server is completed, the board is restarted, so that it works correctly, when it is restarted, the following command is executed:

```
vncserver :1
```

- The above command specifies different aspects of VNC, as the number 1 indicates the remote desktop we are going to use. The first time we make use of Raspberry Pi through VNC requests that a key is created (this is entered by the user, this is due to security protocols).
- The next step is the installation of VNC through Chrome, which was chosen because any programmer can make use of Google Chrome, which has the option VNC Viewer for Google Chrome, which allows access to the Raspberry Pi desktop, where it identifies that the images and colors used in the program are of very good quality adjusting to the request for the development of applications in robotics.
- In this window you enter the IP address used by the Raspberry Pi board and the desktop number used, to finally connect to the Internet as shown in Fig. 7.

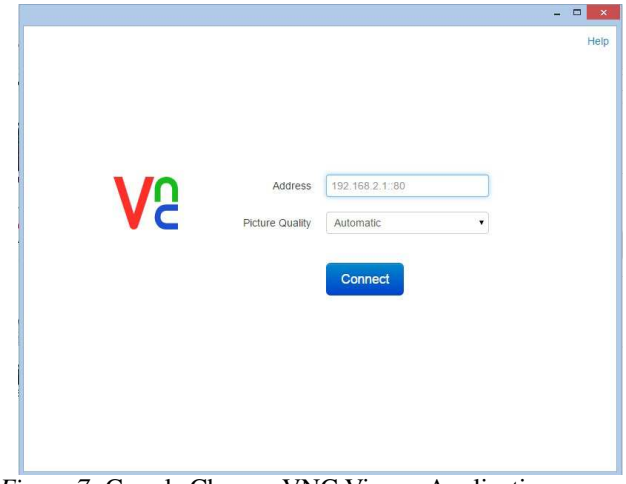

Figure 7. Google Chrome VNC Viewer Application.

- The following window can be omitted because it does not affect the configuration of the program, therefore the option not to display again is selected.
- The last step is to enter the previously created key. In this way, we enter the graphic or desktop mode of the Raspberry Pi board.

Before starting the installation of Arduino on the Raspberry Pi board, it is necessary to specify why the use of this development system. First of all is due to its microprocessor, which has the power and ease of programming for the execution of the project. Additionally, it allows the operation of servomotors (power) and sensors isolated from the control unit.

- To install Arduino correctly, update the database of Raspbian Linux packages by running the command:

```
sudo apt-get update
```

- When the upgrade is complete, proceed to install the Arduino package from the Linux server with the command shown below, where you will get the Electronic icon on the Raspbian start bar, which leads to the Arduino IDE shortcut. The Arduino installation is shown in Fig. 8.

```
sudo apt-get install arduino
```

- This step verifies that the Raspberry Pi normally detects the serial ports of Arduino, that the communication or interface between the microcontroller and the board is correct and executes the programmed application, this is done through the following command:

```
sudo usermod -a -G tty pi
```

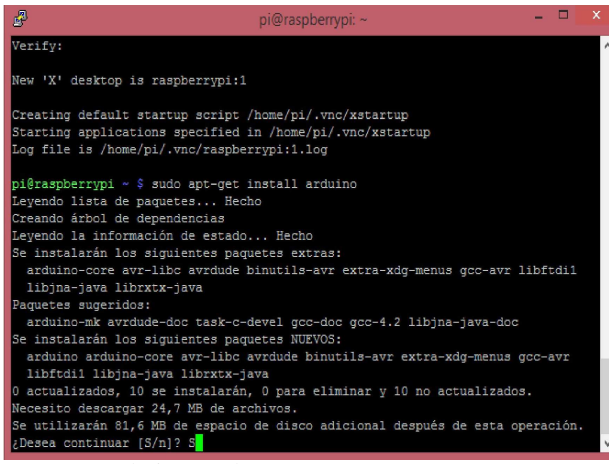

Figure 8. Updating packages.

```
sudo usermod –a –G dialout pi
```

```
ls /dev/tty*
```

Port verification is shown in Fig. 9. The image identifies that the board is working with Arduino via the `/dev/ttyACM0` port.

