Get Up and Running Quickly

Follow this procedure to get OcPoC Zynq Mini up and running quickly. Make sure you have everything listed below to begin this tutorial.

OcPoC Zynq Mini 빠르게 실행하려면 이 절차를 따르십시오.이 튜토리얼을 시작하려면 아래에 나열된 모든 것들이 있는지 확인하십시오.

Development Environment Note:

We have documented this tutorial in a standard Ubuntu 16.04 environment

표준 Ubuntu 16.04 환경에서 이 tutorial 을 정리했습니다.

Required Hardware

You will need:

필요사항들.

- OcPoC Zynq Mini
- Micro SD card (4 GB or more)
- Micro USB to USB cable

Partition SD Card

Before we can really get started you will have to make sure your SD card is ready to go. Follow the link below for instructions on getting your SD card prepared.

진짜로 시작하기 이전에, SD 카드가 있는지 확인해야 합니다. SD 카드를 준비하는 방법은 아래 링크를 참조하시오.

Develop Your Kernel

Customize your kernel to your specific project using Xilinx's linux-xlnx

Xilinx 의 linux-xlnx 를 사용하여 특정 프로젝트에 맞게 Kernel 을 customize 한다.

Before you start

Developing your own kernel takes about 1GB of disk space. Please make sure you have enough storage before continuing.

자신의 Kernel 을 개발하는 데에는 약 1GB 의 디스크 공간이 필요합니다. 계속하기 전에 저장 용량이 충분한지 확인하십시오.

We recommend you make a directory on your computer for keeping all of the files together. Maintaining an isolated kernel development environment if you are pursing kernel customization is highly encourage. For example, start by making this separate directory:

모든 파일을 같이 보관할 수 있도록 컴퓨터에 directory 를 만드는 것을 추천합니다. 만약 kernel 을 customization 하고자 한다면, 독립된 kernel 개발환경을 유지하는 것이 좋습니다. 예를 들어, 다음과 같이 별도의 directory 로 시작하십시오.

- **Make a Kernel Dev Directory:**

```
mkdir ocpoc_kernel_dev
cd ocpoc_kernel_dev
```

- **Install Tools:**

```
sudo apt-get update
sudo apt-get install git u-boot-tools lib32ncurses5-dev
```

Download Linux-Xlnx:

We generate our kernel using linux-xlnx kernel provided by Xilinx to support Zynq Based boards. You can download this environment by cloning their git repository.

우리는 Zynq 기반 보드를 지원하기 위해 Xilinx 가 제공한 linux-xlnx 커널을 사용하여 커널을 생성합니다. 제공되는 git repository 를 cloning 하여 환경을 Download 할 수 있습니다.

- **Clone Linux-Xlnx:**

```
git clone https://github.com/Xilinx/linux-xlnx.git
```

Download OcPoC files:

If you have not downloaded the OcPoC Zynq Mini files previously you can acquire them by cloning our git repository.

이전에 OcPoC Zynq Mini 파일을 Download 하지 않았다면 git repository 를 cloning 하여 얻을 수 있습니다.

- **Clone Ocpoc Files:**

```
git clone https://github.com/aerotenna/ocpoc_mini_zynq_files.git
```

Copy OcPoC Configuration:

Now we will have to inject the OcPoC Zynq's specifics into Xilinx's linux-xlnx. Below is an example that would work if you downloaded the two git repositories in the same directory. If you are following our recommendations and keeping these files in one directory, open a terminal in that directory and execute the following command:

이제 우리는 OcPoC Zynq 의 세부 사항들을 Xilinx 의 linux-xlnx 에 넣어야 합니다. 아래의 내용은 만약 같은 directory 에 2 개의 git repositories 를 download 할 때의 작동 예제입니다. 권장 사항을 따르고 이러한 파일을 하나의 directory 에 보관하려면 해당 directory 에서 terminal 을 열고 다음 명령을 실행하십시오.

- **load ocpoc definitions:**

```
cp ocpoc_mini_zynq_files/Kernel_Config/ocpoc_defconfig linux-xlnx/arch/arm/configs
```

Setup Build Environment:

Next you will change to the Xilinx 2015.4 build, utilize the Xilinx 2016.2 i2c drivers, and load the ocpoc configuration. From the same terminal, in the same directory execute the following commands:

다음으로 Xilinx 2015.4 build 로 변경하고 Xilinx 2016.2 I2C driver 를 활용하고 ocpoc configuration 을 load 하십시오. 동일한 터미널에서 같은 directory 에 다음 명령을 실행하십시오.

