**Qualcomm® QCS610 SOC Open Kit DriverWarn-demo2 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 (support NNAPI) to complete the above functions. The front camera collects facial images to detect the driver's fatigue state, and the rear camera is used to display road conditions outside the vehicle.

The current deployment of deep learning is extremely hot, so I think it is necessary to deploy deep learning models in embedded devices. In life, fatigue driving causes a lot of traffic accidents, so I thought of this small project of driving warning.

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

Qualcomm® QCS610 SOC Development board

****

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

**2.1 Other Hardware materials**

Except for the development board, the following hardware materials are also needed:

**2.1.1 Type-C USB line**

Using the USB line to develop on Qualcomm® QCS610 SOC development board.



**2.1.2 Charger**

Direct power supply for Qualcomm® QCS610 SOC development board.



**2.1.3 DP-Line**

Using the universal DP line to connect LED displayer to Qualcomm® QCS610 SOC development board.



**2.1.4 LED-Displayer**

Using a LED Displayer to display the objectTracker-demo interface from Qualcomm® QCS610 SOC development board.



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. **Tensorflow**

1. Download Download Tensoflow source code(version > 2.0):

https://github.com/tensorflow/tensorflow

1. build Tensorflow for support TFLite runtime in your QCS610 open kit.
2. Git clone this project link:

https://github.com/ThunderSoft-XA/C610-DriverWarn-demo2/

1. build the project after solve other dependencies.

Notice: If the system on your development kit already supports TFLite runtime, you can skip step 1 and step 2.

* 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)

Notice:

This project uses the opencv native face detection model, so you need to download a haarcascade\_frontalface\_default.xml.

Or,You can go to your python3 directory to get it,as follow:

xxx/xxx/python3.x/site-packages/cv2/data/haarcascade\_frontalface\_default.xml

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 link is [driver\_warn\_1.0.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 DriverWarn

　　You will get an executable bin file named gst\_test.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/DriverWarn-demo2

1. **Configure Weston and Usage**

**5.1 Configure file explanation**

gstreamer pipeline config:

[conf\_info]

conf\_count = 1 //parse configure total number in current file

[gst\_0]

gstid=0 //gstreamer pipeline ID

gstname=gst\_zero //gstreamer pipeline name

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

gsttype=1 // gstreamer pipeline type ,0 = rtsp ,1 = camera, 2 = local file

enable=on

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

calibration=1 //image frame need calibration or not, 1 = need

AI model config:

[conf\_info]

conf\_count=3 //parse configure total number in current file

[ai\_thread\_0]

ai\_id=0

ai\_name=face\_landmark

data\_source=gst\_zero //AI inference data from anywhere gstreamer pipeline

model\_path=./../models/face\_landmark.tflite

input\_width=192

input\_height=192

channel=3

delegate=1 //use tflite runtime delegate,0 = CPU,1 = NNAPI, 2 = GPU,3 = DSP

input\_mean=127.5

std\_mean=127.5 //model mean value for Calculation output range

input\_layer\_type=uint\_8

max\_profiling\_buffer\_entries=1024 //Not implemented yet

number\_of\_warmup\_runs=2 //Not implemented yet

**5.2 Usage**

This project only provides a simple command line interface.

Run DriverWarn(This is the project name, the actual executable file name is gst\_test, which can be modified in the CMakeListe.txt file in the root directory of the project )：

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

$ cd /data/<source root dir> && mkdir build

$ cp gst\_test ./build && cd build

$ ./gst\_test

Notice:

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