



**THE IMAGINATION UNIVERSITY PROGRAMME**

# **RVfpga-SoC Lab 5**

## **Executar o Tensorflow Lite no SweRVolf**



**Tabela 1. Termos do RVfpga**

Nome	Descrição
<b>Cursos</b>	
<b>RVfpga</b>	Um curso que mostra como usar o RVfpgaNexys e o RVfpgaSim, RISC-V system-on-chips (SoCs), para executar programas e ampliar o sistema adicionando periféricos (RVfpga Labs 1-10) e explorar o núcleo e o sistema de memória executando simulações, medindo o desempenho, adicionando instruções e modificando o sistema de memória (RVfpga Labs 11-20). Ao longo do curso, os utilizadores aprenderão a usar a toolchain RISC-V (compiladores e depuradores) e simuladores, o simulador Verilator HDL e o simulador de conjunto de instruções Whisper (ISS) da Western Digital.
<b>RVfpga-SoC</b>	Um curso que mostra como construir um SweRVolfX SoC a partir do zero usando blocos IP como o núcleo SweRV, memórias e periféricos. O curso também mostra como carregar um sistema operativo de tempo real (RTOS) Zephyr no SweRVolf e executar programas, incluindo o exemplo Hello-World do Tensorflow Lite, sobre o sistema operativo.
<b>Núcleos e SoCs</b>	
<b>SweRV EH1 Core</b>	Núcleo RISC-V comercial de código aberto desenvolvido pela Western Digital ( <a href="https://github.com/chipsalliance/Cores-SweRV">https://github.com/chipsalliance/Cores-SweRV</a> ).
<b>SweRV EH1 Core Complex</b>	Núcleo SweRV EH1 com memória adicional (ICCM, DCCM e cache de instruções), controlador de interrupção programável (PIC), interfaces de barramento e unidade de depuração ( <a href="https://github.com/chipsalliance/Cores-SweRV">https://github.com/chipsalliance/Cores-SweRV</a> ).
<b>SweRVolfX</b>	É o System-on-a-Chip usado no curso RVfpga. É uma extensão do SweRVolf. <b>SweRVolf</b> ( <a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a> ): Um SoC de código aberto criado em torno do SweRV EH1 Core Complex. Adiciona uma boot ROM, uma interface UART, um controlador de sistema, uma interconexão (AXI Interconnect, Wishbone Interconnect, e AXI-to-Wishbone bridge) e um controlador SPI. <b>SweRVolfX</b> : adiciona quatro novos periféricos ao SweRVolf: um GPIO, um PTC, um SPI adicional e um controlador para os 8 mostradores de 7 segmentos de 8 dígitos.
<b>RVfpgaNexys</b>	O SoC SweRVolfX foi realizado para a placa Nexys A7 e seus periféricos. Ele adiciona uma interface DDR2, unidade CDC (clock domain crossing), lógica BSCAN (para a interface JTAG) e gerador de relógio. O RVfpgaNexys é o mesmo que o SweRVolf Nexys ( <a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a> ), exceto que o último é baseado no SweRVolf.

<b>RVfpgaSim</b>	<p>O SoC SweRVolfX tem um encapsulamento de testbench e memória AXI destinados a simulação.</p> <p>O RVfpgaSim é o mesmo que o SweRVolf Sim (<a href="https://github.com/chipsalliance/Cores-SweRVolf">https://github.com/chipsalliance/Cores-SweRVolf</a>), exceto pelo fato de que o último é baseado no SweRVolf.</p>
------------------	--

## 1. Introdução

Neste laboratório, irá criar um projeto do Tensorflow Lite para o Zephyr (um sistema operativo de tempo-real) e, em seguida, executar esse programa do Zephyr no SweRVolf. Semelhante ao que viu no laboratório anterior, executará um programa do Tensorflow sobre o Zephyr em vez de um programa básico em linguagem C ou Assembly.

### 1. Breve Descrição do TensorFlow Lite

O TensorFlow Lite é um conjunto de ferramentas que permite aprendizagem automática no dispositivo, ajudando os programadores a executarem os seus modelos em dispositivos móveis, embebidos e IoT. Ele comprime um modelo do TensorFlow num modelo .tflite que tem um tamanho binário pequeno. Isso permite correr algoritmos de aprendizagem automática no dispositivo e usa a aceleração de hardware para melhorar o desempenho.

