EC 500 D1 Final Project Report

Team BenchBot:
Mike Ferguson
Jo-Ann Meunier

Description:

BenchBot controls a lab robot that aims to be a low cost, easy, and efficient solution to automating bench scale lab tasks. It was used to control a 4 axis, stepper motor controlled robot arm (a Dobot arm) in this case. The software is designed to give the user control over the robot and allow them to add custom “tasks”. A task is any self contained set of operations that the robot should perform (e.g. a protocol). On the software’s menu bar, there are different groups (e.g. a “Cloning” group). Hovering over the cloning group shows a drop down menu with associated tasks (e.g. PCR). Upon clicking on the task menu item (PCR in this case), a dialog pops up that allows the user to configure various properties of the task (number of samples, locations, etc…). The user can add a new task menu item (in a new group if desired) by uploading three files:

1. .ui file - Qt gui file used to create the pop up task dialog gui
2. .json - describes locations of the various items in the robot’s workspace (e.g. tips, tube rack)
3. .py file - contains all of the logic needed for interacting with the GUI and robot to perform the given task

Attributions

Mike - Wrote the manual control GUI software, the arduino firmware/serial protocols, and the inverse kinematics. Also developed the electronic pipettor.
Jojo - Wrote all of the non-manual control GUI software, json file editor, and the 3D workspace. Developed and wrote the pathfinding algorithm.

Major Software Components:

Movement:

The DobotGUIMain.py file contains a stand alone user interface that one can use to manually control the robot arm. A DobotInverseKinematics.py file is used to convert cartesian (x,y,z) coordinates to angles for the robot’s base, upper arm (connected to the base), and lower arm (connected to the upper arm and the end effector). An algorithm is used to generate a sequence of steps that will move the robot in a smooth, straight line from point A to point B. Angle checking, out of bounds, and limited user interaction error checking (e.g. typing a letter instead of a number) is implemented.

Arduino Code

The python code communicates serially with the arduino (C/C++ code). Two different serial protocols were developed: one that just moves to a point using the AccelStepper library one that uses a handshaking algorithm to send packets of step sequence data to the arduino and move the robot in a smooth, straight line. For a more in depth discussion of the code and algorithms, see the relevant readme file in our github repository: https://github.com/jmeunier28/BenchBot/blob/master/NewestManualUserInterfaceFiles/readme

Graphical User Interface:

The graphical user interface was written in python using PyQt creator. This design choice was made due to the fact that PyQt creator makes it very simple for someone with little programming experience to design a UI. Because we wanted to make it possible for other biologist to contribute new and different kinds of tasks, using PyQt was the best way to give them this option. Python is also a very popular language amongst people with little computer science background. The GUI allows for the user to select which task they would like BenchBot to perform for them and then visualize a workspace for that task by taking a user defined json file and rendering objects from it in 3D. This was done using PyOpenGL, which is a python binding for OpenGL's C++ libraries.

Path Finding Algorithm:

The path finding algorithm is written in python, and implemented in the BenchBotGUI.py file. It is used as a way to ensure that the robot avoids objects that have been defined within the workspace. This ensures that nothing will go wrong when the user defines the path the robot should travel during an experiment. In order to do this we defined all possible nodes in a 3D space that the robot could travel to. Then the nodes that the robot could not travel to were defined based on the dimensions and coordinate locations given in the workspace configuration file (json file) and removed from the list of nodes the robot is allowed to move to. From there checks are put in place in which each time a call is made for the robot to move it checks if any of the points in between are ones that should not be traveled to. Mathematically speaking a node, c would lie on the path in between two points a and b if the cross product of (b-a) and (c-a) is 0 and if the dot product of (b-a and (c-a) is positive and less than the square of the distance between a and b. If a point lies between the initial and final location, but is not within the array of

available space the robot may travel to then the path is adjusted for this. A new initial location will be assigned above the obstacle and the robot will travel there and then its path will be checked again until it finally reaches its planned destination.

Special Instructions to Compile:

This project has several dependencies that the user must install before working with the BenchBot. The following are necessary packages:

1. PyQt5
2. QT Creator 5.6
3. Python 3.5
4. PyOpenGL 3.0
    a. This must be installed from the source because GLUT is necessary for rendering the 3D cubes in the GUI. Installed from source ensures the user will have this OpenGL package. More documentation can be found on this here: http://pyopengl.sourceforge.net/documentation/installation.html
5. A program to upload the .ino arduino files to the arduino. Standard one available here: **https://www.arduino.cc/en/Main/Software**

In order to run this program from the command line the user can type the command:

```
python BenchBotGUIMain.py
```