Eder F. Torres Lopez

eder.fernando.torres3@gmail.com

Abstract

One of the reasons to incorporate a single board computer into an embedded system project is to add advanced peripherals such as the ones that are connected through USB. In this document we go over on how to add one of these peripherals, it will be a USB HD camera. The live video will be streamed to either a Windows system or Linux system through the network with very small delay (in order to see real-time video and can be used to control robotics system for example), it will use a small percentage of the CPU, and it will be done using an open-source program, GStreamer, which will be compiled from source code to enable the required features to make the small footprint requirement possible.

Stream h264 using github on BeagleBone black

Using a webcam with an internal h.264 video encoder and GStreamer

Contents

[Introduction and Prerequisites 2](#_Toc120737988)

[1. Install dependencies to build the source code 2](#_Toc120737989)

[2. Install video4linux and dependencies for GStreamer’s uvh264 plugin 3](#_Toc120737990)

[3. Download GStreamer source code 3](#_Toc120737991)

[4. Build and install GStreamer source code 3](#_Toc120737992)

[5. Test GStreamer and the USB camera 7](#_Toc120737993)

[6. Install GStreamer on a separate computer and test the system 8](#_Toc120737994)

[ Install on windows 8](#_Toc120737995)

[ Streaming camera on Windows 11](#_Toc120737996)

[ Install on Linux 13](#_Toc120737997)

[ Streaming camera on Linux 14](#_Toc120737998)

[Conclusion 14](#_Toc120737999)

[Reference and Additional Information 14](#_Toc120738000)

# Introduction and Prerequisites

In this project we will compile and install GStreamer to read h.264 video stream from a webcam (C920) with a BeagleBone Black and read it from another computer.

GStreamer is a multimedia framework, it can be used to encode, decode, stream, parse and modify media sources such as video and audio. It will be used in this application to get video stream from a webcam. Since the BeagleBone Black is not a powerful computer, we don’t want to waste its processing power to encode webcam raw/jpeg video into h264 and then stream it through the network, instead, we will be using a C920 Logitech webcam which contains an onboard h.264 encoder that will compress the video with good quality without wasting machine cycles on the BeagleBone.

To follow this guide successfully it's assumed that the following list has been gathered/followed prior to starting.

* BeagleBone Black powered through the barrel connector, 5V and at least 1 amp of current.
* Webcam with internal H.264 encoder and UVC such as the older Logitech C920, Logitech C930e, and others
* Second Computer (either Linux or Windows) that is connected to the network
* Followed the instructions in the following link that I have posted in hackster.io: <https://www.hackster.io/ederfernandotorres3/setup-beaglebone-black-with-device-tree-overlays-9e0ded>
  + If streaming through Wi-Fi is desired, follow the link that I have posted in hackster.io: <https://www.hackster.io/ederfernandotorres3/8821au-wifi-card-and-bbb-22a7ac>
* If this project will be done using a USB Wi-Fi adapter, an externally powered USB hub will be needed as well.

# Install dependencies to build the source code

After the BeagleBone has been setup and is running, we need to make sure that we have a proper build environment and XZ compression utility to build GStreamer from source.

A build environment, XZ compression utility, and lexer and parser generators need to be installed. Enter the user password if asked while installing any of the following utilities. Some of these programs might already be installed in your board depending on if they have been installed some other way (like following my previous guides).

* 1. “sudo apt update” <- to look for available updates
  2. “sudo apt upgrade” <- to install any available updates
  3. “sudo apt install gcc” <- to install the gnu C compiler
  4. “sudo apt install g++” <- to install the gnu C++ compiler
  5. “sudo apt install pkg-config” <- to install a helper tool used to compile applications
  6. “sudo apt install make” <- to install a utility for building and maintaining groups of programs
  7. “sudo apt install xz-utils” <- is general-purpose data compression software
  8. “sudo apt install libglib2.0-dev” <- is a general-purpose C library that is used in many projects
  9. “sudo apt install flex” <- it’s a program that generates lexical analyzer
  10. “sudo apt install bison” <- is a parser generator
  11. “sudo apt install liborc-0.4-dev” <- is a library for compiling programs that operate on arrays of data
  12. “sudo apt install libasound2-dev” <- This package contains files needed to develop software that makes use of libasound2
  13. “sudo apt install libssl-dev” <- it’s needed since the configuration process of the bad plugins will halt halfway if it doesn’t detect the plug in.

