AM6x Edge AI and Video Stream over Wi-Fi® Utilizing CC33XX

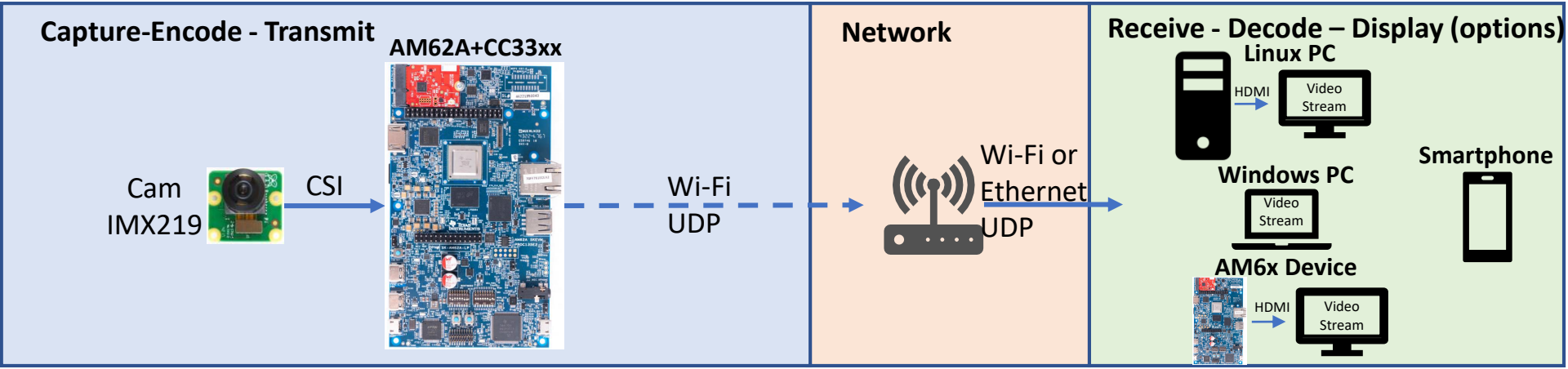
Qutaiba Saleh

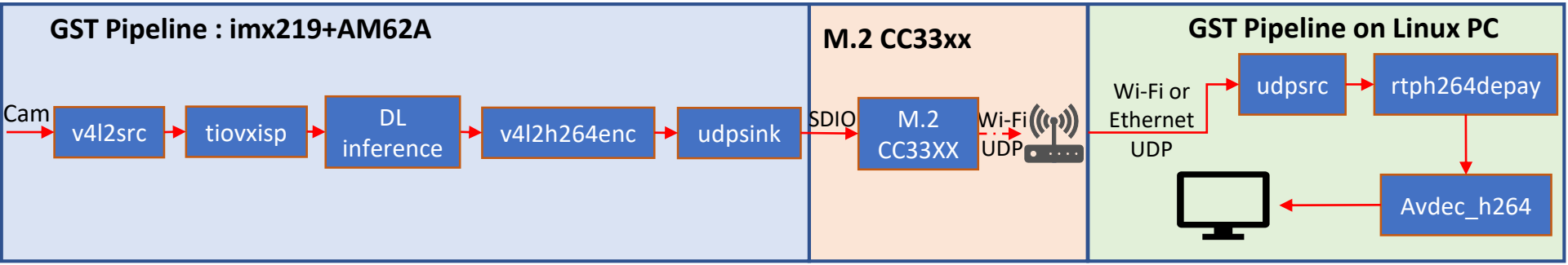
Aug/2024

Summary

- Demonstrates using AM6xA to capture live video, perform ML inference, and stream video over Wi-Fi.
- The streamed video is encode video with H.264/H.265
- UDP stream over Wi-Fi using CC33xx.
- At the receiver side, the video is decoded and displayed on a screen.
- The receiver side could be a Linux PC, Windows PC, smartphone, or another AM6x device.
- Implementation:
  - For demonstration, a gStreamer is used to connect the various blocks of the pipeline.
  - This example uses imx219 camera with resolution 1920x1080 @ 30 FPS. For a list of other sensors supported on AM62A: https://dev.ti.com/tirex/content/am62ax_academy_9_02_00_00_v1/_build_am62ax_academy_9_02_00_00_v1/source/linux/ch-eval/eval-tour-camera.html#tested-csi-2-cameras
  - AM62A captures video stream from imx219 camera processes it with ISP, encode with H.264, and transmits it over Wi-Fi with CC33xx M.2 module.
  - A Linux PC or smartphone are used as receivers. Decode the video stream and display on screen.

