



Lộ trình trở thành Embedded Developer từ Zero tới Hero

Author: Truong Giang

Lập trình nhúng là gì ?

Nói đến lập trình chúng ta nghĩ ngay đến CNTT (IT). Chuyên ngành mà chúng ta có thể có thể tạo ra những tựa game hay, những phần mềm đỉnh cao, các trang web lớn, công nghệ AI, Blockchain ... đủ mọi thứ trong tưởng tượng. Còn với lập trình nhúng, các bạn sẽ đi lập trình các chương trình chạy cho những con chip, các dòng Vi điều khiển, Vi xử lý. Đối với CNTT chúng ta có vô vàn các loại ngôn ngữ bậc cao để lập trình. Còn đối với lập trình nhúng, ngôn ngữ được sử dụng nhiều nhất lại là ngôn ngữ C. Vậy thì lập trình nhúng có gì thú vị ?

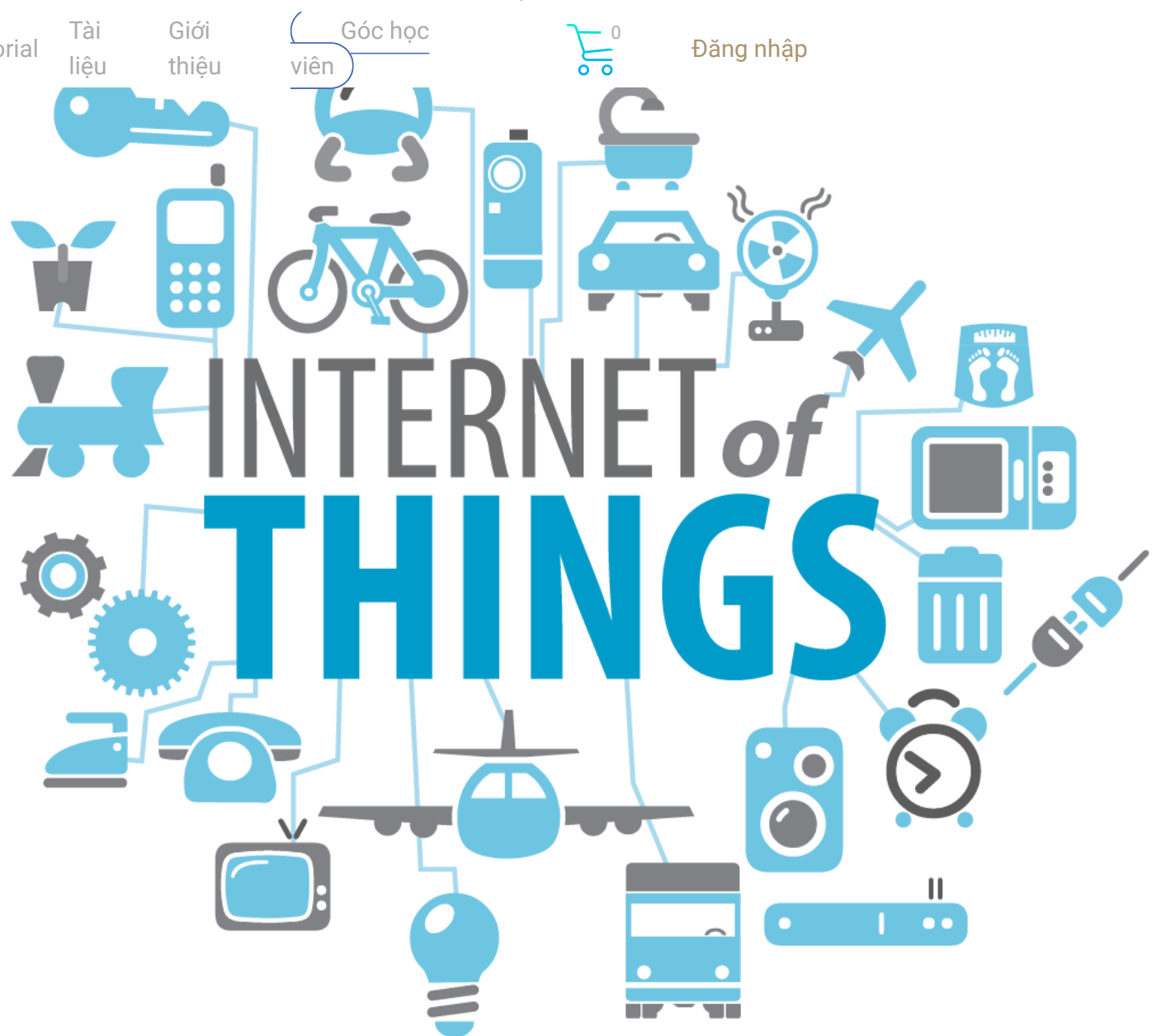
Nếu như các bạn từng xem các cuộc thi Robotcon hoạt động và thi đấu, thì lập trình nhúng sẽ giúp các bạn có thể tạo ra những cỗ máy tương tự như vậy. Nó có quy mô từ đơn giản nhất (đồ án sinh viên) đến phức tạp (máy móc trong công nghiệp). Các bạn sẽ vừa cần hiểu được nguyên lý của phần cứng để thiết kế ra các mạch điện tử, vừa phải biết lập trình cho các mạch điện đó chạy sao cho đúng mục đích. Nghe cũng vẻ phức tạp đấy nhỉ.

