



THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga-SoC Lab 4

SweRVolf 에서 Zephyr 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 Digital 의 Whisper 명령어 세트 시뮬레이터(ISS)를 사용하는 방법도 보여줍니다.
RVfpga-SoC	SweRV 코어, 메모리 및 주변 장치와 같은 빌딩 블록을 사용하여 처음부터 SweRVolfX SoC 의 하위 집합을 구축하는 방법을 보여주는 과정입니다. 이 과정은 또한 Zephyr 실시간 운영 체제(RTOS)를 SweRVolf 에 로드하고 운영 체제 상단에서 Tensorflow Lite 의 hello world 예제를 포함한 프로그램을 실행하는 방법을 보여줍니다.
Cores and SoCs	
SweRV EH1 Core	Western Digital 에서 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 Chip 으로 SweRVolf 의 확장입니다. SweRVolf (https://github.com/chipsalliance/Cores-SweRVolf): SweRV EH1 Core Complex 를 기반으로 구축된 오픈 소스 SoC 입니다. boot ROM, UART 인터페이스, 시스템 컨트롤러, 상호 연결(AXI Interconnect, Wishbone Interconnect 및 AXI-to-Wishbone 브리지) 및 SPI 컨트롤러를 추가합니다. SweRVolfX : SweRVolf 에 GPIO, PTC, 추가 SPI 및 8 자리 7-세그먼트 디스플레이 컨트롤러, 4 가지 새로운 주변 장치를 추가합니다.
RVfpgaNexys	Nexys A7 보드 및 주변 장치를 대상으로 하는 SweRVolfX SoC 입니다. DDR2 인터페이스, CDC(클럭 도메인 교차) 장치, BSCAN 로직(JTAG 인터페이스용) 및 클럭 생성기를 추가합니다. RVfpgaNexys 는 SweRVolf Nexys(https://github.com/chipsalliance/Cores-SweRVolf)와 동일하지만 SweRVolf Nexys 는 SweRVolf 를 기반으로 합니다.
RVfpgaSim	시뮬레이션을 위한 테스트 벤치 래퍼(wrapper) 및 AXI 메모리가 있는 SweRVolfX SoC 입니다.

	RVfpgaSim 은 SweRVolf Sim(https://github.com/chipsalliance/Cores-SweRVolf)과 동일하지만 SweRVolf Sim 은 SweRVolf 를 기반으로 합니다.
--	--

1. 소개

이 실습에서는 SweRVolf 상에서 Zephyr 실시간 운영 체제(RTOS)를 실행하는 방법을 보여줍니다. RTOS 는 대부분 버퍼 지연 없이 들어오는 데이터를 처리하는 실시간 애플리케이션을 제공하기 위한 운영 체제입니다.

실습 2 와 3 에서는 RISC-V 어셈블리 또는 C 언어로 작성된 간단한 프로그램을 실행했습니다. 실제 응용 프로그램에서 SoC 는 거의 항상 운영 체제를 실행하고 응용 프로그램은 운영 체제 위에서 실행됩니다.

임베디드 시스템용 운영 체제에는 임베디드 Linux 기반 운영 체제와 RTOS 의 두 가지 전체 범주가 있습니다. SoC 가 특정 CPU 로 설계될 때 설계는 일반적으로 하나 또는 다른 유형의 운영 체제를 사용하도록 조정됩니다. SweRVolf 는 RTOS 를 실행하기 위한 목적으로 만들어졌습니다. SweRV EH1 CPU 에는 메모리 관리 장치가 없으므로 임베디드 Linux 를 실행하는 데 어려움을 겪을 것입니다.

그림 1 은 전체 시스템의 다양한 하드웨어/소프트웨어 계층을 보여줍니다.

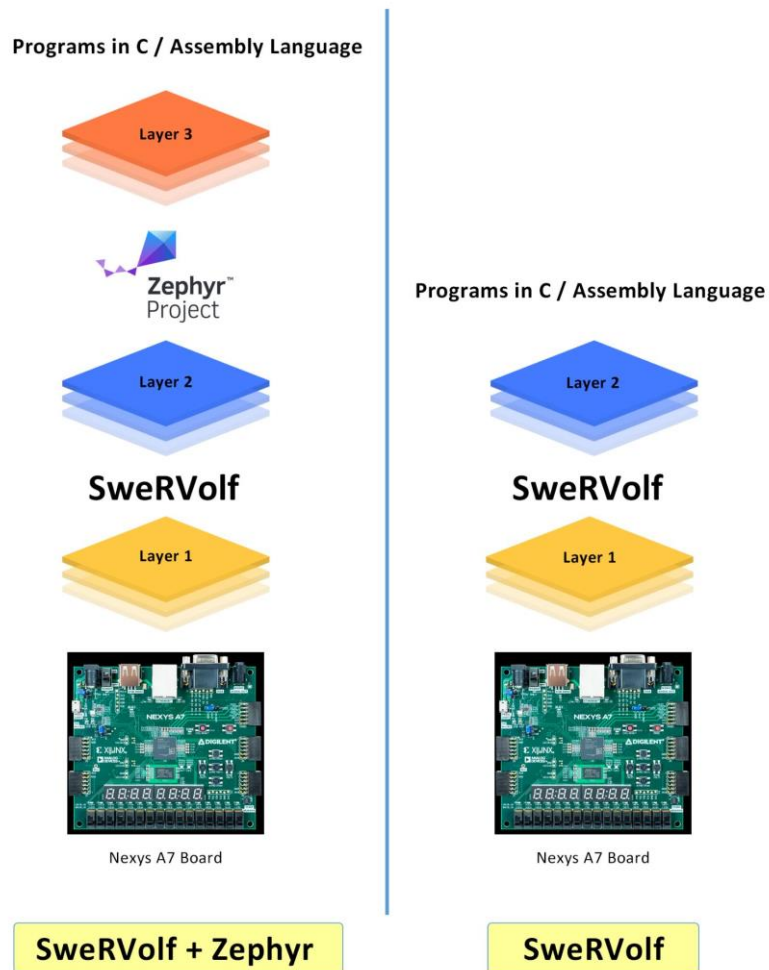


그림 1. FPGA 보드 상단의 레이어

이 실습에서는 Zephyr RTOS 에 대해 설명하고, SweRVolf 에서 Zephyr RTOS 를 빌드 및 실행하고, Zephyr 애플리케이션을 빌드 및 실행합니다.

2. 요구사항

이 실습을 완료하려면 다음을 설치해야 합니다 :

- Vivado 2019.2 Web Pack (설치 가이드(04 페이지) 참조)
- Verilator (v4.106) (설치 가이드(08 페이지) 참조)
- FuseSoC (설치 가이드(08 페이지) 참조)
- OpenOCD (RISC-V-specific version) (설치 가이드(09 페이지) 참조)
- Zephyr Prerequisites (설치 가이드(09 페이지) 참조)
- Zephyr SDK (v0.12.4) (설치 가이드(10 페이지) 참조)
- PuTTY (설치 가이드(11 페이지) 참조)

중요: RVfpga-SoC Labs 를 시작하기 전에 RVfpga-SoC 설치 가이드를 완료하는 것이 좋습니다.

아직 설치하지 않았다면 RVfpga-SoC 설치 가이드의 지침에 따라 Xilinx 의 Vivado 및 Verilator 를 설치하십시오. Imagination 의 대학 프로그램(IUP)에서 다운로드한 RVfpga-SoC 폴더를 컴퓨터에 복사했는지 확인합니다.

