Portland State UNIVERSITY

ECE 478/578 Intelligent Robotics 1

Project 2 - NewtonBot – Report

Fall 2018

Instructor - Dr. Marek Perkowski

Team Members

Kanna Lakshmanan

Lauren Voepel

Jennifer Lara

Rahul Marathe

Table of Contents

Project Description

This project can be broken up into two sections, Human Interaction, and Robot Theatre. Human interaction involved the robot responding to human speech. Speech was interpreted using Google Dialogflow and was synthesized using Amazon Polly. Robot theatre involved cooperating with two other robot teams, Einstein and Dim, to act out a play written by Dr. Perkowski. The script for the play has been included in the “extra files” folder on the GitHub for this project.

Overview of Hardware

Raspberry Pi 3

Figure 1: Raspberry Pi 3 model B courtesy of Raspberypi.org

For project 1, our robot was controlled using a developers version of the Intel Joule which had been donated to the robotics class. Unfortunately, it was found that our version of the board was incompatible with the turtlebot packages installed with ROS kinetic. Upon launching the turtlebot_bringup files which controlled the base, the computer would freeze up and eventually crash. This was tested using a version of the bringup code which was identical to the code which another turtlebot team had been using, so it was determined that the joule was the issue. From there, we had to migrate our code over to a raspberry pi 3.

Initially, we used an install of raspbian stretch on our pi, but we later moved to Ubuntu Mate for raspberry pi. This was because raspbian did not support installing dependencies for our robot using the apt-get install command, which made the install process more complicated.

With Mate, we could install all dependencies using the same process that we had used in project 1.

The raspberry pi 3 has an audio jack, an HDMI plug, 4 USB plugs, and an ethernet jack. The board is also wifi capable, which is important for connecting to Google's Dialogflow and Amazon's Polly servers. The only issue with using the raspberry pi is that it is significantly less powerful than the Intel Joule. As a result, the image processing portion of project 1 had to be cut out for this project.

Overview of Software Tools

ROS

Robot OS or ROS is a modular programming environment designed for working with robots. The version of ROS we used on this project was ROS Kinetic, which only works on Ubuntu 16.

ROStopics

Rostopics are subjects which are sent and received over the ROS system. These subjects can be either published by a program or subscribed to by the program. Publishers send data out for subscribers to receive. A single ros node can be both a subscriber and a publisher, and the publisher/subscriber system is what makes up the backbone of ROS programs.

For robot theatre, the rostopics were /lines and /increment. /lines correspond to which line of the play was currently being read, and /increment was published to after a robot finished speaking to indicate that the line should be incremented by counter.py

DialogFlow

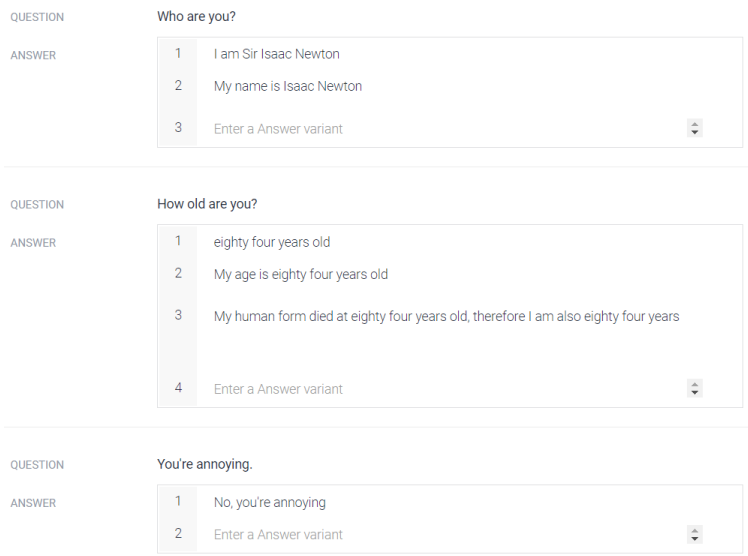
Dialogflow is a chatbot API created by Google. Users can either input an audio or text file which will then be parsed by Google's servers. Depending upon the input contents, a variety of responses can be sent back. These responses must be pre programmed using DialogFlow's "intents and agents" system.

