Qualcomm Developer Project TurboxC610&C410-Performance-demo

Project Submission

|  |  |  |
| --- | --- | --- |
| **Attribution:** |  | |
| **Email address** | [sunzhen@thundersoft.com](mailto:sunzhen@thundersoft.com)  [yangrong0925@thundersoft.com](mailto:yangrong0925@thundersoft.com)  [wangjie0508@thundersoft.com](mailto:wangjie0508@thundersoft.com)  [Kouzw0723@thundersoft.com](mailto:Kouzw0723@thundersoft.com)  [yansh0810@thundersoft.com](mailto:yansh0810@thundersoft.com) | |
| **Project Title**\* | **TurboxC610&C410-Performance-demo** | |
| **Images**  *Upload up to 5 images of your project*  *Please submit/send the original JPEG/PNG files for all images included in the document* | **C610.png C410.png**  c610 c410 **[alt tag: “Turbox610&410-Performance-demo using The TurboX C610/C410 Open Kit ”]**Type-c**.png**  |  | | --- | | **typc** |   [alt tag: “**using the usb line to develop on turbox C610/C410 development board.** ”]  [alt tag: “use DP line to connect display **.**”]  dp  IP\_Camera.png  ipc  Displayer.png  display | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | Using the TurboX C610/C410 Open Kit, the data of 1080 cameras are collected by RTSP or local files , decoded and send output to HDMI display. | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | Show the powerful decoding power of turbox c610/c410 through 6 decoding videos. | |
| **Materials Required / Parts List / Tools** | Part Name | Link to purchase |
| TurboX C610/C410 Open Kit, | https://www.thundercomm.com/app\_zh/product/1593776185472315 |
| Type-c line | https://item.jd.com/40759941966.html |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **Source Code / Source Examples / Application Executable**  *Link to open source / shareable code repository* | Description | Link |
| [Source Code](https://github.com/canyudeguang/Home_Automation) | [https://github.com/ThunderSoft-XA/T](https://github.com/ThunderSoft-XA/demo-Smart-Motion-detector)urboxC610&C410-Performance-demo |
|  |  |
|  |  |
|  |  |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Build / Assembly Instructions** | Sample outline:   1. Overall design framework and Test environment construction method.      1. Software Build Instructions    1. Prepare a PC (Ubuntu 16.04 window MAC);    2. Install adb ;    3. Configure the compilation environment according to the release note document    4. Write BB file(<Turbox610&410-Performance-demo.bb>), compile the executable file of Turbox610&410-Performance-demo into the system image, and burn the system according to the realease note document. 2. Start Turboxc610&c410-Performance-demo   Start the TurboxC610&C410-Performance-demo according to the 《TurboxC610&C410-Performance-demo\_User Guide》document | |
|  | Sample outline:   1. How does it work?   Demo supports 6-channel hardware decoding, including local files and IPC RTSP video stream.  Main.c  **int main(int argc, char \*argv[])**  **{**  **int count=0,i=0;**  **char config\_file[1024] = {0};**  **count = MAX\_CAMERA\_NUM;**  **if(argc> 1)**  **{**  **sprintf(config\_file, "%s",argv[1]);;**  **}**  **else**  **printf("error params\n ");**  **rgb\_frame\_cache = std::make\_shared<BufManager<cv::Mat> > ();**  **TsMulitGstCamPlayer::GstEnvInit();**  **ST\_CameraConf stCameraConf;**  **for(i=0; i<count; i++)**  **{**  **memset(&stCameraConf, 0, sizeof(stCameraConf));**  **sprintf(stCameraConf.camera\_dic, "decoder\_%d",i);**  **camera\_param\_load(config\_file,&stCameraConf);**  **TsCamera \*pCam = new TsCamera();**  **pCam->SetName(stCameraConf.cameraName);**  **pCam->SetUri(stCameraConf.path/\*"rtsp://admin:thundersoft106@10.0.4.222:554"\*/);**  **if(stCameraConf.rtsp)**  **pCam->BuildPipeLine(true);**  **else**  **pCam->BuildPipeLine(false);**  **pCam->Init(TsMulitGstCamPlayer::GetPipeline());**  **\_listTsCam.push\_back(pCam);**  **camera\_num++;**  **}**  **std::thread yuvconvertThread(thread\_convert);**  **std::thread showThread(thread\_show);**  **showThread.join();**  **while(true) sleep(3);**  **TsMulitGstCamPlayer::GstEnvDeinit();**  **return 0;**  **}**  **Show thread**  **static void thread\_show(void)**  **{**  **double elapsed\_time = 0.0;**  **struct timeval nframerate\_time\_end, time\_start;**  **int m\_nDetectFramecount=0;**  **gettimeofday(&time\_start, nullptr);**  **while(true)**  **{**  **std::shared\_ptr<cv::Mat> imgframe;**  **int num\_w = sqrt(\_listTsCam.size());**  **num\_w = num\_w + (pow(num\_w, 2) < \_listTsCam.size() ? 1 : 0);**  **int num\_h = \_listTsCam.size()/num\_w + (\_listTsCam.size()%num\_w > 0 ? 