Smart Model Select (avnet-kv260-vvas-sms) - Getting Started Guide

Overview

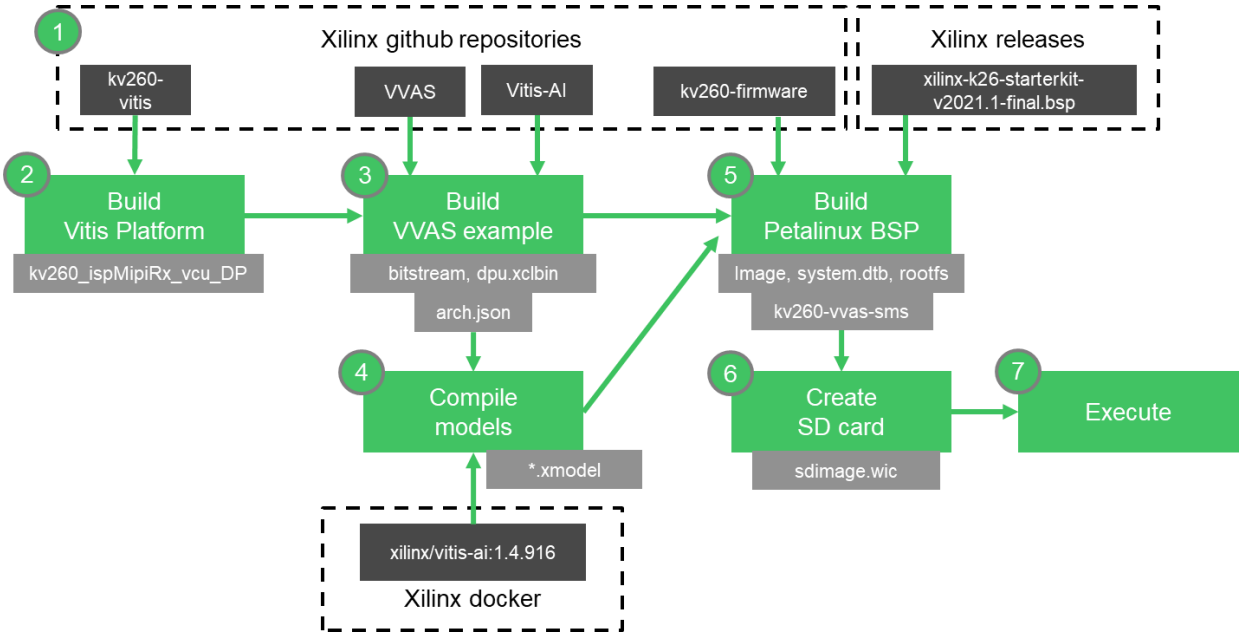
Avnet has gone through the exercise of creating a Kria app, for the purpose of documenting the development flow. Note that there is no intention of licensing of monetizing this app, since it is for demonstration purposes only.

Disclaimer

Development Flow

A complete development flow has been published on Element 14

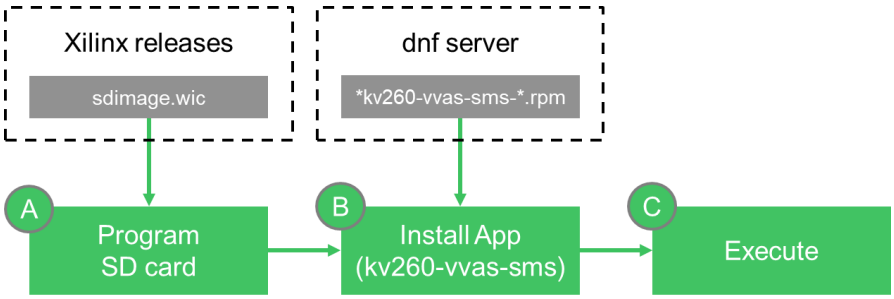
http://avnet.me/kv260-vvas-sms-2021-1-blog



Note that the currently published development flow does not yet take into account the rpm package naming enforced by Accelize. This will updated in the future, once the app is available on the Xilinx app store.

Deployment Flow

The custom Kria app has been made available as a third party app via the Accelize repository.



The following rpm packages are currently uploaded to the Accelize staging repo:

- avnet-kv260-vvas-sms
- avnet-kv260-vvas-sms-models
- avnet-kv260-vvas-sms-app
- packagegroup-kv260-vvas-sms

Setting up the Hardware

This section describes how to setup the KV260 Vision AI Starter Kit for use with the app.

