Rotman Daniel Leiva Henriquez
Email: rdleiva@my.uri.edu
Tel: 401.359.4590

University of Rhode Island
Artificial Intelligence Lab

Supervisor: Harrison Dekker, MLIS
Tel: 401.874.5935
Email: hdekker@uri.edu

Supervisor: Indrani Mandal
Tel: 401.874.2701
Email: indrani_mandal@uri.edu

April 28, 2020

# CSC499 - Final Report

As we have come to the end of the semester, I cannot be anymore grateful to have been part of the Artificial Intelligence Lab team at URI. It has been a challenging but fulfilling experience that has helped me prepare for my future professional endeavors. I am grateful to have been under the supervision and guidance of Dr. Indrani Mandal and Harrison Dekker. I am also grateful for the opportunity Dr. Stegagno gave me of providing me a functional robot to experiment with and test a new voice recognition feature on. If I had the opportunity to do it all over again I would enjoy this journey once again.

Working at the AI Lab provided me with valuable experience in an office work environment as well as remote work environment. Our biweekly meetings to report of project progress was a great opportunity to enhance my interpersonal and communication skills with colleagues and superiors. Being part of the AI Lab gave me the opportunity to be part of a diverse group of people where we all respected and acknowledged each other as equals.

Besides the interpersonal skills and awesome experience at the AI Lab, I was fortunate enough to work on a project that challenged me and helped me enhance my technical skills such as problem solving and critical thinking. I also gained knowledge and experience of ROS and some of its features. Additionally, I gained more experience programming in C++ and Python which gives me more appreciation for the course work I have taken at URI. Using all of this knowledge and skills I was able to implement a voice recognition feature on the TANKD01 robot which Dr. Stegagno provided for me.

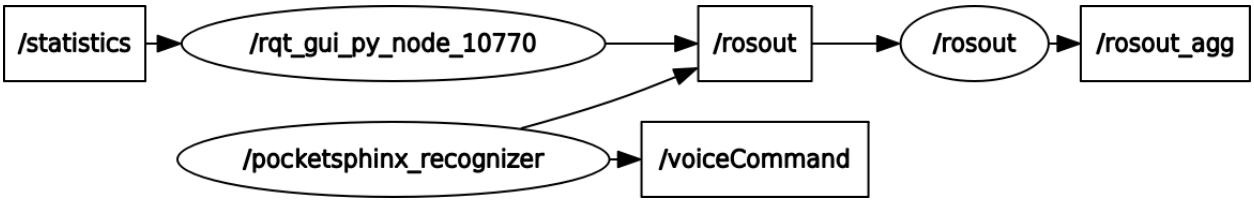
The first month and a half of my internship was probably the most confusing time for me. I did not know which way to go or what to do with so much information about ROS coming my way. I read most of the ros.org website and did all the tutorials that Mr. Dekker had suggested. But even then I did not yet understand how or what I needed to do for my project. I felt a little discouraged at times because it felt like I was not getting anything done at the time. Everything that I was doing at the time was trying to grasp my understanding for ROS and how it worked, so

I could then do something cool with the robot. During my "tutorial" stage I went through all of the beginner tutorials on the the ROS wiki page. I learned about how to install and set up the ROS environment on a Linux machine, even though that had already been done by Mr. Dekker. I learned there are different versions of ROS that have been released over the years. The most up to date one being ROS Melodic and the previous one being ROS Kinetic. They follow an alphabetized trend so the next version to be released should start with an N. The tutorials walked me through the navigation of a ROS filesystem as well as the creation of packages, and the main components of ROS.

