

**THE IMAGINATION UNIVERSITY PROGRAMME**

**RVfpga-SoC Lab 5**

**Executar o Tensorflow Lite**

**no SweRVolf**

**Tabela 1. Termos do RVfpga**

|  |  |
| --- | --- |
| **Nome** | **Descrição** |
| **Cursos** | |
| **RVfpga** | 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. |
| **RVfpga-SoC** | 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. |
| **Núcleos e SoCs** | |
| **SweRV EH1 Core** | Núcleo RISC-V comercial de código aberto desenvolvido pela Western Digital (<https://github.com/chipsalliance/Cores-SweRV>). |
| **SweRV EH1 Core Complex** | Núcleo SweRV EH1 com memória adicional (ICCM, DCCM e cache de instruções), controlador de interrupção programável (PIC), interfaces de barramento e unidade de depuração (<https://github.com/chipsalliance/Cores-SweRV>). |
| **SweRVolfX** | É o System-on-a-Chip usado no curso RVfpga. É uma extensão do SweRVolf.  **SweRVolf** (<https://github.com/chipsalliance/Cores-SweRVolf>): 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.  **SweRVolfX**: 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. |
| **RVfpgaNexys** | 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 (<https://github.com/chipsalliance/Cores-SweRVolf>), exceto que o último é baseado no SweRVolf. |
| **RVfpgaSim** | O SoC SweRVolfX tem um encapsulamento de testbench e memória AXI destinados a simulação.  O RVfpgaSim é o mesmo que o SweRVolf Sim (<https://github.com/chipsalliance/Cores-SweRVolf>), exceto pelo fato de que o último é baseado no SweRVolf. |

# 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.

**Etapa 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.

**Etapa 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.

**Etapa 3. Carregue os programas no SweRVolf.**

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

# 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.

# 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, aceda à documentação oficial do TensorFlow no [link](https://www.tensorflow.org/lite/microcontrollers/get_started_low_level).

# Configurando o Ambiente Para o Tensorflow

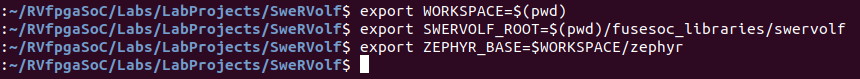
Abra o terminal do Ubuntu e conclua as etapas a seguir:

**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

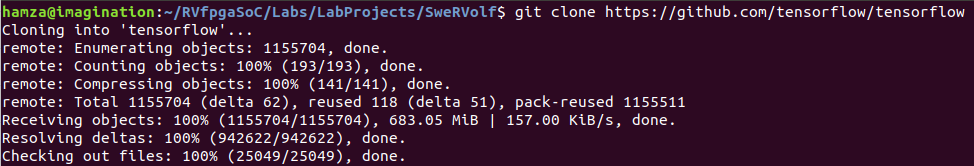
também pode inserir o comando "printenv <variable-name>" na janela do terminal para verificar se as variáveis do shell foram definidas com êxito ou não.



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

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

* git clone https://github.com/tensorflow/tensorflow

****

**Figura 3. Tensorflow**

Agora, navegue até o diretório "tensorflow".

* cd 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



**Figura 5. checkout do git**

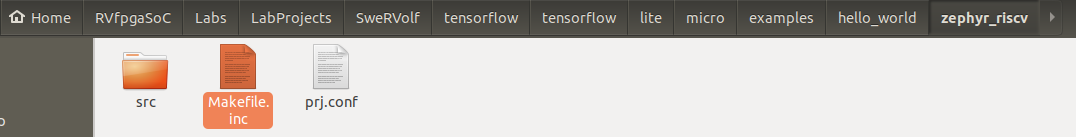
**Etapa 3.** Para adicionar suporte ao Zephyr SweRVolf no TensorFlow, precisa de copiar alguns ficheiros nesse repositório do TensorFlow.

