Project Documentation

Cory Hungate

Project Description:

For my final project I used a MIDI controller as a physical controller for a robot; the robot in this case was a turtle in turtlesim as a stand-in for physical hardware. The Midi controller was connected to a MIDI-in circuit on a breadboard and run through an Arduino Mega2560 which passed the information into a ROS2 node via serial. I used knobs on the MIDI controller to control the speed and direction of the turtle as well as the RGB values of the "pen" (i.e. the color of the path that the turtle traces on the screen) and used keys on the MIDI controller to execute service calls. Additionally, I was able to send data from the ROS2 system back to the Arduino for visualization on a 4-digit LED display circuit.

System Description:

Physical Hardware: This system uses an Arduino Mega2560, Oxygen 8 MIDI controller, and assorted circuit components. See 'Documents' folder for circuit diagram and bill of materials.

ROS2: ROS2 portion of this project was developed in Jazzy Jalisco

This project uses 7 nodes and 2 launch files. Descriptions of each are found below.

Nodes

change_pen.py - a node that monitors the 'knob_data' topic, looks for information from knobs 6-8 on the Oxygen 8 MIDI controller, and uses them to change RGB values of the pen (knob6 for red, knob7 for green, knob8 for blue), sets values for the RGB values based on this data, then changes the turtlesim pen color on a service call. This Node also publishes the value of each color change as a 4-digit integer (1---, 2---, and 3--- for RGB respectively) to the topic 'rgb_data'. The actual color change is handled by a service call that is initiated when key pitch 60 is pressed on the MIDI controller.

clear_path.py - a node that monitors the topic 'keystroke_data' for a keystroke on key pitch 70 on the MIDI controller; the service to clear the turtlesim pen path is called if/when the keystroke is detected,

knob1_node.py - simple publisher-subscriber node that subscribes to the 'knob_data' topic and republishes the data if a change to knob-1 is found on the topic 'knob1_data'.

knob2_node.py - simple publisher-subscriber node that subscribes to the 'knob_data' topic and republishes the data if a change to knob-2 is found on the topic 'knob2_data'.

serial_data_publisher.py - this node is the heart of this project. This node monitors the serial port for incoming MIDI data from the Arduino. 'Note On' data is republished as an Int64 to the topic 'keystroke_data' where the message is just the pitch value of the note and knob data is republished to the topic 'knob_data' as an integer array of size 2 with the 0-index as the knob that was turned and the 1-index as a 0-100 value that represents the new value of the knob. This node also subscribes to the topic 'rgb_data' from change_pen.py and sends that data to the Arduino through serial.

turtle_killer.py - a node that monitors the topic 'keystroke_data' for a keystroke on key pitch 48 on the MIDI controller. if the keystroke is detected, the turtlesim service 'kill turtle' is called.

turtle_movement.py - this node subscribes to 'knob1_data' and 'knob2_data' and uses those values to control the movement of the turtle in turtlesim. knob1 controls the x-velocity and knob2 controls the angular velocity.

Launch Files

turtle_launch.py – This launch file launches an instance of turtlesim as well as an instance each of knob1_node.py, knob2_node.py, and turtle_movement.py.

turtle_launch2.py – this second launch file launches an instance of serial_data_publisher.py, turtle_killer.py, clear_path.py, and change_pen.py

Arduino:

There is only a single Arduino file for this project: serial_reader_advanced.ino
This file reads the incoming MIDI data and sends it as a string via serial to the ROS2 part of this project. It additionally reads serial when data is sent to the Arduino and displays that data to the LED display.

Running the System:

The instruction below can also be found in the README document.

Note: You're going to need to install turtlesim on your machine. instruction for downloading are found here: turtlesim

First, open the arduino folder and launch 'midi_reader_advanced.ino'. Compile and upload the file to the Arduino.

NOTE: you will probably want to remove the wire to PIN 19 when uploading if the controller is plugged in, incoming data from the controller can sometimes interfere with the upload.

Next, plug in the controller to the MIDI-IN circuit on the breadboard. In order to use the keys as mapped in the nodes you need to select the correct octave on the controller. Select the "octave/presets" buttons until "--0" is displayed. In this octave, the middle C key is "pitch 60" (calls the service to change the turtle path color), low C is 48 (call the Kill Turtle service), and high C is 72 (call the clear path service)

Before running the ROS2 nodes you'll need to configure the arduino connection to your virtual machine (if you have not already done so). First, find the port on your machine that the Arduino is using for serial connection: make sure the Arduino is unplugged then, in the terminal, run the line:

```
ls /dev/tty*
```

This will give you a list of all the available ports. Now plug in the Arduino. If using VirtualBox, you must also go to Devices->USB and select the Arduino device listed (Arduino Mega2560 for this project) to connect the Arduino to the VM. Again run:

```
ls /dev/tty*
```

There should be a new port listed that wasn't there before. that is your serial port.

IF THIS PORT IS DIFFERENT THAN THE ONE ASSIGNED IN 'serial_data_publisher.py' THEN YOU NEED TO CHANGE THE ONE IN THE NODE

If You also likely need to give yourself permission in order to access the serial port. run the following in terminal:

```
sudo usermod -aG dialout $USER
```

then log out and log back in or reboot the machine

To start up the ROS2 nodes, launch the two included ROS2 launch files with the following commands:

- ros2 launch rob499_project turtle_launch.py
- ros2 launch rob499_project turtle_launch2.py

NOTE: these files should be launched in this order. For some unknown reason the files contained in launch2 fail to launch when included in turtle_launch.py
These launch files will launch every node used in this project.

Play around in turtlesim!