Alternatively, we can include all the programs in the same line of command: “sudo apt install gcc g++ pkg-config make xz-utils libglib2.0-dev flex bison liborc-0.4-dev libasound2-dev libssl-dev”

# Install video4linux and dependencies for GStreamer’s uvh264 plugin

Next, we install video4linux

1. “sudo apt install v4l-utils”. It provides a series of utilities for media devices, allowing to handle the proprietary formats available at most webcams, etc.
2. “sudo apt install gudev-1.0”. This is a requirement for uvh264 plugin
3. “sudo apt install libusb-1.0”. This is to install USB header to compile the uvh264 plugin.
4. “sudo apt install libv4l-dev”. This will install video4linux libraries as an abstraction layer to support devices without having to write separate code.

Alternatively, we can include all the programs in the same line of command: “sudo apt install v4l-utils gudev-1.0 libusb-1.0 libv4l-dev”

# Download GStreamer source code

In this step, use “wget” to download the GStreamer source code from GitHub. There are four packages that we are interested in, the first one is the main GStreamer app, the other three are plug-ins used by GStreamer. All of them must be compiled. We’ll use version 1.16.2 since the newer versions use other build tools and some basic executables seemed to not compile by default. The source code and built code will be contained in a single folder for easy access.

1. “mkdir gst-build” <- Makes a new directory inside the current directory we are currently located
2. “cd gst-build” <- It changes our current directory to the one that was just created

“wget” is a command used to do retrieve content from web servers, it will be used to download the source code. The source file have the extension “.tar.xz”, it means that is a Tar archive, compressed with “xz”, which is a lossless data compression tool. “tar” is just an archive used to store several files into one manageable file.

1. “wget <http://gstreamer.freedesktop.org/src/gstreamer/gstreamer-1.16.2.tar.xz>”
2. “wget <http://gstreamer.freedesktop.org/src/gst-plugins-base/gst-plugins-base-1.16.2.tar.xz>”
3. “wget <http://gstreamer.freedesktop.org/src/gst-plugins-good/gst-plugins-good-1.16.2.tar.xz>”
4. “wget <http://gstreamer.freedesktop.org/src/gst-plugins-bad/gst-plugins-bad-1.16.2.tar.xz>”
5. “ls” <- use ls command to list the files and directories located in your current location

Graphical user interface

Description automatically generated with medium confidenceAfter downloading the files, the current directory should look as in Figure 1.

Figure : Contents of the gst-build directory we created

# Build and install GStreamer from source code

To build the code that was downloaded, it must be decompressed and extracted first. The “tar” command will be used with some arguments to decompress the files. The general process for each of the archives is to: decompress and extract the files (which will create a new folder with the decompressed content) inside that folder. There is a script file named “configure” that is provided by GStreamer that will modify and localize the source code so it can be compatible with the current operating system, in this case, it is a Debian distribution running on an ARM processor. While the script is running, it will let us know if there are any errors happened, and in the case of the plug-ins, it will let us know which plug-ins will be installed or not (this will depend on dependencies for those individual plug-ins can be found). After running that script, the “make” command must be executed to compile and build the code. After that step finishes, it needs to be installed to be used. The steps for all four archives are like each other.

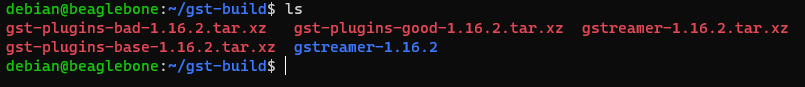
* 1. **GStreamer Main Program:**
     1. “tar -xJf gstreamer-1.16.2.tar.xz” <- decompresses the archive

Figure : After decompressing the file, a new folder is created "gstreamer-1.16.2"

* + 1. A screenshot of a computer

       Description automatically generated with medium confidence“cd gstreamer-1.16.2” <- moves into the directory that was just created after decompressing the archive

Figure : Content of the directory of the archive that was just decompressed

* + 1. Text

       Description automatically generated with low confidence“./configure -prefix=/usr” <- it runs the “configure” script file and it starts checking if all dependencies are installed and what functions are available for the compiler, at the end, it shows a report of what configuration will be used. This will take about 5 minutes.

Figure : Configuration report from GStreamer

* + 1. Text

       Description automatically generated“make” <- This command will start the compilation process using “make”, it will take about 20 minutes.