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

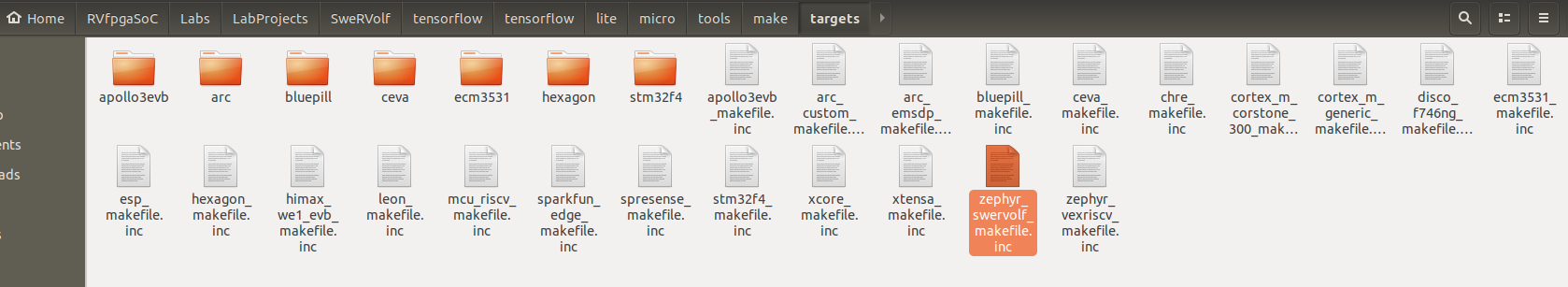
O segundo ficheiro é o ficheiro "**zephyr\_swervolf\_makefile.inc**". Navegue até o seguinte caminho para copiar "**zephyr\_swervolf\_makefile.inc**":

* [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:

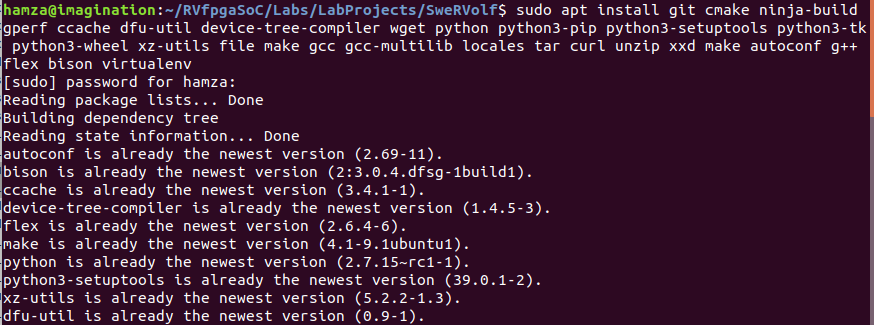
* cd ..



**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



**Figura 9. Instalar pacotes**

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

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

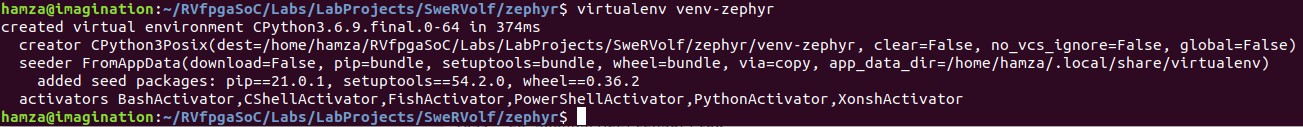
* cd zephyr

****

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

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

* virtualenv venv-zephyr



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

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

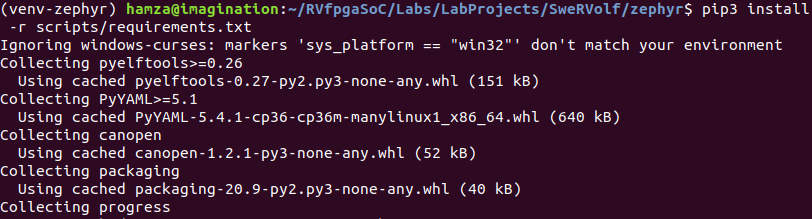
* source venv-zephyr/bin/activate



**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



**Figura 13. Instalação dos pacotes necessários**

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

# 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/



**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

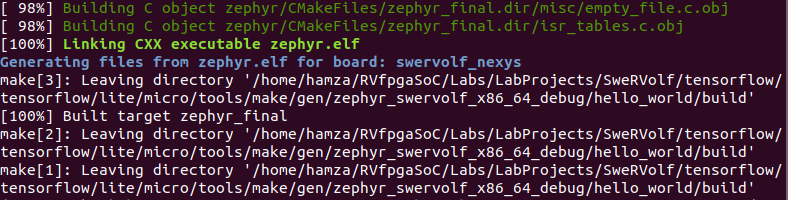


**Figura 15. Compilação do exemplo hello\_world**

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.



