Bitácora de trabajo

Taller De Sistemas Embebidos

Keylor RIvera Gamboa

Proyecto:
Sistema operativo a la medida para aplicaciones embebidas de multimedia con el marco de trabajo de Intel DL Streamer y el flujo de síntesis de Yocto Project

1 S 2025

28 de marzo, 2025

- Se investigó el flujo de trabajo con el marco de trabajo de Intel DL Streamer y sus características. Se leyó sobre cómo utilizar las tuberías (pipes) a través de este marco de trabajo.

29 de marzo, 2025

- Se realizó una primera búsqueda del modelo de interés de Model Zoo.

31 de marzo, 2025

- Se intentó correr un primer modelo de Model Zoo y sus dependencias.

7 de abril, 2025:

- Se analizó el repositorio del primer taller, el cual contenía información sobre el programa de python integrando el flujo de trabajo de Gstreamer.
- Errores encontrados: el path dentro del archivo .py debe ser referente a lo interno del contenedor docker. https://github.com/lumurillo/openvino-workshop-tec/blob/main/opencv_gst_example.ipynb

11 de abril, 2025:

- El modelo se logró correr dentro de un contenedor Docker para el modelo de human estimation.
- Evidencia: comando para correrlo: docker compose run --rm dlstreamer bash >> python3 run_pose.py:


```
keylor@pop-os:~/Documents/I S 2025/Embebidos/proyecto1$ tree
.
├── compose.yaml
├── dockerfile
└── test
    ├── model_proc
    │   └── human-pose-estimation-0001.json
    ├── run_pose.py
    └── utils
        ├── gst_utils.py
        ├── __init__.py
        └── __pycache__
            ├── gst_utils.cpython-310.pyc
            └── __init__.cpython-310.pyc
```

- Se inició la investigación para realizar una imagen mínima con Yocto con OpenVino. A continuación se detallan ciertos errores encontrados y la solución aplicada.

- https://docs.openvino.ai/2023.3/openvino_docs_install_guides_installing_openvino_yocto.html
- Al seguir el recurso anterior hay que notar que al clonar los repos, estos sean compatibles con la versión base de nuestro poky. De modo que si no se contempla esto, puede suceder los siguientes errores:

```
keylor@pop-os:~/YOCTO_PROJECT/build$ bitbake-layers add-layer /home/keylor/YOCTO_PROJECT/meta-openembe
dded/meta-oe
NOTE: Starting bitbake server...
ERROR: Layer openembedded-layer is not compatible with the core layer which only supports these series
: scarthgap (layer is compatible with walnascar)
ERROR: Parse failure with the specified layer added, exiting.
```

Corrección del error:

```
keylor@pop-os:~/YOCTO_PROJECT$ cd meta-openembedded/
keylor@pop-os:~/YOCTO_PROJECT/meta-openembedded$ git fetch
keylor@pop-os:~/YOCTO_PROJECT/meta-openembedded$ git checkout scarthgap
Branch 'scarthgap' set up to track remote branch 'scarthgap' from 'origin'.
Switched to a new branch 'scarthgap'
keylor@pop-os:~/YOCTO_PROJECT/meta-openembedded$
```

De este modo nos pasamos a la misma versión base de scarthgap.

Y de la misma forma con los otros repositorios clonados.

```
keylor@pop-os:~/YOCTO_PROJECT/build$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                 path                                                                      priority
==========================================================================================================
==
core                  /home/keylor/YOCTO_PROJECT/poky/meta                                      5
yocto                 /home/keylor/YOCTO_PROJECT/poky/meta-poky                                 5
yoctobsp              /home/keylor/YOCTO_PROJECT/poky/meta-yocto-bsp                            5
intel                 /home/keylor/YOCTO_PROJECT/meta-intel                                     5
openembedded-layer    /home/keylor/YOCTO_PROJECT/meta-openembedded/meta-oe                      5
meta-python           /home/keylor/YOCTO_PROJECT/meta-openembedded/meta-python                   5
clang-layer           /home/keylor/YOCTO_PROJECT/meta-clang                                     7
keylor@pop-os:~/YOCTO_PROJECT/build$
```

Tras intentar cocinar la imagen de Yocto, el espacio no fue suficiente, por lo que siguió esta guía.
https://4sysops.com/archives/extending-lvm-space-in-ubuntu/
Pasos:

Espacio: sudo vgdisplay (permite ver si hay free space)

If you see free space, use the lvdisplay command to view the LV path.

sudo lvextend -l +100%FREE -r /dev/ubuntu-vg/ubuntu-lv
Agregar meta-openembeded/meta-multi

12 de abril, 2025:

- Imagen base cocinada. Se procede a complementar con las dependencias de gstreamer necesarias.

