R.A.P.T.O.R. Kit

Robotic Arm Project for Teaching and OutReach Kit

**1. Abstract**

The goal of this project is to create self-contained robotic arm demonstration units for educational outreach events and demonstrations. The units must be easy to use and child friendly. This will be done through the use of single board computers, microcontrollers, simple electronic modifications, and a video game controller. This system will be known as the Robotic Arm Project for teaching and OutReach (R.A.P.T.O.R.) kit.

**2. Systems Engineering**

**2.1.   Deliverables**

This project's goals outline three specific deliverables to be produced by the end of the project:

Create Self Contained/Autonomous Units

    2. Ensure the units are wall outlet powered

    3. Xbox 360 Controller Enabled

In order to increase the effectiveness of these standalone units, instructional scenarios built around the specific capabilities of these units need to be developed and tested.

**2.2.   Timeline**

A timetable has been developed to guide the three phases of development of this project: software, hardware, and instructional scenario. See Appendix A for calendar.

**3. Procedure  
3.1.   Software Development**

3.1.1. Preliminary Design - Phase 1S

3.1.1.1. Windows Implementation Trial

The OWI-535 Edge Robotic Arms comes with software for windows that enables one to connect the motors on the arms to keyboard and controller inputs. A simple proof of concept will be performed using this windows software and Xbox 360 controller plugin to prove that the controller is an easier to use device the provided controller. Required software for this trial is as follows:

* Robotic Arm Edge Software (freeware)
  + Released by manufacturer of arm
  + Windows only (confirmed on Windows 7, 8)
  + Requires drivers (unsigned, must be force installed)
  + Communicates with arm over USB
  + Maps keyboard input to motor movement on the arm for direct drive
* J2K (freeware)
  + Windows only (confirmed on Windows 7, 8)
  + Maps inputs from game controller or joystick to keyboard buttons
  + Confirmed with both wireless and wired Xbox 360 controllers
  + Mapping profiles can be saved and loaded
* Xbox 360 Controller Drivers (freeware)
  + Can be acquired from Microsoft
  + Support both wired and wireless controllers
  + Confirmed on official controllers and Afterglow Wired 360 controller

The first step in this trial is to install and format the required software. This can be done by following the procedure listed below:

* Step 1: Xbox 360 Controller Drivers
  + Follow install process
  + If controller is not recognized, manually install driver for controller in Device Manager
    - Update Driver Software
    - Browse my computer for driver software
    - Let me pick from a list of device drivers on my computer. CHoose one of the following:
      * Xbox 360 Peripherals
      * Xbox 360 Controller for Windows OR Xbox 360 Wireless Receiver for Windows)
* Step 2: Robot Arm Edge
  + Follow install process
  + Device driver is unsigned. If prompted, install anyway. If driver install fails, disable driver signature verification and install manually from file location (through Device Manager)
    - Run software
    - Select Basic
    - Change Mode in lower left corner to Keyboard. Lower right corner should show “USB Device Connected” if robot is connected with cable.
  + Must be the active window while operating robot.
* Step 3: J2K
  + Launch J2K.exe
  + File -> Load and select Mapping file
  + Leave running in the background while operating the robot

Once the set up procedure is finished, the Xbox 360 Controller is ready to be mapped to the motors.

**Mapping Instructions??????**

When these instructions are combined with the hardware setup described in section 3.2.1.1., a trial system will be fully setup to allow the Xbox Controller to be used to control the arms.

3.1.1.2. Raspberry-Pi Initial Setup

In order for the Raspberry-Pi to be able to control the arms, it must be formatted properly. The following procedure describes these steps in detail:

* Update and Upgrade
* Install xboxdrv
* Install python software properties
  + $ sudo apt-get install software-properties-common python-software-properties
* Install pyusb
  + $ mkdir ~/pyusb
  + $ cd ~/pyusb
  + $ git clone https://github.com/walac/pyusb.git .
  + $ sudo python setup.py install
* Install subversion
  + $ sudo apt-get install subversion
* Obtain robotarm files
  + $ mkdir ~/robotarm
  + $ cd ~/robotarm
  + $ svn co http://projects.mattdyson.org/projects/robotarm/ .
  + $ git clone https://github.com/zephod/lego-pi.git legopi
  + $ touch legopi/\_\_init\_\_.py
  + $ sudo python driveRobot.py
* NOTE: legopi must be inside robotarm
* Blacklist xpad (may need to process kill it manually unless you restart after blacklisting)
  + $ cd /etc/modprobe.d
  + $ sudo nano raspi-blacklist.conf
* Add line “blacklist xpad” without quotes to bottom of file and save on exit
* Modify robot drive files as needed
  + driveRobot.py
  + Modify controls
  + Suppress all print statements
  + Add GPIO communication to turn light on on start
  + Save under different name
  + /armcontrol/RobotArm.py
  + Suppress all print statements
  + Save under same name
* /legopi/lib/xbox\_read.py
  + Remove word “nohup” from “subprocess” line under “def event\_stream”
  + Add GPIO communication to turn light off under “if ‘Error’ in line:” and turn light on under “if (len(data)==42:”
  + Add “GPIO.setwarnings(False)” (add directly after import)
  + Save under same name
* shut off login
  + In Terminal:
    - sudo nano /etc/inittab
      * Scroll down to:
    - 1:2345:respawn:/sbin/getty 115200 tty1
      * and change to
    - #1:2345:respawn:/sbin/getty 115200 tty1
      * Under that line add:
    - 1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
  + Ctrl+X to exit, Y to save followed by enter twice
* Set­­­­­ up shutdown button with code from here
  + <http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=55410>
* Set up python script to reset the controller on button press using the same format as the shutdown script. Use pkill –f to kill the drive script and xboxdrv then start the drive script again
* Setup code to run from start
  + sudo nano /etc/rc.local
  + add lines “(python $scriptlocation)&” for robot drive script, shutdown script, and

3.1.2. Prototype Development - Phase 2S

3.1.3 - Critical Design Review - Phase 3S

3.1.4. System Verification - Phase 4S

**3.2.   Hardware Development**

3.2.1. Preliminary Design - Phase 1H

3.2.1.1. PC hub Trial

The goal of this trial is to be the hardware test that coincides with the WIndows Implementation Trial described in section 3.1.1.1. This test outlines how to use a PC computer as a hub for the Xbox Controllers to function with the Edge Robotic Arms. The hardware requirements of this test are as follows:

* Edge Robotic Arms with USB adapter boards
  + These boards do not come standard with the arms. They can be purchased from the manufacturer through amazon.com.
* A PC with a minimum of 2 USB ports
* 4 D batteries for the arms

For the system setup, the USB wired Xbox 360 controller and USB cable from the robotic arm both need to be plugged into the PC. Once this hardware setup is combined with Windows implementation Trial software setup, the Xbox 36 controler will be able to control the arm for proof of concept trial.

3.2.1.2. Initial Raspberry-Pi System Setup

The hardware connection and layout of how the raspberry-pi can be connected to the Edge Robotic Arm is shown in the assembly view diagram in appendix E.

3.2.2. Prototype Development - Phase 2H

3.2.3. System Verification - Phase 3H

3.2.4. Manufacturing and Assembly - Phase 4H

**3.3.   Instructional Scenario Development**

3.3.1. Preliminary Design - Phase 1G

In order to make the R.A.P.T.O.R. kits more effective, games to target the specific advantages of the kit need to be developed. Thanks in part to the use of a Raspberry-Pi as the control computer for the kits, multiplayer games become the go to instruction tool for the kits. Two different ways to run these games are present: each arm run off its own Raspberry-Pi or have multiple arms run off one Raspberry-Pi. The first option require no additional hardware setup. The second option however requires the creation of a hub for the controllers. The system setup diagrams for this configuration can be found in Appendix F. The disadvantage of the second option is that it requires the use of a wireless network, however.

3.3.2. Critical Design Review - Phase 2G

3.3.3. System Verification - Phase 2G

**3.4.   Final Design Review (FDR)**

**4. Risk Assessment**

**4.1.   Software Risk**

**4.2.   Hardware Risk**

**4.3.   Safety Review**

**5. Cost Analysis**

A detailed listing of the costs associated with the development of the R.A.P.T.O.R. kits can be found in Appendix D

**6. Conclusion**

Appendix A – Timeline Calendar

Appendix B – Software Risk Chart

Appendix C – Hardware Risk Chart

Appendix D – Cost Chart

Appendix E – Raspberry-Pi Hardware Outline

Appendix A