**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

# 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/

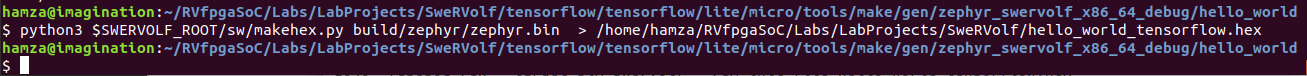


**Figura 17. "Caminho do projeto "hello\_world**

**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)



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

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

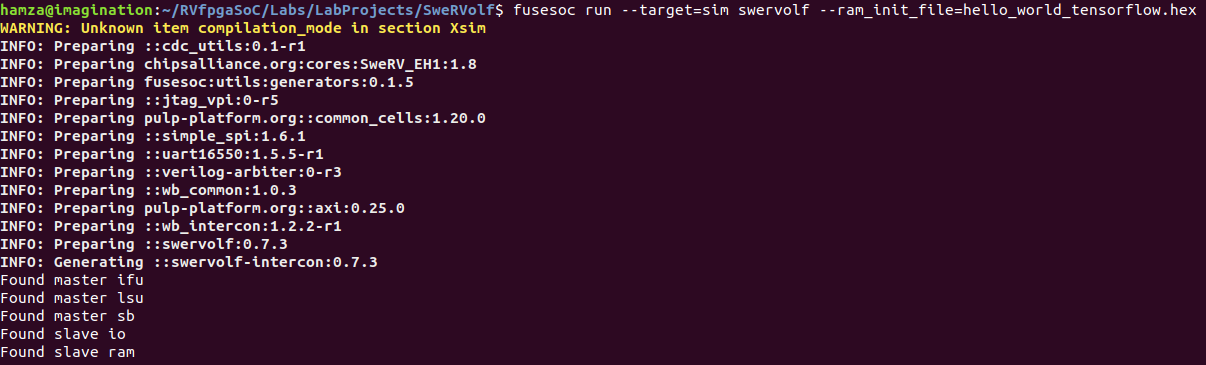
* cd $WORKSPACE



**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



**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".



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

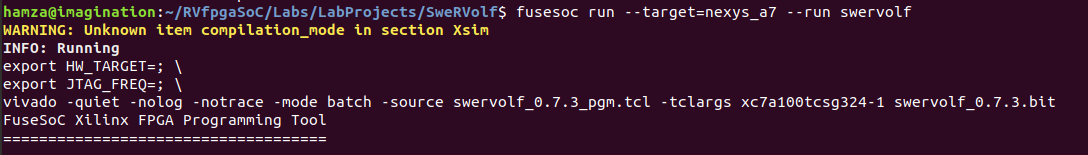
Pressione "CTRL + C" para sair do programa.

# 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.

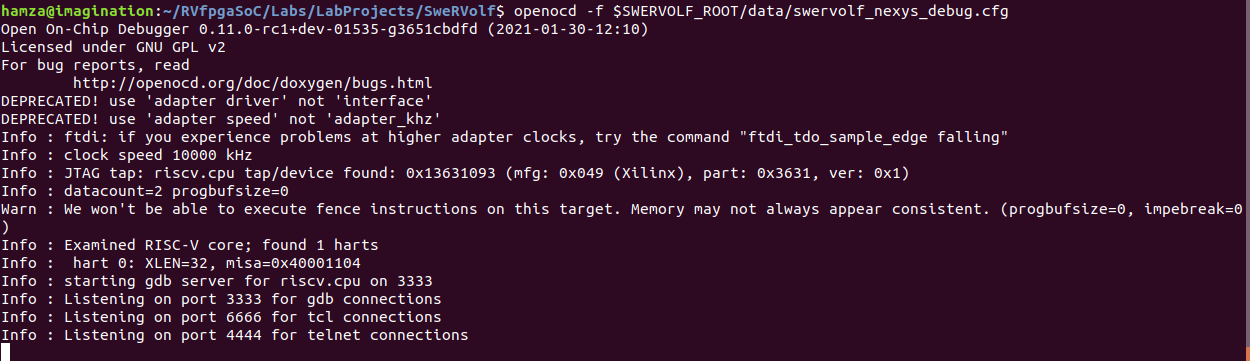
* fusesoc run --target=nexys\_a7 --run swervolf