- Procedimiento:
  (adicción de los layers)

bitbake-layers add-layer ../meta-openembedded/meta-multimedia
(con el repo de meta-openembeded en la versión correcta)
Se procede a configurar el local.config:

```
#gstreamer dependences
IMAGE_INSTALL:append = " \
    gstreamer1.0 \
    gstreamer1.0-plugins-base \
    gstreamer1.0-plugins-good \
    gstreamer1.0-plugins-bad \
    gstreamer1.0-libav \
    gstreamer1.0-python \
    gstreamer1.0-rtsp-server \
    gstreamer1.0-meta-base \
"
LICENSE_FLAGS_ACCEPTED += "commercial"

#dependences for pyhton app
# Dependencias de Python para el modelo
IMAGE_INSTALL:append = " openssh \
    python3-pillow \
    python3-matplotlib \
"

LICENSE_FLAGS_ACCEPTED += "commercial"
#(esta última línea es para aceptar la licencia de uso de uno de los plug in”
```

Las layers completas hasta el momento fueron:

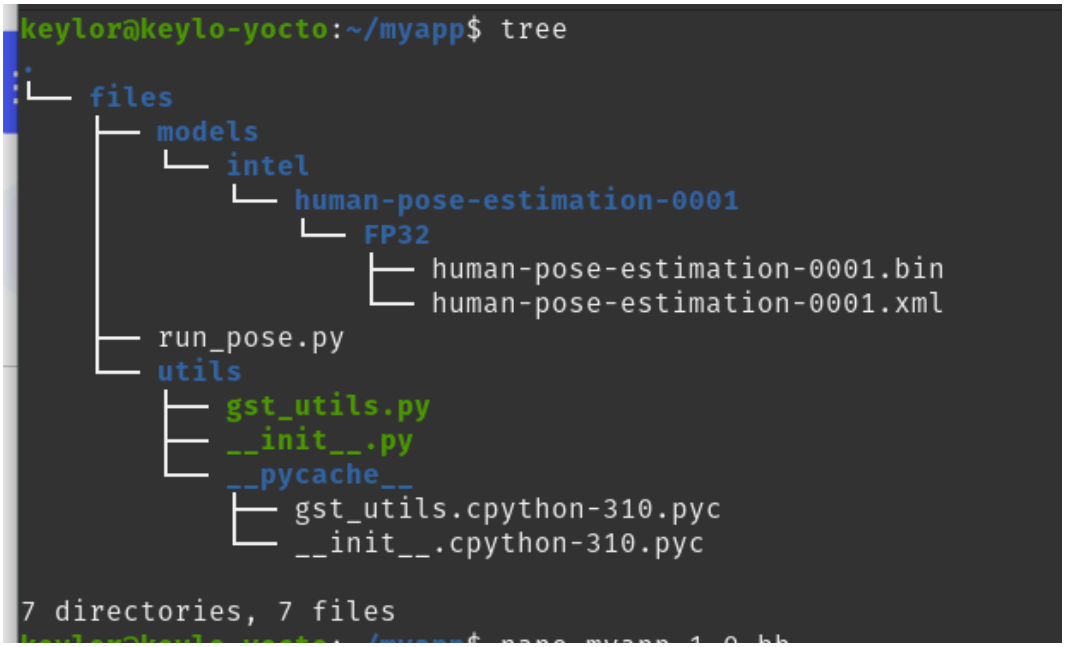
```
keylor@keylo-yocto:~/poky/build$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                 path                                                                      priority
==========================================================================================================
core                  /home/keylor/poky/poky/meta                                               5
yocto                 /home/keylor/poky/poky/meta-poky                                          5
yoctobsp              /home/keylor/poky/poky/meta-yocto-bsp                                     5
intel                 /home/keylor/poky/meta-intel                                              5
openembedded-layer    /home/keylor/poky/meta-openembedded/meta-oe                               5
meta-python           /home/keylor/poky/meta-openembedded/meta-python                           5
clang-layer           /home/keylor/poky/meta-clang                                              7
multimedia-layer      /home/keylor/poky/meta-openembedded/meta-multimedia                       5
networking-layer      /home/keylor/poky/meta-openembedded/meta-networking                       5
keylor@keylo-yocto:~/poky/build$
```