Pipeline

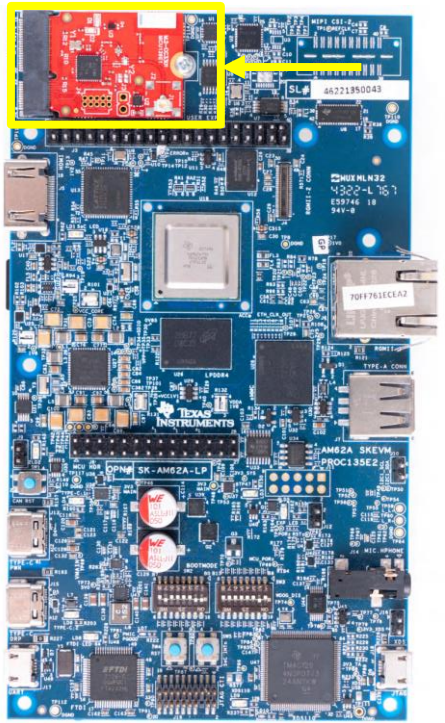




Implementation (Hardware)

- M.2 CC33xx:
  - Remove the screw from the AM62A EVM board to make way for the M.2 CC33xx card.
  - Insert the M.2 CC33xx card into the M.2 expansion slot.
  - Horizontally push the card and secure it with hand.
  - Secure the M.2 CC33xx card with the screw.
  - More information about M.2 CC33xx are in the User's Guide: https://www.ti.com/lit/ug/swau131a/swau131a.pdf
- AM62A EVM:
  - Prepare the AM62A EVM with SDK 10.0. Follow the steps in the AM62A Quick Start Guide: https://dev.ti.com/tirex/content/tirex-product-tree/am62ax-devtools/docs/am62ax_skevm_quick_start_guide.html
  - Linux SDK 10.0: https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-AM62A

CC33xx board



Implementation (Device Tree)

- Detailed instructions about enabling the M.2 CC33xx are available in the 10.0 SDK documentation: https://software-dl.ti.com/processor-sdk-linux/esd/AM62AX/10_00_00/exports/docs/linux/How_to_Guides/Target/How_To_Enable_M2CC3301_in_linux.html
- Add the appropriate dtbo overlay to the /run/media/BOOT-mmcblk1p1/uEnv.txt
- The imx219 camera is supported by default in the AM62A SDK. The only one addition is to include a dtbo overlay. For more details about setting imx219 see: https://dev.ti.com/tirex/content/am62ax_academy_9_02_00_00_v1/_build_am62ax_academy_9_02_00_00_v1/source/linux/ch-eval/eval-tour-camera.html
- Set the uEnv.txt file as following: `name_overlays=ti/k3-am62a7-sk-m2-cc3301.dtbo ti/k3-am62x-sk-csi2-imx219.dtbo`
- Reboot AM62A

```
root@am62axx-evm:/# cat /run/media/BOOT-mmcblk1p1/uEnv.txt
# This uEnv.txt file can contain additional environment settings that you
# want to set in U-Boot at boot time.  This can be simple variables such
# as the serverip or custom variables.  The format of this file is:
#    variable=value
# NOTE: This file will be evaluated after the bootcmd is run and the
#       bootcmd must be set to load this file if it exists (this is the
#       default on all newer U-Boot images.  This also means that some
#       variables such as bootdelay cannot be changed by this file since
#       it is not evaluated until the bootcmd is run.
# Setting the right U-Boot environment variables
dorprocboot=1
name_overlays=ti/k3-am62a7-sk-m2-cc3301.dtbo ti/k3-am62x-sk-csi2-imx219.dtbo
```

Implementation (Connect to Router)

- With the dtbo setup properly, use ifconfig to check the available network.
- wlan0: should show automatically.
- To connect to a Wi-Fi network, follow the steps:

```
root@am62axx-evm:/# cd /usr/share/cc33xx
./sta_start.sh
./sta_connect-ex.sh <SSID> WPA-PSK <PASSWORD>
udhcpc -i wlan0
```

```
root@am62axx-evm:/# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 1c:63:49:17:20:71  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 19  bytes 2253 (2.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 2253 (2.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.5  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::a634:f1ff:feb2:7110  prefixlen 64  scopeid 0x20<link>
        ether a4:34:f1:b2:71:10  txqueuelen 1000  (Ethernet)
        RX packets 92  bytes 9503 (9.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 126  bytes 15184 (14.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Implementation: gStreamer Pipeline

Video Capture and transmit (AM62A)

