Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

EdgeAI-Sistema Embebido para el reconocimiento y clasificación de emociones faciales con la Biblioteca OpenCV, Machine Learning y el flujo de trabajo de Yocto Project

---

Taller de Sistemas Embebidos

Integrantes:

Jorge Andrés Brenes Alfaro

Fabián Bustillos Villavicencio

Kimberly María Carvajal Méndez

Jose Andrés Chaves Williams

Profesor:

Ing. Johan Carvajal Godínez

25 de octubre de 2021

Introducción

Actualmente el desarrollo tecnológico va en incremento a nivel mundial, donde nuevas tecnologías surgen y con ello nuevas aplicaciones e implementaciones son desarrolladas. Cada vez son desarrollados más sistemas embebidos que ayudan a la vida cotidiana. El sistema embebido es un sistema computacional diseñado para ejecutar una función en específico, de tal forma que se pueda satisfacer una necesidad [1].

Los sistemas embebidos junto al desarrollo tecnológico se vuelven en sistemas cada vez más inteligentes donde aplicaciones para inteligencia artificial, aprendizaje automático, visión por computadora, entre otras, se vuelven más frecuentes, como lo es el reconocimiento de patrones, control de gestos y procesamiento de imágenes. Y no sólo es reconocer a través del vídeo si alguien está sonriendo, está triste o enojado, sino que le permite diferenciar si esta sonrisa es una de nerviosismo o de felicidad [2].

Estas aplicaciones crecen a gran velocidad, están presente en un sinfín de aplicaciones y sus oportunidades son cada vez mayores, como lo es dentro del campo de la medicina, donde un gran ejemplo de su uso es en el diagnóstico de la depresión, cuya enfermedad afecta a más de 300 millones de personas en el mundo y la inteligencia artificial emocional puede jugar un rol preventivo [2]. O como es el fin de este proyecto, el poder determinar la reacción de las personas en una sala de cine para evaluar las emociones de las personas ante un estímulo visual determinado.

Gracias a que la tecnología de reconocimiento de emociones está en desarrollo, su conocimiento es de vital importancia en un futuro, además, provee grandes ventajas al ser humano. Por ello, se abarca una aplicación de aprendizaje automático, en donde se desarrolla una arquitectura física de un sistema embebido por medio de un análisis operacional del sistema, para lo cual se debe sintetizar aplicaciones de software capaces de implementar las funcionalidades requeridas para el sistema por medio del uso de herramientas de visión por computador y aprendizaje automático, así como el flujo de trabajo de *Yocto Project* con *tensorflow lite* y *OpenCV*, creando de esta forma un sistema *Edge-AI*. Este es un sistema que utiliza algoritmos de aprendizaje automático para procesar datos generados por un dispositivo de hardware a nivel local, sin la necesidad de conexiones a internet, reduciendo costos y tiempos de latencia, aumenta el nivel de seguridad en cuanto a la privacidad de los datos, la reducción en el ancho de banda requerido y estos dispositivos no requieren un mantenimiento especializado [4].

Características del host

En esta sección se detalla las características de los computadores de cada integrante para el desarrollo del proyecto, las cuales se pueden observar en el cuadro 1.

Cuadro 1: Caracteristicas del computador host de cada integrante

Host	Jorge	Fabian	Kimberly	José
SO nativo	Windows 10	Ubuntu 21.04	Windows 10	Ubuntu 20.04.03
Memoria RAM	16GB	8GB	8GB	4GB
Procesador	Intel Core i7-1165G7 CPU @ 2.80 GHz	Intel Core 2 Duo @ 1.3GHz	Intel Core i5-5200U CPU @ 2.20GHz	Intel Core i3-5005U CPU @ 2.00GHz
Tipo de sistema	SO de 64 bits, basado en x64	SO de 64 bits	SO de 64 bits, basado en x64	SO de 64 bits
Disco duro	1TB SSD	256 GB SSD	1 TB	1 TB
Máquina virtual	Sí	No	Sí	No
SO MV	Ubuntu 20.04.3 LTS	—	Ubuntu 20.04.3 LTS	—
Disco duro MV	256 GB SSD	—	140 GB	—
RAM MV	8 GB	—	4 GB	—