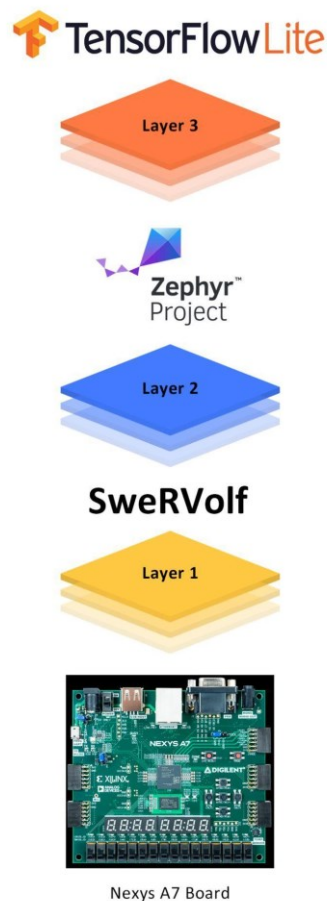
Os seus principais recursos são:

- Otimizado para aprendizagem automática no dispositivo, abordando cinco restrições principais: latência (não há tempo de ida e volta para o servidor), privacidade (nenhum dado pessoal sai do dispositivo), conectividade (não é necessária conectividade com a Internet), tamanho (modelo reduzido e tamanho binário) e consumo de energia (inferência eficiente e falta de conexões de rede).
- Suporte para várias plataformas, abrangendo dispositivos Android e iOS, Linux embebido e microcontroladores.
- O suporte para diversas linguagens inclui Java, Swift, Objective-C, C++ e Python.
- Alto-desempenho, com aceleração de hardware e otimização de modelos.
- Exemplos de ponta-a-ponta para tarefas comuns de aprendizagem automática, como classificação de imagens, detecção de objetos, estimativa de pose, resposta a perguntas, classificação de texto etc., em várias plataformas.

Para obter mais informações, acesse <https://www.tensorflow.org/lite/microcontrollers>

## SweRVolf e Tensorflow Lite

A Figura 1 ilustra as camadas hierárquicas na parte superior da placa Nexys A7 que implementaremos neste laboratório.



**Figura 1. Camadas na parte superior da placa FPGA**

As etapas para executar um programa TensorFlow Lite na placa Nexys A7 são ligeiramente diferentes das do Lab 4.

### **Etapas 1. Configure o SweRVolf na placa FPGA**

Primeiro, configura-se o SweRVolf, o sistema RISC-V implementado para FPGA, para a placa FPGA Nexys A7. O SweRVolf é configurado na placa carregando o Bitstream usando o PlatformIO ou usando o comando de execução do FuseSoC, que carrega o Bitstream gerado para a placa se ela estiver conectada.

### **Etapas 2. Criar Programas do Tensorflow**

Nesta etapa, irá criar uma aplicação Tensorflow Lite para o Zephyr. O Zephyr RTOS é criado como parte dessa criação. O resultado é um ficheiro elf.

### **Etapas 3. Carregue os programas no SweRVolf.**

Nesta etapa, carregará o ficheiro elf gerado durante a Etapa 2 no SweRVolf.

## 2. Requisitos

Para concluir este laboratório, precisará instalar o seguinte:

- Vivado 2019.2 Web Pack (Consulte o Guia de Instalação (Página No.04))
- Verilator (v4.106) (Consulte o Guia de Instalação (página nº 09))
- FuseSoC (Consulte o Guia de instalação (página nº 10))
- OpenOCD (versão RISC-V) (Consulte o Guia de Instalação (página nº 10))
- Zephyr Pré-requisitos (Consulte o Guia de Instalação (página nº 11))
- Zephyr SDK (v0.12.4) (Consulte o Guia de Instalação (página nº 12))
- PuTTY (consulte o Guia de Instalação (página nº 12))

**IMPORTANTE:** antes de iniciar os Labs RVfpga-SoC, é altamente recomendado concluir o Guia de Instalação do RVfpga-SoC.

Por exemplo, se ainda não o fez, instale o Vivado e o Verilator da Xilinx seguindo as instruções do Guia de Instalação do RVfpga-SoC. Certifique-se de ter copiado a pasta RVfpga-SoC que descarregou do programa universitário da Imagination para o computador.

## 3. Exemplo do Hello World do Tensorflow

Neste laboratório, apenas configurará o ambiente do Tensorflow e irá executar uma operação simples de tensor Hello-World.