Agents

The system in Dialogflow allows the use of natural language understanding modules. This system translates spoken text and matches it with the intents under the agent. For this project we had multiple intents, these intents include *aboutself*, which correlated to information about Newton. *Gravity*, which had Newton talking about the great things about gravity, and *robotics*, which would would tell you how awesome robotics class was and who the professor is.

Intents

This project had several intents integrated, as mentioned before. The creation of these intents require the implementation of training phrases, actions & parameters, and correlated responses. For example in the creation of training phrases, for the intent *aboutself*, there were several training phrases given because this increases the likelihood DialogFlow would recognize the question was asking a question about Newtonbot itself. Then for the actions & parameters, this intent used entities to learn how to correlate one words to multiple words. This is similar to how humans use synonyms. The responses given by DialogFlow were essentially saying the same thing but in different ways, like a human would. For example, asking Newton “Tell me about yourself” and they would respond with anything from “I published three editions of Mathematical Principles of Natural Philosophy” to “I studied physics, natural philosophy, mathematics, astronomy, and economics.”



Integrating Small Talk

This feature, Small Talk, is a feature that can be enabled with any agent. The feature allows one to train the robot to correlate certain phrases to certain types of questions or statements. The correlation of these two is implemented by DialogFlow, however, someone still has to enable the feature and add the training phrases for the agent. This integration enabled by DialogFlow allows for small pleasantries like, “Hello” ,”Good Morning”, and “How are you” , to have at least and answer. At most some questions have multiple answers to imitate how humans can say the same thing with different sentences. Small Talk also allows for confrontation interaction between a human and robot with statements like, “You’re boring” , “You’re crazy”, or even “You’re beautiful”. The responses can get quite funny and adding a few different responses allows the human to be surprised when the bot replies, “No, you’re boring.”

Amazon Polly

Polly is a text to speech synthesizer created by Amazon. Users input the text that they want to be read in an ssml format and Amazon returns an audio file of what they requested. Polly has a variety of voices, but for this project we used “Brian” which is a british english male voice.

Amazon Polly converts input text into life-like speech. You call one of the speech

synthesis methods, provide the text you wish to synthesize, select one of the available Text-to-Speech (TTS) voices, and specify an audio output format. Amazon Polly then synthesizes the provided text into a high-quality speech audio stream.

- **Input text** – Provide the text you want to synthesize, and Amazon Polly returns an audio stream. You can provide the input as plain text or in Speech Synthesis Markup Language (SSML) format. With SSML you can control various aspects of speech such as pronunciation, volume, pitch, and speech rate.

- **Available voices** – Amazon Polly provides a portfolio of multiple languages and a variety of voices, including a bilingual voice (for both English and Hindi). For most languages you can select from several different voices, both male and female. You specify the voice ID name when launching the speech synthesis task, and then the service uses this voice to convert the text to speech. Amazon Polly is not a translation service—the synthesized speech is in the language of the text.

- **Output format** – Amazon Polly can deliver the synthesized speech in multiple formats. You can select the audio format that suits your needs. For example, you might request the speech in the MP3 or Ogg Vorbis format to consume in web and mobile applications. Or, you might request the PCM output format for AWS IoT devices and telephony solutions.

Boto 3 Library

Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python, which allows Python developers to write software that makes use of services like Amazon S3 and Amazon EC2.

It uses a data-driven approach to generate classes at runtime from JSON description files that are shared between SDKs in various languages. This includes descriptions for a high level, object oriented interface similar to those available in previous versions of Boto.

Because Boto 3 is generated from these shared JSON files, we get fast updates to the latest services and features and a consistent API across services. Community contributions to JSON description files in other SDKs also benefit Boto 3, just as contributions to Boto 3 benefit the other SDKs

ROS Modules

Gestural Movement

Gestural movement module was adapted from code used in project 1. In part 1, a separate script was used for calling each gesture for the robot. For this part, we've included all gestures in a single script, and the gesture played depends upon rostopics. For the play portion of project 2 this topic is "/lines" which is a 32 bit integer which corresponds to which line is being spoken in the play. For the interactive portion, the gesture code is subscribed to.