OpenCV

OpenCV es una biblioteca de visión por computador y aprendizaje automático de código abierto, lo que facilita su uso y por lo tanto permite que hayan más de 2500 algoritmos optimizados para el uso público. Los algoritmos creados con esta biblioteca permite procesar imágenes para obtener aplicaciones como reconocimiento facial, identificación de objetos e identificación de patrones. [14].

Para la aplicación de reconocimiento de emociones se hará uso de esta biblioteca de python principalmente para el manejo y preprocesamiento de la imagen tomada por la cámara, además del uso de la cámara, toma de imagenes, y el mostrar la ventana de visualización de video. Se hará uso del toolkit de *Image Processing* el cual como indica su nombre nos brinda distintos módulos y funciones para procesar imagenes.

Uno de los módulos utilizado corresponde al *Color Space Conversions* el cual permitirá manipular el color de las imagenes, esto se hace ya que procesar las imágenes en escala de grises facilita el funcionamiento de la aplicación. Además, haremos uso del módulo de *Drawing Functions* que permite dibujar en nuestra imagen de vídeo objetos como el rectángulo que indica la identificación del rostro y también la palabra de la emoción que se ha identificado.

Tensorflow Lite

Tensorflow es la biblioteca que funciona como el cerebro del código, ya que es una biblioteca de aprendizaje automático, que mediante el uso de modelos de aprendizaje de maquina permite ejecutar reconocimiento de patrones [7], lo cual al utilizar un modelo previamente entrenado comparará las caras obtenidas mediante opencv, las comparará contra el modelo y dirá con cual de las emociones es más probable que se identifique en cualquier momento dado. Esto lo hace mediante tensores, los cuales son similares a un vector con varias dimensiones en una gráfica, los cuales reciben una entrada (en este caso el rostro) y van a entregar una salida (un punto), el cual se encontrará más cercano a una emoción, el código se encargará de determinar cual es tal emoción.

¿Por qué usar tensorflow lite?

Tensorflow lite, es la versión ligera de Tensorflow, la cual está optimizada para edge computing, de manera en que reduce la latencia, el tamaño de los modelos utilizados y el consumo de batería. El uso de esta es optimo en sistemas embebidos como el raspberry pi, debido a que no cuenta con tanta capacidad de procesamiento como un pc de escritorio o portatil. Y la implementación de este algoritmo requiere que el código funcione de la manera más rápida posible para así no perder emociones en el transcurso de la operación. Además de los beneficios presentados anteriormente tambien se incluyen otros beneficios como la compatibilidad con múltiples plataformas y con diversos lenguajes, además de un alto rendimiento, con aceleración de hardware y optimización de modelos [8].

Para usar tensorflow lite es necesario utilizar un modelo en formato .tflite, esto se puede hacer de varias formas, creando un modelo desde cero, descargando un modelo de tflite ya entrenado u convirtiendo un modelo de Tensorflow a tflite (este método será utilizado en este proyecto). Esta conversión se da mediante un proceso llamado cuantización el cual toma el módelo que tiene datos en formato float-32 y los convierte a formato float-16 disminuyendo así el tamaño del modelo a utilizar en un mínimo de 50 % sacrificando un poco de la precisión del mismo; sin embargo, la disminución del tamaño y optimización del modelo presenta los suficientes beneficios para justificar la perdida de precisión. [9]

Aplicación de reconocimiento de emociones

Tal como dice el nombre de la sección se implementa una aplicación de reconocimiento de emociones que ayuden a determinar las reacciones de las personas en una sala de cine. Para ello, se toma como base el código que se encuentra en [5], donde este se analiza y se le realizan las modificaciones necesarias para llevar la tarea deseada.