O exemplo Hello-World foi criado para demonstrar os fundamentos básicos do uso do TensorFlow Lite para microcontroladores. Esse programa treina e executa um modelo que replica uma função sinusoidal, ou seja, recebe um único número como entrada e gera o valor sinusoidal desse número.

Para obter mais informações, acesse a documentação oficial do TensorFlow no [link](#).

## 4. Configurando o Ambiente Para o Tensorflow

Abra o terminal do Ubuntu e conclua as etapas a seguir:

**Etapa 1.** Navegue até o diretório "**SweRVolf**". Tem de definir as seguintes variáveis de shell. Para fazer isso, execute os seguintes comandos:

- `export WORKSPACE=$(pwd)`
- `export SWERVOLF_ROOT=$WORKSPACE/fusesoc_libraries/swervolf`
- `export ZEPHYR_BASE=$WORKSPACE/zephyr`

também pode inserir o comando "`printenv <variable-name>`" na janela do terminal para verificar se as variáveis do shell foram definidas com êxito ou não.

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ export WORKSPACE=$(pwd)
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ export SWERVOLF_ROOT=$(pwd)/fusesoc_libraries/swervolf
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ export ZEPHYR_BASE=$WORKSPACE/zephyr
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

**Figura 2. Definir as variáveis do shell**

## Etapa 2. Clone o repositório do GitHub do Tensorflow.

➤ `git clone https://github.com/tensorflow/tensorflow`

```
hanza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ git clone https://github.com/tensorflow/tensorflow
Cloning into 'tensorflow'...
remote: Enumerating objects: 1155704, done.
remote: Counting objects: 100% (193/193), done.
remote: Compressing objects: 100% (141/141), done.
remote: Total 1155704 (delta 62), reused 118 (delta 51), pack-reused 1155511
Receiving objects: 100% (1155704/1155704), 683.05 MiB | 157.00 KiB/s, done.
Resolving deltas: 100% (942622/942622), done.
Checking out files: 100% (25049/25049), done.
```

Figura 3. Tensorflow

Agora, navegue até o diretório "tensorflow".

➤ `cd tensorflow`

```
:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ cd tensorflow/
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$
```

Figura 4. Navegue até o diretório "tensorflow"

Faça "check out" da "v2.5.0" do repositório com o seguinte comando:

➤ `checkout do git v2.5.0`

```
hanza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$ git checkout -b v2.5.0
Switched to a new branch 'v2.5.0'
hanza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$
```

Figura 5. checkout do git

**Etapa 3.** Para adicionar suporte ao Zephyr SweRVolf no TensorFlow, precisa de copiar alguns ficheiros nesse repositório do TensorFlow.

O primeiro ficheiro é o ficheiro "**Makefile.inc**" do exemplo hello\_world. Navegue até o seguinte caminho para copiar o "**Makefile.inc**":

- `[RVfpgaSoCPath]/RVfpgaSoC/Labs/LabResources/Lab5/Makefile.inc`

Agora, cole o ficheiro "**Makefile.inc**" no seguinte local (consulte a Figura 6)

- `[RVfpgaSoCPath]/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/examples/hello_world/zephyr_riscv/`

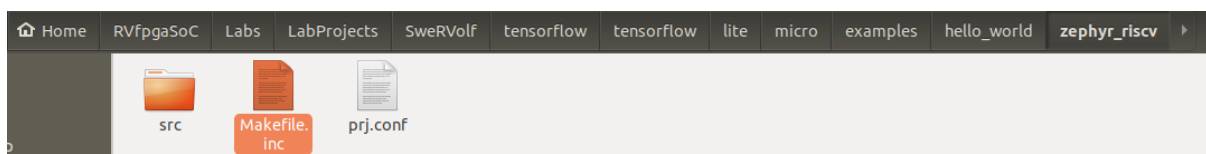


Figura 6. Makefile.inc

**Nota:** Se não conseguir encontrar a pasta "example" no seguinte caminho:

[RVfpgaSoCPath]/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow-lite/micro/examples

significa que não mudou fez o "check out" da v2.5.0 do tensorflow.

pode mudar para a versão 2.5.0 escrevendo o seguinte comando no terminal:

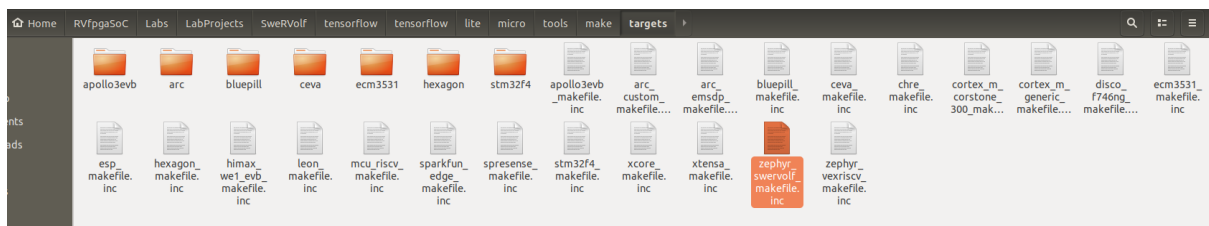
```
checkout do git v2.5.0
```

O segundo ficheiro é o ficheiro "zephyr\_swervolf\_makefile.inc". Navegue até o seguinte caminho para copiar "zephyr\_swervolf\_makefile.inc":

- [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabResources/Lab5/zephyr\_swervolf\_makefile.inc

Agora, cole o ficheiro "zephyr\_swervolf\_makefile.inc" no seguinte local (ver Figura 7)

- [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow-lite/micro/tools/make/targets/



**Figura 7. zephyr\_swervolf\_makefile.inc**

**Etapa 4.** Instale os pacotes necessários.

Navegue até o diretório "WORKSPACE" usando o seguinte comando:

```
> cd ..
```

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$ cd ..
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

**Figura 8. Navegue até o diretório WORKSPACE**

Instale os pacotes necessários com o seguinte comando:

```
> sudo apt install git cmake ninja-build gperf ccache dfu-util
device-tree-compiler wget python python3-pip python3-
setuptools python3-tk python3-wheel xz-utils file make gcc
gcc-multilib locales tar curl unzip xxd make autoconf g++ flex
bison virtualenv
```



```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ sudo apt install git cmake ninja-build
gperf ccache dfu-util device-tree-compiler wget python python3-pip python3-setuptools python3-tk
python3-wheel xz-utils file make gcc gcc-multilib locales tar curl unzip xxd make autoconf g++
flex bison virtualenv
[sudo] password for hamza:
Reading package lists... Done
Building dependency tree
Reading state information... Done
autoconf is already the newest version (2.69-11).
bison is already the newest version (2:3.0.4.dfsg-1build1).
ccache is already the newest version (3.4.1-1).
device-tree-compiler is already the newest version (1.4.5-3).
flex is already the newest version (2.6.4-6).
make is already the newest version (4.1-9.1ubuntu1).
python is already the newest version (2.7.15-rc1-1).
python3-setuptools is already the newest version (39.0.1-2).
xz-utils is already the newest version (5.2.2-1.3).
dfu-util is already the newest version (0.9-1).
```

**Figura 9. Instalar pacotes**

**Etapa 5.** Crie um ambiente virtual.

Primeiro, navegue até o diretório zephyr usando o seguinte comando:

➤ `cd zephyr`

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ cd zephyr/
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$
```

**Figura 10. Navegue até o diretório zephyr**

Crie um ambiente virtual dentro do diretório zephyr usando o seguinte comando:

➤ `virtualenv venv-zephyr`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$ virtualenv venv-zephyr
created virtual environment CPython3.6.9.final.0-64 in 374ms
creator CPython3Posix(dest=/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/venv-zephyr, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/hamza/.local/share/virtualenv)
added seed packages: pip==21.0.1, setuptools==54.2.0, wheel==0.36.2
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$
```

**Figura 11. Criando o venv-zephyr**

**Etapa 6.** Escreva o seguinte comando para ativar o ambiente virtual criado na última etapa.

➤ `source venv-zephyr/bin/activate`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$ source venv-zephyr/bin/activate
(venv-zephyr) hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$
```

**Figura 12. Ativação do venv-zephyr**

**Etapa 7.** Instale os pacotes necessários listados no ficheiro "requirements.txt" usando o seguinte comando.

➤ `pip3 install -r scripts/requirements.txt`