```
gst-launch-1.0 -v \
v4l2src device=/dev/video-imx219-cam0 io-mode=5 ! queue leaky=2 ! video/x-bayer, width=1920, height=1080, format=rggb ! \
tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-isp-file=/opt/imaging/imx219/linear/dcc_viss.bin format-msb=7 \
sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a.bin sink_0::device=/dev/v4l-subdev2 ! \
video/x-raw, format=NV12 ! queue ! tiperfoverlay title="IMx219 to udp stream" overlay-type=graph !\
v4l2h264enc ! h264parse ! rtph264pay ! udpsink host=<receiver ip> port=5000 sync=false \
```

Video Capture, ML inference and transmit (AM62A)

```
gst-launch-1.0 -v \
v4l2src device=/dev/ /video-imx219-cam0 io-mode=5 ! queue leaky=2 ! video/x-bayer, width=1920, height=1080, format=rggb ! tiovxisp sensor-name=SENSOR_SONY_IMX219_RPI dcc-
isp-file=/opt/imaging/imx219/linear/dcc_viss.bin format-msb=7 sink_0::dcc-2a-file=/opt/imaging/imx219/linear/dcc_2a.bin sink_0::device=/dev/v4l-subdev2 ! video/x-raw, format=NV12 ! \
tiovxmultiscaler name=split_01 src_0::roi-startx=0 src_0::roi-starty=0 src_0::roi-width=1920 src_0::roi-height=1080 target=0 \
split_01. ! queue ! video/x-raw, width=480, height=416 ! tiovxmultiscaler target=1 ! video/x-raw, width=416, height=416 ! tiovxdlpreproc model=/opt/model_zoo/ONR-OD-8200-yolox-nano-
lite-mmdet-coco-416x416  out-pool-size=4 ! application/x-tensor-tiovx ! tidlinferer target=1  model=/opt/model_zoo/ONR-OD-8200-yolox-nano-lite-mmdet-coco-416x416 ! post_0.tensor \
split_01. ! queue ! video/x-raw, width=1280, height=720 ! post_0.sink \
tidlpostproc name=post_0 model=/opt/model_zoo/ONR-OD-8200-yolox-nano-lite-mmdet-coco-416x416 alpha=0.200000 viz-threshold=0.600000 top-N=5 display-model=true ! \
queue ! tiperfoverlay title="Object Detection" ! v4l2h264enc ! h264parse ! rtph264pay ! udpsink host=<receiver ip>port=5000 sync=false \
```

Receiver: Linux PC

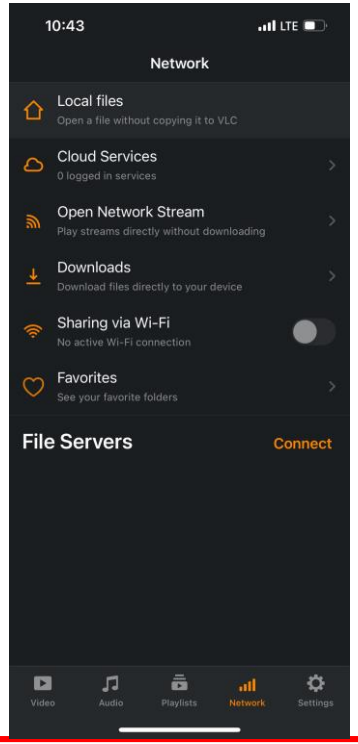
```
gst-launch-1.0 -v \
udpsrc port=5000 ! 'application/x-rtp, encoding-name=H264, payload=96' ! rtph264depay ! avdec_h264  !  videoconvert ! autovideosink
```

Implementation: Receive on smartphone and Windows using VLC media player

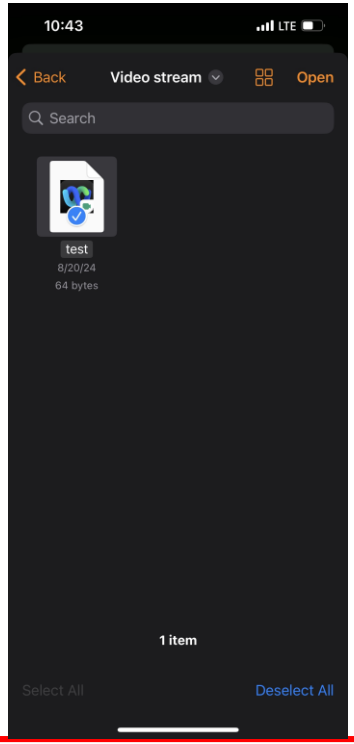
Create test.sdp simple text

