

HỌC VIỆN KỸ THUẬT QUÂN SỰ  
KHOA VÔ TUYẾN ĐIỆN TỬ

NGUYỄN HẢI DƯƠNG

**TÀI LIỆU HƯỚNG DẪN THÍ NGHIỆM  
THIẾT KẾ HỆ THỐNG NHÚNG**

(Dùng cho hệ đào tạo sau đại học)

**Bài thí nghiệm: Truyền dữ liệu đa điểm trong mạng  
cảm biến không dây**

HÀ NỘI, 2019



# MỤC LỤC

<b>LỜI NÓI ĐẦU .....</b>	<b>2</b>
<b>PHẦN 1: MỤC ĐÍCH, YÊU CẦU .....</b>	<b>4</b>
<b>1.1 Mục đích.....</b>	<b>4</b>
<b>1.2 Yêu cầu.....</b>	<b>4</b>
1.2.1 Phương tiện, dụng cụ thí nghiệm .....	4
1.2.2 Yêu cầu đối với sinh viên.....	4
<b>PHẦN 2: TÓM TẮT LÝ THUYẾT .....</b>	<b>5</b>
<b>2.1 Tổng quan về hệ thống nhúng.....</b>	<b>5</b>
2.1.1 Hệ thống nhúng .....	5
2.1.2 Mạch thí nghiệm Lập trình nhúng thời gian thực .....	6
<b>2.2 Giới thiệu về mạng không dây Zigbee .....</b>	<b>9</b>
2.2.1 Sơ lược về mạng không dây WPAN .....	9
2.1.2 Mạng không dây Zigbee/IEEE 802.15.4 .....	10
<b>2.3 Quy trình thiết kế, lập trình điều khiển Truyền dữ liệu đa điểm.....</b>	<b>16</b>
<b>PHẦN 3: NỘI DUNG THÍ NGHIỆM.....</b>	<b>18</b>
<b>3.1 Các nội dung cơ bản .....</b>	<b>18</b>
<b>3.2 Nội dung chính.....</b>	<b>19</b>
<b>3.3 Nội dung nâng cao .....</b>	<b>19</b>
<b>PHẦN 4: TRÌNH TỰ THÍ NGHIỆM.....</b>	<b>21</b>
<b>4.1 Các thiết lập, cài đặt hệ thống chuẩn bị thí nghiệm.....</b>	<b>21</b>
4.1.1 Cài đặt hệ điều hành trên thẻ nhớ micro SD .....	21
4.1.2 Cài đặt hệ điều hành trên thẻ nhớ eMMC.....	23
4.1.3 Kết nối với bàn phím, chuột, USB Stick qua cổng USB.....	24
4.1.4 Kết nối Ethernet.....	25
4.1.5 Kết nối HDMI với màn hình .....	26
4.1.6 Kết nối từ xa thông qua SSH.....	27
4.1.7 Kết nối truyền nhận file qua SFTP.....	29
4.1.8 Giới thiệu về Python .....	31
4.1.9 Kích hoạt truyền thông UART trên KIT.....	31
<b>4.2 Thiết lập chế độ hoạt động cho mô đun Zigbee CC2530 .....</b>	<b>34</b>
<b>4.3 Lập trình truyền nhận không dây sử dụng mô đun CC2530.....</b>	<b>35</b>
<b>PHẦN 5: BÁO CÁO THÍ NGHIỆM .....</b>	<b>40</b>
<b>PHỤ LỤC A: MẪU BÁO CÁO THÍ NGHIỆM.....</b>	<b>41</b>
<b>PHỤ LỤC B: CODE THAM KHẢO.....</b>	<b>43</b>
Code nội dung 3 của mục 3.1 .....	43
Code của mục 3.2.....	49

## LỜI NÓI ĐẦU

Chúng ta không thể không phủ nhận được tầm quan trọng của thông tin liên lạc trong đời sống hiện nay. Trong những năm gần đây, khái niệm mạng không dây đã không còn xa lạ với chúng ta và trở thành một phần tất yếu trong cuộc sống. Hiện nay truyền thông không dây đang phát triển mạnh mẽ, được sử dụng rộng rãi trong các hệ thống điện tử hiện đại và đang trở thành một yếu tố quan trọng trong kỷ nguyên của Internet và các hệ thống thông minh, cũng như cho các ứng dụng trong an ninh-quốc phòng.

Đã có rất nhiều những nghiên cứu và ứng dụng của công nghệ truyền thông không dây trong thực tế. Một trong số đó là công nghệ truyền thông không dây Zigbee với ưu điểm ra đời để bổ xung phần còn thiếu của thế giới mạng – một công nghệ hoạt động trong phạm vi hẹp, tiêu thụ ít năng lượng để phục vụ kết nối và quản lý các cảm biến, truyền dữ liệu an toàn, kết nối mạng nhanh. Đối tượng mà công nghệ Zigbee nhắm đến là các giải pháp nhà thông minh hay các hệ thống tự động. Trong đó các thiết bị sẽ tự thu thập dữ liệu thông qua các bộ cảm biến và tự động trao đổi với nhau để đem đến trải nghiệm tốt nhất cho người dùng.

Nội dung thí nghiệm về thiết kế các điểm nút mạng, cấu trúc mạng không dây Zigbee đã được sử dụng rộng rãi trong chương trình đào tạo của nhiều trường đại học trong nước và trên thế giới. Tuy nhiên, nội dung thí nghiệm này chưa được đưa vào nội dung bất kỳ môn học nào trong các chương trình đào tạo bậc đại học và sau đại học của Học viện Kỹ thuật quân sự. Vì vậy, bài thí nghiệm này được xây dựng nhằm giúp cho học viên nắm được được quy trình, kỹ năng cài đặt, xây dựng hệ thống thông tin truyền dữ liệu đa điểm trong mạng cảm biến không dây, từ đó đặt nền móng trong việc nghiên cứu và phát triển các cấu hình mạng mạng cảm biến không dây trong các hệ thống điện tử hiện đại.

Nội dung tài liệu hướng dẫn thí nghiệm “Truyền dữ liệu đa điểm trong mạng cảm biến không dây” được biên soạn theo hướng giảm nội dung lý thuyết, tập trung vào hướng dẫn cụ thể, chi tiết từng bước làm thí nghiệm của từng mục thí nghiệm trong bài thí nghiệm. Các bài tập nhỏ, kết quả làm thí nghiệm mẫu đã được nhóm môn học kiểm chứng, có vai trò định hướng cho học viên trong quá trình tự tiến hành làm thí nghiệm.

Về cấu trúc, sau phần tóm tắt lý thuyết, nội dung hướng dẫn thí nghiệm được chia thành ba phần chính: phần 1 hướng dẫn chi tiết cách thức tiến hành các bước cài đặt, sử dụng Mạch thí nghiệm để chuẩn bị cho thí nghiệm, phần 2 hướng dẫn chi tiết các bước thực hiện truyền thông không dây sử dụng mô đun Zigbee CC2530, phần 3 là các nội dung thí nghiệm yêu cầu học viên tự làm. Ngoài ra, phụ lục cung cấp thông chi tiết hơn về một số nội dung thí nghiệm.

Chúng tôi rất mong nhận được những ý kiến đóng góp để tài liệu được hoàn thiện hơn nữa. Mọi ý kiến đóng góp xin gửi cho tác giả theo địa chỉ email: bomonvixuly@gmail.com hoặc theo địa chỉ: Bộ môn Kỹ thuật Vi xử lý, Khoa Vô tuyến Điện tử, Học viện Kỹ thuật Quân sự, số 236 Hoàng Quốc Việt, Hà Nội.

## **PHẦN 1: MỤC ĐÍCH, YÊU CẦU**

### **1.1 Mục đích**

Bài thí nghiệm này nhằm giúp cho học viên nắm được nắm được quy trình sử dụng, có năng cài đặt, thiết kế và nâng cấp hệ thống nhúng thông qua việc xây dựng hệ thống thông tin truyền dữ liệu đa điểm trong mạng cảm biến không dây thực hiện trên mạch thí nghiệm nhúng của Bộ môn Kỹ thuật Xung số - vi xử lý, từ đó đặt nền móng trong việc nghiên cứu và phát triển các cấu hình mạng trong mạng cảm biến không dây Zigbee.

Hơn nữa, bài thí nghiệm này cũng giúp học viên có phương pháp phân tích, lập trình hệ thống, tận dụng các mã nguồn mở và các phương pháp gỡ lỗi chương trình.

### **1.2 Yêu cầu**

#### **1.2.1 Phương tiện, dụng cụ thí nghiệm**

➤ Máy vi tính PC có cổng USB, dây nhảy để kết nối các mô đun, dây kết nối mạng, và cài đặt các công cụ: phần mềm PuTTY, Advanced IP Scanner, Filezilla Client.

➤ Bo mạch thí nghiệm Lập trình nhúng thời gian thực tại Phòng thí nghiệm của Bộ môn Kỹ thuật Vi xử lý.

#### **1.2.2 Yêu cầu đối với sinh viên**

➤ Nắm vững kiến thức lý thuyết về hệ điều hành Linux, ngôn ngữ lập trình Python, nguyên lý truyền tin của mạng không dây, có nền tảng cơ bản về hệ thống nhúng.

➤ Đọc kỹ hướng dẫn thí nghiệm trước khi thí nghiệm.

➤ Đã được trang bị các kiến thức an toàn khi làm việc với các thiết bị điện tử.

## **PHẦN 2: TÓM TẮT LÝ THUYẾT**

### **2.1 Tổng quan về hệ thống nhúng**

#### **2.1.1 Hệ thống nhúng**

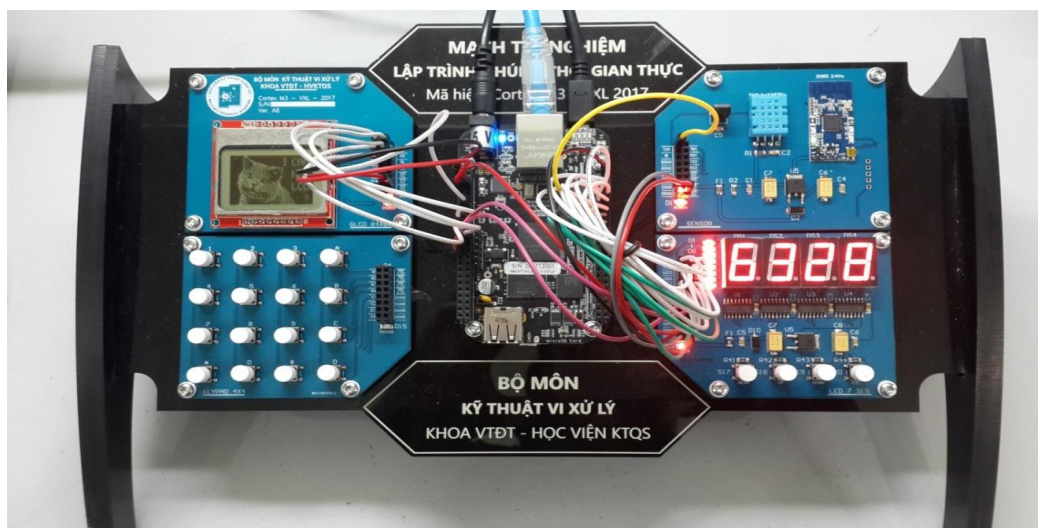
Các hệ thống nhúng là một trong những nguồn gốc của ba lĩnh vực là ubiquitous computing, pervasive computing, và ambient intelligence, chúng cung cấp phần lớn các công nghệ cần thiết. Hệ thống nhúng là hệ thống xử lý thông tin mà được nhúng vào trong một sản phẩm lớn hơn và thường người dùng không trực tiếp nhìn thấy. Đó là các hệ thống tích hợp cả phần cứng và phần mềm để thực hiện một hoặc một nhóm chức năng chuyên biệt cụ thể. Hệ thống nhúng được sử dụng trên hầu hết các lĩnh vực điện tử, y học, gia dụng cũng như quân sự như điện tử ô tô, máy bay, tàu hỏa, viễn thông, y tế, điện gia dụng, nhà thông minh, robotics...

Tầm vóc của thị trường hệ thống nhúng có thể được phân tích từ nhiều góc nhìn. Xét số lượng vi xử lý đang được sử dụng (2003), người ta ước tính có khoảng 79% đang được dùng trong các hệ thống nhúng. Nhiều bộ vi xử lý nhúng là loại vi xử lý 8 bit, mặc dù vậy, 75% các bộ vi xử lý 32bit được tích hợp vào các hệ thống nhúng. Năm 1996, ước tính hàng ngày một người Mỹ làm việc với trung bình 60 bộ vi xử lý. Một số ô tô cao cấp có tới hơn 100 bộ vi xử lý. Những con số này lớn hơn nhiều so với hình dung, do hầu hết mọi người không nhận ra rằng họ đang dùng các bộ vi xử lý. Người ta dự báo rằng thị trường hệ thống nhúng sẽ sớm vượt thị trường của các hệ thống dạng PC. Ngoài ra, số phần mềm dùng cho các hệ thống nhúng cũng được dự đoán là sẽ tăng.

Các hệ thống nhúng tạo thành nền tảng cho cái gọi là thời kỳ hậu PC, trong đó việc xử lý thông tin ngày càng chuyển dịch ra xa các hệ thống toàn PC tiến về các hệ thống nhúng. Số lượng ứng dụng ngày càng tăng dẫn đến nhu cầu các công nghệ thiết kế hỗ trợ các hệ thống nhúng. Các công nghệ và công cụ hiện tại vẫn có những hạn chế quan trọng. Ví dụ, vẫn cần các ngôn ngữ đặc tả tốt hơn, các công cụ sinh mã từ đặc tả, các bộ kiểm chứng các điều kiện thời gian, hệ điều hành thời gian thực, các kỹ thuật thiết kế tiết kiệm năng lượng, và các kỹ thuật thiết kế dành cho các hệ thống có độ tin cậy cao.

### 2.1.2 Mạch thí nghiệm Lập trình nhúng thời gian thực

KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017) được nghiên cứu, thiết kế và phát triển dựa trên nền tảng mã nguồn mở của KIT BeagleBone Black theo định hướng Internet of Thing, tất cả các thiết bị làm việc thông qua Internet.



*Hình 2.1 Tổng quan của KIT*

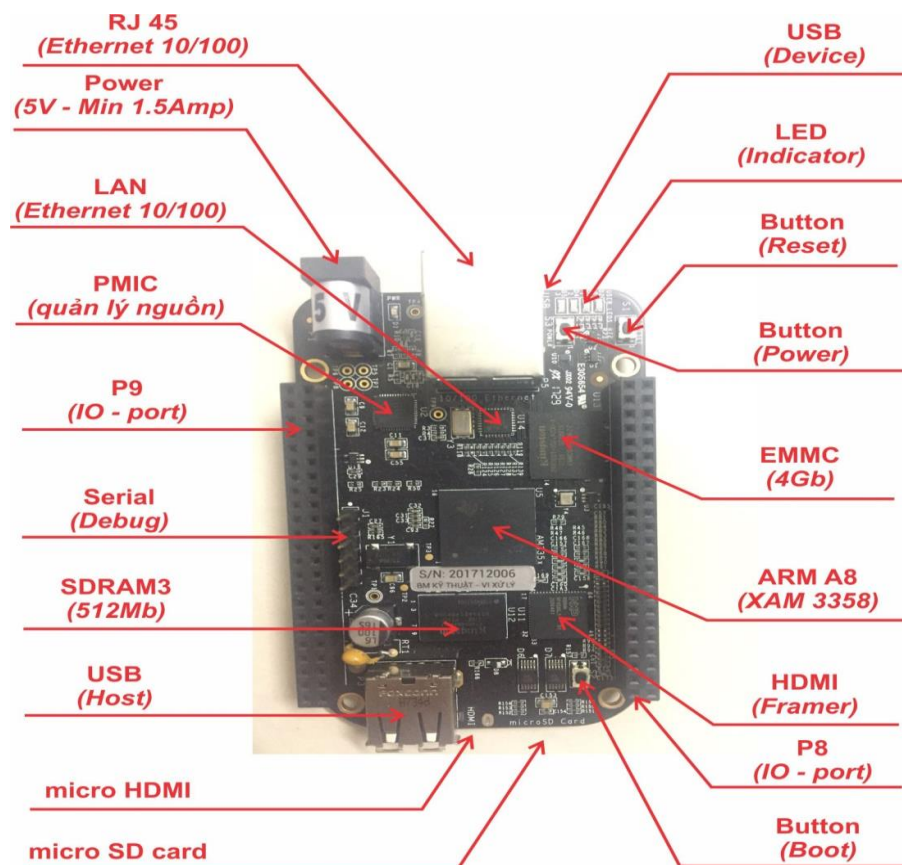
Bo mạch chủ của nó được xem là máy tính nhúng, chi phí thấp với bộ xử lý ARM Cortex A8 32-bit hoạt động từ 720 MHz đến 1 GHz. "Bone" có thể được sử dụng trong một loạt các dự án, triển lãm khoa học sơ cấp cho tới dự án thiết kế cao cấp và các hình mẫu đầu tiên của hệ thống rất phức tạp. Người dùng mới có thể truy cập vào sức mạnh của Bone thông qua môi trường Bonescript user-friendly, một trải nghiệm dựa trên trình duyệt, trong MS Windows, Mac OS X, và hệ điều hành Linux. Người dùng chuyên nghiệp có thể tận dụng sức mạnh của Bone bằng cách sử dụng dựa trên hệ điều hành Linux, một loạt các tính năng mở rộng (Capes) và rất nhiều loại của cộng đồng Linux thư viện mã nguồn mở. Texas Instruments có lịch sử lâu đời về việc đạo tạo các thế hệ kế tiếp chuyên nghiệp về STEM (Science, Technology, Engineering and Mathematics). BeagleBone là một sự đổi mới giáo dục mới nhất trong một di sản lâu dài, trong đó bao gồm các Speak & Spell và máy tính cầm tay TI. Mục tiêu của dự án BeagleBone là tạo ra một máy tính mở rộng mạnh mẽ, trong tầm tay của các nhà sáng tạo trẻ thông qua việc sử dụng thân thiện, dựa trên môi trường trình duyệt Bonescript. Theo những kiến thức và kỹ năng của những



chuyên gia, BeagleBone cung cấp giao diện phức tạp hơn bao gồm việc bổ sung các chức năng dựa trên ngôn ngữ C để truy cập vào hệ thống phần cứng ARM Cortex A8 processor. Những sẽ hữu ích này cho các dự án thiết kế cấp cao bao gồm thiết kế dự án capstone . Toàn bộ sức mạnh của bộ vi xử lý có thể được tung ra khi sử dụng trên hệ điều hành Linux. Để lắp ráp một hệ thống tùy chỉnh, Bone có thể được kết hợp với một loạt các board con "Capes" và các thư viện mã nguồn mở. Những tính năng này cho phép thao tác nhanh chóng trong việc mở rộng hệ thống nhúng phức tạp. Sự đa dạng của Capes để mở rộng các tính năng xử lý có sẵn, hiện đang có hơn 35 Capes khác nhau.

