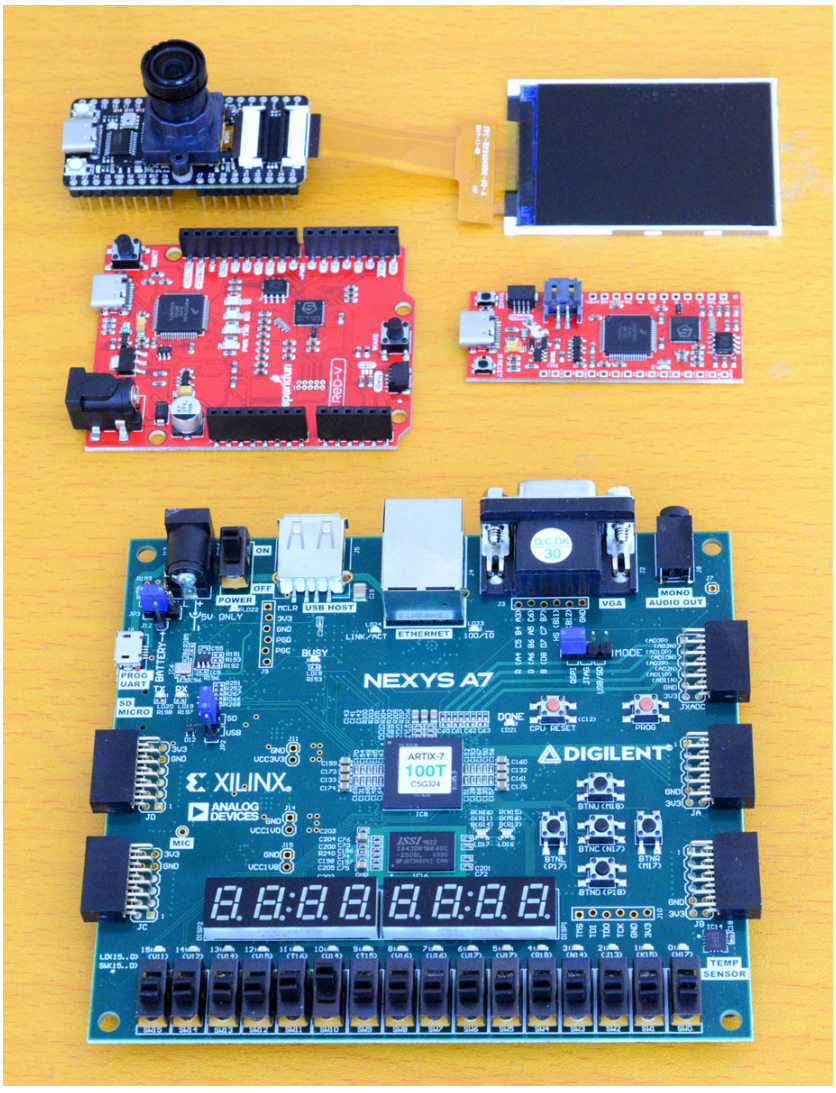
RISC-V 가이드


NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES
ETHERNET
USB HOST
VGA
MONO AUDIO OUT
TEMP SENSOR

Digi-Key 및 Imagination Technologies RISC-V 가이드

목차

1 소개

RISC-V는 ARM의 경쟁 기술로 활발하게 홍보되고 있는 새로운 컴퓨터 기술입니다.

이 가이드는 RISC-V 소프트웨어 및 하드웨어에 대한 지식을 얻고자 하는 Digi-Key 고객과 학생을 위한 간단한 소개 자료로 작성되었습니다.

2020년 9월 3일에 진행된 RISC-V 글로벌 포럼에서 Imagination Technologies는 2020년 11월에 시작한 University Programme의 일환으로 RISC-V를 위한 글로벌 학습 프로젝트인 "RVfpga: Complete Course in Understanding Computer Architecture"(RVfpga: 컴퓨터 아키텍처 이해를 위한 완벽한 과정)를 소개했습니다. 이 과정의 일부 자료는 https://university.imgtec.com 에서 등록 후 다운로드 받으실 수 있습니다.

이 가이드는 프로젝트 시작 시 이 글의 저자가 그랬던 것처럼 RISC-V에 대한 사전 지식이 거의 또는 전혀 없는 사람들을 대상으로 작성되었습니다. 대학교에서 컴퓨터 아키텍처를 공부하고 있는 학생과 강의실 또는 가정에서 좀더 RISC-V에 대하여 공부 하고자 하는 전기 전자 기술자/컴퓨터 엔지니어가 이 글의 독자입니다. Linux에 대한 기본 지식은 약간 필요합니다.

1.1 RISC-V란?

RISC는 "축소 명령어 집합 컴퓨터"(**R**educed **I**nstruction **S**et **C**omputer)를 나타내고, 여기서 **V**는 로마 숫자 5를 나타냅니다. 따라서 RISC-V는 5세대 컴퓨터 코어 제품군이고, "Risk Five"(리스크 파이브)라고 읽습니다.

RISC-V 로고는 RISC-V International의 등록 상표입니다.

그림 1: RISC-V 로고. 이미지 출처: www.RISC-V.org

사양 등 자세한 정보는 www.RISC-V.org에서 확인할 수 있습니다.

1.2 실용적인 정보

본 가이드는 이론적으로 접근하는 대신 Digi-Key에서 구입할 수 있는 3가지 다른 보드를 사용하여 RISC-V에 대한 실제적인 경험을 제공해 드립니다. 아래 명시된 가격은 본 가이드 작성 당시에는 정확했지만 이후 소프트웨어에 따라 변경될 수 있습니다.

1. **쉽고 재미있고, 학생에게도 무리 없는 경제적인 가격**. Seeed Technologies Co Ltd Maix BiT 보드는 듀얼 코어 RISC-V 프로세서, 카메라, LCD 스크린, 이미지 인식 소프트웨어를 단돈 $25에 제공합니다. 이는 MicroPython을 사용하여 프로그래밍 됩니다. (본 가이드의 Maix BiT 섹션 링크).
2. **미드레인지 System on a Chip(Soc)**. SparkFun 보드 2개에서 C 또는 어셈블리 언어로 프로그래밍된 RISC-V SOC의 가격은 $30와 $36 입니다. (본 가이드의 SparkFun 섹션 링크).
3. **매우 지능적이지만 비싼 가격.** Digilent Nexys A7 보드에서 사용되는 Xilinx Field Programmable Gate Array(FPGA)에서 Western Digital SweRV 소프트 코어를 사용하며 가격은 $265입니다. 대신 소프트웨어 시뮬레이터에서 모두 실행할 수 있습니다. (본 가이드의 소프트 코어 섹션 링크).

사용되는 모든 소프트웨어는 무료입니다.

2 RISC-V의 배경

2.1 역사

RISC-V는 2010년 캘리포니아 대학교 버클리 캠퍼스에서 David Patterson, Krste Asanovic, Andrew Waterman, Yunsup Lee가 개발했습니다. 당시에는 Linux라는 오픈 소스 소프트웨어가 있었지만 이에 상응하는 오픈 소스 하드웨어는 없었습니다. 연구를 촉진하기 위해 이들은 자체 명령어 집합 구조(Instruction Set Architecture, ISA)를 만들었습니다. 2011년에 첫 번째 RISC-V 칩을 만들어 2014년에 첫 번째 상용 제품을 출시했습니다.

몇 년에 걸쳐 Microchip PIC, ARM, Atmel AVR, MIPS, SuperH, SPARC 등을 포함한 여러 가지 다른 RISC 구현이 이미 있었습니다. 최초의 RISC-I 구현은 1981년으로 거슬러 올라갑니다.

2.2 오픈 아키텍처와 오픈 소스

프로세서 또는 주문형 반도체(Application Specific Integrated Circuit, ASIC) 등과 같은 집적 회로에 ARM® 코어를 사용하려면 먼저 라이선스를 구입해야 합니다. 최신 코어의 비용은 수백만 달러에 이를 수 있고 시간도 걸립니다. NXP 또는 Freescale 등과 같은 다국적 대기업에게 이와 같은 비용은 총 개발 비용 1억 달러에 비하면 상대적으로 작은 돈입니다. 하지만 새로운 집적 회로가 완성되면 제품 하나가 팔릴 때마다 ARM에 기술 사용료를 지불해야 합니다.

하지만 RISC-V는 다릅니다. 라이선스 비용도 없고 기술 사용료도 없습니다. 즉, 언제든지 RISC-V 구현을 시작할 수 있고, 계속해서 발생하는 비용이 없습니다. 이는 인도, 중국 등 신흥 시장과 중소기업에게 아주 매력적인 점입니다. RISC-V 아키텍처는 고정되어 있어 향후 개발되는 제품과 이전 제품 간에 호환성이 보장됩니다.

또한 오픈 소스이기 때문에 설계를 수정할 수 있습니다. 성능을 개선하거나 해커가 쉽게 접근할 수 없도록 특수 명령어를 만들 수 있습니다.

RISC-V 개발 소프트웨어(특히 PlatformIO)는 무료입니다. Keil 컴파일러를 사용하는 ARM MDK 등과 같이 라이선스가 필요한 소프트웨어는 비용이 많이 들 수 있으며, 중요한 기한을 앞두고 언제든지 라이선스가 만료될 수 있습니다.

RISC-V 코어는 일반적으로 Windows보다는 Linux에서 구현되어 코어와 소프트웨어를 모두 오픈 소스로 만듭니다.

2.3 주요 업체

Andes Technology는 대만의 반도체 제조업체입니다. 이 회사의 제품 중 하나는 700MHz에서 동작하는 RISC-V 코어인 N22로, 0.013mm² 크기의 작은 칩 입니다. 이 제품은 웨어러블과 사물인터넷(Internet of Things, IoT) 응용 분야를 위해 설계되었습니다.

Imagination Technologies는 휴대폰, 태블릿, 컴퓨터, 차량 비전에 사용되는 그래픽 처리 장치(Graphic Processing Units, GPU)를 전문적으로 제조하는 지적 재산권(Intellectual Property, IP) 기업입니다. 가장 유명한 제품으로는 PowerVR이 있으며 10세대 GPU "A 시리즈"는 임베디드 RISC-V 컨트롤러를 사용합니다.

PlatformIO에서는 RISC-V 개발을 위한 무료 소프트웨어 도구를 제공합니다.

SiFive는 RISC-V의 최초 개발자 3명(Krste Asanović, Yunsup Lee, Andrew Waterman)이 설립했습니다. 이 회사는 RISC-V 코어, Systems on a Chip(SoC), IP 및 개발 보드를 제공합니다.

Western Digital(SanDisk 제품의 제조업체)은 향후 제품에 RISC-V를 사용하기 위해 노력 중이며 SweRV라는 자체 인증 RISC-V 코어 제품군을 생산하고 있습니다.

3 RISC-V에 대한 기본 사항

RISC-V를 컴퓨터 코어라고 하지만 보다 정확하게 말하면 RISC-V는 ISA이며, 기능을 구현하는 것은 개발자의 몫입니다.

RISC-V 사양은 RISC-V.org에서 Unprivileged ISA 사양(기본 작동) 및 Privileged ISA 사양(운영 체제와 함께 사용하는 경우)의 두 가지 문서로 제공합니다.

레벨	인코딩	이름	약어
0	00	사용자/애플리케이션	U
1	01	슈퍼바이저	S
2	10	예약됨	
3	11	머신	M

표 1: RISC-V Privilege 레벨. 이미지 출처: RISC-V.org

대부분의 응용 분야에서는 Zephyr 운영 체제를 비롯하여 사용자/애플리케이션 레벨이 사용됩니다.

3.1 코어 명명 규칙

RISC-V는 16 또는 32개 레지스터를 사용하여 32비트, 64비트 또는 128비트에서 구현되었으며, 명시된 명명 규칙을 사용합니다.

이름	기능
RV32I	정수 명령어 집합. 32비트, 32개 레지스터
RV32E	임베디드 디바이스를 위한 정수 명령어 집합. 32비트, 16개 레지스터
RV64I	정수 명령어 집합. 64비트, 32개 레지스터
RV128I	정수 명령어 집합. 128비트, 32개 레지스터

표 2: RISC-V Base ISA. 이미지 출처: RISC-V.org

또한 개발자는 어떤 명령어를 구현할지 선택할 수 있습니다. 가장 일반적인 MAFD에는 집합적으로 G(일반)로 지칭됩니다.

	Letter	Functionality
G	M	Integer multiplication and division
	A	Atomic Instructions
	F	Single precision floating point
	D	Double precision floating point
	Q	Quad precision
	L	Decimal floating point
	C	Compressed instructions (16 bit instructions)
	B	Bit manipulation
	J	Dynamically translated languages
	T	Transactional memory
	P	Packed SIMD instructions
	V	Vector operations
	N	User level interrupts
	H	Hypervisor

표 3: RISC-V Standard ISA 확장. 이미지 출처: RISC-V.org

최소 리소스를 사용하는 소형 SoC의 경우 RV32EMAB를 지정할 수 있습니다. 즉, *임베디드 디바이스를 위한 정수 명령어 집합, 32비트와 16개 레지스터, 정수 곱셈 및 나눗셈, Atomic 명령어*(인터럽트와의 경쟁을 피하기 위해)와 부동소수점 비교 연산 없음을 의미하지만 *비트 조작*은 사용합니다.

반면에, 4가지 일반 명령어 MAFD와 *비트 조작* 및 *사용자 레벨 인터럽트*를 비롯하여 64비트 RISC-V 구현이 필요한 경우 RV64GBN이라고 합니다.

3.2 다운로드 가능한 코어

많은 수의 RISC-V 코어가 전 세계 대학교에서 개발되었습니다. 코어는 VHDL, Verilog, System Verilog, 잘 알려지지 않은 Chisel 등 하드웨어 기술 언어(Hardware Description Languages, HDL)로 작성되었습니다. C 프로그래머의 경우 C와 구문이 유사하기 때문에 Verilog가 가장 사용하기 쉬울 수 있습니다. System Verilog는

Superset이고 유효성 검사에 사용됩니다. Verilog는 프로그래밍 언어가 아닌 하드웨어 기술 언어입니다.

사용 가능한 RISC-V 코어 목록은 https://github.com/riscv/riscv-cores-list를 참조하십시오.

그림 2는 일부 RISC-V 코어의 시점과 성능을 보여줍니다.

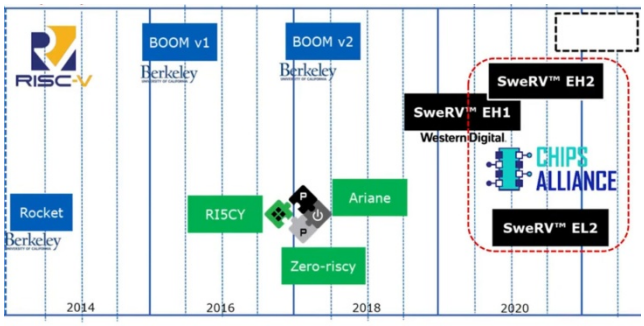


그림 2: Boom, Rocket, Riscy와 SweRV 코어. 이미지 출처: Western Digital

Boom은 Chisel로 작성된 Berkeley Out-of-Order RISC-V 프로세서입니다.

lowRISC의 Rocket 코어는 5단계 파이프라인을 사용하는 RV64G의 변형입니다.

RISCY는 취리히 연방 공과대학교와 볼로냐 대학교에서 개발한 간단한 코어입니다.

SweRV는 Western Digital에서 산업용 응용 분야를 위해 개발한 RISC-V 프로세서 제품군(3개 포함)입니다.

또한 Freedom은 SiFive에서 사용하는 코어입니다.

3.3 ARM 및 Intel과의 성능 비교

Western Digital에서 제공한 그림 3은 ARM, Intel과 RISC-V 코어 3개(Boom, Rocket, SwerRV)의 상대적 성능을 보여줍니다. CoreMark는 지정된 범위의 일반적인 작업(예: 정렬 알고리즘, 순환 중복 검사(Cyclic Redundancy Check) 상태 머신 등)을 수행할 때 마이크로프로세서의 성능을 비교하기 위해 Embedded Microprocessor Benchmark Consortium(EEMBC)에서 고안한 벤치마크입니다.

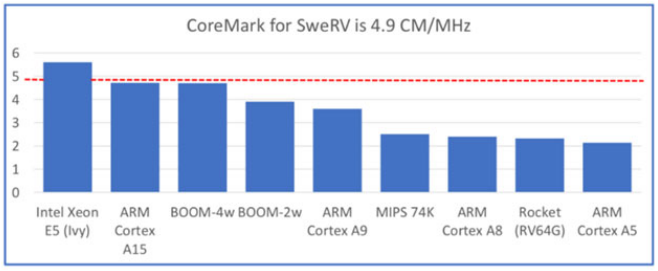


그림 3: Western Digital CoreMark. 이미지 출처: Western Digital

이 그림을 보면 성능 측면에서 다양한 RISC-V 코어가 ARM Cortex A5, ARM Cortex A8, ARM Cortex A9, ARM Cortex A15와 비슷합니다. Intel Xeon의 성능은 더 뛰어납니다.

4 실용적인 정보: Seeed Technologies Maix BiT 보드 사용

4가지 실습 보드 중 가장 저렴하고 사용이 간단합니다.

4.1 Maix BiT 보드 설명

Maix BiT 보드는 사물인터넷(IoT) 및 영상 인식 등과 같은 인공 지능(Artificial Intelligence, AI)을 위해 개발되었습니다. 이 보드는 저렴한 가격에 성능이 뛰어난 프로세서를 제공합니다.

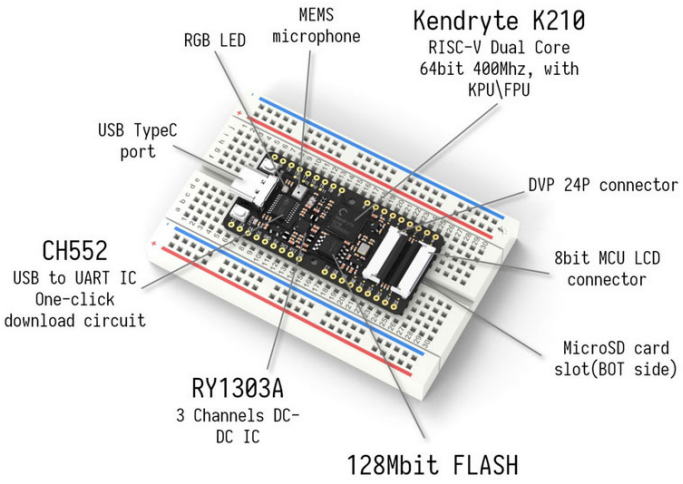


그림 4: 상세한 Maix BiT. 이미지 출처: Seeed Technologies Co Ltd

4.2 기본 Maix BiT 보드

다음 기본 Maix BiT 보드를 사용할 수 있습니다.

Seeed Technology Co. Ltd Sipeed Maix BiT RISC-V AI+IoT, Digi-Key 1597-1714-ND $14.65

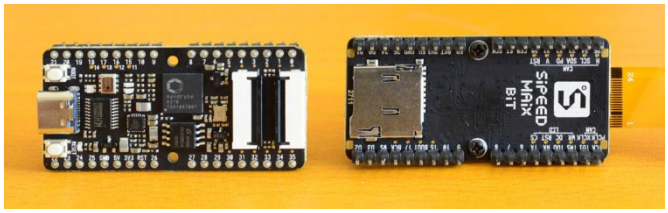


그림 5: Seeed Technologies Maix BiT 보드

4.3 Maix BiT 보드(카메라, LCD 포함)

실습 세션을 위해 카메라와 LCD 스크린이 포함된 다음 키트를 사용하는 것이 좋습니다.

Seeed Technology K210 Sipeed Maix BiT Kit RISC-V AI+IoT Digi-Key 1597-1713-ND $24.21

참고: Maix BiT 모드에는 USB-A to USB-C 케이블이 제공되지 않으므로 다음 제품을 별도로 구입해야 합니다.

Seeed Technology USB-C 케이블 Digi-Key 1597-106990248-ND $2.42

4.3.1 제공되는 Maix BiT 구성품

상자에는 Maix BiT 보드, 카메라, LCD 디스플레이가 들어 있습니다. 또한 드라이버 1개, 나사 + 나일론 스페이서가 각 3개씩 제공되는데, 실제로 나사와 스페이서는 2개씩만 필요합니다.

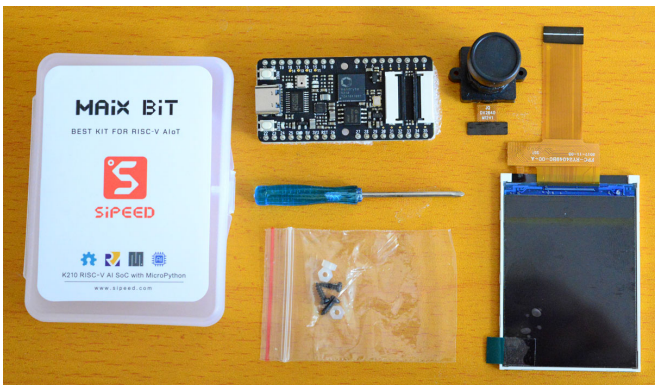


그림 6: 카메라, LCD, 구성품과 함께 MaixPy BiT 상자와 보드

4.3.2 MaixPy BiT 보드 조립

조립 시간: 약 5분. 구성품이 작기 때문에 약간 불편할 수 있습니다.

필요한 추가 도구: 집게(나사를 조이면서 나일론 스페이서를 집기 위해 필요함)

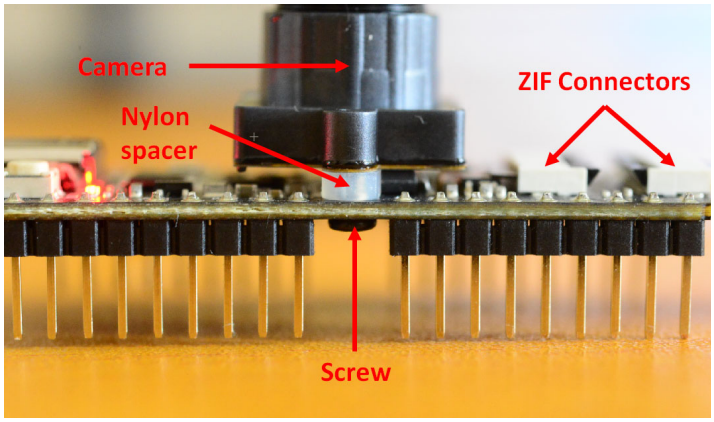


그림 7: 카메라 장착

4.3.3 Maix BiT 조립 지침

1. 나사와 나일론 스페이서 2개를 끼웁니다.
2. 검은색 잠금 장치가 위를 향한 상태에서 카메라 케이블을 보드 가운데 근처 흰색 ZIF 커넥터에 끼웁니다.
3. 검은색 잠금 장치를 아래로 눌러 카메라 케이블을 제자리에 고정시킵니다.
4. 나사 위에 카메라를 맞춰 정렬하려면 카메라 케이블을 구부려야 합니다.
5. 드라이버로 나사 2개를 조여 카메라를 제자리에 고정시킵니다.
6. 보드 가장자리 근처에 있는 ZIF 커넥터에 LCD 케이블을 집어 넣습니다.
7. 검은색 잠금 장치를 아래로 눌러 LCD를 제자리에 고정시킵니다.

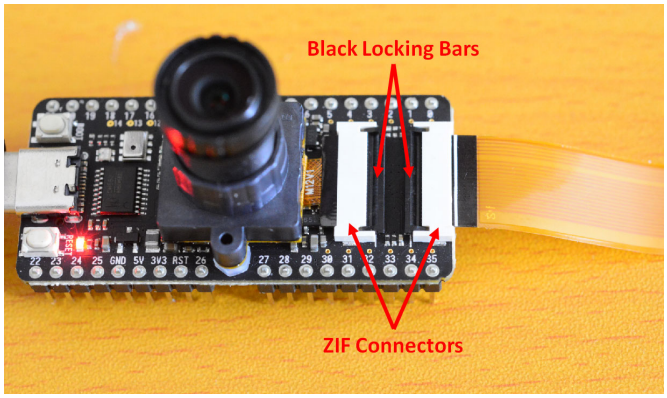


그림 8: ZIF 소켓에 맞춰 끼운 카메라와 LCD

4.4 Maix BiT 보드 소프트웨어

Maix BiT 보드는 MaixPy라고 하는 MicroPython 인터프리터가 미리 프로그래밍된 상태로 제공됩니다. 프로그램은 MaixPy IDE 소프트웨어를 사용하거나 직렬 터미널에 명령을 입력하여 작성할 수 있습니다.

