APPLICATION NOTE

Using Single Board Computers (SBCs) with Basler USB3 Vision and GigE Vision Cameras

Document Number: AW001450
Version: 02 Language: 000 (English)
Release Date: 07 September 2017

BASLER
the power of sight

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany

Tel. +49 4102 463 515
Fax +49 4102 463 599

support.europe@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA

Tel. +1 610 280 0171
Fax +1 610 280 7608

support.usa@baslerweb.com

Asia-Pacific

Basler Asia Pte. Ltd.
35 Marsiling Industrial Estate Road 3
#05–06
Singapore 739257

Tel. +65 6367 1355
Fax +65 6367 1255

support.asia@baslerweb.com

www.baslerweb.com

Table of Contents

1 Introduction

We have tested and reviewed several popular Single Board Computers (SBCs) commonly used with Basler USB3 Vision and GigE Vision cameras and Basler's pylon Camera Software Suite. We hope that by sharing our experience, we might save the reader some time and effort as they explore using SBCs in their own applications.

1.1 List of Single Board Computers Reviewed

The SBCs tested vary in design from low-cost, hobbyist-oriented boards to industrial-grade, professional systems. Therefore, performance and features can vary significantly between them. **However, when your expectations align with the SBC's intentions, you can be very successful**.

SBC	Price	Architecture	More information
Nvidia Jetson TX1	$499	ARM Cortex A57 64 bit	nvidia.com
Nvidia Jetson TK1	$199	ARM Cortex A15 32 bit	nvidia.com
Qualcomm Dragonboard 410c	$75	ARM Cortex A53 64 bit	qualcomm.com
Hardkernel Odroid XU4	$59	ARM Cortex A15 32 bit	hardkernel.com
Raspberry Pi 3	$35	ARM Cortex A53 32 bit	raspberrypi.org
Raspberry Pi 2	$35	ARM Cortex A7 32 bit	raspberrypi.org

1.2 Rankings Based on Basler Test Results

<table>
<tr><th>SBC</th><th>Price</th><th>USB Usability</th><th>GigE Usability</th><th>Relative Performance Level</th><th>Recommended Applications</th></tr>
<tr><td>Jetson TK1</td><td>$199</td><td rowspan="2">1</td><td>1</td><td rowspan="2">“High-End”</td><td rowspan="2">Industrial</td></tr>
<tr><td>Jetson TX1</td><td>$499</td><td>2</td></tr>
<tr><td>Odroid XU4</td><td>$59</td><td>2</td><td>3</td><td rowspan="2">“Mid-Range”</td><td rowspan="2">Light-Industrial</td></tr>
<tr><td>Raspberry Pi 3</td><td>$35</td><td rowspan="2">3[1]</td><td rowspan="2">4[2]</td></tr>
<tr><td>Raspberry Pi 2</td><td>$35</td><td rowspan="2">“Low-End”</td><td rowspan="2">Non-Industrial</td></tr>
<tr><td>DragonBoard 410c</td><td>$75</td><td>4[1]</td><td>5[3]</td></tr>
</table>

[1] Only USB 2.0 is available (~37 MB/sec)
[2] Only 100Base-T Ethernet networking is available (~12 MB/sec)
[3] No RJ45 LAN connection available

1.3 Guidance Based on Basler Test Results

SBC	Guidance
General	▪ Pure processor capability and price are not a perfect measure of how an SBC will perform with USB 3.0 and Gigabit Ethernet based cameras. To what extent the SBC implements supporting hardware like memory, connectivity, etc. is critical. ▪ Likewise, some SBCs are intended mainly as development kits for the underlying SoM / SoC (e.g. Nvidia TK1 / TX1). Therefore, a custom-designed carrier board fitting the specific needs of the user's application may ultimately perform better. ▪ There are currently some concerns regarding the RTL8153 Gigabit Ethernet chipset. We have seen two cases in our testing where a board using the RTL8153 network chipset had severe issues with packet loss / stability during image acquisition (Jetson TX1 and Odroid XU4). Further investigation is needed.
Jetson TK1	▪ Suitable for industrial USB3 Vision and GigE Vision applications. ▪ Ability to use graphics processing unit (GPU) for image processing makes it among the most powerful boards for processing.
Jetson TX1	▪ Suitable for "high end" USB3 Vision applications. ▪ Its implementation of Gigabit Ethernet appears unstable though, possibly due to a GigE-to-USB chipset (RTL8153AI-VB-CG). ▪ Has the most processing "horsepower" of all the boards. Can use GPU (Graphics Processing Unit) for processing.
Odroid XU4	▪ This board does quite well for mid-range USB3 Vision camera applications. ▪ Its implementation of Gigabit Ethernet appears unstable, possibly due to a GigE-to-USB chipset (RTL8153-CG).
Raspberry Pi 2 & 3	▪ Only USB 2.0 and 100Base-T Ethernet connections are available, but they both appear to be quite stable. ▪ These are a good choice for low-end applications, and perhaps the Raspberry Pi3 could be used for light-industrial mid-range applications.
DragonBoard 410c	▪ This board is probably good for USB 2.0 applications that favor mobility and power saving over performance, but it does not have an RJ45 connection for operating Ethernet-based cameras. ▪ A more industrial version of this board exists as the Inforce 6309. Among other improvements, it includes an RJ45 connection.