```
(venv-zephyr) hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$ pip3 install
-r scripts/requirements.txt
Ignoring windows-curses: markers 'sys_platform == "win32"' don't match your environment
Collecting pyelftools>=0.26
  Using cached pyelftools-0.27-py2.py3-none-any.whl (151 kB)
Collecting PyYAML>=5.1
  Using cached PyYAML-5.4.1-cp36-cp36m-manylinux1_x86_64.whl (640 kB)
Collecting canopen
  Using cached canopen-1.2.1-py3-none-any.whl (52 kB)
Collecting packaging
  Using cached packaging-20.9-py2.py3-none-any.whl (40 kB)
Collecting progress
```

**Figura 13. Instalação dos pacotes necessários**

Agora pode fechar este terminal e retornar ao terminal principal, onde criará o exemplo "hello\_world".

## 5. Criando o Exemplo Hello World para Swervolf

Nesta seção, irá criar o exemplo "hello\_world" para o SweRVolf. Irá gerar os ficheiros "zephyr.bin" e "zephyr.elf" para o exemplo "hello\_world".

**Etapa 1.** Primeiro, navegue até o diretório do tensorflow.

```
> cd ../tensorflow/
```

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr$ cd ../tensorflow/
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$
```

**Figura 14. Navegando até o diretório "tensorflow"**

**Etapa 2.** Para este laboratório, irá criar o Hello\_World para o SweRVolf. Isso é feito com o seguinte comando:

```
> make -f tensorflow/lite/micro/tools/make/Makefile
TARGET=zephyr_swervolf BUILD_TYPE=debug hello_world_bin
```

```
(venv-zephyr) hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow$ make -f tensorflow/lite/micro/tools/make/Makefile
TARGET=zephyr_swervolf BUILD_TYPE=debug hello_world_bin
--2021-05-20 19:44:08-- http://mirror.tensorflow.org/github.com/google/flatbuffers/archive/dca1252a9f9e37f126ab925fd385c807ab4f84e.zip
Resolving mirror.tensorflow.org (mirror.tensorflow.org)... 216.58.210.80, 2a00:1450:4018:803::2010
Connecting to mirror.tensorflow.org (mirror.tensorflow.org)|216.58.210.80|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1760478 (1.7M) [application/zip]
Saving to: '/tmp/tmp.kvIHESzFqo/dca1252a9f9e37f126ab925fd385c807ab4f84e.zip'

/tmp/tmp.kvIHESzFqo/dca1252a9f9 100%[=====] 1.68M 1.83MB/s in 0.9s

2021-05-20 19:44:09 (1.83 MB/s) - '/tmp/tmp.kvIHESzFqo/dca1252a9f9e37f126ab925fd385c807ab4f84e.zip' saved [1760478/1760478]

Cloning into 'tensorflow/lite/micro/tools/make/downloads/pigweed'...
remote: Sending approximately 15.02 MiB ...
remote: Counting objects: 33, done
remote: Finding sources: 100% (33/33)
remote: Total 22687 (delta 9753), reused 22681 (delta 9753)
Receiving objects: 100% (22687/22687), 15.01 MiB | 1.78 MiB/s, done.
Resolving deltas: 100% (9753/9753), done.
Note: checking out '47268dff45019863e20438ca3746c6c62df6ef09'.
```

**Figura 15. Compilação do exemplo hello\_world**

Isso levará alguns minutos, pois ele precisa de descarregar algumas ferramentas para as dependências. Quando terminar, verá algumas pastas criadas dentro de um caminho como

- tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/

Essas pastas contêm o projeto gerado e os ficheiros de origem.

```
[ 98%] Building C object zephyr/CMakeFiles/zephyr_final.dir/misc/empty_file.c.obj
[ 98%] Building C object zephyr/CMakeFiles/zephyr_final.dir/isr_tables.c.obj
[100%] Linking CXX executable zephyr.elf
Generating files from zephyr.elf for board: swervolf_nexys
make[3]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/
tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build'
[100%] Built target zephyr_final
make[2]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/
tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build'
make[1]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/
tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build'
```

**Figura 16. Compilação do exemplo hello\_world concluída**

Os binários resultantes (zephyr.bin e zephyr.elf) serão gerados no seguinte caminho:

- tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/build/zephyr

**Etapa 3.** Agora pode sair do ambiente virtual escrevendo o seguinte comando:

➤ deactivate

## 6. Executando o Exemplo Hello World no Verilator

Nesta seção, irá converter o ficheiro "zephyr.bin" num ficheiro ".hex" e, em seguida, o carregará como o ficheiro de RAM inicial ao executar o simulador do SweRVolf.

