***Introduction to Robotic Systems Course***

**LAB 8**

**Voice-Controlled Robot**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc31018363)

[1.1 Lab Overview 1](#_Toc31018364)

[2 Requirements 1](#_Toc31018365)

[3 Voice Control Specification 1](#_Toc31018366)

[4 Task: Create a New ROS Package 2](#_Toc31018367)

[4.1 Create setup.py file 3](#_Toc31018368)

[4.2 Task: Write the Program for the Robot 4](#_Toc31018369)

[4.2.1 Create audioTreatment.py and voiceControl.py files 4](#_Toc31018370)

[4.3 Build the Voice Recognition Package 4](#_Toc31018371)

[5 Task: Install Required Libraries 5](#_Toc31018372)

[5.1 Install Dependent Packages 5](#_Toc31018373)

[5.2 Download Speech Recognition Library 5](#_Toc31018374)

[5.2.1 CMUSphinx Library 6](#_Toc31018375)

[5.2.2 Deepspeech API 6](#_Toc31018376)

[5.2.3 Point to Downloaded Libraries in audioTreatment.py 6](#_Toc31018377)

[6 Task: Test Voice-controlled Robot 6](#_Toc31018378)

[7 Task: Visualize the Graph 7](#_Toc31018379)

[8 Task: Simulate Using Gazebo 7](#_Toc31018380)

[8.1 Install Gazebo 8](#_Toc31018381)

[8.2 Launch Gazebo 8](#_Toc31018382)

# Introduction

## Lab Overview

In this lab, you will be able to use a human voice to control of the robot to perform maneuvers like straight-line movement, left turn, right turn, stop, etc. ROS Graph concept will be applied in this lab. This voice control package will have two python programs. One of the programs uses a speech-to-text library to convert the input from the microphone to a string which is published to a topic. The second program, which subscribes to the topic, will from the received string determine the appropriate control for the robot.

# Requirements

The following hardware and software are required to complete this lab:

* **Hardware:**
	+ TurtleBot 3 Burger.
	+ Four TCRT5000 IR reflector sensors.
	+ Microphone.
		- Connect this to the Remote PC.

In this lab, speech recognition will be provided using any one of the three models: Google cloud speech-to-text, CMUsphinx, and Deepspeech.

Google cloud speech-to-text does not require any downloads.

# Voice Control Specification

The desired characteristics of the voice-controlled robot is defined thus:

1. The user talks into the microphone connected to the Remote PC, which would already have ROS running.
	1. Ensure the microphone is detected by the remote PC and selected as input
2. The word(s) spoken is translated to a string using any of the selected speech recognition library.
3. From the string produced by the library, the appropriate control command is sent to the Dynamixel Motors.
4. When the maneuver is complete, the robot will continue moving in a straight line.
5. The ROS computation Graph is as follows:
	1. Speech-to-text program node publishes the output string message to a topic.
	2. String-to-control node subscribes to the topic to get the published message.

# Task: Create a New ROS Package

* Open a new terminal and change directory to **~/catkin\_ws/src**
* Enter the following command to create a new package called **turtlebot\_speech\_recognition**. This package will depend on various other packages such as **std\_msgs** rospy and **geometry\_msgs**.

$ catkin\_create\_pkg turtlebot\_speech\_recognition rospy std\_msgs geometry\_msgs

* Open the created directory at **~/catkin\_ws/src/turtlebot\_speech\_recognition**. This directory should contain these two files: **package.xml** and **CMakeLists.txt**. Next, you will edit these files.
* Open and edit **package.xml** and edit as follows:

<?xml version="1.0"?>

<package format="2">

 <name> turtlebot\_speech\_recognition</name>

 <version>0.0.0</version>

 <description>Voice control for turtlebot3 from continuous microphone</description>

 <license>TODO</license>

 <maintainer email=”e-------a@a--m.com”>aem</maintainer>

 <buildtool\_depend>catkin</buildtool\_depend>

 <build\_depend>geometry\_msgs</build\_depend>

 <build\_depend>rospy</build\_depend>

 <build\_depend>std\_msgs</build\_depend>

 <build\_export\_depend>geometry\_msgs</build\_export\_depend>

 <build\_export\_depend>rospy</build\_export\_depend>

 <build\_export\_depend>std\_msgs</build\_export\_depend>

 <exec\_depend>geometry\_msgs</exec\_depend>

 <exec\_depend>rospy</exec\_depend>

 <exec\_depend>std\_msgs</exec\_depend>

</package>

* Open the **CMakeLists.txt** file. This file is the build configuration file required by the CMake build system. Edit it as the follows:

cmake\_minimum\_required (VERSION 2.8.3)

project (turtlebot\_speech\_recognition)

find\_package (catkin REQUIRED)

catkin\_package (DEPENDS)

catkin\_python\_setup ()

install (PROGRAMS src/audioTreatment.py src/voiceControl.py

 DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION})

* To simplify the commands, we will create a **.launch** file that allows us to launch a series of executables with options in one command.
	+ In the project folder, **~/catkin\_ws/src/turtlebot\_speech\_recognition**, create a “launch” folder.
	+ Inside the launch folder, create a file **recognitizer.launch** and copy the following lines of code into it.