Theatre

Because robot theatre is entirely scripted it was acceptable to use pre-synthesised sound-bytes for each line. To know which line to say and when, a rostopic called /lines was created. When data is published to /lines, a callback function is entered. From here, the robot checks to see if it has that number line as an audio file.

If the file exists , the robot plays the audio file using the pygame library for python, then publishes the current line to /increment when it's finished speaking. /increment is subscribed to by a separate controller script called "counter.py" which takes this value, increments it, then publishes it to /lines. This separate counter is used to prevent any robot from directly modifying the line value, but it could be removed in future iterations of this design.

If the file does not exist, the robot does nothing and it is assumed that the line will be handled by one of the other two robots. We collaborated with the other two teams to name the lines on our robots based upon what lines of the overall play they would be speaking. For example, for act 1 our robot was in charge of speaking lines 6, 9, 11, 13, and 15.

Human Robot Interaction:

The second part of speech synth sis was human -robot interaction. The task was to get the bot to have a natural conversation with a human wherein the bot would listen to the questions asked by the human and reply appropriately.

The workflow for this task was tri faceted. There were three distinct module in play during the speech synthesis. To simulate a real conversation, emphasis had to be given on proper handshaking so that the timing period between the bot listening and speaking would not overlap so have a smooth and natural conversation.

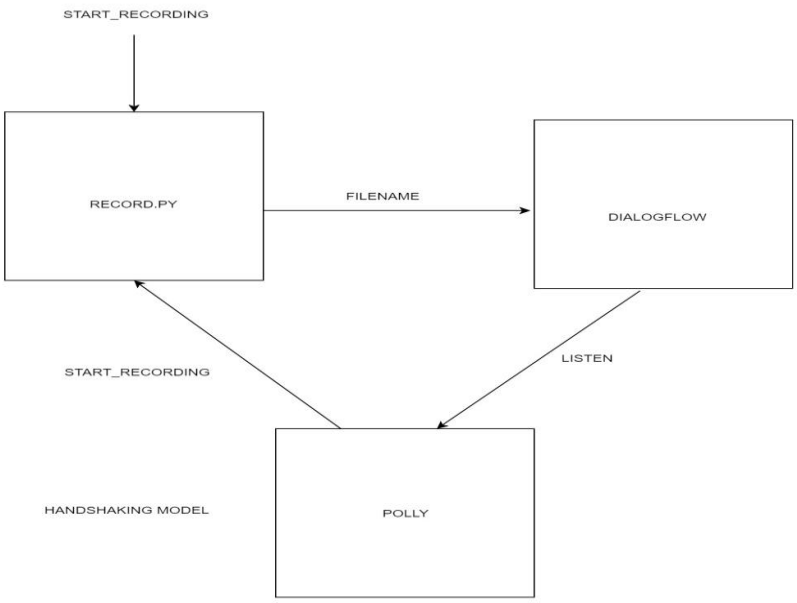


Fig: Handshaking Logic for the three rosnodes

Three Modules Record.py, Dialogflow.py and Polly.py were instantiated as rosnodes.
The Arrows between the Rosnodes are the rostopics that are passed between the nodes.

1.) Record.py:

This is the python script that records the input audio from the user. This audio file can be saved in multiple formats,bit rates, chunks but for our use we used .wav format for the input file. A .wav file was easily picked up by the dialogflow module for processing. The trigger given to the Record.py node is Start_Recording which is a Int32 value representing the number of seconds that the input recording should take place. Pyaudio library was a dependency for this recording to work.

2.) Dialogflow.py:

```
def get_file_name(data):
        filename=data.data
        homedir = os.environ['HOME']
        filepath = homedir + "/catkin_ws/"
        #user_input = os.path.join(filepath, 'user_input3.wav')
        user_input = os.path.join(filepath, filename)
        print(user_input)
        time.sleep(2)
        result = detect_intent_audio("hello-19424", "1-1-1-1-1", user_input, 'en-US')
        print("done")
        print (result)
        if (result.find("Greetings")>=0)or(result.find("Hello")>=0)or(result.find("Hi")>=0):
                pubgesture.publish(6)
        elif result.find("Bye")>0:
                pubgesture.publish(6)
        elif result.find("No")>=0:
                pubgesture.publish(9)
        elif result.find("Gravity")>=0:
                pubgesture.publish(15)
        pubtext.publish(result)
        time.sleep(3)
```

Fig: Code Snippet for appropriate Gestures and responses

Dialog flow was set up with three intents and small talk feature. The intention was to get the bot to respond appropriately to the questions asked by the user and have a real conversation.