- **configure kernel:**

```
cd linux-xlnx
git checkout xilinx-v2015.4
git checkout xilinx-v2016.2 -- drivers/i2c/busses/i2c-xiic.c
```

```
make ARCH=arm ocpoc_defconfig
```

You are now ready to build a kernel to suit your specific needs!

이제 특정 요구에 맞게 kernel 을 build 할 준비가 되었습니다.

Customizing the Kernel

From here you are free to test whatever your heart may desire. The majority of your default options will be made available with the menuconfig command.

여기에서 당신은 원하는 모든 것들을 시험할 수 있습니다. 대부분의 기본 옵션은 menuconfig 명령으로 사용할 수 있습니다.

- **menuconfig**

```
make ARCH=arm menuconfig
```

Beyond this, you now have the proper environment configured for installing your own custom kernel modules and device drivers. Once you are satisfied and ready to produce the kernel itself you can package everything up with the following command.

이 외에도, 사용자 정의 kernel module 과 device driver 를 설치하기에 적당한 환경을 가지게 된다. 만족하고 kernel 자체를 생성할 준비가 되면 다음 명령을 사용하여 모든 것을 패키지화 할 수 있습니다.

- **package uImage**

```
make ARCH=arm UIMAGE_LOADADDR=0x8000 uImage
```

This may take some time to complete, especially if it is your first compilation of the kernel. Once it is completed successfully, the uImage will be in 'linux-xlnx/arch/arm/boot' directory

특히, kernel 을 처음 컴파일한 경우에 완료하는데 시간이 걸릴 수 있습니다. 일단 성공적으로 완료하면, uimage 는 'linux-xlnx/arch/arm/boot directory 에 있습니다.

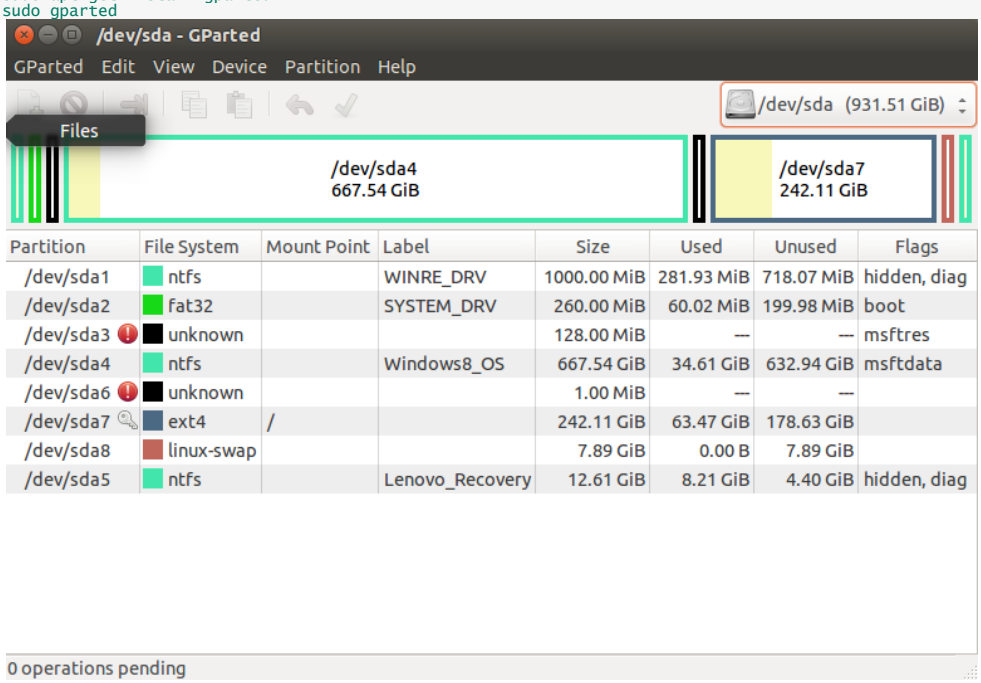
Partition SD Card

Before we begin, be sure to install 'gparted' and open it with the following commands in an Ubuntu terminal. If you're not in Ubuntu, you can use EaseUS Partition Master, or equivalent software, to partition the SD card in the same manner as the following steps.

시작하기 전에, 'gparted'를 설치하고 Ubuntu 터미널에 다음 명령을 사용하여 엽니다. 우분투가 아닌 경우, EaseUS 파티션 마스터 또는 동등한 소프트웨어를 사용하여 다음 단계와 동일한 방식으로 SD 카드를 분할하여 사용할 수 있습니다.