2 Testing Setup

NOTICE

We will not cover the installation of the SBC's Operating System (OS) in this application note.

As these procedures vary for each board and OS, and they may change over time, it is always best to consult the manufacturer of the board for the most current and proper instructions for installing the board's OS.

2.1 SBC Operating Systems and pylon Versions Tested

SBC	Operating System	pylon Camera Software Suite Version
Jetson TK1	Jetpack 2.3.1 (Ubuntu 14.04) https://developer.nvidia.com/embedded/downloads#?tx=$software,l4t-tk1	5.0.9 Linux ARM 32 Bit hardfloat
Jetson TX1	Jetpack 2.3.1 (Ubuntu 14.04) https://developer.nvidia.com/embedded/downloads#?tx=$software,l4t-tx1	5.0.9 Linux ARM 64 Bit
Odroid XU4	Ubuntu_16.04-Mate-odroid-XU3-2061011.img.xz http://odroid.com/dokuwiki/doku.php?id=en:xu3_release_linux_ubuntu	5.0.9 Linux ARM 32 Bit hardfloat
Raspberry Pi 3	Jessie With Pixel (NOOBS v2_1_0) https://www.raspberrypi.org/downloads/noobs/	5.0.9 Linux ARM 32 Bit hardfloat
Raspberry Pi 2	Jessie With Pixel (NOOBS v2_1_0) https://www.raspberrypi.org/downloads/noobs/	5.0.9 Linux ARM 32 Bit hardfloat
DragonBoard 410c	Linaro 17.04 https://www.96boards.org/documentation/ConsumerEdition/DragonBoard-410c/	5.0.9 Linux ARM 64 Bit

Note

You can consult your system's documentation or run "`uname -a`" in a terminal to find your architecture:
`armhf / armv7l` = ARM 32 Bit hardfloat.
`arm64` = ARM 64 Bit

2.2 pylon Camera Software Suite Quick Installation

To download a suitable pylon Camera Software Suite package:

Download a suitable pylon Camera Software Suite package on the Basler website:

www.baslerweb.com/

pylon packages are available for the following architectures:

- Linux ARM
- Linux x86
- Windows (32 & 64 Bit)
- OS X

For Linux installation, there are the following possibilities:

- Linux Manual Installation (For any distribution):
  1. Extract the files in the downloaded .tar.gz file to a folder: (e.g**. /home/me/pylon**).
  2. Install the pylon SDK: `sudo tar -C /opt -xzf /home/me/pylon/PylonSDK*`
  3. If using USB3 Vision cameras:
     Setup pylon USB support: `sudo /home/me/pylon/setup-usb.sh`
  4. Plug in the camera.
  5. Test building the pylon SDK samples:
     Navigate to: **/home/me/pylon/Samples/C++** and run: `make`
  6. Test your camera in pylon Viewer: `sudo /opt/pylon5/bin/PylonViewerApp`

- Linux Easy Installation (**Debian-based distributions only**):

  Do not use built-in software managers like Ubuntu Software Center, etc. for installing pylon. Long installation times and/or problems may occur due to variations in these managers. We highly recommend using dpkg **from a shell only**
  1. Install the **.deb** installation package with dpkg: `sudo dpkg -i ./pylon*.deb`
  2. Plug in camera.
  3. Test building the pylon SDK samples:
     Navigate to: **/opt/pylon5/Samples/C++/** and run: `make`
  4. Test your camera in pylon Viewer: `sudo /opt/pylon5/bin/PylonViewerApp`

3 General Settings for Best Performance

3.1 Linux OS: USB - Specific

Disabling autosuspend keeps the system from powering down the camera.

To disable autosuspend:

You have two possibilities to disable autosuspend:

- The following disabling autosuspend method must be done after each reboot.
  1. Run `sudo sh –c 'echo -1 > /sys/module/usbcore/parameters/autosuspend'`
- To disable autosuspend on every boot, you have two possibilities:
  - Add `echo -1 > /sys/module/usbcore/parameters/autosuspend` to your **/etc/rc.local** script.

    As an alternative, you can do the following:
  - You can use a distribution-specific mechanism like boot parameters, **/etc/modprobe.d/** directory, systemd service units or init scripts.

    These mechanisms are not described in this application note.

To enable support for large image transmission (greater than 2 Mbyte):

1. Run: `sudo sh –c 'echo 1000 > /sys/module/usbcore/parameters/usbfs_memory_mb'`

3.2 Linux OS: GigE - Specific

To increase the socket buffer memory size to help prevent dropped packets:

Example: 32 Mbyte:

1. Set socket receive maximum: `sudo sysctl –w net.core.rmem_max=33554432`
2. Set socket send maximum: `sudo sysctl –w net.core.wmem_max=33554432`
3. Set socket receive default: `sudo sysctl –w net.core.rmem_default=33554432`
4. Set socket send default: `sudo sysctl –w net.core.wmem_default=33554432`

3.3 Sample Script for Linux Operating System Settings

```
#!/bin/sh

# increase the memory buffers for GigE (32 MB for example)
sudo sysctl –w net.core.rmem_max=33554432
sudo sysctl –w net.core.rmem_default=33554432
sudo sysctl –w net.core.wmem_max=33554432
sudo sysctl –w net.core.wmem_default=33554432

# turn off USB power autosuspend
sudo echo -1 > /sys/module/usbcore/parameters/autosuspend

# increase USB memory for support of > 2MByte images
sudo echo 1000 > /sys/module/usbcore/parameters/usbfs_memory_mb
```

3.4 Basler pylon Camera Software Suite: General Information

NOTICE
Please refer to the INSTALL and README files within the pylon Camera Software Suite for complete details and recommendations.