Hardware Requirements

- Kria KV260 Vision AI Starter Kit
- Kria KV260 Basic Accessory Pack
    - KV260 Power Supply & Adapter
    - USB-A to micro-B Cable
    - RJ-45 Ethernet Cable
    - HDMI Cable
    - 16GB microSD Card
    - [Optional] IAS AR1335 sensor module
- [Optional] Logitech BRIO USB camera
- 1080P capable HDMI monitor

Programming the SD card

The first step is to program the 2021.1 version of the SD card image.

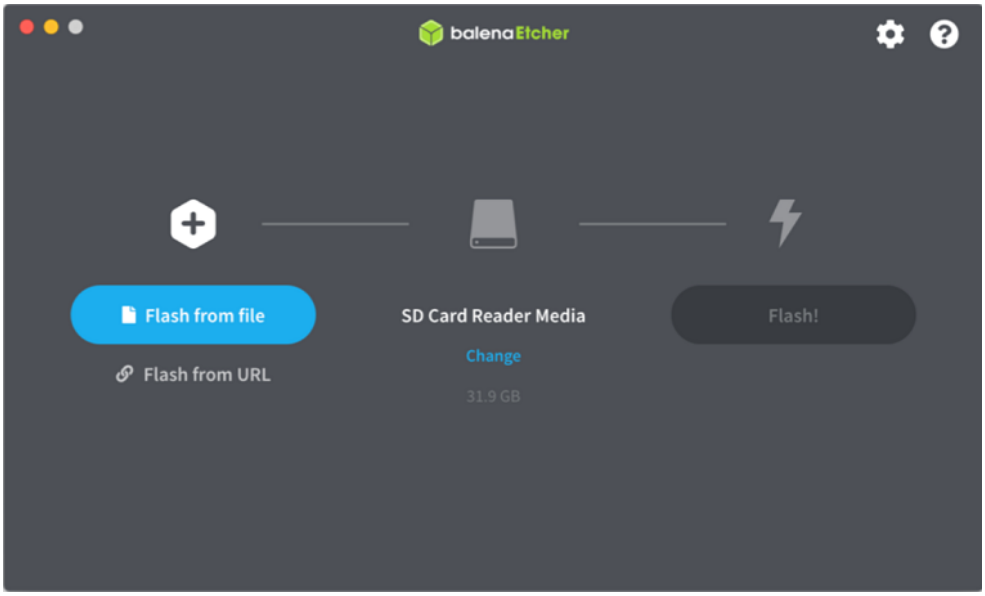
The SD card image for the KV260 can be downloaded from Xilinx's KV260 wiki pages:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/1641152513/Kria+K26+SOM#SD-Card-Images

Perform the following steps to boot your KV260 with the 2021.1 SD card image :

- download SD card image (petalinux-sdimage-2021.1-update1.wic.xz)

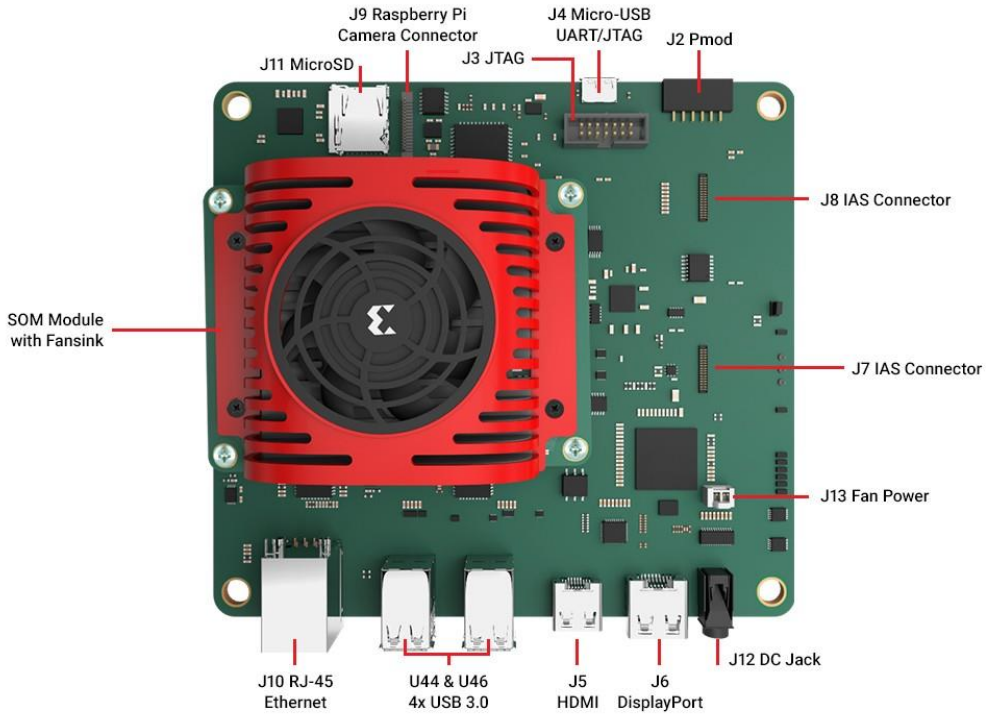
- program the image to a 16GB microSD card using Balena Etcher



