PiBox - Action Item #471
Review Pi monitor for use with Reba's memorial videos/photos.

01 Nov 2015 15:04 - Hammel

Status:	Closed	**Start date:**	01 Nov 2015
Priority:	Immediate	**Due date:**	
Assignee:	Hammel	**% Done:**	100%
Category:	06 - Hardware	**Estimated time:**	0.00 hour
Target version:	0.13.0		
Severity:	01 - Critical		

Description

This is the official monitor, running off DSI port on Pi.

http://swag.raspberrypi.org/products/raspberry-pi-7-inch-touchscreen-display

Associated revisions

Revision 0cef04b0 - 23 Feb 2017 20:25 - Hammel

RM #471: Add generalized support for touchscreens.

Revision f63b86e4 - 23 Feb 2017 20:49 - Hammel

RM #471: Fix touch lib header.

Revision 638d65e3 - 23 Feb 2017 21:46 - Hammel

RM #471: Add loadConfig to touchProcessor so it can properly set display resolution.

Revision 4d8e5b17 - 24 Feb 2017 22:48 - Hammel

RM #471: Extend ability to send either a region or absolute coordinates to a registered callback.

Revision 0159a38b - 25 Feb 2017 14:41 - Hammel

RM #471, RM #567: Migrated touchscreen support to libpibox. Updated libpibox API usage to remove pnc prefixes.

Revision 689a5897 - 26 Feb 2017 17:40 - Hammel

RM #471: Fix up player to work on HDMI as well as LCD after adding support for touchscreens.

Revision 3c4842d6 - 27 Feb 2017 22:25 - Hammel

RM #471: Add overlays directory to installation to SD card.

Revision 88dbba2d - 27 Feb 2017 22:27 - Hammel

RM #471: Generate a display config file at boot time.

Revision 835fe716 - 27 Feb 2017 22:27 - Hammel

RM #471: If on RPi touchscreen, extend config.txt on firstboot to support it.

Revision a9df4675 - 27 Feb 2017 22:29 - Hammel

RM #471: Fix X startup to handle either HDMI or RPi touchscreen (LCD).

Revision 2747c23a - 27 Feb 2017 22:32 - Hammel

RM #471: Add RPi touch specific xorg.conf.

Revision fb5aa34f - 28 Feb 2017 20:11 - Hammel

RM #471: Skip local and parent dirents while looking for an event file for the touchscreen. Fails ts_setup() if no matching touchscreen is found.

Revision aa5673da - 28 Feb 2017 20:31 - Hammel

RM #471: Remove xterm launcher from default player command for PiBox Media Center.

Revision 5b9c9f4a - 02 Mar 2017 22:02 - Hammel

RM #471: Added kiosk mode to videofe to play a series of videos sequentially, using the touchscreen to move to next/prev videos or ff/rewind.

Revision a6cc155c - 05 Mar 2017 09:04 - Hammel

RM #471: Return absolute coordinates in REGION_T structure instead of the 9x9 cell location.

Revision 490d084f - 05 Mar 2017 20:15 - Hammel

RM #471: Fix submission of absolute coordinates even if we don't match a range properly.

Revision c71a4a45 - 05 Mar 2017 20:41 - Hammel

RM #471: Add touchscreen support to launcher.

Revision fe807b2e - 05 Mar 2017 20:42 - Hammel

RM #471: Send SIGUSR1 to launcher when apps exit.

Revision 7be3c7f6 - 07 Mar 2017 22:46 - Hammel

RM #471: Make icons scaled to fit screens. Change rows/cols counts based on the number of apps found by the launcher.

Revision 5e23c83f - 08 Mar 2017 22:02 - Hammel

RM #471: Fix scaling keeping aspect ratio to use gfloats instead of ints.

Revision a3ac82d8 - 11 Mar 2017 17:59 - Hammel

RM #471: Add black top level window in kiosk mode to avoid ugly flashing to launcher on touchscreen display.

Revision fcd4208c - 11 Mar 2017 18:02 - Hammel

RM #471: Add touchscreen drivers if on an LCD screen.

Revision 6cf621ae - 11 Mar 2017 18:03 - Hammel

RM #471: Overscan really isn't needed on HDMI screens, so I'm disabling it in config.txt.

Revision f6d6ba93 - 12 Mar 2017 12:18 - Hammel

RM #471: Resize icon to be larger so scaling will always be down, not up.

Revision 2a9b468b - 12 Mar 2017 12:36 - Hammel

RM #471: Resize icon to be larger so scaling will always be down, not up.

Revision 5ddc764e - 12 Mar 2017 12:38 - Hammel

RM #471: Resize icon to be larger so scaling will always be down, not up.

Revision 5ddc764e - 12 Mar 2017 12:38 - Hammel

RM #471: Resize icon to be larger so scaling will always be down, not up.

Revision 6c8a5c3f - 12 Mar 2017 12:42 - Hammel

RM #471: Resize icon to be larger so scaling will always be down, not up.

Revision 1b826c99 - 13 Mar 2017 22:54 - Hammel

RM #471: Disable key and button press event handlers if on a touchscreen.
This prevents a button press from being processed before the touch event is handled.

Revision f55ab3fc - 25 Mar 2017 16:51 - Hammel

RM #471: More cleanup for a clean firstboot with a Raspberry Pi 7" display.

Revision 33ac179b - 25 Mar 2017 16:52 - Hammel

RM #471: Mount mmc as RW so we can auto-update config.txt on firstboot.

Revision ea42a4c5 - 25 Mar 2017 16:54 - Hammel

RM #471: Change initial xterm under blackbox to be fullscreen so it always fits on whatever display we're using.

Revision bb6bfb93 - 25 Mar 2017 20:10 - Hammel

RM #471: Change /dev/input files to 644 so non-root apps can read the touchscreen devices.

History

#1 - 08 Feb 2017 22:32 - Hammel

- Status changed from New to In Progress
- Priority changed from High to Immediate
- % Done changed from 0 to 10
- Severity changed from 02 - High to 01 - Critical

This became important because of our dog, Reba, passing away. I want to use PiBox to display photos and videos of her.
How to enable the drivers:

- https://github.com/raspberrypi/linux/issues/1147
- https://www.raspberrypi.org/documentation/hardware/display/troubleshooting.md
- http://forum.tinycorelinux.net/index.php?topic=18911.0
- https://www.raspberrypi.org/forums/viewtopic.php?f=108&t=149485

It looks like this is enabled in PiBox 0.11. Technically it should just work now. I need to test with an RPI 3. Currently building a PiBox 0.11 SD for testing.

More general touch screen help:
https://github.com/notro/fbtft/wiki/Touchpanel

#2 - 10 Feb 2017 20:38 - Hammel

I got it working! There are only a few things I needed to do:

Enable the ft5406 driver in the kernel and firmware. The driver is enabled with this (which is part of the defconfig for RPI):

```
CONFIG_INPUT_TOUCHSCREEN=y
```

```
CONFIG_TOUCHSCREEN_RPI_FT5406=m
```

With these we need to load the following two drivers at boot time.

```
rpi-ft5406
rpi_backlight
```

The firmware was added with the following lines to the config.txt:

```
dtoverlay=rpi-ft5406-overlay
lcd_rotate=2
```

The first is required for the firmware to find the overlay image file, which needed to be copied into *overlays/rpi-ft5406.dtbo*. That file was part of the stock firmware build I'm doing with PiBox so it was already available.

The second line makes the screen flip 180 degrees. This is necessary because the power plug is on the bottom by default, making it hard to stand the display (the cable gets in the way). Flipping the display puts the cable out the top.

I'm not sure if these are needed, but I added them just in case:

```
framebuffer_width=800
framebuffer_height=480
fbwidth=800
fbheight=480
```

Then I had to set xorg.conf to have the smaller screen size:

```
Section "Screen"
    Identifier      "Builtin Default fbdev Screen 0"
    Device          "Builtin Default fbdev Device 0"
    Monitor         "Builtin Default Monitor"
    SubSection "Display"
      Depth 24      # 24bit works fine but for USB 2.0 a lot of data
      Modes "800x480"
    EndSubSection
EndSection

Section "InputClass"
  Identifier "calibration"
  Driver "evdev"
  MatchProduct "FT5406 memory based driver"

  Option "EmulateThirdButton" "1"
  Option "EmulateThirdButtonTimeout" "750"
  Option "EmulateThirdButtonMoveThreshold" "30"
```

EndSection

The screen section limits the X.org screen to the size of the display (which is 800x480). The input class sets up the touch screen so events get passed into applications, like the GTK+ launcher.

With these changes I was able to boot into the PiBox Media Center display. It just barely fit (the splash text was cut off) but it worked.
So now I need to replace the launcher with a custom tool that

1. Provides a simple touch screen interface for starting a feature
2. Plays videos from a directory
3. Plays slideshow of images in a directory
4. Stops displays and shows a single image

Programming with the touchscreen can be done using the input system using ioctls. But that's not for X.org. For that, we use the evdev driver and then use GTK+ event handling to get X/Y coordinates on button presses. This is actually already designed into my GTK+ apps, like in the launcher.

This means the front end (launcher replacement) would offer a selection of playing videos or image slideshows as separate apps. Then the apps would exit on a button press. The image slideshow would be easy to handle like that. The video playback has to map a button press to the ESC key for omxplayer, just like the PiBox Media Center.

#3 - 10 Feb 2017 23:32 - Hammel

- Subject changed from Review Pi monitor to Review Pi monitor for use with Reba's memorial videos/photos.
- % Done changed from 10 to 20

It may be possible to set the keyboard config mappings to use a button press (key code 330) to be the quit key. That would allow me to use a modified VFE to play videos (without selecting them) and be able to exit the currently playing video (which would be managed by omxplayer) by tapping the screen. This would also completely exist the modified VFE and take me back to the launcher.

#4 - 11 Feb 2017 17:56 - Hammel

If the omxplayer keyboard config bindings don't work I might try using xbindkeys to map "b:1" (or :b:2 or b:3) to "killall omxplayer". That assumes that the button press while running omxplayer is passed through the X server. It's hard to tell who is getting the button press when omxplayer is running. All I can do is test it.

#5 - 11 Feb 2017 22:23 - Hammel

The omxplayer key config file didn't work and neither did setting up an XResource file for the xterm wrapping the call to omxplayer. In fact, even just doing ESC in omxplayer without these didn't work either (as it does in use with a standard monitor).

So now I think I'm down to monitoring the input device (/dev/input/eventX) manually. I can do this with tslib or I can do it manually (or here). Either will have to be done in VFE as part of the setup for running omxplayer. When an event is caught it needs to be translated into an omxplayer command. These can be piped to omxplayer through a fifo (or here). This has the advantage of being able to map different event types (such as dragging across the screen) to different commands (like skipping forward and back and changing the volume).

#6 - 12 Feb 2017 15:09 - Hammel

I got it working today. I used a new thread in videofe that runs a nonblocking tslib query on /dev/input/event3 (hardcoded for now). When an event comes in (any event from that device, which is the touchscreen in this particular setup) I kill omxplayer with a call to *system("killall omxplayer")*. Works perfectly. I can now exit omxplayer with a single tap to the screen.

Now I need a way to identify which input device to use. This has to be done by name. I'm looking for a device with a name of *FT5406 memory based driver* which can be queried by opening each eventX device and running the following ioctl to get it's name.

```
ioctl (fd, EVIOCGNAME (sizeof (name)), name);
```

See linux-input-codes.h and input.h for details on using the ioctl.

I would still want to map the events captured to commands piped to omxplayer. One simple way to do this is divide the display into 6 sections. Tapping in the top three would do stop/ff/vol up while the bottom three might do pause/rewind/vol down. That's pretty simple to do given the simplistic data returned from tslib. If I want something more complex I'll have to drop tslib in favor of getting input directly from device reads using the kernel input system structures.

So far all my changes are just hacks to get it working on the touchscreen. Later I'll clean them up, such as making them available only if the device is configured with a touchscreen, though I'm not sure that the moment how to know the touchscreen is being used and not the HDMI.

#7 - 13 Feb 2017 20:51 - Hammel

I can get the touchscreen (or HDMI) display size with this:

```
/opt/vc/bin/tvservice -s | cut -f4 -d" "
```

That should be done at startup by appmgr and all apps should be able to query a configuration file to know how to scale their displays.

The output for this command when run with the touchscreen display is:

```
800x480
```

If I don't trim the resolution out it looks like this:

```
state 0x400000 [LCD], 800x480 @ 60.00Hz, progressive
```

For HDMI displays the output is a little different:

```
state 0x120006 [DVI DMT (39) RGB full 16:9], 1360x768 @ 60.00Hz, progressive
```

So the real cut is the comma:

```
/opt/vc/bin/tvservice -s | cut -f2 -d"," | cut -f2 -d" "
```

That should pull the resolution out for either HDMI or Touchscreen.

#8 - 20 Feb 2017 12:04 - Hammel

- % Done changed from 20 to 40

I know have videofe working with the touch screen and properly recognizing finger presses in 9 regions on the screen. I've mapped these to various functions (which I'll document later, and already have in the code). I've also verified that the changes do not affect the way the app works on an HDMI display. At least not the one I'm using for testing.
The next step is to change the way omxplayer is started. I need to

1. create a fifo
2. Add a stdin redirection to omxplayer's command line
3. Setup functions to write to the fifo in the new touchProcessor.c module.

That will pretty much complete this support. Then I can just add support for playing videos from a stock location if there is no database found. This is what I want for playing videos of Reba in memoriam. But I want it to play them in a loop, continuously, unless stopped. I also need a "next video" function, which can map to one of the 9 regions (I still have some that are not yet mapped to anything).

And then I need to add a photo player to videofe. Or (better) a photo player app. Then I'll have just two apps in the PiBox Memoriam distribution: photos and videos, both of which play from a single USB stick.

#9 - 03 Mar 2017 21:17 - Hammel

This has become a catch all for the in-memoriam project. I've finished adding touchscreen support to libpibox, added touchscreen support to videofe, and added a kiosk mode to videofe so it will play videos in sequence without a UI. This works quite well. I've regression tested against the original use-case of TV/desktop monitor displays and that still works. Interestingly, the SMB automount was running during testing and it picked up another Pi I had running with PiBox. The kiosk mode happily played the movies from that Pi too, though that isn't the intended use-case for In-Memoriam.

The next step is an image viewer app. This is just a variation of videofe, using feh as the image viewer. Feh is already in the core distro and it's easy to have it cycle through the images in a directory or via a filelist. To use it:

> *feh -r -F -D 2 --zoom fill path/*

or

> *feh -r -F -D 2 --zoom fill -f filename*

I just copy videofe to the new app, strip the video player component and replace it with an image viewer component. Pretty straight forward stuff. On, and feh support SIGUSR1 and SIGUSR2 to moving to the next/prev, respectively, image. That means the touchscreen support in this app can control the playback of the images. At least with respect to skipping forward and back.

After the image viewer comes touchscreen support for the launcher so the user can touch the app they want. This only requires converting widget coordinates to absolute coordinates. The libpibox touchscreen support provides absolute coordinates instead of regions, if desired. I just have to find out how to convert GTK+ widget x/y to screen coordinates, which should by easy I think.

Then it's just a matter of making an In-Memoriam distro with just the two apps. The user just plugs in a USB stick with images and videos and off they go.

This is nearly done. All for you, my little Reba.

#10 - 08 Mar 2017 15:51 - Hammel

- % Done changed from 40 to 90

The use of feh was dropped because it didn't handle signals very well and crashed regularly. A little googling uncovered the use of gdk_pixbuf with cairo to read various image files and display them scaled to fit the screen. This turned out to be very easy to implement. So pipics has been transformed into a video slideshow app without external dependencies.

After that I implemented touchscreen support in the launcher. This was about the same as the slideshow viewer or videofe except each touch needed to be handled as an absolute position. I'd already added that to libpibox's touchscreen support. And it pretty much worked the first time. What needed to be cleaned up was the scaling and positioning of the icons. I was able to switch to using gdb_pixbuf in the launcher, allowing icons of different file formats while also allowing me to scale the icons while keeping their aspect ratio. I've even added a little padding around each icon. So now the icons display properly under the 7" touchscreen or a full sized monitor, scaled to fit.

All changes are committed and pushed. Now it's time to make the In-Memorium distro. And order a new touchscreen + Pi2 (or 3, which is what I tested with) for it. Oh, and I need to gather Reba's photos and any additional videos my wife has. And build a little wall shelf for it.

But after making the distro, this task is essentially done.

#11 - 25 Mar 2017 17:20 - Hammel

Trying to get the distro working but I'm having problems. I fixed firstboot so it's cleaner. But now I have these problems.

1. The rpi modules are not being loaded. They get added to /etc/modules.conf but aren't loaded automatically at boot time.
2. After loading the modules PiPics just displays gray gradients where there should be a picture.

#12 - 25 Mar 2017 20:12 - Hammel

Drivers are loaded. Problem is that the device files in /dev/input are not readable except by privileged programs. So I need to fix the mdev.conf file for them.

#13 - 25 Mar 2017 20:33 - Hammel

The gradient file was an image found under /media/mmc* which was searched because the dbtop was set to /media/ instead of /media/usb.

The solution is probably to just leave /media as the dbtop and change the code to ignore /media/mmc* directories.

#14 - 25 Mar 2017 21:52 - Hammel

mdev.conf updated. PiPics default dbtop set to /media/usb.

All changes committed.

Now I'll make a disk image and push it to the release repo.

#15 - 29 Mar 2017 09:32 - Hammel

- Status changed from In Progress to Closed

- % Done changed from 90 to 100

I decided not to make a release. It's a manual process to make the media center into the digital photo frame so the only release I can make is a disk

image. That would be huge. Instead, I'll just note here how to do it.

1. Install the core system
2. Add the following packages: appmgr_0.10_arm.opk launcher_0.10_arm.opk omxplayer_0.10_arm.opk piboxd_0.10_arm.opk piboxmediaserverui_0.10_arm.opk pipics_0.10_arm.opk psplash_0.10_arm.opk videofe_0.10_arm.opk
3. On target
    1. mkdir -p /etc/launcher/.noshow
    2. cd /etc/launcher
    3. mv netconfig.xml restart.xml terminal.xml /etc/launcher/.noshow/

That's it. This could be automated. Maybe someday I'll do that.

Closing issue.