Escuela de Ingeniería Electrónica
EL5617 Trabajo Final de Graduación

TEC | Tecnológico de Costa Rica

Trabajo Final de Graduación

Bitácora de Trabajo - Entrega 3

David F. Duarte Sánchez
II Semestre 2024

1. Semana 8

1.1. Corrección de Tesis

Fecha de trabajo: 06/09/2024.
Objetivo: Corrección de observaciones realizadas a la Tesis.

Reporte de actividades
-Se comienza a elaborar la sección 1 y 3 de la Tesis con el fin de revisar el estado final de estos en la próxima reunión programada para control de las tesis, además de esto se puede comenzar a realizar el primer capítulo de la Tesis, ya que el comité ha aprobado el anteproyecto.
- Se comienza la elaboración de un manual de usuario, con el fin de hacer más fácil la evaluación del flujo de trabajo y la depuración del mismo, de esta forma se tendrán estandarizadas las versiones requeridas y las salidas esperadas en cada uno de los procedimientos que se realizan a lo largo de la experimentación.
- Exploración sobre mejores versiones de Ubuntu sobre la cual se puedan trabajar los contenedores primeramente se creía que un contenedor de Ubuntu 18.04 era la mejor aproximación, pero se ha encontrado que la versión 16.04 es más estable y permite un mejor desarrollo del flujo de trabajo
- Generación de un archivo Docker con el fin de generar una receta para montar más sencillamente soluciones en el ambiente propuesto de trabajo, de esta forma se estandarizan las versiones requeridas, evitando de esta forma futuros problemas de versión los cuales son muy delicados cuando se trata con ambientes embebidos.
Productos obtenidos
Se comienza la elaboración del capítulo 1 y 3 además de la finalización del capítulo 2 esto con el fin de buscar referencias bibliográficas de utilidad a la hora de poder implementar los pasos descubiertos en este periodo de investigación, seguido de esto se comienza con la elaboración de un manual de usuario el cual sea de utilidad a la hora de replicar el sistema realizado. Al trabajar con Ubuntu y Yocto, las versiones de los requerimientos son fundamentales, ya que de estas depende la correcta implementación del flujo de trabajo, al hacer referencia a esto se tenían dos panoramas, en el primero de ellos trabajamos con un ambiente virtual, pero este mostraría limitantes a la hora de tratar de exportarlo y llevarlo al siguiente paso. Es por esto que se utiliza un contenedor en Docker donde se genera con la versión de Ubuntu requerida mediante un archivo de generación. Es así como se logra determinar que la versión 16.04 es más estable que la 18.04 para la elaboración del flujo de trabajo.

Como se mencionó anteriormente se logró encontrar un BSP disponible para la Zead-Board, el mismo se encuentra actualmente disponible para la distribución de Yocto Zeus la cual tuvo el lanzamiento en el año de 2019, por tanto es necesario instalar Ubuntu 18.04 para poder llevar a cabo la elaboración del conjunto de flujos de trabajo. Los mismos se realizan siguiendo esta serie de pasos.

Tabla con las versiones de Yocto

Tabla con las versiones de paquete de soporte de tarjeta según el repo de Xilinx

Cuadro 1

Codename	Yocto Project Version	Release Date	Current Version	Support Level	Poky Version	BitBake branch
Zeus	3.0	October 2019	3.0.4 (August 2020)	EOL	22.0.3	1.44
Dunfell	3.1	April 2020	3.1.33 (May 2024)	EOL - LTS[1]	23.0	1.46

Indicar que se sigue bajo la investigación si hay alguna versión LTS que soporte el ZedBoard - Se logra encontrar que la versión 16.04 LTS puede ejecutar el flujo de trabajo requerido para la correcta generación del flujo de trabajo

Primeramente se genera el ambiente de trabajo.

```
docker pull ubuntu:16.04
docker run -it ubuntu:16.04
```

Con las instrucciones mostradas anteriormente se genera un contenedor con la versión de Ubuntu 16.04.