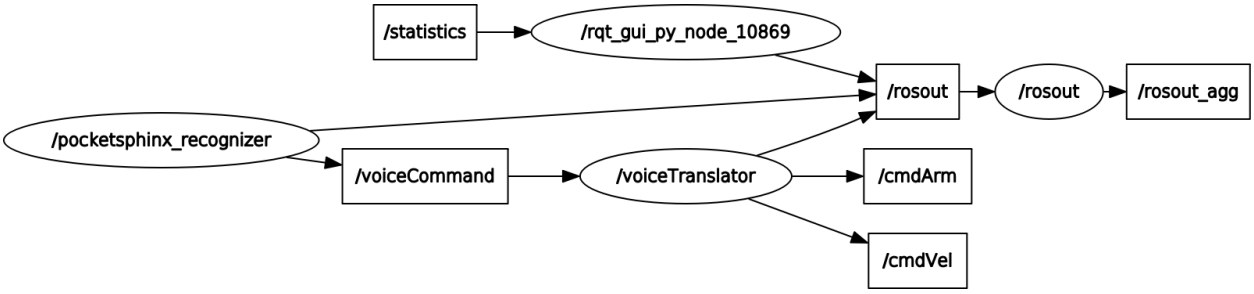
There are three main components to ROS that are very important to understand the structure of this system. First, packages are composed of nodes which are the programs that will give functionality to your robot. Nodes can be specific functions of the robot such as motor controls, voice recognition, voice translation and anything you can imagine that a robot could do. These nodes can be written in any language of your choosing but the two main languages used by most developers are C++ and Python. This part was convenient since I have experience with both. The second component are the ROS communication tools which include topics, services, and actions. With the tutorials I learned mostly about topics and a little bit about services. I learned about actions through the roboticsbackend.com website. Third, the data that you send between nodes is called messages which are data of different types such as ints, floats, strings, etc. From the communication tools the one that I became most comfortable with was the topics, and this is what I ended up using for the implementation of the voice recognition feature. The way topics work is that they can be published and subscribed to. That's a little difficult to understand at first and a diagram definitely helps with this notion. Topics receive messages through their publisher node and these messages can be *published* through the topic to another node that is *subscribed* to the topic. This is essentially the was most things work on ROS and this is what makes it so powerful to use because one does not have to create this whole architecture from scratch. It is readily available to use by any developer since ROS possesses open source licensing.

After learning the basics of ROS I was able to focus on my new gained knowledge into my project with the robots. Dr. Indrani suggested the idea of adding voice recognition capabilities to the robot. At the time it seemed like a challenge that was doable given the time restrictions of the spring semester but I was able to accomplish it in the end. I used the pocketsphinx wrapper package from the University of Texas Nuclear Robotics Group to allow the robot to gain input from a microphone and understand the voice commands. I then created another package which took the voice commands and turned them into keyboard commands for the keyboard command interface that already existed. I chose to implement it this way because I did not want to "hard-wire" anything to the current architecture of the robot. And having the commands go through the keyboard instead of directly to the robot would allow me to undo the changes easily since this was just an experimental feature.

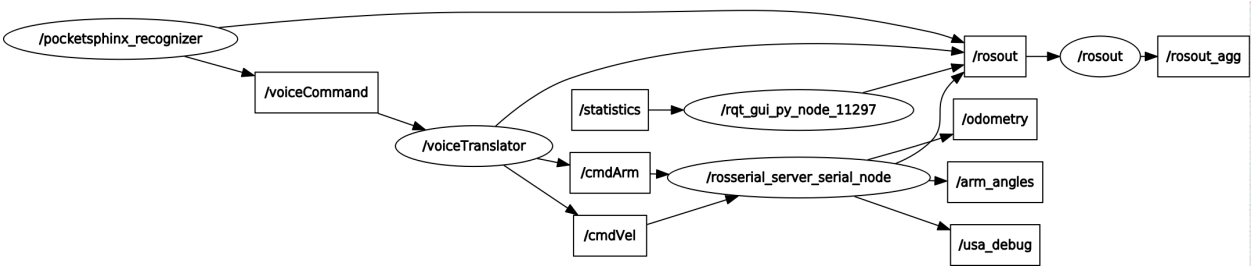
The diagram below shows the pocketsphinx voice recognition node which publishes to the voiceCommand topic the voice commands as string messages.




The diagram below shows the voiceTranslator node which is subscribed to the voiceCommand topic to receive the voice command string messages to turn them into velocity commands which are then published to the cmdVel topic.




The diagram below shows the rosserial_server_serial_node suscribed to the cmdVel topic to receive the velocity commands that originated from the pocketsphinx_recognizer node. The rosserial_serve_serial_node node then executes the appropriate motion commands on the robot's motors.