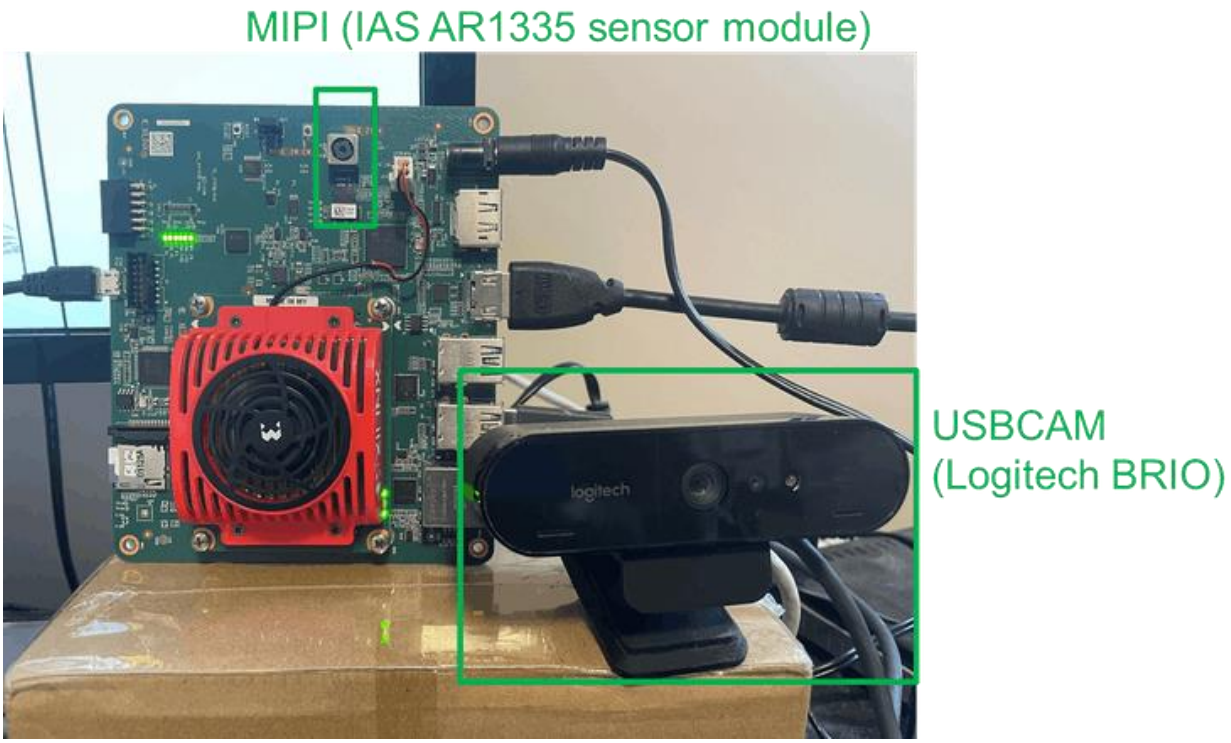
Connecting up the Hardware

The following steps describe how to connect up the KV260 Vision AI Starter Kit for use with the app.

1. Insert the microSD card into the slot on the KV260 carrier card (J11)
2. Connect a micro-USB cable to J4 on the KV260 carrier card. This USB serial interface will be leveraged to interact with the application and view boot time messages
3. Connect an RJ-45 Ethernet cable from J10 on the KV260 carrier card to your local network
4. Connect a DisplayPort cable and DisplayPort monitor to J6 (or HDMI to J5) on the KV260 carrier card
5. [Optional] Connect the IAS AR1335 sensor module to IAS1 (J8) on the KV260 carrier card
6. [Optional] Connect the Logitech BRIO USB camera to one of the USB 3.0 connectors (U44, U46) on the KV260 carrier card
7. Power on the board



The following image illustrates the use of the optional IAS AR1335 sensor module and Logitech BRIO USB camera.