Recommendations

- **Ensure that pylon has sufficient permissions for real-time thread priorities.**
  This will allow pylon to increase the priorities of the internal threads to Real-Time class (Receive, Transfer Loop, Grab Engine, and Grab Loop threads).
  This will in turn maximize stability and help to prevent image data losses.
  See the INSTALL document on how to set these permissions.
- **Consider using triggering instead of free-run**.
  Triggering an image only when it is needed, reduces system load, thus increasing system stability.
- **Consider increasing MaxNumBuffer (default = 10)**
  The host system always processes images at a different speed than the camera sends them (either slower or faster). If the host is slower than the camera, enough buffers must be ready to hold incoming images or they will be dropped.
  Dropped frames in this case are logged by pylon as “Missed Frame Count” (USB3 Vision) or “Buffer Underrun Count” (GigE Vision).
  Allowing your application to use more buffers by increasing the MaxNumBuffer (GenICam name) parameter will help to avoid this condition. It is also recommended to evaluate your host processor and application code as a whole, to determine if it is capable of supporting the desired speed from the camera.

For more information on the MaxNumBuffers, MissedFrameCount and BufferUnderrunCount parameters refer to the *pylon Programmer's Guide and API Reference* documentation.

3.5 pylon Camera Software Suite: USB – Specific

Recommendations

- Try to use the highest maximum transfer size possible for best performance (GenICam name: MaxTransferSize).
  Lower this value if signs of instability are seen (image data loss).
- Try to use the highest Device Link Current Throughput Limit possible for best performance (GenICam name: DeviceLinkThroughputLimit)..
  Lower this value if signs of instability are seen (image data loss).
- Increasing the Num Max Queued Urbs can increase system stability as well (GenICam name: NumMaxQueuedUrbs).

For more information on the MaxTransferSize, DeviceLinkThroughputLimit and NumMaxQueuedUrbs parameters refer to the *pylon Programmer's Guide and API Reference* documentation.

3.6 Basler pylon Camera Software Suite: GigE – Specific

Recommendations

- **Use the largest Packet Size possible**.
  - On the camera this is called Packet Size (GenICam name: GevSCPSPacketSize) and
  - on the network adapter, this is called MTU.

  This will also maximize stability. Packets which are too small will overwhelm switches and NICs and lead to instability (packet loss and camera connection loss).
  Packets which are too large may simple be unsupported (100 % packet loss and possible camera connection loss).
  As a rule of thumb, start with a Packet Size = 1500.
- **Adjust the inter-packet delay** (GenICam name: GevSCPD) such that the BandwidthAssigned parameter is equal to, or slightly greater than, the DeviceMaxThroughput value.
  This will ask the network adapter for only the amount of bandwidth needed for the camera. This in turn may help to increase stability, as it reduces the load on the network adapter to only what is necessary.

For more information on the GevSCPSPacketSize, GevSCPD, BandwidthAssigned and DeviceMaxThroughput parameters, refer to the *pylon Programmer's Guide and API Reference* documentation.

4 Board-Specific Testing Notes and Tips

NOTICE

Only the stability and performance of Image Acquisition was tested in this Application Note.

As every application is different, it is impossible to completely test all possible use cases.Therefore, only stability and performance of image acquisition is tested at this time.

However, if user feedback shows that it would help to benchmark certain application tasks, like image saving, preprocessing, image transmission, and so on, we will gladly update this Application Note as much as possible.

And of course, we always welcome the opportunity to discuss your specific application with you and lend tailored advice wherever possible.
We invite you to contact Basler support at any time through the contact information in the beginning of this Application Note.

4.1 Jetson TK1 and Jetson TX1

4.1.1 USB Testing Notes

- **USB 3.0 is not turned on by default in the Jetsons**.
  See below for instructions to turn it on.
- The Jetsons only have one USB 3.0 port.
  A powered USB 3.0 hub is therefore necessary if using keyboard, mouse, etc. alongside cameras. This worked quite well though.
- Image acquisition was very stable after the standard settings in Section 3 were applied.

TIP:
To enable USB 3.0 Support on the Jetson TX1 and TK1:

1. Open this file in a text editor: **/boot/extlinux/extlinux.conf**
2. Change `usb_port_owner_info=0 to: usb_port_owner_info=2`
3. Reboot.

