How to build Raspberry pi zero w BUILDROOT image

Lately I was reading about embedded linux and came to know about two such custom embedded linux system build sytem the Yocto project and Buildroot.I wanted to make my own custom linux for raspberyy pi zero W i had in my *ahem* attic.

Requirements:

1. Raspberry pi zero w (ofcourse)
2. A PC (with ubuntu 18.04 or higher)
3. microSD card
4. microSD card reader

Step 1: Preparing and downloading BUILDROOT

Open Ubuntu terminal (Ctrl+ALT+T) and type the below

```
$ wget https://www.buildroot.org/downloads/buildroot-
2019.08.tar.gz
$ tar xvjf buildroot-2019.08.tar.gz
$ cd buildroot-2019.08
```

Building

Buildroot is now ready for initial configuration. There are few commands that can help:

```
$ make help
$ make list-defconfigs
```

output:

```
  raspberrypi0_defconfig             - Build for
raspberrypi0
  raspberrypi0w_defconfig            - Build for
raspberrypi0w
  raspberrypi2_defconfig             - Build for
raspberrypi2
  raspberrypi3_64_defconfig          - Build for
raspberrypi3_64
  raspberrypi3_defconfig             - Build for
raspberrypi3
  raspberrypi3_qt5we_defconfig       - Build for
raspberrypi3_qt5we
```

```
  raspberrypi4_defconfig                - Build for
raspberrypi4
```

REAL FUN STARTS!

Type inside terminal

```
$ make raspberrypi0w_defconfig
$ make all
```

If there was no errors then continue editing the image to add wifi, bash , ssh, and whatever you need for your project.

```
$ make menuconfig
```

A small GUI pops up (should be in maximized terminal window), go through each sections . Use Y key to enable N to remove , press escape twice to go back or use exit option near the select option .

1. Target options —> leave it default
2. Build options —> select enable compiler cache
3. Toolchain —> Enable wchar support
4. System config → change system hostname, system banner, root password , enable install timezone info
5. Hardware Handling

```
1 Hardware Handling -> Firmware -> rpi-wifi-firmware
```

6. Network applications

```
Networking applications -> wpa_supplicant
Networking applications -> wpa_supplicant - Enable 80211
support
Networking applications -> dropbear
Networking applications -> openssh
```

7. Target Packages → Shell and utilities

```
Target Packages -> Shell and utilities ->bash
```

Also under System configuration → under root password change shell to bash, also run getty login prompt after boot.

Finally

type the below code (fingers crossed!)

```
make all
```

once you get no error in output , output image files will be under

```
buildroot/output/images/
```

sdcardimage.img will be your image to burn to sdcard for Raspberry pi zero w.

Enabling Wi-Fi

In this subsection, we enable the Wi-Fi interface of the Raspberry Pi Zero W, so it will be able connect to any Wi-Fi networks.

wpa_supplicant

Create a file, named `interfaces` in `buildroot/board/raspberrypi/` (all the other raspberrypi* are symlinks to this folder). The auto wlan0 will make sure that wlan0 is started when `ifup -a` is run, wich is done by the init scripts.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
    pre-up /etc/network/nfs_check
    wait-delay 15

auto wlan0
iface wlan0 inet dhcp
    pre-up wpa_supplicant -D nl80211 -i wlan0 -c /etc/wpa_supp
    post-down killall -q wpa_supplicant
    wait-delay 15

iface default inet dhcp
```

Create another file, named `wpa_supplicant.conf` with `wpa_passphrase` in `buildroot/board/raspberrypi/` (all the other raspberrypi* are symlinks to this folder). It should look like something like this:

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1

network={
   ssid="EDIT_THIS"
   psk="EDIT_THIS"
}
```

ALSO

post-build.sh

The hotplug helper must be set as mdev and write /etc/mdev.conf file. The mdev package itself has some helper script for this and can be used directly. Also the above created files must be copied, so **add the following lines** to `buildroot/board/raspberrypi/post-build.sh`:

```
cp package/busybox/S10mdev ${TARGET_DIR}/etc/init.d/S10mdev
chmod 755 ${TARGET_DIR}/etc/init.d/S10mdev
cp package/busybox/mdev.conf ${TARGET_DIR}/etc/mdev.conf

cp board/raspberrypi/interfaces ${TARGET_DIR}/etc/network/inter
cp board/raspberrypi/wpa_supplicant.conf ${TARGET_DIR}/etc/wpa_
cp board/raspberrypi/sshd_config ${TARGET_DIR}/etc/ssh/sshd_con
```

sshd config file

open/mount your sdimage.img file previously generated and copy /**etc/ssh/sshd_config** to `buildroot/board/raspberrypi/` and add

```
PermitRootLogin yes
PermitEmptyPassword yes
```

One last ride

finally do

```
$ make all
```

you should have a fully working raspberry pi zero w linux custom image with ssh and wifi also you can add anything to this base build like wiringpi, gpio library, python etc even qt and xorg , keyboard support , mouse etc can be added.

CREDITS:

https://ltekieli.com/buildroot-with-raspberry-pi-what-where-and-how/

https://unix.stackexchange.com/questions/205271/sshd-not-starting-after-boot-on-embedded-linux-built-with-buildroot

https://blog.crysys.hu/2018/06/enabling-wifi-and-converting-the-raspberry-pi-into-a-wifi-ap/

https://unix.stackexchange.com/questions/396151/buildroot-zero-w-wireless

amrith October 1, 2019 linux, Raspberry pi

BUILDROOT, raspberry pi zero w, ssh, wifi

8 responses to “How to build Raspberry pi zero w BUILDROOT image”

Raspberry Pi HD webcam – Maglazana
December 4, 2020 at 4:54 pm

[…] step 1, I followed the tutorial on ARM fever to have to build a baseline image. Except having to tweak some wireless issues so I could remotely […]

★ Liked by 1 person

Reply

@mББосkег
January 2, 2022 at 10:02 am

Thank you.

★ Like

Reply

Artem
January 10, 2021 at 5:57 pm

Interesting and helpful article. Have you experimenting trying to improve boot times while keeping wifi connectivity? If so, what's the boot time you've achieved on the PI Zero W?

★ Like

Reply

@mBBocKer
January 2, 2022 at 9:59 am

unfortunately no. I was just trying to learn Buildroot and i hat a pi zero w with me . I had to send a day or two to get it working. I just made a proper step by step documentation so somebody else can benefit

★ Like

Reply

Jon Cedarleaf
February 10, 2022 at 1:26 am

I was able to achieve a boot time of 28 seconds with wifi connectivity, on the Raspberry Pi Zero W, using the above method. It could probably go faster, but I have not tried to optimize this. I did notice that at about 9 seconds in, it waits for the eth0 connection to complete before continuing the bootup process, so if it is possible to boot without waiting for a connection it may be faster.

★ Like

Reply

@mBBocKer
May 11, 2022 at 12:50 pm

Yes. You could disble ethernet also other services that are unnecessary .

★ Like

Jon Cedarleaf
February 10, 2022 at 1:23 am

Very helpful article, I used this to get my networking up and running on Buildroot!
I've got a question for you, is it possible to back up all this customization to a private github account? I want to save all the progress I've made with Buildroot, the packages added, new files (like wpa_supplicant.conf), etc. I'm hesitant to make a branch of the whole Buildroot project, as it is so large (something like 7 Gb).

★ Like

Reply

@mBBocKer
May 11, 2022 at 12:54 pm

the makefiles and defconfigs are already premade so those can be backep up / edited for your customisation.
Yes the build is huge because it downloads all the necessary source code and compiles it from the source.
Just go through buildroot documentation and check what files are used for making pi zero and backup only those.
You can also create a bash script to use these files and make a new build.
Try cloud platforms like aws or google collab to help build and improve build speed

★ Like

Reply

Leave a Reply

Enter your comment here...

EMBEDDED SYSTEM ROBOTICS NOTES

My blog about my journey in embedded systems and robotics

FIND ME ON SOCIAL MEDIA