**ECE 578/478 Final Report**

Nixon-Bohr, formerly Bohr

11/19/2013

Jordan Fluth, Vernon Jones



Contents

# Description

Our project modified Bohr to be larger, and added software to control the modified robot using a Microsoft Kinect and a Lego NXT.

A Halloween robot's skeleton was grafted onto the existing Bohr robot. Bohr's existing head was added to the top of the robot, while his existing arm was left at the base. We wrote a suite of programs that control the wheels and the arm of the new Nixon-Bohr robot. This suite uses a Microsoft Kinect, a Visual Basic script, Bluetooth, and a Lego NXT.

The Microsoft Kinect uses the skeleton trace feature of the upper half of the body to recognize body positions and motion. These positions and movements are converted into keyboard key presses. The key presses are input to a Visual Basic script, which transmits the key presses as ASCII characters over Bluetooth to the Lego NXT brick. The NXT brick converts these ASCII characters to motor and servo movements. Using this method we achieve wireless user-defined remote control over the robot.

The motions to control the robot are simple. The motor control uses height differences between wrists and elbows to send directional commands. Move both wrists above their elbows for forward, move wrists below the elbows for backward. Move the left wrist above the right wrist for left, and the opposite for right.

Short description what was done, what was achieved. To be used by me in the combined journal/conference papers that I am writing. This part should be about 1 page and should not go too deeply to technical aspects. You will be co-authors of these papers and we will keep working on them next quarter. We published several papers with students from this class, based on their work.

# Technical documentation

1. Components and subsystems
   1. FAAST Program – Translates gestures detected by the Microsoft Kinect into key presses
   2. Personal Computer with Bluetooth capability – This is the central hub for receiving and transmitting data.
   3. Lego NXT Brick – Provided by the class. The brick runs a RobotC program that translates commands sent over a Bluetooth connection into motion.
   4. Two Hi-Technic Motor Controllers – Provided by the class. One of the controllers controls the wheel motion. The other controls shoulder and elbow motion.
   5. One Hi-Technic Servo Controller – Provided by the class. This controls the motion of the hand, wrist, and left and right rotations of the shoulder.
   6. Microsoft Xbox 360 Kinect – Provided by the class. This uses two cameras and a range finder to give the user vision with depth. It also uses a skeleton trace which is needed and used by the FAAST program.
2. Control Motions
   1. Both Modes:
      1. Toggle Modes: left hand to the left, quickly
      2. Finish: both hands outward, quickly
   2. Motor Mode:
      1. Forward: both wrists above elbows
      2. Backward: both wrists below elbows
      3. Left: left wrist above right wrist
      4. Right: right wrist above left wrist
      5. Stop: both wrists near both elbows in height
   3. Arm Mode:
      1. Hand Open: right hand to the right, medium quickness
      2. Hand Close: right hand to the left, medium quickness
      3. Wrist Up: Not defined yet
      4. Wrist Down: Not defined yet
      5. Shoulder Left: Not defined yet
      6. Shoulder Right: Not defined yet
      7. Shoulder Up: Not defined yet
      8. Shoulder Down: Not defined yet
      9. Elbow Up: Not defined yet
      10. Elbow Down: Not defined yet
3. Notes
   1. The Hi-Technic Motor and Servo controllers time out after a period of inactivity. Cycle their power to reactivate them
   2. Ensure the battery being used has sufficient power, otherwise the robot will not function well
   3. You may need a Bluetooth donge with Cambridge Silicon Radio chipset. See Lego's Bluetooth documentation at <http://www.lego.com/en-us/mindstorms/support/bluetoothsupport/>

Technical documentation to software of your robot. The code must be very well documented. Comments for each subroutine, each variable, each block of code. The source code should be in separate files. BUT IT MUST BE ALSO IN THE MAIN DOCUMENT, WELL FORMATTED.

* What technical troubles did you find in your work? How you solved them, ultimately?
* Write a complete “troubleshooting manual” for the next users of your robot. The weakness of our previous projects was that the next students were starting all-over from scratch as there was no good documentation. Now it has to be dramatically changed.
* Give your personal advice to the next students about the project work like yours. This must be directly related to your project but should be written in a more general way. How should they organize their work? Where should they find information? How should they set up meetings? Where to look for information? For Components? Useful webpages and books. What should be added to my class slides that would help in this project?

