Can a software engineer create an embedded system?

I want to create an embedded system using Linux similar to E book reader using ARM9 processor. I am not an electronics expert but I would love to learn it. I know basics of electronics like transistors, flip-flops, multiplexers. etc. I love software and would like to create something like an E book reader. Is it possible for a software engineer to create an embedded system? I do not want to buy single board computer available in market, I want to create it myself.

- Where do I get some kind of tutorial?
- Is my knowledge of operating systems enough to create such a system?

embedded-linux

edited May 28 '12 at 10:33

Jens
42.9k 10 69 115

asked Feb 2 '10 at 9:59

Sphinx
131 4 9

2 You haven't said *anything* about your "knowledge of operating systems". – pavium Feb 2 '10 at 10:06

I know basic working of each operating system... So I can create device drivers for Linux. – Sphinx Feb 2 '10 at 10:15

You will probably get device drivers from the platform vendors. – Hassan Syed Feb 2 '10 at 10:20

Device drivers are the least of your worries... see my answer. I've actually done something like this (different device)... it's hard and extremely expensive to get to a product. Given a few million, it can be done though. – Andrew McGregor Feb 2 '10 at 10:27

5 Answers

Building a system requires knowledge from multiple engineering disciplines. You can only achieve such a task by buying off-the-shelf modular components and assemble them, and in the case of an e-book putting together the modular components won't be pleasant.

Also learning any of the single disciplines needed will take you a long and concentrated effort.

To (loosely) indicate the problem areas:

1. you need a computing platform of the right form-factor with all the right chipsets (Apple integrate their own single CPU, as of recently, using hardware designs from multiple companies). You will not find a suitable computing platform of the right form-factor. (Electronic Engineer: Digitial designer, Analog Designer)
2. You need to try to attach an LCD to the right platform, and other peripherals such as USB/ charging port/ WIFI etc etc. (Electronic Engineer, Product Designer)
3. You need to build a case for the platform. (Product Designer)
4. You need to get a embedded operating system (potentially real-time) (working on your platform) that fits your needs. (Embedded programmer, Kernel Programmer)
5. You need to extend said operating system to behave the way you want it. (Application Programmer, Graphics Programmer)

The most important part is the platform, and getting a suitable one is very hard and very expensive. The original iphone had a platform created by a third party that apple bought and used to apply points 2-5 -- and it still took their best engineers a long time to make a prototype.

edited Feb 2 '10 at 10:58

answered Feb 2 '10 at 10:04

Hassan Syed
12.8k 6 55 132

3 +1 You could build your own E Book reader, but by the time you finished they'll be handing them out for free. – Dead account Feb 2 '10 at 10:06

You forgot building a gcc toolchain for the computing platform. – mouviciel Feb 2 '10 at 10:24

@mouviciel added (working on your platform) to point 4. – Hassan Syed Feb 2 '10 at 10:29

Not really; hardware engineering is a degree-level subject in it's own right, and you need at least three different specialities to do that job. Not to mention that CAD software and CNC machines cost a heck of a lot more than gcc, so hardware engineers' overheads are huge.

However, you can hire that done, for a substantial fee. Or you can use embedded boards and get the case design done for you.

For example, a beagleboard with these accessories in a custom case.

Or, a Gumstix overo with one of these and one of these in a custom case.

In either case, running some embedded linux.

Development boards save a lot of time and money, but in both cases, if you have the capital you can get those boards boiled down into a custom board that will do just what you need for your application, and cost less in large numbers.

Do not underestimate the case design; you're looking at the thick end of a hundred thousand dollars just for the tooling to manufacture a plastic, die-cast metal or stamped metal case, without paying for the design work.

answered Feb 2 '10 at 10:26

Andrew McGregor
12.4k 2 20 26

Creating embedded hardware from scratch requires a lot of expertise and resources. It would be better to start off with a low-cost evaluation board in order to learn the basics of embedded programming and interfacing first. That should keep you busy for a few months. Beyond that, embedded CPU suppliers typically have reference designs that you can incorporate into your own embedded product, but at this point you will need to start investing a lot of time, effort and money into tooling up for hardware design and development.

edited Feb 2 '10 at 10:42

answered Feb 2 '10 at 10:04

Paul R
160k 18 257 401

There is basically no need to create (I mean to solder) the embedded system. A good approach may be to buy some controller board like this this or this. You need to be careful with the board but there is nothing about it a software engineer could not manage; it has the familiar serial, USB and RJ45 ports and normally already boots Linux. Finding enclosure, connecting peripherials (including analog/digital converters, or adding some relays to the output ports) is fully in the range of capabilities of someone who wants also some work with hardware. Expect to develop in C.

answered Feb 16 '13 at 9:50

h22
11.8k 7 36 62

You can buy off the shelf hardware for embedded software development.

PC 104 Boards

answered Feb 2 '10 at 10:09

PeanutPower

PC 104 is obsolete, and they're a curse from a reliability point of view... those stacking connectors work loose if there is any vibration, even if you bolt the boards together. – Andrew McGregor Feb 2 '10 at 11:31

@Andrew so did you have a recommendation? – PeanutPower Feb 2 '10 at 13:14

2 Depends on what you want. Routerboard and Ubiquiti make nice routers. The Beagleboard is nice for media players and things with UI, Gumstix are nice if the Beagleboard is too big. Arduinos are good if you can get away with a 20 MHz AVR. For the bigger stuff, there are lots of nice Atom-based boards from people like Supermicro and Advantech. – Andrew McGregor Feb 2 '10 at 21:40

@Andrew thanks :) – PeanutPower Feb 3 '10 at 0:00