- **Partition Software:**

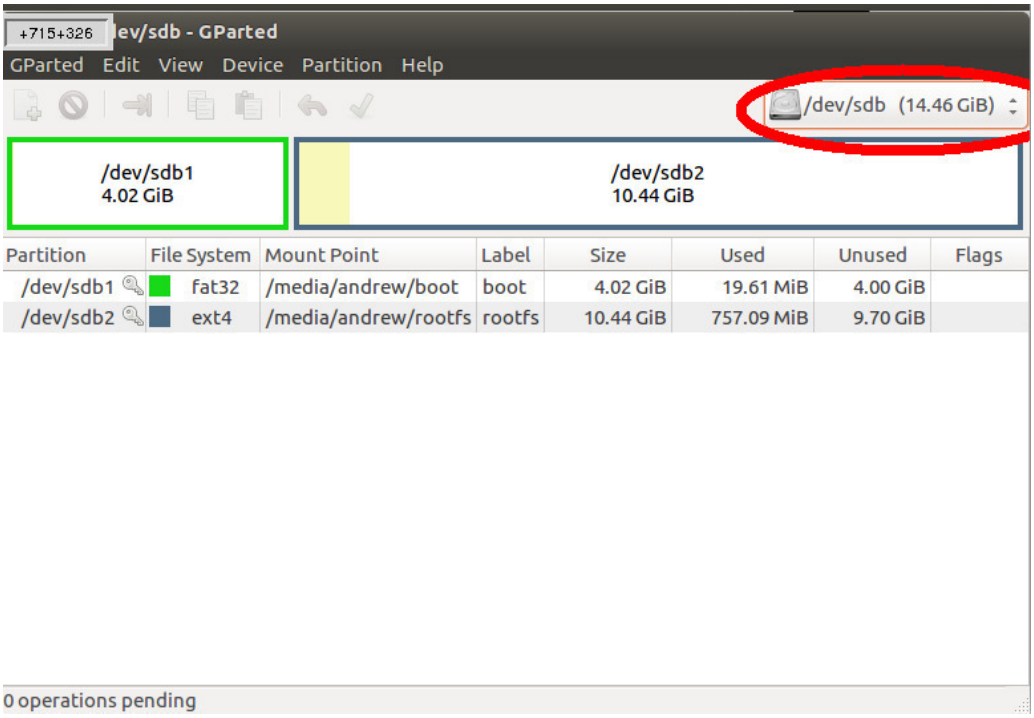
```
sudo apt-get install gparted
sudo gparted
```



GParted Disk Utility

1. First identify which partition is the SD card. Be careful to not pick the wrong one. With your SD card plugged in, you should be able to click the drop-down menu at the upper right corner and select the correct "sda" based on the size that best matches the size of your card.

먼저, SD 카드가 어던 파티션인지 확인하십시오. 잘못된 것을 선택하지 않도록 주의하십시오. SD 카드를 연결하면 오른쪽 상단에 있는 드롭다운 메뉴를 클릭하고 카드 크기와 가장 비슷한 크기의 'sda'를 선택할 수 있습니다.

