RoboJay Alpha: Navigation on JHU Campus

David Samson

530.707 Robot Systems Programming Independent Class Project
Johns Hopkins University | Whiting School of Engineering | Baltimore, MD

Introduction

RoboJay is project started in 2016 in the Johns Hopkins Robotics Club with the goal of creating a robot tour guide that can navigate campus. Before the start of the 2018 Robot Systems Programming course, a frame and basic electrical hardware (motors, battery, controllers, etc.) had been procured. The goal for the project in 530.707 was to develop the back end robotics software in ROS that would be necessary to allow RoboJay to navigate campus. Additional goals for the project included developing a control algorithm to balance RoboJay upright, and developing rudimentary obstacle avoidance algorithms.

RoboJay is designed to function in the same capacity as a regular Hopkins tour guide. This means RoboJay would be able to lead a group of people around campus, and interact with them by talking about campus, and potentially responding to questions (thought all human interaction aspects of RobJay were deemed out of the scope of this project). In order to be able to lead people around campus, RoboJay needs to be able to navigate campus itself. Because the route that RoboJay follows can be fixed/preplanned, RoboJay simply needs to be able to localize its position on campus, and then move along a precomputed path.

While RoboJay gives a tour, it will stay confined to the brick paths on campus (in fact, because a RoboJay tour could be classified as a special tour, it would be reasonable for the tour path to be short and or confined to easily traversable sections of campus, e.g. the path around Decker Quad). As such, RoboJay's localization needs to be precise enough to allow it to stay within the boundaries of the paths which are several meters wide at any point.

To accomplish localization, RoboJay would employ sensor fusion via the robot_localization package to compute a position given sensor information from onboard GPS, rotary encoders, and a 9-axis IMU.

Hardware and Infrastructure

Existing available hardware employed by this project include the following:

- RoboJay: A two-wheeled robot, roughly the size of a person designed to navigate campus and take prospective students on tours. Developed by the JHU Robotics Club.
  - RoboJay included the motors and rotary encoders to be used for this project.
- EduMIP Kit: The robotics kit employed in 530.707 for developing a small self balancing robot. Mainly the BeagleBone Blue from the EduMIP kit was used
- Netbook: a simple computer running Ubuntu with an installation of ROS

New hardware purchased for this project include the following:

- Portable WiFi router
- U-blox USB GPS sensor

No new hardware was designed or fabricated for this project

Figure 1—RoboJay Images
Left to Right: RoboJay at the start of the project, side view, and internals used for localization.

Figure 2—Block Diagram
Layout of communication between components of RoboJay's systems