Figure 9. Arduino serial ports.

- It is now possible to verify the installed program by entering the Arduino IDE icon. This icon is located in the Start menu, Electronics - Arduino IDE as shown in Fig. 10.
- The next step is to configure the communication port between the Raspberry Pi board and the Arduino. This is done by accessing the tools icon, Serial Port and selecting the option `/dev/ttyACM0`. The configuration is shown in Fig. 11.
- The next configuration is to specify which type of Arduino board is available, in this case, the board is Arduino Uno, and the specification in the Raspbian system is shown in Fig. 12.

Installation of ROS OS Groovy Galapagos on Raspberry Pi

Installing ROS on the Raspbian operating system can be done in two ways (ROS.org, 2016), which are described below:

Figure 10. Location of the Arduino IDE Program.

Figure 11. Serial port configuration.

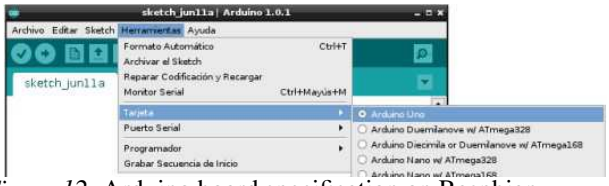

Figure 12. Arduino board specification on Raspbian.

Installation from source code. This process can take a long time to install due to the number of packages it contains. It is not recommended because the installation has compatibility problems.

Installation by binary packages. This installation method is much simpler and faster, although it lacks several packages for different applications, but has the right requirements for the development and execution of the robotic application in navigation. Its installation is done in the following way:

- ROS repositories are installed on the Raspbian system. It consists of adding the ROS OS repositories from command mode, entering a terminal and typing the following code:

ISSN 1692-8407

```
$ sudo sh -c 'echo\\ "deb http://packages.ros.org/ros/ubuntu wheezy main" >/ etc/apt/sources.list.d/ros-latest.list'
```

```
$ wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add -
```

• Update packages (again given the new repository entered into the system):

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

• Installation of ROS OS packages:

```
$ sudo apt-get install ros-groovy-ros-comm
```

• Installation of ROSDEP. These are dependencies of the ROS OS system that the system requires to run basic ROS OS components, not installed by default:

```
$ sudo rosdep init
```

```
$ rosdep update
```

• Completion of installation. The last step is to tell Raspbian the location of ROS, the next command should be executed whenever programming is done on ROS OS:

```
$ echo "source /opt/ros/groovy/setup.bash" >> ~/.bashrc
```

```
$ . ~/.bashrc
```

• Installation of rqt plot. This tool allows visualizing numerical values in a Cartesian plane using traces of the infrared sensor reading. This application is used to verify on screen the distance at which the IR sensor is located from the different objects that are close to its path. The command is shown below:

```
$ sudo apt-get install ros-groovy-rqt
```

```
$ sudo apt-get install ros-groovy-rqt-common-plugins
```

• The second way to install this application is through the command:

```
$ rosdep install rqt_plot
```

• To start or verify the plan with the rqt plot package, enter the command:

```
$ rqt_plot
```

• Creation of the workspace. It consists of creating an exclusive workspace for ROS and the programs to be developed. This workspace is basic and essential for the non-affectation of the Raspbian system.

```
$ mkdir -p ~/catkin_ws/src
```

```
$ cd ~/catkin_ws/src
```

```
$ catkin_init_workspace
```

• In the second command, the acronym src is empty inside the folder `catkin_ws` or workspace. The workspace is opened using the command:

```
$ cd ~/catkin_ws/
```

```
$ catkin_make
```

• When the workspace creation process is finished, the `catkin_ws` folder will get two subfolders: `build` and `devel` where the relevant devel folder is because the `.sh` files or packages are stored in this folder.