Các mô đun mở rộng của KIT được thiết kế thành các bo mạch rời (các mạch mở rộng) thuận tiện cho việc học tập, bảo dưỡng, thay thế hoặc nâng cấp theo xu hướng phát triển của công nghệ. KIT bao gồm 5 mô đun, trong đó có một mô đun Vi xử lý và 4 mô đun mở rộng. Nội dung thí nghiệm sẽ sử dụng mô đun Vi xử lý, mô đun hiển thị GLCD và mô đun Sensor.

### 2.1.2.1 Mô đun Vi xử lý



Hình 2.2 Mô đun Vi xử lý

Mô đun Vi xử lý chính là bo mạch chủ của Mạch thí nghiệm lập trình nhúng thời gian thực. Nó sử dụng chip xử lý AM335x 1GHz ARM® Cortex-A8 và được thiết kế với 512MB DDR3 RAM, thẻ nhớ 4GB 8-bit eMMC gắn trên bo, khe thẻ nhớ microSD hỗ trợ thẻ nhớ microSDHC 32GB (Class 10); các kết nối HDMI, USB Host, USB Device, Ethernet và các chân IO. Và đặc biệt bo mạch này được cài đặt hệ điều hành Debian, một phiên bản phân phối (distro) của hệ điều hành Linux, rất thuận tiện cho việc phát triển lập trình ứng dụng cho hệ thống.

#### 2.1.2.2 Mô đun GLCD 84x48

Mô đun GLCD 84x48 được dùng làm bộ phận hiển thị các dữ liệu hệ thống. Nó sử dụng màn hình Graphic 84x48 điểm ảnh, giao tiếp với hệ thống bằng chuẩn SPI; đồng thời mô đun cũng có led báo nguồn và đầu cắm giao tiếp là đầu cắm chờ.



Hình 2.3 Mô đun GLCD 84x84

#### 2.1.2.3 Mô đun Sensor

Mô đun Sensor là ngoại vi gồm có một bộ cảm biến hồng ngoại với giao tiếp vào ra cơ bản (GPIO), một cảm biến nhiệt độ, độ ẩm DHT11 với giao tiếp 1 dây (1-Wire), một mô đun Zigbee CC2530 với giao tiếp SPI. Đồng thời mô đun cũng có led báo nguồn và đầu cắm giao tiếp đầu cắm giao tiếp là đầu cắm chờ. Đây chính là mô đun được sử dụng trong bài thí nghiệm truyền dữ liệu đa điểm trong mạng cảm biến không dây.



*Hình 2.4 Mô đun Sensor*

Mô đun Zibee CC2530 là module truyền sóng 2.4 GHz theo chuẩn giao thức Zigbee, khoảng cách rất xa, sử dụng giao tiếp Serial (UART) rất thông dụng. Rất phù hợp với các dự án robot thám hiểm, thăm dò, tình báo, thu thập dữ liệu cảm biến...

## **2.2 Giới thiệu về mạng không dây Zigbee**

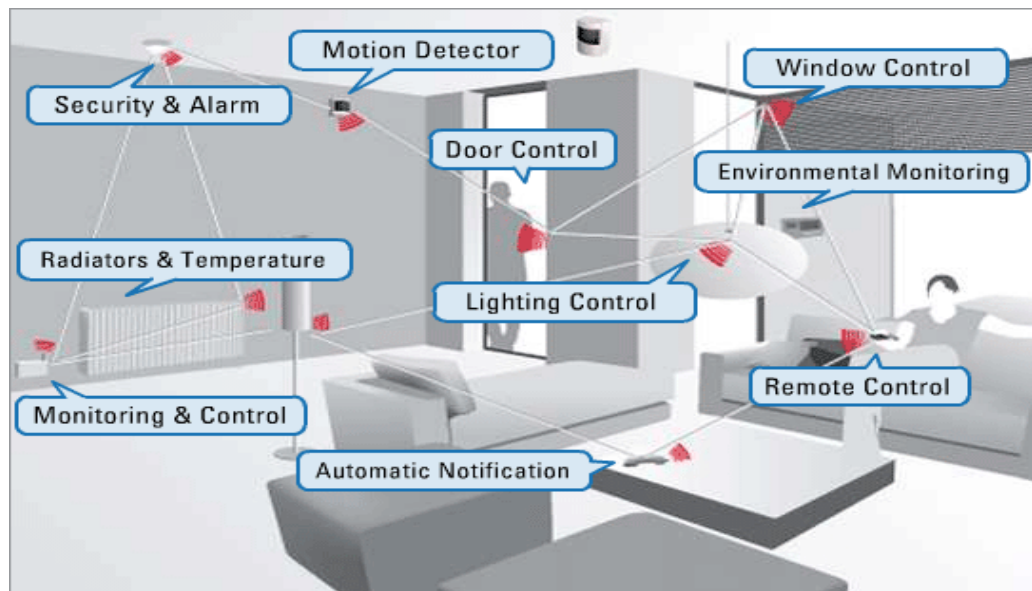
### **2.2.1 Sơ lược về mạng không dây WPAN**

Mạng không dây cá nhân, được sử dụng để kết nối các thiết bị trong phạm vi hẹp, băng thông nhỏ, ví dụ: kết nối giữa máy tính cá nhân với tai nghe (headphone), máy in, bàn phím, chuột; kết nối giữa tai nghe với điện thoại di động. Không giống như WLAN, WPAN có thể liên lạc mà không đòi hỏi nhiều về cơ sở hạ tầng. Sử dụng WPAN được coi là một hướng giải quyết hợp lý cả về giá cả, nhỏ gọn mà vẫn đem lại hiệu quả cao trong liên lạc ở băng tần hẹp.

Trong các dạng WPAN thì có lẽ chúng ta cũng đã phần nào nắm được tầm quan trọng của kết nối Zigbee đối với các ứng dụng thuộc lĩnh vực Internet Of Things trong tương lai không xa. Các kết nối Bluetooth hay Wifi tuy có tốc độ ngày càng cải thiện, tiện dụng cho người dùng và hỗ trợ nhiều ứng dụng, phần mềm khác nhau nhưng cũng vì thế các đòi hỏi về phần cứng, điện năng tiêu thụ cũng lớn hơn. Zigbee ra đời để bổ khuyết cho phần còn thiếu của thế giới mạng: một công nghệ hoạt động trong phạm vi hẹp, tiêu thụ ít năng lượng để phục vụ việc kết nối và quản lý các cảm biến – sensor.

Đối tượng mà công nghệ Zigbee nhắm đến là các giải pháp nhà thông

minh (SmartHome) hay các hệ thống tự động. Trong đó các thiết bị sẽ tự thu thập dữ liệu thông qua các sensor và tự động trao đổi với nhau để đem đến trải nghiệm tốt nhất cho người dùng. Các thiết bị tự động của thế giới Internet of Things trong tương lai phụ thuộc rất nhiều vào những dạng kết nối như Zigbee để truyền tải dữ liệu, còn các kết nối Bluetooth hay Wifi hiện nay chủ yếu vẫn sẽ chỉ dùng để phục vụ.



Hình 2.5 Ứng dụng của mạng không dây zigbee

### 2.1.2 Mạng không dây Zigbee/IEEE 802.15.4

ZigBee là một chuẩn được phát triển để xây dựng một mạng cảm biến không dây cỡ lớn (WSN). IEEE, cùng một tiêu chuẩn cơ sở của WiFi và Bluetooth, đã xác định các thông số kỹ thuật cho các lớp vật lý và MAC. IEEE 802 chịu trách nhiệm quản lý mạng cục bộ và Middle-scale Network, và 15 nhóm làm việc liên quan đến công nghệ mạng không dây đã được thiết lập ZigBee / IEEE 802.15.4 và IEEE 802.15.1 / Bluetooth. IEEE 802.11 / WiFi, IEEE 802.15.1 / Bluetooth, và ZigBee / IEEE 802.15.4 không phải là các công nghệ cạnh tranh nhau, nhưng được chuẩn hóa bởi IEEE cho từng mục đích khác nhau để chúng có thể cùng tồn tại.

Các phần trên của lớp vật lý và MAC đã được ZigBee Alliance thành lập, một cơ quan tiêu chuẩn hóa. Trong các thông số kỹ thuật của ZigBee, số lượng thiết bị đầu cuối có thể tham gia vào mạng là tối đa 65.535 (thông số kỹ thuật

của IEEE802.15.4). Nó được tối ưu hóa cho một mạng cảm biến không dây (WSN), trong đó một số lượng lớn các cảm biến có thể tham gia vào một mạng và số lượng thiết bị đầu cuối có thể được kết nối rất lớn, đó là một tính năng rất khác với WiFi (tối đa 32), Bluetooth (tối đa 7). Mạng ZigBee bao gồm các điều phối viên, bộ định tuyến và thiết bị kết thúc. Trong tiêu chuẩn, một mạng lưới đòi hỏi độ phân giải định tuyến phức tạp có thể được sử dụng.

Từ khi ZigBee là một chuẩn được phát triển để thiết lập một mạng cảm biến không dây (WSN), dữ liệu được truyền đạt là dữ liệu từ bộ cảm biến được kết nối và các tín hiệu điều khiển đến các thiết bị. Nó không giả định truyền thông dữ liệu khối lượng lớn. Do đó, tốc độ truyền dữ liệu của ZigBee là 250 kbps, không cao so với WiFi và Bluetooth. Bằng cách tối ưu hóa tốc độ truyền thông (tốc độ dữ liệu) tới một mạng cảm biến, nó sẽ giữ mức tiêu thụ điện năng thấp và độ nhạy giao tiếp cao.

Kết nối nhanh với mạng cũng là một trong những tính năng của ZigBee, nơi thiết bị đầu cuối có thể hoàn tất việc xử lý rất nhanh sau khi thức dậy từ trạng thái ngủ, kết nối với mạng, gửi dữ liệu, trở về trạng thái ngủ. Đây là một trong những tính năng làm giảm điện năng tiêu thụ trong việc áp dụng một mạng cảm biến không dây (WSN).

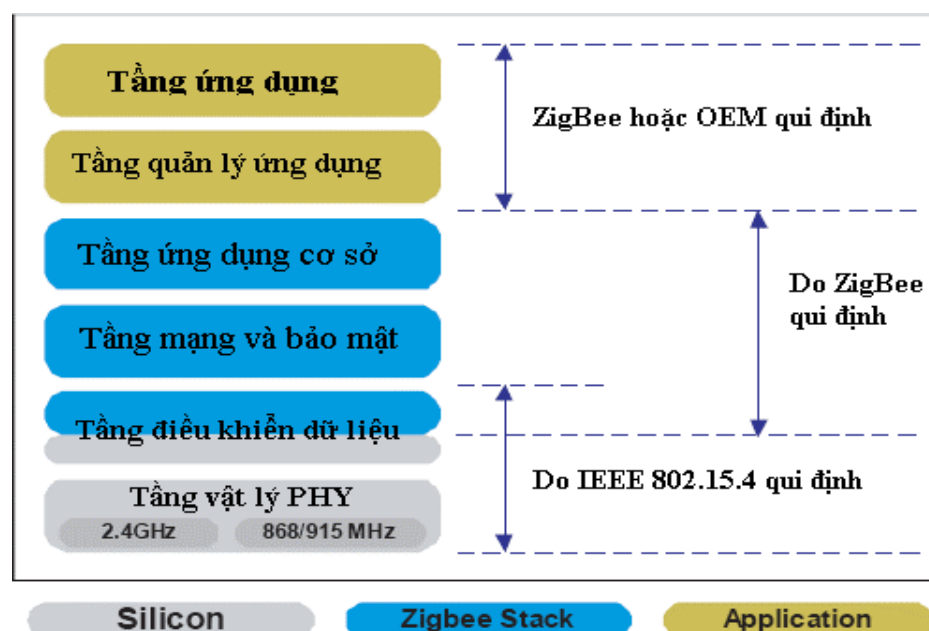
ZigBee cho thấy hiệu suất tốt nhất trong ứng dụng như: Dữ liệu truyền thông không lớn. (ví dụ dữ liệu cảm biến và dữ liệu điều khiển); truyền thông dữ liệu được thực hiện liên tục; một thiết bị đầu cuối di chuyển trong quá trình hoạt động. (Bổ sung, xóa, di chuyển).

ZigBee đảm bảo tính bảo mật của truyền thông, đây cũng là một yếu tố quan trọng trong việc xây dựng một mạng cảm biến không dây (WSN). Để thực hiện một giao thức ZigBee stack, tiến hành xử lý mạng phức tạp và có chức năng cao, giữa một số lượng lớn các nút, yêu cầu tài nguyên tương đối lớn như bộ nhớ và tốc độ xử lý. Nếu một mô-đun ZigBee không có hiệu suất đủ, đôi khi bạn có thể gặp phải hoạt động không ổn định khi số nút đạt đến vài chục. Hoạt động không ổn định như vậy không phải là do các đặc điểm kỹ thuật của ZigBee. Tuy nhiên, nó thực hiện truyền thông phức tạp (ví dụ như kiểm tra sự tồn tại của thiết bị) tất cả thời gian để duy trì một mạng lưới không dây, do đó, nó có thể hiển thị hoạt động không ổn định dưới một tình huống sóng vô tuyến



ngheo. Trong ZigBee / IEEE802.15.4 TOCOS Wireless Engine, một ngăn xếp ZigBee PRO có sẵn mà không có sự tin cậy, và CPU 32-bit mạnh mẽ và bộ nhớ với dung lượng lớn được xây dựng trong một mô-đun không dây cho hoạt động đáng tin cậy.

Công nghệ Zigbee được xây dựng và phát triển các tầng ứng dụng và tầng mạng trên nền tảng là hai tầng PHY và MAC theo chuẩn IEEE 802.15.4, chính vì thế nên nó thừa hưởng được ưu điểm của chuẩn IEEE802.15.4. Đó là tính tin cậy, đơn giản, tiêu hao ít năng lượng và khả năng thích ứng cao với các môi trường mạng. Dựa vào mô hình như hình 2.5, các nhà sản xuất khác nhau có thể chế tạo ra các sản phẩm khác nhau mà vẫn có thể làm việc tương thích cùng với nhau.