Figure : No errors after finishing compiling successfully

* + 1. “sudo make install” <- This will install the compiled code so it can be used by the user, it should be installed in the “/home/<user>/gst-build/gstreamer-1.16.2” directory. It takes less than a minute to do this process
  1. **Base Plug-ins:**
     1. “cd ..” <- this is to move back to the previous directory, “gst-build”
     2. “tar -xJf gst-plugins-base-1.16.2.tar.xz” <- decompresses the archive
     3. “cd gst-plugins-base-1.16.2” <- moves into the directory that was just created after decompressing the archive
     4. Text

        Description automatically generated./configure -prefix=/usr” <- same as before, it took about 4 minutes to finish

Figure : Results after configuring Base plugins

* + 1. “make” <- This command will start the compilation process using “make”, it will take about 30 minutes to compile

Figure : This is the message after a successful build

* + 1. “sudo make install” <- This will install the compiled code so it can be used by the user, it took about 3 minutes.
  1. **Good Plug-ins:**
     1. “cd ..” <- this is to move back to the previous directory, “gst-build”
     2. “tar -xJf gst-plugins-good-1.16.2.tar.xz” <- decompresses the archive
     3. “cd gst-plugins-good-1.16.2” <- moves into the directory that was just created after decompressing the archive
     4. Text

        Description automatically generated./configure -prefix=/usr” <- same as before, it took about 5 minutes to finish

Figure : Results after configuration

* + 1. “make” <- This command will start the compilation process using “make”, it will take about 40 minutes to compile

Figure : Results after compiling

* + 1. “sudo make install” <- This will install the compiled code so it can be used by the user, it took about 1 minute
  1. **Bad Plug-ins:**
     1. “cd ..” <- this is to move back to the previous directory, “gst-build”
     2. “tar -xJf gst-plugins-bad-1.16.2.tar.xz” <- decompresses the archive
     3. “cd gst-plugins-bad-1.16.2” <- moves into the directory that was just created after decompressing the archive
     4. Text

        Description automatically generated./configure -prefix=/usr” <- same as before, it took about 5 minutes to finish

Figure : After configuration, make sure that uvch264 shows on plug-ins what will be built

In this step, make sure that the uvch264 plug in will be installed, it is the heart of the project, it makes it possible to grab h264 video feed directly from the web camera. If it doesn’t show there, make sure that “gudev-1.0” and “libusb-1.0” are installed.

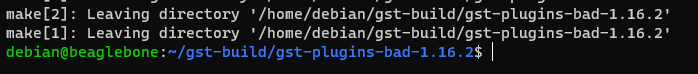
* + 1. “make” <- This command will start the compilation process using “make”, it will take about 35 minutes to compile

Figure : Results after compiling

* + 1. “sudo make install” <- This will install the compiled code so it can be used by the user, it took about 4 minutes

# Test GStreamer and the USB camera

Before continuing further, we need to check if GStreamer works and if it can read the USB camera. In this step we will connect the camera to the USB port (or USB Hub if using a Wi-Fi adapter). A small program compiled in the Base Plug-ins can read the properties of video sources such as web cameras. In this case, it should be able to detect that our C920 webcam can send h264 video steam.

* 1. Plug-in the web cam into the BBB
  2. Run “gst-device-monitor-1.0” <- it will read the properties of all video sources.

A picture containing graphical user interface

Description automatically generatedText

Description automatically generatedAfter running the command “gst-device-monitor-1.0”, the media sources get listed, in this case we have a video source with the name: HD Pro Webcam C920. It also lists the video properties it can stream and can be read by GStreamer such as: video/x-raw (which is raw video from the sensor, it needs to be encoded/decoded), image/jpeg (it can source still images), and video/x-h264 (which is video encoded with the camera’s h264 internal encoder). Resolution, framerate, pixel aspect ratio, etc. are included too. So, whenever we want to grab video from this video source, it needs to match one of these properties.

Figure : After executing the gst-device-monitor-1.0 command, the video sources connected to the BBB get listed

Figure : The command also shows what kind of video can be grabbed from such source and other properties

# Install GStreamer on a separate computer and test the system

To see the video sent by the BBB, setup a computer that can receive RTP video through an RTP player. VLC player can do it, but the delay can be as long as two seconds. In this case we will install GStreamer on a Windows and a Linux machine to play the RTP feed. In the Windows case, the installation folder must be added to the system’s path to be able to call GStreamer’s executables from either the Command Prompt or the PowerShell, this is not the case while running GStreamer on a Linux system.