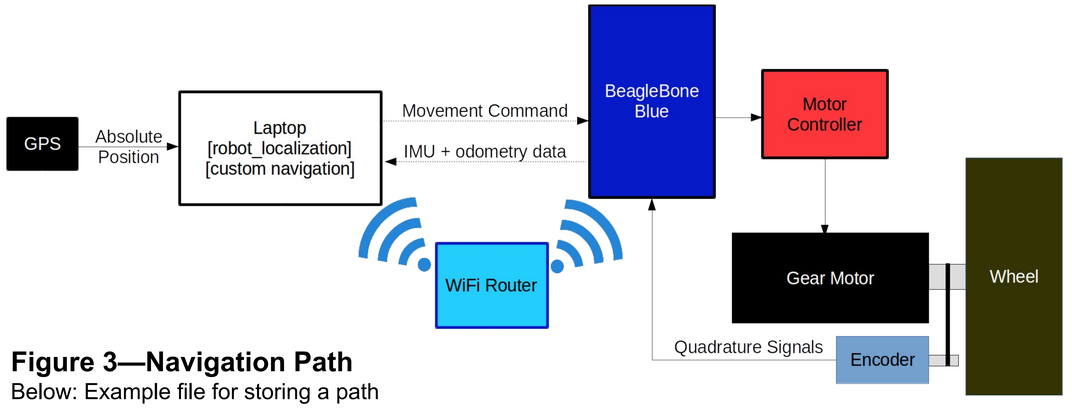
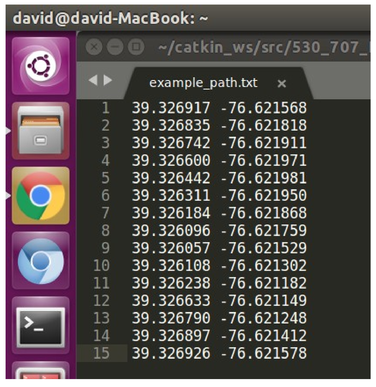
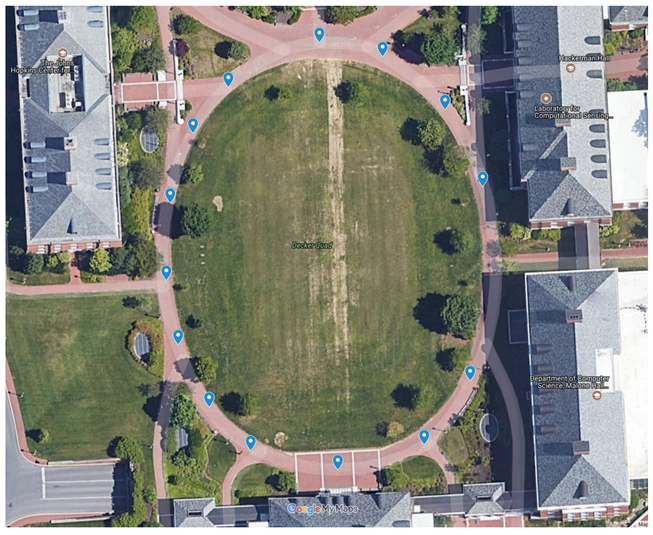


Figure 3—Navigation Path
Below: Example file for storing a path
Right: Overlay of waypoints on map





Software

Existing available software employed by this project include the following:

1. robot_localization: ROS package for fusing sensor data with Kalman filter to produce estimate of robot position: http://wiki.ros.org/robot_localization
2. ROS GPSD Client: ROS client for running GPS sensor: http://wiki.ros.org/gpsd_client
3. BeagleBone Robotics Cape library: https://github.com/StrawsonDesign/Robotics_Cape_Installer

New software that we designed and coded for this project include the following:

4. robojay_bbbl_controller.cpp: ROS node to run on BeagleBone. Reads in rotary and sensor information, and publishes them on topics for the robot_localization package to use. Would also control balancing RoboJay: https://git-teach.lcsr.jhu.edu/dsamson4/530_707_Independent_Project/blob/master/src/robojay_bbbl_controller.cpp
5. robojay_laptop_controller.cpp: ROS node run on laptop. Reads sensor topics and publishes transforms (map, odom, base_link) for robot_localization. Also runs the navigation program: https://git-teach.lcsr.jhu.edu/dsamson4/530_707_Independent_Project/blob/master/src/robojay_laptop_controller.cpp
6. dirty_controller.cpp: Alternative node run on laptop which ignores the robot_localization package and computes position/orientation directly from averaged sensor data. Used for project demo due to frame issues with the robot_localization package: https://git-teach.lcsr.jhu.edu/dsamson4/530_707_Independent_Project/blob/master/src/dirty_controller.cpp

Lessons Learned

- Be careful to scope the project to the number of people on the team. The original project proposal contained a much larger scope than was ultimately completed for this project. What was proposed would have been a good project if three or so people were on the team, however since this project was attempted by only one person, much of the project had to be cut back.
- Get something quick and dirty working before doing any optimization. After writing and testing the dirty_controller (which was developed because there were issues with improperly formatted frames preventing the robot localization package from functioning properly), it was discovered that the raw GPS signal averaged over several sensor readings was both accurate and frequent enough to provide a position to be used by the navigation program, without the need for the robot_localization package. If I had tried writing this at the beginning, I probably would have skipped using the robot_localization package, and worked on some of the other aspects of the project that had to be cut.

References and Acknowledgements

1. Thanks to Max Novick for his prior work on the development of the RoboJay
2. Credit to Google Maps for RoboJay path image