Hình 2.6 Mô hình giao thức của ZigBee

Tầng vật lý (PHY) cung cấp hai dịch vụ là dịch vụ dữ liệu PHY và dịch vụ quản lý PHY, hai dịch vụ này có giao diện với dịch vụ quản lý tầng vật lý PLME (physical layer management). Dịch vụ dữ liệu PHY điều khiển việc thu và phát của khối dữ liệu PPDU (PHY protocol data unit) thông qua kênh sóng vô tuyến vật lý. Các tính năng của tầng PHY là sự kích hoạt hoặc giảm kích hoạt của bộ phận nhận sóng, phát hiện năng lượng, chọn kênh, chỉ số đường truyền, giải phóng kênh truyền, thu và phát các gói dữ liệu qua môi trường truyền. Chuẩn

IEEE 802.15.4 định nghĩa ba dải tần số khác nhau theo khuyến nghị của Châu Âu, Nhật Bản, Mỹ.

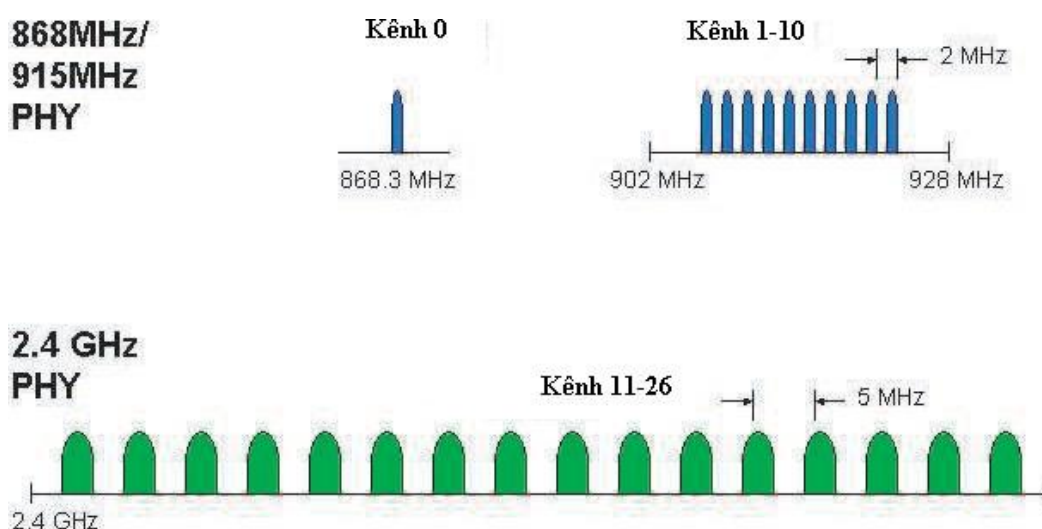
PHY (MHz)	Băng tần (MHz)	Tốc độ chip (kchips/s)	Điều chế	Tốc độ bit (kb/s)	Tốc độ ký tự (ksymbol/s)	Ký tự
868	868-868.6	300	BPSK	20	20	Nhị phân
915	902-928	600	BPSK	40	40	Nhị phân
2450	2400-2486.5	2000	O-QPSK	250	62.5	Hệ 16

*Bảng 2.1 Băng tần và tốc độ dữ liệu*

Có tất cả 27 kênh truyền trên các dải tần số khác nhau được mô tả như bảng dưới đây:

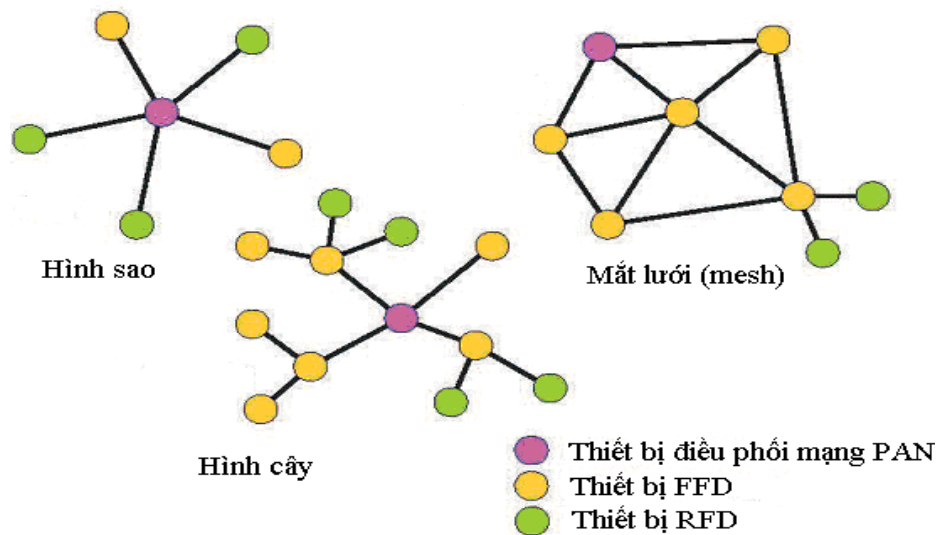
Tần số trung tâm (MHz)	Số lượng kênh (N)	Kênh	Tần số kênh trung tâm (MHz)
868	1	0	868.3
915	10	1 – 10	$906+2(k-1)$
2450	16	11 – 26	$2405+5(k-11)$

*Bảng 2.2 Kênh truyền và tần số*



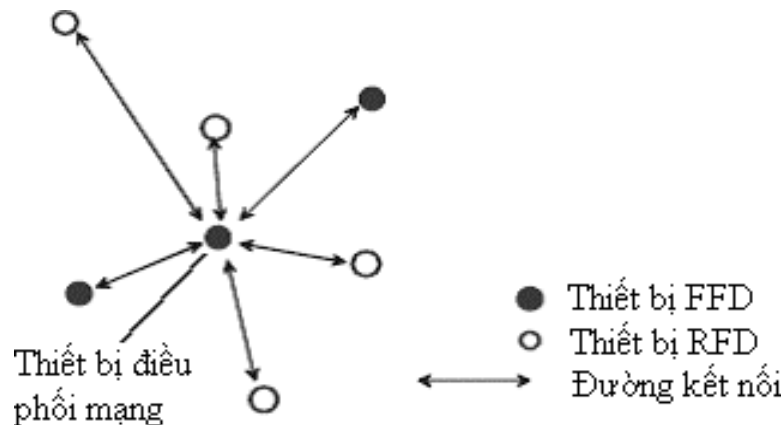
*Hình 2.7 Băng tần hệ thống của ZigBee*

Hiện nay Zigbee và tổ chức chuẩn IEEE đã đưa ra một số cấu trúc liên kết mạng cho công nghệ Zigbee. Các node mạng trong một mạng Zigbee có thể liên kết với nhau theo cấu trúc mạng hình sao (star) cấu trúc mạng hình lưới( Mesh) cấu trúc bó cụm hình cây. Sự đa dạng về cấu trúc mạng này cho phép công nghệ Zigbee được ứng dụng một cách rộng rãi. Hình 2.8 cho ta thấy ba loại mạng mà ZigBee cung cấp:



Hình 2.8 Cấu trúc liên kết mạng

a) Cấu trúc liên kết mạng hình sao (Star)



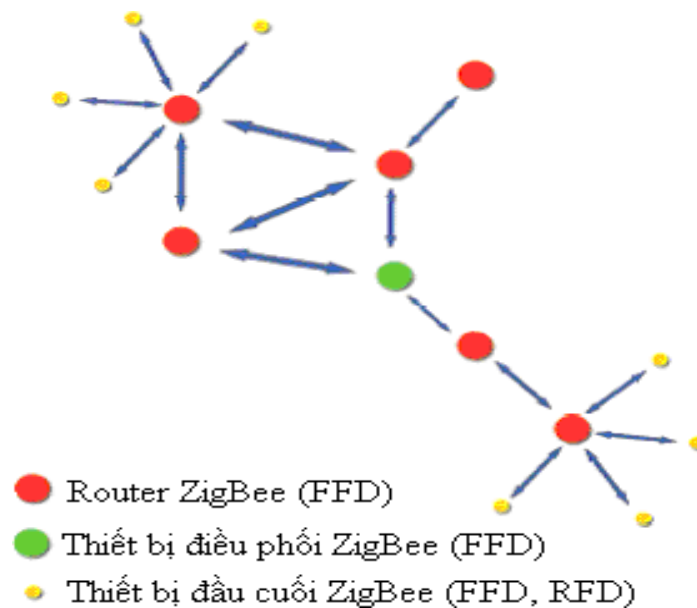
Hình 2.9 Cấu trúc mạng hình sao

Đối với loại mạng này, một kết nối được thành lập bởi các thiết bị với một thiết bị điều khiển trung tâm điều khiển được gọi là bộ điều phối mạng PAN. Sau khi FFD được kích hoạt lần đầu tiên nó có thể tạo nên một mạng độc lập và trở thành một bộ điều phối mạng PAN. Mỗi mạng hình sao đều phải có một chỉ số



nhận dạng cá nhân của riêng mình được gọi là PAN ID(PAN identifier), nó cho phép mạng này có thể hoạt động một cách độc lập. Khi đó cả FFD và RFD đều có thể kết nối tới bộ điều phối mạng PAN. Tất cả mạng nằm trong tầm phủ sóng đều phải có một PAN duy nhất, các nút trong mạng PAN phải kết nối với (PAN coordinator) bộ điều phối mạng PAN.

b) Cấu trúc liên kết mạng mắt lưới (Mesh)



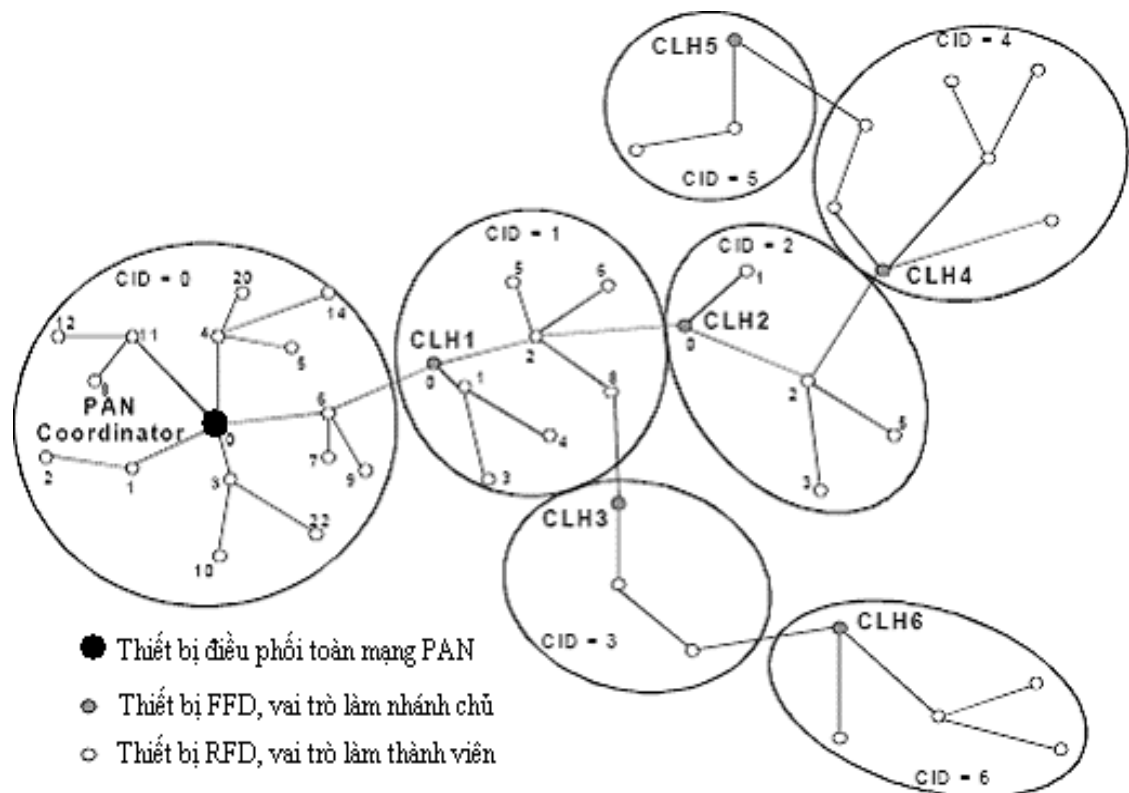
*Hình 2.10 Cấu trúc mạng mesh*

Kiểu cấu trúc mạng này cũng có một bộ điều phối mạng PAN (PAN coordinator). Thực chất đây là kết hợp của 2 kiểu cấu trúc mạng hình sao và mạng ngang hàng, ở cấu trúc mạng này thì một thiết bị A có thể tạo kết nối với bất kỳ thiết bị nào khác miễn là thiết bị đó nằm trong phạm vi phủ sóng của thiết bị A. Các ứng dụng của cấu trúc này có thể áp dụng trong đo lường và điều khiển, mạng cảm biến không dây, theo dõi cảnh báo và kiểm kê (cảnh báo cháy rừng...)

c) Cấu trúc liên kết mạng hình cây (Cluster-tree)

Cấu trúc này là một dạng đặc biệt của cấu trúc mắt lưới, trong đó đa số thiết bị là FFD và một RFD có thể kết nối vào mạng hình cây như một nốt rời rạc ở điểm cuối của nhánh cây. Bất kỳ một FFD nào cũng có thể hoạt động như là một coordinator và cung cấp tín hiệu đồng bộ cho các thiết bị và các coordinator khác vì thế mà cấu trúc mạng kiểu này có qui mô phủ sóng và khả năng mở rộng

cao. Trong loại cấu hình này mặc dù có thể có nhiều coordinator nhưng chỉ có duy nhất một bộ điều phối mạng PAN (PAN coordinator).



Hình 2.11 Cấu trúc mạng hình cây

Bộ điều phối mạng PAN coordinator này tạo ra nhóm đầu tiên cách tự bầu ra người lãnh đạo cho mạng của mình, và gán cho người lãnh đạo đó một chỉ số nhận dạng cá nhân đặc biệt gọi là CID-0 bằng cách tự thành lập CLH (cluster head) bằng CID-0 (cluster identifier), nó chọn một PAN identifier rồi và phát khung tin quảng bá nhận dạng tới các thiết bị lân cận. Thiết bị nào nhận được khung tin này có thể yêu cầu kết nối vào mạng với CLH. Nếu bộ điều phối mạng PAN (PAN coordinator) đồng ý cho thiết bị đó kết nối thì nó sẽ ghi tên thiết bị đó vào danh sách. Cứ thế thiết bị mới kết nối này lại trở thành CLH của nhánh cây mới và bắt đầu phát quảng bá định kỳ để các thiết bị khác có thể kết nối vào mạng. Từ đó có thể hình thành được các CLH1, CLH2...

### 2.3 Qui trình thiết kế, lập trình điều khiển Truyền dữ liệu đa điểm

Mạch thí nghiệm Lập trình nhúng thời gian thực sau khi được cấu hình hệ điều hành Debian sẽ được kết nối với máy tính PC thông qua cổng mạng hoặc cổng USB. Thông qua môi trường lập trình nền Web Cloud 9, dùng ngôn ngữ

lập trình Python để lập trình điều khiển các ứng dụng đối với mạch thí nghiệm Lập trình nhúng thời gian thực. Python là một ngôn ngữ tuyệt vời và mạnh mẽ đó là dễ dàng để sử dụng (dễ đọc và viết) và cho phép bạn kết nối với dự án của bạn với thế giới thực. Cú pháp Python rất sạch sẽ, nó nhấn mạnh về khả năng đọc và sử dụng từ khóa tiếng Anh chuẩn. Python hiện tại đang có 2 phiên bản được sử dụng song song là 2.7 và 3. Phiên bản Python 2.7 là phiên bản mặc định trên hệ điều hành Debian. Chương trình lập trình trên nền tảng Web Cloud 9 có thể chạy trực tiếp trên mạch thí nghiệm.

Quy trình thiết kế mạng không dây truyền dữ liệu đa điểm sử dụng công nghệ Zigbee bao gồm các bước sau:

- **Bước 1:** Phân tích bài toán: cấu trúc mạng, chức năng và mối tương tác của các Node mạng.
- **Bước 2:** Xây dựng sơ đồ hệ thống truyền tin sử dụng công nghệ Zigbee cần thiết kế.
- **Bước 3:** Cài đặt hệ điều hành, thiết kế và kích hoạt các phần cứng cho hệ thống.
- **Bước 4:** Thiết lập các chế độ truyền trên các kênh khác nhau của mô đun Zigbee CC2530 và lập trình điều khiển các chế độ truyền tin cơ bản: Point to point, Broadcast .
- **Bước 5:** Xây dựng lưu đồ tổng quát đến chi tiết (nếu cần) điều khiển quá trình truyền tin theo mô hình mạng Zigbee được xây dựng (truyền, nhận thông tin trên kênh khác nhau và truyền theo khe thời gian Timeline tùy chọn).
- **Bước 6:** Xây dựng lưu đồ thuật toán chi tiết cho các khối chương trình thu và phát thông tin, chương trình tạo khe thời gian truyền thông tin Timeline, chương trình điều khiển LCD tự động cập nhật hiển thị.
- **Bước 7:** Sử dụng ngôn ngữ lập trình Python và các thư viện hỗ trợ của ngôn ngữ này lập trình, chạy thử và đánh giá kết quả hiện thực hóa hệ thống.
- **Bước 8:** Tổng hợp và viết báo cáo thí nghiệm, kết quả hiện thực hóa hệ thống truyền dữ liệu đa điểm qua sử dụng mô đun Zigbee CC2530.