3. Zephyr 개요

Zephyr 프로젝트는 여러 하드웨어 아키텍처를 지원하고 리소스가 제한된 장치에 최적화되어 있으며 보안을 염두에 두고 구축된, 확장 가능한 실시간 운영 체제입니다. Zephyr OS 는 간단한 임베디드 환경 센서 및 LED 웨어러블에서 정교한 스마트 시계 및 IoT 무선 게이트웨이에 이르기까지 리소스가 제한된 시스템에서 사용하도록 설계된, 소규모 커널을 기반으로 합니다.

Zephyr 는 다중 스레딩, 인터럽트, 메모리 할당, 스레드 간 동기화, 스레드 간 데이터 전달 및 전원 관리와 같은, 개발을 위한 여러 서비스를 제공합니다.

Zephyr 는 다양한 CPU 아키텍처와 개발자 도구로 다양한 보드를 지원합니다. 연구 개발자들은 점점 더 많은 SoC, 플랫폼 및 드라이버에 대한 지원을 추가했습니다.

Zephyr 커널은 다음을 포함한 여러 아키텍처를 지원합니다.

- RISC-V(32 비트 및 64 비트)

Zephyr 프로젝트에 대한 자세한 내용은 <http://docs.zephyrproject.org> 에서 Zephyr 프로젝트 문서를 참조하십시오.

이 실습에서는 먼저 Zephyr 의 버전 2.4 를 Workspace 에 추가하는 방법을 보여줍니다. 그런 다음 Zephyr 와 함께 제공되는 몇 가지 샘플 예제 코드를 빌드합니다. 이 실습에서는 하드웨어와 시뮬레이션 모두에서 Zephyr 를 사용하는 예를 보여줍니다.

4. 하드웨어/소프트웨어 계층 이해

실습 2 와 3 에서 FPGA 보드에서 프로그램을 실행하는 프로세스는 다음 단계를 따랐습니다:

1 단계. SweRVolf 를 FPGA 보드에 다운로드

먼저 FPGA 를 대상으로 하는 RISC-V 시스템인 Nexys A7 FPGA 보드에 SweRVolf 를 다운로드합니다. PlatformIO 를 사용하여 보드에 비트스트림을 업로드하거나, 생성된 비트스트림을 업로드하는 FuseSoC 실행 명령을 사용하여 SweRVolf 를 보드에 다운로드합니다.

2 단계. SweRVolf 에서 프로그램 빌드 및 실행

두 번째 단계는 RISC-V 프로그램을 빌드한 다음 SweRVolf 에 다운로드하는 것입니다.

이 실습에서는 이러한 단계를 수정하여 다른 계층인 Zephyr RTOS 를 SweRVolf 에 추가하고 Zephyr 에서 프로그램을 실행합니다. 이를 수행하는 단계는 다음과 같습니다.

1 단계. SweRVolf 를 FPGA 보드에 다운로드

위 1 단계 내용과 같음.

2 단계. Zephyr 빌드

이 단계에서는 Zephyr 용 애플리케이션을 빌드합니다. 애플리케이션 구축 프로세스는 기본 Zephyr RTOS 도 구축합니다. 출력은 elf 파일입니다.

3 단계. SweRVolf 에서 프로그램을 로드합니다.

이 단계에서는 2 단계에서 생성된 elf 파일을 SweRVolf 에 로드합니다.

위의 그림 1 은 프로그램을 실행하는 두 가지 모드를 나란히 보여주는 그림입니다.

이제 Zephyr 애플리케이션을 빌드한 다음 Zephyr 에서 해당 애플리케이션을 실행하는 방법을 보여드리겠습니다.

5. SweRVolf 에 Zephyr 지원 추가

이 실습 섹션에서는 WORKSPACE 에 Zephyr 를 추가하는 방법을 보여줍니다.

Ubuntu 터미널을 열고 다음 단계를 완료하십시오.

1 단계. 프로젝트의 루트로 사용하기 위해서, 이전 실습에서 작업 공간으로 만든 "SweRVolf" 디렉터리로 이동합니다. 우리는 그것을 **\$WORKSPACE** 라고 부릅니다. 이제 동일한 shell 변수를 다시 설정해야 합니다. 이를 위해 다음을 실행합니다.

```
> export WORKSPACE=$(pwd)

> export SWERVOLF_ROOT=$WORKSPACE/fusesoc_libraries/swervolf
```

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

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

그림 2. Shell 변수 설정

2 단계. Zephyr 및 SweRVolf 전용 드라이버 추가

다음을 실행하여 FuseSoC 작업 영역과 동일한 디렉터리에 West(Zephyr의 빌드 도구) 작업 영역을 만듭니다.

➤ `west init`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ west init
=== Initializing in /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf
--- Cloning manifest repository from https://github.com/zephyrproject-rtos/zephyr, rev. master
Initialized empty Git repository in /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/.west/manifest-tmp/.git/
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 551804 (delta 2), reused 1 (delta 1), pack-reused 551800
Receiving objects: 100% (551804/551804), 375.87 MiB | 435.00 KiB/s, done.
Resolving deltas: 100% (418833/418833), done.
From https://github.com/zephyrproject-rtos/zephyr
* branch                master                                -> FETCH_HEAD
* [new branch]          backport-23821-to-v1.14-branch        -> origin/backport-23821-to-v1.14-branch
* [new branch]          backport-24971-to-v1.14-branch        -> origin/backport-24971-to-v1.14-branch
* [new branch]          backport-25852-to-v1.14-branch        -> origin/backport-25852-to-v1.14-branch
* [new branch]          backport-26571-to-v1.14-branch        -> origin/backport-26571-to-v1.14-branch
* [new branch]          backport-29181-to-v2.4-branch         -> origin/backport-29181-to-v2.4-branch
* [new branch]          backport-31759-to-v2.5-branch         -> origin/backport-31759-to-v2.5-branch
* [new branch]          backport-31908-to-v2.4-branch         -> origin/backport-31908-to-v2.4-branch
.....
* [new tag]             zephyr-v2.2.0                      -> zephyr-v2.2.0
* [new tag]             zephyr-v2.2.1                      -> zephyr-v2.2.1
* [new tag]             zephyr-v2.3.0                      -> zephyr-v2.3.0
* [new tag]             zephyr-v2.4.0                      -> zephyr-v2.4.0
* [new tag]             zephyr-v2.5.0                      -> zephyr-v2.5.0
b0b20112e80187705a08240919613ca9937baae6 refs/remotes/origin/master
Branch 'master' set up to track remote branch 'master' from 'origin'.
Already on 'master'
--- setting manifest.path to zephyr
=== Initialized. Now run "west update" inside /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf.
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 3. west 초기화

3 단계. 다음 명령을 사용하여 SweRVolf 관련 드라이버 및 BSP(Board Support Package)를 추가합니다.

➤ `west config manifest.path fusesoc_libraries/swervolf`

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ west config manifest.path fusesoc_libraries/swervolf
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 4. west 구성

➤ west update

인터넷 다운로드 속도에 따라 다운로드 프로세스를 완료하는 데 몇 분이 소요될 수 있습니다.