**Etapa 1.** Navegue até o diretório do projeto "hello\_world". Escreva o seguinte comando para entrar nesse diretório:

➤ cd  
tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/

```
/tensorflow$ cd tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/
/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world$
```

**Figura 17. "Caminho do projeto "hello\_world"**

**Etapa 2.** Converta o ficheiro ".bin" num ficheiro ".hex". Para criar o ficheiro ".hex", execute o seguinte comando no diretório hello\_world :

➤ python3 \$SWERVOLF\_ROOT/sw/makehex.py build/zephyr/zephyr.bin > /home/<username>/RVfpgaSoC/Labs/LabProjects/SweRVolf/hello\_world\_tensorflow.hex

(Substitua o <username> pelo nome do utilizador)

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world$ python3 $SWERVOLF_ROOT/sw/makehex.py build/zephyr/zephyr.bin > /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/hello_world_tensorflow.hex
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world$
```

**Figura 18. Converter ".bin" em ".hex"**

### Etapa 3. Navegue de volta para o diretório "WORKSPACE"

```
> cd $WORKSPACE
```

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world$ cd $WORKSPACE
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

**Figura 19. Instalação de pacotes no venv-zephyr**

### Etapa 4. Carregue o ficheiro ".hex" no simulador:

```
> fusesoc run --target=sim swervolf --
ram_init_file=hello_world_tensorflow.hex
```

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ fusesoc run --target=sim swervolf --ram_init_file=hello_world_tensorflow.hex
WARNING: Unknown item compilation_mode in section Xsim
INFO: Preparing ::cdc_utils:0.1-r1
INFO: Preparing chipsalliance.org:cores:SweRV_EH1:1.8
INFO: Preparing fusesoc:utils:generators:0.1.5
INFO: Preparing ::jtag_vpi:0-r5
INFO: Preparing pulp-platform.org::common_cells:1.20.0
INFO: Preparing ::simple_spi:1.6.1
INFO: Preparing ::uart16550:1.5.5-r1
INFO: Preparing ::verilog-arbiter:0-r3
INFO: Preparing ::wb_common:1.0.3
INFO: Preparing pulp-platform.org::axi:0.25.0
INFO: Preparing ::wb_intercon:1.2.2-r1
INFO: Preparing ::swervolf:0.7.3
INFO: Generating ::swervolf-intercon:0.7.3
Found master ifu
Found master lsu
Found master sb
Found slave io
Found slave ram
```

**Figura 20. Carregamento do ficheiro ".hex" no simulador**

Podemos ver o resultado do exemplo hello\_world (ver Figura 21). O programa imprime os valores X e Y da função "sine".

```
g++ jtagServer.o jtag_common.o tb.o verilated.o verilated_dpi.o verilated_vcd.o Vswervolf_core_tb_ALL.a -o Vswervolf_core_tb
make[1]: Leaving directory '/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/build/swervolf_0.7.3/sim-verilator'
INFO: Running
INFO: Running simulation
Loading RAM contents from /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/hello_world_tensorflow.hex
Releasing reset
*** Booting Zephyr OS build zephyr-v2.4.0 ***
x_value: 1.0*2^-127, y_value: 1.0*2^-127
x_value: 1.2566366*2^-2, y_value: 1.4910772*2^-2
x_value: 1.2566366*2^-1, y_value: 1.1183078*2^-1
x_value: 1.8849551*2^-1, y_value: 1.677462*2^-1
x_value: 1.2566366*2^0, y_value: 1.9316229*2^-1
```

**Figura 21. Saída do programa "hello\_world"**

Pressione "CTRL + C" para sair do programa.

## 7. Execução do Exemplo Hello World na Placa Nexys A7

Nesta seção, irá executar o projeto "hello\_world" na placa usando o OpenOCD.

**Etapa 1.** Ligue a placa Nexys A7 ao computador e, em seguida, execute o comando de configuração da FPGA no diretório Workspace.

```
> fusesoc run --target=nexys_a7 --run swervolf
```

