RemotePlus & RemotesServer for AXIOM Beta Cameras

Developers / Users Manual

Plus24 Systems Ltd.

Table of Contents

Introduction

This application, and when released the associated hardware, is designed to be a comprehensive remote control for the AXIOM range of cameras - currently this is targeted at the Beta model.

The remote control software is made up of two separate components – the RemotePlus application which is designed to run on a specially designed embedded (Raspberry Pi based) Linux hardware controller, but is also runnable on a Linux desktop/laptop to aid in development.

To control the AXIOM camera a RemoteServer application is also required. Connectivity is over the USB UART interface between the AXIOM camera and the host – be it the hardware remote or a Linux desktop/laptop.

This initial guide is intended for early access developers and users to get themselves up and running with the system.

As with all software & hardware that is pre-production the information contained in this document is subject to changes.

Software Prerequisites

The current desktop/laptop development environment is a 64bit Linux desktop (Ubuntu used for development) and utilities the following frameworks.

`wxWindows` `(>v.3.1)`
`wiringPi` `(>2.32)`

For building on the hardware device the same framework requirements must be met – at this time there are no instructions for building a cross-compiler tool-chain, but this will be looked at in the near future.

All of the dependencies need to be met for the above and details can be found on their respective websites. Additionally the git toolchain is recommended.

While wiringPi is a requirement for building the application for use with the hardware interface, it’s not a requirement for desktop/laptop use as it will not be possible to use the GPIO subsystem on a regular machine. The use of wiringPi is set as a #define flag, detailed in the Developer Notes section of this document.

AXIOM Beta Configuration Prerequisites

In order for the RemoteServer application to function the AXIOM Beta Camera will need to disable the default boot tty process, the pre-requisite to this is enabling ethernet access so the camera is still accessable.

As you will be altering the runstate and connection methods of the camera you *really should* make a full backup of the cameras SD card first.

I normally use the dd command, so as an example (with the Beta's sd card on /dev/sdb) I would use the following on my Linux workstation. It would be wise to check if you have space first though, my camera has a 16GB card so I require 16GB of space to do this.

```
sudo dd if=/dev/sdb of=AXIOMBetaBackup.img bs=512
```

Restoring is the reverse:

```
sudo dd if=AXIOMBetaBackup.img of=/dev/sdb bs=512
```

Once the backup has been made you can then start the Beta's configuration.

Follow the details in the AXIOM Beta Wiki until you have configured the UART interface and a basic network connection. Ensure that you can connect to the Beta by both methods.

https://wiki.apertus.org/index.php/AXIOM_Beta/AXIOM_Beta_Software

The next step is to make the network setting permanent, as the basic ifconfig will not survivie a reboot.

By default the Devs have set up the AXIOM Beta to pick up an IP address from a DHCP server, normally on your home network your broadband router will provide this.

An important point is if the nextwork environment you will be using the camera in does not have a DHCP server, or you want to use a static IP address, you will need to configure a static IP fallback address.

The Arch Linux Wiki has a number of pages on setting the network settings, go though the links below and carefully follow the details.

https://wiki.archlinux.org/index.php/Network_configuration https://wiki.archlinux.org/index.php/dhcpcd

When following this process I found that the IP address was not being set after a reboot, this was cured by running the following command.

```
[root@beta ~]# dhcpcd eth0
```

Then reboot the camera a few times and test if logging on works using ssh each time, test if you can do so via it's DHCP and Static IP's – you may have to run nmap or check your DHCP server to get the dynamic IP address of the camera.

Do Not Proceed if the above does not work reliably!

The final steps are to disable the boot getty process which is spawned when the kernel boots, which can be seen with the following ps command.

```
[root@beta ~]# ps -ef | grep agetty
root      1513     1  0 20:44 ttyPS0   00:00:00 /sbin/agetty --keep-baud 115200 38400 9600 ttyPS0 vt220
root      1843  1817  0 20:50 pts/1    00:00:00 grep --color agetty
```

First, run the following command.

```
[root@beta ~]# systemctl disable serial-getty@PS0.service
```

Next go into the /boot/ directory and make a backup of the devicetree.dts as well as the uEnv.txt, then edit devicetree.dts like so – removing the text marked in red and insert a comment as marked in green.

```
    chosen {                                                /* defaults */
        bootargs = "console=ttyPS0,115200 root=/dev/ram rw earlyprintk";
        #linux,stdout-path = "/axi@0/serial@e0001000";
    };
```

Then edit uEnv.txt removing the text marked in red.

```
baudrate=115200
bitstream_image=system.bit.bin
boot_image=BOOT.bin
boot_size=0xF00000
bootargs=console=ttyPS0,115200n8 root=/dev/mmcblk0p2 rw rootfstype=ext4
rootdelay=2 debug verbose systemd.log_level=warning
systemd.log_target=console kernel.sysrq=1 init=/usr/lib/systemd/systemd
```

Then run the following commands:

```
[root@beta ~]# dtc -I dts -O dtb -o /boot/devicetree.dtb /boot/devicetree.dts
[root@beta ~]# sync
```

And then reboot the Beta.

Keep the UART console open, or re-open one, as you boot to check if there are no console messages being displayed, then run the ps command from above to check that there isn't an agetty process started now.

There are also a number of helper scripts that need to be placed on the AXIOM camera, these are helper scripts for the RemoteServer, and should be located in the /root/ directory.

They can be found in the GitHub repository.

`zynq_info.sh`

Make a backup of the original file before copying over this new version.

This checks if the script has been called with a flag, then checks if it's -s (for Silent – just print out the temp value). If not it prints it out in it's standard format.

```
set_exposure.sh
set_gain.sh
set_hdmi_live_stream.sh
set_overlays.sh
```

The files above are new files so no need to back any existing ones up.

Make sure they are executable.

Build / Install Instructions

RemotePlus

Either git clone from the repository or download and extract the tarball available on Github, the location where you download to on your machine doesn't really matter.

https://github.com/Plus24Systems/RemotePlus

Once done cd into the source directory of the RemotePlus and run the following.

```
make
sudo make install
```

This will copy the compiled binary and associated RemotePlus files to the following locations:

```
/usr/local/bin/RemotePlus
```

This is the main application.

```
/usr/share/icons/RemotePlus/
```

This is the location of the application icon files, which are as follows.

Used on the left of the application	Used on the right of the application
go-next.png gtk-cancel.png go-previous.png	user-home.png gtk-ok.png applications-utilities.png

Prior to running the application make sure that the RemotePlus binary is executable and that the icons are world readable.

RemoteServer

The RemoteServer will need to be compiled separatly to the RemotePlus, but prior to this step you will need to check that you have set-up connectivity to the AXIOM camera over Ethernet as the RemoteServer will re-use the UART interface fto communicate to the RemotePlus.

Either git clone from the repository or download and extract the tarball available on Github, the location where you download to on your machine doesn't really matter.

https://github.com/Plus24Systems/RemoteServer

If you have not done so please check the AXIOM Beta Configuration Prerequisites section above.

Once you have established connectivity via Ethernet you will now disable the AXIOM camera from spawning a TTY terminal on the UART interface.

copy `AxiomRemoteServer.c` to the AXIOM camera (either into the `/root` directory or create a new one such as RemoteServer), then compile it like so:

```
gcc RemoteServer.c -o RemoteServer -lpthread
```

Prior to using the RemotePlus you will need to SSH onto the AXIOM camera and start this manually – in the future this will be a service started at boot-time.

Usage

The RemotePlus interface is made up from a number of discrete sections.

Along the top and bottom are the menu widgets – they are activated by their corresponding RemotePlus hardware buttons or keyboard keys on a desktop/laptop.

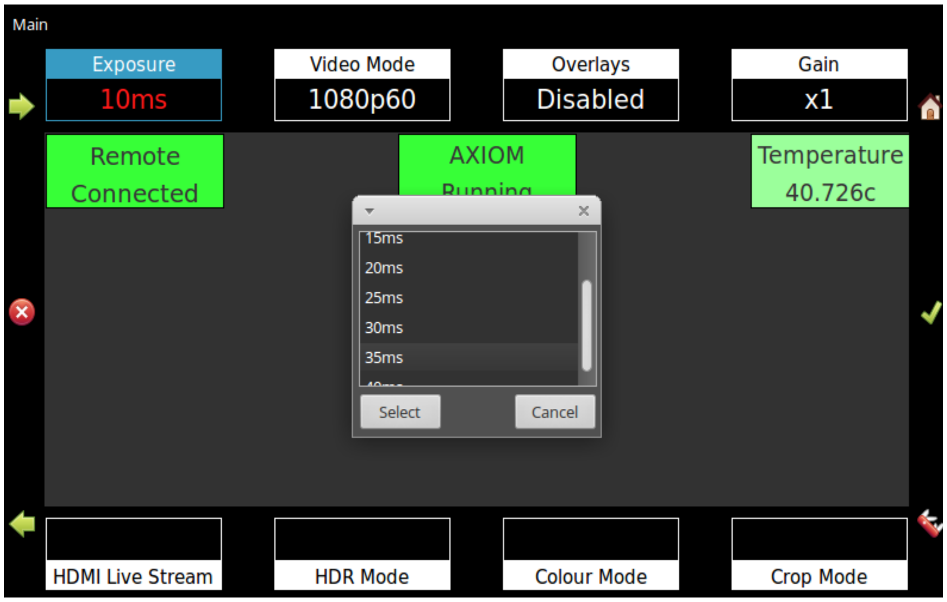
The defaulk key mappings, as defined in RemotePlusDefines.h are as follows.

	Menu Widget 1	Menu Widget 2	Menu Widget 3	Menu Widget 4
Top Row	**q**	**w**	**e**	**r**
Bottom Row	**z**	**x**	**c**	**v**

On either side of the screen are the function buttons – they are (or will once the application is ready for production) map to the following.

Left Hand Side	Right Hand Side
➡ Page Right	🏠 Home Page
✖ Cancel	✓ OK / Select
⬅ Page Left	🔧 Config Page

When one of the menu widgets is activated the colours change as below and the values pop-up is activated – which is in the middle of the screen.


Exposure 10ms
Video Mode 1080p60
Overlays Disabled
Gain x1
Remote Connected
AXIOM Running
Temperature 40.726c

The menu values pop-up in the screenshot above is from a Linux desktop, on the RemotePlus hardware there are no buttons and the frame is not shown.

The user can select their required value by either turning the rotary knob on the RemotePlus hardware or use the mouse to select.

Once the required value is selected the user can confirm this on the RemotePlus hardware by either pressing the rotary knob wheel, or pressing the OK function button. For desktop/laptop users they can click on the required OK/Cancel buttons.

In the center is the status display area, this is where all of the status widgets are contained.

In this pre-production version the three status widgets are as as follows.

Connection Widget – this is the left hand widget and is responsible for indicating the status of the UART connection to the AXIOM Camera.

AXIOM System Widget – this is in the top center and this widget is used to indicate the status of the AXIOM firmware, is it running or not. **NOTE**: See the Dev notes below concerning this widget.

Temperature Widget – this is the righthand widget and is used to indicate the cameras temperature.

Developer Notes

The RemotePlus GUI application is constructed from the following files:

RemotePlusApp.cpp

RemotePlusApp.h

These two files instantiate the main wxWidget RemotePlus application.

RemotePlusMain.cpp

RemotePlusMain.h

RemotePlusDefines.h

The RemotePlusMain files are for the main RemotePlus class, this forms the main application framework that all of the other class artefact's integrate into.

This is a multi threaded class with two separate worker threads: *RemotePlusUARTThread* which is to handle communication with the AXIOM camera via the hosts UART interface (hard-coded as `/dev/ttyUSB0`) and *RemotePlusGPIOThread* which is intended to handle all of the IO to the physical GPIO sub-system – but currently not available in this release.

The RemotePlusDefines header is a common file used across the whole application and is used as a central location for all of the #defines – this has been done to keep the code more readable and maintainable.

There should be no need to adjust anything in this file in normal usage apart from two key sections.

The first is the keyboard mapping section – this allows a developer, when using a desktop/laptop, to re-map the main menu widgets key assignment. As the key for the bottom right menu widget is set as a default to ‘Z’ this allows German keyboard users to quickly re-map the ‘Y’ key – so a handy feature for the AXIOM team in Austria!

Another key #define, which will be implemented in the next release, is *UseWiringPi.* This is used to enable or disable the use of the wiringPi library, when set to 0 the application will compile and run without wiringPi being installed. For usage on the hardware remote this **must** be set to 1.

ConnectionWidget.cpp

ConnectionWidget.h

These files are responsible for indicating the status of the UART connection to the AXIOM camera. This is handled by the *ConnectionThread* class thich polls to see if the UART interface (`/dev/ttyUAB0`) is available every 1000ms.

All of the other widgets that interact with the UART will send any connection events to this widget to indicate if there is an error sending information to the AXIOM camera over the UART interface..

AxiomSystemWidget.cpp

AxiomSystemWidget.h

These files are responsible for indicating the status of the AXIOM firmware runstate.

Inbound UART messages are handled by the *RemotePlusUARTThread* class in the RemotePlusMain files and passed though using event.

NOTE: This Widget us currently under review as an update to the Beta's Arch Linux now starts this process on boot-up – in this release a dummy function in RemoteServer sends back a response that kick_manual.sh has been called. If your Beta does not start the firmware automatically you will need to uncomment a line in RemoteServer.c – see Known Issues below.

Release History

v0.1 – Initial Release – 2x.12.2016

This initial release is intended to be used by the AXIOM developer team and other developer-orientated users of the AXIOM Beta. It primarily consists of the main RemotePlus application and the RemoteServer.

Usage is intended to be Linux desktop/laptop-based and no wiringPi GPIO interface code is present.

The UI is made up of the main menu widgets together with three function widgets – Connection, AXIOM System and Temperature.

Interaction with the AXIOM Beta, while limited to a small number of functions, is working.

Set Exposure
Set Gain
Set HDMI Live Stream
Set Overlays

In addition to a getter for the zynq temerature monitor.

Known issues

If the user presses a second menu keyboard key while the pop-up is already active the remote will overlay this second pop-up on the first, resulting in the Select or Cancel button having to be clicked more than once to dismiss the pop-ups.

While considered a defect this is low priority as this issue cannot occur on the RemotePlus hardware due to the fact that the GPIO interface operates in this mode (technically a keyboard *could* be used on the remote hardware, but this is unlikely in normal operation).

On start-up the RemotePlus sends a GET_GAIN_REQUEST message which the RemotePlus acknowledges with a GET_GAIN_RESPONSE. While the event is captured by the main *RemotePlusUARTThread* thread the code in MenuWidget to handle this has not been implemented.

Another observed issue is that the AXIOM STARTING UART message from the RemoteServer is sometimes missed by the RemotePlus, this is minor for now but needs to be investigated.

The function icons are purely decorative in this release as they rely on the GPIO interface, however the Left Page & Right Page will be accessed via the Left & Right cursor keys in a future release as this will be required when the application starts to use multiple wxWidget workbook pages (for configuration screens and so on).

As noted above regarding the AxiomSystemWidget the corrisponding call in RemoteServer has been commented out to prevent kick_manual.sh being called when the firmware is running on the Beta.

If your Beta does not have this feature you will need to uncomment the following line in RemoteServer.c and re-compile.

```
//status = execv ("/root/kick_manual.sh", NULL);
```

Next Release Overview

The next release will focus on the following:

- Move the hard-coded menu widget stereotypes to a config file.
- Remove the seperate UART TX methods into the *RemotePlusGPIOThread* class fully.
- Initial Configuration Page.
- Integrate the GPIO sub-system.
- Intial i18n/Translation framework (By Ümit Dogan)
- Any upstream new features from the AXIOM Dev Team.
- Bug fixes.

Acknowledgements

Thanks goes to the Apertus team, especially Sebastian and Herbet.

Imprint

Author: Phil Kerr.

The latest version of this document can be found in the Plus24 Systems GitHub repository:

https://github.com/Plus24Systems/Documentation