IAR Journal of Engineering and Technology
ISSN Print : 2708-5155 | **ISSN Online :** 2708-5163
Frequency : Bi-Monthly
Language : English
Origin : Kenya
Website : https://www.iarconsortium.org/journal-info/IARJET

OPEN ACCESS JOURNALS

Research Article

Building an Embedded Linux System for Raspberry Pi

Article History

Received: 09.03.2021
Revision: 24.03.2021
Accepted: 23.03.2021
Published: 30.03.2021

Author Details

Alabi, A.A[*1], Ojoawo, A.O[2], Tade, A.A[3], Kolawole, I.G[3], Adekanmbi, A.O[3], Fawole, T.G[4] and Abiola, B.E[5]

Authors Affiliations

[1]Department of Science Laboratory Technology (Physics Unit), Adeseun Ogundoyin Polytechnic, Eruwa, Oyo State, Nigeria

[2]Department of Computer Science , Adeseun Ogundoyin Polytechnic, Eruwa, , Oyo State, Nigeria

[3]Department of Electrical/Electronic Engineering, Yaba College of Technology, Lagos State State, Nigeria

[4]Department of Civil Engineering, Adeseun Ogundoyin Polytechnic, Eruwa, , Oyo State, Nigeria

[5]Augustine University, Epe, Lagos State, Nigeria

Corresponding Author*
Alabi, A.A

How to Cite the Article:

Alabi, A.A., Ojoawo, A.O., Tade, A.A., Kolawole, I.G., Adekanmbi, A.O., Fawole, T.G. & Abiola, B.E . (2021). Building an Embedded Linux System for Raspberry Pi. *IAR J Eng Tech*; *2*(2),32-35.

Abstract: This project work was carried out using LINUX to build an operating system for Raspberry pi which is a small sized computer system with a lot of benefits such as: strong processing power in a compact board, several interfaces (HDMI, multiple USB, ONBOARD Wi-Fi and Bluetooth), support for many languages (e.g. LINUX and python) among others.

Keywords: LINUX and python, Wi-Fi and Bluetooth.

INTRODUCTION

Raspberry pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and is capable of using a standard keyboard and mouse. It enables people that are zealous in computing to explore how to use different languages to develop operating system and application programs for it.

It has very limited resources, so development and compilation of SDK on it will be very slow. For this reason cross-compilation was implored in this coursework to rescue the situation. Buildroot was the compilation tool that was implored to build the BSQUASK SDK development environment for Raspberry Pi.

For any application to run on any system, it must be compiled. This is done by the use of a compiler. However, a compiler is not sufficient, but can only work in conjunction with some other tools (such as: libraries, an assembler, a linker, with some other tools). The compiler with all these tools are called a development tool chain.

When using the system compiler, and other development machines that are of the same architecture and operating system are used so that generated object files are not used on another platform, the development tools used are called **native toolchain**.

But in a situation where an executable code generated can be run on a platform that is currently not the resident platform for the compiler, a **cross-platform development tool chain** is needed.

Like the case of embedded system which is the target system with a processor that is different from the processor of a host system, a cross-compilation toolchain that runs on the host system but generates codes for target processor (or system) is needed.

Buildroot is a type of tool that can make use of a compilation toolchain in a particular host system to build a cross-compilation toolchain and other tools that can run on the development host. This means buildroot can be used in this type of situation.

Buildroot is a tool that consists of a set of makefiles and patches that make it easy to generate a complete Linux system. It is able to generate any or all of cross-compilation toolchain, a root file system, a kernel image and a bootloader image.

Buildroot uses existing cross-compilation toolchain or provides different solutions to build:

- The internal tool toolchain backend(called Buildroottoolchain in the configuration interface)
- The external toolchain backend (called external toolchain in the configuration interface)
- The crosstool-NGtoolchain backend (crosstool-NGtoolchain in the configuration interface).

Buildroot needs some software to be already installed on the host system before cross-compilation can take place. They are called mandatory and optional packages.