4.1.2 GigE Testing Notes

- The **TK1** was quite stable at full bandwidth.
  Only 0.000097 % of packets were lost, and all were recovered by pylon's resend mechanism, so no images were lost. The TK1 uses a RTL8111GS chipset for its NIC, which connects to the host processor via PCIe.
- The **TX1** on the other hand, suffered heavy packet and image losses.
  Despite our best efforts, only low-bandwidth stability was achieved. The TX1 uses a different Gigabit Ethernet chipset than the TK1, the RTL8153AI-VB-CG, which connects to the host processor via USB 3.0 instead of PCIe. This could be a problem and should be investigated further.

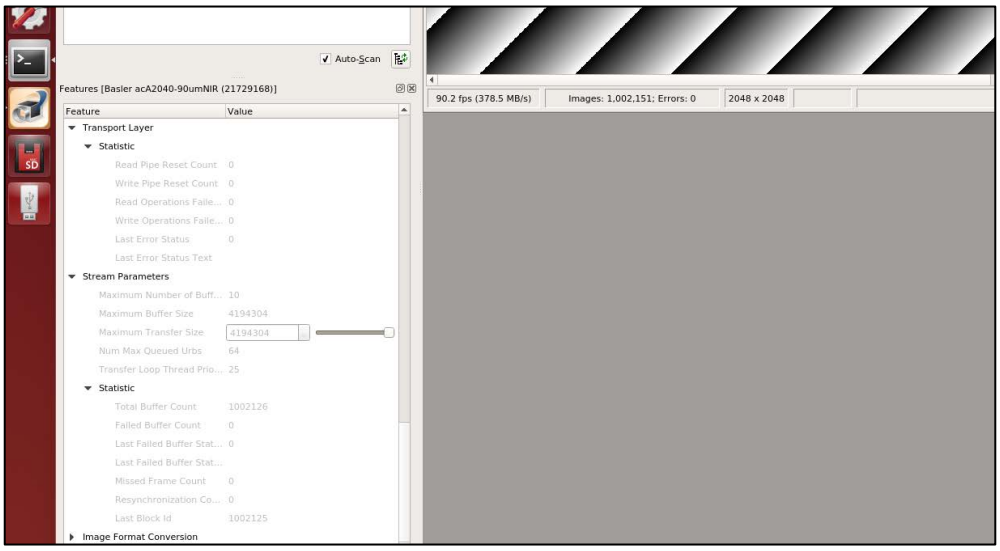
TIP:
To obtain best performance on the TK1:

1. Set camera Packet Size and NIC MTU = 8000

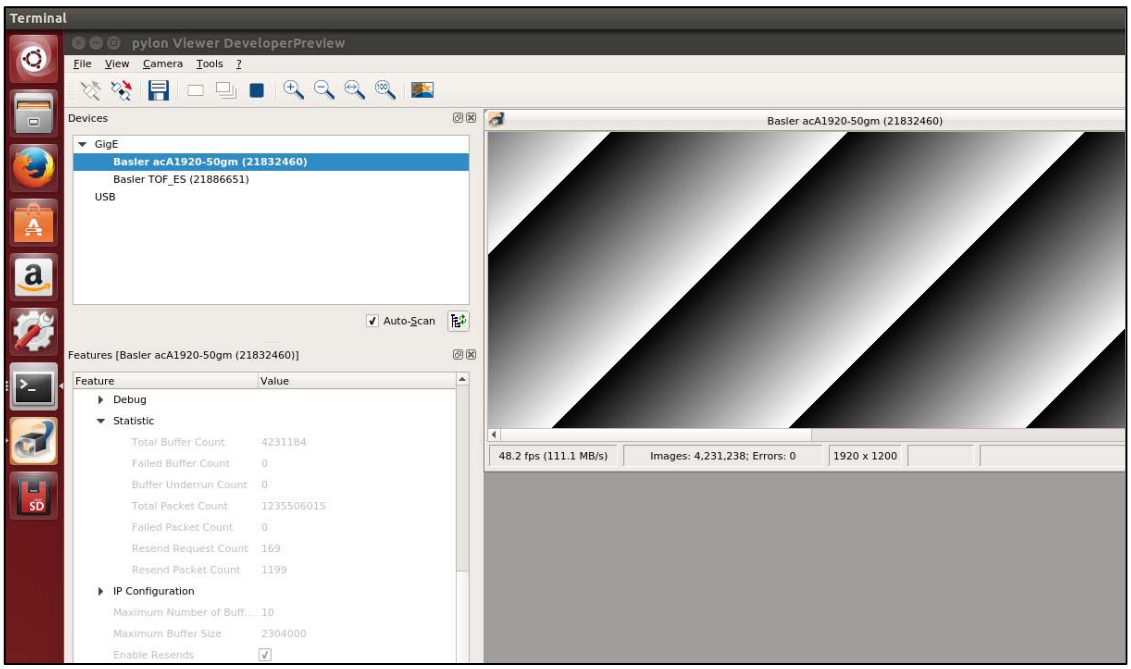
Example results, see next page.

4.1.3 Example Results

- **Jetson TK1, USB 3:**
  2048 x 2048, 90.2 fps, 8 bit mono (379 MB/sec)
  1 million images, no errors



- **Jetson TK1, GigE:**
  1920 x 1200, 48.2 fps, Mono 8 bit (111 MB/sec)
  4.2 Million images, no errors



4.2 Odroid XU4

4.2.1 USB Testing Notes

- Image acquisition was very stable out-of-the-box, even up to 360 MB/sec.
- We did see some occasional “Missed Frames” during USB 3.0 testing.
  This could be a result of some temporary increase in CPU load from other processes.
  It is likely that increasing MaxNumBuffer could compensate for this.