• To finish with the installation, go to Raspbian with the ROS system and the new packages or workspaces by means of the following command:

```
$ source devel/setup.bash
```

This completes the installation of ROS on Raspbian and the right conditions for the development of navigation applications.

Integration between Arduino IDE and ROS OS

The procedure for installing the Arduino IDE was mentioned earlier, which specifies that the program is compatible with the Raspbian system, but lacks compatibility with ROS OS. Thus, the integration of the two systems is done through the `Rosserial` protocol or package (A., 2014). The `Rosserial` installation takes place in the `scr` subfolder of the `catkin_ws` workspace folder by executing the following commands:

```
$ cd ~/catkin_ws/src
```

```
$ git clone https://github.com/ros-drivers/rosserial.git
```

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

```
$ catkin_make install
```

```
$ source ~/catkin_ws/install/setup.bash
```

At the end of the Rosserial installation, the ROS `lib` folder is created containing the libraries for communication between Arduino and ROS.

This package must be moved to be recognized in both ROS (workspace) and the Arduino IDE. It is taken to the `sketchbook/libraries` folder, which is located in the Arduino IDE folder in Raspbian. This step is done by executing the following commands:

```
$ cd ~/sketchbook/libraries
```

```
$ rm -rf ros_lib
```

```
$ rosrun rosserial_arduino make_libraries
.py
```

Robotic platform

For the system performance evaluation, the SERB robot was selected for the execution of navigation tasks (Oomlout, 2013). It is a differential robot of design *open source* moved by two servomotors. The original design was modified in order to mount the Raspberry Pi board, distance sensors and a rechargeable battery for power supply. Taking into account the characteristics of the movement of this robot, the tasks of navigation and avoidance of obstacles are designed. The final structure of the robotic platform is composed as follows:

- Two wheels of 13 cm of diameter connected to each side of the robot in order to offer stability and independence of mobility, in addition, they are connected each one to a servomotor that is responsible for realizing the rotation in the axes for the displacement, these are fed by means of a portable battery charger that delivers 5 Vdc
- A third wheel of three cm diameter totally independent. It will not be linked to any other element, serving only to support the fixed horizontal base.
- In the horizontal plate is the different peripherals such as the Arduino Uno board, the IR GP2Y0A41SK0F infrared sensor, two portable chargers, two servomotors, a protoboard, the Raspberry Pi board, and Wi-Fi antenna, as shown in Fig. 13 and Fig. 14.
- The navigation task was designed for the robot to move independently, so the power will be supplied by two portable chargers. The first charger is feeding the two servo motors and the IR infrared sensor. The second charger will exclusively feed the Raspberry Pi board. The Arduino Uno is connected to the Raspberry Pi, so its power supply is received via the board. Fig. 15 shows the electrical connection of the first portable charger. Fig. 16 shows the electrical connection of the second portable charger.

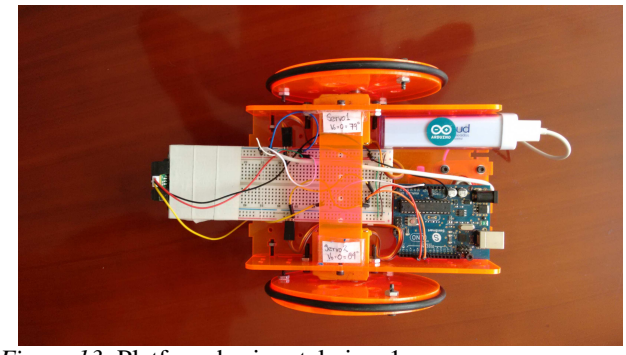

Figure 13. Platform horizontal view 1.

Robotic task

The execution of the robotic task is performed through the following procedure:

- When we turn on the Raspberry Pi board, which is connected by Wi-Fi and previously synchronized with the computer that performs the monitoring task, we access the command line to start ROS, through the code:

```
$ $ roscore
```

Fig. 17 shows the starting of ROS OS on the Raspberry Pi.