<launch>

 <arg name="mode" default="deepspeech"/>

 <arg name="samplerate" default="16000"/>

 <arg name="duration" default="2"/>

 <node name="audioTreatment" pkg="turtlebot\_speech\_recognition" type="audioTreatment.py" output="screen" cwd="node">

 <param name="mode" value="$(arg mode)"/>

 <param name="samplerate" value="$(arg samplerate)"/>

 <param name="duration" value="$(arg duration)"/>

 </node>

 <node name="voiceControl" pkg="turtlebot\_speech\_recognition" type="voiceControl.py" output="screen" cwd="node" />

</launch>

* The launch code has three editable arguments: mode, samplerate and duration.
	+ - Mode sets the speech recognition provider. Some options are:
			* “"google”: neural networks and requires an Internet connection.
			* “deepspeech” (default): neural network, works without Internet, but you must first download an existing model.
			* “sphinx”: uses CMUsphinx, a fairly light speech recognition API.
		- Samplerate sets the sampling frequency of the voice signal from the microphone
		- Duration sets the length of the audio sample before processing starts.

## Create setup.py file

Open the project folder (location of the **CMakeList.txt** and **package.xml** files) and create a **setup.py** file.

* Copy the following into the file.

## ! DO NOT MANUALLY INVOKE THIS setup.py, USE CATKIN INSTEAD

from distutils.core import setup

from catkin\_pkg.python\_setup import generate\_distutils\_setup

# fetch values from package.xml

setup\_args = generate\_distutils\_setup(

 packages=['turtlebot\_speech\_recognition'],

 package\_dir={'': 'src'},

)

setup(\*\*setup\_args)

* Save and close the file.

## Task: Write the Program for the Robot

In this task, you will add the codes that process the sound from the microphone and provide the appropriate control commands to the robot.

Two python programs will be created in this task. We have provided these files with the complete code for you.

* The first file **audioTreatment.py** performs word/sentence recognition using the defined library passed as an argument when the package is launched in the command line.
* The second file **voiceControl.py** provides the appropriate control command for the robot using the strings from the voice to speech conversion.

### Create audioTreatment.py and voiceControl.py files

* Navigate to **catkin\_ws/src/turtlebot\_speech\_recognition/src/**. This is the src folder inside **turtlebot\_speech\_recognition** package directory.
* Copy the files **audioTreatment.py** and **voiceControl.py** to this directory.

The program **audioTreatment.py** uses the arguments entered by the user (command line) to initialize the mode, sampling frequency and sample period to be used for the voice to string operation. By default, the program uses the “deepspeech” API, with a frequency of 16000 Hz and audio clips of 2 seconds.

In the rest of the code, a ROS node is created as a publisher to publish the results from the speech recognition operation to the topic.

The program **voiceControl.py** subscribes to the topic where it retrieves the recognized word(s) as a string from which it determines the correct operation command to send the robot. The implemented operations are, movements of the robot in any direction (left, right, forward, reverse, etc.), acceleration, and deceleration.

## Build the Voice Recognition Package

* Open the terminal, enter the following command to build the package:

$ cd ~/catkin\_ws && catkin\_make

* Double-check this using **rospack list | grep turtlebot\_speech\_recognition** command. If the package’s path appears, there is no error.
* From the **catkin\_ws** directory, source the ROS workspace by using the following command:

$ source ./devel/setup.bash

* To make the file executable go to the package folder run the following command:

$ sudo chmod u+x \*

# Task: Install Required Libraries

In the remote PC, we need to install the dependent libraries need by the two python files we have created. These libraries are:

* pyaudio, soundfile, sounddevice for audio capture and save.
* speech\_recognition (library), the google speech recognition API, deepspeech, pocketsphinx for speech recognition.

## Install Dependent Packages

* First, run the following commands so that the dependent libraries can be installed without error.
1. $ sudo apt install python-pip
2. $ sudo pip install --upgrade pip
3. $ sudo pip install --upgrade cryptography
4. $ sudo rm -rf /usr/lib/python2.7/dist-packages/OpenSSL
5. $ sudo rm -rf /usr/lib/python2.7/dist-packages/pyOpenSSL-XXX.egg-info
6. $ sudo pip install pyopenssl
7. $ sudo pip install testresources
8. $ sudo pip install -U setuptools
9. $ sudo rm -rf /usr/lib/python2.7/dist-packages/httplib2
10. $ sudo pip install httplib2
11. $ sudo rm -rf /usr/lib/python2.7/dist-packages/pyasn1\_modules
12. $ sudo pip install pyasn1-modules
13. $ sudo pip install --upgrade pyasn1
* Now run the following commands to install the dependent packages.
1. $ sudo apt-get install portaudio19-dev python-all-dev python3-all-dev && sudo pip install pyaudio
2. $ sudo python -m pip install --upgrade pip setuptools wheel
3. $ sudo apt-get install gcc automake autoconf libtool bison swig libpulse-dev
4. $ sudo pip install SoundFile sounddevice SpeechRecognition pocketsphinx deepspeech==0.5.1 google-api-python-client numpy scipy

## Download Speech Recognition Library

In this task, we will download the speech recognition libraries that are enabled to be used in the audioTreatment.py file. We will also edit the audioTreatment.py file to point to the appropriate files in the downloaded libraries.