## Install on windows

* + 1. Graphical user interface, text, application

       Description automatically generatedDownload the MSVC 64-bit executable, this was tested with the runtime installer. The link to download it is: <https://gstreamer.freedesktop.org/download/>

Figure : 1.20.4 runtime installer was used

* + 1. Graphical user interface, text, application, email

       Description automatically generatedRun the installer and follow the installation, on the page where it asks for to “Choose Setup Type”, select: “Custom”

Figure : Choose “Custom” to select the location to install GStreamer

* + 1. Graphical user interface, text, application

       Description automatically generatedOn the following window, choose the location where you want to have GStreamer. It’s important to know the path to add it to the Windows environment path. In this case, I chose the “E:” drive and created a folder name “gstreamer”. Also, make sure that none of the features are deselected (have a red “x”). Again, make sure to install all plugins and features (click on the “down arrow” on the left of “GStreamer 1.0” and make sure everything is selected to install.)

Figure : Notice the “Location:” option. "E:\gstreamer\"

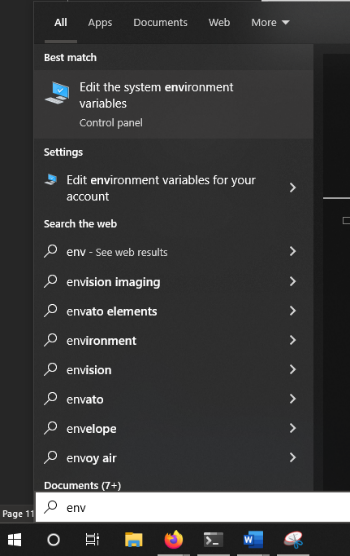
* + 1. Press the “Start” key and type “env” and click on “Edit the system environment variables”

Figure :We need to edit the system's path to add the GStreamer directory

* + 1. Graphical user interface, text, application, email

       Description automatically generatedThe “System Properties” windows will open. Click on “Environment Variables”

Figure : System Properties window

* + 1. Graphical user interface, text, application, email

       Description automatically generatedOn the “Environment Variables” windows, under the “System Variables” section, select “Path” and then select “Edit”.

Figure : Environment Variables window

* + 1. Graphical user interface, application

       Description automatically generatedOn the “Edit environment variable” windows, click on “Browse”. Once the “Browse For Folder” opens, search for the path of the installation of “gstreamer”. Navigate inside that folder, go inside “msvc\_x86\_64” and select the “bin” folder

Figure : On the left window "Browse" needs to be pressed for the right window to open.

* + 1. Graphical user interface, text, application

       Description automatically generatedAfter finding the folder, the path will be added to the previous window.

Figure : The last entry of the system path belongs to GStreamer

* + 1. Click “OK” on all windows to accept the modifications

## Streaming camera on Windows

To test the functionality on Windows, we need to connect to the BBB through SSH. We will also need the IP address of the Windows machine, we can do it the same way it was done to find the IP address of the BBB on the guide: <https://www.hackster.io/ederfernandotorres3/setup-beaglebone-black-with-device-tree-overlays-9e0ded>

1. Once connected to the BBB, enter the following command (modify “host=192.168.1.162” to match your Windows machine”):

“gst-launch-1.0 uvch264src initial-bitrate=800000 average-bitratge=800000 iframe-period=1000 name=src auto-start=true src.vidsrc ! video/x-h264,width=640,height=480,framerate=30/1 ! h264parse ! rtph264pay ! udpsink host=192.168.1.162 port=4000”

Note that there is no space between the words joined by dashes, for example: “video/x-h264” and “iframe-period=1000”.

This will run “gst-launch-1.0” which is one of the applications that was built when compiling GStreamer and its plugins. The rest of the words are arguments telling “gst-launch” how we want to fetch the video: uvch264src is specified to tell gst-launch to use that plug in to read the video feed and use a UVC video source that provides a h264 stream, in our case, the C920 camera. Initial and average bitrate let us specify how many bits per seconds should the transmission use, by default it uses around 2,000,000 bits per second, which is too much data for the kind of video we want to fetch. Video/x-h264 specifies which stream to get from the camera, as previously shown, C920 can give us three different types of video streams, we want h264 which is already compressed. Width, height, and frame rate is to specify the corresponding specifications, they must be available in the list of formats available that was given by “gst-monitor”: previously. H264parse and rtph264pay are meant to extract missing information, and if needed, split the video into packets and/or transforming the packets’ format. Also, they are used to prepare the stream to be sent through the network. Udpsink is used to let “gst-launch” know that it is going to be “sinking into the host” or sending video through the UDP protocol. In this argument, the IP address belongs to the computer that will be receiving the video, a port number is arbitrarily chosen, 4000 was chosen since it’s a high number and unlikely to be taken in the current network.