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ west update
=== updating zephyr (zephyr):
HEAD is now at 7a3b253ced release: Zephyr 2.4.0
WARNING: left behind zephyr branch "master"; to switch back to it (fast forward):
  git -C zephyr checkout master
=== updating cmsis (modules/hal/cmsis):
--- cmsis: initializing
Initialized empty Git repository in /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/modules/hal/cmsis/.git/
--- cmsis: fetching, need revision 542b2296e6d515b265e25c6b7208e8fea3014f90
remote: Enumerating objects: 563, done.
remote: Counting objects: 100% (563/563), done.
remote: Compressing objects: 100% (291/291), done.
remote: Total 563 (delta 288), reused 528 (delta 267), pack-reused 0
Receiving objects: 100% (563/563), 2.21 MiB | 618.00 KiB/s, done.
Resolving deltas: 100% (288/288), done.
From https://github.com/zephyrproject-rtos/cmsis
* [new branch]      master      -> refs/west/master
HEAD is now at 542b229 DSP: Integrate CMSIS-DSP 1.8.0 (CMSIS 5.7.0)
HEAD is now at 542b229 DSP: Integrate CMSIS-DSP 1.8.0 (CMSIS 5.7.0)
=== updating hal_atmel (modules/hal/atmel):

=====
=== updating trusted-firmware-m (modules/tee/tfm):
--- trusted-firmware-m: initializing
Initialized empty Git repository in /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/modules/tee/tfm/.git/
--- trusted-firmware-m: fetching, need revision 143df675557305b61f7930a50459a53a8d2bb097
remote: Enumerating objects: 1970, done.
remote: Counting objects: 100% (1970/1970), done.
remote: Compressing objects: 100% (1249/1249), done.
remote: Total 16030 (delta 633), reused 1498 (delta 536), pack-reused 14060
Receiving objects: 100% (16030/16030), 36.00 MiB | 872.00 KiB/s, done.
Resolving deltas: 100% (7538/7538), done.
From https://github.com/zephyrproject-rtos/trusted-firmware-m
* [new branch]      master      -> refs/west/master
HEAD is now at 143df67 CMakeLists.txt: make BL2 configurable
HEAD is now at 143df67 CMakeLists.txt: make BL2 configurable
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 5. west 업데이트

이제 작업 공간이 다음과 같이 보일 것입니다.

```
$WORKSPACE
├── fusesoc_libraries
│   ├── ...
│   └── swervolf
└── ...
    └── zephyr
```

6. Verilator 에서 Zephyr 애플리케이션 빌드 및 실행

이 섹션에서는 Zephyr 에서 실행할 수 있는 프로그램을 빌드하는 방법을 단계별로 설명합니다. 그런 다음 Verilator 시뮬레이터에서 이러한 프로그램을 시뮬레이션하는 방법을 보여줍니다. 이 섹션에서는 두 가지 예제 프로그램을 보여줍니다.

1. Zephyr Hello World 예제

이 예제는 터미널에 “Hello World” + “Configured Board Name”을 출력합니다.

소스 코드는 그림 6 을 참조하십시오.

```

1
2 #include <zephyr.h>
3 #include <sys/printk.h>
4
5 void main(void)
6 {
7     printk("Hello World! %s\n", CONFIG_BOARD);
8 }
9

```

그림 6. hello_world 예제의 main.c

단계 1. 다음 경로에 있는 이 예제의 디렉터리로 이동합니다.

```
$WORKSPACE/zephyr/samples/hello_world
```

이렇게 하려면 다음 명령을 사용하십시오.

```
> cd zephyr/samples/hello_world
```

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

그림 7. hello_world 디렉터리로 이동

2 단계. 다음 명령을 사용하여 "hello_world" 예제에 대한 코드를 빌드합니다.

```
> west build -b swervolf_nexys
```

```

hamza@Imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world$ west build -b swervolf_nexys
-- west build: generating a build system
Including boilerplate (Zephyr base): /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/cmake/app/boilerplate.cmake
-- Application: /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world
-- Zephyr version: 2.4.0 (/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr)
-- Found Python3: /usr/bin/python3.6 (found suitable exact version "3.6.9") found components: Interpreter
-- Found west (found suitable version "0.9.0", minimum required is "0.7.1")
-- Board: swervolf_nexys
-- Cache files will be written to: /home/hamza/.cache/zephyr
ZEPHYR_TOOLCHAIN_VARIANT not set, trying to locate Zephyr SDK
-- Found toolchain: zephyr (/home/hamza/zephyr-sdk-0.12.2)
-- Found dtc: /home/hamza/zephyr-sdk-0.12.2/sysroots/x86_64-pokysdk-linux/usr/bin/dtc (found suitable version "1.5.0", minimum required is "1.4.6")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world/build
-- west build: building application
[1/109] Preparing syscall dependency handling
[42/109] Building C object zephyr/CMakeFiles/zephyr.dir/drivers/interrupt_controller/intc_swerv_pic.c.obj
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c: In function 'swerv_pic_read':
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c:47:10: warning: cast to pointer
from integer of different size [-Wint-to-pointer-cast]
  47 | return *(volatile uint32_t *) (DT_INST_REG_ADDR(0) + reg);
     |         ^
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c: In function 'swerv_pic_write':
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c:52:3: warning: cast to pointer
from integer of different size [-Wint-to-pointer-cast]
  52 | *(volatile uint32_t *) (DT_INST_REG_ADDR(0) + reg) = val;
     |     ^
[104/109] Linking C executable zephyr/zephyr_prebuilt.elf
Memory region      Used Size  Region Size  %age Used
RAM:                17952 B      8 MB      0.21%
IDT_LIST:           41 B        2 KB      2.00%
[109/109] Linking C executable zephyr/zephyr.elf
hamza@Imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world$

```

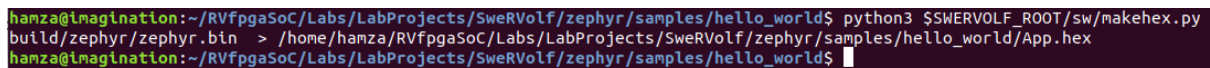
그림 8. hello_world 빌드

이것은 **hello_world** 예제를 위한 `zephyr.elf` 및 `zephyr.bin` 파일을 생성합니다.
시뮬레이터에서 **".bin"** 파일을 사용하지만 먼저 적절한 Verilog hex file 로 변환해야 합니다.

3 단계. ".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/zephyr/sam
ples/hello_world/App.hex
```

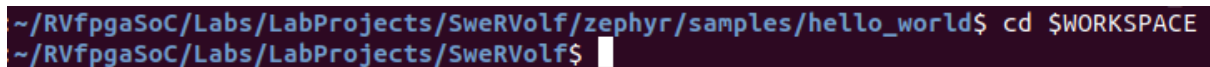


```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world$ python3 $SWERVOLF_ROOT/sw/makehex.py
build/zephyr/zephyr.bin > /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world/App.hex
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world$
```

그림 9. hello_world hex 파일 생성

4 단계. WORKSPACE 디렉터리로 이동합니다.

```
> cd $WORKSPACE
```



```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/hello_world$ cd $WORKSPACE
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 10. 기본 Workspace 디렉터리로 이동

5 단계. 시뮬레이터에서 ".hex" 파일을 로드합니다.

```
> fusesoc run --target=sim swervolf --
ram_init_file=zephyr/samples/hello_world/App.hex
```

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ fusesoc run --target=sim swervolf
--ram_init_file=zephyr/samples/hello_world/App.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
=====
INFO: Generating ::swervolf-swerv_default_config:0.7.3
INFO: Generating ::swervolf-version:0.7.3
INFO: Generating ::swervolf-wb_intercon:0.7.3
Found master io
Found slave rom
Found slave spi_flash
Found slave sys
Found slave uart
=====
INFO: Setting up project
```

그림 11. fusesoc 실행

터미널에 다음 출력이 표시됩니다(그림 12 참조).

```
make[1]: Leaving directory '/home/hamza/SweRVolf/build/swervolf_0.7.3/sim-verilator'
INFO: Running
INFO: Running simulation
Loading RAM contents from /home/hamza/SweRVolf/zephyr/samples/hello_world/App.hex
Releasing reset
*** Booting Zephyr OS build zephyr-v2.4.0 ***
Hello World! swervolf_nexys
```

그림 12. hello_world 예제 출력

프로그램을 중지하려면 "ctrl + c"를 누르십시오.

2. Zephyr Philosophers 예제

Dining Philosophers Problem (클래식 다중 스레드 동기화 문제)에 대한 솔루션 구현.

이 특정 구현은 우선 순위가 다른 협력 스레드와 여러 선점형(preemptible), 또한 다른 동적 mutexes (Mutual Exclusion, 상호배제) 사용하여 스레드를 sleep 하는 방법을 보여줍니다.

철학자는 항상 가장 낮은 포크를 먼저 얻으려고 시도합니다(f1 다음 f2). 완료되면 역순으로 포크를 돌려줍니다(f2, f1). 포크 2 개를 얻으면 먹는 것입니다. 그렇지 않으면 그는 생각하고 있습니다. 철학자가 배고프지만 포크를 사용할 수 없을 때 STARVING, 철학자가 두 번째 포크를 사용할 수 있기를 기다리고 있을 때 HOLDING ONE FORK 와 같은 과도기 상태도 표시됩니다.

각 철학자는 EATING 과 THINKING 상태를 무작위로 번갈아 가며 나타냅니다.

이 예제의 소스 코드를 보려면 다음 경로로 이동하십시오:

```
$WORKSPACE/zephyr/samples/philosophers/src/main.c
```

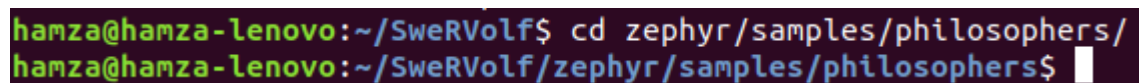
이 예제에서는 동일한 프로세스를 다시 반복하지만 철학자 디렉토리 안에서 진행됩니다.

1 단계. 이 예제 프로그램은 다음 디렉토리에 있습니다.

```
$WORKSPACE/zephyr/samples/philosophers
```

다음 명령을 사용하여 해당 디렉토리로 변경하십시오:

```
> cd zephyr/samples/philosophers
```

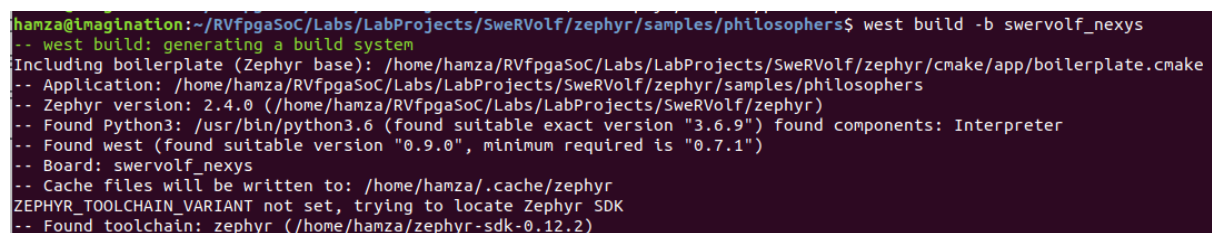


```
hamza@hamza-lenovo:~/SweRVolf$ cd zephyr/samples/philosophers/
hamza@hamza-lenovo:~/SweRVolf/zephyr/samples/philosophers$
```

그림 13. philosophers 디렉토리로 이동

2 단계. 다음 명령을 사용하여 철학자 예제에 대한 코드를 빌드합니다.

```
> west build -b swervolf_nexys
```



```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers$ west build -b swervolf_nexys
-- west build: generating a build system
Including boilerplate (Zephyr base): /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/cmake/app/boilerplate.cmake
-- Application: /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers
-- Zephyr version: 2.4.0 (/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr)
-- Found Python3: /usr/bin/python3.6 (found suitable exact version "3.6.9") found components: Interpreter
-- Found west (found suitable version "0.9.0", minimum required is "0.7.1")
-- Board: swervolf nexys
-- Cache files will be written to: /home/hamza/.cache/zephyr
ZEPHYR_TOOLCHAIN_VARIANT not set, trying to locate Zephyr SDK
-- Found toolchain: zephyr (/home/hamza/zephyr-sdk-0.12.2)
```

```
-- Configuring done
-- Generating done
-- Build files have been written to: /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers/build
-- west build: building application
[1/110] Preparing syscall dependency handling

[44/110] Building C object zephyr/CMakeFiles/interrupt_controller/intc_swerv_pic.c.obj
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c: In function 'swerv_pic_read':
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c:47:10: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   47 | return *(volatile uint32_t *) (DT_INST_REG_ADDR(0) + reg);
      |         ^
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c: In function 'swerv_pic_write':
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c:52:3: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   52 | *(volatile uint32_t *) (DT_INST_REG_ADDR(0) + reg) = val;
      |     ^
[105/110] Linking C executable zephyr/zephyr_prebuilt.elf
Memory region      Used Size  Region Size  %age Used
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers$
```

그림 14. 철학자 빌드

이렇게 하면 철학자 예제에 대한 zephyr.elf 및 zephyr.bin 파일이 생성됩니다. 다시 ".bin" 파일을 적절한 Verilog 16 hex 파일로 변환합니다.

3 단계. ".bin" 파일을 ".hex" 파일로 변환

".hex" 파일을 생성하려면 철학자 디렉토리에서 다음 명령을 실행하십시오.

```
> python3 $SWERVOLF_ROOT/sw/makehex.py build/zephyr/zephyr.bin
>
/home/{YourUsername}/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers/App.hex
```

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers$ python3 $SWERVOLF_ROOT/sw/makehex.py
build/zephyr/zephyr.bin > /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers/App.hex
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers$
```

그림 15. philosophers hex 파일 생성

4 단계. WORKSPACE 디렉터리로 이동합니다.

```
> cd $WORKSPACE
```

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/philosophers$ cd $WORKSPACE
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 16. 메인 디렉터리

5 단계. 시뮬레이터에서 .hex 파일을 로드합니다.

```
> fusesoc run --target=sim swervolf --
ram_init_file=zephyr/samples/philosophers/App.hex
```

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ fusesoc run --target=sim swervolf
--ram_init_file=zephyr/samples/philosophers/App.hex
WARNING: Unknown item compilation_mode in section Xsin
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
=====
INFO: Generating ::swervolf-swerv_default_config:0.7.3
INFO: Generating ::swervolf-version:0.7.3
INFO: Generating ::swervolf-wb_intercon:0.7.3
Found master io
Found slave rom
Found slave spi_flash
Found slave sys
Found slave uart
=====
INFO: Setting up project
```

그림 17. fusesoc 실행

이제 다음 출력이 표시됩니다:

```
make[1]: Leaving directory '/home/hamza/SweRVolf/build/swervolf_0.7.3/sim-verilator'
INFO: Running
INFO: Running simulation
Loading RAM contents from /home/hamza/SweRVolf/zephyr/samples/philosophers/App.hex
Releasing reset
*** Booting Zephyr OS build zephyr-v2.4.0 ***

Philosopher 0 [P: 3]    HOLDING ONE FORK
Philosopher 1 [P: 2]    EATING  [ 125 ms ]
Philosopher 2 [P: 1]    THINKING [ 175 ms ]
Philosopher 3 [P: 0]    EATING  [ 325 ms ]
Philosopher 4 [C:-1]    THINKING [ 400 ms ]
Philosopher 5 [C:-2]    STARVING

Demo Description
-----
An implementation of a solution to the Dining Philosophers
problem (a classic multi-thread synchronization problem).
This particular implementation demonstrates the usage of multiple
preemptible and cooperative threads of differing priorities, as
well as dynamic mutexes and thread sleeping.
```

그림 18. Zephyr philosophers 출력

7. 하드웨어용 Zephyr 애플리케이션 구축

이제 하드웨어에서 Zephyr 를 실행하는 SwerVolf 용 프로그램을 빌드하는 방법을 보여줍니다.

1. Zephyr Blinky 예시

Blinky 는 다음을 사용하여 LED 를 영원히 깜박이는 간단한 응용 프로그램, 'GPIO API <gpio_api>' 입니다. 아래 소스 코드는 GPIO 핀을 출력으로 구성하는 방법으로, GPIO 를 켜고 끕니다.