TIP:
To reduce the likelihood of Missed Frames with USB 3.0:

1. Increase MaxNumBuffer in pylon (the default used in these tests is 10 buffers.)

TIP:
To disable USB autosuspend in a different way:

1. Edit the file **/etc/default/tlp**
2. Change `Autosuspend=1` to `Autosuspend=0`

4.2.2 GigE Testing Notes

Performance was very disappointing.
Stability was only achieved after several adjustments, and even then, only a maximum of ~50 MB/sec was possible. These are quite similar results to the Jetson TX1 testing, and both the XU4 and TX1 use a variant of the RTL8153 GigE-to-USB chipset.
So, this supports the need to investigate the RTL8153 chipset further.

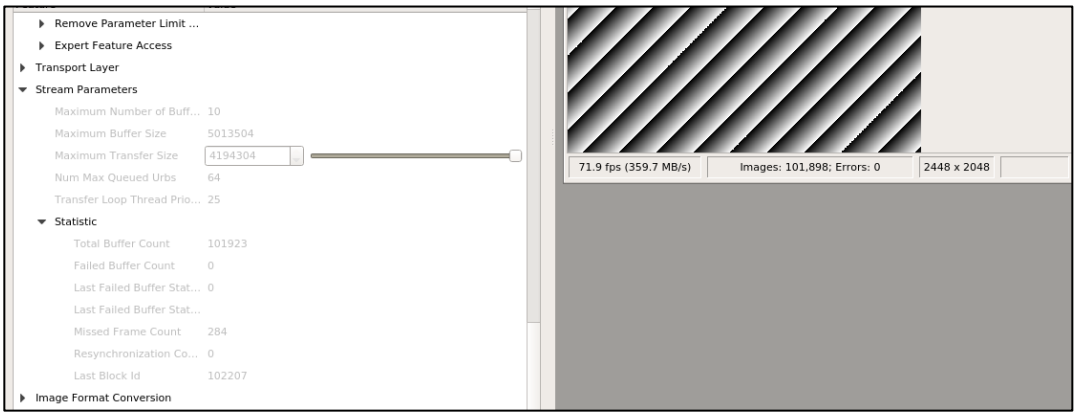
TIP:
To obtain stable (but low bandwidth) GigE Vision image acquisition:

1. Set NIC MTU to 6975.
   Example command: `ifconfig eth0 mtu 6975 up`
2. Set camera Packet Size = 6972
3. Set camera Inter Packet Delay = 8639

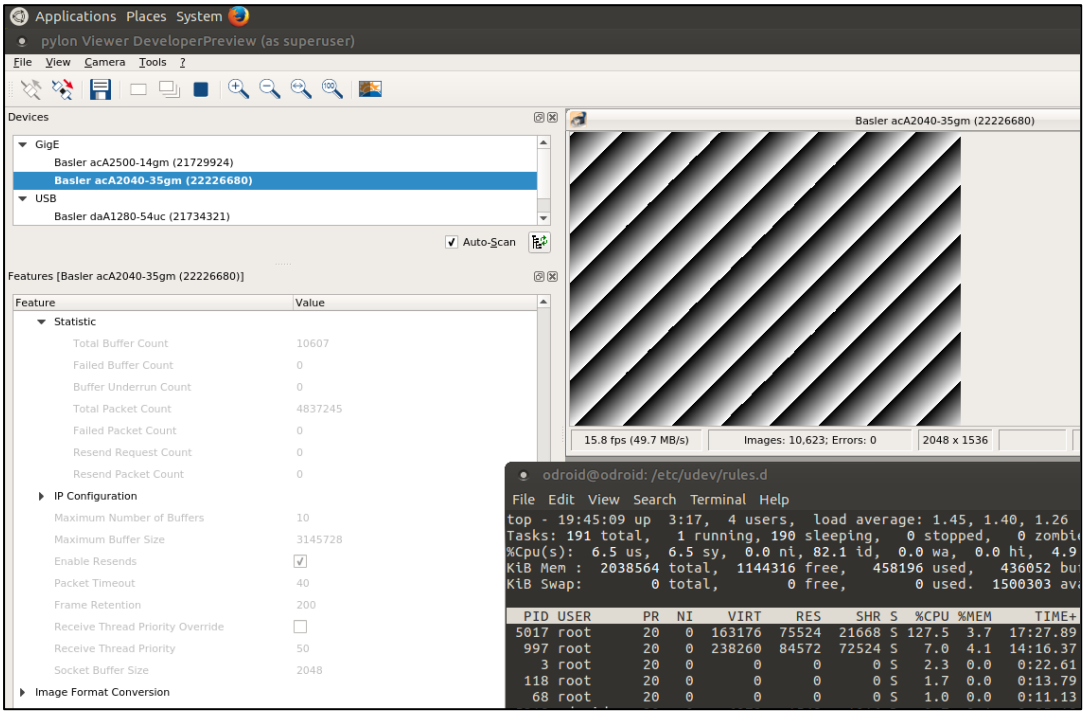
Example results, see next page.

