**Qualcomm® QCS610 SOC Open Kit HelmetDetect-demo Developer documentation**

1. **Introduce**

This project relies on the QCS610 development kit, using the AI computing power and image processing capabilities of the development kit to collect images in real time,perform AI reasoning after preprocessing, and output the reasoning results. I use gstreamer and TFLITE to complete the above functions. When the pipeline obtains valid image data, it will automatically collect the image, perform AI inference, and return the position information for the helmet in the image frame.

At present, ARTIFICIAL intelligence has entered many industries requiring worker safety supervision, such as construction sites and open-air factories, and helmets are one of the necessary safety devices for this mode of operation. Therefore, I want to implement a demo to complete helmet detection on embedded devices.

The project was built in x86 host with across complier tool and has been tested in Qualcomm® QCS610 Open Kits.

Qualcomm® QCS610 SOC Open Kits

****

1. **Materials and Tools used for the project**

**2.1 Other Hardware materials**

Except for the Open Kits, the following hardware materials are also needed:

**2.1.1 Type-C USB line**

Using the USB line to develop on Qualcomm® QCS610 SOC Open Kits.



**2.1.2 Charger**

Direct power supply for Qualcomm® QCS610 SOC Open Kits.



**2.1.3 DP-Line**

Using the universal DP line to connect LED displayer to Qualcomm® QCS610 SOC Open Kits.



**2.1.4 LCD-Displayer**

Using a LED Displayer to display demo interface from Qualcomm® QCS610 SOC Open Kits.



1. **Environment configuration**

This section mainly describes the source and configuration of some open source projects or third-party tools needed in the development process.

* 1. **Snapdragon Beural Processing Engine Sdk**

1. Download Snapdragon Neural Processing Engine SDK(version == 1.37.0)

<https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/tools>

1. Enter include/zdl/ in Snapdragon Neural Processing Engine SDK root directory,copy all dir to demo project\_root\_dir/inference/snpe/
   1. **Qualcomm FastCV**
2. Download Qualcomm FastCV-Linux

https://developer.qualcomm.com/software/fastcv-sdk

2. Get fastcv.h header file and copy to project\_root\_dir/gst\_pipe/

* 1. **OpenCV**

The version of OpenCV is 3.4.3.There are many tutorials for opencv installation on the network.Therefore, you can completely refer to other people's installation methods for installation and configuration.For example:

[https://www.cnblogs.com/uestc-mm/p/7338244.html](https://www.cnblogs.com/uestc-mm/p/7338244.html" \o "https://www.cnblogs.com/uestc-mm/p/7338244.html)

* 1. **google Protobuf**

1. Download google protobuf (version == 3.3)

https://github.com/protocolbuffers/protobuf

2. Enter protobuf/src/google/ dir.Then,select all .h or .hpp .etc header file with maintain the structure of directory

3. copy header file dir to demo project\_root\_dir/pb\_conf

1. **Compile**

The compilation of the whole project is based on the yocto compilation tool, so you need to write some .bb and .conf files according to the specification. The file is [helmet\_detect\_0.1.bb](https://github.com/ThunderSoft-XA/C610-DriverWarn-demo/blob/master/C610-DriverWarn-demo/driver_warn_1.0.bb" \o "https://github.com/ThunderSoft-XA/C610-DriverWarn-demo/blob/master/C610-DriverWarn-demo/driver_warn_1.0.bb) example.

Please refer to [the official Manual of Yocto](https://www.yoctoproject.org) for how to add layers, write layer.conf. Then, execute the command as follows:

bitbake helmet\_detect

　　You will get an executable bin file named HelmetDetect.Move it to the root of the source code. Next, push the whole project code to Qualcomm® QCS610 open kit device`s directory /data/.

$ adb root && adb disable-verity && adb reboot

$ adb root &&adb remount && adb shell mount -o remount,rw /

$ adb push xxx/xxx/sourcepath /data

$ adb shell

$ cd data/HelmetDetect

1. **Configure Weston and Usage**

**5.1 Configure file explanation**

gstreamer pipeline google protobuf config:

gstid : 0 //gstreamer pipeline ID

gstname : “gst\_zero” //gstreamer pipeline name

sinkname : gst\_sink //gstreamer pipeline appsink Plug-in name

gsttype : CAMERA // gstreamer pipeline type ,CAMERA/RTSP/LOCAL

path : 1 // gstreamer pipeline source sink path ,rtsp = url,camera=camera id,local file = local video file path

decode : h264 // gstreamer pipeline decode type

framerate : 30

format : NV12

width : 640

height : 360

hw\_dec : 1 //gstreamer pipeline hardware decode or software decode

need\_calib : True //image frame need calibration or not, 1 = need

ai inference google protobuf config:

model\_path : "/data/HelmetDetect/models/helmet/output\_tflite\_graph\_helmet.tflite" // model path

runtime : GPU // ai inference runtime : CPU/GPU/DSP/NNAPI/APU

data\_format : NCHW //ai inference data format NCHW/NHWC

**5.2 Usage**

This project only provides a simple command line interface.

Run HelmetDetect(This is the project name, which can be modified in the CMakeListe.txt file in the root directory of the project )：

Run the HelmetDetect demo.This test demo has only command line output.

$ cd /data/<source root dir>

$ ./weston\_dp\_cmd helmet

Notice:

DP display uses weston graphical interface support, so I provide weston\_dp\_cmd script in the project root directory