IoT là gì ?

IoT (Internet of Things) vạn vật kết nối internet. Khái niệm này đã xuất hiện từ lâu và đang trở nên cực kì phổ biến trong các năm gần đây. Vì sao vạn vật lại cần kết nối Internet ? Bởi vì chúng ta không thể đứng kè kè để điều khiển các thiết bị các cỗ máy mà chúng ta đã mất công tạo ra. Nhu cầu giám sát và điều khiển từ xa (hoặc tự động) nảy sinh như một phần tất yếu. Suy cho cùng con người làm việc vất vả cũng là để hưởng thụ về sau. Và máy móc cần phải thay con người làm việc. Vì vậy chúng ta cần kết nối tất cả các thiết bị với Internet để có thể điều khiển giám sát chúng từ một khoảng cách nào đó bất cứ đâu, bất cứ khi nào. Nghe đến đây thì IoT có vẻ khá là tương lai đấy nhờ.

Hiện nay có thể kể đến rất nhiều mảng lớn của IoT như:

1. **Smart Home** (ngôi nhà thông minh): các thiết bị thông minh trong ngôi nhà, các hệ thống giám sát an toàn, hệ thống báo cháy quản lý sự cố, các cảm biến đo thông số môi trường sống
2. **Smart Farm** (nông nghiệp thông minh): hệ thống tưới tiêu, quản lý môi trường sống thực động vật, kích thích tăng trưởng, hệ thống nhận diện chất lượng sản phẩm ...
3. **Smart City** (thành phố thông minh): các hệ thống quản lý khu đỗ xe, quản lý đèn đường, thông tin trong thành phố, khu đô thị ...
4. **Ngành công nghiệp ô tô**: cơ sở hạ tầng giao thông, hệ thống bảo dưỡng ô tô, công nghệ xe tự hành, hệ thống giám sát tài xế, hệ thống thông tin giải trí trên ô tô, kết nối với hệ thống quản lý nhà...
5. **Smart Factory** (nhà máy thông minh): Cơ sở sản xuất được số hóa và kết nối dựa trên phương thức sản xuất thông minh. Là sự kết hợp giữa phần mềm ứng dụng với hệ thống máy móc, thiết bị được kết nối Internet. Dữ liệu được tổng hợp và phân tích bằng trí tuệ nhân tạo (AI).



So sánh lập trình nhúng và IT ?

Một cách công bằng mà nói, mặt bằng thu nhập của lập trình nhúng sẽ không thể nào bằng với bên IT được. Thường sẽ chênh nhau cỡ 1.5 lần. Kể cả về số lượng Jobs và Outsource thì IT vẫn chiếm ưu thế. Vì vậy chắc chắn bạn sẽ bắt đầu phân vân khi đọc đến đây.