4.2.3 Example Results

- **Odroid XU4, USB 3:**
  2448 x 2048, 72 fps, mono 8 (360 Mb/sec)
  101,898 images, no errors



- **Odroid XU4, GigE:**
  2048 x 1536, 15.8 fps, Mono 8 bit (49.7 MB/sec)
  10,623 images, no errors



4.3 Raspberry Pi 2 and Raspberry Pi 3

4.3.1 USB Testing Notes

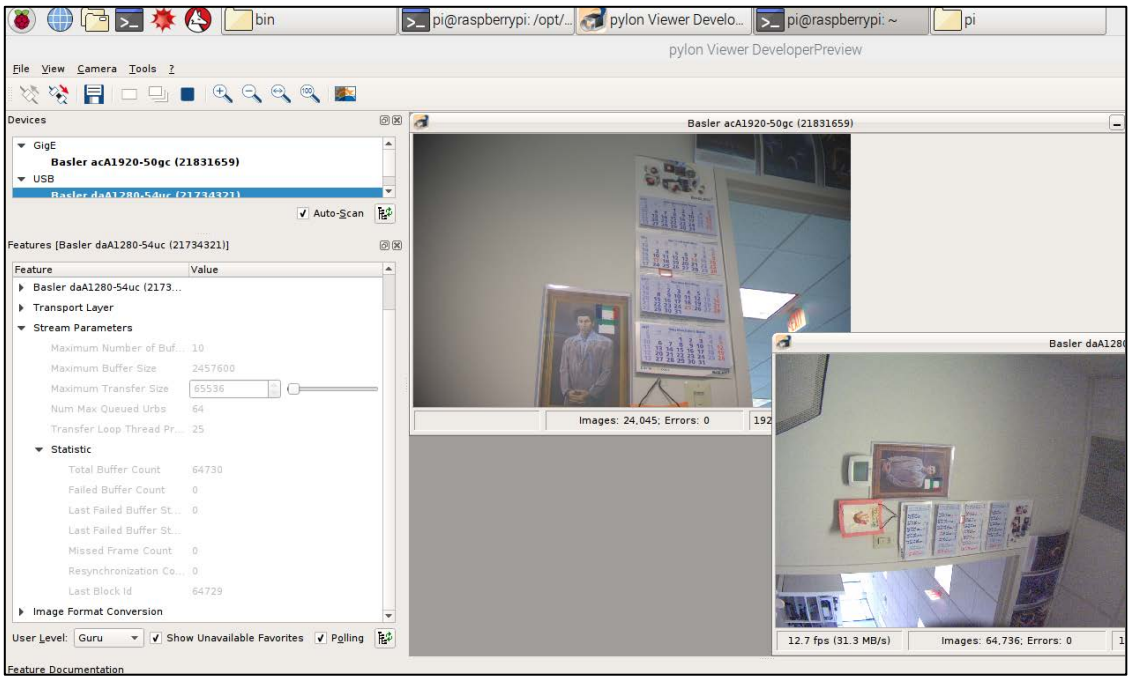
Both the RPi 2 and RPi 3 were remarkably stable ‘out of the box’ when used with USB 2.0 cameras (i.e. Basler dart). There are currently no special things to consider outside of the general optimizations mentioned in this note.

4.3.2 GigE Testing Notes

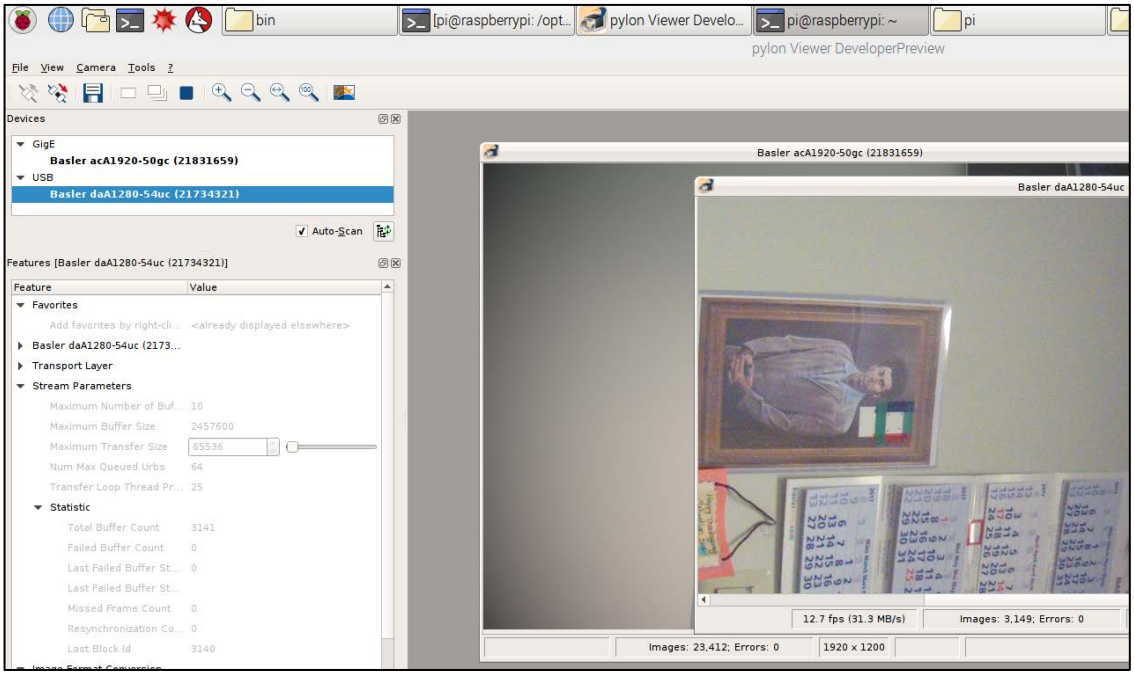
As seen in the USB testing, the RPi 2 and 3 are remarkably stable here as well, as long as one considers there is only 100Base-T network speed available.

