Remote Access and Output Sharing Between Multiple ECUs for Automotive

20/6/2018
Harunobu KUROKAWA

Open system development center, Automotive information solution business division
Renesas Electronics Corporation
Today’s Topics

1. Introduction
2. Multi display and Output sharing
   ➢ Explain Current AGL demo system
3. Usage and customize
4. Current work/status and Future work
   ➢ Update output sharing for next AGL
   ➢ Upstreaming “weston remote access plugin” to community
5. Conclusion
1. Introduction
Who am I?

Name: Harunobu Kurokawa

- Working in Renesas Electronics Corporation, for over 10 years.
  - 2007 – 2013: Mobile platform software development and support.

Experience / Mission for AGL™.

- Support Renesas BSP (2017~)
- AGL Demo integration set-up staff (2017~)
- Gatekeeper (2018 ~)
Renesas R-Car and Open Source Linux

- Renesas provide Yocto based BSP to customer and community
Renesas R-Car and AGL

- Renesas R-Car series were selected as Referenced Board from 2015
2. Multi Display and Output Sharing
Multiple display for infotainment

- In the future requirement, multiple display and share multimedia information.
Example for Multi display

- R-Car reference board Kingfisher has 3 display output (2 HDMI and 1 LVDS)
- One weston controls and manages 3 outputs
- Need high performance and bandwidth in one SoC.

1080x1920
IVI

1280x720
Video play

1280x480
Cluster

Kingfisher Board

application

weston

Output Buffer

Output Buffer

Display

Display
Example for Multi display (AGL CES2018)

- AGL demo use two different board, Renesas (R-Car) and Intel board (minnow)
- Each board has weston.
- Need communicate between ECUs for sharing IVI information (e.g. navigation/map etc.)
Output sharing between ECU via Ethernet

- IVI side transfers image data using H.264 stream (gst-recorder).
  gst-recorder system has Gstreamer feature in Weston.

Receiver gets stream and decode it.

IVI application

Weston core

HDMI Display (Navigation)

Virtual Display (map-view)

EncodeH.264

UDP

Decode

Receiver

Wayland

ECU Cluster side

Wayland

Weston core

Gstreamer

Transfer module

ECU IVI side
Gst-recorder s/w diagram

- Create “Virtual output Display”.
  - Append “Virtual output” structure as wl_output.
- Encode Virtual output buffer.
  - Read from output Buffer and convert NV12 format. The buffer is read from Encoder HW.
  - 2D-HW can support Cropping(size, position).
- Transfer Encoded buffer
  - Buffer is transferred via Ethernet by udpsink.
Pros and Cons

◆ Pros

✓ No modification required to Application. No depend on shell protocol (xdg, ivi).
✓ No modification required to Weston on receiver side.
✓ Video encoding minimizes network data and bandwidth
✓ Virtual display means display size can be made only as large as needed.
✓ No limitation of number of output sharing

◆ Cons

✓ Delay of about 2-3 vsync to encode, transmit, receive, decode and composite
✓ Parameters must be fixed at boot-up.
3. Usage and customize
Build and Setup in AGL 5.0.3

◆ Build

- repo and setup : same as usual setup
  $ source meta-agl/scripts/aglsetup.sh -m m3ulcb -b build agl-devel agl-demo agl-audio-4a-framework

- Add DISTRO_FEATURES in build/conf/local.conf
  
  ```
  DISTRO_FEATURES_append = " virtual-display gst-record agl-mapviewer-demo"
  ```

  $ bitbake agl-demo-platform

  Update weston.ini file and weston.service (systemd) file
## Parameters in weston.ini

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
<th>sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>[core]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>virtual</td>
<td>int</td>
<td>Number of created output</td>
<td>virtual=1</td>
</tr>
<tr>
<td>[output]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Output name: virtual1, virtual2, …</td>
<td>name=virtual1</td>
</tr>
<tr>
<td>mode</td>
<td>string</td>
<td>Output size and fps. width x height @ fps</td>
<td>mode=800x480@60</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>IP address for receiver side</td>
<td>ip=192.168.20.99</td>
</tr>
<tr>
<td>port</td>
<td>int</td>
<td>Port number</td>
<td>port=5005</td>
</tr>
<tr>
<td>bitrate</td>
<td>int</td>
<td>Bitrate for encoding</td>
<td>bitrate=100000</td>
</tr>
<tr>
<td>recoder</td>
<td>bool</td>
<td>Enable flag</td>
<td>recoder=true</td>
</tr>
<tr>
<td>crop</td>
<td>string</td>
<td>(option) : set cropping rectangle width/height and position x/y</td>
<td>crop=384x386@0x0</td>
</tr>
</tbody>
</table>
**Board setup example**

Sender side

1) Set modified weston.ini
2) Start weston process by systemd.
   Then, add "--gst-record" argument in weston.service.

Receiver side

1) use default weston.ini.
2) Run gst-launch

```
$ gst-launch-1.0 udpsrc port=5005 ! application/x-rtp,media=video,clock-rate=90000,encoding-name=H264 !
   rtpjitterbuffer latency=0 ! h264parse ! omxh264dec no-reorder=true ! waylandsink
```

**<weston.ini file>**

```
[core]
virtual=1

<snip>

[output]
name=virtual1
mode=384x368@60
ip=192.168.20.99
port=5005
bitrate=100000
recorder=true

<weston.service>

ExecStart=/usr/bin/weston --idle-time=0 --tty=1 --gst-record
```

IP : 192.168.20.93
IP : 192.168.20.99
Demo

Sender side

1) Run gst-launch

$ gst-launch-1.0 udpsrc port=5006 ! application/x-rtp,media=video,clock-rate=90000,encoding-name=H264 ! rtph264depay ! h264parse config-interval=1 disable-passthrough=true ! decodebin ! autoimagesink
4. Current status and Future work
Current work/status

1. Remove R-Car HW depend code : on going.
   1. Test on AGL reference board except R-Car
   2. Push to AGL gerrit. ( I want it will merge to AGL 6.0)

2. Upstreaming to Wayland community.
   1. Restructure Plugin APIs
   2. Optimize the performance (zero-copy)
Current work: Remove R-Car specific code

- Independ on R-Car HW specification. Remove v4l2 API and omx element.

Prototype: Done
- Need to test before submit to AGL
Future work: Upstreaming and Optimization

- **Upstreaming**
  - Restructure to weston/4.0 or later master branch.

- **Optimization**
  - Replace zero-copy and measure CPU load vsync count.
  - Investigate and apply sync mechanism.

- **Handling Input event from receiver**
  - Need to apply Waltham protocol.
Conclusion
Conclusion

▪ AGL already supports Output Sharing for Multi Display
  – Renesas provided “gst-recorder” plugin for AGL
  – “gst-recoder” makes remote access feature in Automotive
▪ Update gst-recorder for multi-platform

▪ Future work
  – Develop new plugin for upstreaming, and optimization.
Thank you

harunobu.kurokawa.dn@renesas.com

• Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
• AGL™ is the registered trademark of The Linux Foundation in the United States and/or other countries.
• GENIVI® is a registered trademark of the GENIVI Alliance in the USA and other countries.
• All names of other products or services mentioned in this press release are trademarks or registered