EINSTEIN TURTLEBOT

PROJECT #1

TYSON FOSDICK
TODD TOWNSEND
RUBEN MALDONADO
NICK LONG

Project Report Date: 10/31/2019

Version Number: 1.0

ECE 478

Department of Electrical and Computer Engineering

Table of Contents

INTRODUCTION

The Einstein TurtleBot is comprised of a Kobuki base and two arms and head that all have the ability to move. The arms utilize ten servos with four in each arm and two in the head. There is a range of limits that you must adhere to for the arms and head of Einstein so that the servos are not damaged. You must follow the Roboplus Dynamixel Wizard, to allow for the degrees of operation of the servos. In the arms, the servers are located at the shoulder, elbow, and wrist and in the head there is a servo for the pan of the head and a servo for the tilt of the head. There are raw values that must be accounted for they are the minimum and maximum value that can be sent to the servo. If these values are not accounted for and followed, the commands that are sent to Einstein can damage him. You also must account for the torque. Using precise calculations we also need to calculate the torque before administering commands to the servos.We have the individual components of Einstein working but we need to mount our Raspberry pi to the robot to have him successfully perform. We can run all of his acting capabilities but the cords are getting tangled as he spins and raises his arms.

BEHAVIOR

Our goal was to make our Einstein Turtlebot an experienced actor with funny behavior and profound arm and head gestures. After the setup and discovery of all the materials that we needed to learn to accomplish this took many hours we decided to have einstein navigate his surroundings and wave his arms when he discovered an obstacle. We plan on expounding on this behavior for the next project as we are finally becoming comfortable with Einstein's hardware and Software.

SETUP

We used our own host machine to set up our Einstein Turtlebot and there are several configuration steps, following each of the steps in the links below:

1. http://wiki.ros.org/kinetic/Installation/Ubuntu
2. http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem
3. http://wiki.ros.org/ROS/NetworkSetup

Now implement this Command as one command:

$ sudo apt install ros-kinetic-turtlebot* libudev-dev ros-kinetic-find-object-2d ros-kinetic-rtabmap-ros roskinetic-
moveit ros-kinetic-octomap-ros ros-kinetic-manipulation-msgs
ros-kinetic-controller-manager
Python-wxgtk3.0

Setup TurtleBot2i Source Code & Catkin Environment:

$ source /opt/ros/kinetic/setup.bash
$ cd ~

SETUP

$ mkdir -p ~/turtlebot2i/src
$ cd ~/turtlebot2i/src
$ git clone https://github.com/Interbotix/turtlebot2i.git .

Setup TurtleBot2i Source Code & Catkin Environment Cont:

$ git clone https://github.com/Interbotix/arbotix_ros.git -b turtlebot2i
$ git clone https://github.com/Interbotix/phantomx_pincher_arm.git
$ git clone https://github.com/Interbotix/ros_astra_camera -b filterlibrary
$ git clone https://github.com/Interbotix/ros_astra_launch
$ cd ~/turtlebot2i
$ catkin_make
$ bashrc setup

ROS:

1. Place the following at the bottom of your ~/.bashrc file
2. Replace example.hostname with your computer's hostname

$ source /opt/ros/kinetic/setup.bash
$ alias goros='source devel/setup.sh'
$ export ROS_HOSTNAME=example.hostname
$ export TURTLEBOT_3D_SENSOR=astra
$ export TURTLEBOT_3D_SENSOR2=sr300
$ export TURTLEBOT_BATTERY=None
$ export TURTLEBOT_STACKS=interbotix
$ export TURTLEBOT_ARM=pincher

Dialout permission

$ sudo usermod -a -G dialout <username_of_host>

CONNECT

Run Commands:

$ goros
$ roscore

1. Open another terminal/tab in terminal #2

$ goros

2. Now run the demos in the second terminal window terminal #2

Choose one of the following to establish Map:

Start robot and create a new map (erasing data from previous sessions):

Run: $ roslaunch turtlebot2i_bringup turtlebot2i_demo1.launch new_rtabmap:=true

CONNECT

To resume mapping:

Run: $ roslaunch turtlebot2i_bringup turtlebot2i_demo1.launch

To use existing map for localization purposes only:

Run: $ roslaunch turtlebot2i_bringup turtlebot2i_demo1.launch localization:=true

ARM DEMO:

1. Run: $ cd turtlebot2i
2. Run: $ goros
3. Run: $ roslaunch turtlebot2i_block_manipulation block_sorting_demo.launch

KEYBOARD NAVIGATE & CONNECT:

Run: $ cd turtlebot2i
Run: $ goros
Run: $ roslaunch turtlebot_teleop keyboard_teleop.launch

SERVO MOTORS SETUP:

1. Ubuntu 16.04 on the pi from https://downloads.ubiquityrobotics.com/pi.html. I selected the build 2019-06-19-ubiquity-xenial-lxde

2. Connected the Kokuki base to the pi.
3. Connected the motors to the adapter and then to the pi.
4. SSH into the pi.

5. Followed the steps in https://github.com/philiparola/ece478-turtlebot - starting at 'Building the project' section.
6. Built their project with the following commands

   $ source /opt/ros/kinetic/setup.bash
   $ cd <repository_dir>/catkin_ws
   $ source devel/setup.bash
   $ catkin_make

7. After building

   $ sudo chmod a+rw /dev/ttyUSB0

   $ roslaunch my_dynamixel_tutorial controller_manager.launch
   this will initialize all the motors

   $ roslaunch y_dynamixel_tutorial start_all_motor_controller.launch (this will start all the motors)

CONNECT

$ roslaunch face_detection face_detection.launch

$ roslaunch gesture_controls gesture_controls.launch

Note: Script have the port hard coded to have the servos connected to USB0. Make sure the Kokuki base is not enumerated to USB0. You can connect servo USB first and then the base USB.

TURTLEBOT BASE SETUP:

Run: $ goros

Run: $ roscore

Map Base, start with new map

Run: $ roslaunch turtlebot2i_bringup turtlebot2i_demo1.launch new_rtabmap:=true

Set Keyboard Controls

Run: $ roslaunch turtlebot_teleop keyboard_teleop.launch

Dependencies:

$ sudo apt-get install libgl1-mesa-dev-lts-trusty

Environment setup:

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

$ source ~/.bashrc

Ros Tools:

$ sudo apt-get install python-rosinstall

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

$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install ros-indigo-turtlebot ros-indigo-turtlebot-apps ros-indigo-turtlebot-interactions ros-indigo-turtlebot-simulator ros-indigo-kobuki-ftdi ros-indigo-rocon-remocon ros-indigo-rocon-qt-library ros-indigo-ar-track-alvar-msgse

$ sudo apt-get install python-catkin-tools

Workspace Setup:

$ mkdir ~/rocon

$ cd ~/rocon

$ wstool init -j5 src https://raw.github.com/robotics-in-concert/rocon/release/indigo/rocon.rosinstall

$ source /opt/ros/indigo/setup.bash

$ rosdep install --from-paths src -i -y

$ catkin_make

$ mkdir ~/kobuki

CONNECT

$ cd ~/kobuki

$ wstool init src -j5

https://raw.github.com/yujinrobot/yujin_tools/master/rosinstalls/indigo/kobuki.rosinstall

$ source ~/rocon/devel/setup.bash

$ rosdep install --from-paths src -i -y

Workspace Setup:

$ catkin_make

$ mkdir ~/turtlebot

$ cd ~/turtlebot

$ wstool init src -j5

https://raw.github.com/yujinrobot/yujin_tools/master/rosinstalls/indigo/turtlebot.rosinstall

$ source ~/kobuki/devel/setup.bash

$ rosdep install --from-paths src -i -y

$ Catkin_make

NTP setup:

$ sudo apt-get install chrony

Colcon install:

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

$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE

$ sudo apt update

$ sudo apt install python3-colcon-common-extensions

AWS polly Install:

$ sudo apt-get install python3-pip

$ pip3 install -U boto3

$ sudo apt-get install -y ros-$ROS_DISTRO-tts

AWS polly for newer builds:

$ cd ~/ros-workspace/src

$ git clone https://github.com/aws-robotics/tts-ros2.git

$ cd ~/ros-workspace && sudo apt-get update

$ rosdep install --from-paths src --ignore-src -r -y

$ cd ~/ros-workspace && colcon build

$ source ~/ros-workspace/install/local_setup.bash

$ colcon test --packages-select tts && colcon test-result --all

AWS robomaker install:

$ cd /opt/robomaker/cross-compilation-dockerfile/

$ sudo bin/build_image.bash

CONNECT

```
$ cd ~/environment/HelloWorld/robot_ws
$ sudo docker run -v $(pwd):/ws -it ros-cross-compile:armhf
$ cd ws
$ apt update
$ rosdep install --from-paths src --ignore-src -r -y
```

AWS robomaker install Cont:

```
$ colcon build --build-base armhf_build --install-base armhf_install
$ colcon bundle --build-base armhf_build --install-base armhf_install --bundle-base
```

VISION PROCESSING

We found that we needed to use a simple webcam to start the vision processing for the Einstein Turtlebot. At first we set up the kinetic camera but we all kinds of issues processing the images since the kinetic camera has a more advanced algorithm to process the images. We were able to have Einstein process the images and adjust his position based on obstacles he encountered.

WHAT WE LEARNED

We learned that the setup for the hardware and the software take a great deal of time even when you have the right tools. It seems to be a rule that for every hour of set up there are three hours of debug time. Once that we established communication with Einstein, there were hurdles to climb debugging the commands and their order. We learned that the reading and studying of what it takes to get our robot going needs to be done early and quickly, as you can fall behind very quickly when the project and the robot needs to be functional.

We would also change the orientation of Einstein's movements, we would base it on vision processing of humans instead of based on processing objects. We would also like to refine the movements of Einstein as they are not smoothly in order. Possibly making several movements with both arms and head at the same time. We would also like to refine our facial recognition abilities of Einstein and fine tune his vision processing.

CHALLENGES

It was difficult to follow the steps to setup our robot from the previous teams coursework. It was unclear what steps were vital for functional operation. We also had a difficult finding the right parts, batteries, and hardware to establish our robot. We spent a great deal of time researching the hardware and software needed as well as the connections between them. Once we were able to establish the proper hardware and software we still had substantial debugging. The app for the kinetic camera crashed because the kinetic camera needs extra configuration in OpenCV. We then had to bring a simple webcam in order for the vision application to successfully process. We were able to get the app for the vision processing running but we were unable to control any of Einstein's gestures. Another hurdle was the network setup between the master and secondary IP address that needed to be used between the robot, our host machine, and the raspberry pi. We also had a difficult time collaborating together using the same software. The catkin software behaved differently on each of our hosts and it took a little bit of a different twist for each host machine to get them all working. That was a definite challenge working individually but together on the same project.

ROLES

- Tyson: Speech Synthesis/Hardware
- Ruben: Servo motors/Software
- Todd: Software/scripts/artistic improvements
- Nick: Hardware repair/functionality

PROJECT 2 TO DO LIST

- Establish Einstein as the accomplished performance artist he really is
- Create a stable version of ROS that can move cross platforms
- Make arm and head movements based on facial recognition
- Establish Head, Arm, and Base movements work together with added artistic components
- Einstein raise and waive arms dancing based on voice commands
- Have Einstein follow scripts for the robot theater with added artistry
- Have custom gestures Einstein preforms based on spoken commands

APPENDIX

Resources:

http://wiki.ros.org/kinetic/Installation/Ubuntu
http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem
http://wiki.ros.org/ROS/NetworkSetup
https://github.com/philiparola/ece478-turtlebot

Project Link:

https://github.com/DrFoz/ECE478

Video Link:

Arm Movement:
https://drive.google.com/open?id=1Ir8Mg7wSTL5JatyIwD7_JFDIo8uyXj9Z
Base Movement:
https://drive.google.com/open?id=1gPx3vRITreGPHsqcNhCVngxRq7ifiBCf
Actors Studio: (We still need to mount Raspberry Pi for cord freedom)
https://drive.google.com/open?id=1m7QQjQ-6lbZp1LXzcxQstcScWiaj6BCF
Intent Request/Speech Synthesis:
https://drive.google.com/open?id=1mEtL2i5_8z4I8SRj9xwE46y-nDbxKwf9