- The next step is to synchronize ROS OS with Arduino via command line in a new window or terminal. This is to locate the working space or folder. The next command is entered:

```
$ cd ~/catkin_ws/ && source devel/setup.
bash

$ cd
```

Fig. 18 shows the output that is displayed on the terminal. The communication between ROS OS and Arduino starts with the code:

```
rosrun rosserial_server serial_node _port
:=/dev/ttyACM0
```

- The control of the servomotors is carried out in a new terminal, where the task of speed and direction can be changed by means of the code:

```
$ rostopic pub servo1 std_msgs/UInt16 -
-once 180
```

The characteristics of the servomotors are specified in Fig. 19.

ISSN 1692-8407

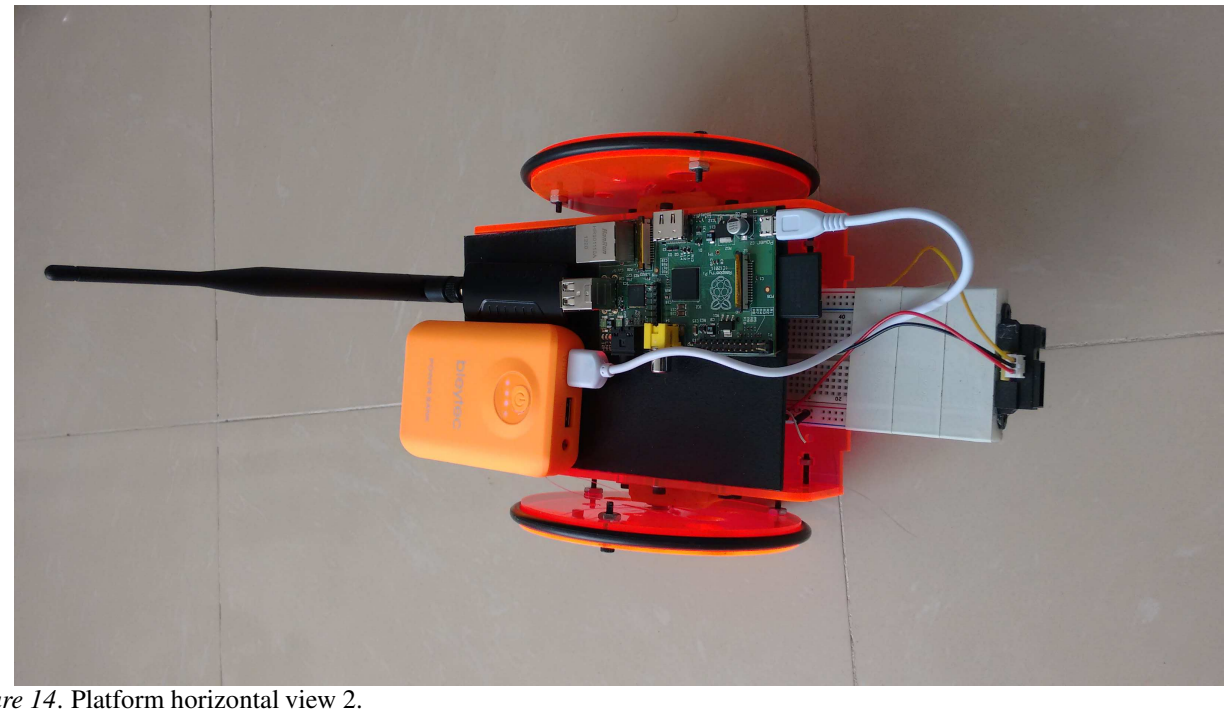

Figure 14. Platform horizontal view 2.

