Building a Four Monitor Media Show using Raspberry Pis

By Michael Groschadl and John Hilgart

As one of the 2,917 *FIRST* (www.usfirst.org) robotics teams worldwide, we were given the task of having four monitors running various photos and videos within a slideshow for our “pit” at competitions. The “pit” is a spot within the robotics competition arena where the team sets up and gets ready to compete. It is also a place to show other teams what we have accomplished in our build season that year.

Our first thought was to have photos on two monitors and videos on the other two; so, we did some research and found a few programs that would accomplish this. The software we found includes fbi, Qiv, pqiv and Omxplayer. This software is free and we thought this was great; after all our time doing research, we had finally found what we needed. When playing around with the software, we ran into some challenges. We found that fbi, Qiv and pqiv are geared more for photos, whereas Omxplayer is geared for videos.

You may ask, “Why is this a problem? Why does this not work? This is what is needed, right?” With our plan, yes, this was perfect. However, when given the materials for the “pit” slideshows we realized that it would be a lot better to use an application that is able to play both photos and videos at the same time.

We discovered another free software called Pi Presents, which was originally intended for use in museums. With this software, we found that we are able to play photos and videos at the same time without the need to change software. This was the perfect software to download on all four of our Raspberry Pis. In order to do so, we followed all the information listed in the document “README.md.”

This is how to download Pi Presents:

1. Open the LX Terminal

2. Type and command the Pi to do the following (wait for the previous command to finish before moving on to the next), and make sure you are doing so in your home directory:
    a. “sudo apt-get update”
    b. “sudo apt-get install python-imaging”
    c. “sudo apt-get install python-imaging-tk”
    d. “sudo apt-get install x11-xserver-utils”
    e. “sudo apt-get install unclutter”
    f. “wget http://pexpect.sourceforge.net/pexpect-2.3.tar.gz”
    g. “tar xzf pexpect-2.3.tar.gz”
    h. “cd pexpect-2.3”
    i. “sudo python ./setup.py install”
    j. “cd ~”
    k. “wget https://github.com/KenT2/pipresents/tarball/master | tar xz”
3. Pi Presents is now installed
4. Next, download the example files in the home directory with the command “wget https://github.com/KenT2/pipresents-examples/tarball/master | tar xz”
5. Pi Presents also requires the installation of xpdf. Do this in the LX Terminal by typing “sudo apt-get install xpdf”

Though the manual was difficult to understand, when working with Pi Presents we were able to learn about the program’s multiple potential usages. For the purpose of using just photos and videos within different slideshows for each Pi, we used the portion of the software called Mediashow. When using the LX Terminal in the Pi, the Mediashow is referred to as ‘pp_mediashow.’ Once all the examples are installed, along with Pi Presents, the Mediashow is able to be tested for correct installation. Once this is done, you can test the example by doing the following:

1. cd ./pipresents
2. python pipresents.py -p pp_mediashow

The cursor will go to the next line and will hold there for a few seconds, and then a screen that says, “Welcome To Pi Presents” should show up. This is the first part of the example Mediashow. It should then go on to a short clip from the show Suits, followed by a default photo of a lady. Another screen will then appear which says, “The Show Will Be Repeating Soon…” ; it should now resume to the first part of the Mediashow. It will keep repeating over and over until you exit the program. To exit, press ALT-F4 on the keyboard.

To create, edit, and test a new mediashow, in LX Terminal, make sure you are in the directory “/home/pi/pipresents/” and then do a directory listing by typing ‘ls’ and hit enter. You should see a python program called ‘pp_editor.’ This program allows you to edit any portion of Pi Presents and to create your own Mediashow, or whatever you wish to do, with this software. To launch the software, type “python pp_editor.py” and a new window should open. You have successfully launched the editor for Pi Presents.

To start making your own Mediashow, go to ‘Profile’ then highlight over “New Form Template” and click ‘Mediashow.’ You will then be prompted to create a filename. You have now completed the first step in the creation of your new Mediashow. The file is located in “/home/pi/pp_home/pp_profiles/your filename”; this is where the pp_editor and pipresents.py look for the file.

In the directory of your Mediashow, you will notice two files including ‘media.json’ and ‘pp_showlist.json,’ in the LX Terminal or GUI (Graphical User Interface...aka the Pi’s desktop). You do not need to do anything with these files; they are only worked on with the pp_editor.

Now that you are in your all new Mediashow, you are going to want to learn how to edit it with the ‘pp_editor.’ In the editor there are three boxes, “Shows,” “Medialists,” and “Tracks In Selected Medialist.” For our purposes, we never touched the “Shows” box and only clicked on ‘media.json’ in “Medialists.” By clicking on ‘media.json,’ it allows the user to edit his Mediashow within the “Tracks In Selected Medialist.” Since we are working with four Pis, we kept everything the same except for the contents of each ‘media.json.’