4.3.3 Example Results:

- **RPi 3, USB 2.0:**
  1280 x 960, 12.7 fps, YCbCr422 16 bit (31.3 MB/sec)
  64,736 images, no errors
- **RPi 3, GigE:**
  1920 x 1200, 4.7 fps, Bayer 8 bit (10.7 MB/sec)
  24,045 images, no errors



- **RPi 2, USB 2.0:**
  1280 x 960, 12.7fps, YCbCr422 16 bit (31.3 MB/sec)
  3,149 images, no errors
- **RPi 2, GigE:**
  1920 x 1200, 4.7fps, Bayer 8 bit (10.7 MB/sec)
  23,412 images, no errors



4.4 Dragonboard 410c

4.4.1 USB Testing Notes

USB 2.0 image acquisition was quite stable. We did notice a significant improvement in stability when the maximum number of queued URBs was increased, and the bandwidth to the camera was limited to 27 MB/sec.

TIP:
To obtain stable operation, use the following camera settings:

- Device Link Current Throughput Limit = 27 MB
- Num Max Queued Urbs = 100

4.4.2 GigE Testing Notes

We could not test GigE Vision cameras because this board did not have a network port.

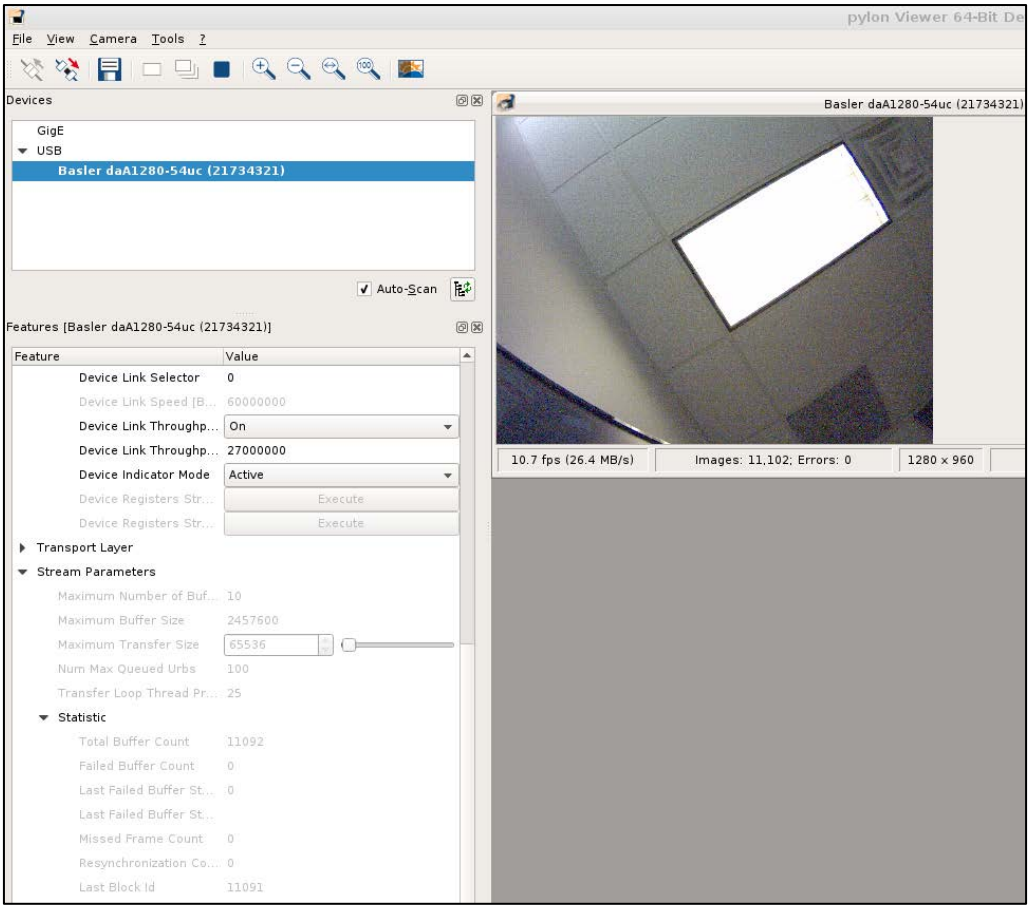
TIP:

There are other boards using the Qualcomm Snapdragon 410c with a network port.
The Inforce 6309 is one such board supporting GigE via a USB 2.0 to GigE converter.