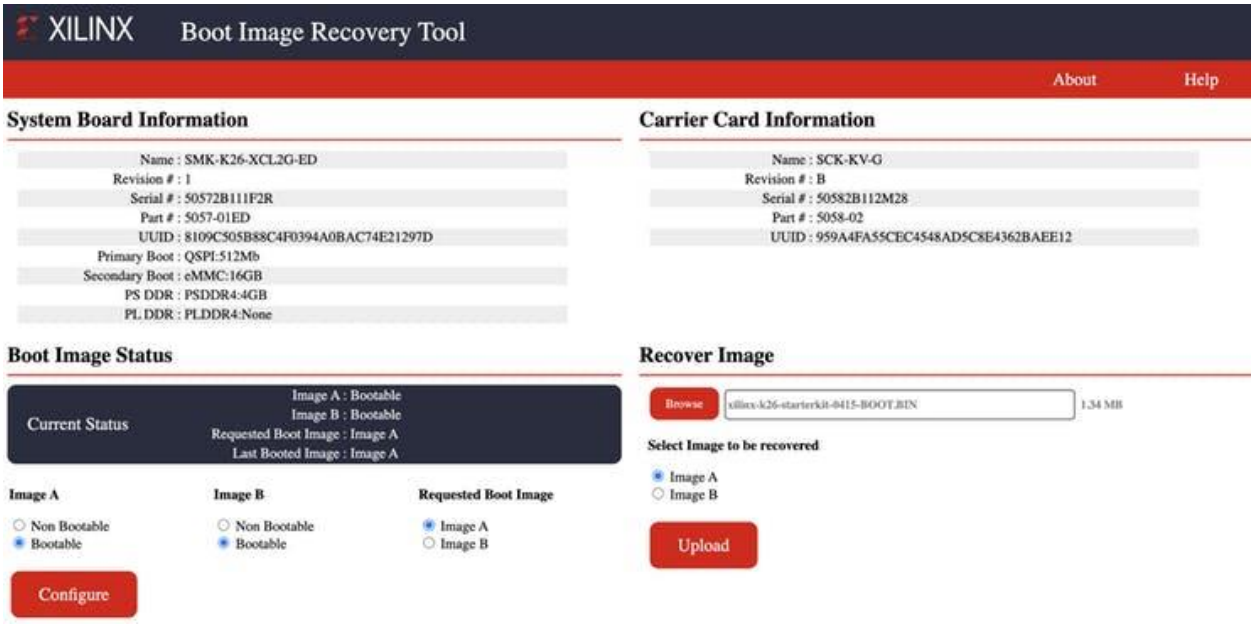
If the 2021.1 SD card image boots successfully, skip the next section.

Otherwise, you probably need to update the KV26 SOM's boot firmware for the 2021.1 image.

[Optional] Updating the KV260 Firmware

If the SD card image does not boot, you may need to update the KV26 boot firmware (BOOT.BIN), as described in the following wiki page:

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/1641152513/Kria+K26+SOM#Stand-alone-FW-Update-&-Recovery-Utility



Run-time Installation

After first boot, you must login as “petalinux” user, and specify your password (twice).

Next, the following components need to be installed on the system:

- avnet-kv260-vvas-sms

Before installing the app, the ext4 partition needs to be extended with the following commands:

```
# sudo parted /dev/mmcblk1 resizepart 2

Warning: Partition /dev/mmcblk1p2 is being used. Are you sure you want
to continue?
```

```
Yes/No? yes
End?  [6442MB]? 100%
Information: You may need to update /etc/fstab.

# sudo resize2fs /dev/mmcblk1p2

resize2fs 1.45.6 (20-Mar-2020)
Filesystem at /dev/mmcblk1p2 is mounted on /; on-line resizing
required
old_desc_blocks = 1, new_desc_blocks = 4
The filesystem on /dev/mmcblk1p2 is now 7267455 (4k) blocks long.
```

Prior to installing the app, we want to make certain that we are up to date with the latest Xilinx package feeds. This can be done with the following commands:

```
# sudo dnf clean all


# sudo dnf update
```

This may take a significant amount of time, as several hundreds of packages will be upgraded.