```
c=IN IP4 0.0.0.0
m=video 5000 RTP/AVP 96
a=rtpmap:96 H264/90000
```

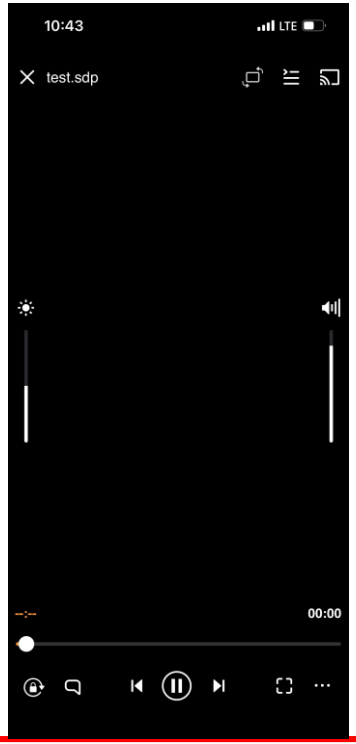
VLC: Network Tab Local files



Navigate to /Open test.sdp



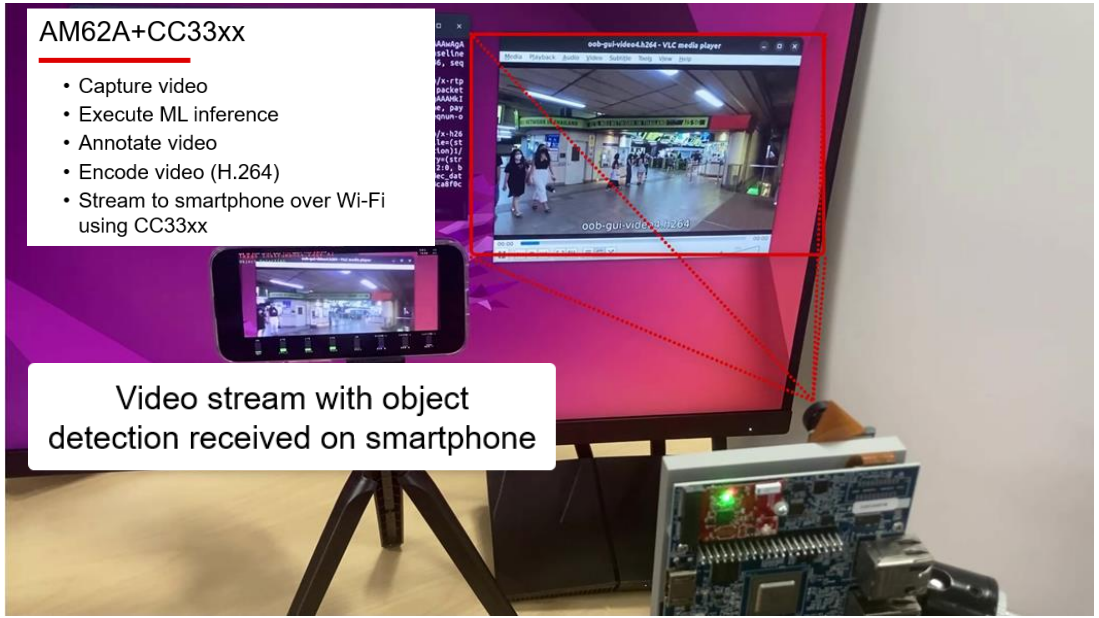
Waiting for stream



Stream Received

Results

- The videos are successfully captured, encoded (H.264), and streamed over Wi-Fi using CC33xx.
- The stream can be received/displayed using Linux PC, Windows PC, smartphone, and another AM6x device.
- The videos shows streams from AM62A to a smartphone.
- Camera: imx219 with 1920x1080 @ 30 FPS.
- Table below shows AM62A resource utilization.



ML Model	Rate each [FPS]	CPU (4x A53 @ 1.2 GHz) [%]	DLA (C7x-MMA) @ 850 MHz [%]	VISS [%]	MSC0 [%]	MSC1 [%]	DDR Rd [MB/s]	DDR Wr [MB/s]	DDR Total [MB/s]
NA	30	7.86	0	17.3	0	0	860	309	1169
Yolox-nano-lite	30	10.61	28	20.38	28.27	3.3	1972	400	1772