1 :0);**  **int width = 640;**  **int height = 360;**  **int left = 0,top = 0;**  **imgframe = rgb\_frame\_cache->fetch();**  **if(imgframe == NULL || imgframe.get() == NULL)**  **{**  **usleep(40\*1000);**  **continue;**  **}**  **cv::Mat showframe;**  **cv::resize(\*imgframe, showframe, cv::Size(1920,1080), 0, 0, cv::INTER\_LINEAR);**  **cv::imshow("sink", showframe);**  **cv::waitKey(1);**  **#ifdef DEBUG\_INFO**  **usleep(40000);**  **m\_nDetectFramecount++;**  **gettimeofday(&nframerate\_time\_end, nullptr);**  **elapsed\_time = (nframerate\_time\_end.tv\_sec - time\_start.tv\_sec) \* 1000 +**  **(nframerate\_time\_end.tv\_usec - time\_start.tv\_usec) / 1000;**  **if(elapsed\_time > 1000\*10)**  **{**  **printf("[show] showframerate=%f\n", GetLocalTimeWithMs().c\_str(),**  **m\_nDetectFramecount\*1000/elapsed\_time);**  **memcpy(&time\_start,&nframerate\_time\_end,sizeof(struct timeval));**  **m\_nDetectFramecount = 0;**  **}**  **#endif**  **}**  **}**  **Decoder process in class TsCamera :**  **void TsCamera::Init(GstElement \*pipeline)**  **{**  **GError \*error = NULL;**  **if (!m\_bInited)**  **{**  **printf("[Camera %s]not enable or init str pipeline\n", GetName().c\_str());**  **goto exit;**  **}**  **frame\_cache = std::make\_shared<BufManager<GstSample> > ();**  **m\_nFramecount = 0;**  **m\_nFramerate\_time\_start.tv\_sec = 0;**  **m\_nFramerate\_time\_start.tv\_usec = 0;**  **m\_pPipeline = pipeline;**  **m\_pPipeline = gst\_parse\_launch (m\_strline.c\_str(), &error);**  **if (error != NULL) {**  **printf ("[Camera %s]could not construct pipeline: %s\n", GetName().c\_str(), error->message);**  **g\_clear\_error (&error);**  **goto exit;**  **}**  **/\* get sink \*/**  **m\_pSink = gst\_bin\_get\_by\_name (GST\_BIN (m\_pPipeline), GetName().c\_str());**  **gst\_app\_sink\_set\_emit\_signals((GstAppSink\*)m\_pSink, true);**  **gst\_app\_sink\_set\_drop((GstAppSink\*)m\_pSink, true);**  **gst\_app\_sink\_set\_max\_buffers((GstAppSink\*)m\_pSink, 1);**  **gst\_base\_sink\_set\_last\_sample\_enabled(GST\_BASE\_SINK(m\_pSink), true);**  **gst\_base\_sink\_set\_max\_lateness(GST\_BASE\_SINK(m\_pSink), 0);**  **{//avoid goto check**  **GstAppSinkCallbacks callbacks = { onEOS, onPreroll, onBuffer };**  **gst\_app\_sink\_set\_callbacks (GST\_APP\_SINK(m\_pSink), &callbacks, reinterpret\_cast<void \*>(this), NULL);**  **}**  **/\* Putting a Message handler \*/**  **m\_pBus = gst\_pipeline\_get\_bus (GST\_PIPELINE (m\_pPipeline));**  **gst\_bus\_add\_watch (m\_pBus, MY\_BUS\_CALLBACK, reinterpret\_cast<void \*>(this));**  **gst\_object\_unref (m\_pBus);**  **/\* Run the pipeline \*/**  **printf ("[Camera %s]Playing: %s\n", GetName().c\_str(), GetUri().c\_str());**  **gst\_element\_set\_state (m\_pPipeline, GST\_STATE\_PLAYING);**  **return;**  **exit:**  **if(m\_pSink!=NULL)**  **{**  **}**  **if(m\_pPipeline!=NULL)**  **{**  **gst\_element\_set\_state (m\_pPipeline, GST\_STATE\_NULL);**  **gst\_object\_unref (m\_pPipeline);**  **m\_pPipeline = NULL;**  **}**  **m\_bInited = false;**  **}**  **// onBuffer**  **GstFlowReturn TsCamera::onBuffer(GstAppSink \*appsink, void \*user\_data)**  **{**  **TsCamera \*dec = NULL;**  **GstSample \*sample = NULL;**  **double elapsed\_time = 0.0;**  **struct timeval nframerate\_time\_end;**  **dec = reinterpret\_cast<TsCamera \*>(user\_data);**  **if(dec == NULL || appsink == NULL)**  **{**  **printf ("[Camera %s]decode or appsink is null\n", dec->GetName().c\_str());**  **return GST\_FLOW\_OK;**  **}**  **dec->gst\_pull\_block();**  **if(!dec->m\_nFramecount)gettimeofday(&dec->m\_nFramerate\_time\_start, nullptr);**  **dec->m\_nFramecount++;**  **sample = gst\_base\_sink\_get\_last\_sample(GST\_BASE\_SINK(appsink));**  **if(sample != NULL)**  **{**  **dec->frame\_cache->feed(std::shared\_ptr<GstSample>(sample, deleterGstSample));**  **}**  **gettimeofday(&nframerate\_time\_end, nullptr);**  **elapsed\_time = (nframerate\_time\_end.tv\_sec - dec->m\_nFramerate\_time\_start.tv\_sec) \* 1000 +**  **(nframerate\_time\_end.tv\_usec - dec->m\_nFramerate\_time\_start.tv\_usec) / 1000;**  **if(elapsed\_time > 1000)**  **{**  **if(dec->m\_nFramecount\*1000/elapsed\_time < m\_framerate)**  **printf("[Camera %s %s]TVMF framerate=%f\n", dec->GetName().c\_str(), GetLocalTimeWithMs().c\_str(),**  **dec->m\_nFramecount\*1000/elapsed\_time);**  **dec->m\_nFramecount = 0;**  **}**  **return GST\_FLOW\_OK;**  **}**  **void TsCamera::BuildPipeLine(bool rtsp)**  **{**  **std::ostringstream cameraPath;**  **if(rtsp)**  **{**  **cameraPath << "rtspsrc location=" << GetUri() << " latency=0 tcp-timeout=500 drop-on-latency=true ntp-sync=true" << " ! ";**  **cameraPath << "queue ! rtp" << "h264" << "depay ! "<< "h264" << "parse ! queue ! qtivdec ! ";**  **cameraPath << "videoscale ! video/x-raw,width=640,height=360 ! appsink name=" << GetName() << " sync=false max-lateness=0 max-buffers=1 drop=true";**  **}**  **else**  **{**  **cameraPath << "filesrc location=" << GetUri() << " " << " ! ";**  **cameraPath << "qtdemux name=demux demux. ! queue ! h264parse ! qtivdec !" ;**  **cameraPath << " videoscale ! video/x-raw,width=640,height=360 ! appsink name=" << GetName() << " sync=false max-lateness=0 max-buffers=1 drop=true";**  **}**  **m\_strline = cameraPath.str();**  **printf("[Camera %s]gstreamer decoder pipeline string:%s\n", GetName().c\_str(), m\_strline.c\_str());**  **m\_bInited = true;**  **}** | |
| **Usage Instructions** | Some performance log：  Cmd top d1 tto show cpu usage  1   1. Decoding frame rate of demo peride ten seconds.   25FPS  3、GPU usage.  GPU  The display effect is as follows：    The result of performance:   1. CPU.   cpu   1. GPU.   gpu   1. Frame Rate.   framereate | |
| **Contributor(s) Info**  *Feel free to include headshots!* | Name | Title  Company |
|  |  |
|  |  |
|  |  |

