EE565: MOBILE ROBOTICS

LAB # 3: HARDWARE INTERFACING OF ARDUINO WITH ROS

DESCRIPTION & MOTIVATION

Once the students are familiar with a Robotics engine (ROS) & a simulation environment (Gazebo), an important aspect to learn is the hardware interfacing part. Real world experiments rely on such skillset and almost all research work in robotics, today, requires a sound experimental validation that takes into account physical, mechanical and other real-world inaccuracies and non-modeled discrepancies.

This lab combines basic knowledge of robotic framework, ROS, with a simple & easy-to-use hardware controller that uses Arduino board. We will provide each student group with a motion controller board, a serial communication interface, with a ROS-installed PC. Students will learn to interface the Arduino board with ROS from scratch. In the end, they will be able to make a standalone system driven from a ROS node that controls Arduino board based on required robotic application.

IN-LAB WORK

1. ARDUINO BASICS:
   i. Install Arduino IDE using `sudo apt-get install arduino`.
   ii. Run by executing `arduino` in terminal.
   iii. Select the correct Tools->Board and the right Serial Port. If your Serial Port option is greyed out, run `sudo chmod a+rw /dev/ttyACM0`.
   iv. Follow SerialEvent example, burn it to your board and check by sending any string using Arduino IDE's Serial Monitor. [You may use a character other than end-of-line, in the code, if using Serial Monitor to mark string completion.]
   v. Download and run the motor speed example, 'motorSpeed.ino' available on LMS for direction and speed control. Modify it to for different speeds and direction.
   vi. Run the quadrature encoder example, 'encoder.ino' to periodically transmit motor velocity, and view it on Serial Monitor. Modify it to print motor speed in radians/second.

2. INTERFACING ARDUINO & ROS:

Open the rosserial_arduino tutorials {url}, and complete the following

   i. Arduino IDE Setup (this will install the Arduino library “ros_lib”)
   ii. Hello World (Example Publisher)
   iii. Blink (Example Subscriber)

OPTIONAL (WILL HELP FOR LAB ASSIGNMENT):

Instructions will be given for the following

- Use of turtlebot_teleop package instead of basic turtlesim package.
- Installing and running PID library for Arduino.

DELIVERABLES:

In lab work has five deliverables: 1.iv, 1.v, 1.vi, 2.ii and 2.iii. Kindly get all of these checked in due time.

LAB ASSIGNMENT

Build a complete DC Motor Speed Control application, interfaced with ROS. Use the Arduino code available on LMS. Each group will be provided with the following equipment:

- Motion controller board (H-Bridge + Arduino Board)
- DC Motor having an attached encoder sensor.
- Cable for serial communication between PC and Arduino

Boiler code for Motor Speed Control (using PID library) is available on LMS. This should be interfaced with ROS framework, through ROS Topics. Motion controller will take a reference motor speed as input from the serial port, and with its built-in feedback loop, control the DC Motor. The controller will also publish the Odometery data (current motor speed) to another topic for internal ROS use (as geometry_msgs/Twist).

DELIVERABLES:

1. Publish motor encoder data as rostopic. This will require writing a publisher node in Arduino code that will take the encoder's data to publish to a ROS topic (geometry_msgs/Twist in revolutions/second). This topic may not be used inside ROS for now, but it should be visible and working. [For this deliverable you can write your own Arduino code as well.]
2. Using the sign of linear velocity (x-axis) (i.e. +ve/-ve) from turtlebot's cmd_vel to decide the direction of motor rotation. This will require a subscriber node. Move the motor with any constant speed with the given direction.
3. Implement speed control. Based on turtlebot's linear velocity (x-axis), specifiy reference speed and direction for your motor. In the end, your motor should move according to the speed and direction of turtlebot. [It is highly recommended to use turtlebot_teleop package instead of turtlesim.]