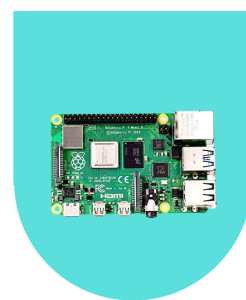
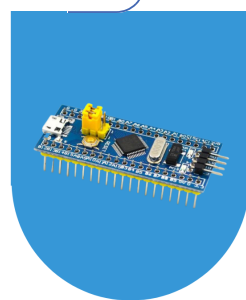
Đam mê, để mà theo được lâu dài ngành này chắc chắn bạn cần có một đam mê thật lớn. Vì việc bắt đầu với lập trình nhúng khó hơn nhiều so với IT. Bạn sẽ phải có kiến thức về cả phần cứng, đọc được mạch, sơ đồ nguyên lý rồi lập trình cho con chip trên mạch. Chưa kể việc lập trình cho chip khó hơn nhiều so với viết một phần mềm đã có rất nhiều tutorial trên mạng. Kể cả việc tìm tài liệu tiếng việt ngành này gần như cực ít.

Vậy thì lợi thế nào cho lập trình nhúng ?

- Đầu tiên là dễ làm sản phẩm. Các sản phẩm của lập trình nhúng thường không quá phức tạp. Đương nhiên sẽ có sản phẩm phức tạp có sản phẩm không, nhưng chung quy chúng ta đều có thể làm ra được các sản phẩm phù hợp với nhiều mức độ năng lực. Mình từng biết rất nhiều bạn bằng tuổi kém tuổi mình có thể làm và thương mại được các sản phẩm tương đối đơn giản nhưng hiệu quả cao. Có thể kể như các sản phẩm về led trang trí, led pháo hoa, máy rửa tay khử khuẩn, máy rót rượu...
- Dễ dàng chuyển ngành. Bởi vì đặc thù công việc phải lập trình nên các bạn theo lập trình nhúng cũng có thể dễ dàng chuyển ngành sang IT nếu thích. Thời gian để chuyển ngành có thể rơi vào từ 6 tháng đến 1 năm. Việc có sẵn tư duy lập trình nhúng sẽ giúp các bạn hiểu sâu hơn về ngôn ngữ lập trình.
- Sản xuất trong lập trình nhúng là một mảng mồi khá béo bở. Cái này mình nhìn ra xung quanh những người mà mình quen biết. Họ khi nắm được công nghệ sản xuất thường sẽ có tư duy ra làm chủ.
- Công nghệ ít thay đổi. Theo thời gian kinh nghiệm của bạn tăng lên và bạn sẽ được đánh giá cao trong ngành của mình.
- Được làm thứ mà bạn yêu thích. Mình từng hỏi 1 người em của mình là liệu rằng nó có chuyển sang ngành IT hay không. Và câu trả lời là không. Bởi vì đơn giản là thích làm lập trình nhúng. Thích được làm mạch và làm ra sản phẩm của riêng mình. Nói chung làm nhúng rồi thì sẽ có một thứ sức hút vô cùng lớn.

Lộ trình học lập trình nhúng và IoT từ Zero

Nói lan man thì cuối cùng cũng phải chỉ ra được một cái lộ trình cụ thể.



LỘ TRÌNH TRỞ THÀNH EMBEDDED DEVELOPER TỪ ZERO TỚI HERO

Mình tạm thời chia hành trình của chúng ta thành 3 giai đoạn.

Giai đoạn 1: Khởi đầu

Đây là giai đoạn gần như chưa biết gì. Chúng ta cần xây dựng một cái móng thật chắc chắn đã. Bạn cũng không cần phải quá sợ hãi khi mình đang ở trong giai đoạn này đâu. Vì theo mình biết thì lập trình nhúng không đòi hỏi phải quá thông minh đâu, chỉ cần chăm chỉ là được.

Đầu tiên, các bạn sẽ học thật chắc cho mình ngôn ngữ C (đặc biệt là kiến thức về chuỗi, mảng, con trỏ và struct dùng rất nhiều). Sau đó thì đá qua học C++. Học chắc 2 ông này là có thể đi xin lên Lab hoặc xin thực tập ở FPT được rồi. Cái sai của nhiều bạn đó chính là không học chắc ngôn ngữ lập trình đã vội vã đi vào lập trình chip. Sẽ rất khó khăn và rồi lại phải quay lại học từ đầu hoặc học mãi mà không thấy trình độ tăng lên. Thì xin thưa là không học chắc nền tảng chính là nguyên nhân đó ạ.

Tiếp theo thì các bạn nên xem qua chuỗi video cực nổi tiếng về dòng 8051 của anh Nguyễn Thanh Dâng ([tại đây](#)). Đây là seri khá quan trọng mà hầu hết ai học nhúng cũng từng xem qua. Nó sẽ giúp bạn có cái nhìn tổng quát về lập trình nhúng sẽ làm được cái gì. Một số các khái niệm cơ bản về điện tử đơn giản như cách lắp led, vẽ mạch, lập trình gpio, interrupt, timer, pwm, uart... bạn có thể vừa xem vừa thực hành theo. Học về điện tử mà không biết cách lắp led thì chắc sẽ buồn cười lắm.

Học xong series này thì các bạn có thể tự làm các đề tài đơn giản được rồi. Kinh điển như là mạch led trái tim, led cube, mạch đo nhiệt độ độ ẩm hoặc điều khiển động cơ...

Giai đoạn 2: Đam mê

Ở giai đoạn này, chúng ta học cách sử dụng phần mềm Altium để thiết kế một số mạch cơ bản trước, có thể tập vẽ một chiếc kit hoặc mạch đo cảm biến. Các bạn có thể học tại series video ([ở đây](#)) khá đầy đủ. Học xong nhớ thực hành rồi ủi tay ra một vài con mạch. Nếu muốn xịn xò hơn có thể đặt mạch PCB do mình thiết kế ([tại đây](#)). Các bạn đừng tiếc tiền nhé vì đầu tư cho bản thân không bao giờ lỗ cả.

Và con chip thứ 2 các bạn nên bắt đầu sẽ phải là STM32. Không cần nói dòng chip này thực sự quá quan trọng trong lập trình nhúng. Mình sẽ ví nó như cuốn sách giáo khoa vậy. Khi mà hầu hết các loại ngoại vi hay sử dụng nó đều có. Tài nguyên nhiều và dễ dàng tìm kiếm tutorial. Và dễ mua. Đa dạng các chủng loại. Con kit rẻ nhất các bạn có thể bắt đầu có thể là STM32F103C8T6 ([bluepill](#)) bán đầy trên shopee. Các bạn có thể tham khảo tài liệu học tập nó ([tại đây](#) và [tại đây](#)).

Ở giai đoạn này mình khuyên mọi người nên tập trung làm thật nhiều dự án, có thể tìm hiểu trên mạng hoặc hỏi các anh khóa trên để kiếm đề tài. Hoặc đơn giản hơn là tìm cơ hội vào Lab trên trường, tiếp cận với các anh khóa trên và học hỏi. Mình cũng từng như vậy và trình độ lên rất nhanh khi gặp được với một anh khóa trên khá giỏi. Nên chủ động không nên bị phụ thuộc vào các thầy cô trên trường vì các thầy cô rất bận sẽ khó dành thời gian cho sinh viên được.

Nếu bạn bí quá thì mình có thể kể ra một vài đề tài như: STM32 kết hợp module SIM để đẩy dữ liệu lên server. Hoặc làm bộ định xe. Thiết bị đo thông số môi trường và quản lý trên server. Làm bộ điều khiển robot kết hợp với cảm biến gia tốc, làm bộ đóng cắt thiết bị, các thiết bị điều khiển qua hồng ngoại...

Quan điểm của mình là các bạn không cần phải học lan sang nhiều dòng chip chỉ cần nắm thật chắc về một con chip thôi cũng được. Nhưng phải làm nhiều đề tài dự án. Hầu như khi biết cách tiếp cận rồi thì chúng ta chuyển sang dòng chip mới cũng sẽ rất nhanh thôi. Nên thời gian này hãy tập trung cày con chip STM32 thật tốt. Làm từ ngoại vi đơn giản đến phức tạp. Từ đề tài dễ đến đề tài khó. Sau đó tìm cách tối ưu chương trình code. Đặt mục tiêu làm ra sản phẩm chạy được lâu dài. Trải qua tầm vài dự án bạn sẽ thấy tư duy của mình lên một tầm cao mới.



- Làm sao để tối ưu chương trình. Hạn chế các điểm blocking.
- Làm sao để mạch chạy được trong môi trường khắc nghiệt (nhiệt độ cao, bụi bẩn...).
- Làm sao để update phần mềm từ xa cho mạch.
- Làm sao để mạch tiết kiệm chi phí nhất có thể.
- Làm sao để mạch tiêu thụ ít năng lượng nhất có thể (bài toán Low power).
- Làm thế nào để quy trình sản xuất mạch tốn ít thời gian nhất.

Kết thúc giai đoạn này thì các bạn đã có thể nghĩ đến việc thực tập tại các công ty bên ngoài để có thể tiếp cận với nguồn kiến thức thực tế. Sẽ hoàn toàn khác xa so với những gì các bạn tưởng tượng.

Giai đoạn 3: Lựa chọn

Ở giai đoạn này bạn có thể lựa chọn một số hướng đi khác nhau.

Hướng 1: Trở thành kĩ sư phần cứng, thiết kế mạch

Để theo được hướng này bạn sẽ cần tìm hiểu kĩ hơn về các kiến thức điện tử, đặc điểm linh kiện, cách sử dụng phần mềm thiết kế (Altium), các nguyên tắc vẽ mạch để hạn chế các nhiễu có thể sinh ra hay đảm bảo dòng hoạt động cho mạch... Nâng cao hơn có thể là vẽ mạch nhiều lớp, vẽ mạch High Speed, thiết kế anten, thiết kế nguồn, thiết kế mạch công suất ...

Theo hướng này thì các bạn phải tự đọc tài liệu tiếng anh rất là nhiều, xem nhiều video nước ngoài vì đơn giản tài liệu nâng cao mảng này mà tiếng việt thì hoàn toàn không có. Các bạn có thể tham gia [Group này](#) có nhiều bài viết rất chuyên sâu.

Hướng 2: IoT

Đây là hướng mình đang theo đuổi và mình cũng nghĩ sẽ có nhiều bạn đam mê theo hướng đi này. Để theo hướng này các bạn sẽ cần tìm hiểu thêm về các giao thức mạng không dây có thể kể đến như Wifi, Bluetooth, Zigbee, Zwave, Lora, GSM, 3G, 4G, RF433, RF315, RF 2.4G, hồng ngoại...

Hiện nay thì xu hướng IoT đang bùng nổ khi mà các tập đoàn lớn như Vin, FPT, Rạng Đông, Điện Quang, Phenikaa, Lumi, VNPT, Viettel, ... và rất nhiều các công ty startup khác đều đang chạy đua mảng công nghệ này, đặc biệt trong lĩnh vực là Smart Home, Smart Farm, Smart City. Mở ra thị trường việc làm đầy tiềm năng và cạnh tranh. Một trong cá Group mình thấy rất thú vị để bắt xu hướng và có thể là khởi nguồn của bất kì anh em nào có máu startup trong người là [đây](#).

Ở hướng đi này, các bạn phải bắt đầu học code các dòng chip như ESP32-IDF ([học tại đây](#)) hoặc NRF52832 ([tại đây](#)), các dòng chip của Silabs (dòng EFR32, dòng này thì phải tự bơi thôi), các module truyền thông Lora (sx1278) hay NRF24L01, hồng ngoại hoặc RF433, RF315. Ngoài ra thì cũng cần học thêm về GSM, 3G, 4G, Ethernet. Có rất nhiều lựa chọn. Và tùy theo công ty mà các bạn theo đuổi mà bạn sẽ cần chuyên sâu vào một số công nghệ nào đó. Vì để master các công nghệ này thì ít nhất các bạn cũng dành tối thiểu 1 năm cho mỗi loại. Thời gian có thể dài hơn hoặc ngắn hơn tùy vào từng người nữa.

Nói chung thì học lên phần này sẽ cực hơn so với lập trình MCU thuần rất là nhiều tuy nhiên lại thú vị hơn rất nhiều, đam mê rồi thì khó mà dứt ra được. Bởi vì học đến đây các bạn đã có thể làm ra các sản phẩm hoàn chỉnh, các mô hình mạng rất to rồi. Đã có thể đi outsource để kiếm thêm tiền. Nếu bạn nào muốn startup thì đây cũng là một thời điểm để trải nghiệm.

Hướng 3: Embedded Linux

Đây là một hướng đi khá khó nhằn. Tuy nhiên nhu cầu việc làm luôn có và mức thu nhập rất tốt. Vậy Embedded Linux là gì ? Nói cho đơn giản thì đó là những hệ thống chạy trên nền tảng hệ điều hành Linux, các hệ thống này hiện diện rất nhiều trong cuộc sống của chúng ta cụ thể chính là các thiết bị điện thoại Android (lõi của android là linux kernel), các thiết bị wifi (router, modem), màn hình giải trí trên xe ô tô, android TV vv.. Nhìn chung, công việc về Embedded Linux rất nhiều chỉ là ta chưa đáp ứng được nhu cầu của doanh nghiệp và tiếp cận nó đúng cách.

Trước khi đi vào các mảng công việc cụ thể, các bạn cần phải biết tới 4 thành phần quan trọng của một Linux OS, bao gồm:

- **Bootloader:** Bộ nạp khởi động, có chức năng load OS và các thành phần khác lên trên hệ thống.
- **Linux Kernel:** Nhân của Linux OS, chứa các trình điều khiển thiết bị (device drivers).
- **Rootfs:** Userspace, là không gian để cho developers chúng ta làm việc, phát triển applications.
- **Toolchain:** Là một công cụ để giúp cho việc phát triển phần mềm, debug như compiler (gcc/g++), debugger (gdb), linker (ld) và các libs cần thiết.

Khi các bạn mua một thiết bị nhúng Linux trên thị trường, lấy ví dụ như board Raspberry pi, board IMX, board Beaglebone thì bên nhà sản xuất thiết bị (vendor) sẽ cung cấp cho chúng ta một bộ 4 thành phần trên và được gọi chung là bộ BSP - Board Support Package.

Các công ty về Embedded Linux hiện nay nhìn chung được chia theo 2 hướng:

- **Công ty out-source** (điển hình là FPT):



- **Công ty làm product** (VNPT, Lumi, Rạng đồng vv..):

Trước tiên phải xét tới quy trình tạo ra 1 sản phẩm:

Về mặt phần cứng.

Ra bài toán cụ thể về sản phẩm, lựa chọn thiết bị thích hợp của một vendor nào đó để phát triển.

Đưa ra được danh sách các tính năng. VD: Gateway có chức năng kết nối wifi, sử dụng được bluetooth, zigbee vv... Từ đó thiết kế lại mạch sao cho các ngoại vi cần thiết hoạt động ổn định, lược bỏ các thành phần không cần thiết để tối ưu chi phí.

Về phần mềm.

Từ bộ BSP nhận được khi hợp tác với vendor, triển khai tối ưu/sử đổi lại uboot, device drivers, applications đảm bảo các ngoại vi và các chức năng chạy đúng, ổn định. Loại bỏ các module không cần thiết ra khỏi hệ thống, dành tài nguyên cho việc phát triển sau này, càng tối ưu được nhiều thì sản phẩm càng có thể phát triển thêm nhiều tính năng. Giai đoạn này được gọi là giai đoạn Porting.

Song song với giai đoạn Porting là phát triển ứng dụng trên tầng userspace. Tùy vào từng công ty mà các ứng dụng phát triển là khác nhau. Tuy nhiên, do tài nguyên là giới hạn nên hầu hết ứng dụng sẽ được phát triển bằng ngôn ngữ C/C++.

Câu hỏi là, tại sao không tăng thêm bộ nhớ, sử dụng tài nguyên tốt hơn mạnh hơn?