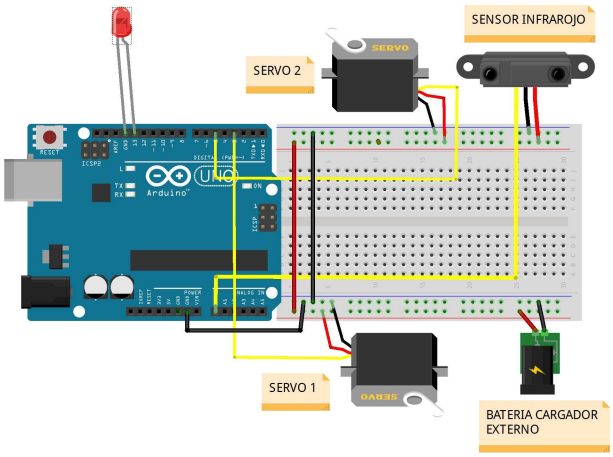

Figure 15. Electrical connection, portable charger 1.

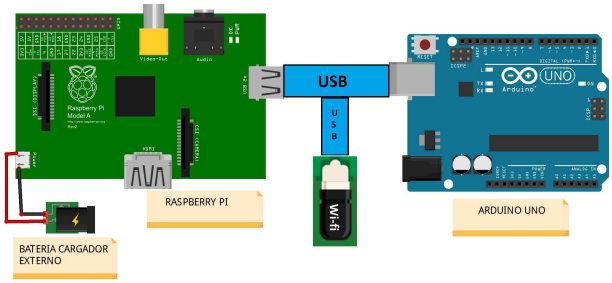

Figure 16. Electrical connection, portable charger 2.

• The servomotor to which speed and direction are assigned is indicated, this characteristic is modified through the angle. For this case, the maximum speed and direction of advance are assigned with an angle of 180 degrees. This speed can also be reached with an angle of 0 degrees but will be displaced in reverse. When the angle is equal to 90 degrees the speed of the servomotors is zero.

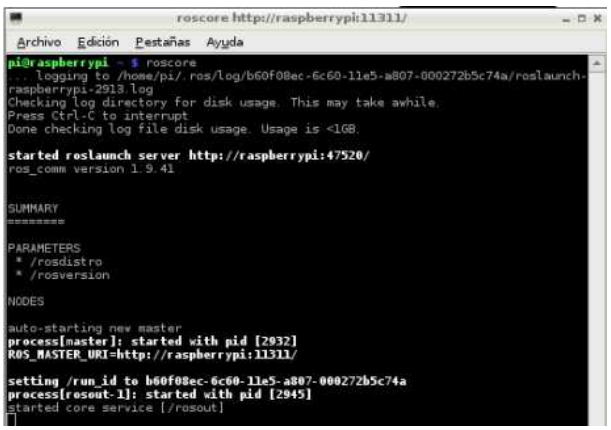

Figure 17. Beginning ROS OS on Raspberry Pi.

The above command shows control over the servo motor called servo1. The control of the two servomotors (servo1 and servo2) is done by means of a terminal, and the command to be entered is:

```
$ rostopic pub servo1 std_msgs/UInt16 --
once 180 && rostopic pub servo2 std_msgs/
UInt16 --once 0
```

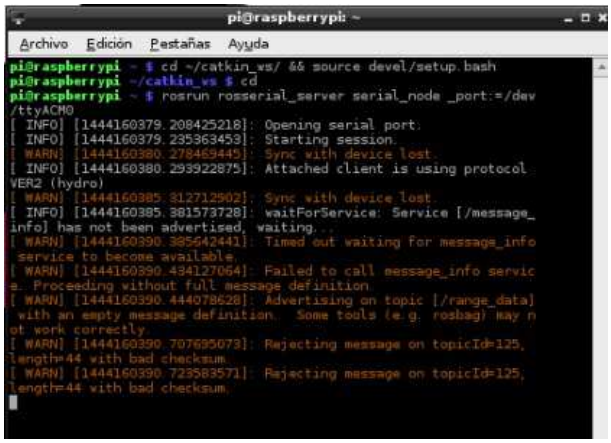

Figure 18. Opening the workspace.