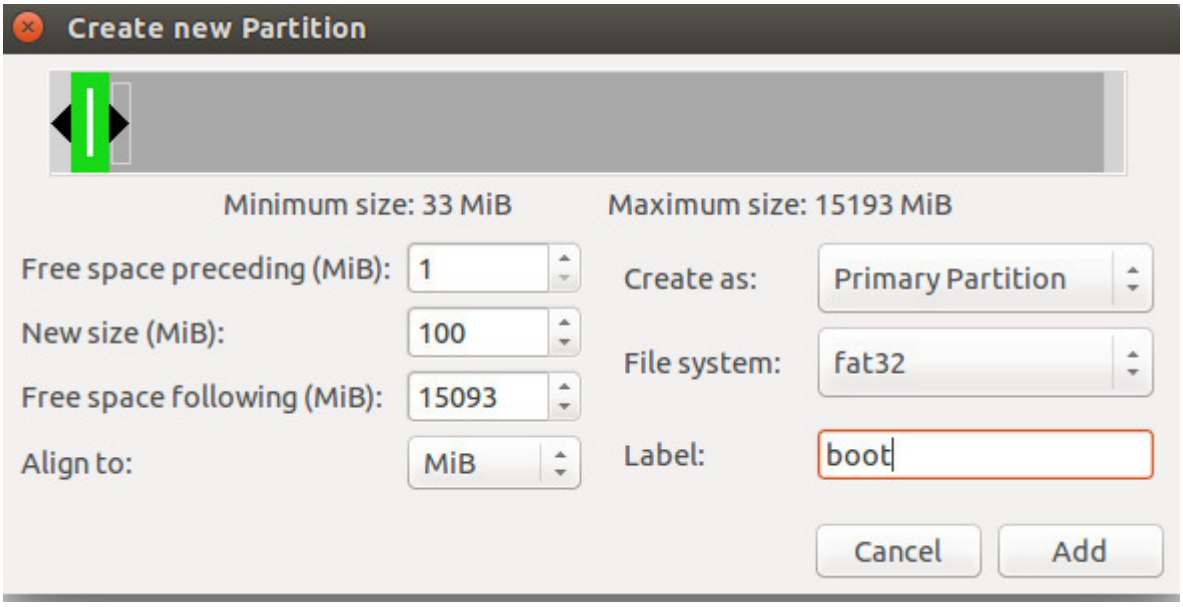


2. Clear every partition inside the SD card so it has no allocated partitions. You can simply right click each partition and select "Delete". You may have to "Unmount" a partition before deletion is enabled.

   SD 카드 안의 모든 파티션을 지우고 할당된 파티션이 없도록 하십시오. 각 파티션을 마우스 오른쪽 버튼을 클릭하고 'Delete'를 클릭하면 됩니다. 삭제가 활성화되기 전에 'Unmount' 해야 할 수 있습니다.

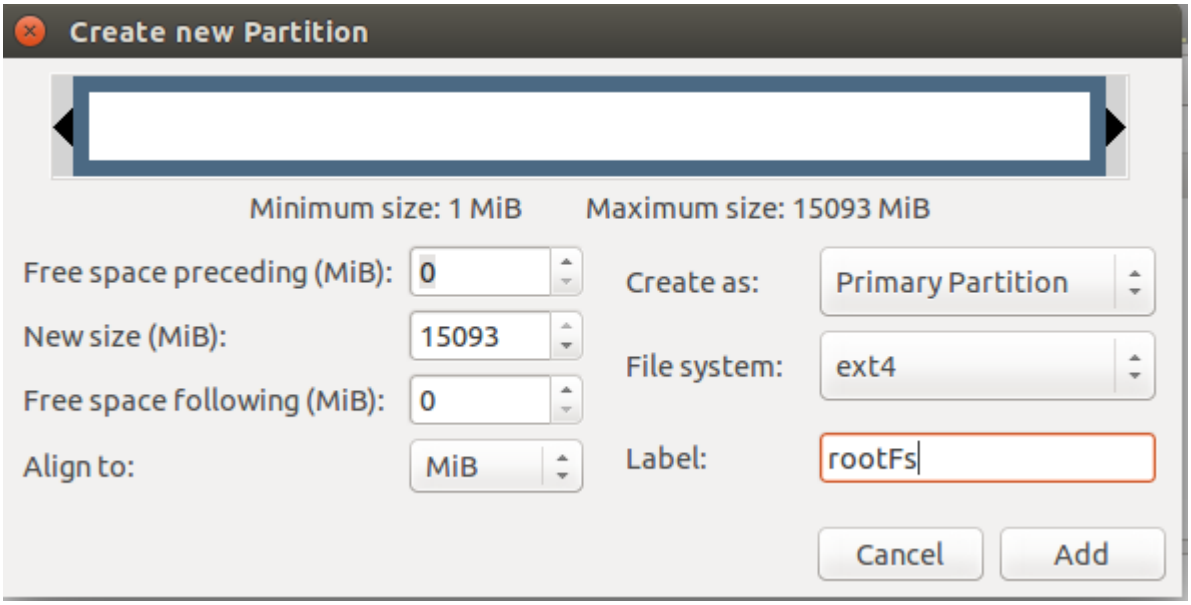
3. Create your first partition and set the following attributes. New size(MiB): '100', File system: 'fat32', and Label: 'boot'.