Linux의 경우 추가 단계가 있긴 하지만 소프트웨어는 Windows용과 Linux용으로 다운로드할 수 있습니다. (본 가이드의 Linux 설치 부분 링크)

4.5 Windows에서 MaixPy IDE 설치

Maix BiT 보드를 실행하는 데 사용하는 통합 개발 환경(Integrated Development Environment, IDE)입니다.

설치 시간: 약 10분

MaixPy IDE 소프트웨어는 위치가 비어 있어 찾기가 약간 어렵습니다.

다음으로 이동: https://maixpy.sipeed.com/en

이 링크를 클릭하면 유용한 정보가 포함된 MaixPy 설명서 페이지로 이동합니다.

4.5.1 MaixPy 설명서 웹페이지

왼쪽에서 *Install MaixPyIDE(optional)(MaixPyIDE 설치 (선택 사항))*를 클릭합니다.

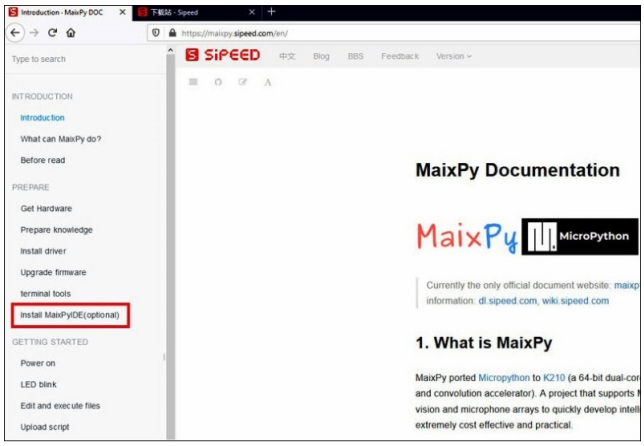


그림 9: MaixPy 설명서 페이지

4.5.2 MaixPy 다운로드 페이지로 이동

*dl.sipeed.com*을 클릭합니다.

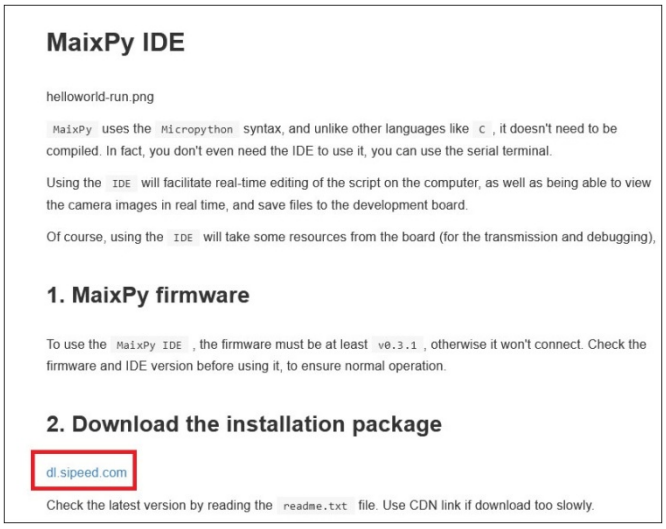


그림 10: 설치 패키지 다운로드

4.5.3 디렉터리 선택

_를 클릭합니다.

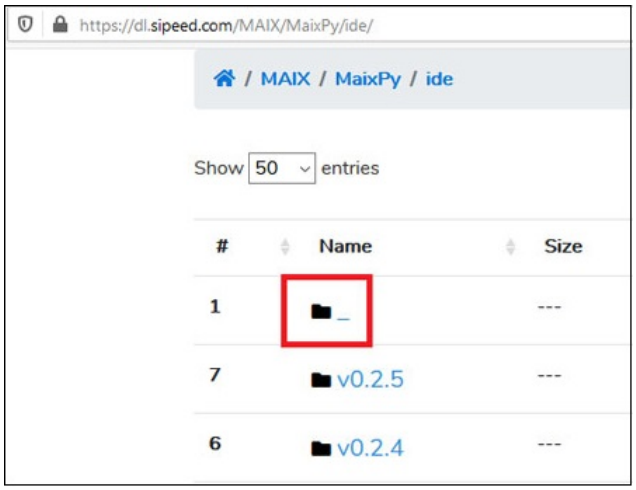


그림 11: _가 표시된 MaixPy IDE 폴더

4.5.4 최신 버전 선택

최신 버전을 클릭합니다. 여기서는 `v0.2.5`입니다.

그림 12: v0.2.5 선택

4.5.5 Windows용 MaixPY IDE 선택

`maixpy-ide-windows-0.2.5.exe`를 클릭합니다. 파일 크기는 85.5MB입니다.

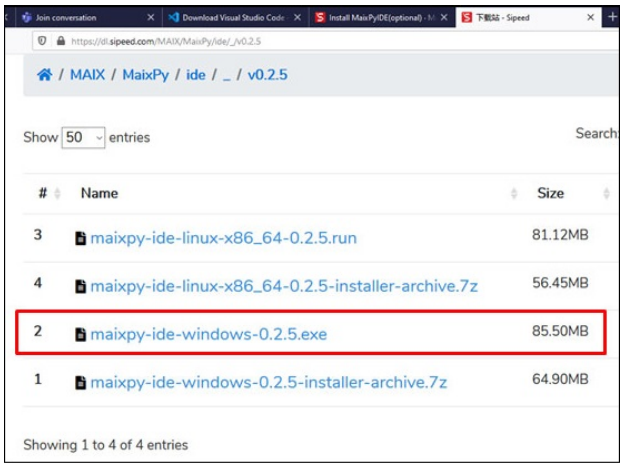


그림 13: Windows용 MaixPy IDE 클릭

4.5.6 MaixPy IDE Windows 다운로드

*Save File(파일 저장)*을 클릭하고 `maixpy-ide-windows-0.2.5.exe`를 두 번 클릭하여 설치합니다.

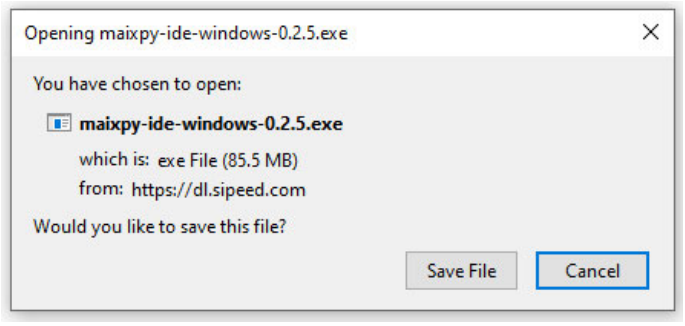


그림 14: maixpy-ide-windows 다운로드

프로그램이 다음 위치에 설치됩니다.

```
C:\Program Files
(x86)\MaixPyIDE\bin\maixpyide.exe
```

4.5.7 MaixPy Windows 바로 가기

기본적으로 MaixPy는 바탕 화면에 바로 가기 아이콘을 표시하지 않습니다. MaixPy의 바로 가기를 만들려면 `maixpyide.exe`를 마우스 오른쪽 버튼으로 클릭하고 *Create shortcut(바로 가기 만들기)*를 선택합니다. 그러면 바탕 화면에 바로 가기가 저장됩니다.

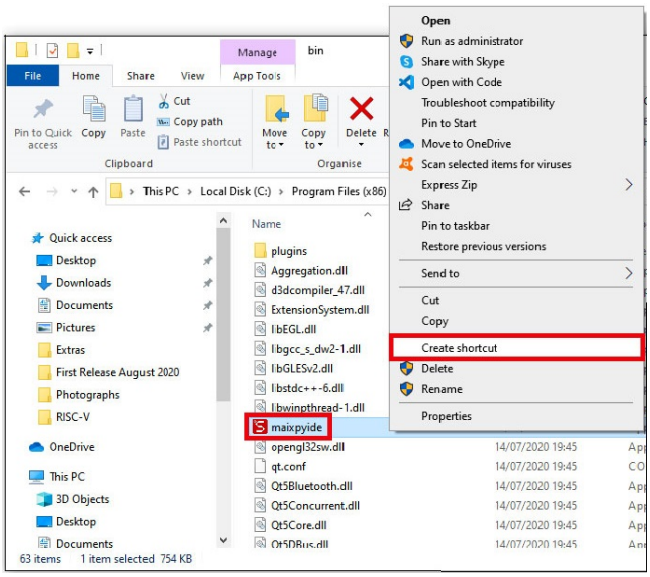


그림 15: MaixPy IDE 바로 가기

(MaixPy IDE에서 프로그램 실행 링크)

4.6 Linux를 사용하여 MaixPy IDE 설치

설치 시간: 약 10분 다운로드한 파일의 크기는 287MB입니다.

다운로드 및 설치하는 데 40분이 걸리는 명령어를 제공됩니다.

MaixPy IDE 소프트웨어는 위치가 비어 있어 찾기가 약간 어렵습니다.

다음으로 이동: https://maixpy.sipeed.com/en

이 링크를 클릭하면 유용한 정보가 포함된 MaixPy 설명서 페이지로 이동합니다.

4.6.1 MaixPy 설명서 웹페이지

*Install MaixPyIDE (optional)(MaixPyIDE 설치(선택 사항))*를 클릭합니다.

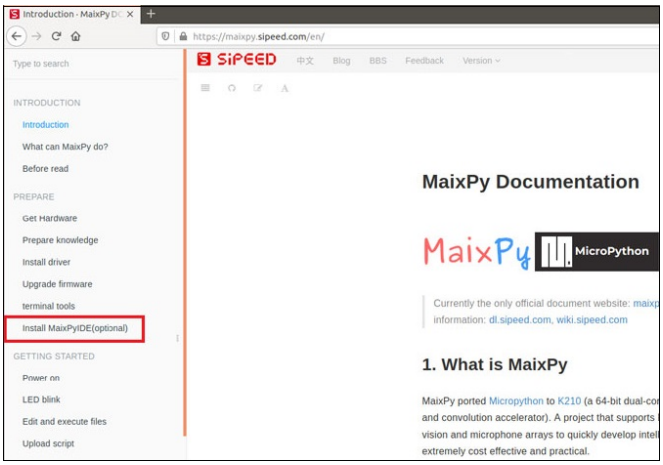


그림 16: MaixPy 설명서 페이지

4.6.2 Seeed 다운로드 페이지

*dl.seeed.com*을 클릭하여 다운로드 페이지로 이동합니다.

그림 17: Linux용 MaixPy 다운로드 설치 패키지

4.6.3 폴더 선택

_가 표시된 폴더를 클릭합니다.

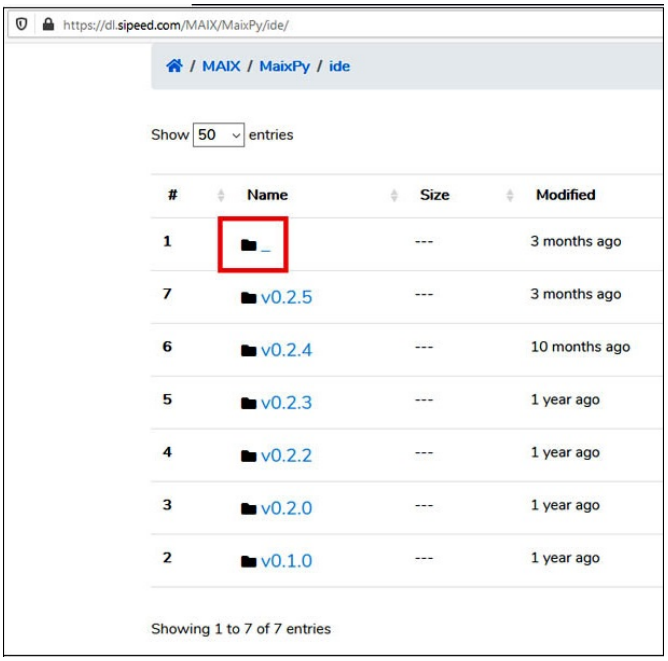


그림 18: _가 표시된 폴더 선택

4.6.4 최신 버전 선택

*v0.2.5*를 클릭합니다

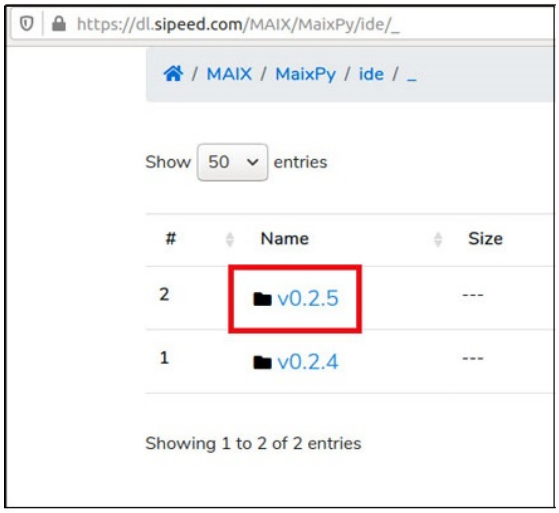


그림 19: v0.2.5 선택

4.6.5 파일 선택

다음 파일을 클릭합니다.

```
maixpy-ide-linux-x86_64.0.2.5.run
```

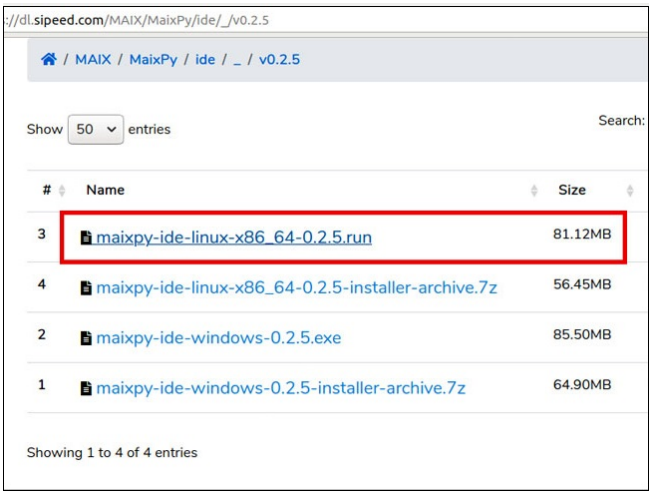


그림 20: Linux 실행 파일 선택

4.6.6 Linux 설치 파일 저장

*OK(확인)*를 클릭하여 파일을 저장합니다.

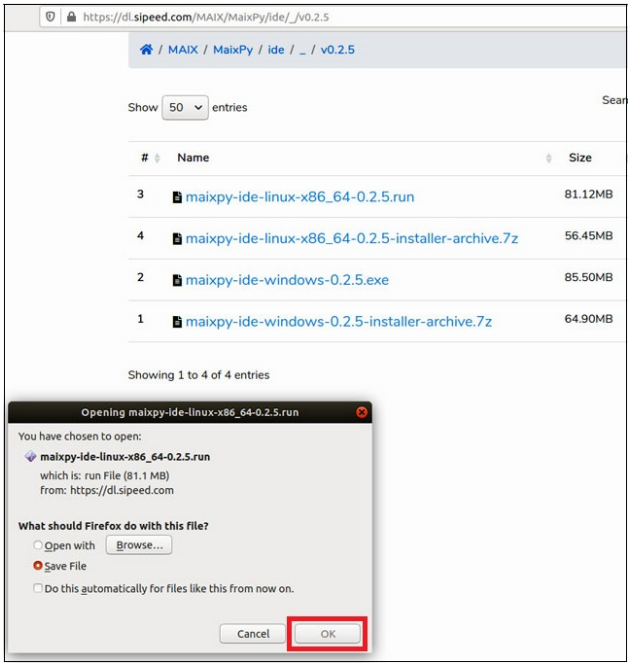


그림 21: Linux 설치 파일 저장

4.6.7 USB 포트에 접근

디폴트에서는 USB 포트를 사용할 수 없기 때문에 MaixPy 프로그램을 실행할 수 없습니다.

Ubuntu Terminal(Ubuntu 터미널) 

을 열고 다음과 같이 입력합니다.

```
sudo adduser myname dialout
```

여기서 myname은 컴퓨터 사용자 이름으로 바꾸고 /home/myname/과 동일한 형식으로 입력합니다.

컴퓨터를 다시 시작하여 변경 내용을 적용합니다.

이 단계를 수행하지 않으면 나중에 컴퓨터를 다시 시작하라는 메시지가 나타납니다.

4.6.8 다운로드한 파일을 Linux에서 실행

Ubuntu Terminal(Ubuntu 터미널) 

을 엽니다.

MaixPy에 제공된 지침은 버전 0.2.5가 아닌 버전 0.2.2에 대해 작성되었으므로 최신 버전이 아닙니다.

```
chmod +x maixpy-ide-linux-x86_64-0.2.2.run
./maixpy-ide-linux-x86_64-0.2.2.run
```

그림 22: MaixPy 명령어

다음과 같이 최신 버전의 파일 이름을 입력합니다.

```
chmod +x maixpy-ide-linux-x86_64-
0.2.5.run

./maixpy-ide-linux-x86_64-0.2.5.run
```

*MaixPy IDE 설치 마법사*가 나타나야 합니다.

4.6.9 MaixPy IDE 설치 마법사

그림 23: MaixPy 설치 마법사

*Next>(다음>)*를 클릭한 다음 기본값을 사용하여 마법사의 단계를 진행합니다. 마치면 열기 화면이 나타나야 합니다.

4.6.10 MaixPy 열기 화면

열기 화면에 카메라의 영상(프레임 버퍼)과 빨간색 녹색 파란색(Red Green Blue, RGB) 신호가 표시되지 않으면 화면 오른쪽에서 ◀ 아이콘을 클릭합니다.

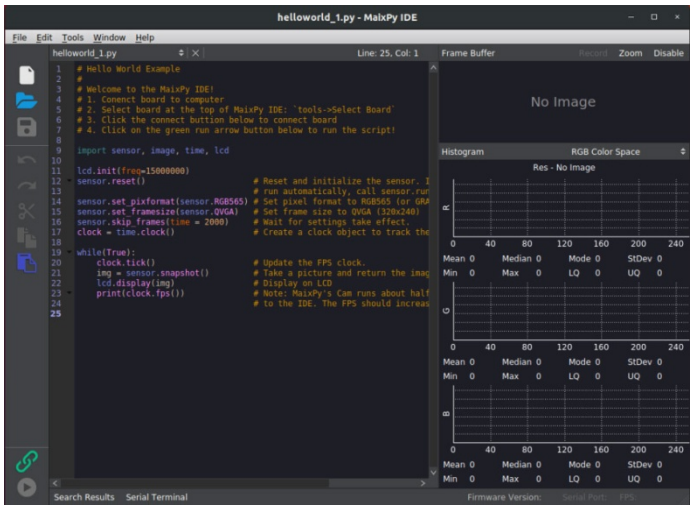


그림 24: MaixPy Linux IDE 화면

4.7 MaixPy IDE에서 프로그램 실행

Linux와 Windows는 절차가 비슷합니다.

4.7.1 보드 꽂기

USB 케이블을 컴퓨터에 꽂으면 환영 메시지가 LCD에 표시됩니다.

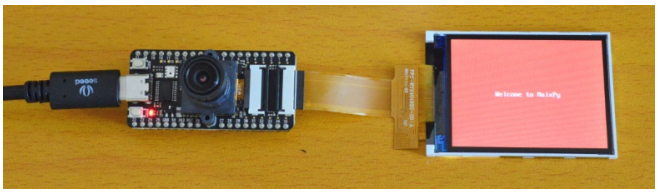


그림 25: Maix BiT 카메라와 작동하는 LCD

4.7.2 MaixPy 시작

MaixPy IDE가 아직 실행 중이 아닌 경우 Ubuntu 검색 도구로 이동하여 `MaixPy`를 입력합니다. *MaixPy IDE* 아이콘을 클릭합니다.

그림 26: MaixPy IDE Linux 아이콘

4.7.3 프로그램 선택

처음 전원을 켜면 기본적으로 `helloworld_1.py` 프로그램이 로드됩니다.

4.7.4 보드 선택

도구 모음에서 *Tools(도구)*, *Select Board(보드 선택)*, *Sipeed Maix Bit (with Mic)(Sipeed Maix Bit(마이크 포함))*를 차례대로 선택합니다.

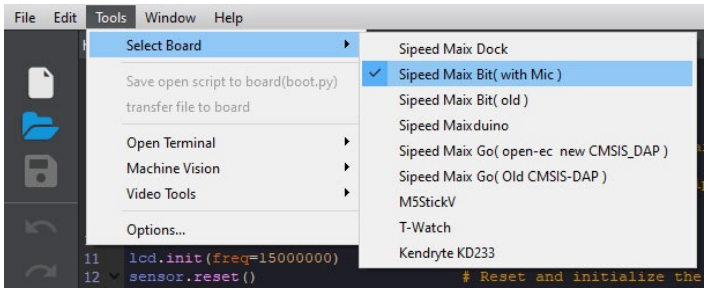


그림 27: MaixPy IDE 보드 선택

4.7.5 직렬 포트 선택

화면 왼쪽 아래에서 녹색 아이콘  을 클릭하면 *Connect – MaixPy IDE(연결 - MaixPy IDE)* 메뉴가 표시됩니다. 선택할 수 있는 직렬 포트가 두 개 이상 있습니다.

올바른 직렬 포트를 선택하면 몇 초 이내에 연결되고 왼쪽 아래 녹색 아이콘이 빨간색으로 바뀝니다.

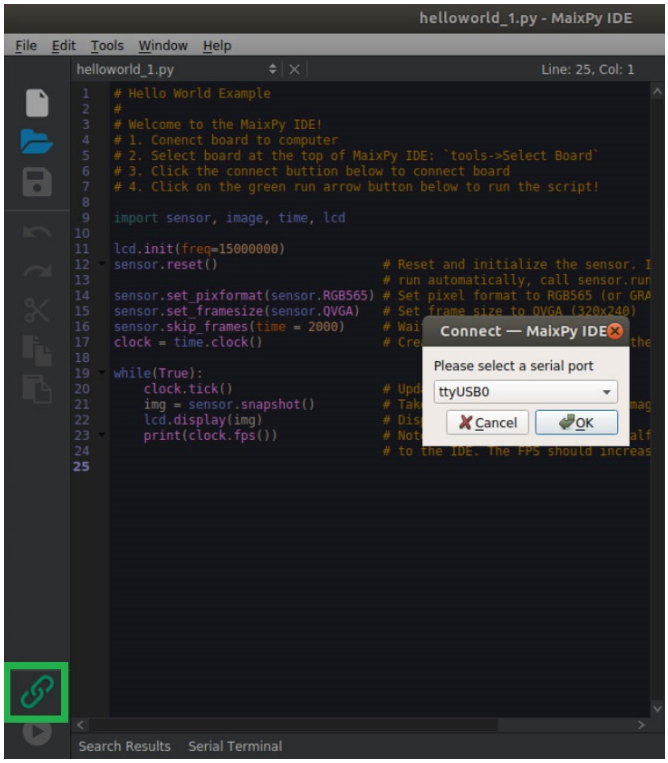


그림 28: MaixPy 직렬 연결

4.7.6 처음 표시되는 Linux 팝업

Linux USB 포트를 MaixPy IDE에 사용할 수 없는 경우 이와 유사한 미리 알림이 표시되며, 'richard'가 아닌 실제 컴퓨터 사용자 이름이 표시됩니다. 지침을 따릅니다.

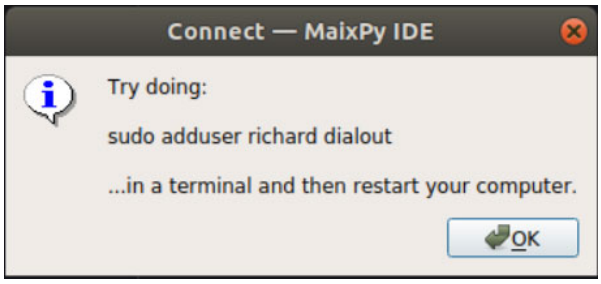


그림 29: Dialout 그룹

4.7.7 프로그램 실행

화면 왼쪽 아래에서 녹색 화살표  을 클릭하여 프로그램을 실행합니다.

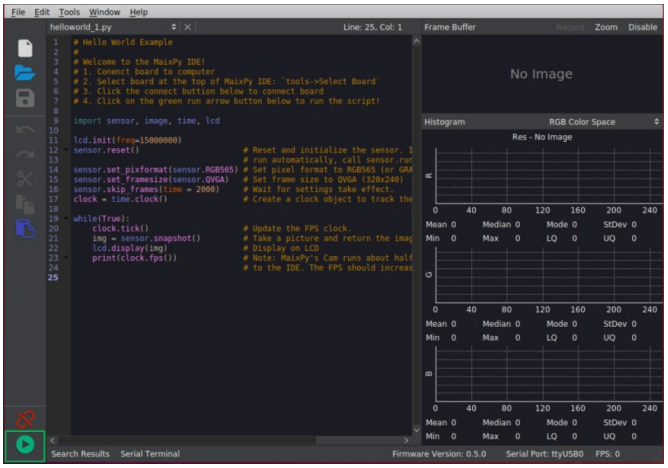


그림 30: 프로그램 설정 실행 준비됨

4.7.8 Hello World 프로그램 실행

`helloword_1.py` 프로그램이 지금 실행 중입니다. 카메라의 사진이 RGB 스펙트럼과 함께 컴퓨터에 표시됩니다. 이 사진은 LCD 화면에도 표시됩니다.

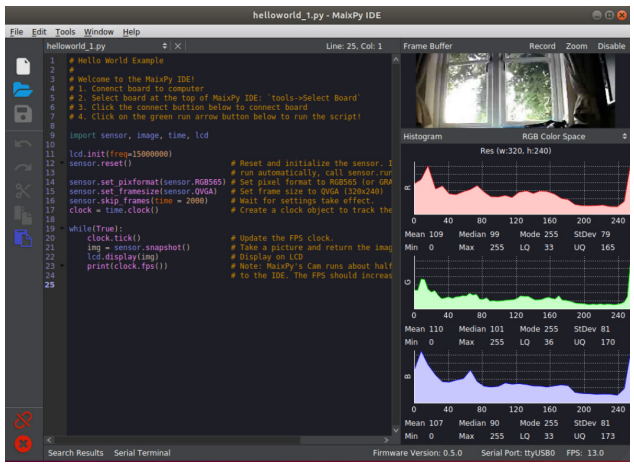


그림 31: Linux에서 실행 중인 Hello World 애플리케이션

프로그램을 중지하려면 *Stop(중지)*  아이콘을 클릭합니다.

4.8 추가 MaixPy 프로젝트

추가 프로젝트를 사용하기 위하여 다운로드를 해야 합니다. MaixPy IDE에서 *File(파일)*, *More examples(추가 예제)*, *Examples on github repo(github repo의 예제)*를 차례로 클릭하면 Github로 이동합니다. 이 내용은 Linux 및 Windows 버전에 모두 적용됩니다.

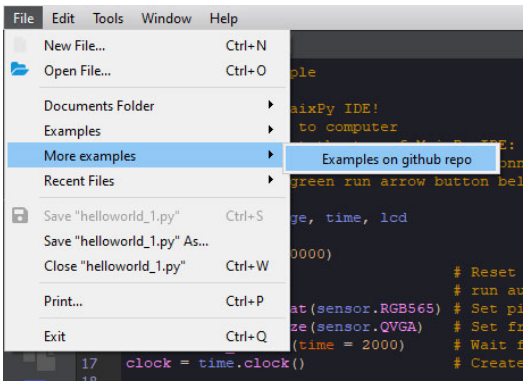


그림 32: 추가 MaixPy 예제

4.8.1 MaixPy 예제 다운로드

`MaixPy_scripts-master.zip`을 다운로드하여 저장한 다음 파일의 압축을 풉니다.

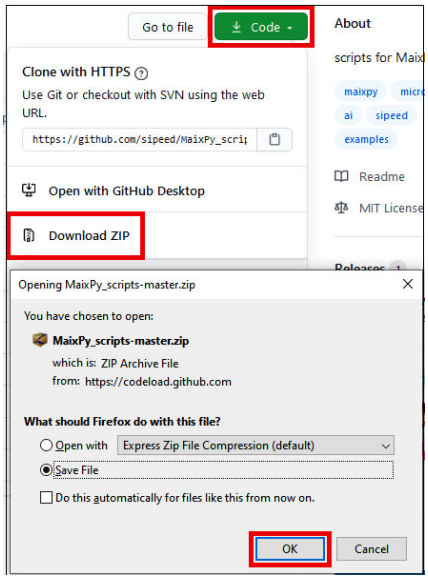


그림 33: MaixPy 예제

4.9 얼굴 인식 프로젝트

소요 시간: 약 15분. 이 단계를 진행하려면 프로그램 2개를 더 다운로드하여 설치해야 합니다.

MaixPy에 대한 추가 예제에서는 `machine_vision` 디렉터리에서 얼굴 인식 프로그램인 `demo_find_face.py`를 제공합니다. 이 프로그램을 MaixPy IDE로 로드합니다.

4.9.1 추가 파일 필요

이 프로젝트를 실행하려면 그 전에 `face_model_at_0x300000.kfpkg`라는 추가 모델을 MaixPy BiT 보드에 로드해야 합니다. 이렇게 하려면 kflash_gui 도구가 필요합니다.

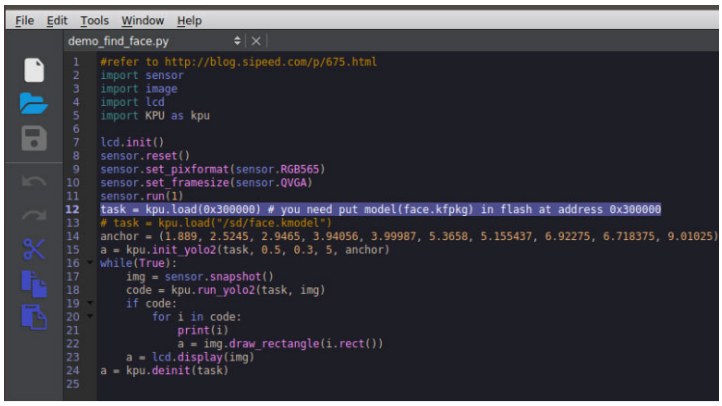


그림 34: 얼굴 인식 프로그램 추가 파일

참고: kflash_gui의 이전 버전에서 기본 다운로드 위치는 0x00000이었습니다. 즉, 정확한 주소 0x300000 대신 `face_model_at_0x300000.kfpkg`를 다운로드하여 0x00000 주소 지정을 수행할 수 있었습니다. 이렇게 하면 MicroPython 인터프리터를 덮어썼고 보드가 실행될 수 없었습니다. 따라서 보드를 다시 프로그래밍해야 했습니다.

4.10 Maix BiT Flash 업데이트

소요 시간: 약 10분

kflash_gui 도구는 Windows와 Linux에서 모두 실행됩니다. 최신 소프트웨어로 보드를 플래시하는 것이 빠르고 쉽습니다.

4.10.1 kflash_gui 다운로드

kflash_gui는 https://github.com/sipeed/kflash_gui/releases 에서 다운로드할 수 있습니다.

여기서 `kflash_gui_v1.5.3_windows.7z`를 사용했습니다. 최신 버전은 Avast 바이러스 검사기와 관련해 문제가 발생합니다.

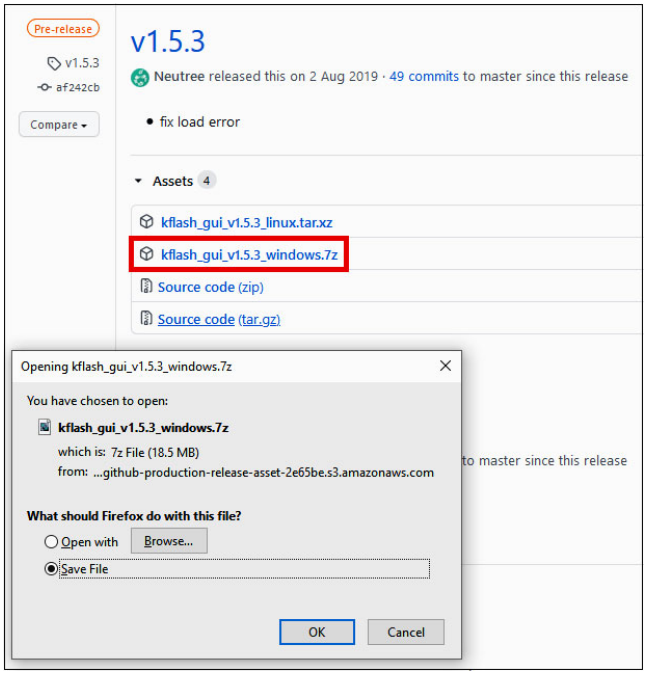


그림 35: kflash GUI 다운로드

`.7z` 파일의 압축을 풉니다. 컴퓨터에서 `.7z` 파일을 지원하지 않으면 다음 위치에서 Express Zip 등 무료 온라인 도구를 다운로드할 수 있습니다.

https://www.nchsoftware.com/software/utilities.html.

프로그램을 실행하려면 추출한 파일로 이동하여 kflash_gui 아이콘을 두 번 클릭합니다.

참고: kflash_gui 설치에서는 바탕 화면에 바로 가기를 표시하지 않습니다. 필요한 경우 수동으로 수행해야 합니다.

4.10.2 얼굴 모델 다운로드

https://github.com/sipeed/MaixPy/releases에서 `face_model_at_0x300000.kfkpg`를 다운로드하고 파일을 저장합니다.

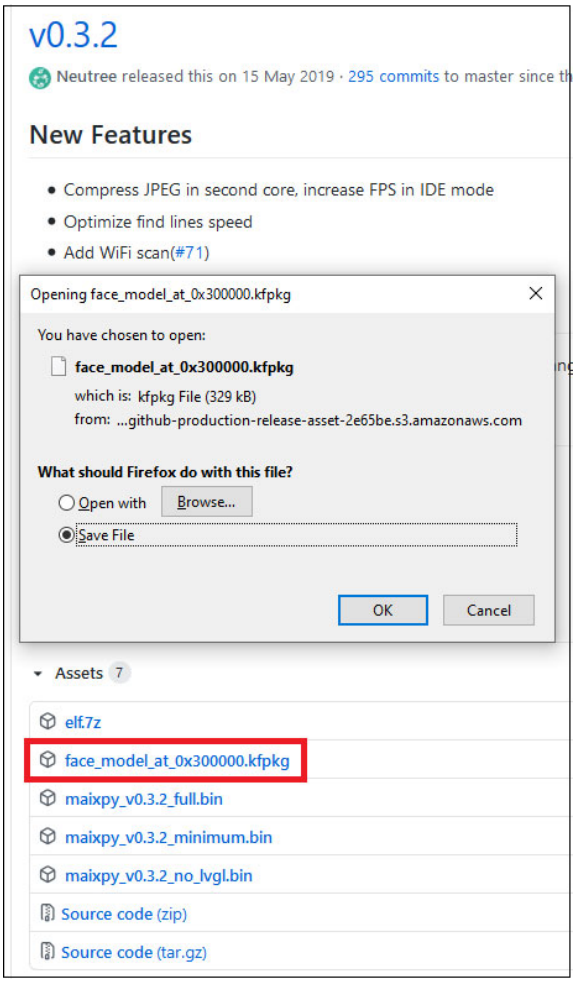


그림 36: 얼굴 모델 다운로드

4.10.3 kflash GUI를 사용하여 얼굴 모델 프로그래밍

kflash_gui 아이콘을 클릭합니다. *Open File(파일 열기)*을 클릭하고 `face_model_at_0x300000.kfpkg`를 선택합니다. *Sipeed Maix BiT (with Mic)(Sipeed Maix BiT (마이크 포함)) 보드*, 직렬 *Port(포트)*를 선택한 다음 *Download(다운로드)*를 클릭합니다.

다운로드에 약 20초 가량 걸립니다.

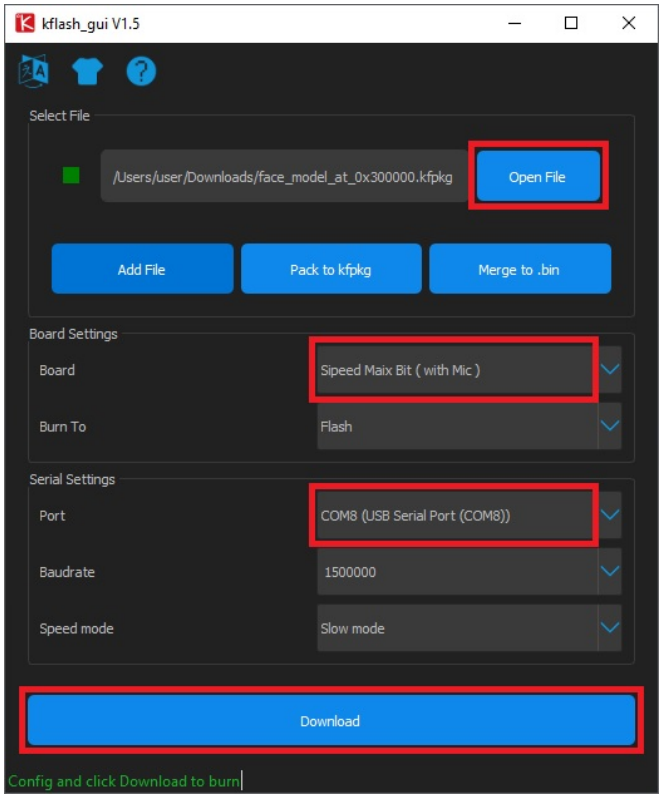


그림 37: kflash GUI

4.11 얼굴 인식 프로그램 실행

MaixPy IDE로 돌아가 `machine_vision` 디렉터리에서 `demo_find_face.py`를 실행합니다.

얼굴이 인식되면 LCD와 컴퓨터 화면에 직사각형이 그려집니다.

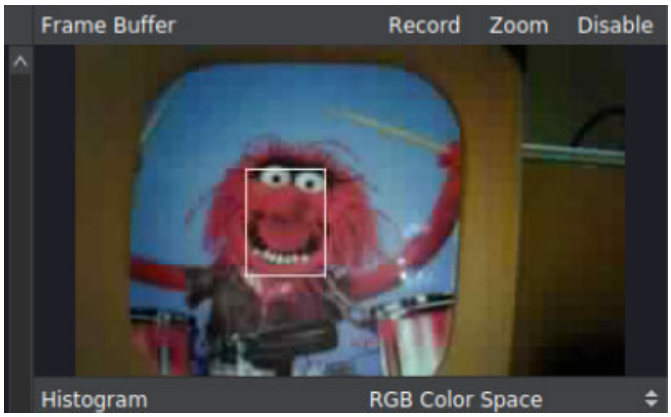


그림 38: 얼굴 인식 예제

4.12 Maix BiT 모드 메인 소프트웨어 다시 플래시

kflash_gui는 Maix BiT 보드에서 MicroPython 인터프리터를 다시 프로그래밍하는 데 사용됩니다. 펌웨어 업데이트가 빈번하지만 일부는 MaixPy IDE와 호환되지 않습니다.

4.12.1 최신 소프트웨어 다운로드

다음으로 이동:

https://dl.sipeed.com/MAIX/MaixPy/release/master. 여기에는 MaixPy의 버전 목록이 포함되어 있습니다. 최신 버전을 클릭합니다.

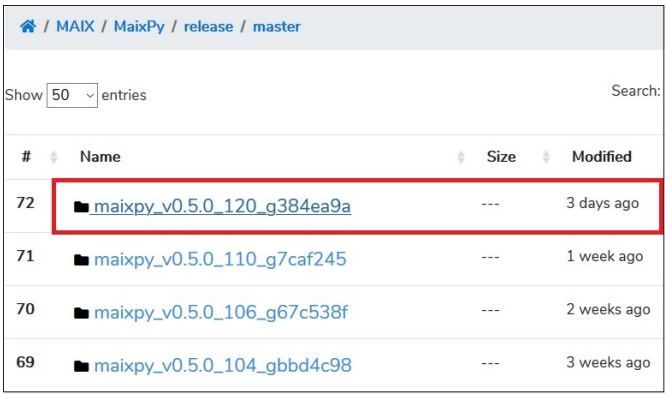


그림 39: MaixPy 릴리스 마스터 인덱스

4.12.2 최신 MaixPy 선택

`_minimum_with_ide_support.bin`으로 끝나는 파일을 클릭하고 저장합니다.

IDE에서 해당 파일을 지원해야 합니다.

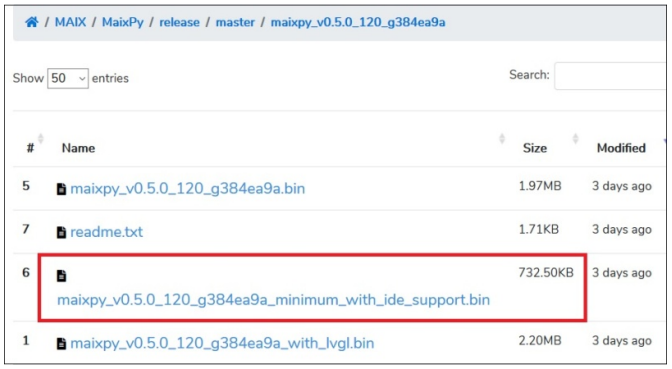


그림 40: Maix BiT 메인 소프트웨어 다운로드

4.12.3 Maix BiT 보드 프로그래밍

kflash_gui를 엽니다. *Open File(파일 열기)*을 클릭하고 다음을 선택합니다.

```
maixpy_v0.5.0_120_g3943a9a_minimum_with_ide_support.bin
```

Board(보드), 직렬 *Port(포트)*를 선택한 다음 *Download (다운로드)*를 클릭합니다.

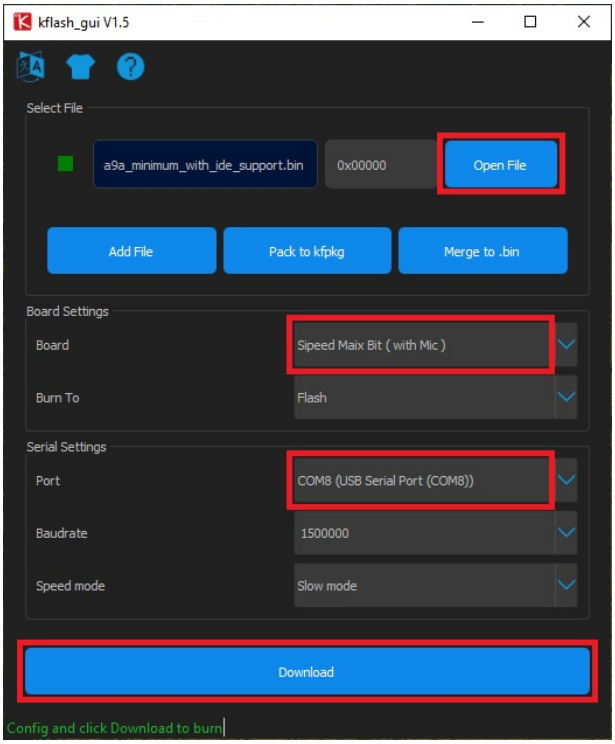


그림 41: Maix BiT flash 다시 프로그래밍

4.13 직렬 터미널에서 MaixPy 실행

소요 시간: 약 5분

MaixPy IDE를 사용하는 것 외에 프로그램을 직렬 터미널에서도 실행할 수 있습니다. 이 단계는 빠르고 쉽게 진행할 수 있으며 Maix BiT 보드의 구성과 소프트웨어 버전을 확인하는 데 유용합니다. 또한 MicroPython을 학습하기 위한 도구로 사용할 수 있습니다.

4.13.1 직렬 터미널 설정

아직 설치하지 않은 경우 무료 직렬 터미널 프로그램(예: Putty 또는 TeraTerm)을 다운로드하여 설치합니다.

직렬 포트 속도를 115200 보드로 설정합니다. 다른 컴퓨터에서는 포트 COM8이 다를 수 있습니다.

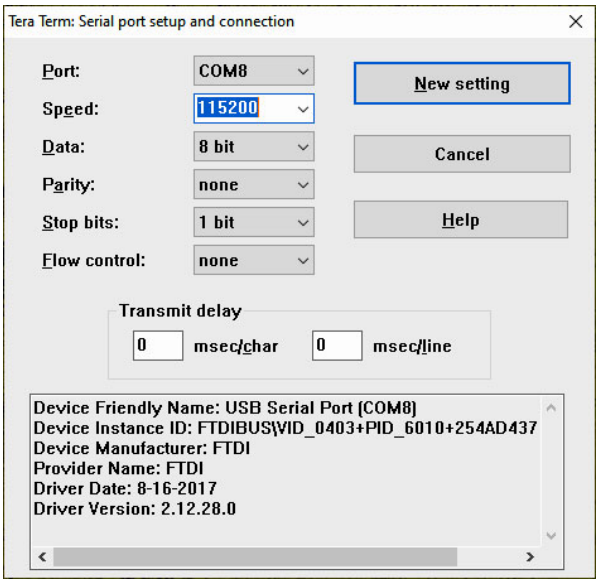


그림 42: 직렬 터미널 구성

4.13.2 리셋 버튼 눌렀다가 놓기

MaixPy BiT 보드에서 *리셋 버튼*을 눌렀다가 놓습니다.

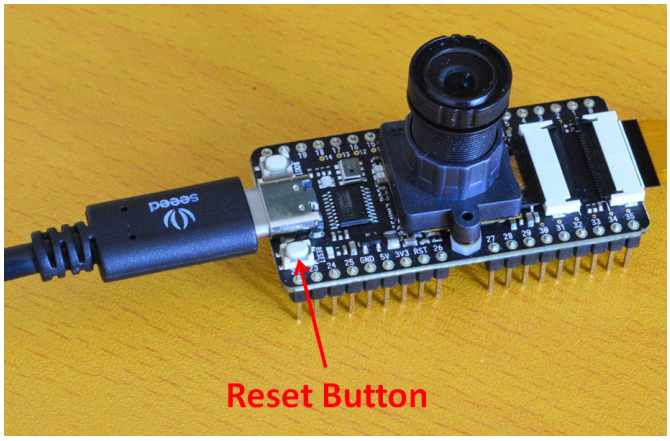


그림 43: Maix BiT 보드 리셋 버튼

4.13.3 직렬 터미널에서 MaixPy 실행

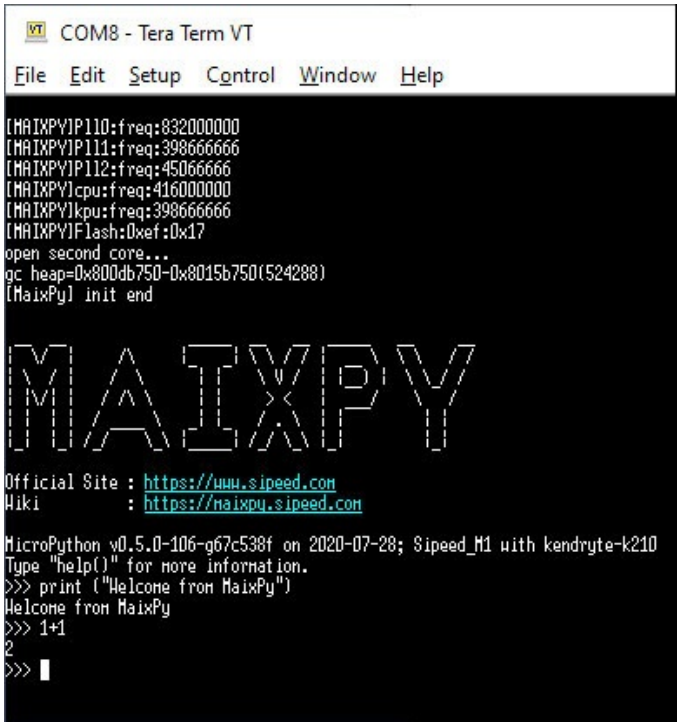


그림 44: 직렬 터미널의 MaixPy

`print ("Welcome from MaixPy")`를 입력하면 "Welcome from MaixPy"라고 응답합니다.

`1+1`을 입력하면 `2`라고 응답합니다.

5 실용적인 정보: SparkFun RED-V 보드에서 SiFive SoC 사용

FreedomStudio IDE를 사용하여 C로 프로그래밍하도록 설계된 보드 2개가 있는 미드레인지 옵션입니다.

5.1 SparkFun 동영상

Digi-Key 웹사이트에는 보드 사용 방법을 보여주는 Shawn Hymel의 유용하고 짧은 동영상이 있습니다. 보드 구입, 소프트웨어 설치 전 또는 나중에 소프트웨어 다운로드를 기다리면서 해당 동영상을 보십시오. 유용할 것입니다.

그림 45: RED-V로 시작하기. 출처: Digi-Key 웹사이트.

동영상 링크는 Digi-Key 웹사이트의 Red Board와 RED-V Thing Plus 페이지에 있습니다.

5.2 하드웨어

SiFive Freedom Everywhere RISC-V SoC에 따라 SparkFun Electronics에서 보드 2개를 구입할 수 있습니다. 다음 보드는 모두 SiFive1 Rev B 보드 및 소프트웨어와 호환됩니다.

소형 SparkFun RED-V SIFIVE RISC-V THING PLUS - 1568-DEV-15799-ND $29.95

대형 SparkFun Electronics FE310 Red Board - 1568-DEV-15594-ND $39.95

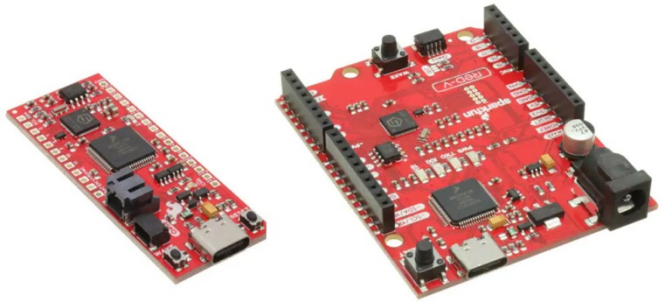


그림 46: SparkFun RED-V Thing Plus(왼쪽) 및 Red Board(오른쪽). 이미지 출처: SparkFun Electronics

참고: Thing Plus와 Redboard 모두 USB-C to USB-A 케이블이 함께 제공되지 않으므로 하나를 구입해야 합니다. 예:

Seeed Technology USB-C 케이블 Digi-Key 1597-106990248-ND $2.42

소프트웨어는 Windows용 및 Linux용 SiFive에서 모두 다운로드할 수 있습니다. 안타깝게도 본 문서의 작성자가 Linux 버전을 실행할 수 없었기 때문에 여기서는 Windows 버전에 대해서만 설명합니다.

5.3 SiFive Freedom Studio 소프트웨어 설치 창

설치에는 약 20분 가량 걸리며 다운로드한 파일 크기는 1.3GB입니다.

www.SiFive.com/software로 이동하여 Freedom Studio 섹션까지 페이지 아래로 스크롤합니다.

Windows 를 클릭한 다음 .zip 파일을 저장합니다.

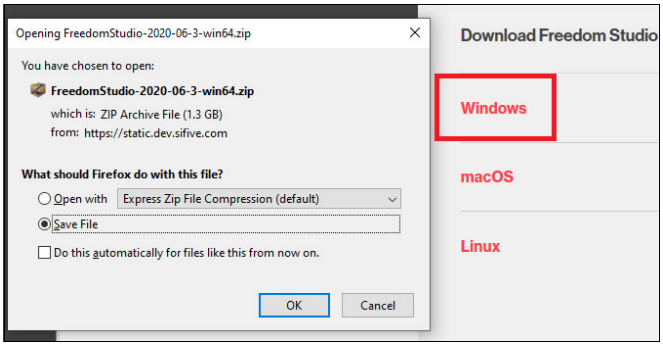


그림 47: SiFive Freedom Studio 창

5.3.1 Freedom Studio 파일 위치

파일의 압축을 풉니다. 설치 디렉터리로 이동합니다. 예:

```
C:\Users\user\FreedomStudio-2020-06-3-win64\
```

FreedomStudio 을 두 번 클릭하여 실행합니다.

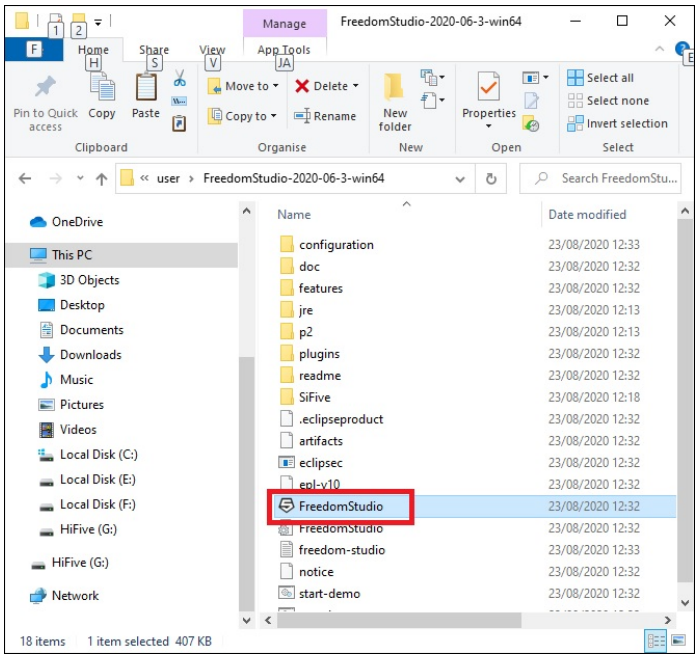


그림 48: FreedomStudio 파일 위치

5.3.2 Freedom Studio 시작 화면

Freedom Studio 아이콘이 표시되어야 합니다. 필요한 경우 바탕 화면에 바로 가기를 추가할 수 있습니다.

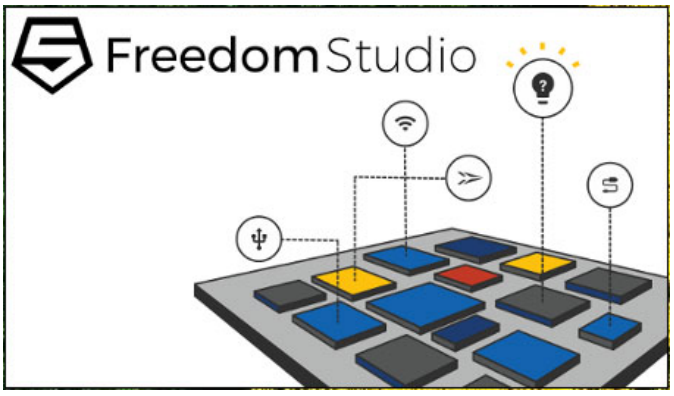


그림 49: Freedom Studio 시작 화면

5.4 새 소프트웨어 프로젝트 만들기

아직 실행 중이 아닌 경우 Freedom Studio를 시작합니다.

5.4.1 보드 꽂기

보드를 꽂으면 디버그 구성이 자동으로 설정되기 때문에 보드를 꽂은 후 프로그램을 만드는 것이 좋습니다. USB-C to USB-A 케이블을 사용합니다. SparkFun 보드 중 하나를 사용할 수 있습니다.

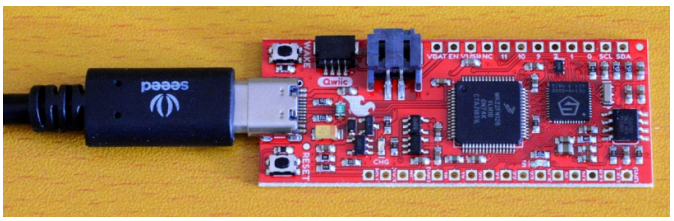


그림 50: USB-C 케이블이 연결된 Thing Plus 보드

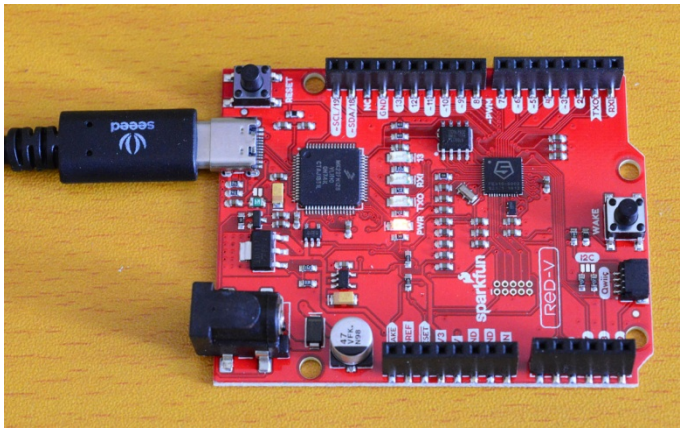


그림 51: USB-C 케이블이 연결된 Red Board

5.4.2 새 Freedom E SDK 소프트웨어 프로젝트 만들기

화면 맨 위에서 *SiFive Tools(SiFive 도구)* 탭으로 이동하고 *Create a new Freedom E SDK Software Project(새 Freedom E SDK 소프트웨어 프로젝트 만들기)*를 클릭합니다.

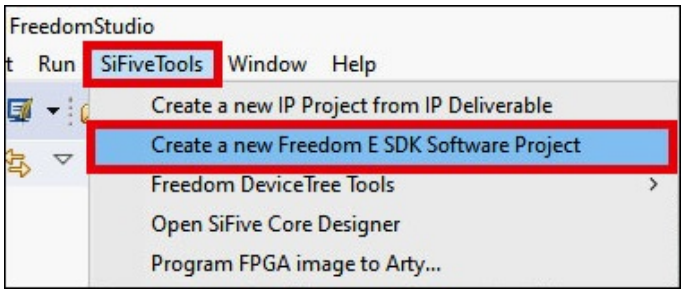


그림 52: 새 Freedom E SDK 소프트웨어 프로젝트 만들기

5.4.3 보드 선택

여러 가지 보드가 지원됩니다. 풀다운 메뉴에서 *sifive-hifive1-revb*를 선택합니다. 이 보드는 Red Board 및 Thing Plus 보드와 모두 호환됩니다.

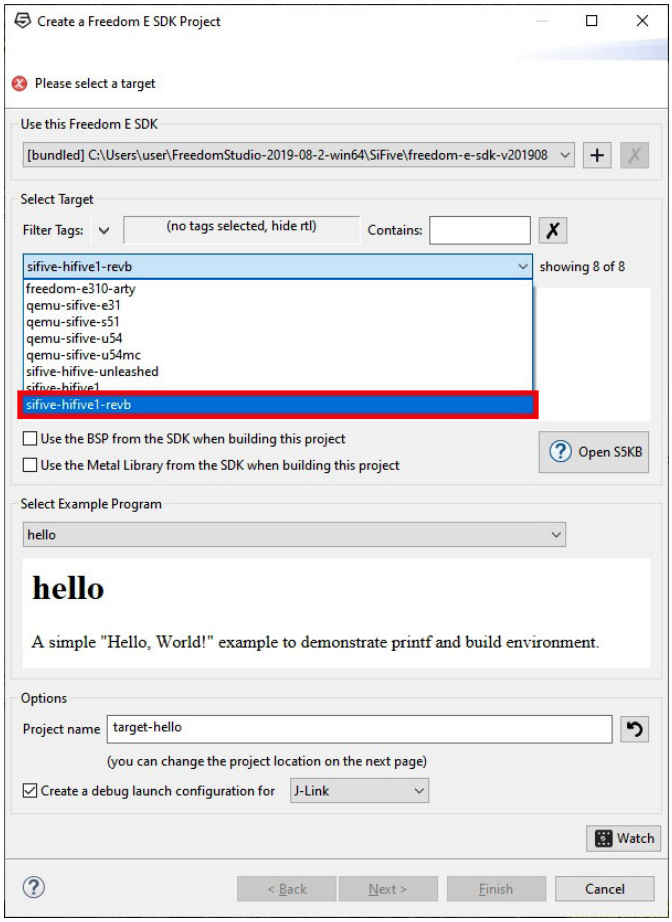


그림 53: SiFive 보드 선택

5.4.4 예제 프로그램 선택

Select Example Program(예제 프로그램 선택) 풀 다운 메뉴에서 다양한 예제 프로젝트를 사용할 수 있습니다.

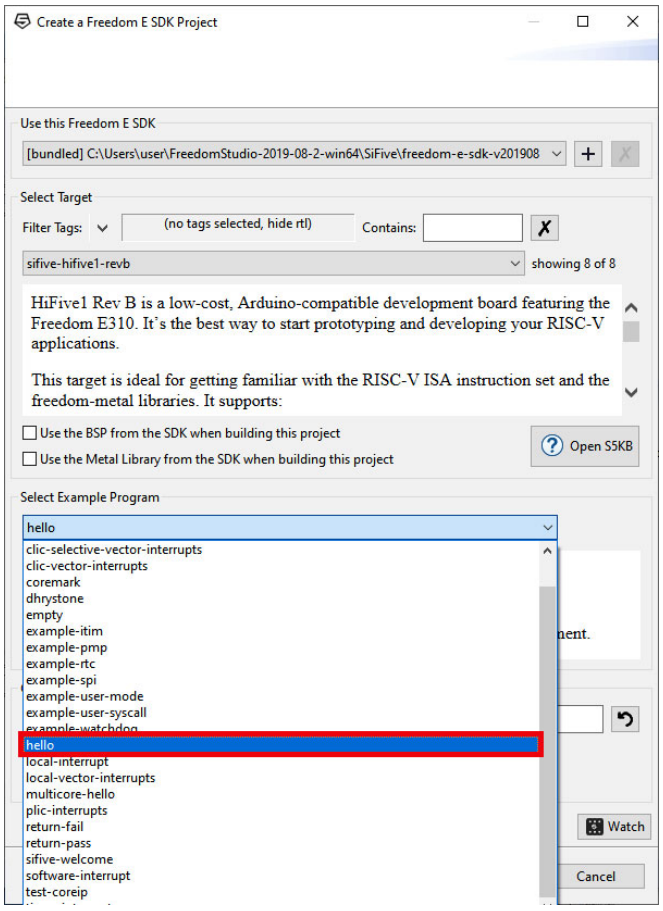


그림 54: SiFive 샘플 프로젝트

이러한 예제 프로젝트는 다른 프로젝트를 만드는 데 아주 훌륭한 시작점입니다. 지금은 *hello* 프로젝트를 계속해서 사용합니다.

5.4.5 디버그 실행 구성

창 맨 아래에 *Create a debug launch configuration for J-Link(J-Link에 대한 디버그 실행 구성 만들기)*가 자동으로 설정되었습니다. 보드를 꽂았기 때문에 자동으로 설정된 것입니다. *Finish(마침)*를 클릭합니다. 프로젝트가 자동으로 빌드 됩니다.

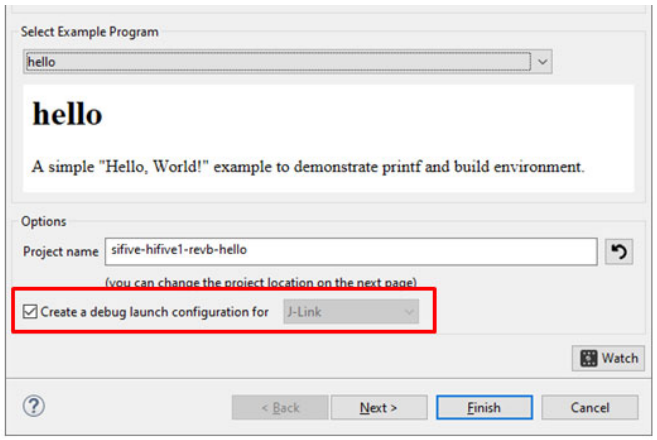


그림 55: 예제 프로그램과 디버그 실행 구성 선택

5.4.6 .elf 파일 확인

Edit Configuration(구성 편집) 창이 자동으로 표시됩니다. **E**xecutable and **L**inkable **F**ile인 `hello.elf`가 생성되었는지 확인합니다. 이 파일은 보드를 프로그래밍하는 데 필요합니다.

*Debug(디버그)*를 클릭하여 바로 디버그로 이동할 수 있지만 지금은 설정할 항목이 몇 가지 더 있으므로 *Close(닫기)*를 클릭합니다.

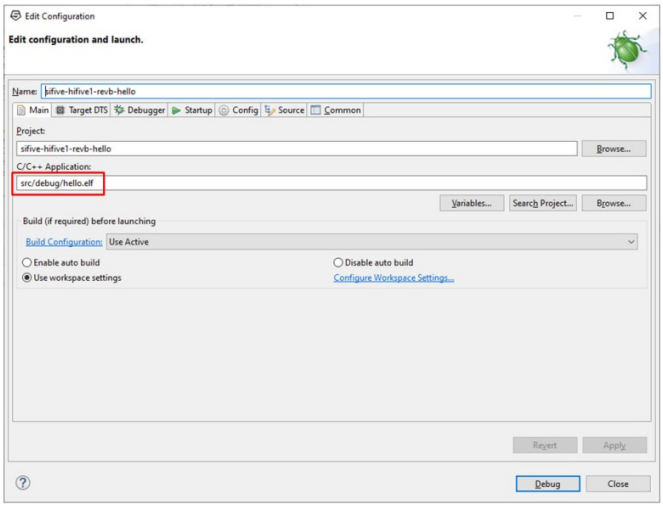


그림 56: 구성 편집 메뉴

5.4.7 콘솔과 소스 코드 보기

콘솔 창에는 사용된 컴파일러, 코드 크기, 빌드 시간 등 빌드에 대한 정보가 나와 있습니다.

첫 번째 빌드는 프로젝트 내 파일을 전부 컴파일해야 하기 때문에 시간이 걸립니다. 이후 빌드에서는 실제로 변경된 파일만 다시 컴파일하기 때문에 시간이 훨씬 더 단축됩니다.

`hello.c` 창에 C 코드가 표시됩니다.

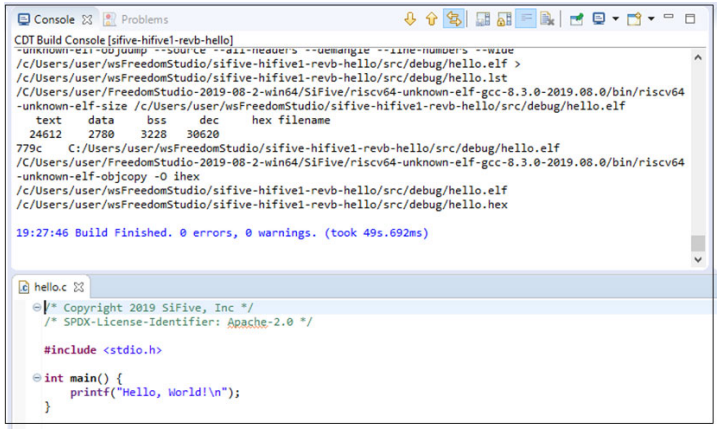


그림 57: 콘솔 및 소스 창

5.4.8 *hello.c* 소스 코드 수정

`hello.c` 프로그램을 현재 상태로 디버그하면 종료 작업만 수행하기 때문에 출력되는 내용이 별로 없을 것입니다. 따라서 그림 58에 표시된 것처럼 `hello.c`를 수정하고 `Hello, World!`를 10회 출력합니다.

그림 58: 수정된 hello.c 파일

구문을 확인하기 위해 도구 모음에서 ***Build(빌드)*** 아이콘  을 클릭하여 프로그램을 빌드할 수 있지만 ***Debug(디버그)***가 실행되면 이 작업이 자동으로 수행됩니다.

5.4.9 프로그램 디버그

프로그램 디버그를 시작하려면 도구 모음에서 ***Debug (디버그)*** 아이콘  을 클릭하거나 키보드에서 **F11** 키 또는 도구 모음에서 ***Run(실행)***을 누른 다음 ***Debug (디버그)***를 누릅니다.

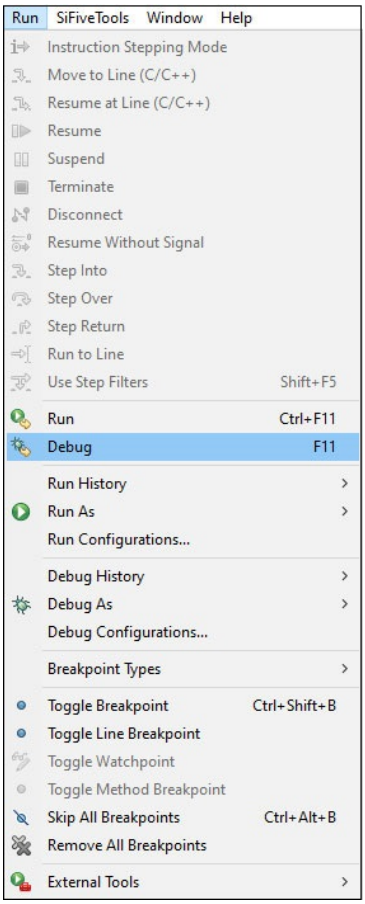


그림 59: 디버그 시작

5.4.10 *main()*에서 디버거 중지

`main()`의 시작 부분에서 디버거가 멈춥니다.

그림 60: *main()*에서 디버거 중지

5.4.11 printf에 대해 직렬 터미널 설정

Terminal(터미널) Terminal 탭을 클릭하고 회면 오른쪽에서 *Open a Terminal(터미널 열기)* 아이콘을 클릭합니다.

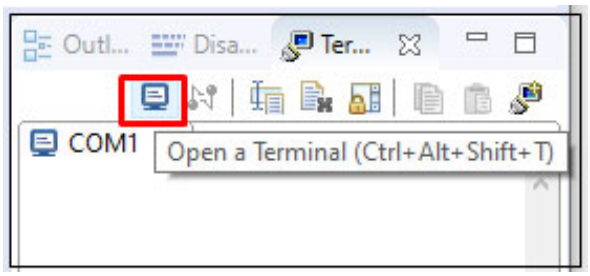


그림 61: 터미널 열기 아이콘

5.4.12 터미널 실행

직렬 터미널을 115200보드에서 설정합니다. *Serial port (직렬 포트)*가 두 개 이상 있기 때문에 여러 번 시도해야 할 수 있습니다.

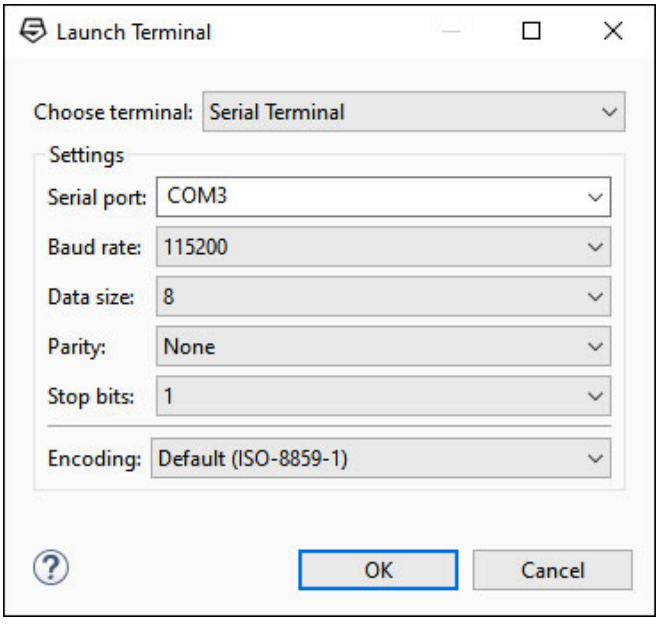


그림 62: 직렬 터미널 설정

5.4.13 디버그 출력

코드를 통해 설정하려면 도구 모음에서 *Step Over(단위 실행)*  아이콘을 클릭하거나 키보드에서 F6 기능 키를 여러 번 눌러 Hello, World! 가 10회 출력되도록 합니다.

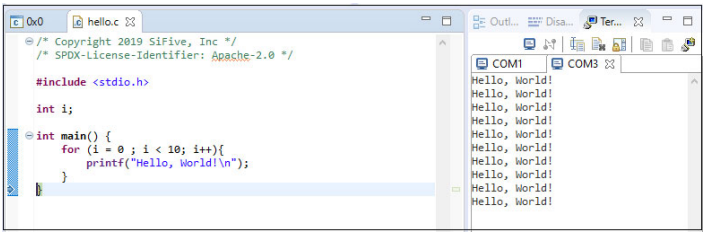


그림 63: COM3에서 디버그 출력

`printf` 메시지가 동일한 프로그램의 창에 표시된다는 사실은 디버그 및 코드 검사에 매우 유용합니다. 별도 터미널에 연결하는 것보다 훨씬 쉽기 때문입니다.

5.4.14 디스어셈블리 창

그림 64는 ARM®과의 비교를 위해 `hello.c` 프로그램의 약간 수정된 버전을 보여줍니다.

C 코드를 구성하는 어셈블리 언어 명령을 보려면 *Disassembly(디스어셈블리)* 탭 

을 클릭합니다.

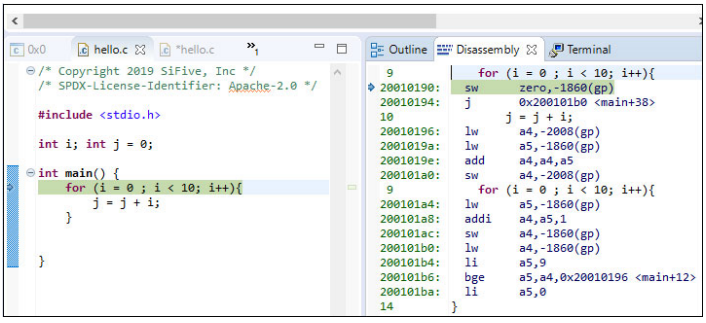


그림 64: 디스어셈블리 창

5.4.15 ARM Cortex M0와 비교

그림 65에서는 상응하는 ARM® Cortex® M0+ 디스어셈블리 코드를 보여줍니다.

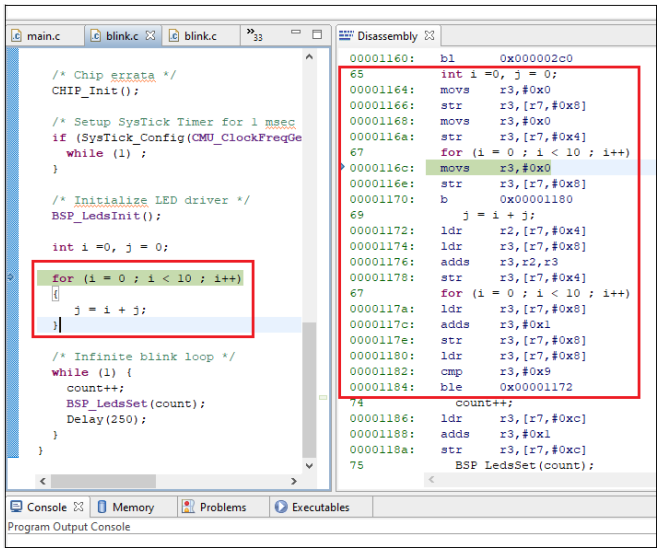


그림 65: ARM Cortex M0와 비교

이 그림을 보면 이 경우 RISC-V에는 ARM보다 명령어가 더 적게 필요합니다.

5.4.16 프로그램 중지

프로그램을 중지하고 처음부터 다시 실행하려면 *Toolbar(도구 모음)* 또는 *Console(콘솔)*에서 *Terminate (종료)* 아이콘을 클릭합니다.

프로그램 디버그를 다시 시작하려면 도구 모음에서 *Debug(디버그)* 아이콘  을 클릭하거나 키보드에서 F11 키를 누릅니다.

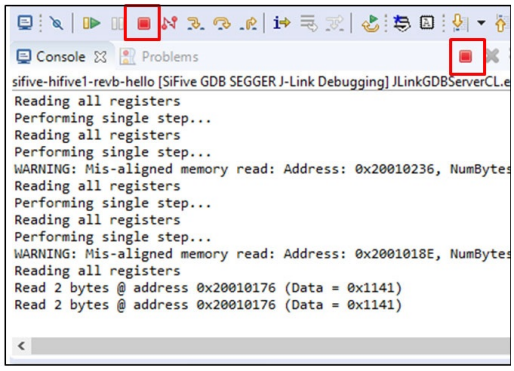


그림 66: 프로그램 종료

5.4.17 기존 프로젝트를 닫고 다시 열기

기존 Freedom Studio 프로젝트를 닫고 다시 열 때 약간 번거로울 수 있습니다. Freedom Studio를 닫은 다음 다시 엽니다. 프로젝트를 클릭한 다음 ***Debug(디버그)***  아이콘을 클릭합니다. 프로그램이 디버그하지 않고 아래와 같이 오류 메시지가 표시됩니다.

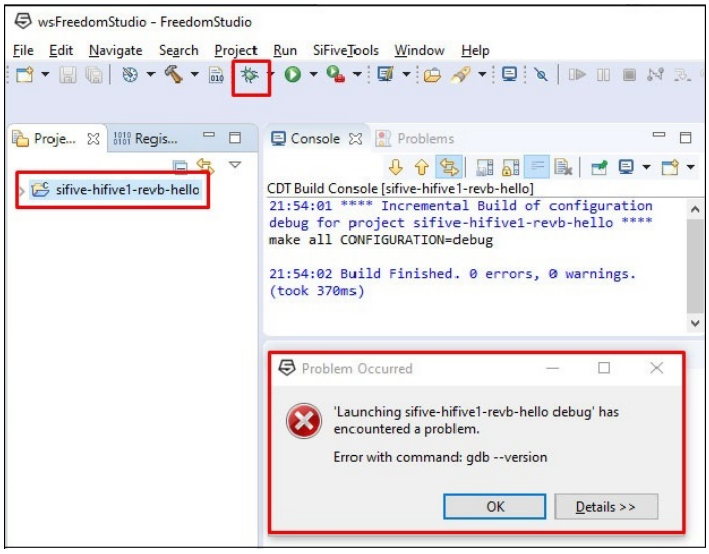


그림 67: 프로젝트를 다시 여는 동안 발생하는 문제

5.4.18 디버그 구성 선택

화면 상단의 ***Toolbar(도구 모음)***에서 ***Run(실행)***을 선택한 다음 ***Debug Configurations(디버그 구성)***를 선택합니다.

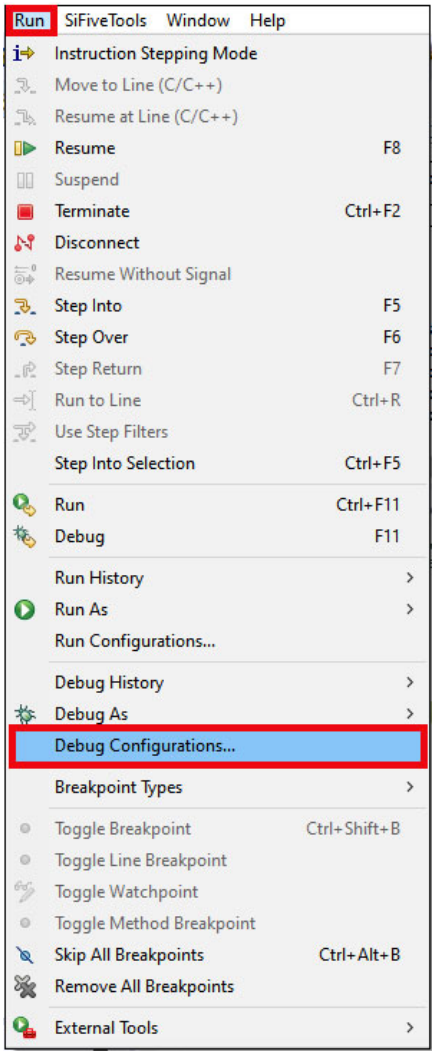


그림 68: 디버그 구성 선택

5.4.19 구성 만들기, 관리, 실행

Sifive GDB SEGGER J-Link Debugging 탭을 확장하고 디버그할 프로젝트(여기서는 `sifive-hifive-1-revb-hello`)를 클릭합니다. ***Debug(디버그)***를 클릭합니다. 이제 프로그램을 디버그할 수 있어야 합니다.

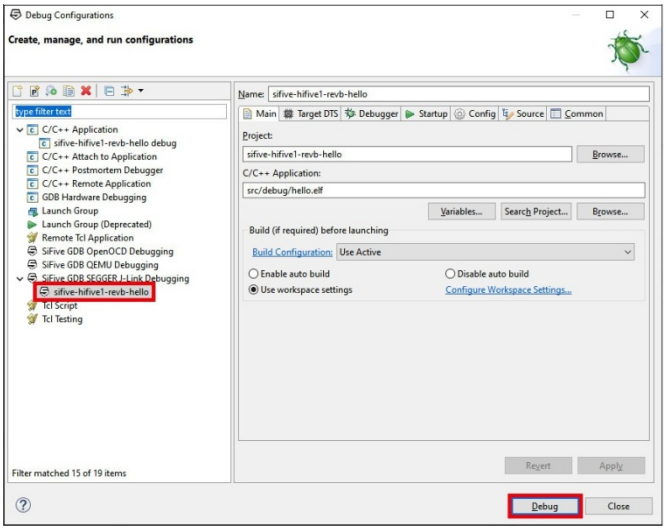


그림 69: 디버그할 프로젝트 선택

6 실용적인 정보: Digilent Nexys A7을 사용하여 소프트 코어 구현

이는 가이드에서 기술적으로 가장 까다로운 부분이며, RISC-V 코어를 FPGA에 내장해야 합니다. 컴퓨터 아키텍처에 대해 자세한 연구에는 이상적이지만 일부 독자에게는 너무 복잡할 수 있습니다.

6.1 Imagination Technologies University Programme의 지원

FPGA에서 소프트 코어를 구현하는 것이 주요 작업입니다. 다행히, Imagination Technologies University Programme에서 Xilinx Artix-7 FPGA에서 RISC-V 코어를 구현하기 위한 학습 자료인 "RVfpga: The Complete Course in Understanding Computer Architecture"(RVfpga: 컴퓨터 아키텍처 이해를 위한 완벽한 과정)를 전 세계에 배포하고 있습니다. 이 자료는 솔루션과 함께 20개 연구실의 자료, 문서, 예제 및 연습으로 구성된 전체 패키지로 이루어져 있습니다. 필요한 소프트웨어는 모두 무료로 다운로드할 수 있습니다. 덕분에 사용자는 엄청나게 많은 작업과 복잡한 일을 건너뛸 수 있습니다. 여기서 제시하는 자료는 전체 과정의 10% 정도입니다.

RVfpga 자료에 액세스하려면 먼저 아래 링크에서 등록해야 합니다.

https://university.imgtec.com/forums/?wpforo=signup

안타깝게도 등록 양식은 대학교를 위해 디자인되어 약간 깁니다. 하지만 RVfpga 자료는 대학교 외부 사람들도 사용할 수 있으므로 대학교 관련 필드의 경우 영리 단체 또는 기타 조직의 상응하는 답변을 입력하면 됩니다. 모든 필드를 다 작성할 필요는 없으므로 등록에 걸리는 시간은 10분 미만입니다. 전화번호에 공백을 입력하지 마십시오.

등록하고 로그인한 후 아래 Teaching Resources(학습 리소스) 페이지에서 자료 다운로드를 요청하십시오.

https://university.imgtec.com/teaching-download/

다운로드 자료는 대학교 학부 3학기 동안 배우는 내용에 해당하며 20개 연구소의 자료를 담고 있습니다. 모두 무료로 제공됩니다. 또한 문제에 대해 논의하고 질문에 대한 답변을 얻을 수 있는 포럼도 있습니다.

석사 수준 "Creating a SoC"(SoC 만들기) 과정은 2021년 1분기에 제공될 예정입니다.

6.2 소프트 코어 옵션

비용에 너무 민감한 프로젝트가 아닌 경우 표준 프로세서를 사용하는 대신 FPGA에 소프트 코어를 사용하면 설계자가 훨씬 더 자유롭게 작업할 수 있습니다. www.opencores.org에서 다운로드할 수 있는 다양한 기성 주변 장치가 있습니다. Fast Fourier Transform(FFT), 디지털 필터 및 복잡한 암호화 알고리즘 등과 같은 알고리즘은 소프트웨어가 아니라 하드웨어에서 구현할 수 있어 처리 속도를 더 높입니다. 주변 장치를 모든 핀에 가상으로 라우팅할 수 있기 때문에 PCB 레이아웃은 훨씬 쉽습니다.

6.3 프로그램 실행에 필요한 보드

소프트웨어 시뮬레이션 및 소프트웨어 디버깅만 하는 경우에는 보드가 필요 없습니다.

사용되는 보드:

Digilent Nexys A7 Nexys A7 ECE FPGA Trainer Board
Digi-Key 1286-1081-ND $265.00

매우 유사하기 때문에 현재 단종되긴 했지만 이전 보드인 Nexys4 DDR 보드로 사용할 수 있습니다.

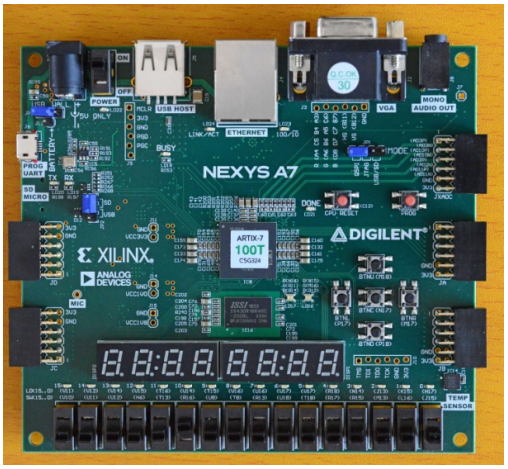


그림 70: Digilent Nexys A7 보드.

6.4 필요한 소프트웨어

다음 프로그램이 필요합니다.

프로그램	용도
Xilinx Vivado 2019.2	RISC-V 코어를 구성하는 Verilog `.v`와 System Verilog `.sv` 파일을 빌드합니다. 생성된 `.bit` 파일을 하드웨어에 다운로드합니다.
Visual Studio Code(VSCode)	C 및 어셈블리 언어 소프트웨어 개발용 IDE
PlatformIO	RISC-V 프로세서를 위한 지원을 제공하는 VSCode의 추가 기능
Open OCD	오픈 소스 **O**n **C**hip **D**ebugger
RISC-V 툴체인	RISC-V 프로젝트 빌드에 필요한 컴파일러 및 기타 파일
Verilator	Verilog `.v` 및 System Verilog `.sv` 파일용 하드웨어 시뮬레이터 및 프로세서
Whisper	Western Digital의 Instruction Set Simulator(ISS). 하드웨어가 없어도 컴퓨터에서 코드를 실행 및 디버그 가능
Digilent Board Files	Digilent Nexys A7 보드의 구체적인 구성 파일
GTKWave	낮은 수준의 시스템 타이밍을 보기 위한 Waveform 뷰어
RVfpga	소프트웨어 코어, 코드 예제, Verilog/System Verilog 프로젝트
RVfpga Labs	20개 연구소

표 4: 필요한 프로그램

현재 Windows 및 Mac에 대한 지원이 개발 중이지만 모든 프로그램은 Linux에서만 실행됩니다. 전체 다운로드 및 설치 시간은 약 4시간입니다.

6.5 RISC-V 구현 실행

Xilinx FPGA에서 RISC-V에 대한 코드를 실행하려면 다음 두 가지 작업을 수행해야 합니다.

1. `.bit` 파일을 사용하여 FPGA 하드웨어를 구성합니다.
2. 코드를 컴파일하고 `.elf` 파일을 FPGA에 다운로드합니다.

6.5.1 FPGA 하드웨어와 소프트웨어 구성 요소 간의 관계

그림 71은 Xilinx Vivado가 Verilog `.v` 및 System Verilog `.sv` 파일을 처리하여 `.bit` 파일로 변환하는 데 어떻게 사용되는지 보여주며, 이는 FPGA로 프로그래밍할 수 있습니다.

C 또는 어셈블리 코드는 PlatformIO를 사용하여 컴파일되며 `.elf` 파일은 FPGA로 다운로드되어 여기서 실행 및 디버그할 수 있습니다.

Digilent Nexys A7 보드와는 신호 공유 USB 포트를 통해 통신합니다. Vivado와 Visual Studio Code를 동시에 사용하여 보드와 통신하는 것은 불가능합니다.

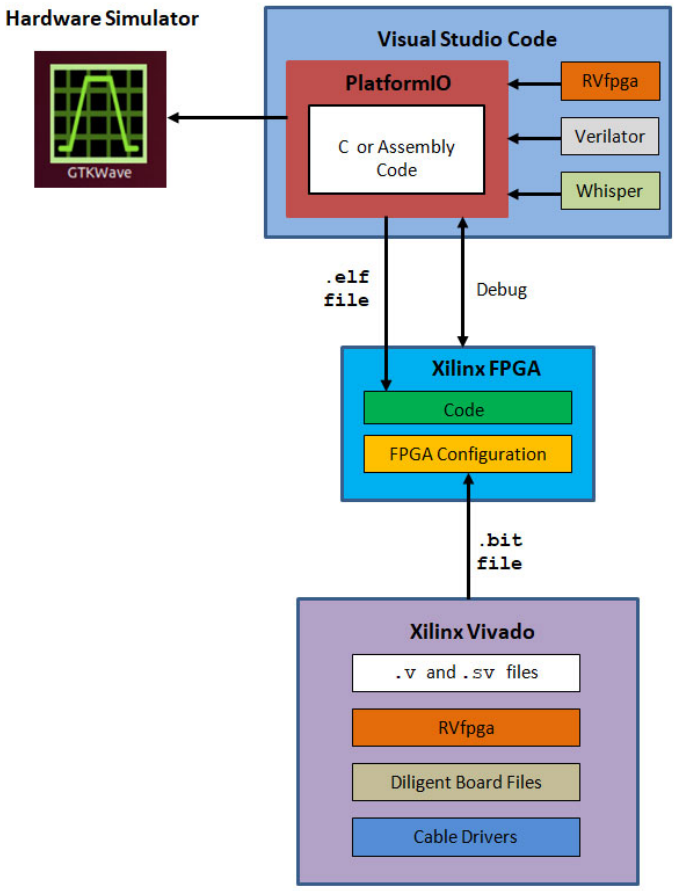


그림 71: Visual Studio, FPGA와 Xilinx Vivado의 관계

6.6 Western Digital SweRV 코어

다양한 RISC-V 코어를 사용할 수 있지만, 이들 중 다수는 완전한 인증을 받기보다는 학술 연구의 일부입니다. 해당 코어가 제대로 작동하지 않는다고 할 수는 없지만 산업용 제품에 사용 시 제조물 책임으로 인해 큰 비용이 발생할 상황을 피하려면 완전히 검증하고 인증해야 합니다.

Western Digital(2019년 매출 165억 7000만 달러)에서는 상용 제품에 사용하기 위해 인증된 SweRV 코어 3개(EH1, EH2, EL2)로 구성된 제품군을 개발했습니다.

6.6.1 SweRV 코어 목록

코어 이름	파이프라인 스테이지	스레드	크기(mm @TSMC)	CoreMarks / MHz
EH1	9	싱글	0.110@28nm	4.9
EH2	9	듀얼	0.067@16nm	6.3
EL2	4	싱글	0.023@16nm	3.6

표 5: Western Digital SweRV 코어. 이미지 출처: Western Digital

여기서 크기는 Taiwan Semiconductor Manufacturing Company(TSMC)에서 받은 그림을 참조했습니다.

3개 코어는 모두 RV32IMC이며, 이는 정수 명령어 집합, 32비트 및 32개 레지스터, 정수 곱셈 및 나눗셈과 압축 명령어(16비트)를 의미합니다.

6.6.2 SweRV 코어 간의 차이점

EH1과 EH2 코어는 모두 파이프라인 측면에서 완전한 기능을 갖추고 있으며, EH1은 싱글 스레드, EH2는 듀얼 스레드를 사용합니다. EH2에는 6.3 CoreMarks/MHz라는 높은 사양 조건이 있지만 아키텍처는 더 복잡합니다.

크기와 비용이 중요한 경우 성능이 3.6 CoreMarks/MHz로 저하되긴 하지만 EL2 코어를 선택하는 것이 좋습니다. EL2 코어는 스테이지가 9개가 아니라 4개인 간소화된 파이프라인을 사용합니다.

EH1 코어는 Digilent Nexys A7 보드에서 이미 구현 및 테스트되었기 때문에 본 가이드의 실습 부분에서 사용했습니다. 향후 성능이 더 뛰어난 EH2 코어와 리소스를 적게 사용하는 EL2 코어를 추가할 계획입니다.

6.6.3 SweRV EH1 코어 자세히 알아보기

그림 72에는 SweRV EH1 코어의 블록 다이어그램이 나와 있습니다.

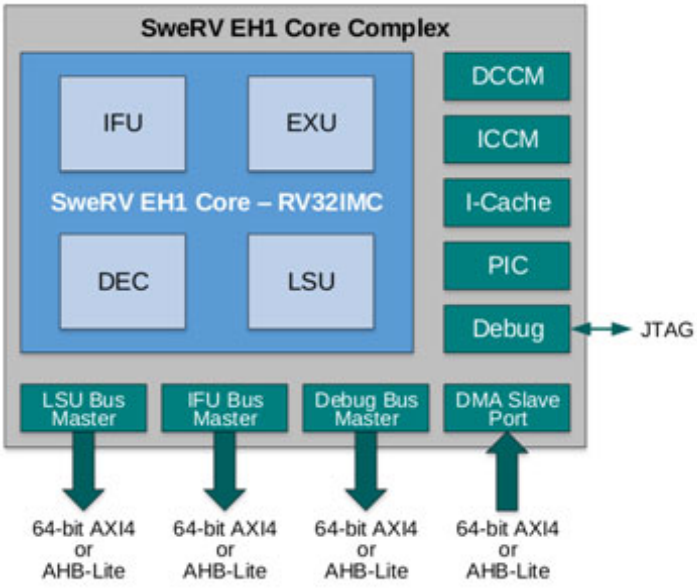


그림 72: SwerRV EH1 코어 컴플렉스. 이미지 출처: Western Digital

코어는 명령어 페치(IFU), 실행(EXU), 디코딩(DEC), 로드/저장(LSU)의 4개 유닛으로 나뉩니다.

6.6.4 SweRV EH1의 개별 구성 요소

약어	설명
IFU	명령어 페치 유닛(Instruction Fetch Unit)
EXU	실행 유닛(Execution Unit)
DEC	디코딩 유닛(Decode Unit)
LSU	로드 저장 유닛(Load Store Unit)
DCCM	Closely-Coupled Data Memory
ICCM	Closely-Coupled Instruction Memory
I-Cache	명령어 캐시
PIC	Programmable Interrupt Controller
AXI	Advanced eXtensible Interface
AHB	Advanced High Performance Bus

표 6: SwerRV EH1 구성 요소

AXI4 및 AHB-Lite는 두 가지 산업 표준 상호 연결 버스입니다. JTAG는 코드 다운로드 및 디버그에 사용됩니다.

6.7 SweRVolf 코어

SweRV EH1은 기본 컴퓨터 코어로, 실제 사용하기 위해서는 추가 구성 요소가 필요합니다. Olof Kindgren은 SweRVolf라고 하는 SweRV를 위한 래퍼 세트를 작성했습니다. RVfpga 저자들은 몇 가지 유용한 주변 장치로 SweRVolf를 확장하고 HDL 소스를 재구성했습니다.

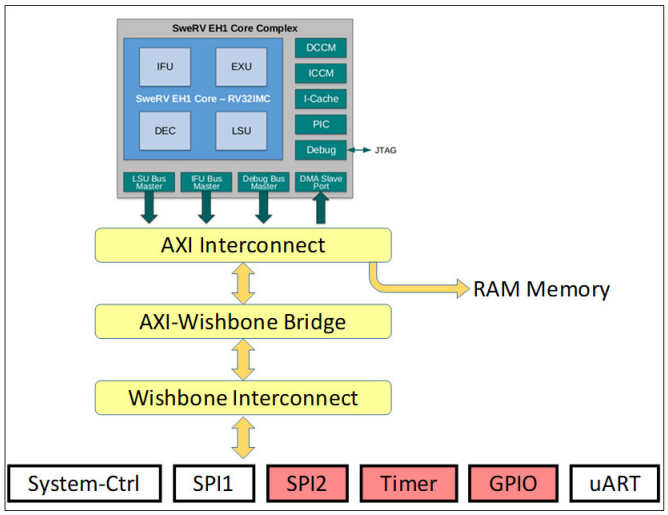


그림 73: SwerRVolf 코어. 이미지 출처: Imagination Technologies UP

6.7.1 확장된 SweRVolf 구성 요소

확장된 SweRVolf 구현은 다음 구성 요소로 나뉩니다.

- Wishbone 인터커넥트: Opencores 인터커넥트, AXI보다 간단함
- axi2wb: Wishbone 버스 컨버터에 대한 AXI
- GPIO: 64개 입력/출력 포트가 있는 Opencores의 범용 입력 출력
- 타이머: Opencores의 제품
- SPI: 오픈 소스 직렬 주변 장치 인터페이스(Serial Peripheral Interface) 컨트롤러 (https://opencores.org/projects/simple_spi에서 얻음)
- UART: 오픈 소스 UART 컨트롤러 (https://opencores.org/projects/uart16550에서 얻음)
- Boot ROM: Boot ROM에는 스테이지 1 부트로더가 포함되어 있습니다. 시스템 리셋 후 SweRV가 이 영역에서 최초 명령어를 가져오기 시작합니다.
- RAM: SweRVolf 코어에는 메모리 컨트롤러가 없지만 메모리 맵의 처음 128MB를 예약하여 AXI 버스에 대한 액세스를 제공합니다. 따라서 사용자가 RAM 메모리를 포함할 수 있습니다.
- SPI Flash: SPI Flash 메모리(또는 Quad SPI) 이전 섹션에서 설명한 SPI 컨트롤러를 사용하여 포함할 수 있습니다.

6.7.2 SweRVolf 메모리 맵

모든 구성 요소의 메모리 맵은 다음과 같습니다.

코어 맵	메모리 주소 범위
Boot ROM	0x80000000 ~ 0x80000FFF
시스템 컨트롤러	0x80001000 ~ 0x8000103F
SPI1	0x80001040 ~ 0x8000107F
SPI2	0x80001100 ~ 0x8000113F
타이머	0x80001200 ~ 0x8000123F
GPIO	0x80001400 ~ 0x8000143F
UART	0x80002000 ~ 0x80002FFF

표 7: SweRVolf 메모리 맵

6.8 SwerRVolf Verilog 및 System Verilog 파일

그림 74는 Verilog `.v`와 System Verilog `.sv` 포맷의 파일 계층 구조입니다. 모두 RVfpga 패키지로 제공됩니다.

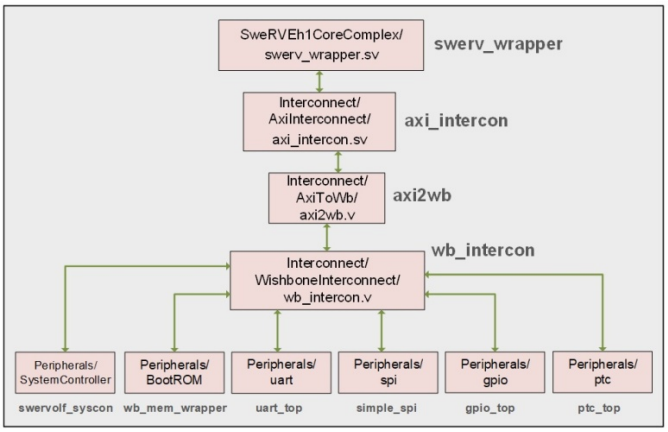


그림 74: SweRVolf 파일 구조. 이미지 출처: Imagination Technologies UP

6.9 Nexys A7의 SwerRVolf 계층 구조

SweRVolf 프로젝트의 모든 구성 요소는 RVfpga 패키지로 제공되며 다운로드할 수 있습니다.

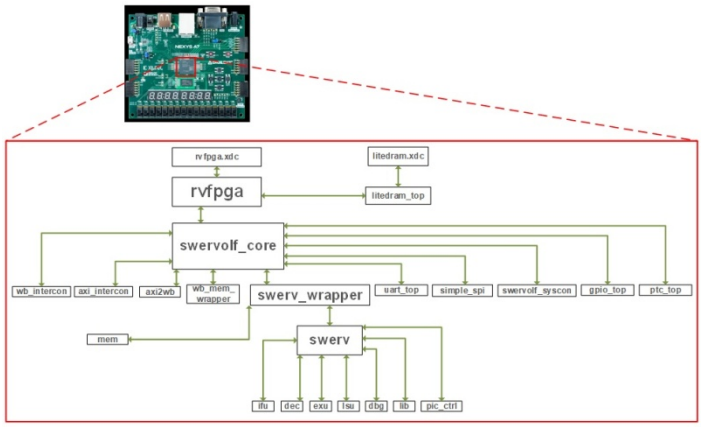


그림 75: Nexys A7 내 SweRVolf 프로젝트 계층 구조. 이미지 출처: Imagination Technologies UP

6.10 SweRVolf 구현 개선

원래 SweRVolf 구현은 주변 장치를 추가하여 확장할 수 있습니다.

6.10.1 Opencores에서 구할 수 있는 주변 장치

SweRVolf는 오픈 규격으로, Github에서 모든 소스 코드를 다운로드할 수 있습니다. 즉, 수정이 가능하고 새 주변 장치를 추가할 수 있습니다. 모든 프로젝트는 `www.opencores.org/projects`에서 다운로드할 수 있습니다. 그림 76은 인증된 오픈 소스 코어의 몇 가지 예를 보여줍니다.

그림 76: 몇 가지 인증된 오픈 소스 코어. 출처: www.opencores.org

6.10.2 연구실에서 추가된 주변 장치

연구실에서 Nexys A7 보드를 위해 다음 주변 장치가 추가되었습니다.

- 새 GPIO를 통해 푸시버튼 5개
- PWM를 사용하는 3색 LED
- I²C를 사용하는 온도 센서
- 7-세그먼트 디스플레이 드라이버
- SPI 가속도계 드라이버

7 소프트 코어 소프트웨어 설치

7.1 RVfpga 다운로드

소프트 코어 개발을 위한 소프트웨어를 설치하기 전에 다수의 필수 파일이 들어 있는 RVfgpa 패키지를 다운로드해야 합니다. 자세한 내용은 섹션 6.1을 참조하십시오.

모든 파일을 다운로드합니다. RVfpga 패키지의 적절한 위치는 홈 디렉터리입니다(예: `/home/myusername/RVfpga`). 여기서 "myusername"은 실제 컴퓨터 이름입니다. 예제가 다음과 같은 경우:

```
[RVfpgapath]/RVfpga/
```

다음과 같이 입력하면 됩니다.

```
~/RVfpga/
```

모든 파일에 읽기/쓰기 권한이 있는지 확인합니다.

이어지는 예제에서는 RVfpga의 최신 버전이 사용되지만 변경될 수 있습니다. 자세한 내용은 Imagination Technologies "RVfpga: Getting Started Guide"(RVfpga: 시작 가이드)를 참조하는 것이 좋습니다.

7.2 Xilinx Vivado 소개

이 부분은 Ubuntu 18.04 Linux에서 설치해야 할 프로그램이 여러 개이기 때문에 가장 시간이 오래 걸립니다. Windows와 Mac에 대한 지원은 현재 개발 중입니다. Xilinx 다운로드 및 설치에는 약 두 시간 반가량 걸립니다.

Xilinx를 선택한 이유는 다음과 같습니다.

1. Xilinx는 매우 다양한 사양의 제품을 제공하는 FPGA 시장에서도 최고의 제품입니다.
2. Xilinx는 FPGA 기술에 대해 가장 널리 요구되는 직업적 관점 및 시장성 있는 기술입니다. 작업 웹 사이트를 빠르게 검색하면 확인됩니다.
3. SweRVolf는 Xilinx를 사용하여 작성되었으며,
4. SiFive는 또한 코어에 Xilinx를 사용하고 있습니다.

7.3 Linux에서 Xilinx Vivado 설치에 대한 중요한 정보

Xilinx를 다운로드하기 전에 아직 수행하지 않았다면 컴퓨터에서 Ubuntu 18.04를 설치해야 합니다. Ubuntu 파티션은 100GB 이상이어야 합니다. **Ubuntu 20.04는 사용하지 마십시오.**

Xilinx에서는 자사 소프트웨어와 호환되는 Ubuntu 버전을 명시했는데, 현재 Ubuntu 20.04와 호환되지 않습니다.

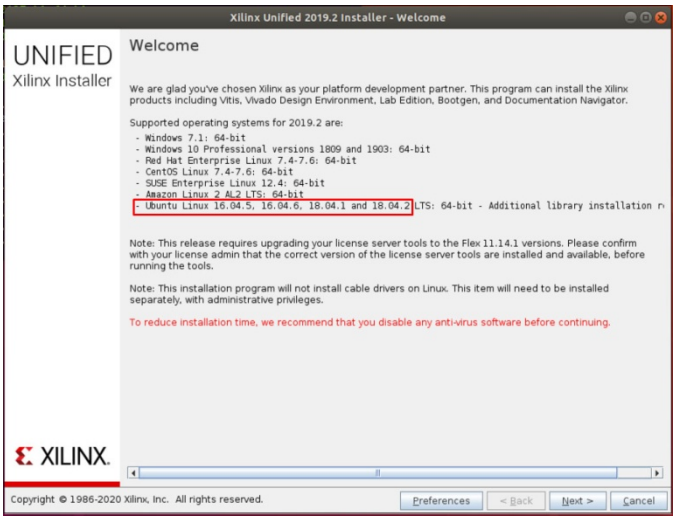


그림 77: Xilinx Unified Installer 호환성

본 가이드의 저자처럼 Ubuntu 20.04를 다운로드하고 두 시간 반이나 작업한 후에 호환 및 실행되지 않는다는 사실을 깨닫지 마십시오. 모든 항목을 삭제한 다음 올바른 Ubuntu 18.04 버전을 설치하여 이 프로세스를 처음부터 시작해야 합니다.

7.3.1 Xilinx Vivado 시스템 요구 사항

Xilinx Vivado는 크기가 16GB이고(따라서 다운로드에 오래 걸림) 설치하려면 56GB가 필요합니다(따라서 파티션 크기가 100GB여야 함). 디스크 공간이 부족하면 오류가 발생합니다.

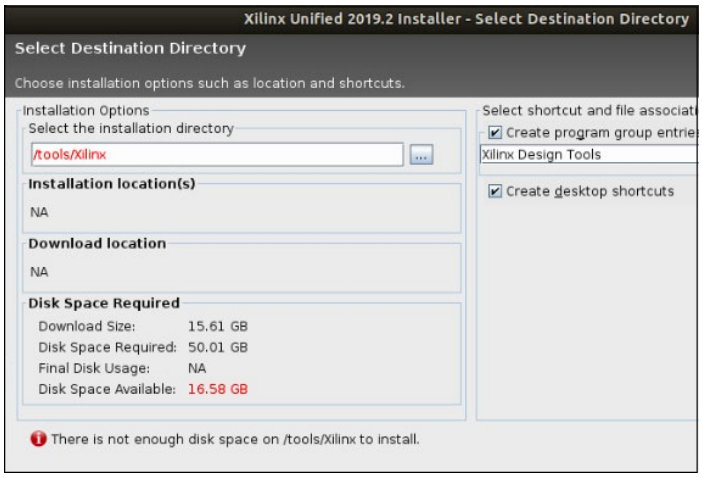


그림 78: Xilinx Vivado 시스템 요구 사항

7.4 Linux에서 Xilinx Vivado 설치

이 프로그램은 소프트 코어를 위한 Verilog `.v` 및 System Verilog `.sv` 파일을 수정 및 빌드하는 데 사용되며 해당 파일을 Digilent Nexys A7 보드에 다운로드합니다.

Digilent Nexys A7 보드를 사용할 수 없거나 컴퓨터 시뮬레이션만 실행하려는 경우 나중에 Vivado를 설치할 수 있습니다.

7.4.1 Xilinx 소프트웨어 다운로드

Xilinx 소프트웨어를 다운로드하려면 Xilinx 계정이 아직 없는 경우 계정을 만들어야 합니다.

https://www.xilinx.com/support/download.html으로 이동하여 2019.2용 다운로드를 선택합니다.

`Xilinx Unified Installer 2019.2: Linux Self Extracting Web Installer`를 클릭하여 다운로드합니다.

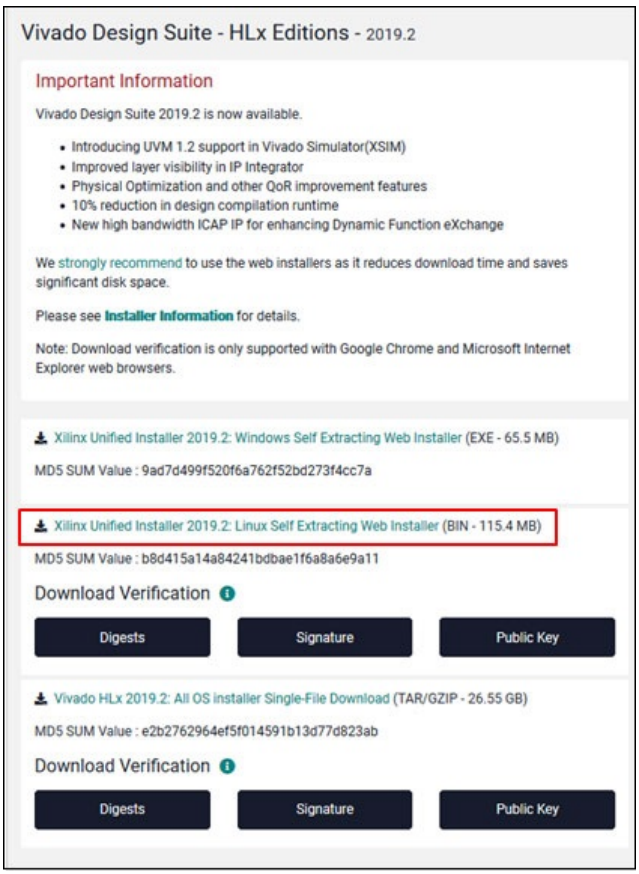


그림 79: Xilinx Unified Installer 2019.2

7.4.2 바이너리 파일의 권한 변경

`Xilinx_Unified_2019.2_1106_2127_Lin64.bin` 파일이 `/Downloads` 디렉터리에 다운로드될 경우 실행할 수 없습니다.

파일 이름을 마우스 오른쪽 버튼으로 클릭하고 *Properties-Permissions(속성-권한)*을 선택하여 *Ubuntu Files(Ubuntu 파일)*  도구로 실행할 수 있도록 만들어야 합니다. *Allow executing file as program(파일이 프로그램으로 실행되도록 허용)*을 클릭합니다.

7.4.3 바이너리 파일 실행

Ubuntu Terminal(Ubuntu 터미널) 

을 열고 다음과 같이 입력하여 루트로 설정합니다.

```
sudo su
```

Ubuntu Files(Ubuntu 파일) 
도구로 다시 이동하여 다음 파일을 *Terminal(터미널)* 창으로 끌어 놓습니다.

```
Xilinx_Unified_2019.2_1106_2127_Lin64.bin
```

*Ubuntu Terminal(Ubuntu 터미널)*에서 캐리지 리턴을 눌러 바이너리 파일을 실행합니다.

7.4.4 설치할 Vivado 제품 선택

Select Product to Install(설치할 제품 선택) 창에서 *Vivado*를 선택합니다. *Vitis*를 선택하면 안 됩니다.

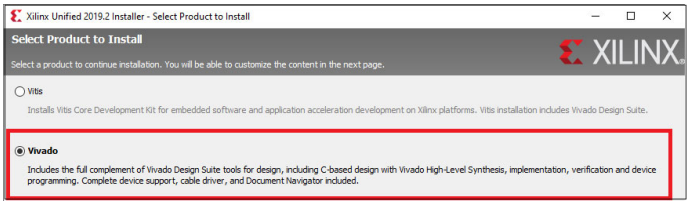


그림 80: Vivado 선택

7.4.5 WebPACK 선택

Select Edition to Install(설치할 버전 선택) 창에서 무료 버전인 *Vivado HL WebPACK*을 선택합니다.

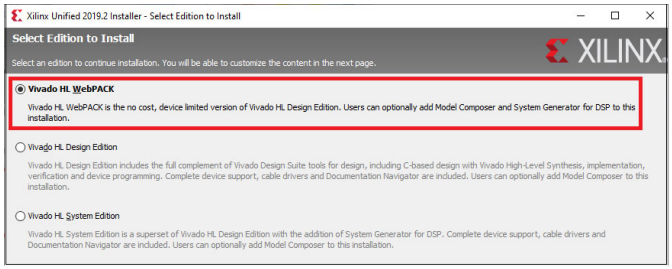


그림 81: 설치할 버전 선택

나머지 단계에 따라 설치하고 기본값을 사용합니다.

7.4.6 Digilent Nexys A7 보드용 케이블 드라이버 설치

Ubuntu Terminal(Ubuntu 터미널) 

을 열고 다음으로 이동합니다.

```
/tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/install_script/install_drivers/
```

다음과 같이 입력합니다.

```
sudo ./install_drivers
```

7.4.7 보드 파일 설치

https://github.com/Digilent/vivado-boards로 이동합니다.

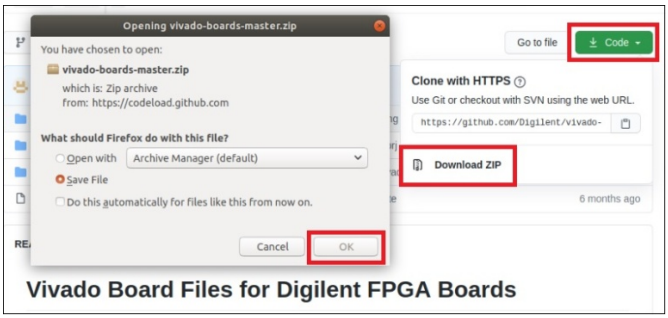


그림 82: Vivado Boards Master 다운로드

`.zip` 파일을 다운로드하여 *Ubuntu Files(Ubuntu 파일)*  도구를 사용하여 압축을 풉니다.

Ubuntu Terminal(Ubuntu 터미널)  에서 다음으로 이동합니다.

```
/Downloads/vivado-boards-
master/new/board_files
```

이 디렉터리의 파일을 `board_files` 디렉터리로 복사해야 합니다. 다음과 같이 입력합니다.

```
sudo cp -r *
```

```
/tools/Xilinx/Vivado/2019.2/data/boards/b
oard_files
```

중요한 보드 파일은 `nexys-A7-100t`입니다.

7.4.8 Linux에서 Vivado 실행

Ubuntu Terminal(Ubuntu 터미널)  을 엽니다.

다음 디렉터리로 이동합니다.

`/tools/Xilinx/Vivado/2019.2`. 그런 다음 아래와 같이 입력합니다.

```
source settings64.sh
```

```
vivado &
```

7.5 Visual Studio Code 및 PlatformIO 설치

소요 시간: 약 20분

Visual Studio Code(VSCode)는 소프트웨어 개발 및 디버그를 위한 IDE입니다.

PlatformIO는 RISC-V와 기타 여러 프로세서를 위한 지원을 제공하는 VSCode의 추가 기능입니다.

7.5.1 Visual Studio Code 다운로드

https://code.visualstudio.com/Download로 이동합니다. `.deb`를 클릭하여 파일을 다운로드합니다. 파일 크기는 현재 60.5MB입니다.

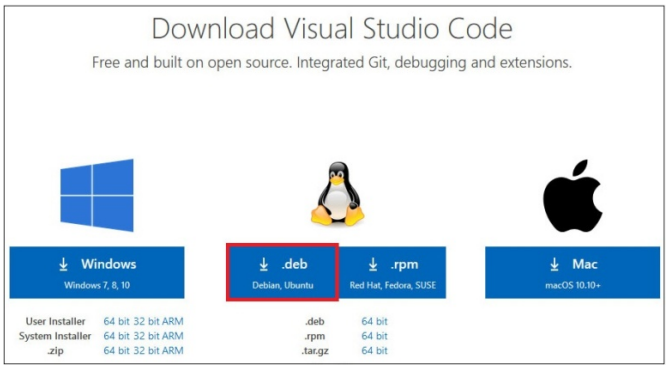


그림 83: Visual Studio Code 다운로드

7.5.2 Visual Studio Code 압축 풀기

Ubuntu Terminal(Ubuntu 터미널)  을 열고 다음과 같이 입력합니다.

```
cd ~/Downloads
```

```
sudo dpkg -i code*.deb
```

VSCode를 실행하려면 다음과 같이 입력합니다.

```
code
```

7.5.3 PlatformIO 설치

PlatformIO는 RISC-V에 대한 특정 지원을 제공합니다.

Ubuntu Terminal(Ubuntu 터미널)  에서 다음과 같이 입력하여 필요한 유틸리티 몇 가지를 로드합니다.

```
sudo apt install -y python3-distutils
python3-venv
```

VSCode가 아직 실행 중이 아닌 경우 검색 메뉴에 `Visual Studio Code`를 입력하고 찾은 다음 선택합니다. 또는 *Visual Studio Code* 아이콘을 클릭합니다.

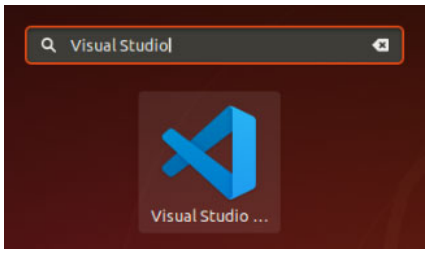


그림 84: Visual Studio Code 아이콘

7.5.4 VS Code에서 PlatformIO 확장

VSCode에서 *Extensions(확장)* 아이콘  을 클릭합니다.

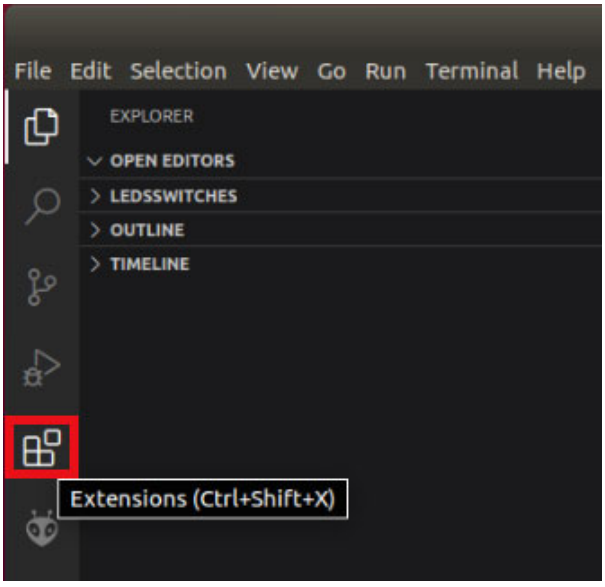


그림 85: VSCode 확장 모음

7.5.5 PlatformIO 확장 설치

검색 상자에 `platformio`를 입력합니다. *PlatformIO IDE*로 이동한 다음 *Install(설치)* 버튼을 클릭합니다.

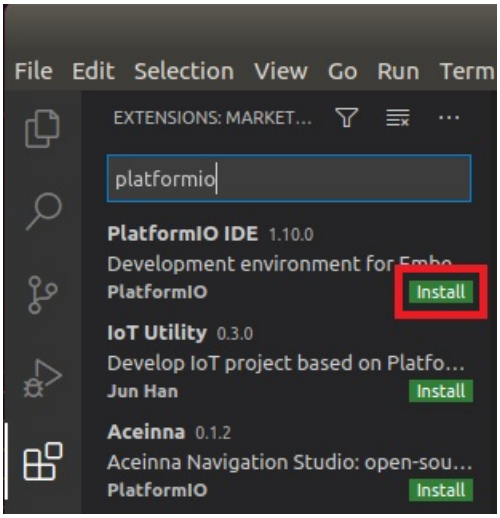


그림 86: PlatformIO IDE 확장

7.5.6 PlatformIO 다시 로드

하단의 *OUTPUT(출력)* 창에는 설치 진행에 대한 정보가 표시됩니다.

마치면 *Reload Now(지금 다시 로드)*를 선택하면 PlatformIO Core가 VSCode에 설치됩니다.

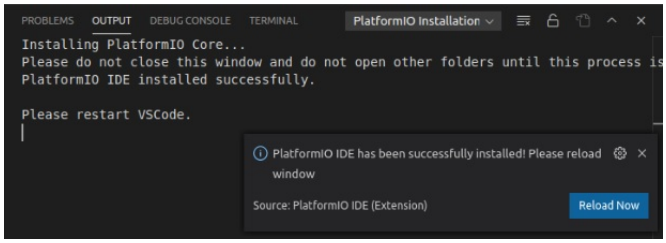


그림 87: PlatformIO 설치 후 지금 다시 로드

7.5.7 Chips Alliance 패키지 설치

다음 단계에서는 프로세서 대상을 선택합니다. 이 경우에는 다운로드한 RVfpga 패키지의 일부인 SweRVolf입니다.

(왼쪽 사이드바에 있는) Platformio  버튼을 클릭하여 *Quick Access(빠른 액세스)* 메뉴를 보고 *PlatformIO Core CLI* 옵션을 클릭합니다.

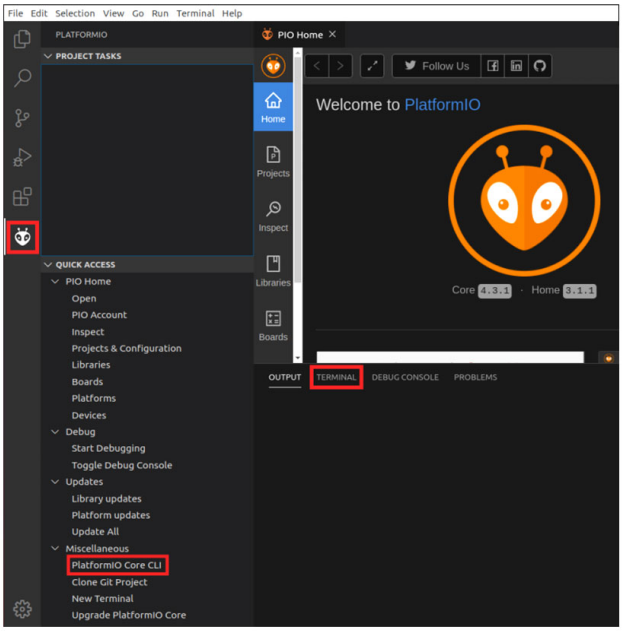


그림 88: PlatformIO 코어 대상 선택

하단 창에서 *OUTPUT(출력)* 창에서 *TERMINAL(터미널)* 창으로 변경합니다. 그러면 VSCode에서 명령을 입력할 수 있습니다.

7.5.8 Chips Alliance 플랫폼 설치

*TERMINAL(터미널)*에서 다음 명령을 실행하여 Chipsalliance 플랫폼을 설치합니다.

```
~/.platformio/penv/bin/pio platform
install [RVfpgaPath]/RVfpga/platform-
chipsalliance --with-all-packages
```

여기서 `[RVfpgaPath]`는 RVfpga 파일이 저장된 위치입니다. 예를 들어, 해당 파일은 `/home/myname/`에 있을 수 있으며 여기서 `myname`은 실제 컴퓨터 이름입니다.

패키지에는 미리 빌드된 RISC-V 툴체인, RISC-V용 OpenOCD 디버거, Verilator, Whisper, JavaScript, Python 스크립트가 들어 있습니다. RVfpga 비트 파일과 예제 코드도 제공됩니다.

7.5.9 SweRVolf가 로드되었는지 선택적으로 확인

`swervolf_nexys`가 로드되었는지 확인하기 위해 *TERMINAL(터미널)* 창에서 다음과 같이 입력합니다.

`~/.platformio/penv/bin/pio boards | more.`
그런 다음 `swervolf_nexys`를 입력합니다

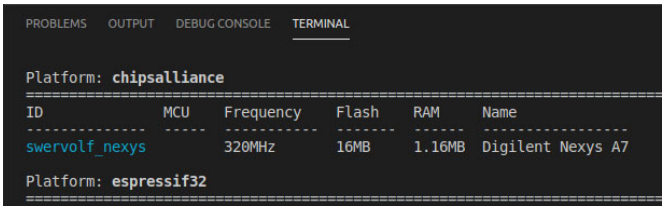


그림 89: swervof_nexys가 로드되었는지 확인

7.5.10 VSCode를 닫고 다시 열기

PlatformIO가 활성화되면 VSCode를 닫고 다시 엽니다.

참고: 일반적으로 `.platformio` 파일은 숨겨져 있습니다.

Ubuntu Terminal(Ubuntu 터미널) 

에서 `.platformio`를 검색하려면 다음과 같이 입력합니다.

```
ls -a
```

이렇게 입력하면 숨겨진 파일이 표시됩니다.

또는 *Ubuntu Files(Ubuntu 파일)* 도구  을 사용하여 숨겨진 파일을 표시합니다.

7.6 Verilator 및 GTKWave 설치

소요 시간: 약 20분

Verilator는 PlatformIO에서 사용하는 Verilog 및 System Verilog 파일용 하드웨어 시뮬레이터입니다.

Linux에서 기본적으로 실행되도록 Verilator를 설치하려면 *Ubuntu Terminal(Ubuntu 터미널)* 

을 엽니다.

기본적으로 설치되지 않는 필수 유틸리티를 설치하여 시작합니다.

```
sudo apt-get install git make autoconf
g++ flex bison libfl2 libfl-dev
```

Verilator 프로그램 설치

```
git clone
https://git.veripool.org/git/verilator
cd verilator
git pull
git checkout v4.020
autoconf
./configure
make
sudo make install
```

마지막으로 파형 뷰어를 설치합니다.

```
sudo apt-get install -y gtkwave
```

그림 90: GTK 파형 아이콘

7.6.1 일반적인 Verilator 파형

Verilator는 하드웨어 시뮬레이터로 사용하기에 복잡하고 본 문서의 범위를 벗어납니다. 자세한 내용은 "RVfpga: Getting Started Guide"(RVfpga: 시작 가이드)를 참조하십시오. 그러나 클록 및 명령어 페치 유닛(IFU)의 일반적인 파형은 다음과 같습니다.

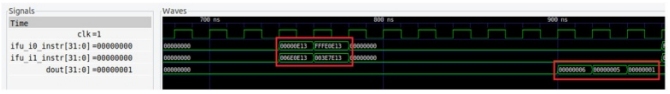


그림 91: 일반적인 Verilator 파형

7.7 PlatformIO에서 Whisper를 사용하여 C 프로그램 실행

Western Digital에는 하드웨어 없이 SweRV RISC-V 코어의 소프트웨어 확인을 위해 개발된 Whisper라고 하는 Instruction Set Simulator(ISS)가 있습니다. RVfpga 다운로드에 포함되어 있습니다.

7.7.1 폴더 열기

VSCode를 열고 PlatformIO로 이동합니다.

PlatformIO의 상단 메뉴 모음에서 *File(파일)*, *Open Folder (폴더 열기)*를 차례로 클릭합니다. 파일을 실제로 열 필요는 없습니다.

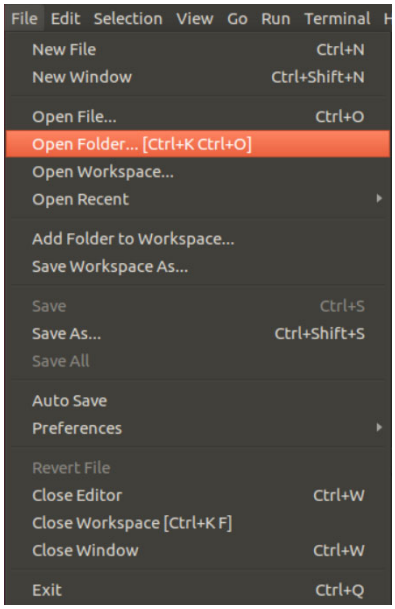


그림 92: 시뮬레이션 파일의 폴더 열기

7.7.2 벡터 정렬 Whisper 예제 코드

`VectorSorting` 디렉터리를 선택하고 *OK(확인)*를 클릭합니다.

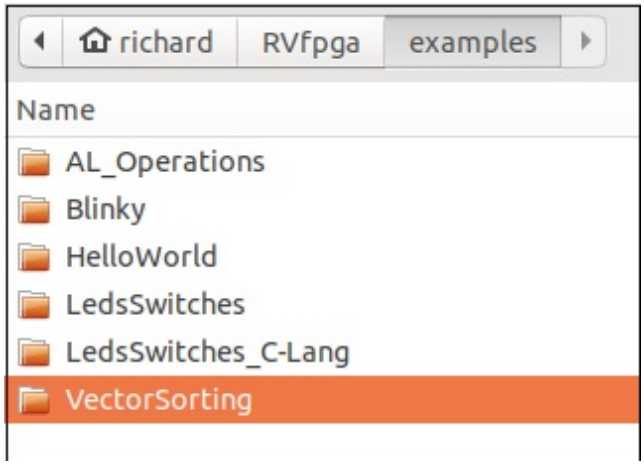


그림 93: VectorSorting 폴더

7.7.3 platformio.ini 수정

Whisper를 사용하려면 `platformio.ini` 파일을 수정해야 합니다. 해당 파일을 엽니다.

다음과 같이 명령줄의 주석 처리를 제거합니다.

```
debug_tool = whisper
```

그런 다음 저장합니다.

중요: `debug_tool`은 위에 표시된 것처럼 정확하게 입력하고 맨 앞에 공백이 없어야 합니다. 그림 94는 맨 앞에 예기치 않은 공백이 있는 경우 발생하는 오류는 보여줍니다.

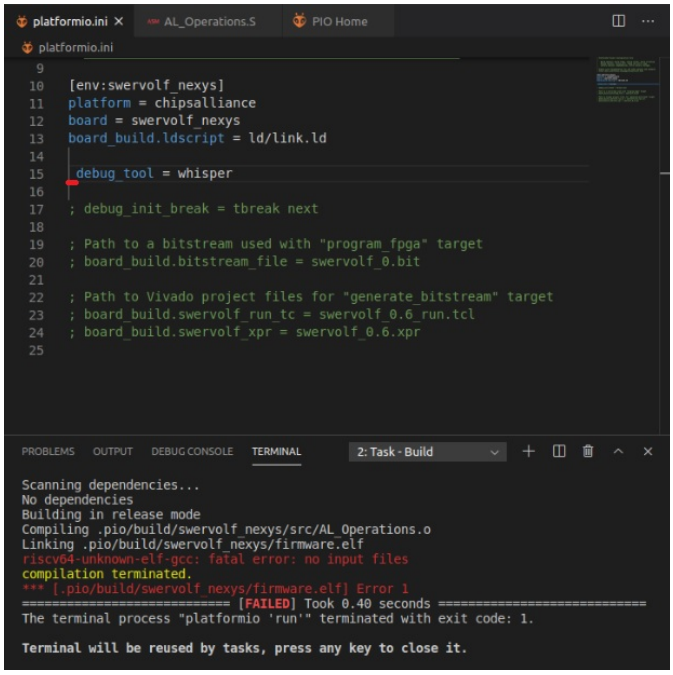


그림 94: 잘못된 platformio.ini 설정

7.7.4 Whisper를 사용하여 시뮬레이션 실행

10번 줄 왼쪽을 클릭하여 중단점을 배치합니다.

 및 RUN PIO Debug 을 차례로 클릭하여 디버거를 실행합니다.

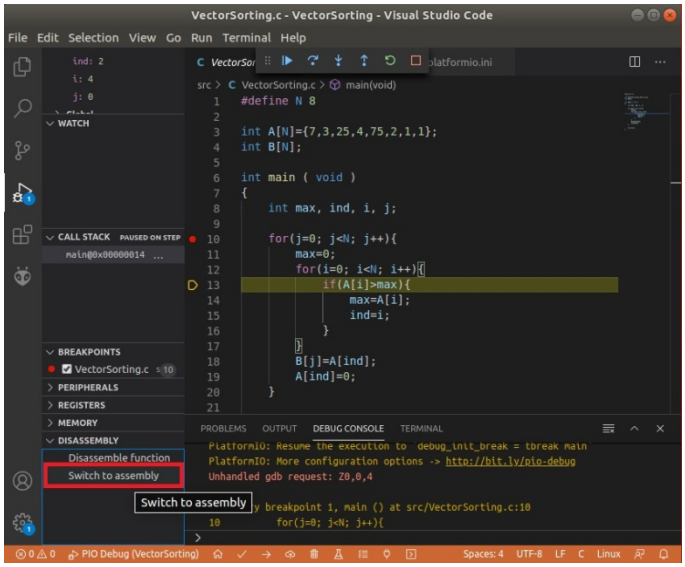


그림 95: 프로그램 RISC-V 명령어와 10번 줄의 중단점

이제 키보드 기능 키 F10을 사용하여 코드를 한 단계씩 실행할 수 있습니다.

화면 왼쪽 구석 아래에서 *Disassembly(디스어셈블리)*를 클릭하면 RISC-V 지침이 표시됩니다.

7.7.5 RISC-V 어셈블리 언어 보기

C 컴파일러로 생성된 어셈블리 언어는 그림 96에 나와 있습니다.

C 소스 코드로 되돌아가려면 *Switch to code(코드로 전환)*를 클릭합니다.

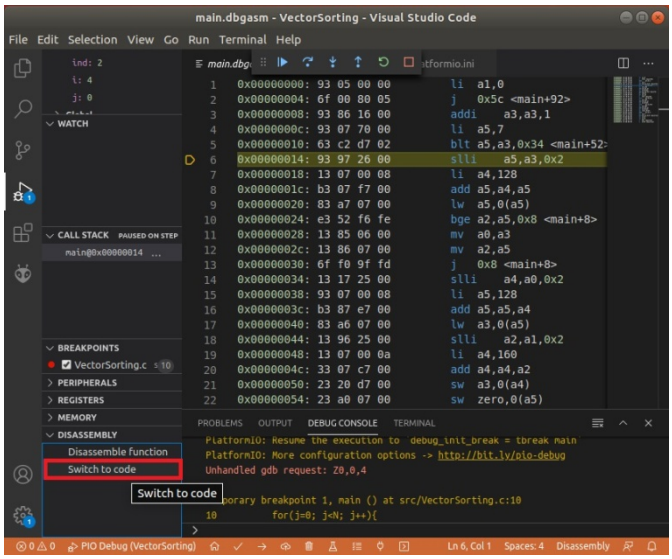


그림 96: RISC-V 어셈블리 언어 보기

디버깅에 대한 자세한 내용은 "RVfpga: Getting Started Guide"(RVfpga: 시작 가이드)를 참조하십시오.

8 SweRVolf 코어를 빌드하여 FPGA로 다운로드

소요 시간: 약 5분

8.1 SweRVolf 코어 빌드

다행히 마드리드 콤플루텐세 대학교 RVfpga 팀 덕분에 이 부분은 수행할 필요가 없습니다. RVfgpa 패키지의 일부로, Xilinx Vivado 프로젝트가 Digilent Nexys A7 하드웨어로 직접 다운로드할 수 있는 RVfpga.bit 파일과 함께 제공되었습니다.

8.2 Nexys A7 보드 연결

제공된 USB 케이블을 사용하여 Nexys4 A7 보드를 연결합니다. 공장에서 설정한 기본값이 꺼져 있으므로 전원 스위치를 켜짐 위치로 변경 합니다.

8.3 Digilent Nexys A7으로 코어 다운로드

그 전에 `RVfpga.bit` 파일이 포함된 RVfpga 패키지를 다운로드해야 합니다. 섹션 6.1을 참조하십시오.

8.3.1 Vivado에서 하드웨어 관리자 열기

섹션 7.4.8에 설명된 대로 Vivado를 엽니다.

*Open Hardware Manager >(하드웨어 관리자 열기>)*를 클릭합니다

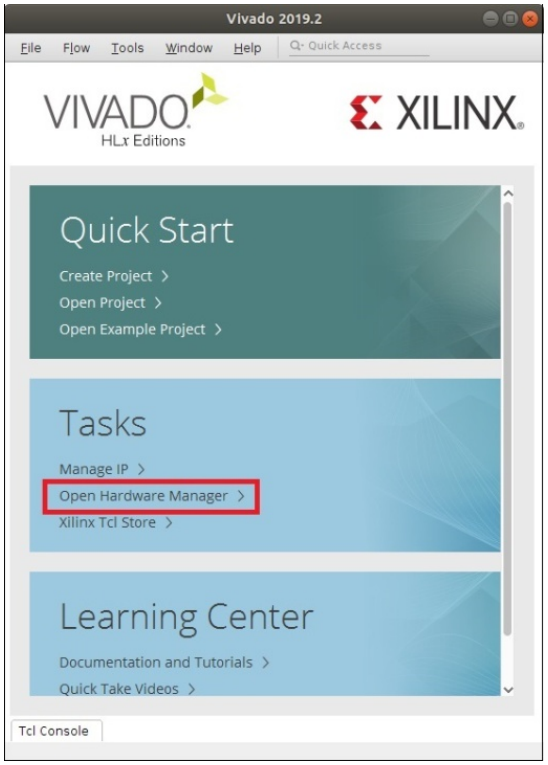


그림 97: Vivado 하드웨어 관리자 열기

8.3.2 하드웨어 대상 열기

하드웨어 관리자가 하드웨어가 열리지 않았는지 자동으로 감지합니다. *Open Target(대상 열기), Auto-detect (자동 감지)*를 차례로 클릭합니다.

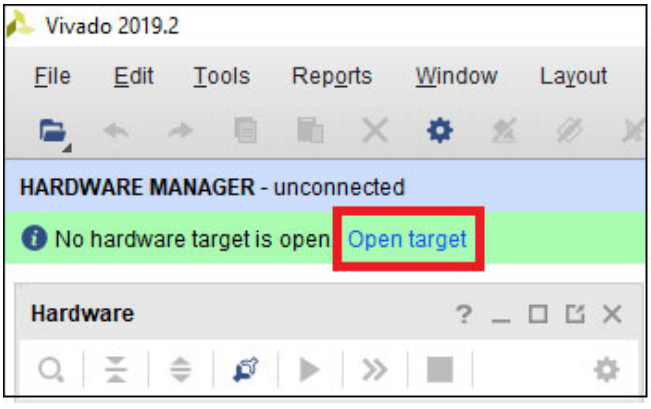


그림 98: 하드웨어 대상 열기

8.3.3 FPGA 구성 프로그래밍

*Program device(디바이스 프로그래밍)*를 클릭합니다.

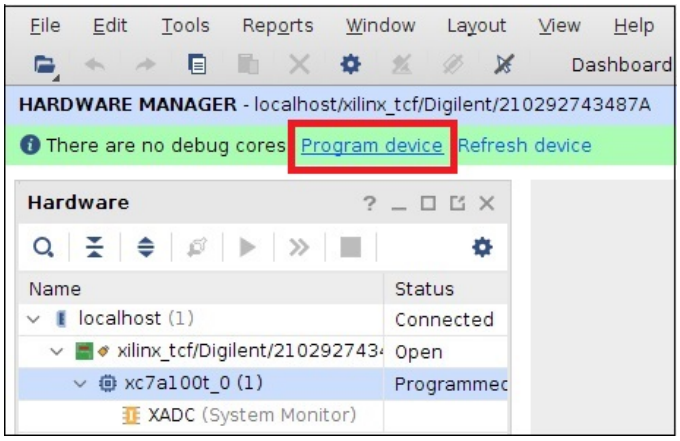


그림 99: FPGA 구성 프로그래밍

8.3.4 Bitstream 파일 선택

Bitstream 파일로 다음 파일을 선택합니다.

```
[RvfpgaPath]/RVfgpa/src/RVfpga.bit
```

Debug probes file(프로브 파일 디버그) 필드는 비워둘 수 있습니다. *Program(프로그래밍)*을 클릭합니다. 보드는 약 10초 후에 프로그래밍됩니다.

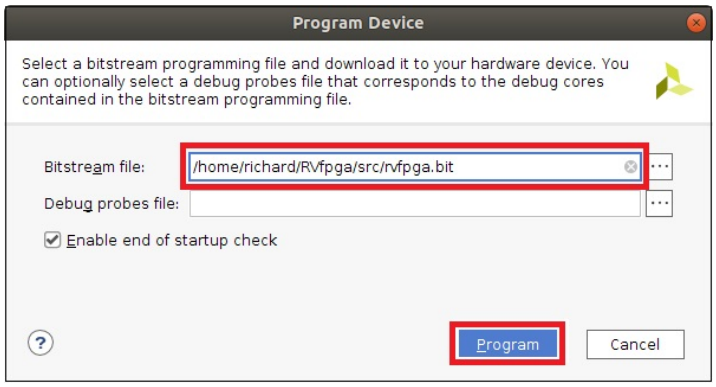


그림 100: Bitstream 파일 다운로드

8.3.5 Vivado 닫기

중요한 단계입니다. USB 포트를 사용할 수 있도록 Vivado를 종료합니다. 그러지 않으면 VSCode를 사용하여 프로그램을 다운로드한 후 실행할 수 없습니다.

오른쪽에서 *Close Hardware Manager(하드웨어 관리자 닫기)* 아이콘  을 클릭합니다.

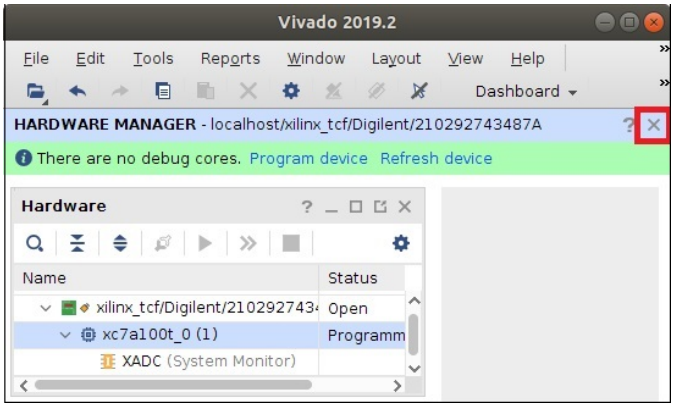


그림 101: Vivado 종료

8.4 NexysA7 보드에서 프로그램 실행

소요 시간: 약 5분

8.4.1 실행할 프로그램 열기

검색 창에 `Visual Studio Code`를 입력하고 *VSCode*  아이콘을 클릭하여 VSCode를 엽니다.

상단 메뉴 모음에서 *File(파일)*, *Open Folder(폴더 열기)*를 차례로 클릭합니다.

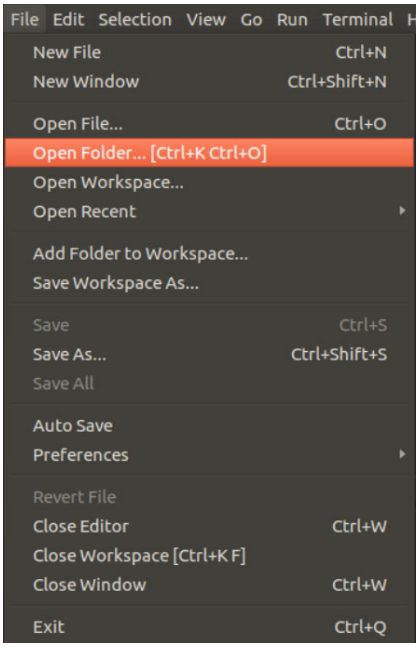


그림 102: 폴더 열기

파일을 열 필요는 없습니다.

8.4.2 프로그램 LedsSwitches 선택

`RVfpga/examples`로 이동한 다음 `LedsSwitches`를 클릭합니다. 다른 예제는 나중에 시도할 수 있습니다.

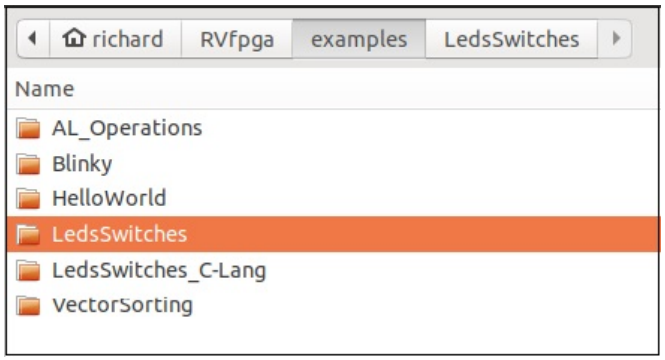


그림 103: LED 및 스위치 프로그램

8.4.3 LED 및 스위치 프로그램 열림

이 예제는 16개 스위치를 계속해서 읽은 다음 그 값을 16개 LED로 출력하는 작은 어셈블리 언어 프로그램입니다. RVfpga 예제에는 C 코드 버전도 있습니다.

*>src*를 확장한 다음 `LedsSwitches.S`를 클릭하면 어셈블리 코드가 표시됩니다.

화면 왼쪽 아래에서 *PlatformIO Build(PlatformIO 빌드)* 아이콘  을 클릭합니다.

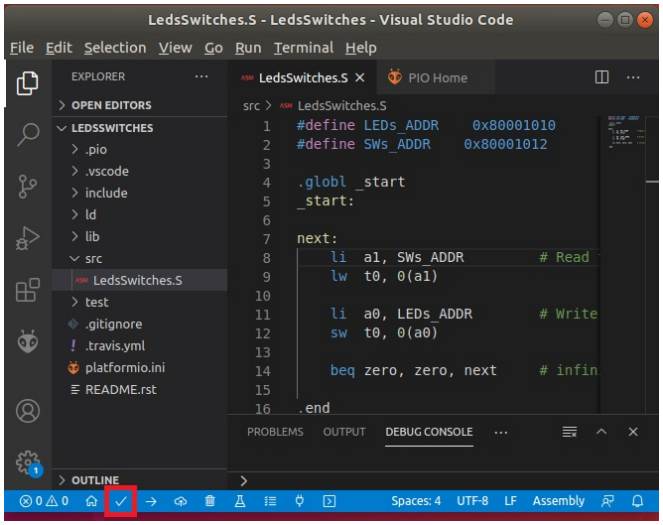


그림 104: LedsSwitches 프로그램 빌드

8.4.4 LedsSwitches 프로그램 실행

도구 모음에서 *Run(실행), Start Debugging(디버깅 시작)*을 차례로 선택하거나 키보드 기능 키 F5를 누릅니다. 프로그램이 계속 실행됩니다.

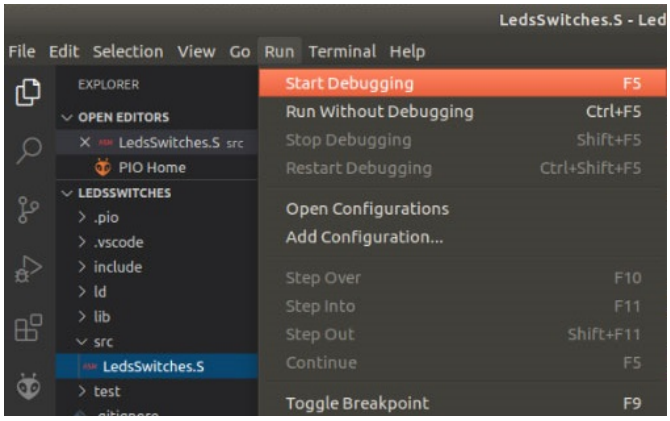


그림 105: LED 및 스위치 프로그램 디버깅 시작

8.4.5 RISC-V 코어를 실행하는 Nexys A7 보드 - 스위치로 제어되는 LED

보드 하단에서 슬라이드 스위치를 작동합니다. 녹색 LED가 개별적으로 켜지고 꺼질 수 있습니다.

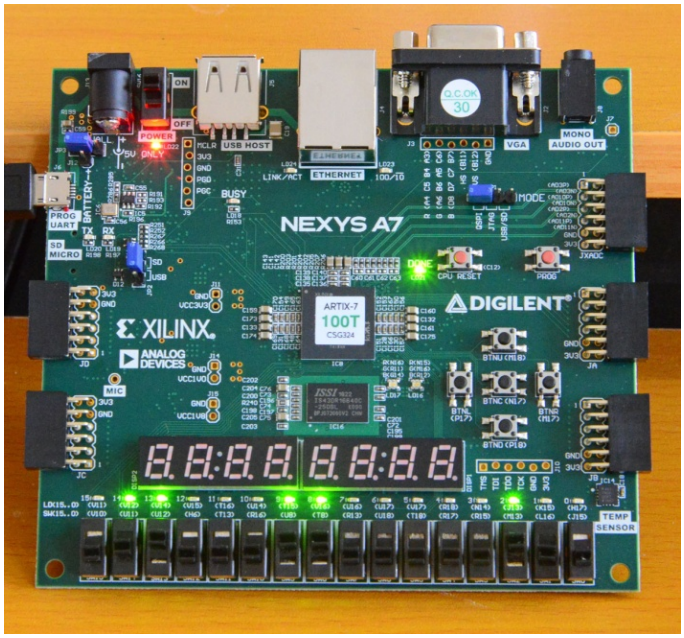


그림 106: RISC-V 코어를 실행하는 Nexys A7 보드

참고: `RVfpga.bit`의 최신 버전에서 7-세그먼트 디스플레이에는 0000 0000이 표시됩니다. 확실히 알 수 있도록 여기서는 더 간단한 버전을 보여줍니다.

9 RISC-V를 사용한 제품 개발

본 가이드에서는 실습 세션에 Kendryte K210 및 SiFive FE310-G002, 두 개의 SoC RISC-V 프로세서를 사용했습니다. 이 섹션에서는 추가 개발 또는 대량 생산된 제품을 위해 사용자의 보드에 빌드된 이 두 가지 디바이스가 무엇을 제공할 수 있는지 살펴봅니다.

9.1 Kendryte K210

Seeed Technologies Co Ltd Maix BiT 보드에 사용되며, 적당한 가격의 매우 강력한 프로세서입니다.

9.1.1 Kendryte K210 프로세서 코어

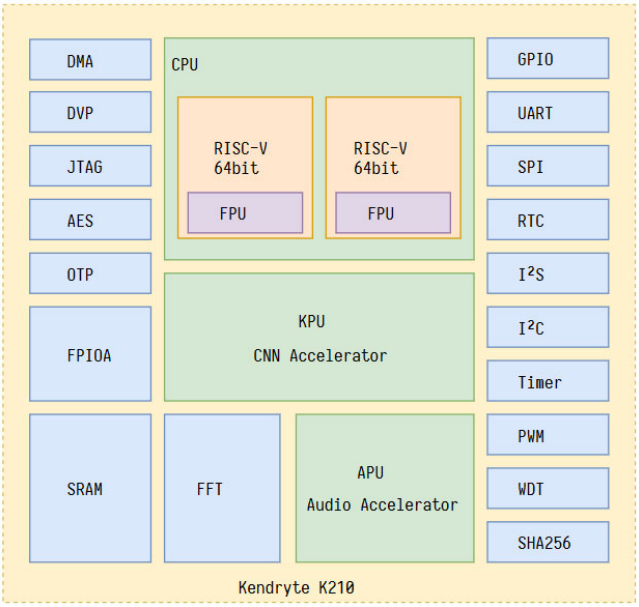


그림 107: Kendryte K210 아키텍처 개요. 이미지 출처: Kendryte K210 데이터시트

여기에는 400MHz에서 실행되는 RISC-V 64비트 부동소수점 프로세서가 1개가 아닌 2개가 들어 있습니다. 또한 유용한 인터넷 구성 요소와 인터페이스가 광범위하게 들어 있습니다.

약어	설명
CNN	신경망
DMA	Direct Memory Address
DVP	디지털 비디오 포트(Digital Video Port)
JTAG	프로그래밍 및 디버깅 인터페이스(Programming and Debug Interface)
AES	고급 암호화 표준(Advanced Encryption Standard)
OTP	One Time Programmable Flash
FPIOA	Field Programmable Input and Output Array(GPIO를 임의 핀으로 매핑 가능)
SRAM	Static Random Access Memory
FFT	고속 푸리에 변환(Fast Fourier Transform)
GPIO	범용 입력/출력(General Purpose Input/Output) 3V3 및 1V8
UART	직렬 포트
SPI	직렬 주변 장치 인터페이스(Serial Peripheral Interface)
RTC	실시간 시계(Real Time Clock)
I2S	Inter-Integrated Sound. 오디오 코덱을 위한 표준 인터페이스
I2C	저속 주변 장치를 위한 Inter-integrated Circuit 버스
PWM	Pulse Width Module
WDT	워치독 타이머(Watch Dog Timer)
SHA256	데이터 무결성 검증을 위한 안전한 해시 알고리즘

표 8: Kendryte K210 약어

GPIO가 듀얼 3V3/1V8이면 매우 유용합니다. 일부 무선 모듈(예: LTE/GSM 모듈)은 1V8 작동만 지원합니다. 이중 전압 GPIO가 없으면 추가 레벨 컨버터가 필요해 비용이 추가로 들어갈 수 있습니다.

Field Programmable Input/Output Array(FPIOA)는 핀을 그룹화할 수 있어 인쇄 회로 기판(Printed Circuit Board, PCB)의 배선을 더 쉽게 만듭니다.

Kendryte K210 프로세서는 Ball Grid Array(BGA) 패키지에서만 사용할 수 있어 장비를 배치하는 경우에만 적합합니다.

9.1.2 Seeed Technologies MaixPy BiT 구현

또는 실험하거나 적은 양만 생산하는 경우 기본 MaixPy BiT는 장착 핀을 사용하여 브레드보드에 장착하거나 다른 PCB에 납땜할 수 있습니다.

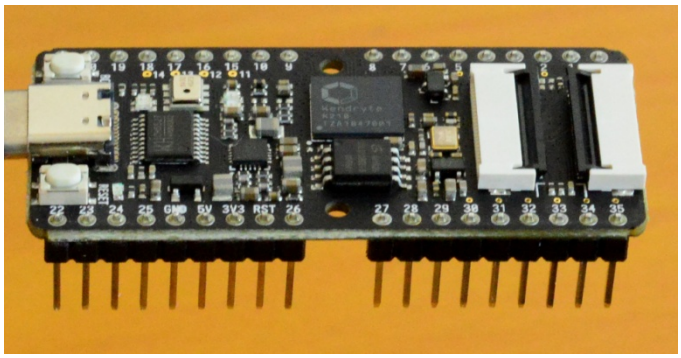


그림 108: 장착 핀이 있는 Maix BiT 보드. Seeed Technologies Maix-I 모듈

9.1.3 Maix-I

PCB에 프로세서를 구현할 때 문제는 디커플링 캐패시터, 크리스털, DC-DC 컨버터, 인덕터, 리셋 회로 등 프로세서가 작동하기 위해 필요한 추가 구성 요소가 굉장히 많다는 점입니다.

다음 두 Maix-I 모듈을 사용하면 하드웨어 엔지니어가 좀 더 편해질 수 있습니다.

MAIX-I Digi-Key 1597-1717-ND $8.91

Maix-I with Wi-Fi Digi-Key 1597-1715-ND $10.82.

적은 양을 생산하는 경우 또는 초기에 개발한 보드의 경우 해당 모듈은 매력적인 옵션입니다.

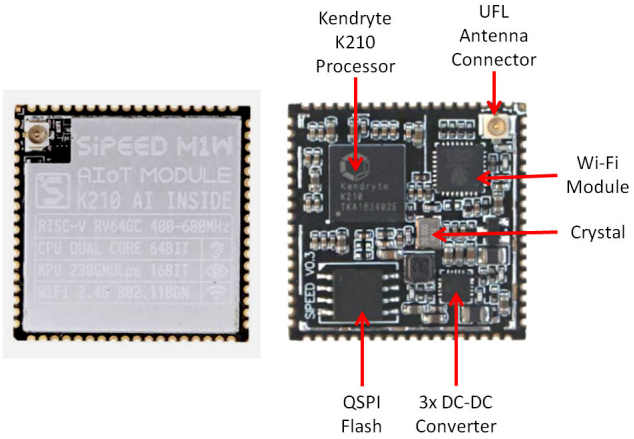


그림 109: Maix-I with Wi-Fi 모듈. 이미지 출처: Seeed Technologies Co Ltd(수정됨)

두 Maix-I 모듈에는 Kendryte K210 프로세서뿐만 아니라 Quad SPI Flash, 크리스털, DC-DC 컨버터가 포함되어 있고, 옵션으로 비용을 좀 더 지불하면 Wi-Fi 모듈을 사용할 수 있습니다.

가능한 응용 분야는 간단한 직렬 포트를 통해 메인 프로세서에서 파생되는 전용 그래픽 프로세서입니다. MicroPython에서 프로젝트가 개발되었기 때문에 소프트웨어 개발 시간이 단축될 수 있습니다.

9.1.4 MaixPy BiT에 대한 Visual Studio Code 지원

MaixPy BiT에서는 프로그래밍에 MicroPython을 사용하지만 C 또는 C++도 사용할 수 있습니다.

Xilinx FPGA 소프트 코어에 사용되었으므로 PlatformIO 역시 MaixPy BiT 보드와 Maix-I 모듈에 사용된 Kendryte K210을 지원합니다. 그러나 본 가이드의 저자는 이 부분은 시도해 보지 않았습니다.

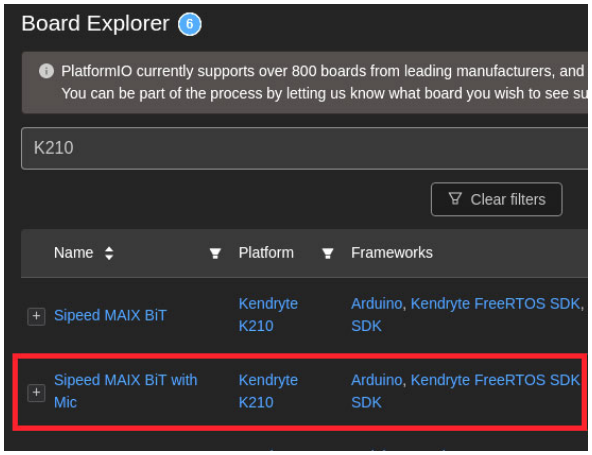


그림 110: MaixPy BiT에 대한 VSC PlatformIO 지원

9.2 SiFive

RED-V Redboard에 사용된 프로세서는 SiFive Freedom Everywhere FE310-G002입니다.

9.2.1 Freedom Everywhere FE310-G002 RISC-V 핀 배치도

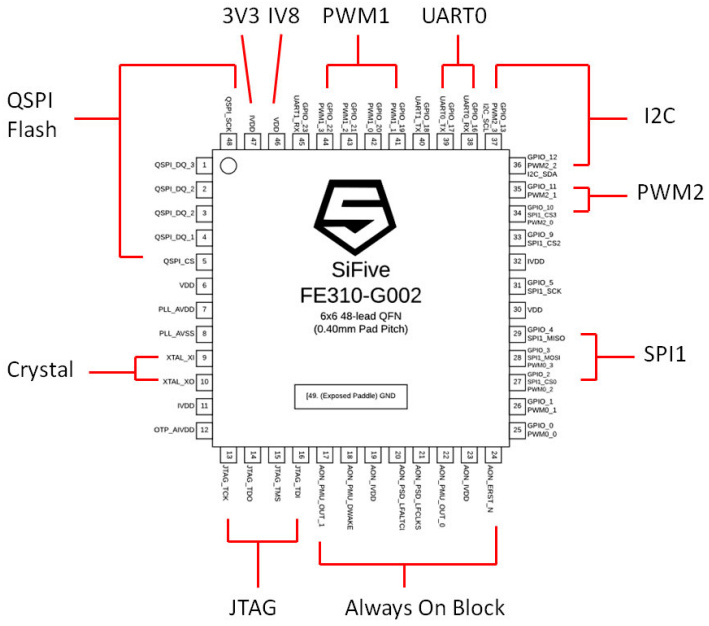


그림 111: SiFive FE310-G002 핀 배치도. 이미지 출처: SiFive FE310-G002 데이터시트(수정됨)

FE310-G002는 48 핀 QFN 패키지에서 사용할 수 있으며 최대 320MHz까지 실행됩니다.

9.2.2 FE310-G002 GPIO 핀 다중 기능

Name	Pin	GPIO	PWM	SPI	UART	i2C
GPIO_0	25	0 I/O	PWM0_0 O			
GPIO_1	26	1 I/O	PWM0_1 O			
GPIO_2	27	2 I/O	PWM0_2 O	SPI1_SS0		
GPIO_3	28	3 I/O	PWM0_3 O	SPI1_MOSI		
GPIO_4	29	4 I/O		SPI1_MISO		
GPIO_5	31	5 I/O		SPI1_SCK		
GPIO_9	33	9 I/O		SPI1_SS2		
GPIO_10	34	10 I/O	PWM2_0 O	SPI1_SS3		
GPIO_11	35	11 I/O	PWM2_1 O			
GPIO_12	36	12 I/O	PWM2_2 O			I2C0_SDA
GPIO_13	37	13 I/O	PWM2_3 O			I2C0_SCL
GPIO_16	38	16 I/O			UART0_RX I	
GPIO_17	39	17 I/O			UART0_TX O	
GPIO_18	40	18 I/O			UART1_TX O	
GPIO_19	41	19 I/O	PWM1_1 O			
GPIO_20	42	20 I/O	PWM1_0 O			
GPIO_21	43	21 I/O	PWM1_2 O			
GPIO_22	44	22 I/O	PWM1_3 O			
GPIO_23	45	23 I/O			UART1_RX I	

표 9: SiFive 포트 핀 보조 기능. 이미지 출처: SiFive FE310-G002 데이터시트

GPIO 핀은 다중적 입니다. 즉, 최대 3xPWM, 1xSPI, 1xI2C, 2xUART가 있을 수 있습니다. 아날로그-디지털 컨버터(Analog to Digital Converter , ADC)는 없습니다. QFN48 패키지이기 때문에 손으로 직접 납땜(또는 납땜 제거)이 어려워 장비를 이용하여야 합니다.

9.2.3 SparkFun Electronics RED-V SiFive Thing Plus 구현

실험하거나 적은 양만 생산하는 경우 RED-V SIFIVE RISC-V THING PLUS도 사용할 수 있습니다. 1568-DEV-15799-ND 가격은 $29.95이며, 연결 핀을 추가해야 할 수 있습니다. 회로도는 SparkFun 웹사이트에서 얻을 수 있습니다.

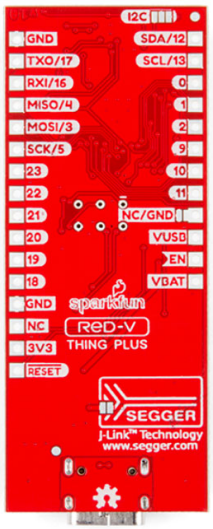


그림 112: SparkFun Electronics SiFive Thing Plus 저면도. 이미지 출처: SparkFun Electronics

10 RISC-V에 대한 추가 정보

RISC-V에 대한 추가 정보를 제공하기 위해 RISC-V Summits의 공식 기록을 YouTube와 RISC-V.org 웹사이트에서 확인할 수 있습니다. RISC-V의 발명자인 David Patterson과 Krste Asanovic이 발표자들을 중재하며 대화를 이끕니다. 또한 YouTube에는 Maix BiT, SparkFun, Digilent 보드에 대한 동영상도 올라와 있습니다.

여기서 설명한 대로 FPGA에서 SweRV 코어를 실행한 후 더 깊은 지식을 쌓고자 하는 독자에게 2020년 11월에 배포한 Imagination Technologies "RVfpga: The Complete Course in Understanding Computer Architecture" (RVfpga: 컴퓨터 아키텍처 이해를 위한 완벽한 과정)는 많은 도움이 될 수 있습니다. 해당 자료는 다음과 같이 4개 파트로 나뉘어 20개 연구소의 자료를 다루고 있으며, 파트 1의 일부 내용은 본 가이드에서 다뤘습니다.

파트 1: Vivado 프로젝트 및 프로그래밍 – Vivado 및 Verilator, C 프로그래밍, Whisper, RISC-V Application Binary Interface(ABI), 프로시저 호출 규약, C와 어셈블리 코드 병합.

파트 2: I/O 시스템 – Digilent Nexys A7 보드, 인터럽트, 타이머, 직렬 버스(SPI, I2C 및 UART)의 7-세그먼트 디스플레이 구동.

2021년 3분기에 다음 파트가 배포될 예정입니다.

파트 3: RISC-V 코어 – 코어 구조, 파이프라인, 해저드 이해, 새로운 명령어 구현

파트 4: RISC-V 메모리 시스템 – 캐시 컨트롤러, 캐시 적중률 및 누락 이해, 캐시 수정, Closely-Coupled Instruction Memory(ICCM) 및 Closely-Coupled Data Memory(DCCM) 이해

RISC-V에 대한 관련 서적 2권에 대한 정보는 참조 자료 섹션에 나와 있습니다.

11 결론

RISC-V를 위한 여러 보드를 살펴본 후 가장 사용하기 쉽고 흥미로운 보드는 MaixPy BiT로 입증되었습니다. 가장 저렴한 보드는 카메라와 LCD가 장착된 듀얼 코어 RISC-V 프로세서의 형태로 적당한 가격에 가장 뛰어난 컴퓨팅 성능을 제공한다는 점이 흥미롭습니다. 해당 보드는 머신 비전과 얼굴 인식에 대해 배우기에 아주 좋은 도구입니다. MicroPython을 사용하여 프로그래밍되었기 때문에 프로그래밍 배경 지식이 없어도 쉽게 사용할 수 있습니다.

SparkFun Electronics RED-V Thing Plus와 Red Board는 취미로 배우거나 자체 제품을 개발하겠다는 의지를 가지고 C 언어 또는 어셈블리 언어로 RISC-V를 프로그래밍하는 방법을 배우고자 하는 전기 기술자/소프트웨어 엔지니어에게 적합합니다. SiFive의 소프트웨어 도구는 쉽게 사용할 수 있고, 다른 응용 분야에 적용할 수 있는 유용한 예제가 많습니다. SiFive의 문서도

어렵지 않습니다. ARM 또는 TI 프로세서용 Eclipse 도구 사용자는 읽는 데 전혀 어려움이 없을 것입니다.

RISC-V에 대한 본 가이드에서는 Xilinx FPGA에서 소프트 코어 구현을 간단하게 소개하는 것입니다. 본 가이드는 컴퓨터 아키텍처를 공부하는 학생과 주요 프로젝트를 진행하는 엔지니어를 대상으로 합니다. Xilinx FPGA에서 소프트 코어 구현은 복잡하기 때문에 자세한 지침 없이는 권장하지 않습니다. 소프트 코어를 사용해 본 적은 없지만, Imagination Technologies "RVfpga: Getting Started Guide(RVfpga: 시작 가이드)"를 주의 깊게 따르고 RVfppga 패키지를 사용함으로써, 하루 만에 Nexys A7 보드에서 RISC-V 코어를 성공적으로 실행할 수 있었습니다. 4개 보드 중 Nexys A7 보드가 기술적으로 가장 뛰어났습니다. 더 많이 배우고 싶어 하는 독자를 위한 후속 활동으로 Imagination Technologies "RVfpga: The Complete Course in Understanding Computer Architecture"(RVfpga: 컴퓨터 아키텍처 이해를 위한 완벽한 과정)가 있습니다.

RISC-V는 많은 노력이 들어가 있으며, 시간이 흐르면 ARM의 경쟁 기술로 얼마나 효과가 있는지 알게 될 것입니다. 오픈 소스 ISA와 기술 사용료가 없다는 점은 RISC-V의 가장 확실한 경쟁력이며 매우 강력한 일부 프로세서도 저렴한 비용으로 사용할 수 있습니다.

12 감사의 글

본 가이드의 저자가 RISC-V 강의 자료를 사용하도록 허락해 주신 마드리드 대학교의 Daniel Chaver Martinez 교수님과 석사 논문을 사용하도록 허락해 주신 대학원생 Daniel Gonzalez에게 깊은 감사를 드립니다. 이러한 도움이 없었다면 본 가이드를 작성하지 못했을 것입니다.

마지막으로, 본 가이드에 대한 아이디어를 생각해 실현하도록 만든 Imagination Technologies의 Robert Owen에게도 특별히 감사드립니다.

13 저자 프로필

Richard Sikora는 30년 넘게 하드웨어 및 소프트웨어 엔지니어로 근무하고 있으며, 30가지가 넘는 다양한 프로세서와 디지털 신호 처리 장치(Digital Signal Processor, DSP)를 다룹니다. 수년 동안 저울, 공정 관리 기기, 항공기용 점화 장치, 자동판매기, 스마트카드 리더, 흡연 및 일산화탄소 모니터를 설계했습니다. 또한 Texas Instruments DSP와 Matlab의 여러 학습 자료를 집필하기도 했습니다.

14 참조 자료

Embedded Microprocessor Benchmark Consortium
https://www.eembc.org/coremark/

RISC-V 로고 https://riscv.org/RISC-V-branding-guidelines-and-materials/

RISC-V 사양 https://riscv.org/technical/specifications/

기타 코어 https://chipsalliance.org

Kendryte K210 데이터시트 https://s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_datasheet_20181011163248_en.pdf

Wishbone 버스
http://indico.ictp.it/event/a11204/session/35/contribution/22/material/0/0.pdf

https://venturebeat.com/2019/12/11/risc-v-grows-globally-as-an-alternative-to-arm-and-its-license-fees/

SweRV 코어 다운로드:
https://github.com/chipsalliance/Cores-SweRV

SweRV 프로그래머 참조 설명서
https://github.com/chipsalliance/Cores-SweRV/blob/master/docs/RISC-V_SweRV_EH1_PRM.pdf

SweRVolf:
https://github.com/chipsalliance/Cores-SweRVolf

PlatformIO:
https://github.com/platformio/platformio-core

Verilator:
https://github.com/verilator/verilator

Whisper:
https://github.com/westerndigitalcorporation/swerv-ISS

DANIEL LEÓN GONZÁLEZ, FPGA IMPLEMENTATION OF AN AD-HOC RISC-V SYSTEM-ON-CHIP FOR INDUSTRIAL IOT, Master's Degree Thesis, Universidad Complutense, Madrid, July 2020.

Digital Design & Computer Architecture, Sarah Harris & David Money Harris, second edition (RISC-V Edition due 2021)
https://www.amazon.com/Digital-Design-Computer-Architecture-Harris/dp/0123944244/ref=sr_1_1?crid=1232QZSYQ20GW&dchild=1&keywords=digital+design+and+computer+architecture+2nd+edition&qid=1597397347&sprefix=digital+design+and+%2Caps%2C219&sr=8-1

The RISC-V Reader, David Patterson & Andrew Waterman
https://www.amazon.com/RISC-V-Reader-Open-Architecture-Atlas/dp/0999249118/ref=sr_1_3?dchild=1&keywords=Patterson+RISC-V&qid=1597397389&sr=8-3