## PHẦN 3: NỘI DUNG THÍ NGHIỆM

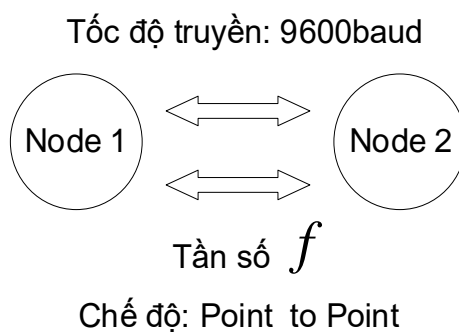
### 3.1 Các nội dung cơ bản

Dựa yêu cầu của bài thí nghiệm ở trên, học viên cần thực hiện các nội dung thực hành sau đây:

**Nội dung 1:** Cài đặt hệ điều hành Debian 8.0 cho mạch thí nghiệm Lập trình nhúng thời gian thực qua thẻ nhớ micro SD và cập nhật các thư viện: GPIO, thư viện màn hình Nokia GLCD 84x48; lập trình cập nhật thời gian thực cho hệ thống hiển thị lên màn hình GLCD thông qua giao tiếp SPI trên mạch thí nghiệm Lập trình nhúng thời gian thực.

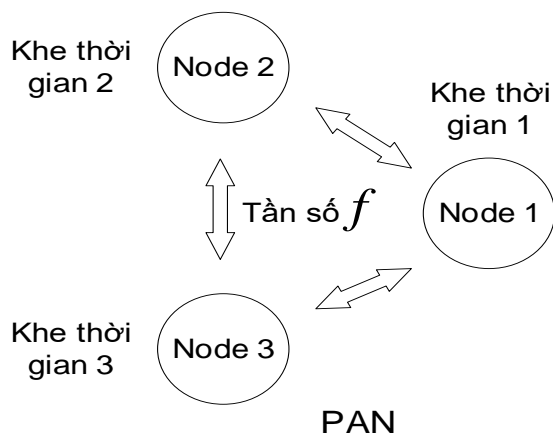
**Nội dung 2:** Lập trình điều khiển mô đun cảm biến DHT11 trong hệ thống nhúng thời gian thực: cập nhật thông tin về nhiệt độ, độ ẩm từ cảm biến DHT11 sau đó đưa ra hiển thị trên màn hình GLCD (định thời 60s cập nhật một lần).

**Nội dung 3:** Cài đặt và lập trình điều khiển hệ thống gồm 2 node cảm biến truyền dữ liệu cho nhau thông qua mô đun Zigbee CC2530 theo chế độ Point to point với tốc độ truyền là 9600 baud, truyền trên kênh số 16 của dải kênh truyền với tần số trung tâm là 2.4GHz.



*Hình 3.1 Chế độ truyền nhận Point to Point*

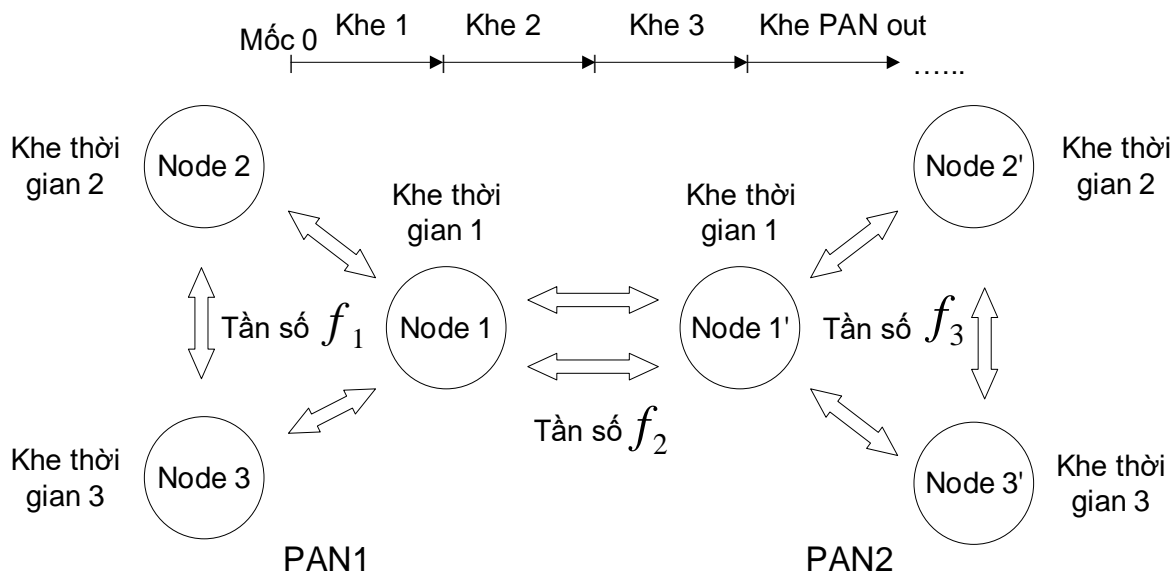
**Nội dung 4:** Cài đặt và lập trình điều khiển hệ thống mạng PAN gồm 3 node cảm biến truyền dữ liệu cho nhau thông qua mô đun Zigbee CC2530 theo chế độ Broadcast (truyền trong các khe thời gian khác nhau) với tốc độ truyền là 9600 baud, truyền trên kênh số 18 của dải kênh truyền với tần số trung tâm là 2.4GHz.



Hình 3.2 Sơ đồ mạng PAN với 3 node mạng

### 3.2 Nội dung chính

Cài đặt và lập trình điều khiển hệ thống mạng gồm 2 PAN trao đổi thông tin với nhau, với mỗi mạng PAN gồm 3 node cảm biến, chức năng của các node: có thể truyền dữ liệu cho nhau nhưng chỉ có một node đóng vai trò là PAN out để giao tiếp với mạng PAN khác của hệ thống. truyền tin với tốc độ truyền tốc độ và kênh tùy chọn của dải kênh truyền với tần số trung tâm là 2.4GHz.

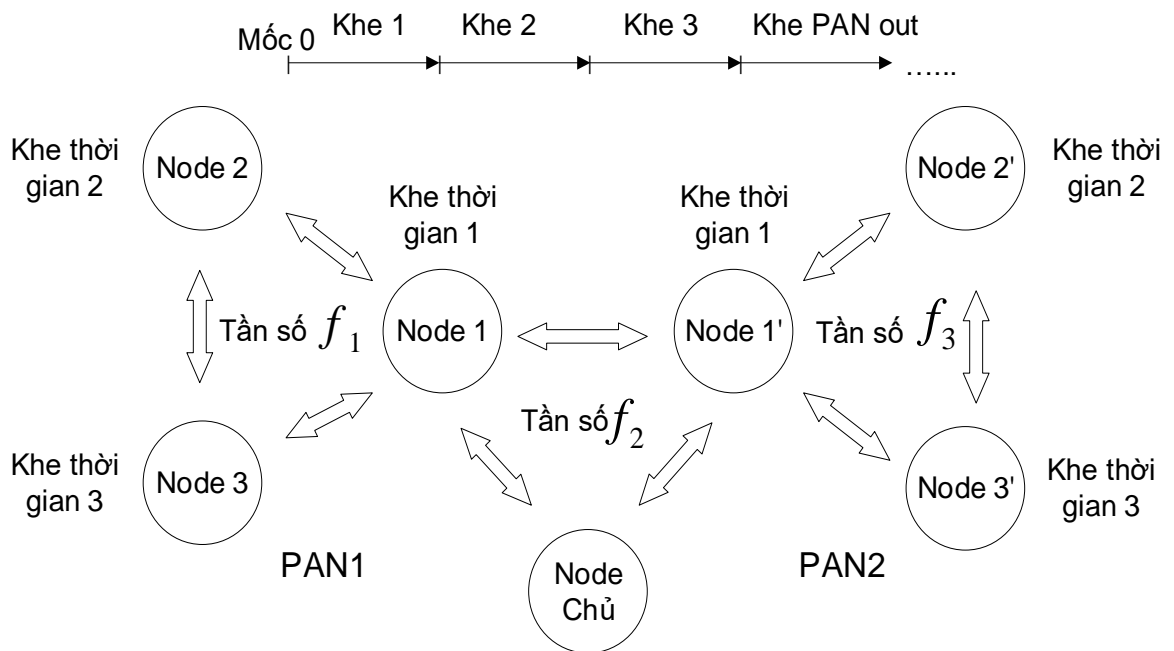


Hình 3.3 Hệ thống 2 mạng PAN trao đổi thông tin với nhau

### 3.3 Nội dung nâng cao

Cài đặt và lập trình điều khiển hệ thống mạng gồm 1 node chủ trao đổi thông tin với 2 node ra của 2 mạng PAN khác nhau, với mỗi mạng PAN gồm 3 node cảm biến, chức năng của các node: có thể truyền dữ liệu cho nhau nhưng

chỉ có một node đóng vai trò là PAN out để giao tiếp với mạng PAN khác và node chủ của hệ thống. truyền tin với tốc độ truyền tốc độ và kênh tùy chọn của dải kênh truyền với tần số trung tâm là 2.4GHz.



Hình 3.4 Hệ thống mạng Mesh tối giản

## PHẦN 4: TRÌNH TỰ THÍ NGHIỆM

Những trang thiết bị đã có sẵn tại phòng thí nghiệm phục vụ thực hiện bài thí nghiệm này gồm có:

- Mạch thí nghiệm Lập trình nhúng thời gian thực;
- Máy tính PC có cổng USB, kết nối mạng, dây nhảy để kết nối các mô-đun và cài đặt các công cụ: phần mềm PuTTY, Advanced IP Scanner, Filezilla Client.

### 4.1 Các thiết lập, cài đặt hệ thống chuẩn bị thí nghiệm

Phần này giúp người học nắm được các cài đặt hệ điều hành, làm quen với các phương thức kết nối, truy cập với bo mạch để lập trình điều khiển chúng thông qua ngôn ngữ lập trình Python.

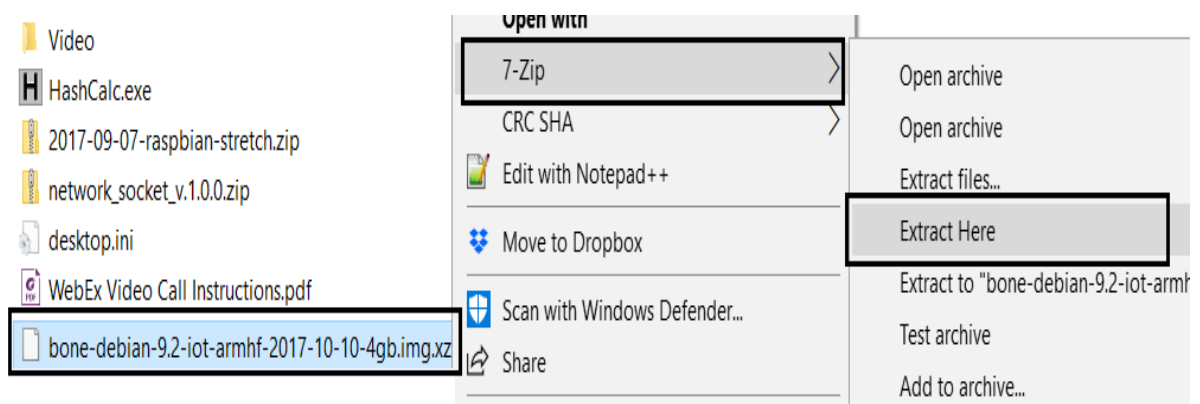
KIT được cài sẵn hệ điều hành Debian, xong cũng như các hệ điều hành khác, sẽ gặp lỗi sau một thời gian triển khai các phần mềm, hoặc đơn giản là chúng ta cần cập nhật hệ điều hành. Bài viết dưới đây sẽ hướng dẫn các bạn cách thức để cài đặt lại (hoặc nâng cấp cứng) hệ điều hành cho KIT; tất cả những hướng dẫn cài đặt, các phần mềm triển khai code trên KIT đều dựa trên Debian.

#### 4.1.1 Cài đặt hệ điều hành trên thẻ nhớ micro SD

**Bước 1:** Chuẩn bị thẻ nhớ microSDHC (Class 10) từ 4GB trở lên

**Bước 2:** Tải về bản cài đặt mới nhất của hệ điều hành dành cho thẻ nhớ

**Bước 3:** Giải nén file cài đặt mới được tải về



Hình 4.1 Giải nén file cài đặt hệ điều hành bằng phần mềm 7-Zip

**Bước 4:** Sử dụng SD Adapter hoặc đầu đọc thẻ để kết nối thẻ microSD với máy tính

**Bước 5:** Sử dụng phần mềm Etcher hoặc Win32Disk Imager để giải mã và ghi hệ điều hành lên thẻ nhớ

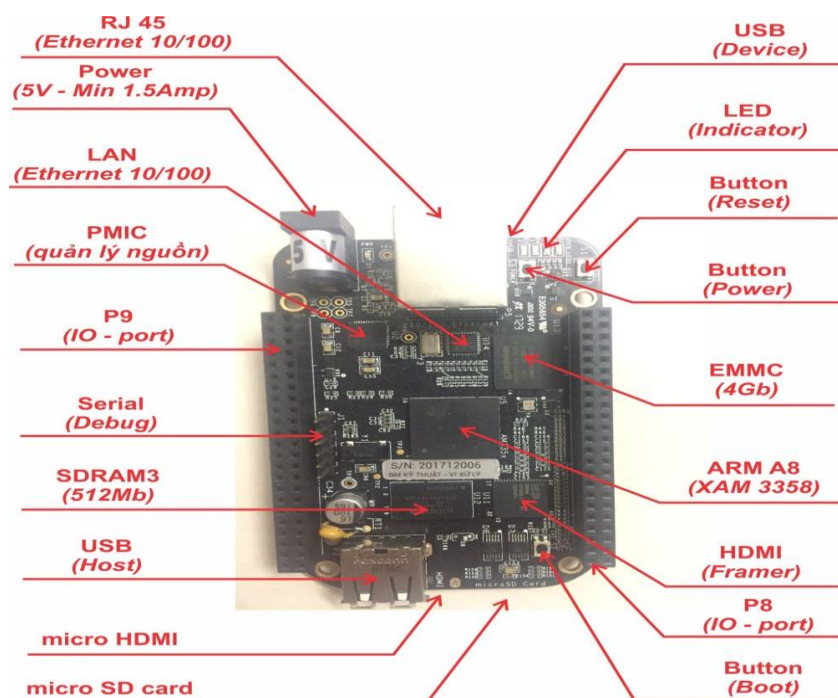


Hình 4.2 Phần mềm Etcher

**Bước 6:** Ngắt kết nối thẻ nhớ khỏi máy tính

**Bước 7:** Gắn thẻ nhớ vào khe thẻ của mô đun vi xử lý

**Bước 8:** Nhấn giữ nút BOOT và cấp nguồn cho mô đun. Tiếp tục nhấn giữ nút BOOT cho tới khi các User Led nhấp nháy.



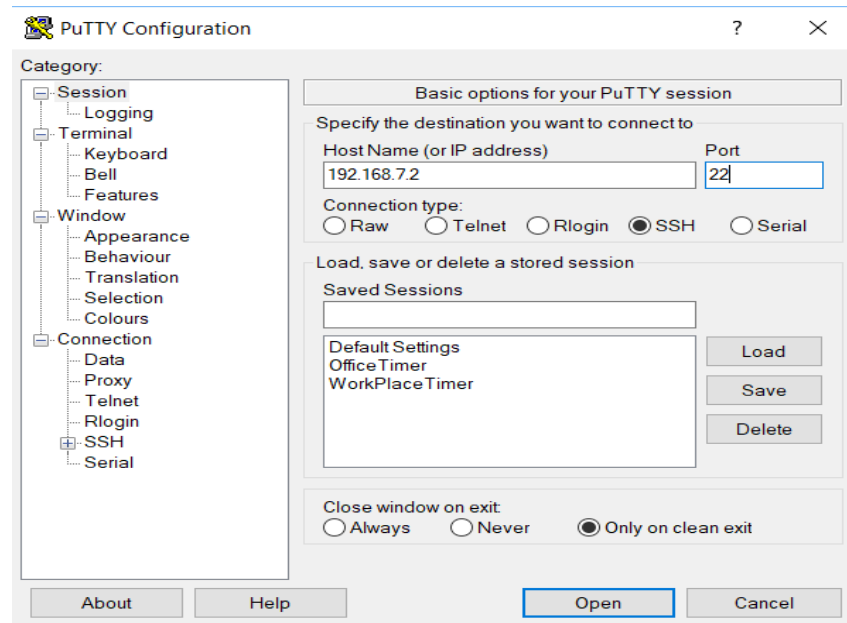
Hình 4.3 Vị trí nút nhấn BOOT và các User Led