El código aplicado contiene 3 archivos, los cuales se exponen a continuación:

- **Emotions.py**: es el archivo principal donde se lleva a cabo toda la operación para el reconocimiento de las emociones. En este archivo se realiza el llamado de los otros 2 necesarios para su funcionamiento.
- **model.tflite**: dicho archivo contiene el modelo diseñado y preentrenado, además, se encuentra convertido en .tflite para mejorar el rendimiento de la *raspberry pi* al ejecutar la operación.
- **haarcascade_frontalface_default.xml**: es un archivo necesario para la ejecución del programa, pues se trata de un clasificador en Haar Cascades. OpenCV ofrece clasificadores pre entrenados no solo de rostros de personas (es el utilizado), sino de ojos, sonrisa, caras de gatitos, entre otros [6].

Para la ejecución del código se hace uso de la diversas bibliotecas para realizar más amena la tarea. Estas bibliotecas son: numpy, la cuál nos ayuda a realizar cálculos complejos de una forma más simple. La biblioteca tensorflow, la cual es una biblioteca de código abierto especializada para desarrollar y entrenar modelos de aprendizaje automático [7]; no obstante, dicha biblioteca es pesada en cuanto a tamaño y ejecución por lo que se requiere mejorar su desempeño para la aplicación en una *raspberry pi* 2. Es por ello que se hace uso de tensorflow lite, ya que ayuda a reducir el tamaño del modelo y de los objetos binarios. De esta biblioteca se usa *tflite_runtime.interpreter* para poder leer el modelo convertido a .tflite.

Otra biblioteca importante y necesaria para el desarrollo de la aplicación es OpenCV, la cual se encarga principalmente de obtener los rostros. Este captura el rostro del vídeo por medio de la cámara, le aplica una escala de grises a la imagen obtenida para facilitar la clasificación y predicción de la emoción, además, enmarca el lugar el rostro e imprime la emoción. Esta biblioteca hace uso de unos de los archivos necesarios para la ejecución del programa.

Por último pero no menos importante, las bibliotecas *time* y *datetime* que nos permiten obtener el tiempo y fecha de las emociones, esto con el fin de hacer un registrar las emociones detectadas por la aplicación. Esto se lleva a cabo mediante un archivo .csv en donde se le agrega la emoción, la fecha y la hora de detección.

Para el funcionamiento de la detección de emociones se hace uso de un modelo preentrenado en tensorflow lite correspondiente al archivo "model.tflite". Para ejecutar el programa es necesario abrir el archivo Emotions.py y tener una cámara disponible para poder realizar la detección. Con esto definido, solo se debe ejecutar dicho archivo para llevar a cabo el programa y obtener un archivo excel donde se registran las emociones obtenidas y su tiempo de obtención, esto para su posterior análisis y procesamiento.

El código se puede observar en el siguiente link: `https://github.com/Josechaves19/Edge-Computing`.

Yocto Project

Yocto Project es un proyecto colaborativo de código abierto que ayuda a los desarrolladores a crear sistemas personalizados basados en el sistema operativo Linux independientemente de la arquitectura de hardware [10]. Este método de trabajo provee código abierto, infraestructura y un conjunto de herramientas de calidad para ayudar a desarrollar distribuciones propias de Linux en cualquier arquitectura [11], permitiendo colaborar compartiendo tecnologías, software, configuraciones y mejores prácticas para crear estas imágenes.

En este proyecto se hará uso de Yocto Project junto a qemu para emular una imagen que cuente con las especificaciones necesarias para la aplicación de reconocimiento de emociones en una raspberry pi. Desde esta imagen seremos capaces de instalar las dependencias necesarias para esta aplicación y probar que esta funcione correctamente antes de probarlo en la raspberry pi.

Flujo de trabajo de Yocto Project

Para llevar a cabo la implementación de una imagen de Linux a la medida con Yocto Project se debe seguir una serie de pasos que se muestran a continuación [12].