```

1
2  /*
3   * Copyright (c) 2016 Intel Corporation
4   *
5   * SPDX-License-Identifier: Apache-2.0
6   */
7
8  #include <zephyr.h>
9  #include <device.h>
10 #include <devicetree.h>
11 #include <drivers/gpio.h>
12
13 /* 1000 msec = 1 sec */
14 #define SLEEP_TIME_MS 1000
15
16 /* The devicetree node identifier for the "led0" alias. */
17 #define LED0_NODE DT_ALIAS(led0)
18
19 #if DT_NODE_HAS_STATUS(LED0_NODE, okay)
20 #define LED0 DT_GPIO_LABEL(LED0_NODE, gpios)
21 #define PIN DT_GPIO_PIN(LED0_NODE, gpios)
22 #define FLAGS DT_GPIO_FLAGS(LED0_NODE, gpios)
23 #else
24 /* A build error here means your board isn't set up to blink an LED. */
25 #error "Unsupported board: led0 devicetree alias is not defined"
26 #define LED0 ""
27 #define PIN 0
28 #define FLAGS 0
29 #endif
30
31 void main(void)
32 {
33     const struct device *dev;
34     bool led_is_on = true;
35     int ret;
36
37     dev = device_get_binding(LED0);
38     if (dev == NULL) {
39         return;
40     }
41
42     ret = gpio_pin_configure(dev, PIN, GPIO_OUTPUT_ACTIVE | FLAGS);
43     if (ret < 0) {
44         return;
45     }
46
47     while (1) {
48         gpio_pin_set(dev, PIN, (int)led_is_on);
49         led_is_on = !led_is_on;
50         k_msleep(SLEEP_TIME_MS);
51     }
52 }

```


그림 19. Blinky 예제의 main.c

이 예제의 경로는 다음과 같습니다.

```
$WORKSPACE/zephyr/samples/basic/blinky/
```

위 경로로 이동한 후 터미널에서 다음 명령을 실행하여 예제를 빌드하고 ".elf" 및 ".bin" 파일을 생성합니다.

```
> west build -b swervolf_nexys
```

코드를 빌드하면 이제 build/zephyr/zephyr.elf 에 실행 가능한 .elf 파일이 있고 build/zephyr/zephyr.bin.에 .bin 파일이 있습니다.

실행 파일은 디버거를 사용하여 SweRVolf 에 로드할 수 있으며 바이너리 파일은 다음 섹션에 설명된 대로 시뮬레이션을 위해 .hex 파일로 변환하고 RAM 에 로드할 수 있습니다.

8. 하드웨어에서 Zephyr 애플리케이션 실행

Nexys A7 보드에서 애플리케이션을 실행하려면 OpenOCD 를 사용하여 프로그램을 로드해야 합니다.

1 단계. Nexys A7 보드를 컴퓨터에 연결하고 전원을 켜 다음 Workspace 디렉터리에서 FPGA 빌드 명령을 실행합니다.

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

```
***** Xilinx cs_server v2019.2.0
**** Build date : Nov 07 2019-10:41:48
** Copyright 2017-2019 Xilinx, Inc. All Rights Reserved.

INFO: Trying to use hardware target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A
INFO: Opened hardware target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A on try 1.
INFO: Found xc7a100tcsg324-1 as part of xc7a100t_0.
INFO: Programming bitstream to device xc7a100t_0 on target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A.
INFO: [Labtools 27-3164] End of startup status: HIGH
INFO: [Labtoolstcl 44-464] Closing hw_target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A

INFO: SUCCESS! FPGA xc7a100tcsg324-1 successfully programmed with bitstream swervolf_0.7.3.bit.
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 20. FPGA 빌드 실행

2 단계. OpenOCD 로 보드를 프로그래밍합니다.

```
> openocd -c "set BITFILE build/swervolf_0.7.3/nexys_a7-
vivado/swervolf_0.7.3.bit" -f
$SWERVOLF_ROOT/data/swervolf_nexys_program.cfg
```



```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ openocd -c "set BITFILE build/swervolf_0.7.3/nexys_a7-vivado/swervolf_0.7.3.bit"
-f $SWERVOLF_ROOT/data/swervolf_nexys_program.cfg
Open On-Chip Debugger 0.11.0-rc1+dev-01535-g3651cbdfd (2021-01-30-12:10)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
build/swervolf_0.7.3/nexys_a7-vivado/swervolf_0.7.3.bit
DEPRECATED! use 'adapter driver' not 'interface'
DEPRECATED! use 'adapter speed' not 'adapter_khz'
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi_tdo_sample_edge falling"
Info : clock speed 10000 kHz
Info : JTAG tap: xc7.tap tap/device found: 0x13631093 (mfg: 0x049 (Xilinx), part: 0x3631, ver: 0x1)
Warn : gdb services need one or more targets defined
shutdown command invoked
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 21. OpenOCD 실행

3 단계. OpenOCD 를 SweRVolf 와 연결합니다.

➤ `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-g3651cbdfd (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
```

그림 22. OpenOCD 연결

4 단계. "ctrl + shift + t"를 사용하여 세 번째 터미널을 열고 다음 명령을 사용하여 OpenOCD 를 통해 디버그 세션에 연결합니다.

➤ `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
>
```

그림 23. telnet

OpenOCD 는 `load_image /path/to/file.elf` 를 실행하여 ELF 프로그램 파일 로드를 지원합니다. 경로는 OpenOCD 가 시작된 디렉토리에 상대적임을 기억하십시오.

➤ `load_image zephyr/samples/basic/blink/build/zephyr/zephyr.elf`

```
> load_image zephyr/samples/basic/blink/build/zephyr/zephyr.elf
14848 bytes written at address 0x00000000
downloaded 14848 bytes in 1.420706s (10.206 KiB/s)
> 
```

그림 24. 이미지 .elf 파일 로드

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

➤ reg pc 0

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

그림 25. 프로그램 카운터를 0 으로 설정

이제 다음 명령을 사용하여 프로그램을 시작하십시오.

➤ resume

```
> resume
> 
```

그림 26. 프로그램 시작

이제 Nexys A7 보드의 맨 오른쪽 LED 가 깜박이기 시작하는 것을 볼 수 있습니다.

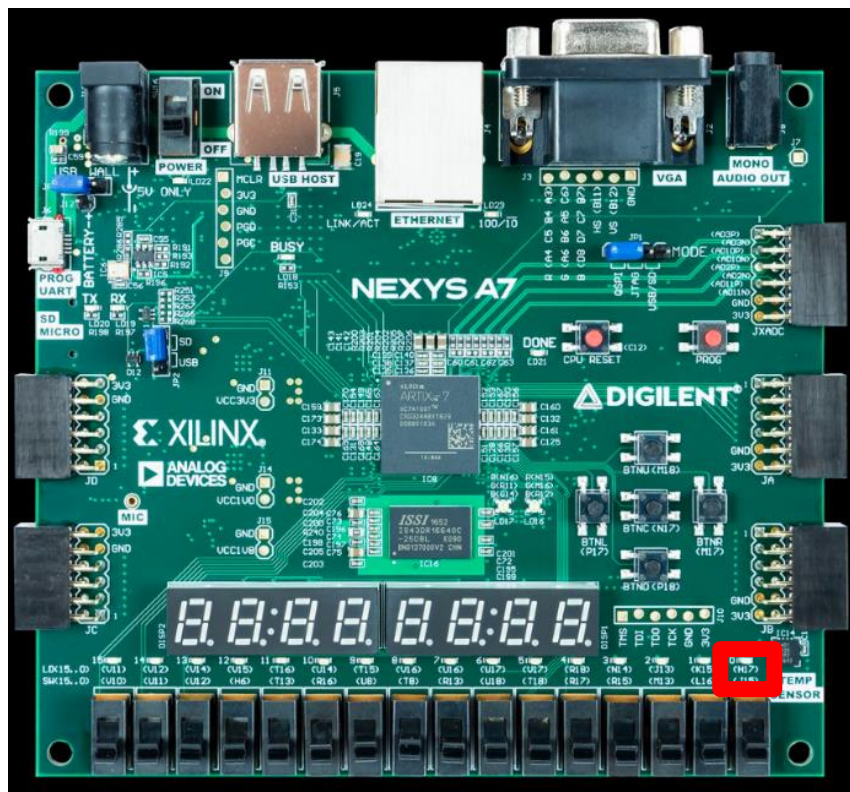


그림 27.LED 깜박임

이제 "ctrl + c"를 눌러 프로그램을 종료할 수 있습니다.

9. Zephyr 애플리케이션 개발 개요

Zephyr의 빌드 시스템은 CMake를 기반으로 합니다. 빌드 시스템은 애플리케이션 중심이며 커널 소스 트리 구축을 시작하려면 Zephyr 기반 애플리케이션이 필요합니다. 애플리케이션 빌드는 구성을 제어하고 애플리케이션과 Zephyr 자체의 프로세스를 빌드하여 단일 바이너리로 컴파일합니다.

Zephyr의 기본 디렉토리는 Zephyr의 소스 코드, 커널 구성 옵션 및 빌드 정의를 호스팅합니다.

응용 프로그램 디렉토리의 파일은 Zephyr를 응용 프로그램과 연결합니다. 이 디렉토리에는 구성 옵션 및 소스 코드와 같은 모든 응용 프로그램별 파일이 포함되어 있습니다.

가장 단순한 형태의 애플리케이션에는 여기에 나열되고 아래에 설명된 내용이 있습니다:

```
/App
├── CMakeLists.txt
├── prj.conf
├── src
│   └── main.c
```

CMakeLists.txt: 이 파일은 빌드 시스템에 다른 애플리케이션 파일을 찾을 위치를 알려주고 애플리케이션 디렉토리를 Zephyr의 CMake 빌드 시스템과 연결합니다. 이 링크는 보드별 커널 구성 파일, 실제 또는 에뮬레이트된 하드웨어에서 컴파일된 바이너리를 실행 및 디버깅하는 기능 등과 같이 Zephyr의 빌드 시스템에서 지원하는 기능을 제공합니다.

커널 구성 파일: 응용 프로그램은 일반적으로 하나 이상의 커널 구성 옵션에 대한 응용 프로그램별 값을 지정하는 Kconfig 구성 파일(일반적으로 prj.conf 라고 함)을 제공합니다. 이러한 애플리케이션 설정은 보드별 설정과 병합되어 커널 구성을 생성합니다.

응용 프로그램 소스 코드 파일: 응용 프로그램은 일반적으로 C 또는 어셈블리 언어로 작성된 하나 이상의 응용 프로그램별 파일을 제공합니다. 이러한 파일은 일반적으로 **src** 라는 하위 디렉토리에 있습니다.

10. 새로운 Zephyr 애플리케이션 생성

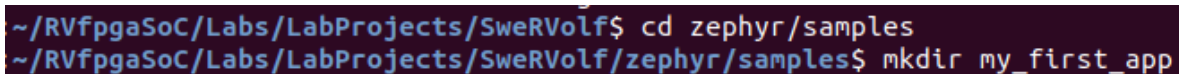
새 응용 프로그램 디렉토리를 만들려면 다음 단계를 따르십시오.

1 단계. Samples 디렉토리로 변경합니다.

```
> cd zephyr/samples
```

2 단계. 애플리케이션을 위한 새 디렉토리 생성:

```
> mkdir my_first_app
```



```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ cd zephyr/samples
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples$ mkdir my_first_app
```

그림 28. 프로젝트 디렉토리 만들기

3 단계. 모든 애플리케이션 소스 코드는 src 라는 하위 디렉토리에 두는 것이 좋습니다.

이렇게 하면 프로젝트 파일과 소스 파일을 더 쉽게 구별할 수 있습니다.

```
> cd my_first_app
> mkdir src
```

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples$ cd my_first_app
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app$ mkdir src
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app$
```

그림 29. 프로젝트 디렉토리 안에 src 디렉토리 만들기

4 단계. src 디렉토리에 들어가 애플리케이션의 메인 소스 파일인 "main.c"를 생성합니다.

- cd src
- nano main.c

```
~/SweRVolf/zephyr/samples/my_first_app$ cd src/
~/SweRVolf/zephyr/samples/my_first_app/src$ nano main.c
```

그림 30. "main.c" 파일 생성

Nano Editor 는 아래 그림과 같이 ubuntu 터미널에서 열립니다.



그림 31. GNU nano 편집기

5 단계. 나노 편집기에서 다음 코드를 복사합니다. 이 코드는 "hello_world"와 "blinky" 예제 소스 코드를 혼합한 것입니다.

```
#include <zephyr.h>
#include <sys/printk.h>
#include <device.h>
#include <devicetree.h>
#include <drivers/gpio.h>

/* 1000 msec = 1 sec */
#define SLEEP_TIME_MS 1000
```

```

/* The devicetree node identifier for the "led0" alias. */
#define LED0_NODE DT_ALIAS(led0)

#if DT_NODE_HAS_STATUS(LED0_NODE, okay)
#define LED0 DT_GPIO_LABEL(LED0_NODE, gpios)
#define PIN DT_GPIO_PIN(LED0_NODE, gpios)
#define FLAGS DT_GPIO_FLAGS(LED0_NODE, gpios)
#else
/* A build error here means your board isn't set up to blink an LED. */
#error "Unsupported board: led0 devicetree alias is not defined"
#define LED0 ""
#define PIN 0
#define FLAGS 0
#endif

void main(void)
{
    const struct device *dev;
    bool led_is_on = true;
    int ret;

    dev = device_get_binding(LED0);
    if (dev == NULL) {
        return;
    }

    ret = gpio_pin_configure(dev, PIN, GPIO_OUTPUT_ACTIVE | FLAGS);
    if (ret < 0) {
        return;
    }

    while (1) {
        gpio_pin_set(dev, PIN, (int)led_is_on);
        led_is_on = !led_is_on;
        k_msleep(SLEEP_TIME_MS);
        printk("This Zephyr Application is Running on %s\n", CONFIG_BOARD);
    }
}

```

그림 32. "main.c" 코드

코드 작성이 끝나면 **"ctrl + x"**를 눌러 종료합니다.

```

GNU nano 2.9.3

#include <zephyr.h>
#include <sys/printf.h>
#include <device.h>
#include <devicetree.h>
#include <drivers/gpio.h>

/* 1000 msec = 1 sec */
#define SLEEP_TIME_MS 1000

/* The devicetree node identifier for the "led0" alias. */
#define LED0_NODE DT_ALIAS(led0)

#if DT_NODE_HAS_STATUS(LED0_NODE, okay)
#define LED0 DT_GPIO_LABEL(LED0_NODE, gpios)
#define PIN DT_GPIO_PIN(LED0_NODE, gpios)
#define FLAGS DT_GPIO_FLAGS(LED0_NODE, gpios)
#else
/* A build error here means your board isn't set up to blink an LED. */
#error "Unsupported board: led0 devicetree alias is not defined"
#define LED0 ""
#define PIN 0
#define FLAGS 0
#endif

void main(void)
{
    const struct device *dev;
    bool led_is_on = true;
    int ret;

    dev = device_get_binding(LED0);
    if (dev == NULL) {
        return;
    }

    ret = gpio_pin_configure(dev, PIN, GPIO_OUTPUT_ACTIVE | FLAGS);
    if (ret < 0) {
        return;
    }

    while (1) {
        gpio_pin_set(dev, PIN, (int)led_is_on);
        led_is_on = !led_is_on;
        k_msleep(SLEEP_TIME_MS);
        printf("This Zephyr Application is Running on %s\n", CONFIG_BOARD);
    }
}

```

[^]G Get Help [^]O Write Out [^]W Where Is [^]K Cut Text [^]J Justify
[^]X Exit [^]R Read File [^]\ Replace [^]U Uncut Text [^]T To Spell

그림 33. main.c 파일 코드

그런 다음 파일을 저장할지 묻는 메시지가 표시되며 예를 보려면 "y"를 눌러야 합니다.

```

Save modified buffer? (Answering "No" will DISCARD changes.)
Y Yes
N No      ^C Cancel

```

그림 34. main.c 파일 저장

"Enter"를 눌러 "main.c"라는 이름으로 파일을 저장합니다.

```

File Name to Write: main.c
^G Get Help
^C Cancel

```

그림 35. main.c 이름 확인

6 단계. 이제 src 디렉토리를 벗어나 "CMakeLists.txt" 및 "prj.conf" 파일을 생성해야 합니다.

- cd ..
- nano CMakeLists.txt

```
~/SweRVolf/zephyr/samples/my_first_app/src$ cd ..
~/SweRVolf/zephyr/samples/my_first_app$ nano CMakeLists.txt
```

그림 36. CMakeLists.txt 생성

나노 편집기에 다음 코드를 복사합니다.

```
cmake_minimum_required(VERSION 3.13.1)

find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(my_first_app)

target_sources(app PRIVATE src/main.c)
```

그림 37. "CMakeLists.txt" 파일 코드

이제 "main.c" 파일을 저장하기 위해 수행한 것과 동일한 단계를 수행하십시오.



그림 38. 나노 편집기

이제 프로젝트 구성 파일을 만듭니다.


```
~/SweRVolf/zephyr/samples/my_first_app$ nano prj.conf
~/SweRVolf/zephyr/samples/my_first_app$
```

그림 39. 프로젝트 구성 파일 생성

응용 프로그램 구성 옵션은 응용 프로그램 디렉터리의 prj.conf 에 설정됩니다. 소스 코드에서 LED 를 사용하고 있으므로 "CONFIG_GPIO" 매개변수를 yes 로 설정해야 합니다.

```
CONFIG_GPIO=y
```

그림 40. "prj.conf" 코드

```
GNU nano 2.9.3 prj.conf Modified
CONFIG_GPIO=y
^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit ^R Read File ^\ Replace ^U Uncut Text
```

그림 41. "prj.conf" 나노 편집기

이제 "prj.conf" 파일을 저장합니다.

7 단계. "my_first_app"에 대한 코드를 빌드합니다:

```
> west build -b swervolf_nexys
```

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app$ west build -b swervolf_nexys
-- west build: generating a build system
Including boilerplate (Zephyr base): /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/cmake/app/boilerplate.cmake
-- Application: /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app
-- Zephyr version: 2.4.0 (/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr)
-- Found Python3: /usr/bin/python3.6 (found suitable exact version "3.6.9") found components: Interpreter
-- Found west (found suitable version "0.9.0", minimum required is "0.7.1")
-- Board: swervolf_nexys
-- Cache files will be written to: /home/hamza/.cache/zephyr
ZEPHYR_TOOLCHAIN_VARIANT not set, trying to locate Zephyr SDK
-- Found toolchain: zephyr (/home/hamza/zephyr-sdk-0.12.2)
-- Found dtc: /home/hamza/zephyr-sdk-0.12.2/sysroots/x86_64-pokysdk-linux/usr/bin/dtc (found suitable version "1.5.0",
red is "1.4.6")
```

```
-- Configuring done
-- Generating done
-- Build files have been written to: /home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app/build
-- west build: building application
[1/109] Preparing syscall dependency handling

[43/109] Building C object zephyr/CMakeFiles/zephyr.dir/drivers/interrupt_controller/intc_swerv_pic.c.obj
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c: In function 'swerv_pic_read':
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c:47:10: warning: cast to pointer
from integer of different size [-Wint-to-pointer-cast]
  47 |     return *(volatile uint32_t *) (DT_INST_REG_ADDR(0) + reg);
      |           ^
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c: In function 'swerv_pic_write':
/home/hamza/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/drivers/interrupt_controller/intc_swerv_pic.c:52:3: warning: cast to pointer
from integer of different size [-Wint-to-pointer-cast]
  52 |     *(volatile uint32_t *) (DT_INST_REG_ADDR(0) + reg) = val;
      |     ^
[104/109] Linking C executable zephyr/zephyr_prebuilt.elf
Memory region      Used Size  Region Size  %age Used
RAM:                17968 B      8 MB         0.21%
IDT_LIST:           41 B        2 KB         2.00%
[109/109] Linking C executable zephyr/zephyr.elf
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app$
```

그림 42. "my_first_app" 빌드

바이너리가 성공적으로 생성되었습니다. 이제 Nexys A7 보드에서 "my_first_app" 프로그램을 실행합니다.

9 단계. WORKSPACE 디렉터리로 이동합니다:

➤ `cd $WORKSPACE`

```
~/RVfpgaSoC/Labs/LabProjects/SweRVolf/zephyr/samples/my_first_app$ cd $WORKSPACE
~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 43. Workspace 디렉터리로 이동

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

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

```
***** Xilinx cs_server v2019.2.0
**** Build date : Nov 07 2019-10:41:48
** Copyright 2017-2019 Xilinx, Inc. All Rights Reserved.

INFO: Trying to use hardware target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A
INFO: Opened hardware target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A on try 1.
INFO: Found xc7a100tcsg324-1 as part of xc7a100t_0.
INFO: Programming bitstream to device xc7a100t_0 on target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A.
INFO: [Labtools 27-3164] End of startup status: HIGH
INFO: [Labtoolstcl 44-464] Closing hw_target localhost:3121/xilinx_tcf/Digilent/210292B0EF83A

INFO: SUCCESS! FPGA xc7a100tcsg324-1 successfully programmed with bitstream swervolf_0.7.3.bit.
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 44. FPGA 빌드 실행

11 단계. OpenOCD 로 보드를 프로그래밍합니다.

➤ `openocd -c "set BITFILE build/swervolf_0.7.3/nexys_a7-vivado/swervolf_0.7.3.bit" -f $SWERVOLF_ROOT/data/swervolf_nexys_program.cfg`

```
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$ openocd -c "set BITFILE build/swervolf_0.7.3/nexys_a7-vivado/swervolf_0.7.3.bit"
-f $SWERVOLF_ROOT/data/swervolf_nexys_program.cfg
Open On-Chip Debugger 0.11.0-rc1+dev-01535-g3651cbdfd (2021-01-30-12:10)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
build/swervolf_0.7.3/nexys_a7-vivado/swervolf_0.7.3.bit
DEPRECATED! use 'adapter driver' not 'interface'
DEPRECATED! use 'adapter speed' not 'adapter_khz'
Info : ftdi: if you experience problems at higher adapter clocks, try the command "ftdi_tdo_sample_edge falling"
Info : clock speed 10000 kHz
Info : JTAG tap: xc7.tap tap/device found: 0x13631093 (mfg: 0x049 (Xilinx), part: 0x3631, ver: 0x1)
Warn : gdb services need one or more targets defined
shutdown command invoked
hamza@imagination:~/RVfpgaSoC/Labs/LabProjects/SweRVolf$
```

그림 45. OpenOCD 실행

12 단계. OpenOCD 를 SweRVolf 와 연결합니다.

➤ `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-g3651cbdfd (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
```

그림 46. OpenOCD 연결됨

13 단계. "Ctrl + Shift + t"를 사용하여 세 번째 터미널을 열고 다음 명령을 사용하여 OpenOCD 를 통해 디버그 세션에 연결합니다.

➤ `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
>
```

그림 47. telnet

OpenOCD 는 `load_image /path/to/file.elf` 를 실행하여 ELF 프로그램 파일 로드를 지원합니다. 경로는 OpenOCD 가 시작된 디렉토리에 상대적임을 기억하십시오.

➤ `load_image zephyr/samples/my_first_app/build/zephyr/zephyr.elf`

```
> load_image zephyr/samples/my_first_app/build/zephyr/zephyr.elf
14672 bytes written at address 0x00000000
downloaded 14672 bytes in 1.410859s (10.156 KiB/s)
> █
```

그림 48. 이미지 .elf 파일 로드

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

➤ reg pc 0

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

그림 49. 프로그램 카운터를 0 으로 설정

이제 다음 명령을 사용하여 프로그램을 시작하십시오.

➤ resume

```
> resume
> █
```

그림 50. 프로그램 시작

보드의 LED 가 깜박이기 시작합니다.

14 단계. "Ctrl + Shift + t"를 사용하여 새 터미널 탭을 열고 다음 명령을 사용하여 PuTTY 를 엽니다.

➤ 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
```

그림 51. PuTTY 열기

여기에서 PuTTY 를 Nexys A7 보드용 직렬 콘솔로 사용할 것입니다.

15 단계. 다음 구성을 설정합니다.

연결 유형을 "**직렬(serial)**"로 선택한 다음 직렬 회선으로 **"/dev/ttyUSB1"**을 입력하고 속도를 **"115200"**으로 설정합니다. 이제 "open"를 클릭하여 직렬 콘솔을 시작합니다.

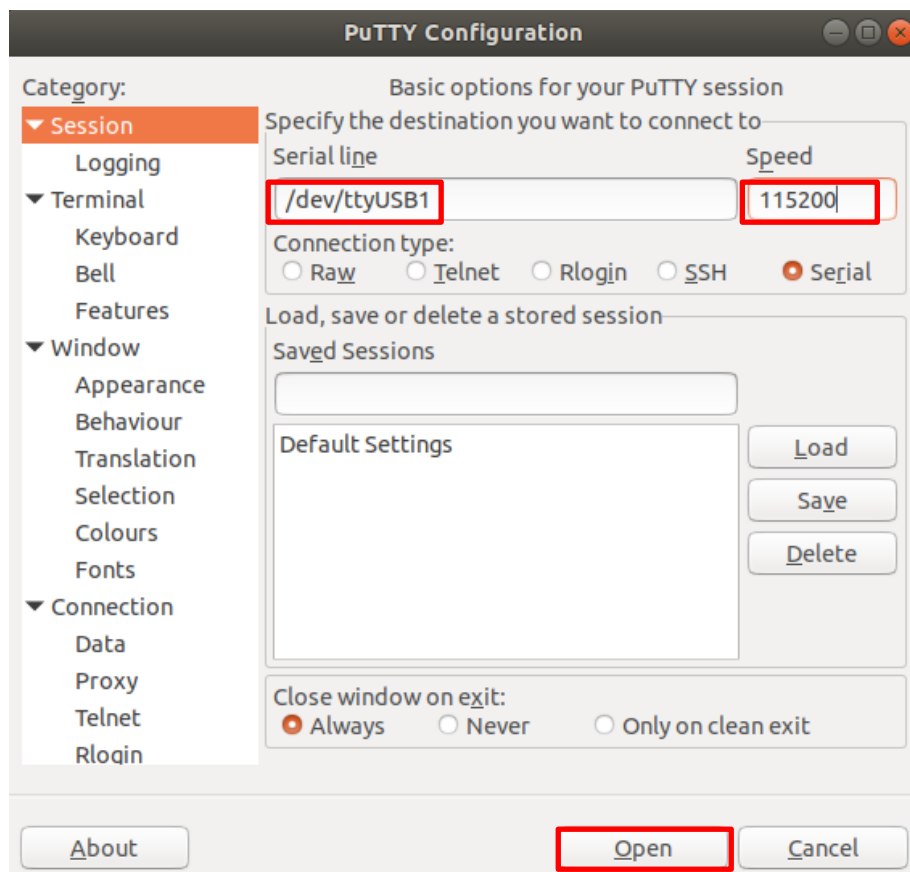


그림 52. PuTTY 구성

직렬 콘솔에서 "my_first_app" 프로그램의 출력을 볼 수 있습니다(그림 53 참조).

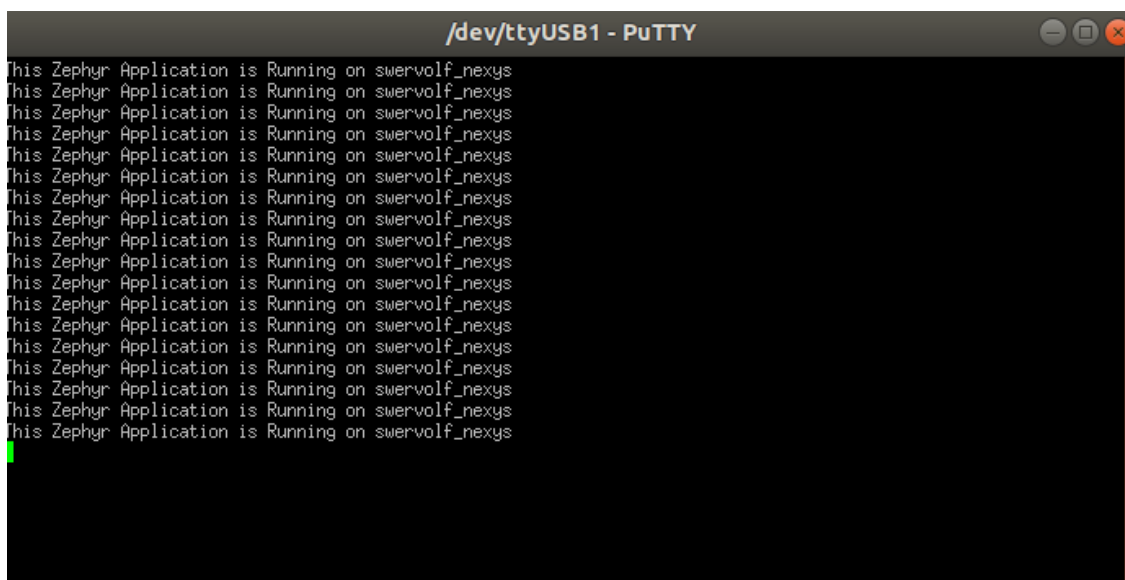


그림 53. 직렬 콘솔 출력

참고: 직렬 콘솔을 열 수 없는 경우 직렬 라인으로 "/dev/ttyUSB0"을 입력하십시오.

직렬 콘솔의 출력 텍스트에서 볼 수 있듯이 이 zephyr 애플리케이션은 SweRVolf Nexys 에서 실행됩니다.