- **Mandatory packages**
  a) build tools =>e.g which, sed, make, binutils, build-essential, gcc, g++, bash, patch, gzip, tar, bzip2, perl, cpio, python,unzip, rsync.
  b) sourcefectching tools =>e.gwget

- **Optional packages**
  (a) source fectching tools=>e.g bazaar, git, cvs, mercurial, rsync, subversion, etc
  (b) Configuration interfave dependencies=>e.gncurses, glib2, gtk2, glade2
  (c) Java-related packages=>e.g jar tool, java compiler
  (d) Documentation orientation tools=>e.gasciidoc

IMPORTANCE OF CROSS-PLATFORM DEVELOPMENT TOOL CHAIN

i. Speed=> target platforms are usually much slower than host platforms
ii. Capability=> compiling is resource intensive. Because target platform might not have enough memory it makes use of the host memory.
iii. Convenience => cross-compilatin is necessary where native compilation is not convenient or practiceable.
iv. Flexibility=> it gives room for interrelated execution of components on a system.
v. Availability=> since cross-compiled programs are meant for many targeted systems, many systems are available for inter-usability.
vi. Reduction in overall cost=> since cross-compiled programs are meant for many targeted systems, overall cost effectiveness on production is reduced.

METHODOLOGY

Buildroot is a set of makefiles and patches that makes it easy to generate a complete embedded linux system.

Ubuntu as an easy-to-use desktop Linux distribution was downloaded and installed on the laptop. The laptop was connected to the internet before the task commenced.

Terminal of Ubuntu operating system was entered into and a local Code directory was created for raspberryPi-Buildroot to be cloned into. This was done with the command:

"mkdir Code"

And the RaspberryPi-BuildRoot package was cloned into the created directory using the command below:

"git clone git://github.com/nezticle/RaspberryPi-BuildRoot.git BuildRoot"

A directory was also created for the SDK to be built into. The command below made SDK available from the buildroot:

"export BSQUASK_DIR=/opt/bsquask"

And another directory was created to build SDK into with the following command:

"mkdir -p $BSQUASK_DIR"

BuildRootdirectory was then changed to using the folloeing command:

"cdBuildRoot"

In this directory, a Makefile was generated for the SDK using the command below:
"make raspberrypi_defconfig 0=$BSQUASK_DIR"

Some build dependency packages had been earlier installed for the makefile to be generated using the following command:

"sudo apt-get install build-essential, flex, bison,tex, git, cvs, subversion, unzip, bc, whois, ncurses-dev, mercurial, textinfo, git-core, gettext, gparted, mesa-common-dev, libgl1-mesa-dev"

This 'make' process took about seven minutes to complete. After this 'make' stage, SDK directory was changed to with the command below:

"cd $BSQUASK_DIR"

And SDK was built using the command given below:

"make"

This process took three hours forty five minutes for the image to be built.

After this stage, embedded Linux operating system has been built and the generated image was now installed on an SD card with the procedure described below:

Using the Generated Image on the raspberry Pi

SD card was inserted into the card reader of the system on which operating system was being built and in the 'gparted' environment in the Ubuntu O.S. it was formatted and put in the following partitions:

75MB fat32 partition
3.6GB ext4 partition

After this, I entered into the directory of the images generated with the use of the command:

"cd /opt/bsquask/images"

The two partitions created were: /dev/mmcblk0p1 and /dev/mmcblk0p1 respectively. These two partitions were mounted using the command below:

"sudo mount /dev/mmcblk0p1 /media/BOOT"
"sudo mount /dev/mmcblk0p2 /media/rootfs"

Then the following commands were used to install the rootfs on the SD card:

"sudoumount /dev/mmcblk0p1"
"sudoumount/dev/mmcblk0p2"

The SD card was then removed from the card reader and inserted into the raspberry pi.
The raspberrypi was then ready for use with its own embedded Linux operating system.

The folders of ~/Buildroot/ have the following subfolders and files as shown below:

Folder	subfolders		Files	
Buildroot	i.	arch	i.	CHANGES
	ii.	board	ii.	Makefile
	iii.	configs	iii.	Makefile.legacy
	iv.	docs	iv.	README.md
	v.	package	v.	COPYING
	vi.	system	vi.	config.in
	vii.	support		
	viii.	boot		
	ix.	linux		
	x.	output		
	xi.	fs		
	xii.	toolchain		

Testing the raspberry pi with the embedded Linux operating system built
Apparatus:

i. Raspberry pi (with power cord)
ii. SD card
iii. VDU (monitor)
iv. Mouse(USB type)
v. Keyboard(USB type)
vi. VGA cord
vii. piView (HDMI to VGA converter)

All the apparatus mentioned above were connected together appropriately and the raspberry pi was powered on.

The figure1 below shows how piView, SD card, and power cord were connected to the raspberry pi. The piView, as shown below was connected to the VGA cord and the VGA cord conncted to the monitor. Other ports are still available to connect other devices to. Examples are the mouse and keyboard which were connected to the USB ports.