# Translating this Knowledge to ECE 479/579

We will build off of our knowledge from this quarter to add more fine-grained control of Nixon-Bohr, something we didn't have time to do for this quarter. We would especially focus on the arm in the next quarter, since the wheels work well, since the arm is much more technically challenging than the motors. Solidifying the code to control the motors allows us to start from a good baseline next quarter.

How you may use knowledge that you learned in ECE 478 in the next quarter to add more interesting perceptions and behaviors to your robot? The goal of this part is that you will show your knowledge and critical/creative understanding of class material. You will be not asked to execute what you write here, it is only a plan. This part should be especially well written by students who take it for ECE 578 credit. Add as much technical/mathematical detail as you deem useful to explain your idea in full detail.

# Technical or Research Problems

1. Nixon-Bohr is heavier than his previous incarnation was. Adding more power to his wheels would help a lot on this. One way to do this would be to add motors to the back wheels. These motors could be in series with the motors on the front, or they could be independent.
2. Nixon-Bohr's face is not utilized by our code. Later versions of our code could be used to mimic the facial expressions of the controller.
3. The arm of the robot is mounted to the base of the robot. This arm could be moved to where the arm should be, mounted to Nixon-Bohr's shoulder, so he could shake hands.
4. Nixon-Bohr has no sensors; he is entirely remote-controlled. Adding bumpers, range-finders, and/or cameras in useful locations could significantly improve the operation of the robot. Adding a camera around Nixon-Bohr's eyes could send feedback to the controller's computer, adding a first-person view of what the robot is seeing.
5. Using the newer version of the Kinect could add more fine-grained control to the robot. The newer version of the Kinect adds finger tracking, meaning the controller's arm and hand could be used directly to control Nixon-Bohr's arms and hands.
6. Add larger wheels to allow Nixon-Bohr to navigate more complex terrain than carpets.
7. Add control for a second arm

Technical or research problems that should be solved to achieve the goals set to your robot better. For instance, iSOBOT has no sensors. What kind of sensors you can propose to give feedback to the computer to create complete behaviors – for instance camera.

# Instructions

**Step by Step:**

1. Setup FAAST and Kinect and interface between the two
   1. To start download and install the Microsoft Kinect for windows SDK. It doesn't matter if you have a XBOX 360 Kinect; it will still work. [Click Here to download](http://www.microsoft.com/en-us/kinectforwindows/develop/)
   2. Download FAAST. FAAST translates movements on the Kinect into functions on a keyboard or keypad. [Click Here to download](http://projects.ict.usc.edu/mxr/faast/)
   3. Connect your Kinect to your PC and allow the drivers to install from the SDK and the Windows Database.
   4. Once the drivers have been installed, open the FAAST program that you downloaded earlier.
   5. From the Tracker drop down menu, select Microsoft so that the program will know that you are using the Kinect with the drivers you've installed.
   6. Click the connect button.
   7. Next, click Gestures. Here you can make the movements you do that are recorded by the Kinect could translate to a key, or multiple keys on a keyboard.
2. Click [here](http://www.extremenxt.com/vbpart1.htm) for instructions on setting up a Visual Basic script to communicate over Bluetooth between the computer and the Lego NXT brick.
   1. Notes: We will be modifying the Visual Vasic script so that the first textbox will be for debug purposes and the second textbox will be for operational purposes.
   2. Additional instructions not on the website:
      1. Add a second textbox and a third label.
      2. Make the third label say “Auto Send”
      3. Make the second textbox say “button click” when the textbox changes.
      4. Add code to clear the text box at the end of the function.
3. Install ROBOTC on your computer and setup the IDE for your application (connect NXT to PC through USB)
   1. Use the configuration wizard to setup servos and motors
      1. Go to menu: ROBOTPLATFORM TYPELEGO NXT + TETRIX/MATRIX
      2. Go to configuration wizard: ROBOTMOTORS AND SENSORS SETUP
         1. Click tab: \*\*\*
            1. X
         2. Click tab: \*\*\*
            1. X
         3. Click tab: \*\*\*
            1. X
4. Copy the ROBOTC source code, referenced below, to the IDE. Ensure it compiles
5. Download the ROBOTC source code to the NXT
6. Connect the computer to the NXT via Bluetooth connection using the Visual Basic code. You should hear a beep when the Bluetooth connects
7. Start the program. If a Bluetooth connection has not been established, it will beep and halt the program
8. Start FAAST. Select “Start Emulation.” Select the textbox in the VB application
9. Move back until FAAST sees your skeleton. You may now control the robot

# Project Code

## Visual Basic Code:

## ROBOTC Code: