

**THE IMAGINATION UNIVERSITY PROGRAMME**

**RVfpga-SoC Lab 5**

**SweRVolf에서**

**Tensorflow Lite 실행**

**표 1. RVfpga 용어**

|  |  |
| --- | --- |
| **Name** | **Description** |
| **Courses** | |
| **RVfpga** | RVfpgaNexys 및 RVfpgaSim, RISC-V SoC(System-on-Chip)를 사용하여 프로그램을 실행하고 주변 장치를 추가하여 시스템을 확장하고(RVfpga Labs 1-10), 시뮬레이션, 성능 측정, 명령 추가 및 메모리 시스템 수정(RVfpga Labs 11-20)을 실행하여 코어 및 메모리 시스템을 탐색하는 방법을 보여주는 과정입니다. 과정 전반에 걸쳐 사용자는 RISC-V 툴체인(컴파일러 및 디버거) 및 시뮬레이터, Verilator HDL 시뮬레이터 및 Western Digital의 Whisper 명령어 세트 시뮬레이터(ISS)를 사용하는 방법도 보여줍니다. |
| **RVfpga-SoC** | SweRV 코어, 메모리 및 주변 장치와 같은 빌딩 블록을 사용하여 처음부터 SweRVolfX SoC의 하위 집합을 구축하는 방법을 보여주는 과정입니다. 이 과정은 또한 Zephyr 실시간 운영 체제(RTOS)를 SweRVolf에 로드하고 운영 체제 상단에서 Tensorflow Lite의 hello world 예제를 포함한 프로그램을 실행하는 방법을 보여줍니다. |
| **Cores and SoCs** | |
| **SweRV EH1 Core** | Western Digital에서 개발한 오픈 소스 상용 RISC-V 코어 (<https://github.com/chipsalliance/Cores-SweRV>). |
| **SweRV EH1 Core Complex** | 추가된 메모리(ICCM, DCCM 및 명령 캐시), 프로그래밍 가능한 인터럽트 컨트롤러(PIC), 버스 인터페이스 및 디버그 장치가 있는 SweRV EH1 코어 (<https://github.com/chipsalliance/Cores-SweRV>). |
| **SweRVolfX** | RVfpga 과정에서 사용하는 System on Chip으로 SweRVolf의 확장입니다.  **SweRVolf** (<https://github.com/chipsalliance/Cores-SweRVolf>): SweRV EH1 Core Complex를 기반으로 구축된 오픈 소스 SoC입니다. boot ROM, UART 인터페이스, 시스템 컨트롤러, 상호 연결(AXI Interconnect, Wishbone Interconnect 및 AXI-to-Wishbone 브리지) 및 SPI 컨트롤러를 추가합니다.  **SweRVolfX:** SweRVolf에 GPIO, PTC, 추가 SPI 및 8자리 7-세그먼트 디스플레이 컨트롤러, 4가지 새로운 주변 장치를 추가합니다. |
| **RVfpgaNexys** | Nexys A7 보드 및 주변 장치를 대상으로 하는 SweRVolfX SoC입니다. DDR2 **인터페이스, CDC**(클럭 도메인 교차) 장치, BSCAN 로직(JTAG 인터페이스용) 및 클럭 생성기를 추가합니다.  RVfpgaNexys는 SweRVolf Nexys(<https://github.com/chipsalliance/Cores-SweRVolf>)와 동일하지만 SweRVolf Nexys 는 SweRVolf를 기반으로 합니다. |
| **RVfpgaSim** | 시뮬레이션을 위한 테스트 벤치 래퍼(wrapper) 및 AXI 메모리가 있는 SweRVolfX SoC 입니다.  RVfpgaSim은 SweRVolf Sim(<https://github.com/chipsalliance/Cores-SweRVolf>)과 동일하지만 SweRVolf Sim 은 SweRVolf를 기반으로 합니다. |

# 소개

이 실습에서는 Zephyr(실시간 운영 체제)용 Tensorflow Lite 프로젝트를 빌드한 다음 SweRVolf에서 해당 Zephyr 프로그램을 실행하는 방법을 보여줍니다. 이전 실습에서 본 것과 유사하게 기본 C 또는 어셈블리 언어 프로그램 대신 Zephyr에서 Tensorflow 프로그램을 실행할 것입니다.

1. **TensorFlow Lite의 간략한 배경**

TensorFlow Lite는 개발자가 모바일, 임베디드 및 IoT 기기에서 모델을 실행할 수 있도록 지원하여 기기 내 머신 러닝을 지원하는 툴 세트입니다.

TensorFlow 모델을 바이너리 크기가 작은 .tflite 모델로 압축합니다. 이를 통해 기기 내 기계 학습이 가능하고 하드웨어 가속을 사용하여 성능이 향상됩니다.

주요 기능은 다음과 같습니다:

* 5가지 주요 constraint(제약 사항)을 해결하여 on-device machine learning 최적화: 대기 시간(서버로 왕복하지 않음), 개인 정보 보호(기기에서 나가는 개인 데이터 없음), 연결성(인터넷 연결이 필요하지 않음), 크기(축소된 모델 및 바이너리 크기) 및 전력 소비(효율적인 추론 및 네트워크 연결 부족).
* 다중 플랫폼 지원: Android 및 iOS 장치, 임베디드 Linux 및 마이크로컨트롤러를 다룹니다.
* 다양한 언어 지원: Java, Swift, Objective-C, C++, Python 등등.
* 고성능: 하드웨어 가속 및 모델 최적화를 이용
* 여러 플랫폼에서 이미지 분류, 물체 감지, 포즈 추정, 질문 답변, 텍스트 분류 등과 같은 일반적인 기계 학습 작업에 대한 예제.

자세한 내용은 <https://www.tensorflow.org/lite/microcontrollers>를 방문 하십시요.

**SweRVolf 및 Tensorflow Lite**

그림 1은 이 실습에서 구현할 Nexys A7 보드 상단의 계층적 레이어를 보여줍니다..



**그림 1. FPGA 보드 상단의 레이어**

Nexys A7 보드에서 TensorFlow Lite 프로그램을 실행하는 단계는 Lab 4의 단계와 미묘하게 다릅니다.

**1단계. SweRVolf를 FPGA 보드에 다운로드**

먼저 FPGA를 대상으로 하는 RISC-V 시스템인 Nexys A7 FPGA 보드에 SweRVolf를 다운로드합니다. PlatformIO를 사용하여 보드에 비트스트림을 업로드하거나, 연결되어 있는 경우 보드에 생성된 비트스트림을 업로드하는 FuseSoC 실행 명령을 사용하여 SweRVolf를 보드에 다운로드합니다.

**2단계. Tensorflow 프로그램 빌드**

이 단계에서는 Zephyr용 Tensorflow Lite 애플리케이션을 빌드합니다. Zephyr RTOS는 이 빌드의 일부로 빌드됩니다. 출력은 elf 파일입니다.

**3단계. SweRVolf에서 프로그램을 로드합니다.**

이번 단계에서는 2단계에서 생성된 elf 파일을 SweRVolf에 로드합니다.

# 요구사항

이 실습을 완료하려면 다음을 설치해야 합니다:

* Vivado 2019.2 Web Pack (설치가이드(04페이지) 참조)
* Verilator (v4.106) (설치가이드(08페이지) 참조)
* FuseSoC (설치가이드(08페이지) 참조)
* OpenOCD (RISC-V-specific version) (설치가이드(09페이지) 참조)
* Zephyr Prerequisites (설치가이드(09페이지) 참조)
* Zephyr SDK (v0.12.4) (설치가이드(10페이지) 참조)
* PuTTY (설치가이드(10페이지) 참조)

**중요:** RVfpga-SoC Labs을 시작하기 전에 RVfpga-SoC 설치 가이드를 완료하는 것이 좋습니다.

아직 설치하지 않았다면 RVfpga-SoC 설치 가이드의 지침에 따라 Xilinx의 Vivado 및 Verilator를 설치하십시오. Imagination의 대학 프로그램(IUP)에서 다운로드한 RVfpga-SoC 폴더를 컴퓨터에 복사했는지 확인합니다.

# Tensorflow의 Hello World 예제

이 실습에서는 Tensorflow 환경만 설정하고 간단한 Hello-World 텐서(tensor) 작업을 실행합니다.

Hello World 예제는 마이크로컨트롤러용 TensorFlow Lite 사용의 기본 사항을 보여주기 위해 설계되었습니다. 이 프로그램은 사인 함수를 복제하는 모델을 훈련하고 실행합니다. 즉, 단일 숫자를 입력으로 사용하고 숫자의 사인 값을 출력합니다.

자세한 내용은 이 링크에서 TensorFlow의 공식 문서를 참조하세요. [link](https://www.tensorflow.org/lite/microcontrollers/get_started_low_level).

# Tensorflow를 위한 환경 설정

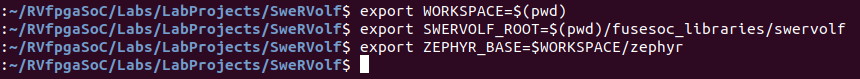
Ubuntu 터미널을 열고 다음 단계를 완료하십시오.

**1단계. "SweRVolf"** 디렉토리로 이동합니다. 다음 Shell 변수를 설정해야 합니다. 이를 위해 다음을 실행합니다.

* export WORKSPACE=$(pwd)

* export SWERVOLF\_ROOT=$WORKSPACE/fusesoc\_libraries/swervolf
* export ZEPHYR\_BASE=$WORKSPACE/zephyr

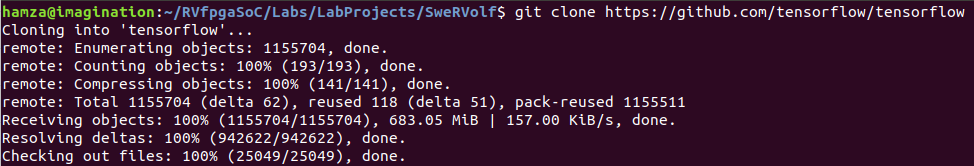
터미널 창에서 "printenv < variable-name >" 명령을 입력하여 Shell 변수가 성공적으로 설정되었는지 확인할 수도 있습니다.



**그림 2. 쉘 변수 설정**

**2단계.** Tensorflow GitHub 저장소를 복제합니다.

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

****

**그림 3. Tensorflow**

이제 "tensorflow" 디렉토리로 이동합니다.

* cd tensorflow



**그림 4. "tensorflow" 디렉토리로 이동**

다음 명령으로 저장소의 "v2.5.0" 분기를 확인하십시오.

* git checkout -b v2.5.0



**그림 5. git checkout**

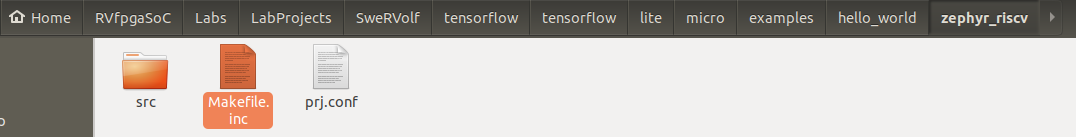
**3단계.** TensorFlow에서 Zephyr SweRVolf에 대한 지원을 추가하려면 이 TensorFlow 저장소에 몇 개의 파일을 복사해야 합니다.

첫 번째 파일은 hello\_world 예제의 "**Makefile.inc**" 파일입니다. 다음 경로로 이동하여 "**Makefile.inc**"를 복사합니다.

* [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabResources/Lab5/Makefile.inc

이제 "**Makefile.inc**" 파일을 다음 위치에 붙여넣습니다(그림 6 참조).

* [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabProjects/SweRVolf/tensorflow/tensorflow/lite/micro/examples/hello\_world/zephyr\_riscv/



**그림 6. Makefile.inc**

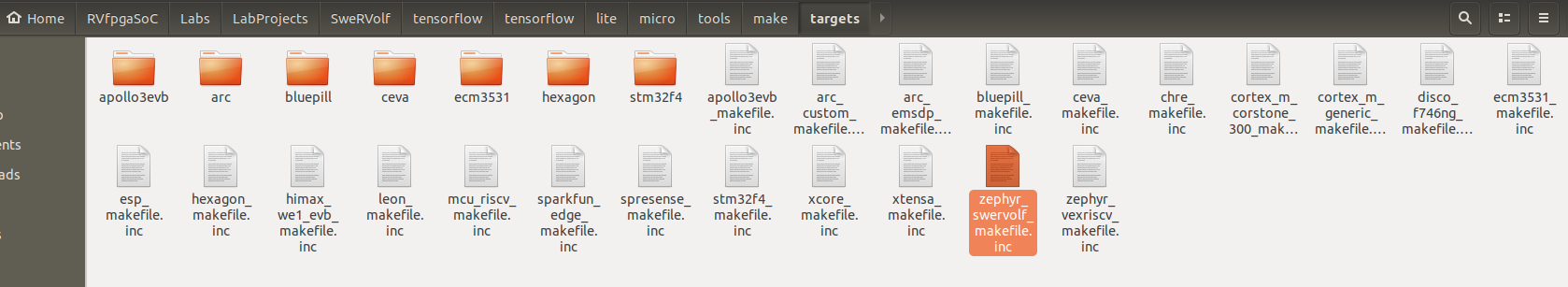
두 번째 파일은 "**zephyr\_swervolf\_makefile.inc**" 파일입니다. 다음 경로로 이동하여 "**zephyr\_swervolf\_makefile.inc**"를 복사합니다.

* [RVfpgaSoCPath]/RVfpgaSoC/Labs/LabResources/Lab5/

zephyr\_swervolf\_makefile.inc

이제 "**zephyr\_swervolf\_makefile.inc**" 파일을 다음 위치에 붙여 넣습니다(그림 7 참조).

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



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

**4단계.** 필요한 패키지를 설치합니다.

다음 명령을 사용하여 "WORKSPACE" 디렉토리로 이동합니다.

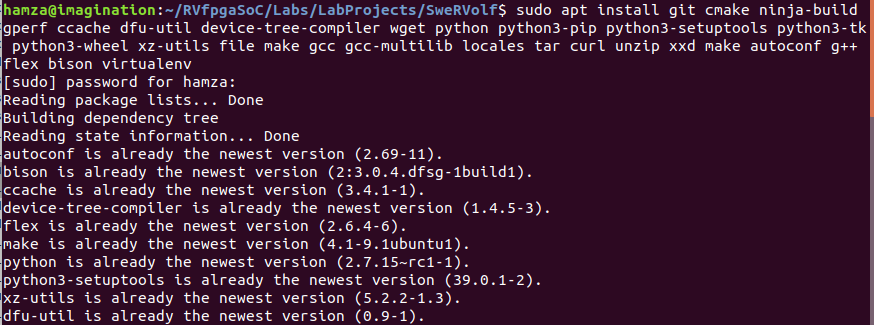
* cd ..



**그림 8. WORKSPACE 디렉터리로 이동**

다음 명령으로 필요한 패키지를 설치합니다.

* 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



**그림 9. 패키지 설치**

**5단계.** 가상 환경을 만듭니다.

먼저 다음 명령을 사용하여 zephyr 디렉터리로 이동합니다.

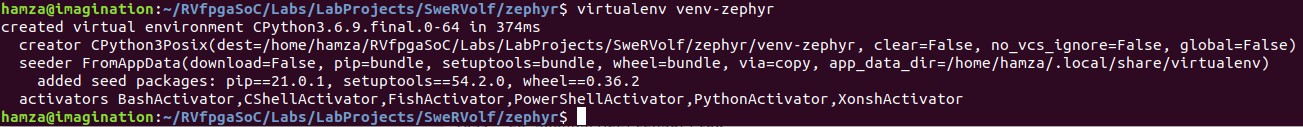
* cd zephyr

****

**그림 10. zephyr 디렉터리로 이동**

다음 명령을 사용하여 zephyr 디렉터리 내에 가상 환경을 만듭니다.

* virtualenv venv-zephyr



**그림 11. venv-zephyr 만들기**

**Step 6.** 다음 명령어를 입력하여 마지막 단계에서 생성한 가상 환경을 활성화합니다.

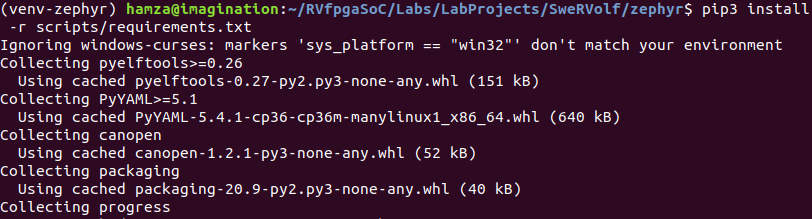
* source venv-zephyr/bin/activate



**그림 12. venv-zephyr 활성화**

**7단계.** 다음 명령을 사용하여 "requirements.txt" 파일에 나열된 필수 패키지를 설치합니다.

* pip3 install -r scripts/requirements.txt



**그림 13. 필수 패키지 설치**

이제 이 터미널 탭을 닫고 "hello\_world" 예제를 빌드할 기본 터미널 탭으로 돌아갈 수 있습니다.

# Swervolf를 위한 Hello World 예제 빌드

이 섹션에서는 SweRVolf에 대한 "hello\_world" 예제를 빌드합니다. "hello\_world" 예제에 대해 "**zephyr.bin**" 및 "**zephyr.elf**" 파일을 생성합니다.

**1단계.** 먼저 tensorflow 디렉토리로 이동합니다.

* cd ../tensorflow/



**그림 14. "tensorflow" 디렉토리로 이동**

**2단계.** 이 실습에서는 SweRVolf용 Hello World를 구축할 것입니다. 이것은 다음 명령으로 수행됩니다.

* make -f tensorflow/lite/micro/tools/make/Makefile TARGET=zephyr\_swervolf BUILD\_TYPE=debug hello\_world\_bin

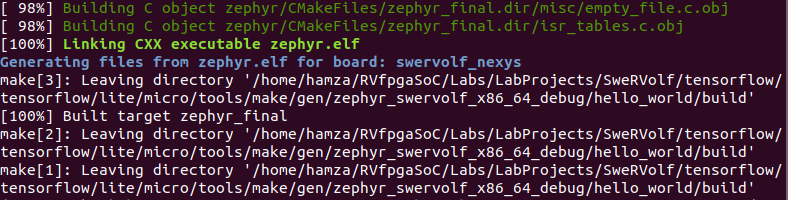


**그림 15. hello\_world 빌드 예제**

종속성에 대한 일부 툴 모음을 다운로드해야 하므로 몇 분 정도 걸립니다. 완료되면 다음과 같은 경로 내에 생성된 일부 폴더가 표시되어야 합니다.

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

이러한 폴더에는 생성된 프로젝트 및 소스 파일이 포함됩니다.



**그림 16. hello\_world 예제 빌드 완료**

결과 바이너리(zephyr.bin 및 zephyr.elf)는 다음 경로에서 생성됩니다.

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

**3단계.** 이제 다음 명령을 입력하여 가상 환경을 종료할 수 있습니다.

* deactivate

# Verilator에서 Hello World 예제 실행하기

이 섹션에서는 "zephyr.bin" 파일을 ".hex" 파일로 변환한 다음 SweRVolf용 시뮬레이터를 실행하는 동안 초기 램(ram) 파일로 로드합니다.

**1단계.** "hello\_world" 프로젝트 디렉토리로 이동합니다. 다음 명령을 입력하여 해당 디렉토리를 입력하십시오.

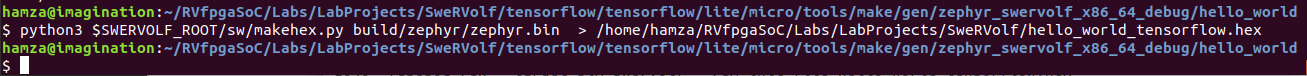
* cd tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/



**그림 17. “hello\_world” 프로젝트 경로**

**2단계. ".bin"** 파일을 **".hex"** 파일로 변환합니다. **".hex"** 파일을 생성하려면 hello\_world 디렉토리에서 다음 명령을 실행하십시오 :

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



**그림 18. ".bin"을 ".hex"로 변환**

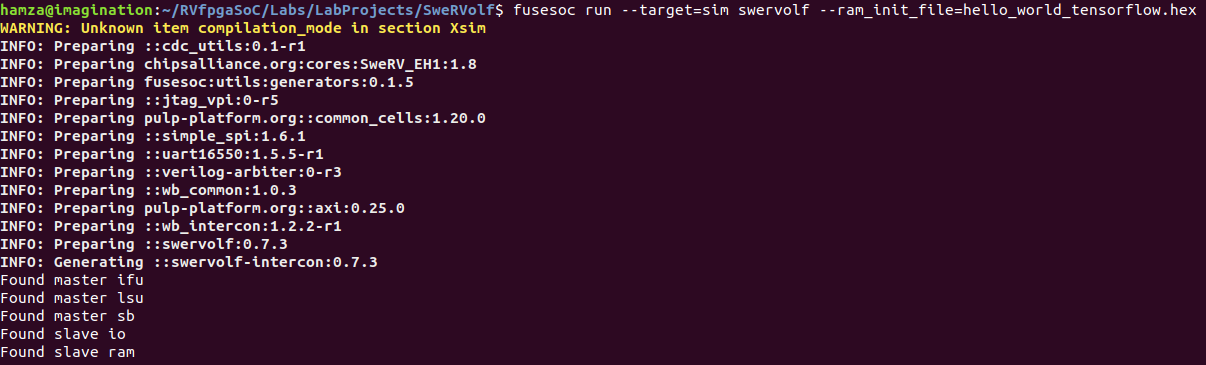
**3단계.** "WORKSPACE" 디렉토리로 이동합니다.



**그림 19. venv-zephyr에 패키지 설치하기**

**4단계.** 시뮬레이터에서 **".hex"** 파일을 로드합니다.

* fusesoc run --target=sim swervolf --ram\_init\_file=hello\_world\_tensorflow.hex



**그림 20. 시뮬레이터에서 ".hex" 파일 로드**

hello\_world 예제의 출력을 볼 수 있습니다(그림 21 참조). 프로그램은 "sine" 함수의 X 및 Y 값을 인쇄합니다.



**그림 21. "hello\_world" 출력**

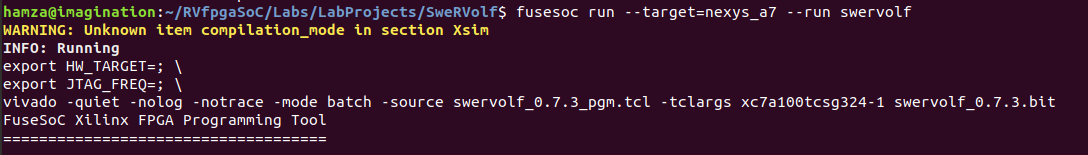
"ctrl + c"를 눌러 프로그램을 종료합니다.

# Nexys A7 보드에서 Hello World 예제 실행하기

이 섹션에서는 OpenOCD를 사용하여 보드에서 "hello\_world" 프로젝트를 실행합니다.

**1단계.** Nexys A7 보드를 컴퓨터에 연결한 다음 Workspace 디렉터리에서 FPGA 빌드 명령을 실행합니다.

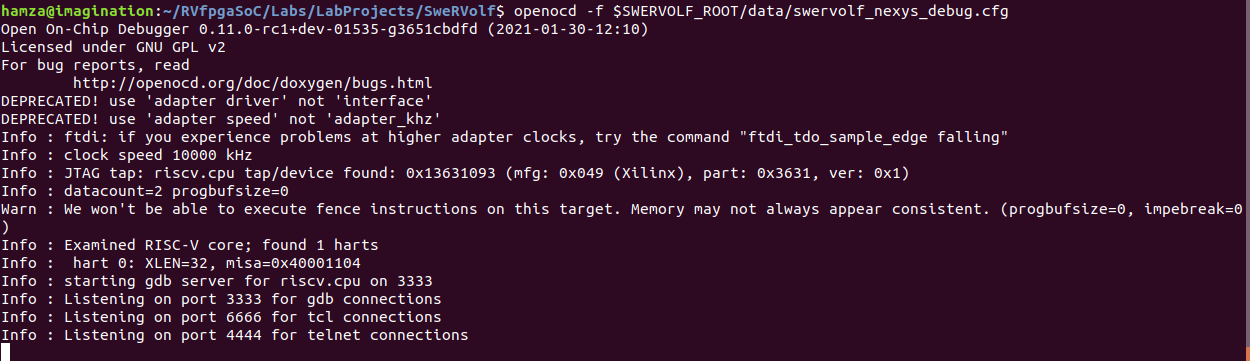
* fusesoc run --target=nexys\_a7 --run swervolf



**그림 22. FPGA 빌드 실행**

**2단계.** OpenOCD를 SweRVolf와 연결합니다.

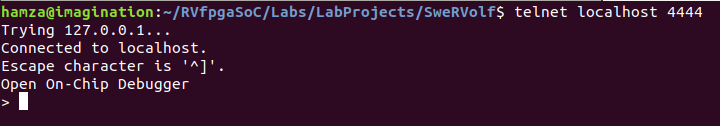
* openocd -f $SWERVOLF\_ROOT/data/swervolf\_nexys\_debug.cfg



**그림 23. OpenOCD 연결**

**3단계.** "Ctrl + Shift + t"를 사용하여 새 터미널을 열고 다음 명령을 사용하여 OpenOCD를 통해 디버그 세션에 연결합니다.

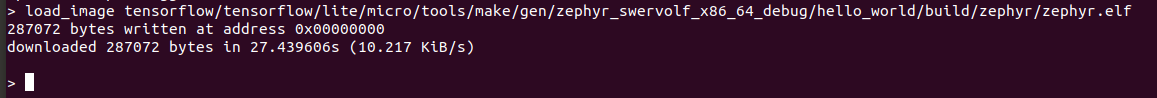
* telnet localhost 4444



**그림 24. telnet 로컬 호스트 4444**

OpenOCD는 *load\_image /path/to/file.elf*를 실행하여 ELF 프로그램 파일 로드를 지원합니다. 경로는 OpenOCD가 시작된 디렉토리에 상대적임을 기억하십시오.

* load\_image tensorflow/tensorflow/lite/micro/tools/make/gen/zephyr\_swervolf\_x86\_64\_debug/hello\_world/build/zephyr/zephyr.elf



**그림 25. ".elf" 파일 로드**

프로그램이 로드된 후 다음 명령을 사용하여 프로그램 카운터를 주소 0으로 설정합니다.

* reg pc 0



**그림 26. 프로그램 카운터를 0으로 설정**

이제 다음 명령을 사용하여 프로그램을 시작하십시오.

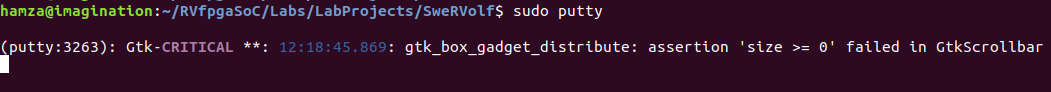
* resume



**그림 27. 프로그램 시작**

**4단계.** "Ctrl + Shift + t"를 사용하여 새 터미널을 엽니다. 명령을 사용하여 "PuTTY"를 엽니다.

* sudo putty

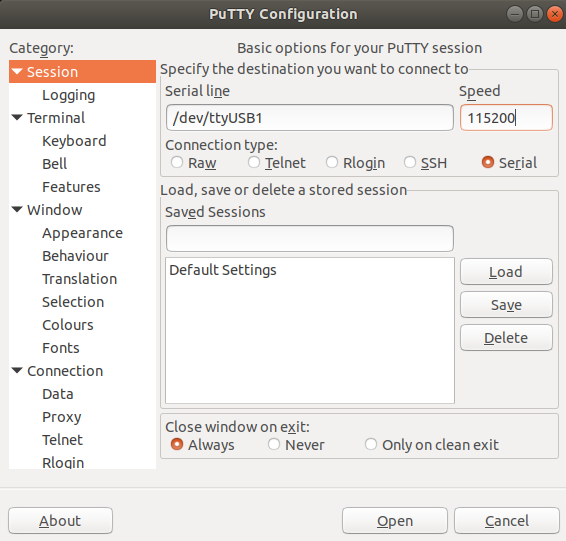


**그림 28. PuTTY 열기**

여기에서 PuTTY를 Nexys A7 보드용 직렬 콘솔로 사용할 것입니다.

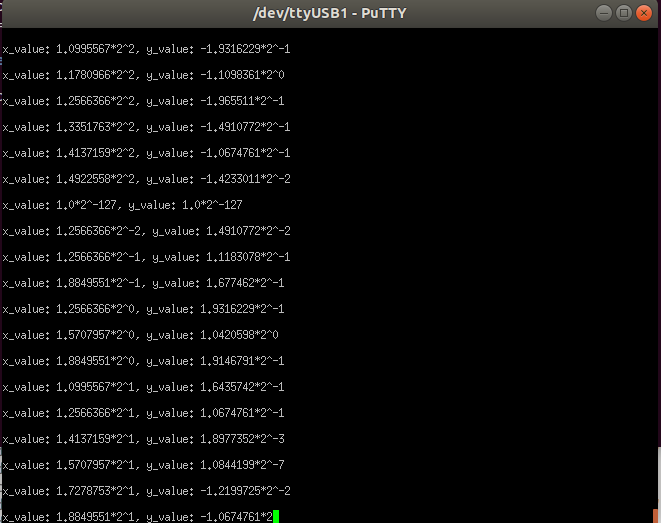
**5단계.** 다음 구성을 설정합니다 :

연결 유형을 "**Serial**"로 선택한 다음 "**/dev/ttyUSB1**"을 serial line 으로 입력하고 속도를 "**115200**"으로 설정합니다. 이제 "Open"를 클릭하여 직렬 콘솔을 시작합니다.



**그림 29. PuTTY 구성**

직렬 콘솔에서 hello\_world 예제의 출력을 볼 수 있습니다(그림 30 참조).



**그림 30. 직렬 콘솔**

시뮬레이션 섹션에서 보았듯이 프로그램은 TensorFlow 모델이 그리는 사인 함수의 "X" 및 "Y" 좌표를 인쇄합니다.

|  |
| --- |
| **참고:** 직렬 콘솔을 열 수 없는 경우 직렬 라인으로 "/dev/ttyUSB0"을 입력 하십시오. |

따라서 이 실습에서는 TensorFlow의 "hello\_world" 예제를 Zephyr 애플리케이션으로 성공적으로 빌드한 다음 SweRVolf에서 해당 예제를 실행했습니다.