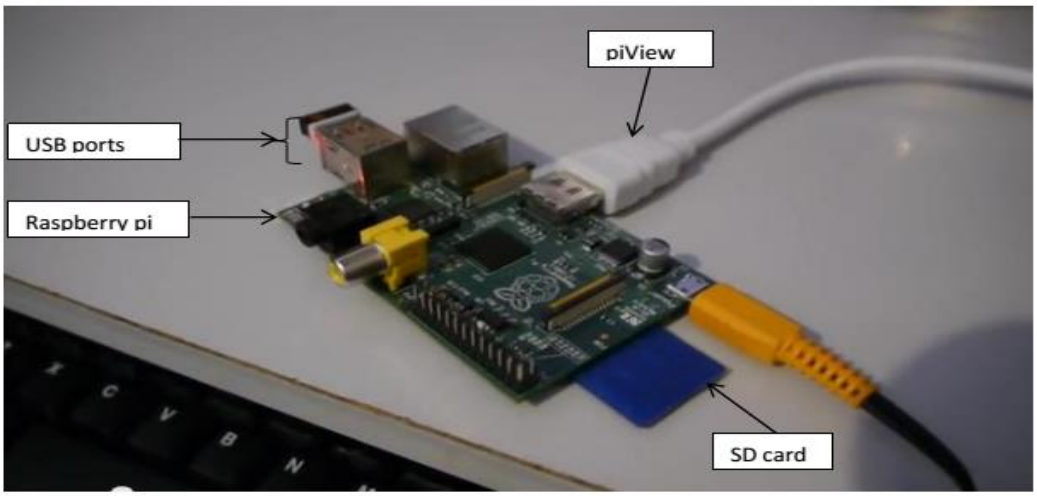

Figure1: Connection of Raspberry pi to the system on which operating system was built

The figure2 below was the image shown on the monitor. "root" is the raspberry login and also the password.

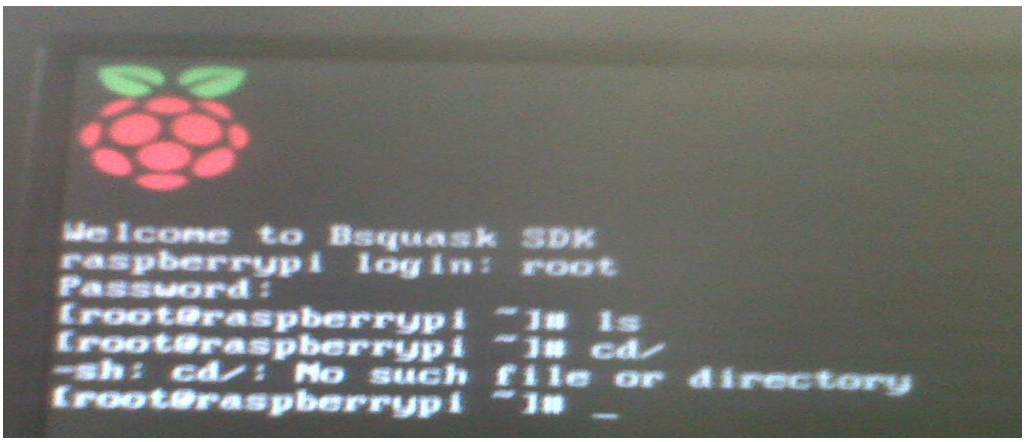

Figure2: image shown on the monitor as displayed by the Raspberry pi with a new operating system

REFERENCES

1. Buildroot. (2013). The Buildroot user manual [online] available from http://buildroot.uclibc.org/downloads/manual/manual.pdf [29 October 2013]
2. Ibanez, L., & Maclean, A. (2013). Cross-Compiling for Raspberry Pi [online] available from http://www.kitware.com/blog/home/post/426 [29 October 2013]
3. Nichols, A. (2013). Bsquask SDK [online] available from http://www.bsquask.com/blog/2012/12/21/bsquask-sdk/ [23 October 2013]
4. Raspberry, Pi. (2013). [online] available from http://www.raspberrypi.org/[23 October 2013]
5. Robert, S. (2013). How to create a buildroot environment for Raspberry Pi. [online] available from http://www.xappsoftware.com/wordpress/2013/06/06/how-to-create-a-buildroot-environment-for-raspberry-pi/ [2013]