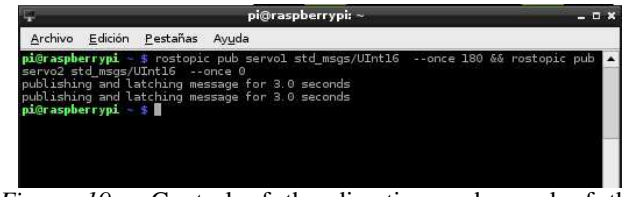

Figure 19. Control of the direction and speed of the servomotors.

The above command indicates that both servomotors, servo1 and servo2, move in the opposite direction at maximum speed.

Rqt_plot Application: The rqt_plot tool works with two-dimensional graphics, where a sample of the signal taken by the infrared sensor when it detects a near object within a radius of five cm is displayed. To enter this tool the following code is written in a separate terminal:

```
$ rqt_plot range_data/range
```

Fig. 20 shows the graph showing the infrared sensor by means of the Rqt_Plot function.

Navigation task

We implement a basic navigation task in unknown, static and observable environments. The SERB robot is programmed to perform displacement by means of two servomotors, which are monitored by means of a computer and the operating system ROS OS. When it is close to an obstacle, it will be detected by the robot by means of the infrared sensor, which sends the signal to ROS OS, which indicates to stop the robot, to turn one of its wheels in the opposite sense to the direction to which it moves. This causes a turn in the robot which after a second is put back into gear, advancing again with their wheels driven by servomotors 1 and 2.

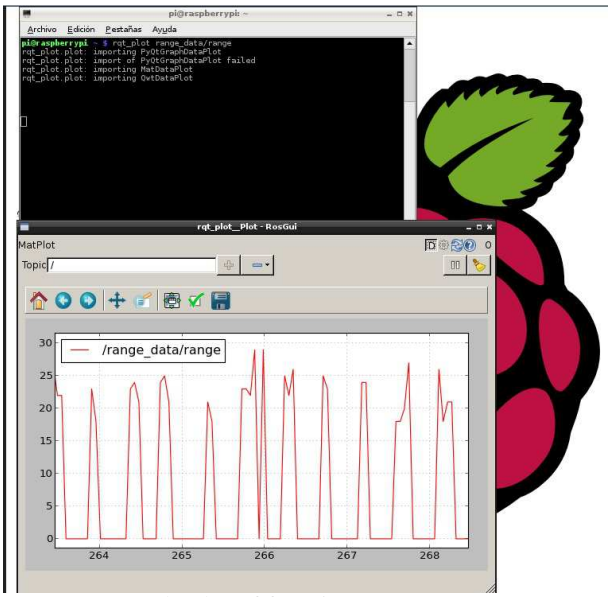

Figure 20. Sample plot of function `Rqt_Plot`.

Results and performance on SERB robot

The first result obtained is the interface achieved through Raspberry Pi with Raspbian and ROS OS with the version ROS Groovy Galapagos, for which to perform the steps exposed the synchronization can be given in a time of 8 s. However, the different times of execution of the programming of the robot are expressed below:

- Starting time of the Raspberry PI: 62 s (time needed to start communicating our pc and the Raspberry Pi through the Putty program).
- Starting Roscore 8 s (time required to open ROS in a Raspbian terminal).
- Communication ROS and Arduino 15 s (time required to link the communication of the Arduino board with ROS through a Raspbian terminal).
- Servo control 4.9 s (time required by a ROS command to control the direction or speed of a Servo).
- Control of a second servo 13 s (time taken by the second servomotor in the same order sent by ROS).
- Time opening Rqt-plot 58 s (time spent by the Rqt-Plot application to open and show the signal of an obstacle).

With this, it can be observed that the response times of the robot to sent orders are not immediate due to processing tasks of the Raspberry Pi that can become saturated with this type of applications.