To install the app, we need to confirm the presence of the app with the "xmutil getpkgs" command:

```
# sudo xmutil getpkgs

Searching package feed for packagegroups compatible with: kv260

...
avnet-packagegroup-kv260-vvas-sms.noarch          1.0-1.pl2021_1.0
      accelize-2021_1-noarch
...
```

The rpm packages for the app will be hosted on the Accelize stable repository.

https://tech.accelize.com/rpm/stable/2021_1

If the "avnet-packagegroup-kv260-vvas-sms" package is not found, it can be obtained from the Accelize staging repository.

https://tech.accelize.com/rpm/staging/2021_1

In this case, dnf can be instructed to use the Accelize staging repo with the following command:

```
# sudo sed -i 's/stable/staging/g' /etc/yum.repos.d/accelize.repo

# sudo xmutil getpkgs

Searching package feed for packagegroups compatible with: kv260

accelize-packagegroup-kv260-drmdemo.noarch       1.0-1.pl2021_1.0
      accelize-2021_1-noarch
avnet-packagegroup-kv260-vvas-sms.noarch         1.0-1.pl2021_1.0
      accelize-2021_1-noarch
...
```

Next, install the “avnet-packagegroup-kv260-vvas-sms” package, as follows:

```
# sudo dnf install avnet-packagegroup-kv260-vvas-sms
```

This will take a significant amount of time, as several hundreds (900+) of dependencies will be installed.

Loading the Custom App

After each boot, the KV260 Vision AI starter kit will be running the default app:

- kv260-dp

This can be queried with the "xmutil listapps" command.

```
# sudo xmutil listapps
         Accelerator                  Base     Type  #slots Active_slot

            kv260-dp              kv260-dp XRT_FLAT       0           0,
avnet-kv260-vvas-sms avnet-kv260-vvas-sms XRT_FLAT       0           -1
Socket 9 closed by client
```

Before we can load our custom app, we first need to unload the kv260-dp app with the "xmutil unloadapp {app}" command:

```
# sudo xmutil unloadapp kv260-dp

DFX-MGRD> daemon removing accel at slot 0
DFX-MGRD> Removing accel kv260-dp from slot 0

Accelerator successfully removed.

Socket 9 closed by client
```

```
# sudo xmutil listapps
         Accelerator                  Base     Type  #slots Active_slot

            kv260-dp             kv260-dp XRT_FLAT       0          -1
avnet-kv260-vvas-sms avnet-kv260-vvas-sms XRT_FLAT       0          -1
Socket 9 closed by client
```

We then load the custom app with the "xmutil loadapp {app}" command:

```
# sudo xmutil loadapp avnet-kv260-vvas-sms
DFX-MGRD> daemon loading accel avnet-kv260-vvas-sms
DFX-MGRD> Successfully loaded base design.

Accelerator loaded to slot 0

Socket 6 closed by client


# sudo xmutil listapps
         Accelerator                  Base     Type  #slots Active_slot

            kv260-dp             kv260-dp XRT_FLAT       0          -1
avnet-kv260-vvas-sms avnet-kv260-vvas-sms XRT_FLAT       0          0,
Socket 9 closed by client
```

Launching the smart_model_select example

With the avnet-kv260-vvas-sms app loaded, we can run the smart_model_select example application.

The application is located in the "/opt/avnet/kv260-vvas-sms/app" directory:

```
# cd /opt/avnet/kv260-vvas-sms/app
```

First, we need to configure our monitor for 1080P resolution:

```
# source ./setup.sh
setting mode 1920x1080-60.00Hz on connectors 43, crtc 41
testing 1920x1080@NV12 overlay plane 39

[1]+  Stopped(SIGTTIN)        modetest -M xlnx -s 43@41:1920x1080-
60@AR24 -P 39@41:1920x1080@NV12 -w 40:alpha:0
```

This will display a diagonal color bar pattern on the monitor as show below:

With our monitor configured for 1080P resolution, we can launch the application:

```
# sudo ./smart_model_select
```