Text

Description automatically generated

Figure : How the command looks on the BBB before pressing “enter”

Text

Description automatically generated

Figure : gst-launch returns this if the previous command was successfully entered, and no other issues were found

After entering the command successfully, the camera’s front blue led light will turn on.

1. Open the Command Prompt or PowerShell from the Windows computer and enter the following command:

Graphical user interface, text

Description automatically generated“gst-launch-1.0 -v udpsrc port=4000 caps='application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)H264' ! rtph264depay ! avdec\_h264 ! autovideosink sync=false”

Figure : PowerShell before pressing "enter" after typing the command

Graphical user interface, text

Description automatically generated

Figure : gst-launch will return this if no issues were found

A picture containing text, electronics

Description automatically generated After entering the command, gst-launch will return a lot of information about the stream, it will also open a window with a video renderer.

Figure : Video stream by the webcam connected to the BBB

## Install on Linux

Ubuntu is the Linux flavor that is chosen for to host the video being stream by the BBB. To install in Fedora, follow the following link: <https://gstreamer.freedesktop.org/documentation/installing/on-linux.html?gi-language=c>

* Run the following command in the terminal to install all pre-compiled GStreamer apps and plugins: “sudo apt install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-gl gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio”

## Streaming Video on Linux Machine

To start streaming using GStreamer, we use almost the same command as we did for Windows with just one difference, “autovideosink gets replaced with “xvimagesink”:

* Enter this command into the terminal: “gst-launch-1.0 -v udpsrc port=4000 caps='application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)H264' ! rtph264depay ! avdec\_h264 ! xvimagesink sync=false”

# Conclusion

In many embedded systems projects, it’s desired to have remote vision of the project. For example, in projects related to UAV (Unmanned Aereal Vehicle), mobile robots, surveillance, or anything that requires low-cost remote vision. This is a method that can be used to add video streaming capabilities to a system with low latency for real-time requirements. It also has low processing power requirements as can be seen in Reference 1, that even after managing a network connection, two SSH instances and fetching video feed from a USB webcam and streaming it over the network, the BBB is utilizing less than half of its processor power. This project can be used in conjunction with my two previous projects in hackster.io to build a system that has Wi-Fi, video and UART capabilities.

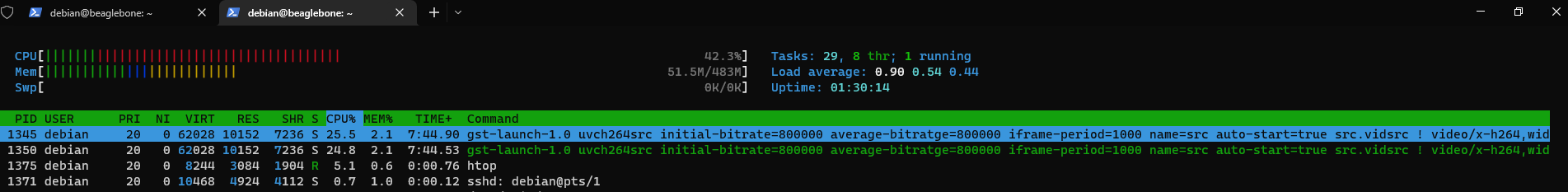


Figure : HTOP show the total CPU usage on the BBB

# Reference and Additional Information

<https://gstreamer.freedesktop.org/download/>

<https://gstreamer.freedesktop.org/documentation/installing/on-linux.html?gi-language=c>

<https://www.hackster.io/ederfernandotorres3/setup-beaglebone-black-with-device-tree-overlays-9e0ded>

<https://www.hackster.io/ederfernandotorres3/8821au-wifi-card-and-bbb-22a7ac>

Special thanks to [soren.kuula@gmail.com](mailto:soren.kuula@gmail.com) who had posted a guide back when Ubuntu was still in version 13 and GStreamer had just made the jump from 0.10 to 1.0, I can’t longer find the guide but I hope this guide to be as useful as the original.