The independence level of the SERB robot is reached thanks to the independence in the power supply and the control exercised remotely, for this reason, the power supply is through 2 chargers of 5 V and 0.5 to 0.7 A, which independently feed the Raspberry Pi board, the Arduino Uno board and the infrared sensor, as well as the second charger,

ISSN 1692-8407

is in charge of supplying the motion energy for servomotor 1 and servomotor 2, this means of feeding provides the degree of freedom for the robot to move autonomously over the entire navigation platform.

The level of control implemented in this robotic application exceeds the performance and specifications previously studied since the robot does not simply avoid obstacles but it can provide a desired or specific trajectory, this is achieved through remote control with the Putty program, so the observation or detection of objects is done in an average time of 58 s.

Extensions and issues

ROS OS is an operating system with little time on the market when compared to systems like Ubuntu, Windows and Linux nevertheless presents variety in its product so the versions to work are characterized by its tools and specific depending on the requirement of the programmer, for such reason the version used for the execution of the robotic application must comply with the interface between Raspberry Pi and ROS OS, reason for which tests were made of installation of ROS OS Indigo Igloo developed in 2014 and ROS OS Hydro Medusa designed in 2013 on the board, obtaining erroneous results at the time of verifying its compatibility, where the main reason is that these updates or versions of the program are developed by fans of the ROS community, so one of its main shortcomings are the outdated repositories which have the execution error on the Raspbian platform.

The navigation task is based on the detection of objects in order to avoid and take a new direction of movement, for this was taken into account the ultrasound sensor of Arduino HC-SR04, which is used in different robotic applications based on Arduino, but when this sensor is implemented with the card and the ROS OS system we identify that the operating system ROS OS Groovy Galapagos is not compatible with the sensor, we conclude that the sensor does not meet the interface characteristics necessary for the harmony of the system in general, for this reason we implement the infrared sensor IR GP2Y0A41SK0F which is recommended on the official website `www.ros.org`, where the adequate sampling or data collection is obtained for the performance of the robot.

The Raspberry Pi board model B, presents better capacity and resources than the model A, however in the design of the robotic application the idea considered the use of 2 sensors for better navigation and that the robot will perform in a more versatile way to more complex platforms, but when we observe the performance of the sensor and its tool rqt-plot on the plate we find that the resources needed for the robotic application are 100 percent. This optimizes resources by centrally locating the infrared sensor IR GP2Y0A41SK0F in the robot.

Conclusions

The robotic application designed is based on the interface developed between the Raspberry Pi board and the operating system ROS OS, this innovation is achieved through constant research in projects and robotic applications having as support any of these tools or linking them to verify their performance. The interface is based on an old version of ROS OS as it is ROS OS Groovy Galapagos designed in 2012, but as already specified the most current versions have repository problems or lines of code, additionally the working model of ROS is through spaces or folders generated from specific codes for its proper functioning, but it is necessary to add the communication between these two valuable programs which are: Raspbian and ROS OS, for this reason, to take the first steps with the different peripherals was complex because of the reduced ports provided by the board but at the same time a challenge to find and synchronize the best way of working of the board taking advantage of the benefits described above.

During the investigation the monitoring, control or command through a personal computer was one of the key points to find the connection between different equipment such as modem, cell phones and personal computers, reason why the VNC server, Putty and other features were basic and essential for progress and not dependent on the peripherals when executing the task in navigation. The next step consisted in obtaining compatibility of the Arduino board being one of the factors with greater facility to be able to carry out the programming of the robot through angles that in turn controls the servomotors that carry out the task of force, it is necessary to add that the Arduino works basically like actuator and was used by its easy or simple grammar of programming.

In the course of the project, there are many unknown factors such as the control of the SERB robot during the execution of the navigation application. Therefore, the independence achieved through 2 portable chargers, which have specific tasks or specific feeding elements, is achieved through the constant control of the infrared sensor, The idea was to work with an ultrasound sensor and not infrared, although the results were better than expected because with this new component is possible to obtain data in 2D which indicates the proximity of objects in front of the robot in a total range of 10 cm something that is suitable and sufficient for robotic displacement. The limits of the Raspberry Pi are found when we want to program the robot with two infrared sensors to obtain greater coverage at the time of moving on a given surface, which indicates that the resources are not being used in the right way or these are limitations of the board.