In your new Mediashow, the only thing in the “Tracks In Selected Medialist” is an example track. This can be deleted. Do this by selecting the example track and hitting ‘Track” followed by ‘Remove.’ You will notice that it is no longer there. We noticed that when there are multiple tracks that you would like to delete at once, that the ‘pp_editior’ is not capable to doing so. Instead, it acts as if it was not told a command and there will be an error message in the LX Terminal.

To add a file, or files, click the ‘Add’ button next to the “Tracks In Selected Medialist” box. Next, locate the files you want to show in your presentation. You can change the order of the files by selecting each one and hitting the ‘Up’ and ‘Down’ buttons. Just like removing files, you can not add more than one at a time.

You will notice there is not a ‘Save” button within any of the menus or on the boxes GUI interface. This is because the ‘pp_editor’ autosaves. Once all of your changes are made, you can simply click out of the editor.

To test your file, in the LX Terminal, make sure you are in the “/home/pi/pipresents/” directory. Then, as with the ‘pp_mediashow’ before, you do the same exact command; however, instead of ‘pp_mediashow,’ use the filename that you created for your personal Mediashow. It should start to run.

You will notice, unless working on an HDMI TV, that your Mediashow is probably not in fullscreen. To fix this, before the '-p' in the command line, type '-ftop.' This will change your output to fullscreen.

Challenges that we ran into with the Pi Presents Mediashow were with photos, pixel aspect ratios and videos, and the type of video files able to be used.

First, if your photo or video is too large for the screen, it will either not show the full picture or, as in the case of videos, it will likely not play. If you are running into these issues, the solutions are simple.

For photos, make two programs that coincide in order to change the pixel aspect ratio to your monitor's correct pixels. We created a shell script, 'sh' program, called "reduce.sh" and a python program called "resize3.py." The reasoning behind the 3, is that it took three versions before it was perfected. "Reduce.sh" uses "resize3.py" to run.

To make a new file in your home directory ("/home/pi/"), type "nano" and what you would like to call the file. In this case, "nano reduce.sh", and hit enter. Now you can start writing the code. The coding for "reduce.sh" is:

```
#!/bin/sh
echo "Look for the downsampled images in a sub-directory called resize
echo "JPEGs GIFs and PNGs are looked at in the current directory mkdir resize
2>/dev/null
ls -1 *jpg *jpeg *JPG *png *PNG *gif *GIF 2>/dev/null|while read file; do
   echo downsampling $file
# downsample the image file
   python ~/resize3.py "$file"
done
```

Exit out of the nano editor by pressing CTRL+X. You will be asked if you are sure you want to save and you would type "y",or "yes", depending on what is listed as the proper response prompted on the screen. Then, type the filename and exit.

As with the coding of "Reduce.sh," you will notice that "resize3.py" will need to be created. You do this the same way as with "Reduce.sh." Make sure you are in the home directory ("/home/pi/") and type "nano resize3.py" The coding for "resize3.py" is:

```
import Image
import sys
# DrJ 2.2015
# somewhat inspired by http://www.riisen.dk/dop/pil.html
# image file should be provided as argument
```

```
    # Designed for Acer v173 display which the Pi sees as a strange 1232 x 992
pixel display
    # though it really is 1 more run-of-the-mill 1280 x 1024

    imageFile = sys.argv[1]
    im1 = Image.open(imageFile)

    def imgResize(im):
    # Our display as seen by the Pi is a strange 1232 x 992 pixels
       width = im.size[0]
       height = im.size[1]

    # If the aspect ratio is wider than the display screen's aspect ratio,
    # constrain the width to the display's full width
       if width/float(height) > 1232.0/992.0:
          widthn = 1232
          heightn = int(height*1232.0/width)
      else:
          heightn = 992
          widthn = int(width*992.0/height)

      im5 = im.resize((widthn, heightn), Image.ANTIALIAS)  # best down-sizing filter

      im5.save("resize/" + imageFile)

    imgResize(im1)
```

Save as described before and check to make sure that both files are saved into the more directory by doing a directory listing. This can be done by typing 'ls' in the LX Terminal in the home directory.

As with any computer and computer program, it is never recommended to just unplug the Raspberry Pi at the end of the day. To stop the program from running, we created a file called "kill.sh" in the home directory. Make sure you are in the home directory and type "nano kill.sh" and the nano editor should open. The coding for "kill.sh" is:

```
    #!/bin/sh
    pkill -f pipresents.py
    pkill omxplayer
```

You are now able to save as you did with “Reduce.sh” and “resize3.py.”

You may be wondering why we are using coding to stop Omxplayer. You are probably thinking since we decided not to use omxplayer for videos that it was not used; however, Pi Presents actually does use Omxplayer.

As with any computer, the Raspberry Pi will eventually go into screensaver or standby mode. Since there are a couple hours of “pit” time, and the screensaver turns on in minutes, we needed to figure out how to disable the blanking of the screen. This is accomplished by doing the following:

1. Change to the directory “/home/pi/etc/kbd/” in the LX Terminal and “sudo nano config”
2. Scroll to where it says “BLANK_TIME=30” and change the thirty to zero
3. Scroll to where it says “POWERDOWN_TIME=30” and change the thirty to zero
4. Save this file the same way you have been saving all the other files
5. Now, go up a directory and make sure you are in “/home/pi/etc/” in the LX Terminal and then go to “/home/pi/etc/lightdm”
6. Then, “sudo nano lightdm.conf”
7. Scroll to where it says “[SeatDefault] and add a line near the end saying ‘xserver-command=X -s 0 dpms’

Your Raspberry Pi should never again go into screensaver/standby mode.

We managed to get the Pi Presents to auto start upon boot up of the Raspberry Pi. This was done to avoid the need of a keyboard and mouse in the “pit” during competitions. To auto start, we did the following:

1. Type into the LX Terminal, “$ cd /etc/xdg/autostart”
2. Make a new file in this directory by typing ‘nano’ and then typing the name of the new file. The new file name is “pipresents.desktop.” All together, it should look like “nano pipresents.desktop”
3. In the “pipresents.desktop” file, type the following coding:

```
[Desktop Entry]
Type=Application
Name=pipresents
Exec=python pipresents/pipresents.py -ftop -p YourMediaShow
Terminal=true
```

4. You can put this coding into the ".config" directory as well. This is necessary because not all Raspberry Pis support the way just shown. You can insure that this will work by 'cd ~' to go to the home directory and then type "$ mkdir ~/.config/autostart"

5. Then, create the same file as above

Now, we want the Mediashow to autostart. Doing the following will make this happen:

1. In the LX Terminal, type "$ mkdir -p ~/.config/lxsession/LXDE"
2. Then, "$ cd !$; echo "python pipresents/pipresents.py -ftop -p YourMediaShow"> autostart
3. After that, "$ chmod +x autostart"

This should enable the Pi Presents Mediashow to startup upon booting the Raspberry Pi.

So that we do not need to attach a keyboard or mouse to the Pi, we find it convenient to run "pp_editor" in a virtual display created by VNC. The problem is that Pi Presents will also launch in the VNC session and really slow things down. The following is a solution we worked out in order to have only one instance of Pi Presents run, even in the event that xsessions are launched on other displays. Note that this is a general solution and applies to any autostarted program.

You want to put a file called "startpipresents.sh" into your home directory. Type 'nano startpipresents.sh' and then type the following coding into the file:

```
#!/bin/bash
# DISPLAY environment variable in :0.0 for the console display
echo $DISPLAY|grep :0 > /dev/null 2>&1
if [  "$?" == "0"  ]; then
#  matched. start pipresents in this xsession, but not any other one
   python pipresents/pipresents.py -ftop YourMediaShow
fi
```

In doing this, the "pipresents.desktop" file becomes:

```
[Desktop Entry]
Type=application
Name=pipresents
Exec=/home/pi/startpipresents.sh
Terminal=true
```

In order for this to work, you are going to want to install the VNC server application onto the Pi. Do this by typing “sudo apt-get install tightvncserver” into the LX Terminal. (You need to download VNC Viewer onto your PC or Mac to be able to view the VNC Server session.)

Since there will not be a keyboard, you will not be able to type into the LX Terminal or the Raspberry Pi and prompt the VNC Server to turn on. We needed to make it so that the VNC Server autostarts, just like Pi Presents. This is done by:

1. ‘Cd’ into “~/.config/autostart”
2. ‘Nano’ a file called “vnc.desktop”
3. Type in this coding:

```
[Desktop Entry]
Type=application
Name=vncserver
Exec=/home/pi/startvncserver.sh
Terminal=false
```

4. You now want to make a file in the home directory called “startvncserver.sh.” The coding for this file is:

```
#!/bin/bash
# DISPLAY environment variable is :0.0 for the console display
echo $DISPLAY|grep :0 > /dev/null 2>&1
if [  “$?” == “0”  ]; then
# matched. start vncserver in this xsession, but not any other one
    vncserver
fi
```

5. VNC will always autostart when the Pi is booted up

Keep in mind, VNC always requires a password when being logged into from a separate source. Set the password to whatever you want. To set the password, simply launch a VNC Server from the LX Terminal. This is done by typing “vncserver” into the LX Terminal. The first time you start up VNC Server, it will ask for a password. The server is then launched and can be connected from your computer.

Just in case the photos or videos need to be edited or deleted on the spot, we decided to auto launch the “pp_editor” within the VNC Server. To do this, make a file in the Pi’s home directory called “startppeditor.sh.” The coding for this ‘sh’ file is as follows:

```
#!/bin/bash
# DISPLAY environment variable is :1.0 for the vnc display
echo $DISPLAY|grep :1 > /dev/null 2>&1
if [  “$?” == “0”  ]; then
#  matched. start ppeditor in this xsession, but not any other one
   python pipresents/pp_editor.py
fi
```

And in the “~/.config/autostart” directory, make a file called “ppeditor.desktop” and type in this coding:

```
[Desktop Entry]
Type=application
Name=ppeditor
Exec=/home/pi/startppeditor.sh
Terminal=true
```

Now the “pp_editor” will open within the VNC Server window on your computer uplink.

Even though Mac computers come with a terminal application and you are able to download software called PuTTY for PC, in case we wanted to use the LX Terminal within the VNC Server, we made that autostart as well. This was done by making a file in the home directory called “startlxterminal.sh”. The coding for this is:

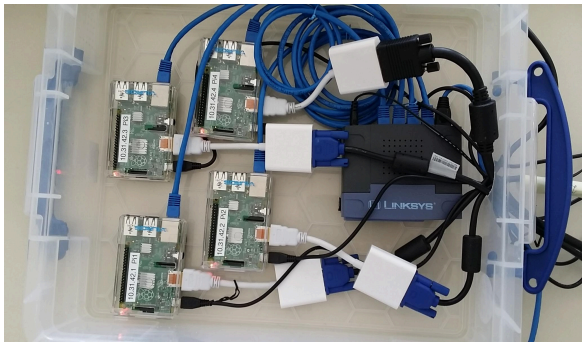
```
#!/bin/bash
# DISPLAY environment variable is :1.0 for the vnc display
echo $DISPLAY|grep :1 > /dev/null 2>&1
if [  “$?” == “0”  ]; then
#  matched. start a large lxterminal in this xsession, but not any other one
  lxterminal --geometry=100x40
fi
```

The file for the autostart of the LX Terminal goes in the same directory as the “ppeditor.desktop” file. This file is now called “lxterminal.desktop.” The coding for autostarting the LX Terminal is:

[Desktop Entry]
Type=Application
Name=lxterminal
Exec=/home/pi/startlxterminal.sh
Terminal=true

When the Raspberry Pi boots up, the “pp_editor” and LX Terminal will autostart in the VNC Server .

Once the Raspberry Pis were all programmed, we needed to connect all four Pis to the monitors and make the Pis easily accessible. This was done in order to allow changes to the contents of the Mediashow as well as fixing issues in the event that something goes wrong and the Pi gets messed up. We got a 5-Port Hub and connected it to each of the four Pis, along with a fifth ethernet cable, to be able to be connected to a computer at any needed time. Since the platform of the monitors are standing upright, we secured the Raspberry Pis to a plastic carrying box with velcro. (Put holes in the side of the box so the Pis can vent as they produce heat, just like any other computer, and they can potentially overheat). We attached the box to the back of the monitors with zip ties. In order to accomplish this, holes were added to the corners of the outside of the box .To avoid wire clutter, we drilled a big hole into one side of the box and have all the VGA, power, and ethernet cables going through that hole. This also enables the use of a lid for the box.



Something to note is that most videos, especially “homemade” videos, have problems playing in Pi Presents. A video directly from a camera, or edited videos in various video editing software, is not supported. Most file extensions (.mov, .avi, .mp4, etc.) are supported, but even if they come out of a camera that way, they still need to be fixed. To fix this problem, we imported the video into Windows Movie Maker and exported the video into a file that is able to be played on an Android phone. This fixed the problems with non-functional videos for the Raspberry Pis. We used the Android phone setting because the Raspberry Pi operating system, Raspbian, and the Android OS are both Linux based.

With some of the videos displayed on the Raspberry Pis, we wanted to be able to hear audio. This would not be a problem with HDMI, right? Well, not necessiarily. Since we are using monitors without speakers, the HDMI cable is not able to transmit the audio of the video. There, thankfully, is a 3.5mm audio jack on the side of the Raspberry Pi that we could plug speakers into. When we did this, still no sound was produced. In

the ‘pp_editor’ we needed to change one setting to enable the 3.5mm jack. In the ‘pp_editor’ go into your mediashow. In the ‘Shows’ box, click on “Mediashow [mymediashow].” Then, click on the edit button to the right of that box. A popup box should appear with a bunch of changeable settings. Find where it says ‘OMX Audio’ and change the setting from ‘hdmi’ to ‘local.’ Now, the sound will be playing out of the 3.5mm audio jack and not the HDMI cable.

Now that we have everything set up with the Raspberry Pi, we need to setup an IP address so that we could access the Pis via VNC and PuTTY. Do this by doing the following:

1. Open LX Terminal and make sure you are in the home directory
2. ‘Cd’ into “/etc/network”
3. Then “sudo nano interfaces”
4. Type:

```
auto lo

iface lo inet loopback
#iface eth0 inet dhcp
iface eth0 inet static
address 10.31.42.1
netmask 255.255.255.0
network 10.31.42.0
broadcast 10.31.42.255

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

5. ‘CTRL-X’ and save the file
6. Keep in mind, if you are using multiple Pis, you need to change the last digit of the “address 10.31.42.1” so that each Pi is identifyable as different computers. Example: “address 10.31.42.2”

Materials used with this process include:

1. Four Raspberry Pi B+ Micro Computers (Complete Starter Kit by Vilros) ($59.95 each, $299.75 total)
2. Four PiView HDMI to VGA Converter Adapters ($45.99 each, $183.96 total)

3. Five-Port Ethernet Hub ($32.99)
4. Five Ethernet Cables ($9.99 for pack of five)
5. One Laptop with VNC Viewer and (On Windows) PuTTY installed, Mac OSX has Terminal by default instead of needing to download the software PuTTY
6. Four Monitors (Acer v173) ($159.99 each, $639.96 total)
7. Nine Electrical Outlets
8. Velcro and plastic box to house the Pis
9. A drill to make holes in the box and zip ties to mount the box and chords

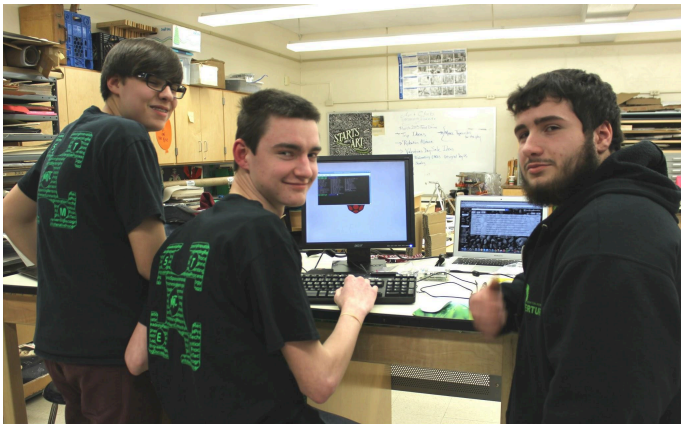
Further Information:

To learn more about our school's robotics team, known as FRC #3142 Team Aperture, visit www.newtonroboticsteam.org

Starting on page 13 of this magizine, one of our articles is published, http://www.omagdigital.com/publication/?i=198323. We have also been in local newspapers.

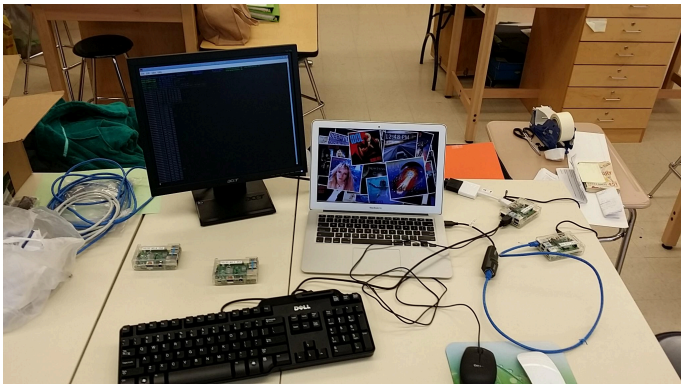
Michael with the Raspberry Pis running in the 'pit'

John and Michael with two representatives from our high tech partner, ThorLabs, www.Thorlabs.com, who wished to see the Raspberry Pi work



Michael and two other members of the robotics programming team (Cameron Osborn, left, and Brian Hoskins, right) working on the Pis

Michael talking about the Pi work at the robotics team STEM Night 2015



Downsampling photos with ‘Reduce.sh’ for the photo MediaShow