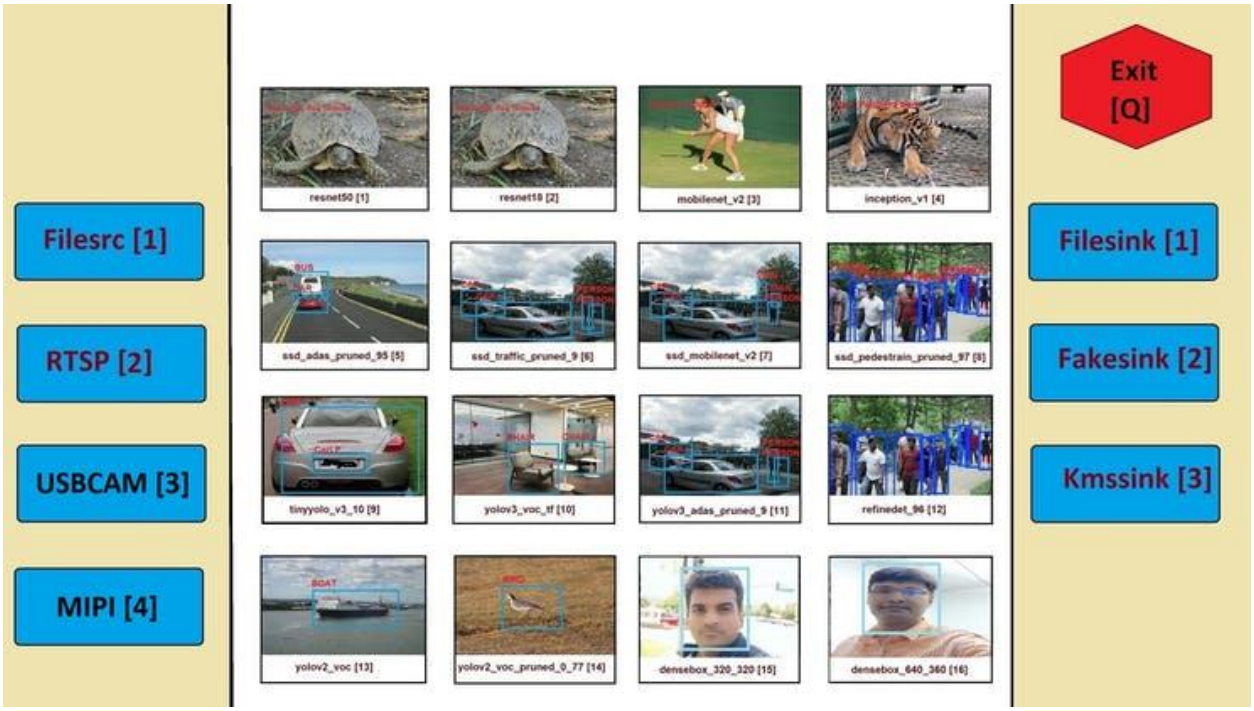
```
##########################################
################## WELCOME #################
##########################################

DEBUG: Got the cmd: gst-launch-1.0 multifilesrc
location=templates/welcome_1080.jpg ! \
        jpegparse ! jpegdec ! \
        queue ! fpsdisplaysink video-sink="kmssink driver-name=xlnx
sync=false" text-overlay=false sync=false

Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Menu displayed on the monitor shows various options available
for input source, ML model, output sink. Each option carry an
index number along side.
Select elements to be used in the pipeline in the sequence of
"input source, ML model, output sink and performance
mode flag" seperated by commas.
eg: 1,1,3,0
Above input will run "filesrc" input, "resnet50" model
"kmssink" used as output sink and performance mode disabled.
Enter 'q' to exit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

This will display the following image on the monitor, which only acts as a visual reference for the choices that can be made on the command line:



Note that the original image was modified to reflect the two additional input sources that were added to the application:

- USB camera
- MIPI sensor

Running the application with video file input

The SD card image comes with the following video files pre-installed:

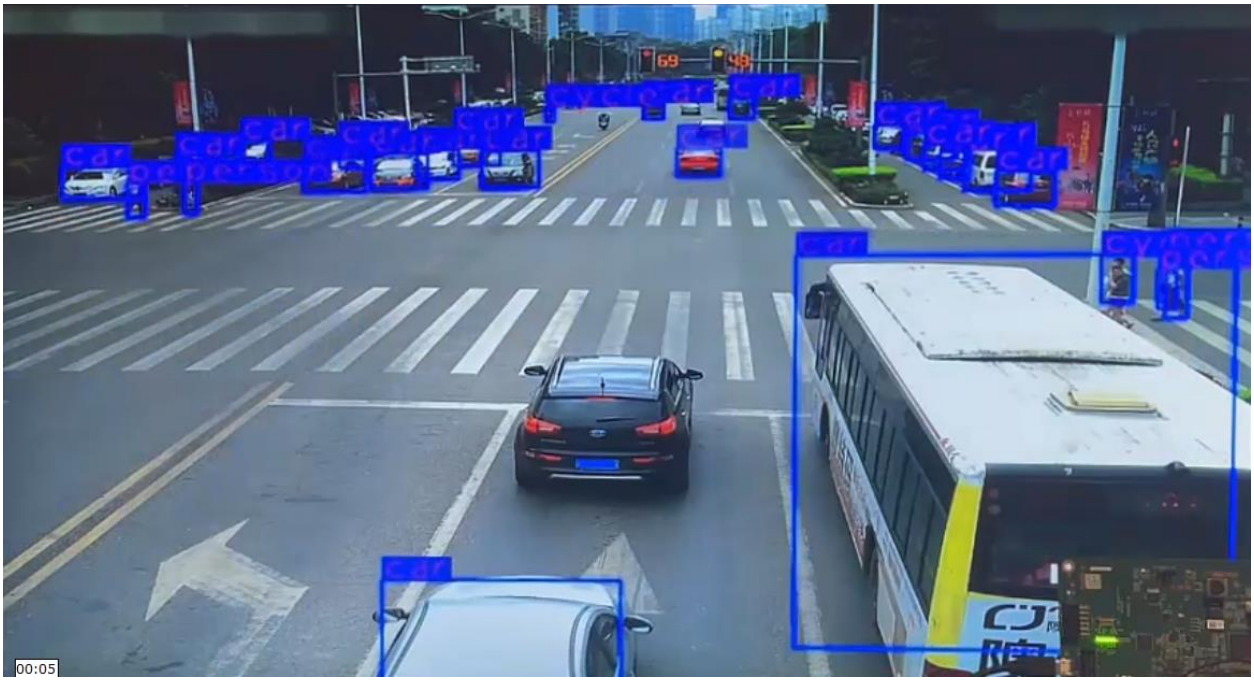
```
videos
├── CLASSIFICATION.mp4
├── FACEDETECT.mp4
├── REFINEDET.mp4
├── SSD.mp4
├── YOLOV2.mp4
└── YOLOV3.mp4
```

In order to apply a model to the video files, use the "1, #, 3, 0" syntax, where # is a value from 1-16, representing one of the supported models.

For example, to apply model #6 to a video file, launch the example as follows:

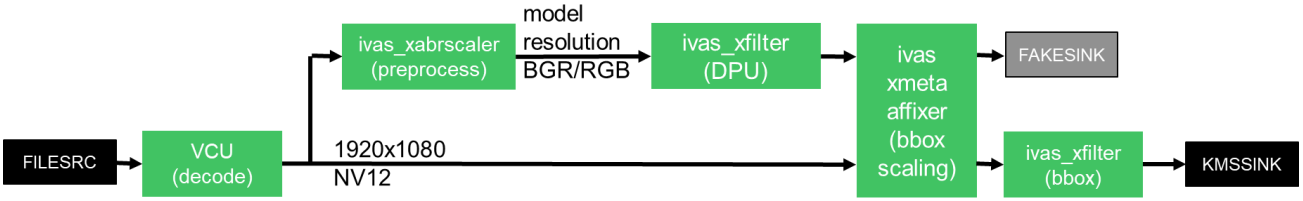