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ fusesoc run --target=nexys_a7 --run swervolf
WARNING: Unknown item compilation_mode in section Xsin
INFO: Running
export HW_TARGET=; \
export JTAG_FREQ=; \
vivado -quiet -nolog -notrace -mode batch -source swervolf_0.7.3_pgm.tcl -tclargs xc7a100tcs9324-1 swervolf_0.7.3.bit
Fusesoc Xilinx FPGA Programming Tool
=====
```

**Figura 22. Executar a configuração da FPGA**

**Etapa 2.** Conecte o OpenOCD ao SweRVolf.

➤ `openocd -f $SWERVOLF_ROOT/data/swervolf_nexys_debug.cfg`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ openocd -f $SWERVOLF_ROOT/data/swervolf_nexys_debug.cfg
Open On-Chip Debugger 0.11.0-rc1+dev-01535-g3651cbfd (2021-01-30-12:10)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
DEPRECATED! use 'adapter driver' not 'interface'
DEPRECATED! use 'adapter speed' not 'adapter khz'
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi_tdo_sample_edge falling"
Info : clock speed 10000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x13631093 (mfg: 0x049 (Xilinx), part: 0x3631, ver: 0x1)
Info : datacount=2 progbufsize=0
Warn : We won't be able to execute fence instructions on this target. Memory may not always appear consistent. (progbufsize=0, impebreak=0)
Info : Examined RISC-V core; found 1 harts
Info : hart 0: XLEN=32, misa=0x40001104
Info : starting gdb server for riscv.cpu on 3333
Info : Listening on port 3333 for gdb connections
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
```

**Figura 23. OpenOCD conectado**

**Etapa 3.** Abra um novo terminal usando "CTRL + SHIFT + T" e conecte-se à sessão de depuração por meio do OpenOCD usando o seguinte comando:

➤ `telnet localhost 4444`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> 
```

**Figura 24. Ligação telnet localhost 4444**

O OpenOCD suporta o carregamento de ficheiros de programa ELF executando `load_image /path/to/file.elf`. Lembre-se de que o caminho é relativo ao diretório de onde o OpenOCD foi iniciado.

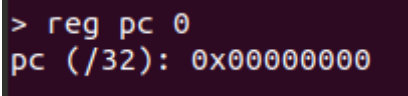
➤ `load_image`  
`tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build/zephyr/zephyr.elf`

```
> load_image tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr_swervolf_x86_64_debug/hello_world/build/zephyr/zephyr.elf
287072 bytes written at address 0x00000000
downloaded 287072 bytes in 27.439606s (10.217 KiB/s)
> 
```

**Figura 25. Carregando o ficheiro ".elf".**

Depois que o programa tiver sido carregado, defina o valor do Program Counter para o endereço zero usando o seguinte comando:

➤ `reg pc 0`

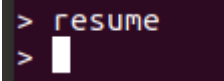


```
> reg pc 0
pc (/32): 0x00000000
```

**Figura 26. Colocar o Program Counter a zero**

Agora, inicie o programa usando este comando:

➤ `resume`

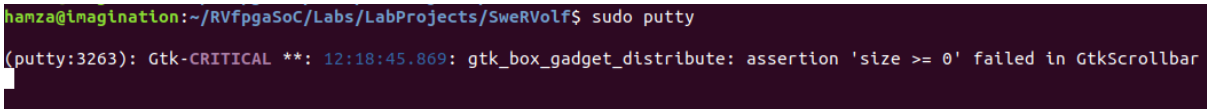