* Navigate to **catkin\_ws/src/ turtlebot\_speech\_recognition** and create a folder “audio\_data.”
* Navigate to **catkin\_ws/src/ turtlebot\_speech\_recognition/audio\_data** and create two folders “deepspeech” and “sphinx.”

### CMUSphinx Library

CMUSphinx library is about 40 MB. In this lab, we will use the pre-trained model (en-us), which was automatically downloaded when pocketsphinx was installed.

* Go to **/usr/local/lib/python2.7/dist-packages/pocketsphinx/model/**, copy all the content inside the folder. This folder contains the pre-trained American English model of CMUSphinx.
* Paste the copied content inside the “sphinx” folder.

### Deepspeech API

Deepspeech API is about 2 GB of data. To download it, open a terminal from the “deepspeech” directory and run the following command.

wget https://github.com/mozilla/DeepSpeech/releases/download/v0.5.1/deepspeech-0.5.1-models.tar.gz

When the download is complete, run the following command to extract the file

tar xvfz deepspeech-0.5.1-models.tar.gz

After extraction, delete the downloaded file.

### Point to Downloaded Libraries in audioTreatment.py

Open the **audioTreatment.py** file and find the function **initModes**. Now, we must edit the paths for Deepspeech and CMUsphinx libraries to point to the correct files.

#### Deepspeech library

1. Replace set path for **self.deepspeech\_alphabet** to the path of **alphabet.txt** file in the deepspeech extracted folder.
2. Set **self.deepspeech\_languageModel** to the path of **output\_graph.pbmm** file in the deepspeech extracted folder.
3. Set **self.deepspeech\_trie** to the path of **trie** file in the deepspeech extracted folder.

#### CMUSphinx:

1. Replace set path for **self.sphinx\_acousticModel** to the path of **en-us** folder in the sphinx directory.
2. Replace set path for **self.sphinx\_languageModel** to the path of **en-us.lm.bin** file in the sphinx directory.
3. Replace set path for **self.sphinx\_dictionary** to the path of **en-us.dict** file in the sphinx directory.

# Task: Test Voice-controlled Robot

* Ensure the OpenCR1.0 board is connected to the TurtleBot PC. Then, on remote PC, open a terminal and run

$ roscore

* Open another terminal, ssh into the Turtlebot PC and run Bringup.

$ roslaunch turtlebot3\_bringup turtlebot3\_robot.launch –-screen

Now the robot is ready to run the package and node we just created.

* On Remote PC, run the following command to launch the nodes from **recognizer.launch** with the default arguments.

$ roslaunch turtlebot\_speech\_recognition recognizer.launch

* + The program should run with the default configuration.
	+ The default configurations are:
		- Deepspeech as the speech recognition library
		- Sample rate of 16 000 Hz
		- Audio duration (before processing) of 2 seconds
* To launch the nodes with specific arguments, enter the launch command like this:

 $ roslaunch turtlebot\_speech\_recognition recognizer.launch mode:=google samplerate:=32000 duration:=5

* + audio samples of 5 seconds.
	+ 32000 samples per second.
	+ google speech recognition API.
* Test the robot
	+ Speak into the microphone, begin with simple commands like “left,” “right,” “front,” “back,” “stop.”

# Task: Visualize the Graph

Visualize the node graph information of the ROS system using “rqt.” Run the following command:

1. $ rqt

*Figure 1: Node diagram of the ROS Graph.*

# Task: Simulate Using Gazebo

In this task, we will use a virtual robot simulated in the Gazebo environment to test the speech recognition package.

## Install Gazebo

If you have not done so in the previous lab, download and install Gazebo and the necessary packages by running the following commands.

1. $ cd ~/catkin\_ws/src/
2. $ git clone https://github.com/ROBOTIS-GIT/turtlebot3\_simulations.git

## Launch Gazebo

* Ensure the microphone is plugged in and is detected by the Remote PC.
* Next, start roscore by running the command below.

$ rocore

* Open another terminal and run the following command:

$ roslaunch turtlebot\_speech\_recognition recognizer.launch mode:=google samplerate:=16000 duration:=2

* Open another terminal and run the following command so you can observe the “Twist” messages transmitted to the robot:

$ rostopic echo /cmd\_vel

* Open another terminal and launch Gazebo with the following command:

$ roslaunch turtlebot3\_gazebo turtlebot3\_empty\_world.launch

You can replace **turtlebot3\_empty\_world.launch** with other **.launch** options. Press the tab key when you write “turtlebot3” to see all the available options.

Now speak into the microphone and watch the virtual robot respond to your command.