Robot Arm Interfacing

Jacob Smith
Brandeis University

ABSTRACT

The goal of this project was to interface an existing robotic arm to the campus rover delivery robot. The scope of the project was planned to go from the high level details of a ROS package and nodes to the low level servo commands to control the motor and everything in between. The final goal of the project was to use the arm in a campus rover application.

CONTACT

Jacob Smith
Computer Science Undergraduate
jsmtih2021@brandeis.edu
https://github.com/jsmith2021Brandeis/

INTRODUCTION

Implementing an arm to the campus rover not only increases its possible behaviors, but would result in a framework to integrate any electronic component to the robot. In addition, such a project would allow the testing of the concept of interfacing by seeing if a user not familiar with the arm can control it with ROS.

The author already had a background in Arduino Programming. The main source of information for this project were online tutorials and electronics forums and consultation with Charlie Squires (see Appendix A). For the evaluation of the arm's performance, two publications on evaluation of robot structures and measuring user experience where consulted.

The requirements of the project were to interface the ROS based campus rover with a four servo robotic arm and an ultrasonic distance sensor. The servos can each move half a rotation at any speed using the VarSpeedServo library, and the ultrasonic sensor returns the distance in inches when requested. Both components are interacted with using the Arduino microprocessor attached to the arm. That microprocessor is attached to the Rasberry Pi that is the main computer of the campus rover (Figure 5). The goal of the project was to allow these components to communicate with each other and to integrate into ROS.

METHODS

The components created consist of: the startup scripts, ROS Publisher, ROS Subscriber, Arduino Program, and the hardware being interfaced with.

```
Initializing Setup Variables
Starting Roscore and Bringup
Please Unplug and Replug USB on t
d
Updating Script on Rasberry Pi
Already up-to-date.
Printing USB Connections
/dev/ttyUSB0
/dev/ttyUSB1
Starting Local Node
Starting Serial Connection
PI:Starting Sequence


        PI:recieved command: f
Moving Elbowo


        PI:recieved command: f
        ARM:Moving Finger
```

Figure 1. Startup Script Output

```
del list_of_names[:]
print real_name
print real_name  ==  "cant_find_face"
if real_name != "cant_find_face":

    pub_arm.publish("MANIP 1 \n")
    pub_arm.publish("ARM 0 330 0 20 \n")
    print "MANIP 1 \n"
    rate2.sleep()
    pub_arm.publish("MANIP 0 \n")
    pub_arm.publish("ARM 0 0 60 20 \n")
    print "MANIP 0 \n"
    rate2.sleep()


print get_name, frame_counter
rate.sleep()
```

Figure 2. ROS Publisher to wave the arm when face viewed

RESULTS

The startup scripts allow the appropriate ROS nodes to start and allow the arm to be controlled by a ROS Publisher (Figure 1). The ROS Publisher publishes to the armcommand topic. The command is formatted in terms of a component tag, along with integers for required parameters (Figure 2). The Arm Interface Node is run locally on the Rasberry Pi on the robot because it needs to communicate to the Arduino. This node establishes the Serial connection to the Arduino on the arm and subscribes to the armcommand topic. Finally, the Arduino Program running on the Arm itself parses data coming over the serial connection and actually moves the arm and returns the distance to the ground (Figure 3).

Goal	ROS Command	Arduino Command	Inputs other than Name	Outputs	Comment
Get Distance	Published to Arm Response topic	Automatic		Distance	Type Output Sensor
Set Arm Coordinates	ARM x y base power	set_arm(100, 200, 90, 0 servoSpeed);	4 coordinates, speed		Type 4 Coordinates
Set Manipulator	MANIP setting	manipulator(true =open/false =code);	open or close		Type Binary Manipulator

Table 1 Electronics Interfacing Commands to control the arm

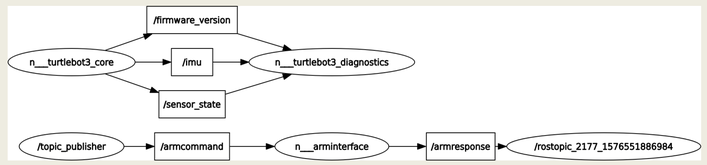

Figure 4 ROS nodes involved in controlling the arm (made with rqt_plot)

```
//move the manipulator to the specified st
void moveManip(int state) {
  //if state is set to close, open the fin
  if(state==0){
    fingerServo.write(35, 50, false);
  //if state is set to open, open the fing
  }else if (state==1){
    fingerServo.write(160, 50, false);
  //if state is set to stop, stop the fing
  }else if (state==2){
    fingerServo.stop();
  //otherwise print error
  }else{
     Serial.println("ERROR: This Manipulat
  }
}
```

Figure 3. Move manipulation segment of Arduino program

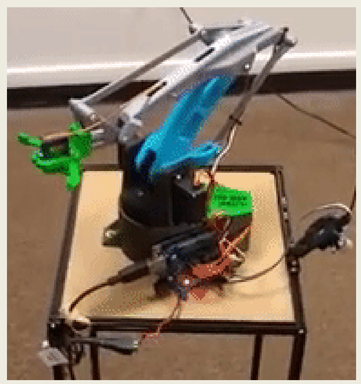

Figure 5 the robotics arm mounted on the campus rover

DISCUSSION

The final project is a comprehensive electronics interacting system that allows the arm, manipulator, and distance sensor to be controlled using ROS framework using abstractions. The interfacing scheme is represented below, and could easily be expanded for other electronics components. The main concept is to separate the command of a component from the low-level instruction, similar to how a user can publishcmd_vel and control a dynamixel motor. The figure below represents the existing commands that can be sent to the arm, utilizing both inputs and outputs (Table 1 and Figure 4).

A major goal of the project was to see how useful it is to people not familiar with Arduino who want to contribute to Campus Rover. To that end, the startup scripts where tested with a TA for the class, who was able to get the arm running using the instructions. In addition, the face recognition team was able to control the arm simply by publishing to the arm command topic, and not going to a lower level.

In future work, sensors and a ROS action could be implemented to allow for more complicated movements, such as grabbing a package. That would be implemented by finding the location of a package and initiating an action to move the arm to those coordinates. The action would then publish to the arm command topic.

CONCLUSION

In summary, the robotic arm is now mounted on the campus rover and can be controlled with ROS commands. In addition, the project presents an overview of interfacing and abstraction problems and should be a strong foundation for other hardware additions to the campus rover robot.

In conclusion, I wish to thank the COSI 119a TAs, Professor Salas, and Mr. Squires for their support of this project, and I hope future students find my work useful.

REFERENCES

Final Report https://campus-rover.gitbook.io/lab-notebook/reports/robot_arm (this poster is an adaptation of my report)

Github Repository: https://github.com/jsmith2021Brandeis/Robot-Arm-Interface

Brandeis Seal: https://www.brandeis.edu/brand/images/logo/seal.jpg, Turtlebot 3 https://www.roscomponents.com/en/mobile-robots/214-turtlebot-3.html (Figure 2 written by Luis Andino)