```
> resume
> 
```

**Figura 27. Iniciar o programa**

**Etapa 4.** Abra um novo terminal usando "CTRL + SHIFT + T". Abra o "PuTTY" usando o comando

➤ `sudo putty`



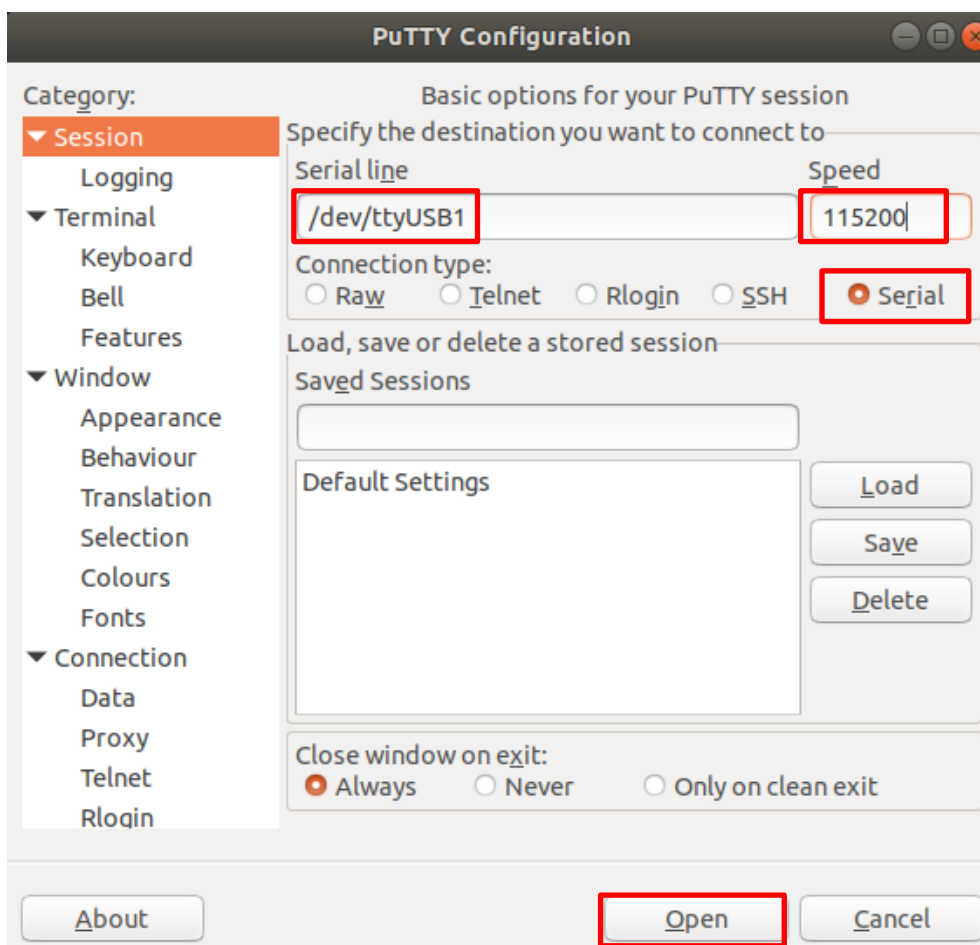
```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ sudo putty
(putty:3263): Gtk-CRITICAL **: 12:18:45.869: gtk_box_gadget_distribute: assertion 'size >= 0' failed in GtkScrollbar
```

**Figura 28. Abrir o PuTTY**

Usará o PuTTY aqui como consola de terminal série para a placa Nexys A7.

**Etapa 5.** Defina a seguinte configuração:

Selecione o tipo de conexão como "**Serial**", insira "**/dev/ttyUSB1**" como a Serial Line e defina a velocidade como "**115200**". Agora, clique em "Open" para iniciar a consola série.



**Figura 29. Configuração do PuTTY**

No consola série, pode ver o resultado do exemplo hello\_world (ver Figura 30).



```

/dev/ttyUSB1 - PuTTY
x_value: 1.0995567*2^2, y_value: -1.9316229*2^-1
x_value: 1.1780966*2^2, y_value: -1.1098361*2^0
x_value: 1.2566366*2^2, y_value: -1.965511*2^-1
x_value: 1.3351763*2^2, y_value: -1.4910772*2^-1
x_value: 1.4137159*2^2, y_value: -1.0674761*2^-1
x_value: 1.4922558*2^2, y_value: -1.4233011*2^-2
x_value: 1.0*2^-127, y_value: 1.0*2^-127
x_value: 1.2566366*2^-2, y_value: 1.4910772*2^-2
x_value: 1.2566366*2^-1, y_value: 1.1183078*2^-1
x_value: 1.8849551*2^-1, y_value: 1.677462*2^-1
x_value: 1.2566366*2^0, y_value: 1.9316229*2^-1
x_value: 1.5707957*2^0, y_value: 1.0420598*2^0
x_value: 1.8849551*2^0, y_value: 1.9146791*2^-1
x_value: 1.0995567*2^1, y_value: 1.6435742*2^-1
x_value: 1.2566366*2^1, y_value: 1.0674761*2^-1
x_value: 1.4137159*2^1, y_value: 1.8977352*2^-3
x_value: 1.5707957*2^1, y_value: 1.0844199*2^-7
x_value: 1.7278753*2^1, y_value: -1.2199725*2^-2
x_value: 1.8849551*2^1, y_value: -1.0674761*2

```

**Figura 30. Consola série**

Novamente, como visto na seção de simulação, o programa imprime as coordenadas "X" e "Y" da função sinusoidal que o modelo do TensorFlow está produzindo.

**Nota:** se não for possível abrir uma consola série, use a Serial Line `"/dev/ttyUSB0"`.

Em conclusão, neste laboratório, criou com êxito o exemplo "hello\_world" do TensorFlow como uma aplicação Zephyr e, em seguida, executou esse exemplo no SweRVolf.