### 4.1.2 Cài đặt hệ điều hành trên thẻ nhớ eMMC

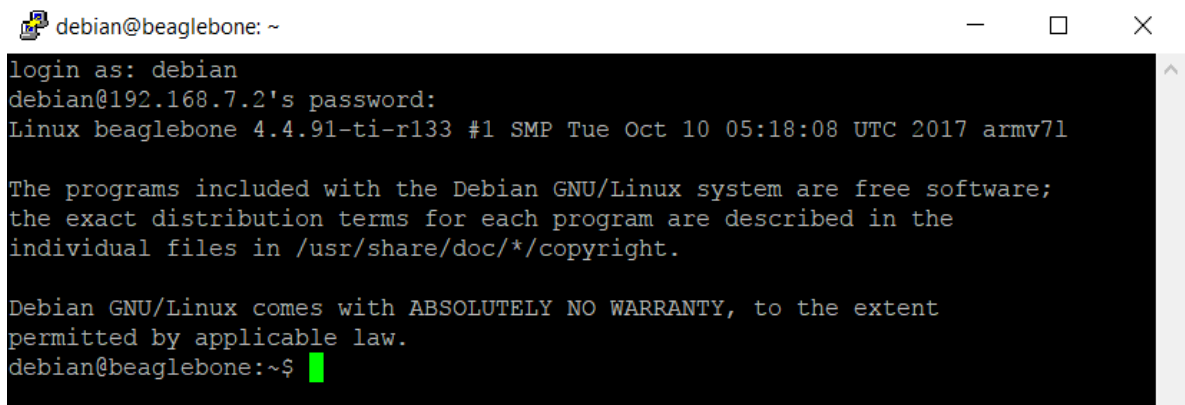
**Bước 1:** Tiến hành cài đặt hệ điều hành trên microSD trước khi cài đặt trên eMMC

**Bước 2:** Sử dụng PuTTY, truy cập vào địa chỉ **192.168.7.2** port: **22**



*Hình 4.4 Giao diện truy cập SSH từ PuTTY*

**Bước 3:** Đăng nhập với username/password: debian/temppwd



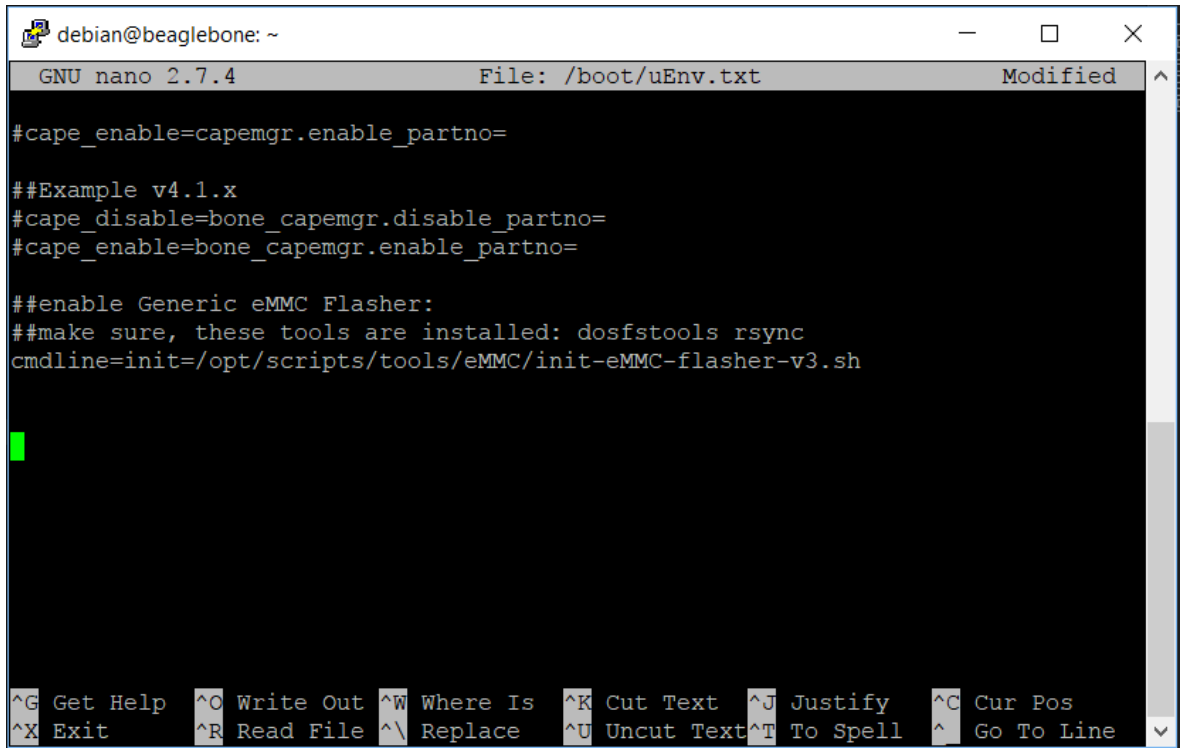
*Hình 4.5 Giao diện kết nối của PuTTY tới KIT*

**Bước 4:** Tìm và sửa file /boot/uEnv.txt:

*sudo nano /boot/uEnv.txt*

Bỏ dấu “#” tại dòng “`#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh`”