   첫 번째 파티션을 만들고 다음 속성을 설정하십시오. New size(MiB) : '100' File system : 'fat32' & Label : 'boot'



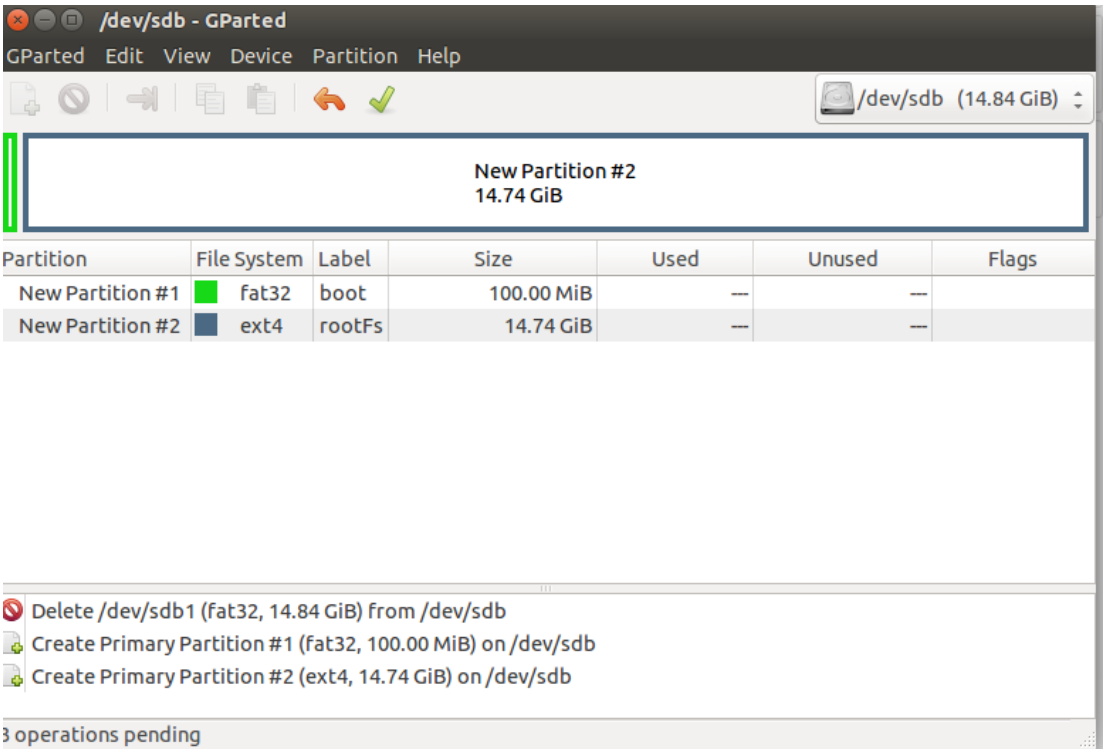
4. Create a second partition and assign the following attributes. Free space preceding(MiB): '0', Free space following(MiB): '0', File system: 'ext4', and Label: 'rootFs'.

두 번째 파티션을 작성하고 다음 속성을 지정하십시오. Free space preceding(MiB): '0', Free space following(MiB): '0', File system: 'ext4', & Label: 'rootFs'.



5. Your partitions should look similar to this. The 'Size' of rootFs will probably be different depending on the size of your SD card, this is okay. click the green check-mark and accept.

파티션이 이와 비슷하게 보입니다. rootFs 의 'Size'는 SD 카드의 크기에 따라 다를 수 있습니다. 괜찮습니다. 초록색 체크 표시를 클릭하고 승인하십시오.



Before continuing to 'Load files onto SD Card' it is best to eject, then physically remove and reinsert your SD card into your computer. This will ensure that it is detected by your system and ready for software to be installed.

‘Load files onto SD Card’를 계속하기 전에 꺼내고 SD 카드를 제거한 다음 컴퓨터에 다시 삽입하는 것이 가장 좋습니다. 이렇게 하면 시스템에서 감지되어 소프트웨어에서 설치 준비가 완료됩니다.

Choosing your Autopilot Software

The OcPoC an extremely versatile flight controller. It is compatible with several different autopilot projects. Depending on which software platform you prefer we have tailored specific documentation to assist you on getting your OcPoC up and flying as quick as possible!

OcPoC 는 매우 다재다능한 비행 컨트롤러입니다. 여러 다른 자동 조종 프로젝트와 호환됩니다. 선호하는 소프트웨어 플랫폼에 따라 OcPoC 를 최대한 빨리 사용하기 위한 구체적인 문서를 제공합니다.

Currently the OcPoC has been tested and provides support for loading 'Ardupilot - Arducopter' and the 'PX4' autopilot flight stacks.

현재 OcPoC 는 테스트를 마쳤으며 ‘Arducoptor’ 및 ‘PX4’ 자동 조종 비행 스택을 Load 할 수 있도록 지원합니다.’