Finally, the aesthetics of the robot is not adequate but if functional so the robot has a degree of freedom wide enough to perform the task of navigation on a flat surface

with obstacles of specific characteristics as their measures, those if the sensor fails to recognize the obstacle the task of navigation will be quite difficult.

The development of this application is a small step for the functionality and take advantage of both the Raspberry Pi board and the operating system ROS OS, with the Arduino board one of the most popular boards in the area of robotics.

Acknowledgements

This work was supported by the Universidad Distrital Francisco José de Caldas, partly through the CIDC, and partly by the Facultad Tecnológica. The opinions expressed in this article are not necessarily shared by the Universidad Distrital. The authors thank the ARMOS research group for the discussion of ideas and strategies.

References

A., R. (2014). Integración de ros (robot operating system) con arduino y raspberry pi. *Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Sevilla , España*.

Calixto, G., Hira, C., Costa, L., & Deus, R. (2013). An open source and low cost solution for consumer electronics middleware validation. In *Ieee 17th international symposium on consumer electronics (isce 2013)* (p. 159-160).

Castiblanco, M., & Martínez, F. (2014). Exploración de un modelo comportamental basado en el quorum sensing bacterial para describir la interacción entre individuos. *Tekhnê*, *11*(1), 21-26.

Chien-Ming, H., & Mutlu, B. (2012). Robot behavior toolkit: Generating effective social behaviors for robots. In *7th acm/ieee international conference on human-robot interaction (hri 2012)* (p. 25-32).

Hold-Geoffroy, Y., Gardner, M., Gagne, C., Latulippe, M., & Giguere, P. (2013). ros4mat: A matlab programming interface for remote operations of ros-based robotic devices in an educational context. In *International conference on computer and robot vision (crv 2013)* (p. 242-248).

Holroyd, A., & Rich, C. (2012). Using the behavior markup language for human-robot interaction. In *7th acm/ieee international conference on human-robot interaction (hri 2012)* (p. 147-148).

Jacinto, E., Giral, M., & Martínez, F. (2016). Collective multi-agent navigation model based on bacterial quorum sensing. *Tecnura*, *20*(47), 29-38.

Jacinto, E., Montiel, H., & Martínez, F. (2016). Implementation of lightweight encryption algorithm based on 32-bit embedded systems. *International Journal of Applied Engineering Research*, *11*(23), 11409-11413.

Jizhong, X., Ramdath, K., Iosilevish, M., Sigh, D., & Tsakas, A. (2013). A low cost outdoor assistive navigation system for blind people. In *8th ieee conference on industrial electronics and applications (iciea 2013).* 828-833.

Kerr, J., & Nickels, K. (2012). Robot operating systems: Bridging the gap between human and robot. In *44th southeastern symposium on system theory (ssst 2012)* (p. 99-104).

Oomlout. (2013, March 1). *Arduino controlled servo robot.* On line. Vancouver, British Columbia. Retrieved from `http://oomlout.com/a/products/serb/`

Remmersmann, T., Tiderko, A., Langerwisch, M., Thamke, S., & Ax, M. (2012). Commanding multi-robot systems with robot operating system using battle management language. In *Military communications and information systems conference (mcc 2012).* 1-6.

ROS.org. (2016, Enero 11 de). *Ros tutorials.* http://wiki.ros.org/ROS/Tutorials. Retrieved from `http://wiki.ros.org/ROS/Tutorials`

Sundaram, G., Patibandala, B., Santhanam, H., Gaddam, S., Alla, V., Prakash, G., et al. (2013). Bluetooth communication using a touchscreen interface with the raspberry pi. In *Ieee southeastcon* (p. 1-4).