Việc này đồng nghĩa với việc công ty phải bỏ thêm chi phí sản xuất khi cấu hình phần cứng tăng lên, đôi khi là hàng nghìn, chục nghìn thiết bị thì số tiền này không hề nhỏ.

Tóm lại từ các công việc trên thì Linux Embedded có thể chia ra thành 3 mảng kiến thức lớn sau:

- **Linux Programming:** Viết các ứng dụng trên môi trường Linux.

Lập trình tốt ngôn ngữ C/C++ trên môi trường Linux (FileIO, Process, Thread, Signal vv..).

Có kiến thức cơ bản về hệ điều hành.

- **Linux Porting:** Phần này sẽ thiên về tối ưu, customize hệ thống.

Cần có kiến thức về Makefile, shell scripts.

Hiểu biết về các build system như build-root, yocto.

Hiểu biết về quá trình khởi động của hệ điều hành.

Biết cách sửa đổi bootloader, kernel, thêm hay loại bỏ các package ở rootfs.

- **Linux Device Drivers:** Viết các trình điều khiển thiết bị.

Thường dành cho những bạn đã có kiến thức cơ bản về hai phần trên.

Học viết một số các drivers cơ bản như I2C/SPI/UART/USB/Watchdog vv...

Để mà học được theo hướng này thì tự học là khá khó. Các khóa học tiếng việt cũng chưa có nhiều nhưng bạn có thể tham khảo các khóa học tại Deviot ([level 1](#), level 2, level 3). Một group cực chất lượng trong mảng này và các bạn không thể bỏ qua đó là [đây](#).

Hướng 4: Lập trình chuyên sâu vi điều khiển

Ở hướng đi này đòi hỏi cần phải có kiến thức cực tốt về ngoại vi vi điều khiển. Các loại protocol hay sử dụng trong công nghiệp và automotive (SPI, LIN, CAN, FR, ETH). Kiến thức về lập trình phải cực kì tốt.

Công việc sẽ chủ yếu là viết driver cho chip. Đơn vị điển hình tuyển dụng mảng này có FPT, Bosch, Vietel.

Bù lại thì sau vài năm bạn sẽ trở thành một pro code chính hiệu.

Nếu bạn xác định đi theo hướng này thì mình khuyên các bạn lên bắt đầu với STM32 càng sớm càng tốt và sau đó xin lên công ty thực tập ngay và luôn.

Ngoài ra ở hướng này mình nghĩ các bạn cần có thêm một số skill cần thiết nữa điển hình như học Python (để viết tool), học Qt C++, học C# để viết UI phần mềm trên máy tính. Các bạn có thể tham khảo một tuyển dụng của FPT để biết mình cần cải thiện thêm gì ([tại đây](#)).

Cuối cùng của bài viết này. Là góc PR bản thân mình :D Nếu các bạn đọc xong bài viết và vẫn hoang mang chưa biết phải bắt đầu từ đâu. Thì hãy tìm đến ngay Deviot để được tư vấn và học tập theo định hướng. Deviot đã xây dựng lên đầy đủ các khóa học lập trình trong lĩnh vực nhúng và IoT đa dạng và đầy đủ theo từng cấp bậc. Mục đích chỉ một là phục vụ việc ra trường và đi làm.

- *Lập trình C cơ bản:* [link](#)

- *Lập trình C nâng cao:* [link](#)

- *Lập trình C++:* [link](#)

- *Lập trình Python và IoT:* [link](#)

- *Lập trình PIC:* [link](#)

- *Lập trình STM32 cơ bản:* [link](#)

- *Lập trình STM32 nâng cao:* [link](#)

- *Lập trình STM32 full từ cơ bản đến nâng cao:* [link](#)

- *Lập trình STM32 và RTOS:* [link](#)

- *Lập trình STM32 và Lora:* [link](#)

- *Lập trình STM32 và module sim:* [link](#) và [link](#)



Blog kỹ
thuật

Tutorial Tài liệu Giới thiệu Góc học viên



Đăng nhập

- *Embedded Linux level 1*: [link](#)
- *Embedded Linux level 2*: [link](#)
- *Lập trình STM32 hướng sự kiện*: [link](#)

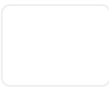
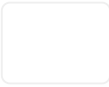
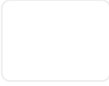
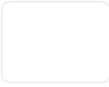
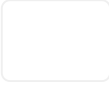

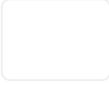

Hình thức học chủ động qua video trên website <https://deviot.vn> hoặc học trực tuyến cùng giảng viên qua Microsoft Teams (tuyển sinh hàng tháng).