```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Menu displayed on the monitor shows various options available
for input source, ML model, output sink. Each option carry an
index number along side.
Select elements to be used in the pipeline in the sequence of
"input source, ML model, output sink and performance
mode flag" seperated by commas.
eg: 1,1,3,0
Above input will run "filesrc" input, "resnet50" model
"kmssink" used as output sink and performance mode disabled.
Enter 'q' to exit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1,6,3,0
...
 >>>>> Enter any key to return to main menu <<<<<
```

The following animated image shows the expected output for this above example:



[Creating a Custom Kria App - Example 1]

This launches the following GStreamer pipeline:



[Optional] Running the application with the USB camera input

If you do not have a Logitech BRIO USB camera, skip this section.

In order to apply a model to a USB camera input, use the "3, #, 3, 0" syntax, where # is a value from 1-16, representing one of the supported models.

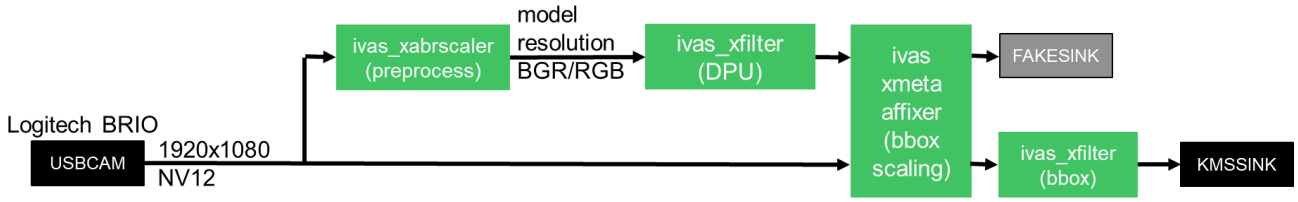
For example, to apply model #16 to the USB camera, launch the example as follows:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Menu displayed on the monitor shows various options available
for input source, ML model, output sink. Each option carry an
index number along side.
Select elements to be used in the pipeline in the sequence of
"input source, ML model, output sink and performance
mode flag" seperated by commas.
eg: 1,1,3,0
Above input will run "filesrc" input, "resnet50" model
"kmssink" used as output sink and performance mode disabled.
Enter 'q' to exit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
3,16,3,0
...
 >>>>> Enter any key to return to main menu <<<<<
```