Sau đó nhấn “Ctrl + X” và “Enter” để hoàn tất việc sửa.



```
debian@beaglebone: ~
GNU nano 2.7.4 File: /boot/uEnv.txt Modified
#cape_enable=capemgr.enable_partno=

##Example v4.1.x
#cape_disable=bone_capemgr.disable_partno=
#cape_enable=bone_capemgr.enable_partno=

##enable Generic eMMC Flasher:
##make sure, these tools are installed: dosfstools rsync
cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

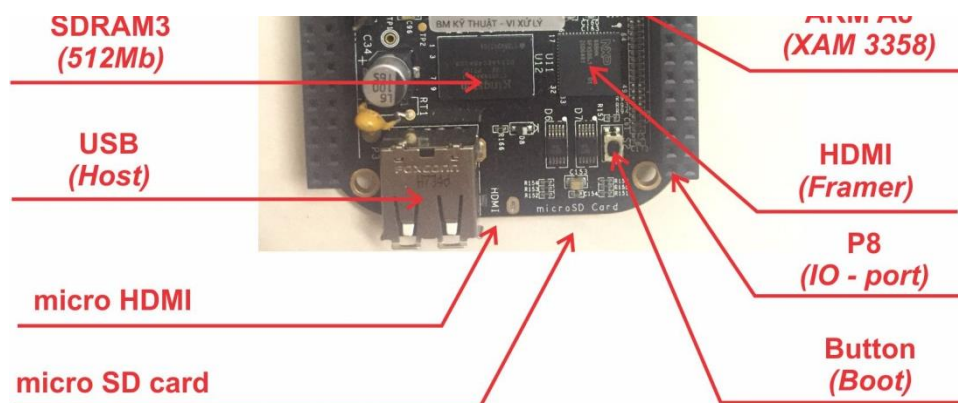
*Hình 4.6 Giao diện chỉnh sửa file cấu hình boot từ eMMC*

**Bước 4:** Rút nguồn của mô đun và cắm lại, lúc này hệ điều hành sẽ được ghi vào thẻ nhớ eMMC

**Bước 5:** Rút thẻ nhớ microSD khỏi khe thẻ, giờ mô đun sẽ chạy hệ điều hành Debian trên thẻ eMMC.

#### 4.1.3 Kết nối với bàn phím, chuột, USB Stick qua cổng USB

KIT hỗ trợ một cổng USB Host cho việc cắm các thiết bị ngoài hỗ trợ giao tiếp USB như bàn phím, chuột, thiết bị nhớ USB



*Hình 4.7 Vị trí cổng USB Host tại mô đun vi xử lý*

Nếu muốn sử dụng đồng thời nhiều thiết bị ngoại vi, ta cần phải có bộ mở rộng cổng USB. Chú ý là dòng cấp ra từ cổng USB Host của KIT rất nhỏ (100mA), nên muốn bảo vệ KIT và sử dụng các thiết bị tiêu thụ dòng lớn hơn thì bộ mở rộng cổng USB cần phải được cấp nguồn bên ngoài

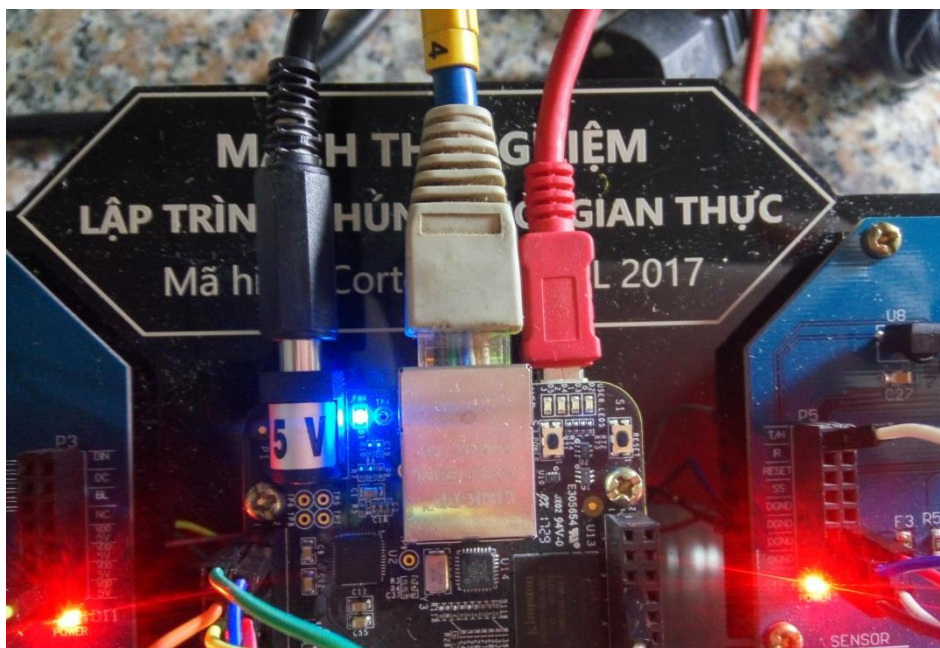


*Hình 4.8 Bộ mở rộng cổng USB có hỗ trợ nguồn bên ngoài*

#### **4.1.4 Kết nối Enternet**

Cổng Enternet giúp KIT có thể kết nối vào Internet để cập nhật phần mềm, cập nhật hệ điều hành và cài các gói thư viện phát triển.

Module Vi xử lý trên KIT có trang bị cổng 10/100 Enternet. Để kết nối, ta cắm dây mạng Cat 5e (hoặc Cat 6) từ KIT tới Switch chia mạng hoặc Router.



*Hình 4.9 Kết nối vào mạng thông qua cổng Enthernet*

#### **4.1.5 Kết nối HDMI với màn hình**

KIT hỗ trợ kết nối với màn hình thông qua cổng microHDMI. Nếu màn hình sử dụng cổng HDMI, thì ta cần có cổng chuyển đổi từ microHDMI sang HDMI



*Hình 4.10 Cổng chuyển đổi microHDMI sang HDMI*

Nếu màn hình sử dụng cổng VGA, ta cần có cổng chuyển đổi từ microHDMI sang VGA



*Hình 4.11 Cổng chuyển đổi từ microHDMI sang VGA*

Vị trí cắm nối như hình 4.12:



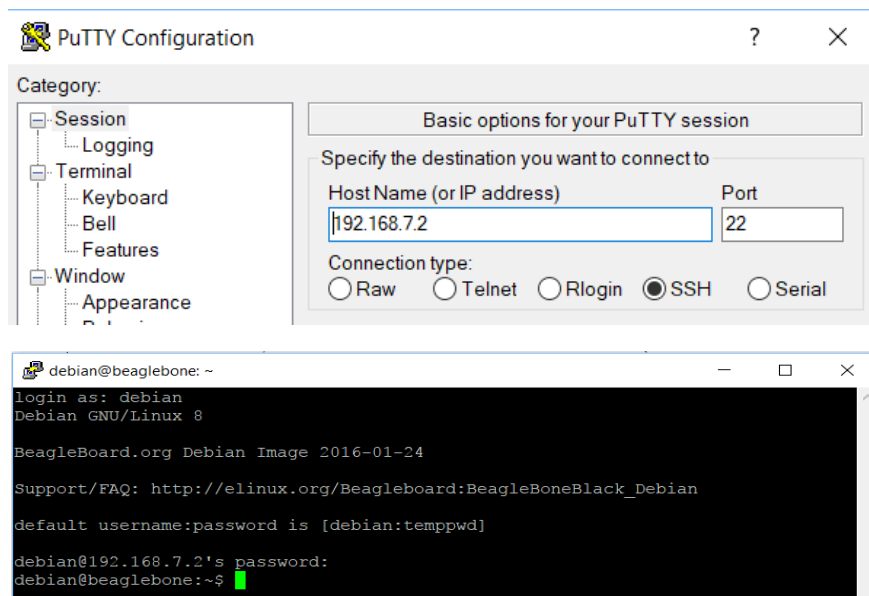
*Hình 4.12 Vị trí cắm cổng HDMI từ mô đun vi xử lý cho màn hình*

#### **4.1.6 Kết nối từ xa thông qua SSH**

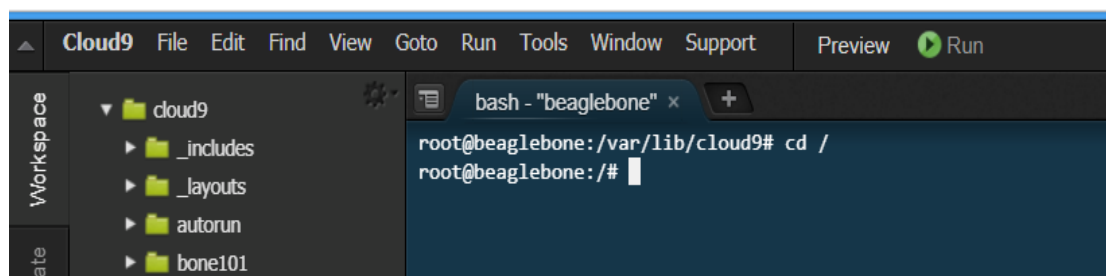
Việc thiết kế, chế tạo KIT dựa theo định hướng IoT, do đó, việc kết nối và điều khiển từ xa là rất quan trọng và cần thiết. KIT hỗ trợ giao thức SSH (Secure Shell) cho các kết nối bảo mật và dựa trên giao diện dòng lệnh. Bên cạnh đó, KIT còn hỗ trợ giao thức Remote Desktop của Microsoft hoặc VPN.

Trong khuôn khổ của tài liệu này, chúng ta chỉ đề cập trọng tâm vào giao thức SSH. Để sử dụng giao thức SSH, chúng ta cần phần mềm hỗ trợ giao thức này và một kết nối mạng theo TCP/IP (có thể là Local hoặc Internet).

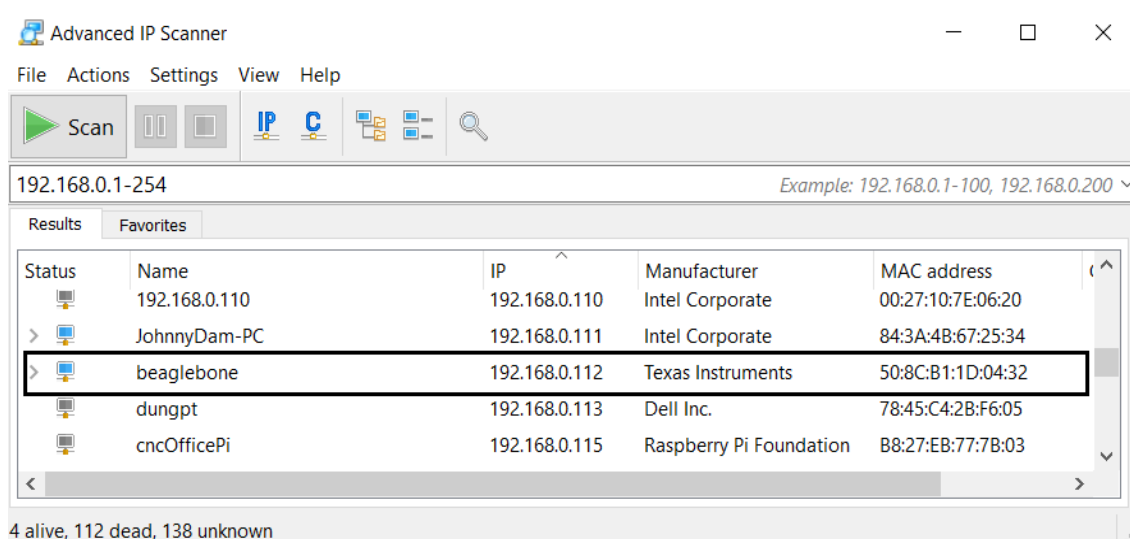
Hỗ trợ giao thức SSH có rất nhiều phần mềm, nhưng ở đây chúng ta sẽ sử dụng 2 phần mềm chủ yếu đó là PuTTY và Cloud 9.



Hình 4.13 Giao diện của phần mềm PuTTY



Hình 4.14 Giao diện của Cloud 9



Hình 4.15 Địa chỉ IP của KIT được quét bởi phần mềm Advanced IP Scanner

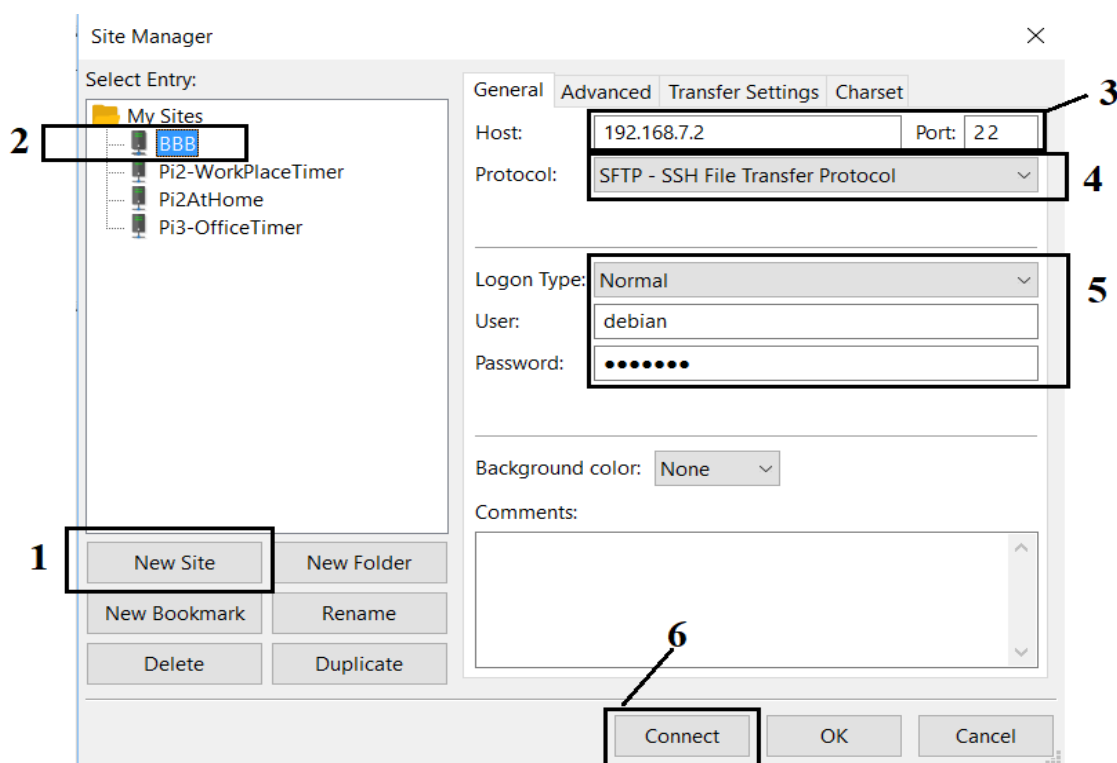
### Địa chỉ IP của KIT:

- Thông qua đường truyền USB: thì địa chỉ IP của KIT là 192.168.7.2

- Thông qua đường truyền Ethernet: Do địa chỉ IP trên mạng LAN hay là địa chỉ động, nên khi muốn kết nối với KIT, ta cần phải quét để biết địa chỉ IP của KIT. Trong tài liệu này, ta sử dụng phần mềm Advanced IP Scanner để quét địa chỉ IP trên mạng. Chú ý để quét được, máy tính của ta phải cùng mạng với KIT (cùng trong LAN).

#### 4.1.7 Kết nối truyền nhận file qua SFTP

KIT được thiết kế theo hướng IoT, vì vậy chúng ta làm việc đều thông qua giao diện dòng lệnh. Điều này sẽ gây khó khăn nhất định có những người mới tiếp cận. Để thuận tiện hơn trong việc tạo mới, sao chép, di chuyển các file cài đặt, file chương trình hoặc thư viện, chúng ta sử dụng giao thức truyền file SFTP (Secure File Transfer Protocol) và sử dụng phần mềm Filezilla Client để kết nối và truyền nhận



Hình 4.16 Thiết lập các thông số kết nối SFTP từ FileZilla

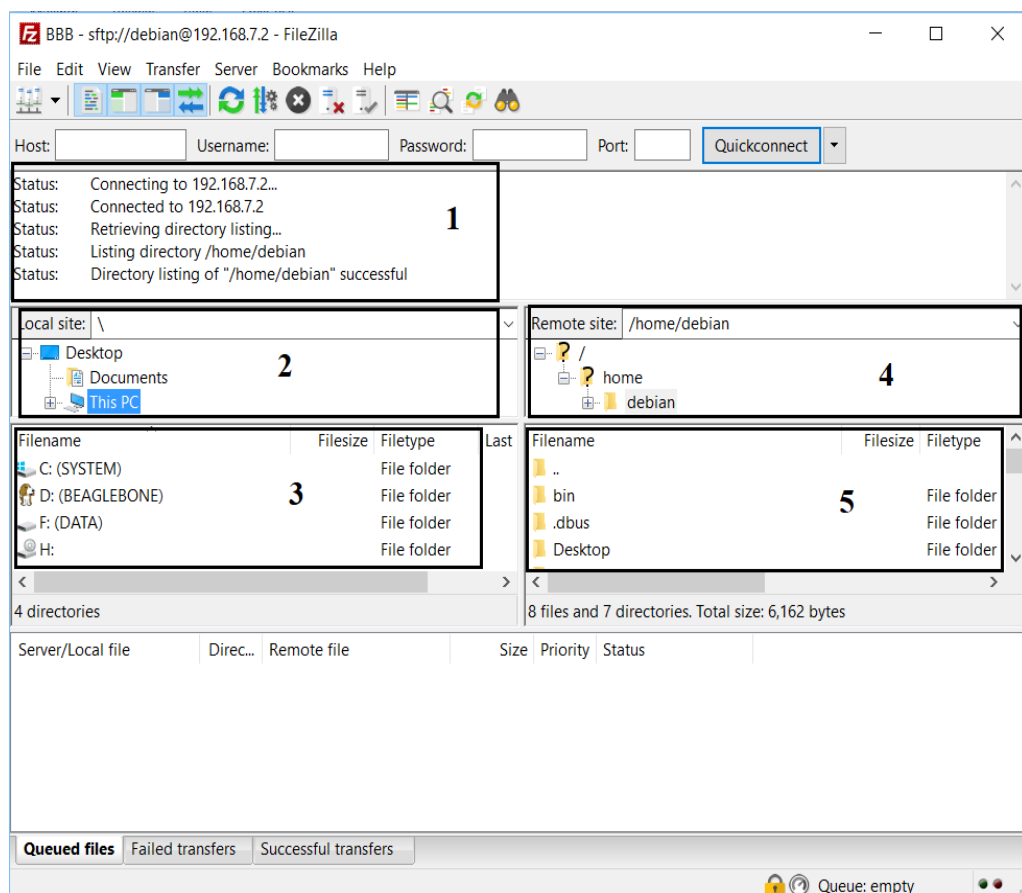
#### Các bước cấu hình và sử dụng:

**Bước 1:** Từ giao diện chính của FileZilla, chọn menu File > Site Manager

**Bước 2:** Thiết lập các thông số:



- ✚ 1: Chọn New Site để tạo mới thiết lập
- ✚ 2: Lưu tên Site, chú ý lưu với tên ngắn gọn, dễ nhớ và không dấu
- ✚ 3: Nhập vào địa chỉ hiện thời của KIT và Port là 22. Nếu dùng kết nối USB thì địa chỉ của KIT luôn là 192.168.7.2
- ✚ 4: Chọn giao thức SFTP
- ✚ 5: Thiết lập kiểu đăng nhập
  - Logon: Normal
  - User: debian (mặc định)
  - Password: temppwd (mặc định)
- 6: Kết thúc cài đặt và tiến hành kết nối, nhấn Connect



*Hình 4.17 Giao diện kết nối của chương trình FileZilla*

### **Bước 3:** Làm quen với giao diện chương trình

- ✚ 1: Trạng thái kết nối. Thông báo kết nối hoặc mất kết nối, các thao tác đã thực hiện
- ✚ 2 và 3: Cây thư mục và chi tiết thư mục tại máy Local



- ✚ 4 và 5: Cây thư mục và chi tiết thư mục tại máy được kết nối đến (Remote)

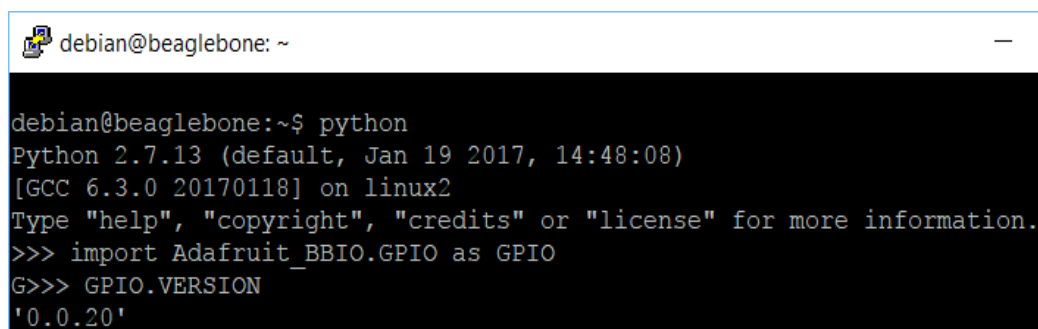
#### **Bước 4: Truyền nhận**

- ✚ Truyền từ máy local đến máy remote bằng cách kéo file hoặc thư mục từ phía local sang phía máy remote (vùng 3 sang vùng 5)
- ✚ Nhận từ máy remote về máy local bằng cách kéo file hoặc thư mục từ phía máy remote sang máy local (vùng 5 sang vùng 3).

#### **4.1.8 Giới thiệu về Python**

Python là một ngôn ngữ tuyệt vời và mạnh mẽ đó là dễ dàng để sử dụng (dễ đọc và viết) và cho phép bạn kết nối với dự án của bạn với thế giới thực. Cú pháp Python rất sạch sẽ, nó nhấn mạnh về khả năng đọc và sử dụng từ khóa tiếng Anh chuẩn. Python hiện tại đang có 2 phiên bản được sử dụng song song là 2.7 và 3. Phiên bản Python 2.7 là phiên bản mặc định trên hệ điều hành Debian.

**Cài đặt Python :** Phiên bản mới nhất của hệ điều hành Debian đã được cài đặt sẵn thư viện giao tiếp phần cứng GPIO. Chúng ta có thể kiểm tra xem phiên bản Python và GPIO bằng lệnh trong Terminal như sau: “\$sudo python”



```
debian@beaglebone: ~  
debian@beaglebone:~$ python  
Python 2.7.13 (default, Jan 19 2017, 14:48:08)  
[GCC 6.3.0 20170118] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import Adafruit_BBIO.GPIO as GPIO  
G>>> GPIO.VERSION  
'0.0.20'
```

*Hình 4.18 Phiên bản của gói thư viện Adafruit\_BBIO.GPIO*

Phiên bản hiện tại của Adafruit\_BBIO.GPIO là 0.0.20. Nếu bạn cần cập nhật phiên bản mới thì chạy lệnh: “sudo apt-get update” sau đó chạy lệnh “sudo apt-get upgrade”.

#### **4.1.9 Kích hoạt truyền thông UART trên KIT**

KIT hỗ trợ 5 cổng UART, trong đó cổng UART3 chỉ có đường truyền TX, không có đường nhận RX.

UART	RX	TX	CTS	RTS	Device
UART1	P9_26	P9_24	P9_20	P9_19	/dev/ttyO1 hoặc /dev/ttyS1
UART2	P9_22	P9_21			/dev/ttyO2 hoặc /dev/ttyS2
UART3		P9_42	P8_36	P8_34	/dev/ttyO3 hoặc /dev/ttyS3
UART4	P9_11	P9_13	P8_35	P8_33	/dev/ttyO4 hoặc /dev/ttyS4
UART5	P8_38	P8_37	P8_31	P8_32	/dev/ttyO5 hoặc /dev/ttyS5

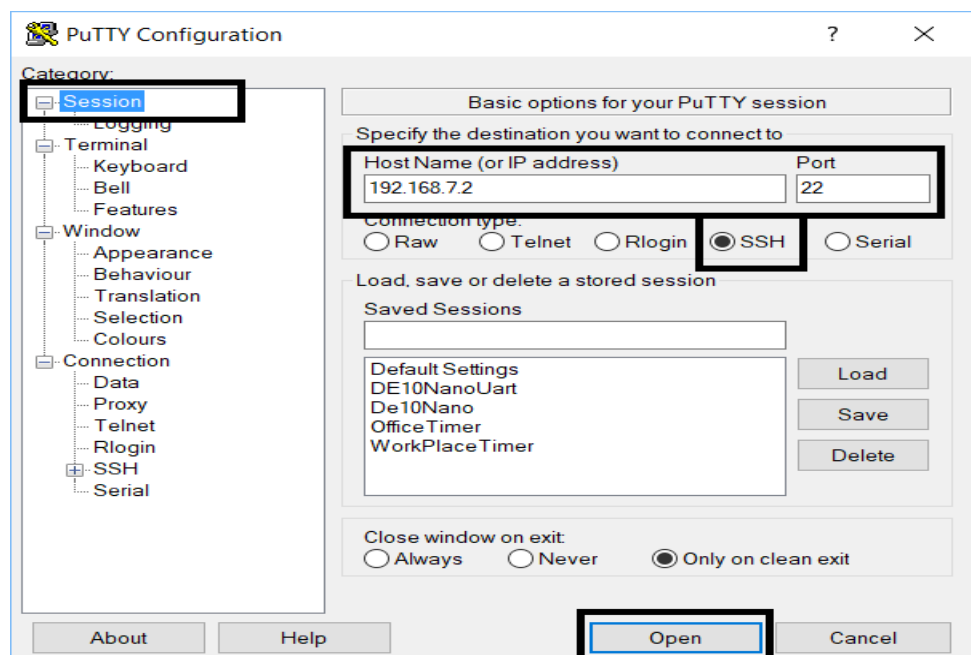
*Bảng 4.1 Các kênh UART của KIT*

Trong nội dung của bài này, ta chỉ kích hoạt UART1 trên KIT mà thôi.

**Bước 1: Sử dụng PuTTY (SSH) để truy cập vào KIT**

- Cắm cáp USB từ máy tính vào KIT. Chú ý là máy tính đã cài sẵn Driver để giao tiếp với KIT.
- Thiết lập thông số SSH trong PuTTY theo địa chỉ 192.168.7.2 và Port

22



*Hình 4.19 Thiết lập SSH cho PuTTY để truy cập vào KIT*

- Sử dụng login as: **debian**, password: **temppwd**

**Bước 2:** Sửa file cấu hình của hệ thống **uEnv.txt**

- Trong PuTTY, đánh dòng lệnh: `sudo nano /boot/uEnv.txt`
- Sửa theo hình dưới đây:

```
##Example v4.1.x
#cape disable=bone capemgr.disable partno=
#cape enable=bone capemgr.enable partno=BB-UART1

##enable Generic eMMC Flasher:
##make sure, these tools are installed: dosfstools rsync
#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