13 de abril, 2025

- Para correr la imagen en la vm y que acepte el root como usuario se debe emular con los siguientes flags:
  runqemu qemux86-64 nographic serialstdio
- Se verificó que gstreamer funcionara con una prueba sencilla, lo cual resultó de forma exitosa, sin embargo, no están disponibles los plug in tales como gvaclassify, por lo que se investigó cómo agregar una meta layer, sin embargo, no se consiguió hacerlo.
  Corrección: para generar el display:
- Con las conexiones correspondientes con la bandera -X, se corrige el error modificando el local config:
  # X11 forwarding requirements
  IMAGE_INSTALL:append = " xauth"

Configure SSH for X11 forwarding
PACKAGECONFIG:append:pn-openssh = " x11"
Se corre la prueba:
gst-launch-1.0 videotestsrc ! videoconvert ! autovideosink

- Y aparece un pequeño display de video.
- Layer personalizada:
  Se generó un directorio con la estructura mostrada a continuación.



- Se agregó el archivo .bb dentro de “myapp”.

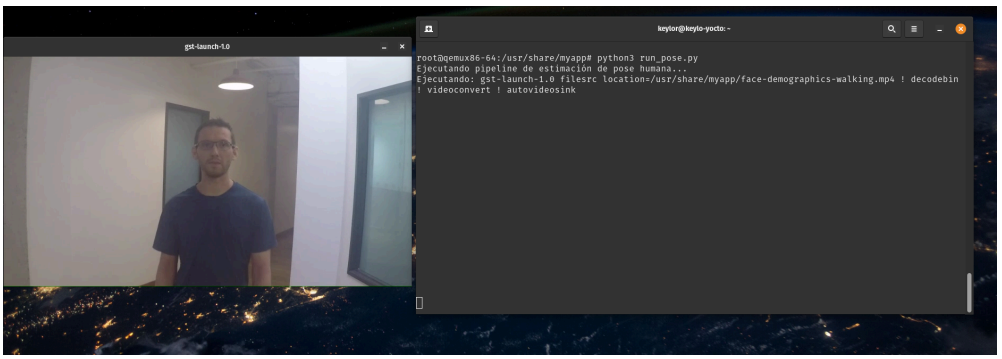
<u>14 de abril, 2025</u>

- Se agregó el archivo model proc que contiene el archivo json. Se modificó el archivo .py principal con las rutas a lo interno de la imagen, en mi caso “/usr/share/myapp”, esto dado la dirección del config de mi layer.
- Ejecutando la aplicación, encontré un error:
  Error en el pipeline:
- ** (gst-plugin-scanner:377): CRITICAL **: 16:43:39.302: Couldn't g_module_open libpython. Reason: /usr/lib/libpython3.12.so: cannot open shared object file: No such file or directory
  WARNING: erroneous pipeline: no element "gvaclassify"

- Este tiene dos componentes, el último debido a que he agregado los elementos necesarios de dlstreamer, lo cual era de esperarse.
  El primero es que no encuentra la librería libpython3.12.so pero al buscar esta si está presente, pero con el nombre libpython3.12.so.1.0, es decir, hay una desactualización en el nombre de búsqueda.
- El plugin de Python para GStreamer (`gst-python`) fue compilado para buscar explícitamente `libpython3.12.so` (la biblioteca de desarrollo).

- En mi sistema, tengo `libpython3.12.so.1.0` (la biblioteca de tiempo de ejecución con número de versión completo).
- Solución: hacer un enlace simbólico o agregar en local.config python3-dev.

