Einstein TurtleBot

Scott Matsuo and Philip Arola

https://github.com/philiparola/ece478-turtlebot

# Overall Behavior

Our robot is designed to speak interactively, and perform in theater plays.

# Subsystems

## Speech Synthesis

The speech synthesis script is a service node which takes in a text phrase as a request, plays an mp3 of the phrase, and returns a success or failure status in regards to executing the mp3. First this script opens up a Amazon Polly session to access the speech synthesis tools provided by Polly. To do this, a AWS access key id and AWS secret access key are required, which you can generate from the AWS website. After this, the node is initialized and waits to be called. When called, the node takes the request, which in this case is a text phrase, and with Polly, turns it into an mp3. The mp3 is then played and the status returns 1 for success.

## Voice Recognition

The voice recognition node consists of three stage: recording, processing, and response. In the voice recording stage, the robot records audio and saves the audio file. This code was mostly provided by Melih, but it has been modified

The voice processing stage is likewise mostly derived from Melih’s code, just modified to better handle my control loop. It takes in the captured audio file from the prior stage, and sends it to a Google server, which processes it and returns a string. It does this by determining what is said (speech to text), and processing a response based off of trained conversation models. The response is a string of like-natural speech.

The response stage takes the response from the processing stage, and vocalizes the response over speakers and will gesture. The gesture is determined by checking a switch statement of pre-selected responses, and choose the matching gesture. The voice synthesis code is explained in the previous node.

The specific modifications made to Melih’s code facilitate the use of the record.py and dialogflow.py as Python modules instead of standalone scripts. I then made my own script (recognition.py) which used these modules in a control loop. This allows us to analyze the output of the dialogflow request so we can perform gestures or reject bad results. Given more time, this would also allow us to set volume detection on the microphone so we don’t have to manually trigger the recording.

## Robot Theater Actor

The robot theater actor node subscribes to the director node written by the Schrodinger Cat team, and publishes whether or not it is currently active.

The actor gestures according to the line entered. Once the gesture is determined, it sends a gesture command. Voice synthesis and playback is handled on the director node for physical reasons (speaker placement, power).

# Assembling the robot, setup the development environment

There are three data connections to the controller; the TurtleBot base (the wheels), the servo controller (Dynamixel), and the audio card (speakers and microphone). The first two devices are hooked into a USB hub, which is then plugged into the main controller.

The controller utilizes Ubuntu 16.04, and uses ROS Kinetic.

# Lessons: what we learned/improvements

We improved collaborating with version control by figuring out how to properly track files. Of particularly difficulty was using .gitignore, which is used to block automatically generated files that contain path/environment information that breaks compilation and execution on other controllers.

We also refined our skills with ROS node, publishing/subscribing, and services. In particular, we learned how to utilize ROS networking to effectively subdivide work.

# Challenges

The primary challenge we faced with this project was library management. Python was particularly problematic with DialogFlow and Amazon Polly. We also had trouble when trying to setup the Raspberry Pi as the controller for the robot, because many essential ROS libraries were not available for ARM.

We encountered difficulties when we shared hardware with the other teams. Specifically, hardware configurations with the servos would often be changed without the others be told (non-maliciously)

Record.py is not run as a module currently: we’ve encountered a bug where initializing and running the main code in the loop breaks recording because the audio device is kept open by the since finished call to main. This warrants further investigation.

# Video Link

<https://drive.google.com/file/d/1NESze2Fq4g-mprWR1u-G-BGPxTxIJqEQ/view>