Hình 4.20 Sửa file cấu hình hệ thống **uEnv.txt** để kích hoạt UART

**Bước 3:** Khởi động lại hệ thống để lưu các thiết lập, dùng lệnh:

`sudo reboot -n`

**Bước 4:** Truy cập lại KIT thông qua PuTTY (theo bước 1)

**Bước 5:** Kiểm tra trạng thái kích hoạt của UART, dùng lệnh:

`cat /sys/devices/platform/bone_capemgr/slots`

```
debian@beaglebone:~$ cat /sys/devices/platform/bone_capemgr/slots
0: PF---- -1
1: PF---- -1
2: PF---- -1
3: PF---- -1
4: P-O-L- 0 Override Board Name,00A0,Override Manuf,BB-UART1
```

Hình 4.21 UART đã được kích hoạt thành công

**Chú ý:** Nếu hệ thống báo không tìm thấy slots, ta cần vô hiệu hóa u-boot Overlay trong file **uEnv.txt**

```
###U-Boot Overlays###
###Documentation: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian#
###Master Enable
#enable uboot_overlays=1
###
###Override capes with eeprom
#uboot_overlay_addr0=/lib/firmware/<file0>.dtbo
#uboot_overlay_addr1=/lib/firmware/<file1>.dtbo
#uboot_overlay_addr2=/lib/firmware/<file2>.dtbo
#uboot_overlay_addr3=/lib/firmware/<file3>.dtbo
```

Hình 4.22 Vô hiệu hóa **uboot\_overlays** để có thể truy cập vào slots

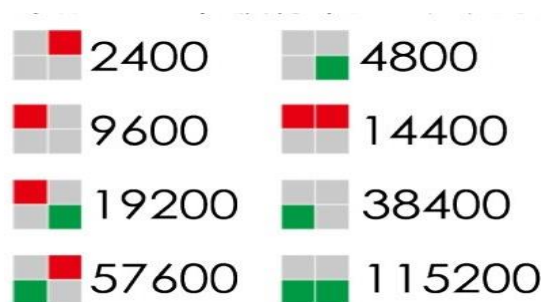
**Bước 6:** Sử dụng các lệnh sau để cài đặt thư viện cho UART:

- `sudo apt-get update -y`
- `sudo apt-get install build-essential python-dev python-setuptools python-pip python-smbus`
- `pip install pyserial`

## 4.2 Thiết lập chế độ hoạt động cho mô đun Zigbee CC2530

### Bước 1: Thiết lập tốc độ Baud (Baud rate)

- Không cấp nguồn, nhấn giữ nút key, sau đó cấp nguồn.
- 4 led trên module sẽ nhấp nháy báo hiệu truy cập chế độ thiết lập cấu hình.
- Thả nút nhấn ra, rồi lại nhấn nút tuần tự để chọn mức baudrate phù hợp theo hình dưới.



Hình 4.23 Chỉ thị Led tương ứng với tốc độ Baud

### Bước 2: Chọn kênh truyền

- Sau khi chọn Baudrate xong, nhấn giữ nút ấn để sang bước 2
- 4 led trên module sẽ nhấp nháy báo hiệu truy cập bước 2
- Thả nút nhấn ra, rồi lại nhấn nút tuần tự để chọn kênh (Chanel). Lưu ý để 2 module truyền được cho nhau thì cần cài đặt chọn kênh giống nhau.

### Bước 3: Chọn chế độ truyền dẫn Point to point hay Broadcast

- Sau khi chọn Chanel xong, nhấn giữ nút ấn để sang bước 3
- 4 led trên module sẽ nhấp nháy báo hiệu truy cập bước 3
- Thả nút nhấn ra, rồi lại nhấn nút tuần tự để chọn chế độ truyền dẫn
- Nó có 2 chế độ truyền dẫn:

+ **Point to Point:** Dùng để truyền nhận giữa 2 mô đun cho nhau. Ta có thể lựa chọn địa chỉ, tránh trùng với mô đun khác ở 16 địa chỉ khác nhau. Ở mô đun thứ nhất chọn Cấu hình Point to Point A.



Cấu hình Point to Point A

Cũng ở bước này, ở module thứ 2 chọn cấu hình Poin to Point B.



Cấu hình Point to Point B

+ **Broadcast:** Người dùng có thể tạo ra một mạng lưới truyền dẫn giữa các node để tạo ra một nhà mạng cho riêng mình.

Lưu ý tất cả các module cần được cấu hình Chanel và Broadcast giống nhau, khi 1 mô đun truyền, tất cả các mô đun còn lại sẽ nhận, chức năng của các mô đun là tương đương.

#### **Bước 4:** Xác nhận lưu và hoàn tất

- Sau khi chọn kiểu truyền nhận xong, nhấn tỉ nút ấn để sang bước 4.
- 4 led trên module sẽ nhấp nháy báo hiệu cài đặt thành công và được lưu mãi mãi.
- Lưu ý thiết đặt chỉ được lưu và có hiệu lực khi đến được bước 4.
- Để thiết đặt lại hãy rút nguồn và quay về bước 1.

### **4.3 Lập trình truyền nhận không dây sử dụng mô đun CC2530**

Thực hiện nội dung thí nghiệm của mục 3.2: Cài đặt và lập trình điều khiển hệ thống mạng gồm 2 PAN trao đổi thông tin với nhau, với mỗi mạng PAN gồm 3 node cảm biến, chức năng của các node: có thể truyền dữ liệu cho nhau nhưng chỉ có một node đóng vai cho là PAN out để giao tiếp với mạng PAN khác của hệ thống. truyền tin với tốc độ truyền tốc độ và kênh tùy chọn của dải kênh truyền với tần số trung tâm là 2.4GHz.

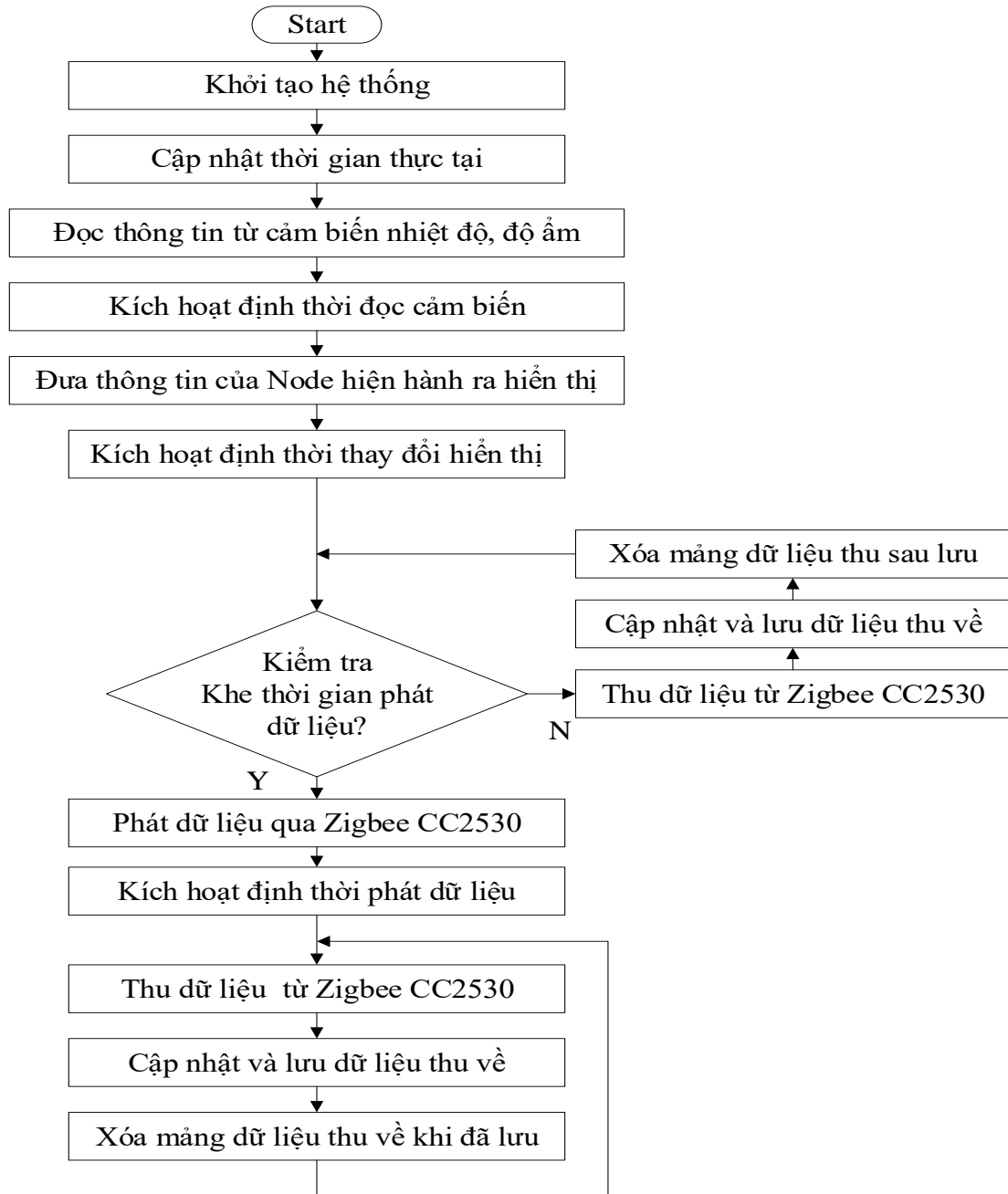
**Bước 1:** Cắm kết nối USB từ máy tính tới mô đun Vi xử lý của KIT

**Bước 2:** Mở trình duyệt web (không sử dụng Internet Explorer)

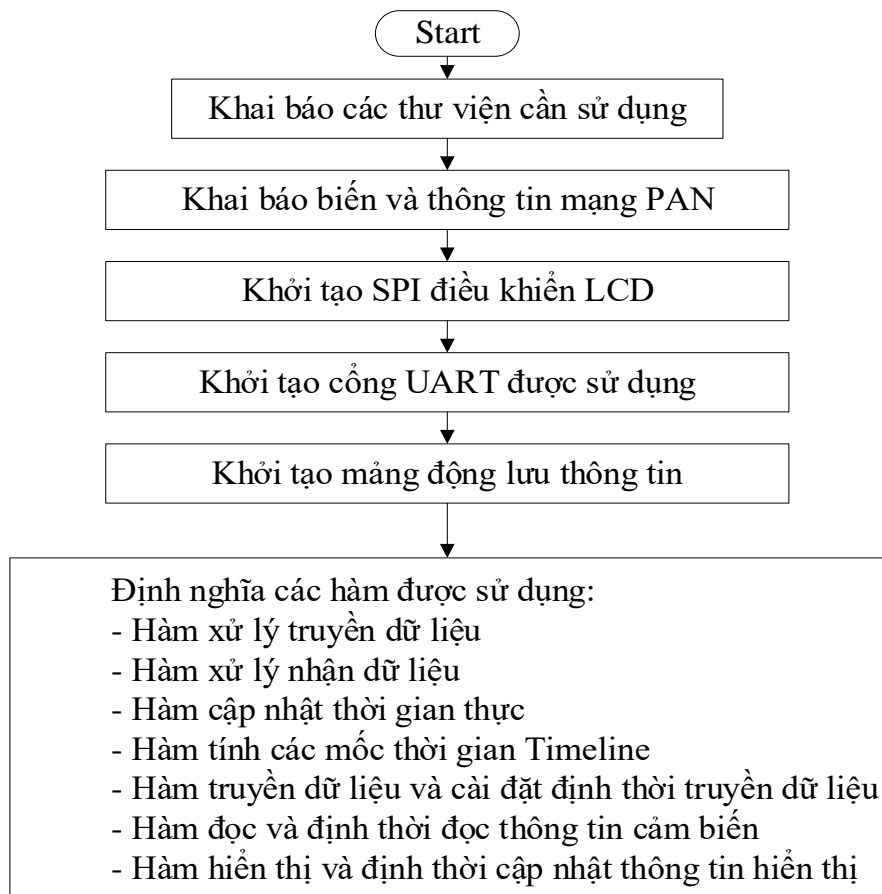
**Bước 3:** Truy cập vào địa chỉ **192.168.7.2 :3000** để mở Cloud 9

**Bước 4:** Tại Cloud 9, tạo file mới tên là **ZigbeeUart.py** và lưu vào thư mục **ThucHanh**

**Bước 5:** Lập trình viết code vào khung soạn thảo của Cloud 9 (Code có thể tham khảo Phụ lục B): Trình tự viết code cụ thể gồm 2 bước: Trước hết tiến hành phân tích yêu cầu bài toán, đưa ra phương án thực hiện nhiệm vụ của bài toán (thông qua lưu đồ thuật toán tổng quát và chi tiết nếu cần):



*Hình 4.24 Lưu đồ thuật toán tổng quát về hệ thống 2 mạng PAN trao đổi thông tin với nhau*



*Hình 4.25 Lưu đồ thuật toán phần khởi tạo của hệ thống 2 mạng PAN trao đổi thông tin với nhau*

- Chương trình có thể thông qua việc nhập tham số đầu vào để thay đổi được tổng số node mạng của mạng PAN, địa chỉ mạng PAN (PAN\_ID), địa chỉ ID của node mạng hiện hành, khe thời gian mà node mạng hiện hành phát dữ liệu và thời gian mà các PAN\_out phát dữ liệu ra ngoài mạng nội bộ của PAN đó.

- Thông qua các hàm của thư viện datetime để tính khoảng thời gian so với mốc thời gian 0 giờ, 0 phút, 0 giây. Sau đó dùng công thức:

$$T_n = t_0 \% (m * t_d)$$

Để xác định thời điểm hiện tại cách mốc thời gian của khe thời gian đầu tiên là bao nhiêu thời gian. Trong đó:  $T_n$  : là khoảng cách thời gian so với mốc phát của

khe thời gian đầu tiên;  $t_0$  : khoảng thời gian so với mốc 0 giờ, 0 phút, 0 giây;  
 $m$  : tổng số khe thời gian;  $t_d$  : độ lớn của một khe thời gian.

- Căn cứ vào thời gian  $T_n$  có thuộc khe thời gian của Node hiện hành và thỏa mãn điều kiện sau đây thì tiến hành phát dữ liệu của Node hiện hành lần đầu tiên đồng thời kích hoạt bộ định thời Timer để định thời phát dữ liệu trong khe thời gian này:

$$\begin{cases} T_n \geq t_d * (n-1) \\ T_n \leq t_d * (n-0.6) \end{cases}$$

Trong đó  $n$  là số thứ tự ID của Node hiện hành.

- Tận dụng timer cài đặt hẹn giờ đọc cảm biến, thay đổi thông tin hiển thị trên LCD.

- Bài toán phụ thuộc vào các tham số đầu vào mà quyết định mở 1 hay 2 cổng UART; hàm truyền sẽ bao gồm 4 thông số: Địa chỉ ID node truyền (hoặc PAN), giá trị nhiệt độ, độ ẩm và giá trị tổng 3 tham số trên dùng làm căn cứ kiểm tra dữ liệu nhận được có bị lỗi hay không; hàm nhận dữ liệu sẽ căn cứ vào hàm nhận của thư viện Serial tiến hành nhận và tách từng thông số của dữ liệu nhận về, thông qua kiểm tra lỗi và các hàm điều kiện để nhập thông tin vào các vùng dữ liệu tương ứng của các Node mạng.

- Cuối cùng dựa vào lưu đồ thuật toán và các phân tích trên tiến hành hoàn thiện viết code điều khiển hệ thống.