1. Primeramente se comprueba que el computador posea los programas Git, tar y python.
2. Seguidamente se realiza la instalación del Yocto Project en el computador según el sistema operativo que se posea.
    a) Para el Ubuntu utilizado *$ 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 libsdl1.2-dev xterm*
3. Descargar la distribución poky más reciente con el comando *git clone*
    a) Para el poky utilizado en el proyecto
    $ git clone -b hardknott git://git.yoctoproject.org/poky.git
4. Inicialización del entorno de compilación lo cual crea la carpeta build con archivos como blayers.bb y layer.conf, esto con el comando *$ source oe-init-build-env*

5. Se inicia la compilación de la imagen del sistema operativo para la máquina objetivo, dicha compilación puede tardar según las variantes del computador host.
   a) En este proyecto se utiliza una imagen Sato con el comando *$ bitbake core-image-sato*
6. Realizar la primera simulación de la imagen usando QEMU en la maquina objetivo con el comando *$ runqemu qemux86* o *$ runqemu qemux86-64 nographic*
7. Una vez compilada la imagen se pueden agregar las capas necesarias para el desarrollo del proyecto en curso.
   a) Para agregar una capa existente se hace una copia local del repositorio utilizando el comando *$ git clone url* dentro de la distribución poky utilizada.
   b) Una vez copiada la capa se ingresa al directorio de build para agregar la capa con el comando *$ bitbake-layers add-layer "layer path"*
   c) Finalmente se compila de nuevo la imagen para agregar la capa a la compilación con *$ bitbake core-image-sato*

Flujo de trabajo para la creación de capas de yocto proyect

Si se desea y es necesario para el proyecto se pueden crear capas propias para la imagen, esto se hace siguiendo los siguientes pasos [13]

1. Primeramente se crea el directorio de la capa dentro de la carpeta poky, esta debe tener el nombre *meta-< capa >*.
2. Dentro del directorio de la capa se crea un directorio llamado *conf*, dentro se ubica el archivo *layer.conf* con las especificaciones necesarias.
3. Se crea el directorio de la receta con el nombre *recipe-< receta >* dentro del directorio de la capa, además de crear el archivo .bb de modo *< receta > .bb*, además de una carpeta *< receta > . < version >* donde se encuentran los archivos de programa necesarios para el proyecto.
4. Se le da el formato al archivo .bb agregando los directorios de los archivos de programa a utilizar, licencias, comandos de configuración, compilación e instalación, entre otras dependencias que se requieran.
5. Se añade la capa ingresando al directorio poky/build y con el comando *$ bitbake-layers add-layer "layer path"*
6. Se compila la receta creada con el comando *$ bitbake < receta >*.

A continuación se exponen las recetas integradas en este proyecto, ya que al realizar la instalación del poky este trae consigo solamente las capas para su funcionamiento .

Meta-openembedded

Esta capa es de utilidad ya que con ella somos capaces de obtener las dependencias necesarias para el uso de python3 en la imagen creada.

Meta-python

Esta capa depende de meta-openembedded y de igual manera ayuda a la correcta instalación de python3 dentro de la imagen, además, de proveernos de bibliotecas de uso frecuenta y necesarias para el proyecto como numpy, time y datetime.

Meta-raspberrypi

Esta capa brinda las especificaciones necesarias del hardware de una raspberry pi para cuando se realice la síntesis de la imagen esta pueda se soportada en dicho sistema embebido.

Meta-tensorflow-lite

Esta capa depende de meta-python para su correcto funcionamiento. Al agregarla se obtiene las dependencias necesarias para el uso de la biblioteca de tensorflow lite con la que se realizan aplicaciones de aprendizaje automático como lo es la aplicación implementada.