––– Continued on next page –––

Filters and Tags for QDN projects page

|  |  |  |
| --- | --- | --- |
| **Platform/Hardware** | CSR 101x/102x Bluetooth  DragonBoard 410c  mangOH Red/Yellow | MDM920x LTE for IoT  QCA-402x WiFi/BLE/Zigbee  √     Qualcomm Robotics RBx Dev Kit |
| **Software Tools** | 3D Audio Plugin for Unity  Adreno GPU SDK  Hexagon DSP SDK | Neural Processing SDK for AI  √ Snapdragon Profiler |
| **Operating System** | √ Android  Linux  ThreadX RTOS | Ubuntu Core  Windows 10 IoT Core |
| **Cloud Services/Platform** | Sierra Wireless AirVantage  Gizwits Cloud Platform  AT&T M2X  IBM Bluemix | IBM Watson IoT  Microsoft Azure IoT  Amazon AWS IoT |
| **Skill Level Required** | √ Advanced  Beginner  Intermediate |  |
| **Areas of Focus** | 3D Printing & Modeling  Alexa Voice Service  Artificial Intelligence  Bluetooth  Computer Vision  Digital Signage  √ Education  √ Embedded  Gaming | √ Healthcare  √ IoT  √ Robotics  √ Security  Sensors  √ Smart Cities  √ Smart Home  √ Toys |

*By submitting your content (“Submission”), you are granting Qualcomm a royalty-free, perpetual, non-exclusive, unrestricted, worldwide license to: (a) post, use, copy, sublicense, adapt, transmit, publicly perform or display any such Submission, (b) use, reproduce, modify, adapt, publish, translate, create derivative works from, distribute, perform, play, host, communicate, make available and publish your Submission without restriction and (c) sublicense to third parties the unrestricted right to exercise any of the foregoing rights granted with respect to the Submission. The foregoing grants shall include the right to exploit any ideas, concepts, intellectual property, or proprietary rights in such Submission, including but not limited to rights under copyright, trademark, servicemark or patent laws under any relevant jurisdiction without Qualcomm owing any monies to you whatsoever. You represent and warrant that you own all right, title and interest in and to the Submission, or you have been granted sufficient rights in and to the Submission allowing the foregoing use of such Submission.*