The snippet above shows that we mapped some of the answers to the gestures like Hi, Bye and arm waving to the answers from the DialogFow api call. The bot looked animated and a proper response with a motion if what we had aimed for and where able to achieve.

3.) Polly.py:

Polly is Amazon's Text to Speech Web Service. A session with the Amazon Polly Web Service was created using a python SDK called Boto3. Boto3 is the Amazon Web Services (AWS) Software Development Kit (SDK) for Python, which allows Python developers to write software Amazon Polly converts input text into life-like speech.

```
def Speak_callback(data):
        global Text

        global polly_client
        Text_Input=data.data
        response = polly_client.synthesize_speech(VoiceId='Brian',
                        OutputFormat='mp3',
                        #Text = 'Robotics Sample Text.')
                        Text = Text_Input)

        file = open('speech.mp3', 'w')
        file.write(response['AudioStream'].read())
        file.close()

        #time.sleep(2)
        mixer.init()
        mixer.music.load('/home/turtle1/catkin_ws/speech.mp3')
        mixer.music.play()
        print("File Played")
```

Fig: Snippet for Polly Text to Speech Processing

On completion of the Text to Speech Conversion, a Python library called Pygame was invoked to play the generated mp3 file as an audio output. Then the Polly node would publish to the Start_Recording rostopic for the Record.py to start recording the next audio input from the user.

Issues We Ran Into

We experienced many problems with playing audio during this project. Initially, we used a preinstalled audio player called aplay to play audio via the terminal, but it was found that the audio would cut out if two instances of aplay were called too soon after one another. We tried to fix this by using ffmpeg's ffplay function which is also preinstalled on the Pi. This caused the same problem as with aplay. We suspect that this is due to core limitations on the Pi causing problems with multi-threading audio. The problem was solved using the pygame library, but this

introduced an issue where audio would sometimes cut out too early or lock up while playing. We edited the Pi's .config files to set audio dithering to false, and this seemed to remedy the issue.

During the play, we ran into a rostopic conflict between our robot and another team's. Specifically, both of our robots were subscribed and publishing to the same topics for moving the HR-OS1 servos on the upper portions of our robots. Because of this, we could not move our servos while connected to the same ROS network as their team's robot. A way to solve this would be to edit the names of the rostopics so that they would be distinct between our two robots. We did not have time to resolve this issue before presenting however, so movement did not work during Robot Theatre.

Another issue we ran into while debugging DialogFlow to work with ROS was the interactions with the sessions client and trying to run the service on our own machines using Virtual Box. For example, audio input issues came up while trying to use VirtualBox to take in a microphone sound. These issues were solved by running DialogFlow from one account only on the Raspberry Pi instead.

Plans for Improvement

Full integration of each section, vision, movement, and servos, speech would be ideal. An example would be having the base move along with a dancing top part of the robot. The trigger for that could be a symbol detected from the camera. Additionally, more use of the sensors available would help improve the robot's capabilities. For example, the depth function of the 3D camera was unused for this project.

To enable better interaction between a human and robot, we are also planning to try and implement use of a playstation controller to move the robot in different ways. For example, the use of the joysticks on a playstation controller to interact with the directional movement of the Turtlebot base. Additionally, the buttons could be used for different predetermined gestures for the servos.

Another improvement would be to add more training phrases and possible responses for the robot. This would improve human and robot interaction and make their interactions more natural. This means a simple or predictable question has a prepared answer to lookup in DialogFlow.

GitHub Page & Links

GitHub : https://github.com/RahulMarathe94/Fall-2018-ECE-478-578-Robotics1-TurtleBot_Project2
Drive with videos:
https://drive.google.com/open?id=1XmTyqDrMqoHCpShqxWBQtJQpcjDelNNQ

The drive must be accessed with a pdx.edu account in order to be viewable