Raspberry Pi

Se ha mencionado anteriormente que se usará un raspberry pi como unidad para el análisis de emociones, este es un sistema embebido de pequeño tamaño con alta capacidad de procesamiento en comparación con su consumo de energía y precio[19]. En nuestro caso se utiliza un raspberry pi 2 el cual incluye las siguientes caracteristicas:

- Memoria RAM de 1 GB
- Capacidad de uso de memoria sd como almacenamiento, con un máximo de 32 GB
- Procesador Cortex A7 de 900 MHz
- 4 puertos USB además de un GPIO de 40 pines

- Puerto HDMI

Además este sistema permite instalar imágenes basadas en Linux, lo cual es esencial para el desarrollo del proyecto en la misma. Lo único necesario es preparar la imagen previamente mediante los pasos que se verán más adelante.

Dependencias del software

Para llevar a cabo el proyecto hay una serie de dependencias que se deben tener en cuenta para su desarrollo y con las que tanto la máquina host, la aplicación y la imagen de Yocto Project necesitan para su correcto funcionamiento.

Para la realización del proyecto se debe procurar que el computador posea una distribución de Linux como sistema operativo, que cuente al menos 4 GB de memoria RAM y suficiente espacio libre para el desarrollo de la imagen en Yocto Project que como mínimo es 50 GB.

Para el uso de Yocto Project se debe contar con los siguientes requerimientos y versiones [15].

- Git 1.8.3.1 o mayor.
- tar 1.28 o mayor.
- Python 3.6.0 o mayor.
- gcc 5.0 o mayor.

Además de ello, Yocto Proyect debe contar con el repositorio poky más actualizado el cual corresponde a Hardknott 3.3.3, los drivers de la cámara USB logitech a utilizar y las bibliotecas utilizadas para la aplicación las cuales corresponde a OpenCV y tensorflow principalmente y otras como numpy, time y datatime. Además, la imagen debe contar con las capas de meta-openembedded la cual ayuda al uso de python, la capa de meta-tensorflow-lite para la biblioteca de igual nombre y la capa meta-raspberry que ayuda con la implementación de dicha imagen en nuestro sistema embebido como capas principales.

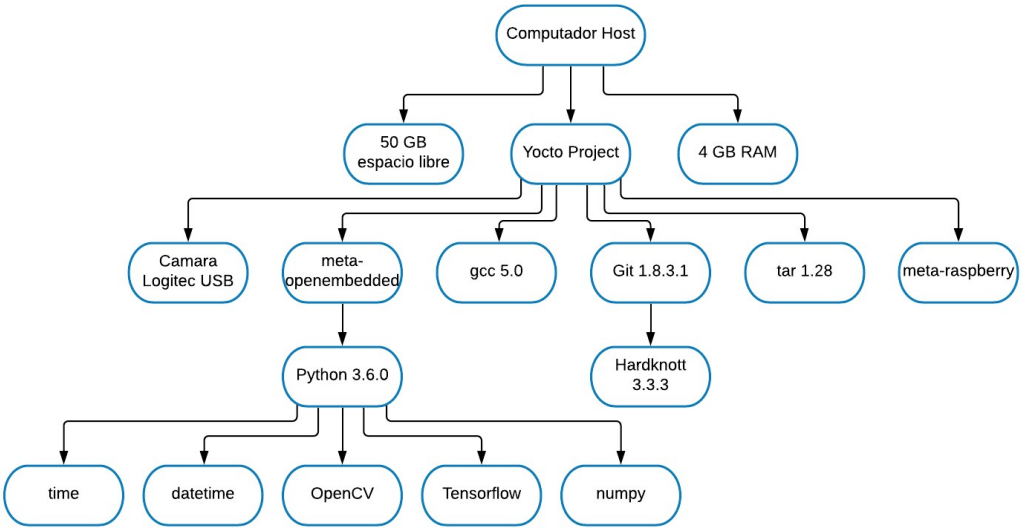

Figura 1: Árboles de dependencias del Computador Host. Fuente: Propia

Finalmente y gracias a la imagen sintetizada nuestra raspberry pi podrá tener las dependencias necesarias para la ejecución del programa los cuales corresponden a los drivers de la cámara logitech, python y sus bibliotecas OpenCV, tensorflow lite y numpy. Se toma en cuenta que la raspberry pi ocupa de al menos 4 GB de memoria RAM y espacio libre suficiente para la imagen de *yocto project*.