Hi vọng qua bài viết này, các bạn đã có thể có thể biết mình đang ở đâu và cần đi tiếp như thế nào.
Good luck,
Truong Giang,

***Note: Nếu bài viết được chia sẻ thì hi vọng mọi người sẽ giữ nguyên nguồn vì mình đã phải thức hơi khuya để hoàn thành một bài viết tổng hợp như này (bài viết thuộc về Deviot.vn)**

22 giờ trước · 191 views

Cùng Tác Giả

-  Chuỗi video điện tử cơ bản
-  Giao thức Modbus trong Công nghiệp
-  Tuyển tập các câu hỏi phỏng vấn C (Phần 1)
-  Latching relay
-  Watchdog Timer
-  DEEP LEARNING DỄ DÀNG HƠN VỚI GOOGLE COLAB
-  ĐÓNG GÓI DỮ LIỆU TRONG TRUYỀN NHẬN UART.
-  Tại sao dùng delay trong ngắt lại bị treo?

Bài viết liên quan

Toan Le 123 · 2055 Views

Công nghệ Bluetooth trong IoT (Phần1)

Tuan Doan · 2236 Views

Công nghệ Bluetooth Low Energy trong IoT (Phần 2)

Tuan Doan · 2755 Views

Một số nền tảng phần cứng hàng đầu cho Internet of Things (IoT)



Tuan Doan · 4570 Views

Nên dùng ESP8266 hay ESP32 để học lập trình IoT?

Tuan Doan · 2903 Views

Những dòng chip được sử dụng trong IoT

Tuan Doan · 3948 Views

AT Command

Đăng Lợi · 2448 Views

SMART CONFIG CHO ESP32



Số 101C, ngõ Xã
Đàn 2, Đống Đa, Hà Nội



0969666522



deviotcore@gmail.com

Chúng tôi là ai?

- DEVIOT là một trung tâm đào tạo về kỹ thuật được xây dựng bởi những kĩ sư giàu kinh nghiệm đến từ các công ty tập đoàn công nghệ nổi tiếng trong nước
-
- Chúng tôi mang trong mình sứ mệnh đem những kiến thức thực tế tích lũy được trong quá trình làm việc đến với các bạn học sinh, sinh viên trên cả nước

- Chúng tôi mang trong mình sứ mệnh đem những kiến thức thực tế tích lũy được trong quá trình làm việc đến với các bạn học sinh, sinh viên trên cả nước

Chuyên ngành đào tạo

- Lập trình nhúng và IoT
- Thiết kế mạch điện tử
- Ngôn ngữ lập trình
- Xử lý ảnh và AI

- Thiết kế mạch điện tử

- Ngôn ngữ lập trình

- Xử lý ảnh và AI

Fanpage Facebook



Deviot.vn - Lập trình nhúng
3.335 lượt thích

3.335 lượt thích



Deviot.vn - Lập trình nhúng & IoT

3 giờ trước

Lập trình nhúng là gì ?

Nói đến lập trình chúng ta nghĩ ngay đến CNTT (IT). Chuyên ngành mà chúng ta có thể tạo ra những tựa game hay, những phần mềm đỉnh cao, các trang web lớn, công nghệ AI, Blockchain ... đủ mọi trong tưởng tượng. Còn với lập trình nhúng, các bạn sẽ đi lập trình các chương trình chạy cho những con chip, các dòng điều khiển, Vi xử lý. Đối với CNTT chúng ta có vô vàn các loại ngôn ngữ bậc cao để lập trình. Còn đối với lập trình nhúng, ngôn ngữ ... [Xem thêm](#)

... [Xem thêm](#)