Below is the code for the voiceTranslator package I created using C++.

```
// glitchy but works with three commands

#include "ros/ros.h"

#include "std_msgs/String.h"
#include "std_msgs/Char.h"

#include "uri_soft_arduino_msgs/DesiredVelocity.h"
#include "uri_soft_arduino_msgs/ArmAngles.h"

#include <istream>
#include <math.h>
#include <string>

#include <unistd.h>
#include <termios.h>

char command = 0;

// function to process the string messages and turn them into keyboard
commands.
void vcTranslator(const std_msgs::String& msg){
    std::string mess = msg.data.c_str();
    if(mess == "forward "){
        command = 'w';
    }
    else if (mess == "backward "){
        command = 's';
    }
    else if (mess == "stop stop "){
        command  = 'p';
    }
    else command = 0;
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "voiceTranslator");

    ros::NodeHandle n;

    // receiving voice command strings from pocketsphinx publisher
voiceCommand
    ros::Subscriber sub = n.subscribe("voiceCommand", 1000, vcTranslator);

    ros::Publisher chatter_pub =
n.advertise<uri_soft_arduino_msgs::DesiredVelocity>("cmdVel", 10);
    ros::Publisher arm_cmd_pub =
n.advertise<uri_soft_arduino_msgs::ArmAngles>("cmdArm", 10);

    uri_soft_arduino_msgs::DesiredVelocity msg;
    msg.lvel = 0.0;
    msg.avel = 0.0;
    msg.seq = 0;
    msg.time = 0.0;
```

```
    uri_soft_arduino_msgs::ArmAngles msg_arm;
    msg_arm.j1 = 60;
    msg_arm.j2 = 100;
    msg_arm.j3 = 95;
    msg_arm.j4 = 50;

    // ros::Rate loop_rate(10);
    sleep(10);

    // publishing the velocity commands
    chatter_pub.publish(msg);
    arm_cmd_pub.publish(msg_arm);

    ros::spinOnce();

    char c = 0;
    bool keepspinning = true;

    std::cout << " Say a command! (e.g. forward)"<< std::endl;

    while (keepspinning && ros::ok())
    {
        c = command;
        switch (c){
            case 'w': {     //forward
                msg.lvel += 0.025;
                break;
            }
            case 's': {     //backwards
                msg.lvel -= 0.025;
                break;
            }
            case 'p': {     //stop
                msg.lvel = 0.0;
                msg.avel = 0.0;
                break;
            }

            default: break;
        }
        // limiting the speed fo the motors
        if (msg.lvel>0.05) msg.lvel = 0.05;
        if (msg.lvel<-0.05) msg.lvel = -0.05;
        if (msg.avel>0.05) msg.avel = 0.05;
        if (msg.avel<-0.05) msg.avel = -0.05;

        chatter_pub.publish(msg);
        arm_cmd_pub.publish(msg_arm);
        ros::spinOnce();
        ++msg.seq;
    }

    return 0;
}
```

This project is not perfect but I was able to deliver positive results. There is a glitching issue with robot that needs to be addressed but I will not be able to fix. Also, the architecture for the implementation of the voice recognition feature is not the most ideal, but since this was just an experimental feature, it is not a major issue at the moment. There are many improvements that could be made to this project in the future. One improvement could be the addition of new commands to the voice recognizer package. Another could be the support for other languages such as Spanish or French to list a few. One of the most important improvements would be resolving the glitching issue which causes the robot to shake. Those suggestions mentioned are a good place to start to continue the work done here.

At the beginning of the semester when I started my project, I had no idea about what I needed to do to start and let alone accomplish something with my project. There was a substantial learning curve I had to overcome with ROS and robot programming that at times seemed never ending. But I stuck with it and towards the end of the semester everything sort of came together in my understanding of it all. All the research, tutorials, videos, readings, error checking, and debugging finally paid off in the end. Working on this project challenged me to enhance my critical thinking and problem solving skills. It also gave me an opportunity to use the knowledge and skills I have learned at URI through all my coursework. This internship at the Artificial Intelligence Lab has helped me improve on my interpersonal skills as well. I has made me a more responsible self starter and has given me more experience working in a diverse team oriented environment. I have learned a great deal from my supervisors Dr. Mandal and Mr. Dekker on what it takes to be a role model and a leader. I believe this experience has definitely prepared me for my future career in software development and I am thankful I had the opportunity to be part of the Artificial Intelligence Lab this semester.