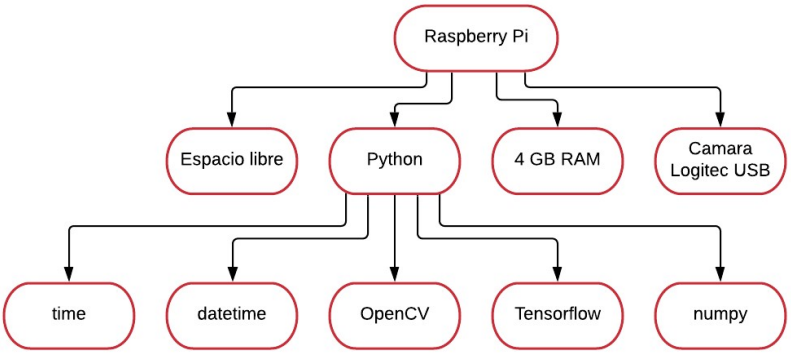

Figura 2: Árboles de dependencias de la Raspberry pi. Fuente: Propia

Tutorial

Instalación de software en Ubuntu

Para correr el código de detección de emociones en Ubuntu es necesario instalar python con las siguientes bibliotecas:

1. OpenCV
2. Numpy
3. Tensorflow Lite

Las otras bibliotecas utilizadas (time y datetime) están incluidas en python por defecto. Además, para la conversión del modelo previamente entrenado es necesaria la instalación de tensorflow y la descarga del modelo en formato .h5 (formato de modelos para tensorflow).

Instalación de Python

Para instalar python es necesario correr los siguientes comandos en la terminal:

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install python3.8
```

Instalación de paquetes

Para instalar los paquetes de OpenCV, Numpy y Tensorflow se puede utilizar pip, usando en la terminal los comandos:

```
pip3 install Opencv-python
pip3 install numpy
pip3 install Tensorflow
```

Finalmente para instalar tensorflow lite es necesario añadir la lista de repositorios y luego instalar el paquete, eso de la siguiente forma[8]:

```
echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main"
sudo tee /etc/apt/sources.list.d/coral-edgetpu.list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg
sudo apt-key add -
sudo apt-get update
sudo apt-get install python3-tflite-runtime
```

Conversión del modelo

Tras haber instalado todos los paquetes se debe ejecutar la conversión del modelo a uno en formato .tflite, para ello solo es necesario correr el código conversión.py que se encuentra en el repositorio del proyecto, tras hacerlo se creará un archivo con nombre model.tflite.

Finalmente, lo único necesario es correr el código Emotions.py el cual está adecuado para el uso de Tensorflow lite.

Creación de la imágen en Yocto Project

1. Primeramente se debe clonar el repositorio con el *build system* o poky más reciente, en este caso el Hardknott 3.3.3 con el siguiente comando obtenido de [16].

   git clone -b hardknott git://git.yoctoproject.org/poky.git

2. Seguidamente se clonan los repositorios que contienen las capas ya creadas de utilidad para el proyecto las cuales corresponden a meta-openembedded, meta-python, meta-raspberry obtenidas de [17] y tensorflow-lite obtenida de [18].

   git://git.openembedded.org/meta-openembedded

   git clone git://git.yoctoproject.org/meta-raspberrypi

   git clone https://github.com/NobuoTsukamoto/meta-tensorflow-lite.git

   Se recuerda comprobar el archivo local.conf de cada capa para comprobar que sea compatible con el poky de hardknott en la línea de código:
   $LAYERSERIES\_COMPAT\_\{nombre\_receta\} = "hardknott"$.

   En nuestro caso las capas de meta-raspberry y meta-tensorflow-lite se agregaron dentro la carpeta meta-openembedded.

   Además se agrega la capa creada para la aplicación meta-emotiondetect.

3. Ahora se procede a inicializar el ambiente con el comando para crear la carpeta build con archivos como blayers.bb y layer.conf

   source oe-init-build-env

4. Para trabajar en la raspberry pi se debe cambiar el tipo de máquina con el que va a trabajar por el de la raspberry pi de modo que escribimos la siguiente línea al inicio del local.conf después de comentar la que viene por defecto correspondiente a qemux86-64.

   $MACHINE?? = "raspberrypi2"$

Además de agregar los siguientes parámetros al final del archivo para la generación de la imagen en raspberry pi.

$IMAGE\_FSTYPES\_DEBUGFS+=$"$tar.xzext4rpi-sdimg$"

$IMAGE\_FSTYPES+=$"$tar.xzext4rpi-sdimg$"

5. Para añadir las capas deseadas se puede utilizar el comando *bitbake-layers add-layer path de la capa*, el cual añadirá automáticamente la dirección de la capa al archivo bblayers.conf o copiando las direcciones directamente al bblayers.conf.

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

   bitbake-layers add-layer ../meta-openembedded/meta-python

   bitbake-layers add-layer ../meta-openembedded/meta-networking

   bitbake-layers add-layer ../meta-openembedded/meta-filesystem

   bitbake-layers add-layer ../meta-openembedded/meta-raspberry

   bitbake-layers add-layer ../meta-openembedded/meta-tensorflow-lite

   bitbake-layers add-layer ../meta-emotiondetect

6. Ahora en el local.conf del build se deben agregar las dependencias a usar al final, en nuestro caso de la capa meta-python agregamos la dependencia de opencv, para la capa de tensorflow-lite la dependencia python3-tensorflow-lite, para el uso de ssh agregamos la misma y finalmente la del la aplicación de reconocimiento de emociones emotiondetect

   $IMAGE\_INSTALL\_append+=$" *opencv*"

   $IMAGE\_INSTALL\_append+=$" *opencv-samples*"

   $IMAGE\_INSTALL\_append+=$" *python3-tensorflow-lite*"

   $IMAGE\_INSTALL\_append+=$" *ssh*"

   $IMAGE\_INSTALL\_append+=$" *emotiondetect*"

7. Al añadir las capas necesarias se debe comenzar con el proceso de cocinar la imagen, esto mediante el comando:

   time bitbake core-image-sato

   en el ambiente de construcción, recordar que este se inicia mediante:

   source oe-init-build-env

Tras esto solo es necesario esperar a que se complete la construcción del sistema y emularlo mediante qemu.

8. Tras terminar la construcción en yocto, es necesario correr el comando:

   sudo dd if=./core image-sato-raspberrypi3.rpi-sdimg of=/dev/sdb