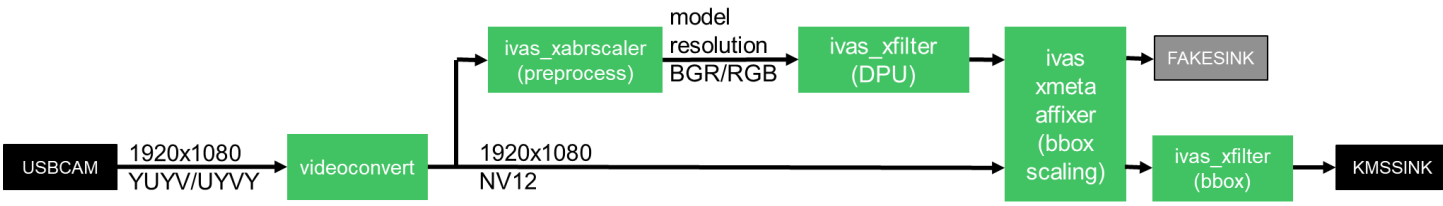
The following animated image shows the expected output for this above example:

Note that for real-time execution, the USB camera must support NV12 format. For this reason, I recommend to use the following USB camera:

- Logitech BRIO



Other USB cameras that do not support NV12 format, will require software color space conversion, which will prevent achieving real-time performance.



It is possible to accelerate this conversion, but this is beyond the scope of this project.

NOTE : If you get a green screen after a certain period, stopping and re-starting the example will resolve this issue.

[Optional] Running the application with MIPI sensor input

If you do not have an AR1335 IAS sensor module, skip this section.

In order to apply a model to the MIPI sensor input, use the "4, #, 3, 0" syntax, where # is a value from 1-16, representing one of the supported models.

For example, to apply model #14 to the MIPI sensor, launch the example as follows:

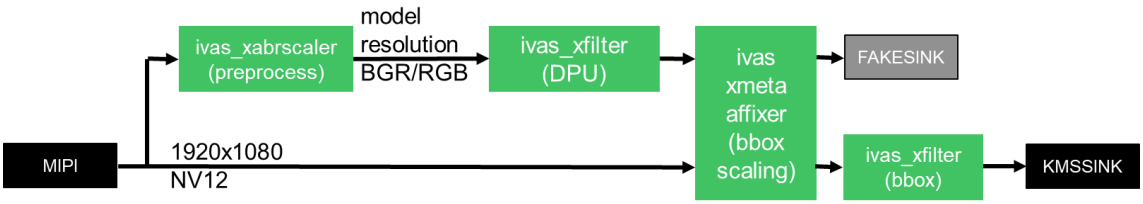
```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Menu displayed on the monitor shows various options available
for input source, ML model, output sink. Each option carry an
index number along side.
Select elements to be used in the pipeline in the sequence of
"input source, ML model, output sink and performance
mode flag" seperated by commas.
```

```
eg: 1,1,3,0
Above input will run "filesrc" input, "resnet50" model
"kmssink" used as output sink and performance mode disabled.
Enter 'q' to exit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
4,14,3,0
...
 >>>>> Enter any key to return to main menu <<<<<
```

The following animated image shows the expected output for this above example:

[Creating a Custom Kria App - Example 3]

This example launches the following GStreamer pipeline:



Revision History

Revision	Date	Description
1.0	2022/02/16	Initial Version