Seguido de esto se debe de generar un usuario no root, esto con el objetivo de no generar alarmas más adelante en la generación de la imagen embebida

```
# Update package lists
apt-get update

# Install sudo if it's not already installed
apt-get install -y sudo

# Create a new user with a home directory
useradd -ms /bin/bash myuser

# Set a password for the new user
echo "myuser:password" | chpasswd

# Add the new user to the sudo group
usermod -aG sudo myuser


su - myuser
```

Una vez generado el usuario no root y registrado en el mismo se deben de seguir estos pasos:

```
apt-get update
apt-get install -y locales

locale-gen en_US.UTF-8

dpkg-reconfigure locales

locale

export LANG=en_US.UTF-8
```

```
  export LC_ALL=en_US.UTF-8

  locale
```

Este paso se realiza para que Python sepa interpretar correctamente los caracteres contenidos en el formato mencionado anteriormente. Una vez concluidos estos pasos se puede continuar con la parte existente del instructivo

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo
gcc-multilib build-essential chrpath socat cpio python python3
python3-pip python3-pexpect xz-utils debianutils iputils-ping
python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3
xterm

git clone -b zeus https://git.yoctoproject.org/git/poky
cd poky

git clone -b zeus https://github.com/Xilinx/meta-xilinx
git clone -b zeus https://github.com/openembedded/meta-openembedded.git

source oe-init-build-env

echo "MACHINE??=\"zedboard-zynq7\"" >> conf/local.conf
echo "IMAGE_FEATURES += \"package-management\"" >> conf/local.conf
echo "DISTRO_HOSTNAME = \"zynq\"" >> conf/local.conf

bitbake-layers add-layer ../meta-xilinx/meta-xilinx-bsp/
bitbake-layers add-layer ../meta-openembedded/meta-oe/

bitbake core-image-minimal
```

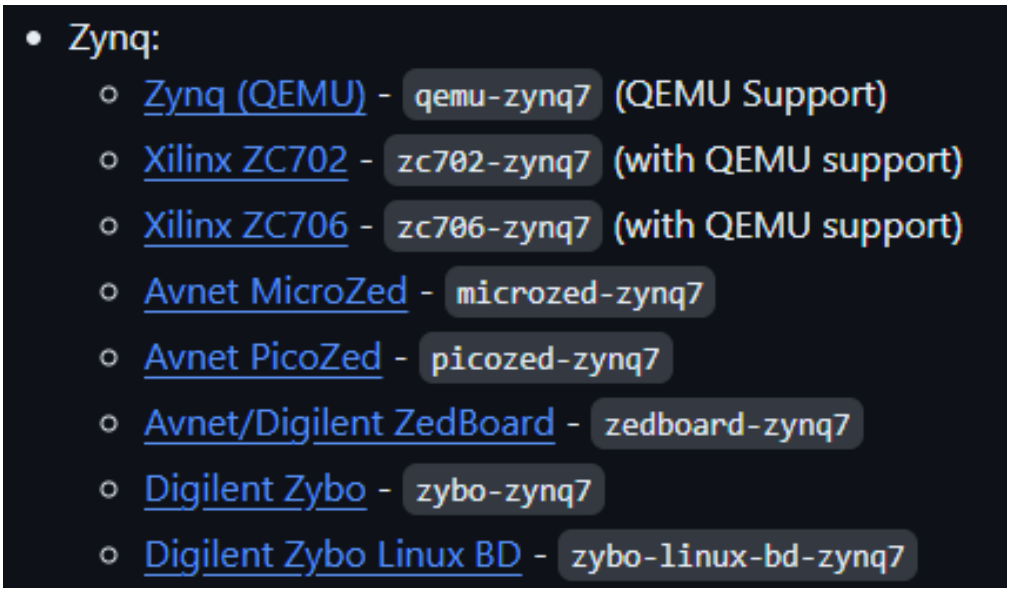

Figura 1: BSP Xilinx