   Este comando coloca la imagen cocinada en la tarjeta SD, donde Y descomprimir el archivo creado tras el comando. Seguido de esto, lo único necesario es insertar dicha tarjeta en el pi, conectar el sistema y después de que este inicie conectar la cámara, un monitor y un teclado al embebido. Al cumplir lo anterior, se busca el archivo Emotions.py en la ubicación que se muestra a continuación

   cd ../../..
   cd usr/bin

   Una vez encontrado se podrá correr el programa de reconocimiento de emociones en el raspberry pi mediante:

   python3 Emotions.py

   También es importante que para exportar el archivo generado (Emotios_File.csv), necesitaríamos previamente montar un servidor SSH, en nuestro caso se hizo en la máquina de Ubuntu, está fue la que nos sirvió como OpenSSH y luego se exporta desde Raspberry Pi con un comando que se explica de la siguiente forma:

   En Ubuntu se dan los siguientes pasos:

   1. Se instala el OpenSSH con el comando: *sudo apt-get install openssh-server* .

   2. Se activa el servicio con el comando: *sudo systemctl enable ssh.*

   3. Se inicia con el comando: *sudo systemctl start ssh.*

   4. Se crea una carpeta para copiar el archivo desde la Raspberry Pi (/home/ubuntu/emotion).

   5. En el Raspberry Pi se ejecuta el comando para sacar el archivo al servidor OpenSSH en la maquina de Ubuntu:

   scp Emotions_File.csv ubuntu@10.10.1.124:/home/ubuntu/emotion

   Tomando en cuenta que ubuntu@ es el usuario y la IP es la del server donde vamos a copiar el archivo junto con su Path (carpeta de destino).

Referencias

[1] S. Luchetti, "Sistemas embebidos y sus características — Conceptos Fundamentales", 2021. [Online] Dsiponible:https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas

[2] M. Oliva, "Inteligencia Artificial Emocional", s.f. [Online]. Disponible en: https://www.fundacionmapfre.org/premios-ayudas/premios/premios-fundacion-mapfre-innovacion-social/tendencias/inteligencia-artificial-emocional/

[3] J. Tados Solano, J. Neira Parra y R. Mur Artal, "Visión por computador", Zaragosa, españa, 2015. Disponible en: https://zaguan.unizar.es/record/53587?ln=es

[4] Vector ITC,"Edge AI: El futuro de la Inteligencia Artificial", 2020. [Online]. Disponible en: https://www.vectoritcgroup.com/tech-magazine/artificial-intelligence/edge-ai-el-futuro-de-la-inteligencia-artificial/

[5] H. Fahrudin, "FastEmotRecognition", Indonesia, 2019. Disponible en: https://github.com/hfahrudin/FastEmotRecognition

[6] G. Solano, "Detección de rostros con Haar Cascades Python – OpenCV", 2020. [Online] Disponible en: https://omes-va.com/deteccion-de-rostros-con-haar-cascades-python-opencv/.

[7] TensorFlow, "Por qué TensorFlow", s.f. [Online] Disponible en: https://www.tensorflow.org/?hl=es-419.

[8] TensorFlow, "Tensorflow Lite", s.f. [Online] Disponible: https://www.tensorflow.org/lite/guide?hl=es-419.

[9] TensorFlow, "Post-training quantization", s.f [Online] Disponible en: https://www.tensorflow.org/lite/performance/post_training_quantization?hl=en.

[10] Yocto Project, "The Yocto Project. It's not an embedded Linux distribution, it creates a custom one for you", s.f. [Online] Disponible en: https://www.yoctoproject.org

[11] P. Pascual Vázquez, "Integración de EPICS en una distribución Linux para la plataforma Raspberry-Pi", Proyecto fin de carrera, Universidad Politécnica de Madrid, Madrid, España, 2017. Disponible en: http://oa.upm.es/49301/1/PFC_PEDRO_PASCUAL_VAZQUEZ.pdf

[12] Linux Foundation, "Yocto Project Quick Start", s.f. [Online]. Disponible en: https://www.yoctoproject.org/docs/2.4.1/yocto-project-qs/yocto-project-qs.html#yp-resources

[13] wolfSSL Inc, "Beginners guide to writting a recipe for Openembedded and Yocto Projects", s.f. [Online]. Disponible en: https://www.wolfssl.com/docs/yocto-openembedded-recipe-guide/

[14] OpenCV team, "About", s.f. [Online]. Disponible en: https://opencv.org/about/

[15] The Linux Foundation, "2 Setting Up to Use the Yocto Project", s.f. [Online]. Disponible en: https://docs.yoctoproject.org/dev-manual/start.html

[16] Yocto Project, "Software: Downloads", s.f. [Online] Disponible en: https://www.yoctoproject.org/software-overview/downloads/

[17] OpenEmbedded, "OpenEmbedded Layer Index", s.f. [Online] Disponible en: http://layers.openembedded.org/layerindex/branch/master/layers/.

[18] N. Tsukamoto, "meta-tensorflow-lite", 2021. [Online] Disponible en: https://github.com/NobuoTsukamoto/meta-tensorflow-lite.

[19] Raspberry Pi Foundation, "What is a Raspberry Pi?", 2018. [Online] Disponible en: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/.