Example results, see next page.

4.4.3 Example Results

Dragonboard 410c, USB 2.0:
1280 x 960, 10.7 fps, YCbCr422 (26.4 MB/sec)
11,102 images, no errors



5 Troubleshooting

5.1 Image Acquisition is Unstable or Fails

Symptoms include:

- “Buffer Incompletely Grabbed”
- “Trailer is corrupt”
- No image capture in pylon Viewer
- `RetrieveResult()` times-out in application
- Lines in image during display
- Failed Buffer Counter > 0
- Camera loses connection

Root causes:

- Primarily caused by sub-optimal settings
- Can also be caused by poor quality host controllers and / or cables, or long cables
- Can also be caused by electronic noise in the system

Remedies:

- See Section 3 and Section 4 for recommended settings.
- Review your settings with Basler support
  To save the settings: pylon Viewer > **Camera** Menu > **Save Features**
- Log your application with the instructions in the script:
  /opt/pylon5/bin/StartPylonViewerWithLogging.sh
- Check host controller and cables against Basler’s recommended accessories:
  - See application note:
    https://www.baslerweb.com/en/support/downloads/document-downloads/recommended-accessories-for-basler-usb-3-0-cameras-usage-recommendation/
  - See application note:
    https://www.baslerweb.com/en/support/downloads/document-downloads/maximum-bandwidth-measurements-of-usb-3-0-host-controllers/
- Check if your system is susceptible to electrical noise:
  See following application note:

  https://www.baslerweb.com/en/support/downloads/document-downloads/avoiding-emi-and-esd-in-camera-installations/

5.2 Keyboard Input not Working in pylon Viewer

Symptoms include:

- pylon Viewer does not respond to keyboard commands in some Linux distributions.
- Error when starting pylon Viewer from a terminal:
  `xkbcommon: ERROR: failed to add default include path auto`

Root causes:

pylon 5.0.9 now uses Qt5 for its GUI.
Some Linux systems do not have a needed environment variable set.

Remedies:

Manually export the required variable `QT_XKB_CONFIG_ROOT`.

- Example: Export variable on-the-fly when running pylon Viewer:
  `sudo QT_XKB_CONFIG_ROOT=/usr/share/X11/xkb ./PylonViewerApp`
- **Note**:
  `sudo` will ignore variables if they are set in `/etc/environment`.

5.3 Errors when using multiple cameras

Symptoms include:

- "PrepareGrab (StartStreaming) failed for device"
- "Insufficient system resources exist to complete the API"

Root causes:

The number of available open file descriptors is exhausted.

Remedies:

Increase the number of open file descriptors with the command:
`ulimit –n <number>`

5.4 Error when Running Programs with sudo or as Root

Symptoms include:

- "SetRTThreadPriority failed:1 Operation not permitted"
- pylon Viewer > **Stream Parameters** > **Statistic** > **Transfer Loop Thread Priority** = `0`

Root causes:

pylon is trying to set real-time priorities, but cannot due to system permissions.

Remedies:

Add the following line to **/etc/security/limits.conf** and reboot:

```
*       -       rtprio      99
```

5.5 Failed to Open Device Error in Application

Symptoms include:

- "Failed to open device" … "Libusb error: LIBUSB_ERROR_ACCESS."
- Camera displayed as "Basler ()" in pylon Viewer Devices list.
- Terminal displays error: **Failed to open device with path …**

Root causes:

User does not have enough system permission.

Remedies:

You have two possibilities:

- Ensure you execute `setup-usb.sh` after having installed pylon.
  After re-login, the permission problem should be gone.
- Run application as root or with sudo: `sudo` **/opt/pylon5/bin/PylonViewerApp**

5.6 Segmentation Fault

Symptoms include:

Segmentation fault when running pylon Viewer or user application.

Root causes:

- Mismatch between host system architecture and pylon architecture.
- Host system is missing an interface which pylon supports.

Remedies:

- Ensure that the pylon variant you are using matches your system
  (i.e: ARM 32 bit hard float = “armhf” variant of pylon, “ARM 32bit soft float = “armel”, etc.)
- If you have the correct pylon variant installed, but the host system is missing an interface that pylon supports, manually remove the library files which support that interface from the pylon installation:

Interface Missing from Host	Libraries to remove from /opt/pylon5/lib
Ethernet	libpylon_TL_gige-5.0.9.so libgxapi-5.0.9.so libpylon_TL_gige.so libgxapi.so
USB	libpylon_TL_usb-5.0.9.so libuxapi-5.0.9.so pylon-libusb-1.0.so libpylon_TL_usb.so libuxapi.so

Revision History

Document Number	Date	Changes
AW001450**01**000	25 Jul 2017	Initial release candidate of this document.
AW001450**02**000	07 Sep 2017	Modified link for the DragonBoard 410c on page 2. Corrected links in 2.2 “pylon Camera Software Suite Quick Installation” on page 5. Updated 3.1 “Linux OS: USB - Specific” section on page 6. Updated 3.4 “Basler pylon Camera Software Suite: General Information” on page 7. Updated 5.5 “Failed to Open Device Error in Application” on page 20.