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

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

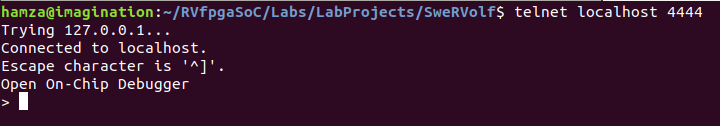
* openocd -f $SWERVOLF\_ROOT/data/swervolf\_nexys\_debug.cfg



**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:

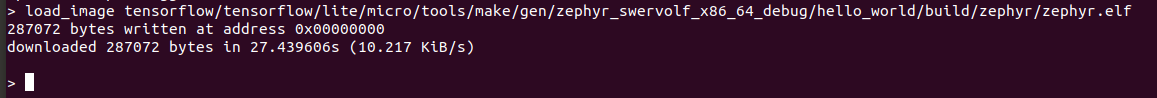
* telnet localhost 4444



**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



**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



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

Agora, inicie o programa usando este comando:

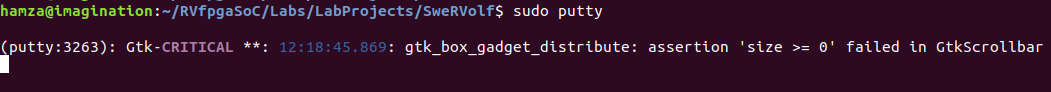
* resume



**Figura 27. Iniciar o programa**

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

* sudo putty

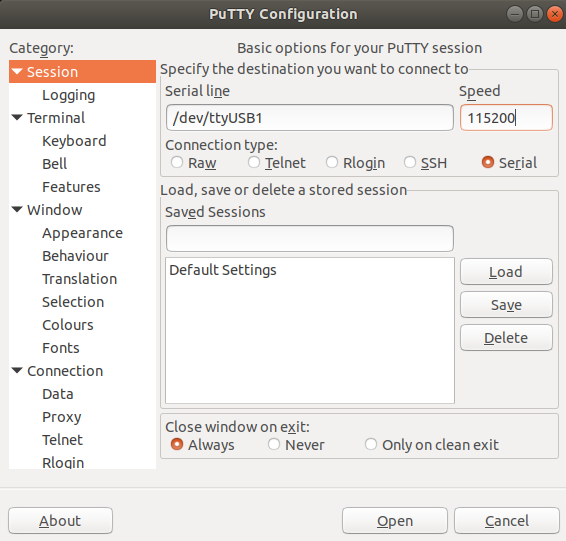


**Figura 28. Abrir o PuTTY**

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

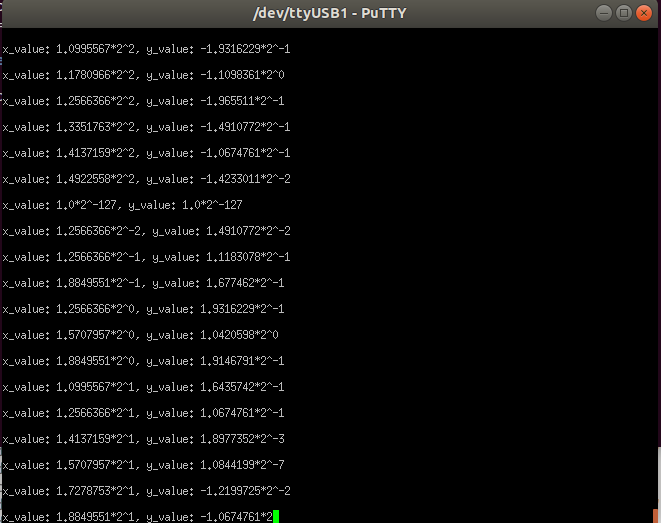
**Etapa 5**. Defina a seguinte configuração:

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).



**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.