- Se modificó temporalmente el script de python para que tomara el video, lo decodificara, lo convirtiera y lo envie al autovideosink. El resultado fue exitoso:



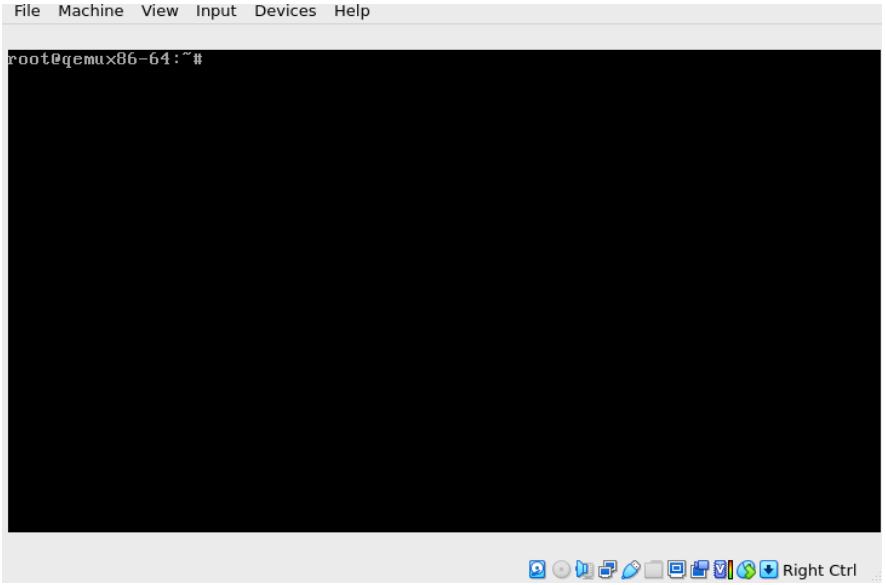
- La tarea pendiente es añadir los elementos de dlstreamer para la detección de la pose humana.

18 de abril, 2025:

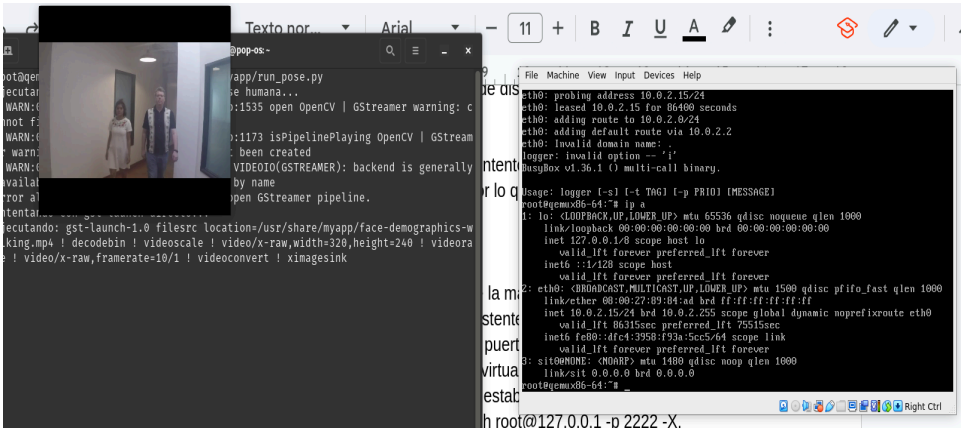
- Se clonó el repo de dlstreamer de intel y se intentó añadir como una layer.
- El intento anterior dio muchos problemas, por lo que no se usó directamente el repo clonado sino la url.
- No hubo un resultado exitoso.

19 de abril:

- Comprobación de la imagen generada sobre la máquina virtual.
- La imagen fue montada como disco duro existente en una maquina virtual.
- Se configuró la red para hacer un mapeo de puertos y hacer una conexión ssh desde el computador host hacia la máquina virtual. AL arrancar la maquina virtual se debe correr el comando ‘dhcpcd eth0 lo que establece una ip para eth0. Luego, en el computador host se hace la conexión con ssh root@127.0.0.1 -p 2222 -X.
  Evidencia:
  Máquina virtual:



Corriendo la aplicación:



- Se inició la documentación - tutorial.

20 de abril, 2025

- Finaliza la documentación.