#### **Bước 6: Đấu nối:**

- Nối lần lượt các chân P9\_26 (RX) và P9\_24 (TX) của Mô đun vi xử lý sang lần lượt các chân TX1 và RX1 của Mô đun Zigbee thứ nhất

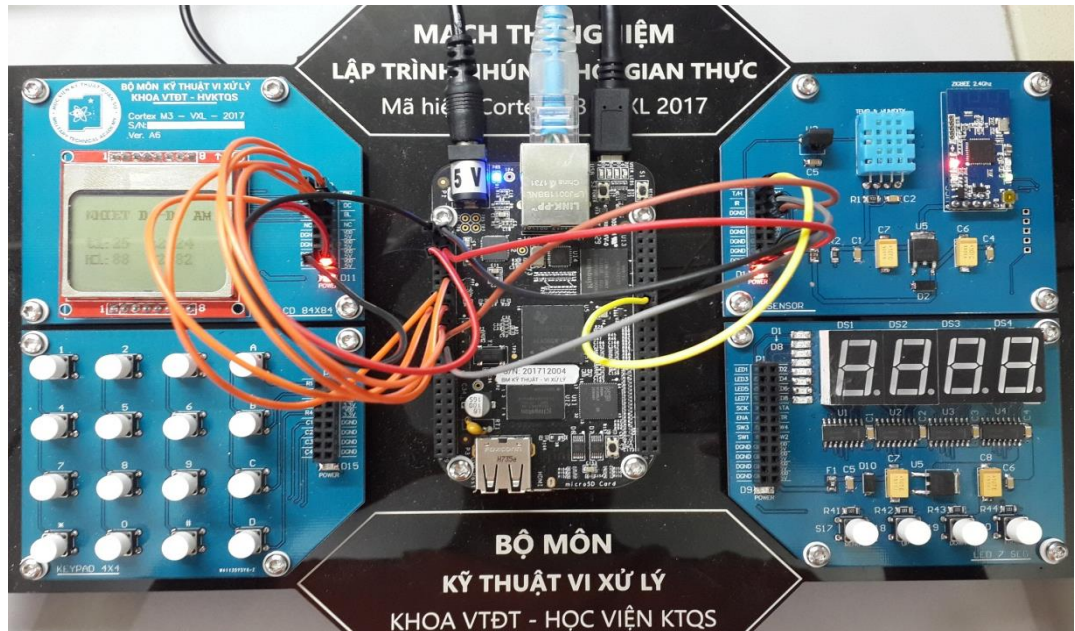
- Nối chân 15 của P8 sang chân số 1 (T/H) của khối Mô đun cảm biến nhiệt độ, độ ẩm thứ nhất

- Nối lần lượt các chân P8\_38 (RX) và P8\_37 (TX) của Mô đun vi xử lý sang lần lượt chân TX1 và RX1 của Mô đun Zigbee thứ hai

- Nối lần lượt các chân P9\_12,15,17,18,22 của Mô đun vi xử lý sang lần lượt các chân RST, DC, CS, DIN, CLK của khối GLCD



- Nối chân 1, 2 (DGND) của P9 của Mô đun vi xử lý sang chân DGND của khối GLCD và Mô đun Zigbee 1 và 2
- Nối chân 5, 6 (VDD\_5V) của P9 của Mô đun vi xử lý sang chân VDD5V của khối GLCD và hai Mô đun Zigbee.



*Hình 4.26 Sơ đồ kết nối sử dụng Mô đun Zigbee CC2530*

#### **Bước 7:** Thực thi chương trình

- Sử dụng nút **Start** để bắt đầu thực thi chương trình, theo dõi giá trị trả về của phím nhấn trên khung thực thi của Cloud 9
- Sử dụng nút **Stop** để dừng chương trình

## **PHẦN 5: BÁO CÁO THÍ NGHIỆM**

Giáo viên đưa ra các yêu cầu chung về sử dụng trang thiết bị và các kỹ năng, kết quả thí nghiệm cần đạt được cho toàn lớp. Trong quá trình thực hiện các bước thí nghiệm nêu trên, học viên tiến hành làm các nội dung giáo viên yêu cầu.

Khi chương trình chạy ra kết quả, báo cáo giáo viên hướng dẫn để giáo viên hướng dẫn đánh giá trực tiếp kết quả của chương trình đó và năng lực của học viên. Đồng thời học viên viết báo cáo theo nhóm (2-3 học viên) theo mẫu ở Phụ lục A để đánh giá tổng thể về kết quả đạt được sau khi thực hiện bài thí nghiệm.

Lưu ý trong phần kết quả của báo cáo cần nêu rõ các thông tin sau: Sơ đồ cấu trúc mạng truyền tin, tốc độ truyền, độ lớn khe thời gian và tần số truyền, các tham số động của hệ thống, lưu đồ thuật toán thực hiện yêu cầu bài thí nghiệm, các vấn đề gặp trong quá trình thực hiện bài toán và lập trình chạy thử, hiệu chỉnh trên mạch.

## PHỤ LỤC A: MẪU BÁO CÁO THÍ NGHIỆM

KHOA VTĐT  
BỘ MÔN KT XS-VXL

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM  
Độc lập - Tự do - Hạnh phúc

*Hà Nội, ngày ... tháng ... năm 20....*

### BÁO CÁO THÍ NGHIỆM

#### I. NHỮNG THÔNG TIN CHUNG

1. Tên bài thí nghiệm:.....
2. Thuộc môn học: .....
3. Học viện thực hiện: .....
4. Lớp: ..... Khoa: .....
5. Giáo viên phụ trách: .....
6. Thời gian tiến hành thực hiện: .....

#### II. MỤC ĐÍCH YÊU CẦU VÀ CÁCH TIẾN HÀNH:

1. Mục đích: .....  
.....
2. Yêu cầu: .....  
.....
3. Nội dung: .....  
.....
4. Cách tiến hành: .....  
.....

#### III. ĐIỀU KIỆN, DỤNG CỤ, THIẾT BỊ VÀ VẬT CHẤT THÍ NGHIỆM/TH:

1. Mô tả điều kiện thí nghiệm/TH:.....  
.....
2. Dụng cụ thí nghiệm/TH: .....

.....  
3. Thiết bị thí nghiệm/TH: .....

.....  
4. Vật chất thí nghiệm/TH: .....

.....  
**IV. ĐÁNH GIÁ, XỬ LÝ KẾT QUẢ THÍ NGHIỆM/TH:**

1. Kết quả thí nghiệm/TH: .....

.....  
*(Trình bày kết quả thí nghiệm dạng lời và Bảng số liệu quy định theo từng bài thí nghiệm)*

2. Đánh giá, xử lý kết quả thí nghiệm/TH:.....

.....  
**V. KẾT LUẬN, KIẾN NGHỊ, ĐỀ XUẤT:**

.....  
**Giáo viên phụ trách**

(Ký, ghi rõ họ tên)

**Người viết báo cáo**

(Ký, ghi rõ họ tên)

## PHỤ LỤC B: CODE THAM KHẢO

(Các nội dung thí nghiệm Phần 3 đi từ dễ đến khó, nội dung sau là phát triển của nội dung trước, chính bởi vậy code tham khảo chỉ đưa ra code của những nội dung quan trọng và tiêu biểu nhất)

### Code nội dung 3 của mục 3.1

```
# Khai bao thu vien can su dung
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.UART as UART
import Adafruit_Nokia_LCD as LCD
import Adafruit_GPIO.SPI as SPI
import Adafruit_DHT
import serial
import datetime
import time
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
#===== Khai bao Hardware SPI =====
DC = "P9_15"
RST = "P9_12"
SPI_PORT = 1
SPI_DEVICE = 0
# ===== Khoi tao SPI dieu khien LCD =====
disp = LCD.PCD8544(DC, RST, spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE,
max_speed_hz=4000000))
disp.begin(contrast=40)
disp.clear()
disp.display()
image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))
draw = ImageDraw.Draw(image)
```

```

draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT), outline=255,
fill=255)
font = ImageFont.load_default()
draw.text((0,0), 'NHIET DO-DO AM', font=font)
disp.image(image)
disp.display()
# ===== Khoi tao UART =====
ser = serial.Serial(port = "/dev/ttyS1", baudrate=9600)
ser.close()
ser.open()
if ser.isOpen():
    print ("Serial is open!")
#===== Khai bao frame truyen cua UART=====
uartTX = [0, 0, 0, 0] # uartTX = [Index, temp, humi, checksum]
uartRX = [0,0,0,0,0,0,0,0,0,0]
strRX = ""
numOfSent = 4    # So du lieu truyen di
indexRX = 2      # Dia chi cua thiet bi
#===== Khai bao bien nhiet do, do am =====
t1 = 0
h1 = 0
t2 = 0
h2 = 0
t1Old = 0
h1Old = 0
t2Old = 0
h2Old = 0
#===== Khai bao chan cua cam bien DHT11 =====
sensorPin = "P8_15" #T/H Pin
#Khai bao loai cam bien
sensorType = Adafruit_DHT.DHT11
# ===== Ham xu ly doc nhiet do =====

```

```

def processDHT11():
    global t1, t1Old, h1, h1Old # Su dung bien Global
    #Doc nhiet do, do am tu cam bien DTH11
    h1, t1 = Adafruit_DHT.read_retry(sensorType, sensorPin)
    #Kiem tra nhiet do, do am co thay doi so voi gia tri cu hay khong, neu k thi
    khong cap nhat len LCD Nokia
    if((t1 != t1Old) or (h1 != h1Old)):
        t1Old = t1
        h1Old = h1
        # Xoa gia tri cu de ghi vao gia tri moi
        draw.rectangle((16,18,30,38), outline=255, fill=255)
        disp.image(image)
        disp.display()
    # Kiem tra gia tri tra ve tu cam bien (do _am va nhiet_do) khac NULL
    if (h1 is not None) and (t1 is not None):
        print ("t1 = {0:0.1f} H1 = {1:0.1f}\n").format(t1, h1);
        draw.text((0,18), 't1:%0.2s' %(t1), font=font)
        draw.text((0,28), 'H1:%0.2s' %(h1), font=font)
        disp.image(image)
        disp.display()
    else:
        print("Loi khong the doc tu cam bien DHT11 :(\n")
    return

# ===== Ham xu ly truyen =====
def processTX(indexTX = 0):
    global uartTX # Su dung bien global
    uartTX[0] = indexTX
    uartTX[1] = int(t1)
    uartTX[2] = int(h1)
    uartTX[3] = uartTX[0] + uartTX[1] + uartTX[2]
    strToSend = str(uartTX[0]) + ":" + str(uartTX[1]) + ":" + str(uartTX[2]) +
    ":" + str(uartTX[3]) + "\n"

```

```

        ser.write(strToSend)
    return

# ===== Ham xu ly nhan =====
def processRX():
    global h2, t2, strRX    # Cho phép thay doi gia tri cua bien toan cuc
    (Global)
    checksum = 0
    if(ser.inWaiting()):    # Doc chuoi nhan ve
        strRX = ser.readline()
    # Chuyen ky tu xuong dong "\n" thanh ":" truoc khi tach
    strRX = strRX.replace("\n", "")
    strRxData = strRX.split(':')    # Tach gia tri luu tru thoi gian
    strRxData = filter(None, strRxData)    # Xoa cac ki tu Null
    # Chuyen doi tu mang ky tu sang mang so nguyen
    # Bay loi truyen sai ky tu khong phai so nguyen
    try:
        arrRxData = [int(strNumber) for strNumber in strRxData]
        print arrRxData
        #Kiem tra xem so phan tu nhan ve co bang so truyen di hay khong
        lenRX = len(arrRxData)
        if(lenRX == numOfSent):
            for i in range(0, lenRX-1):    # Tinh checksum
                checksum = checksum + arrRxData[i]
            # Kiem tra checksum co bang gia tri cuoi cung khong
            if(checksum == arrRxData[numOfSent - 1]):
                if(arrRxData[0] == indexRX):#Kiem tra IndexRX
                    t2 = arrRxData[1]
                    h2 = arrRxData[2]
                    nokiaTHDisplay()
            else:
                print("Ma kiem tra loi khong khop")
        else:

```



```

        print("Qua trinh truyen nhan bi loi")
        # Xoa gia tri cu de ghi vao gia tri moi
        draw.rectangle((58,18,72,38), outline=255, fill=255)
        disp.image(image)
        disp.display()
#        time.sleep(0.2)
    except ValueError:
        print("Du lieu nhan duoc xay ra dot bien")
        print strRxData
    return

# ===== Ham xu ly hien thi =====

def nokiaTHDisplay():
    # Cho phep thay doi gia tri cua bien toan cuc (Global)
    global t2Old, h2Old
    #Kiem tra nhiet do, do am co thay doi so voi gia tri cu hay khong, neu k thi
    khong cap nhat len LCD Nokia
    if((t2 != t2Old) or (h2 != h2Old)):
        t2Old = t2
        h2Old = h2
        # Xoa gia tri cu de ghi vao gia tri moi
        draw.rectangle((58,18,72,38), outline=255, fill=255)
        disp.image(image)
        disp.display()
    print ("t2 = {0:0.1f}  H2 = {1:0.1f}\n").format(t2, h2);
    draw.text((42,18), 't2:%0.2s' %(t2), font=font)
    draw.text((42,28), 'H2:%0.2s' %(h2), font=font)
    disp.image(image)
    disp.display()
    return

# ===== Chuong trinh chinh =====

```

```
while True:
    processDHT11()
    time.sleep(2)
    processRX()
    time.sleep(0.5)
    processTX(1)
    time.sleep(0.5)
    strRX = ""
    t1 = 0
    h1 = 0
    t2 = 0
    h2 = 0
```

### Code của mục 3.2

```
#===== Import các thư viện cần sử dụng =====
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.UART as UART
import Adafruit_Nokia_LCD as LCD
import Adafruit_GPIO.SPI as SPI
import Adafruit_DHT
import serial
import time
import datetime
import threading
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

#===== Khởi tạo hệ thống và định nghĩa các hàm =====
# Khai báo Hardware SPI:
DC = "P9_15"
RST = "P9_12"
SPI_PORT = 1
SPI_DEVICE = 0
Sensor_Pin = "P8_15" #T/H Pin: Khai báo chân của cảm biến DHT11
Sensor_Type = Adafruit_DHT.DHT11 #Khai báo loại cảm biến
Sensor_temp = 0 #Khai báo biến nhiệt độ, độ ẩm
Sensor_humi = 0
Hour = 0 #Khai báo biến thời gian
Minute = 0
Second = 0

#===== Khởi tạo các biến dùng chung cho toàn bộ chương trình =====
Node_Sum = int(input("Nhập tổng số node:"))
Node_ID = int(input("Nhập node đang sử dụng:"))
```

```

Node_time_distance= int(input("Nhap do lon khe thoi gian:"))
PAN_ID = int(input("Nhap PAN ID dang su dung:"))

Node_Display = Node_ID          # Bien dung de hien thi thong tin node
hien tai

Timer_times = 0                 # So lan vao timer ???
Firt_Display = 0                # Lan dau hien thi GLCD
Data_Array = [0 for row in range(3*(Node_Sum + 1))]
# Khoi tao mang dong ( Node_ID, Sensor_temp, Sensor_humi)
# Khai bao frame truyen cua UART
Uart_TX = [0 for row in range (4)]
# Uart_TX = [Node_ID, Sensor_temp, Sensor_humi, Checksum]
Str_TX_Number = 4               # So luong tham so truyen di
TX_times = 0                    # Lan dau phat du lieu di
strRX = ""                      # Khai bao mang du lieu rong

# ===== Khoi tao SPI dieu khien LCD =====
disp = LCD.PCD8544(DC, RST, spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE,
max_speed_hz=4000000))
disp.begin(contrast=40)
disp.clear()
disp.display()
image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))
draw = ImageDraw.Draw(image)
draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT), outline=255,
fill=255)
font = ImageFont.load_default()
draw.text((0,0), 'NHIET DO-DO AM', font=font)
disp.image(image)
disp.display()

# ===== Khoi tao UART1 =====

```

```

ser1 = serial.Serial(port = "/dev/ttyS1", baudrate=9600)
ser1.close()
ser1.open()
if ser1.isOpen():
    print ("Cong UART 1 duoc mo!")
# ----- Khoi tao UART5 -----
ser5 = serial.Serial(port = "/dev/ttyS5", baudrate=9600)
ser5.close()
ser5.open()
if ser5.isOpen():
    print ("Cong UART 5 duoc mo!")

# ===== Ham xu ly truyen UART1 va UART5 =====

def processTX1():
    global Uart_TX, Node_ID
    Uart_TX[0] = int (Node_ID)
    Uart_TX[1] = int(Data_Array[3*Node_ID-2])
    Uart_TX[2] = int(Data_Array[3*Node_ID-1])
    Uart_TX[3] = Uart_TX[0] + Uart_TX[1] + Uart_TX[2]
    strToSend = str(Uart_TX[0]) + ":" + str(Uart_TX[1]) + ":" +
str(Uart_TX[2]) + ":" + str(Uart_TX[3]) + "\n"
    ser1.write(strToSend)
    print Uart_TX
    return
#-----
def processTX5():
    global Uart_TX, Node_ID, PAN_ID
    Uart_TX[0] = int (PAN_ID)
    Uart_TX[1] = int(Data_Array[3*Node_ID-2])
    Uart_TX[2] = int(Data_Array[3*Node_ID-1])
    Uart_TX[3] = Uart_TX[0] + Uart_TX[1] + Uart_TX[2]

```

```

        strToSend = str(Uart_TX[0]) + ":" + str(Uart_TX[1]) + ":" +
str(Uart_TX[2]) + ":" + str(Uart_TX[3]) + "\n"
        ser5.write(strToSend)
        print Uart_TX
        return

# ===== Ham xu ly nhan =====
def processRX1():
    global Data_Array, strRX, Node_Sum
    checksum = 0
    Node_ID_Rx = 0
    if(ser1.inWaiting()):
        strRX = ser1.readline()      # Doc chuoai nhan ve
        strRX = strRX.replace("\n", "")
        # Chuyen ky tu xuong dong "\n" thanh ":" truoc khi tach
        strRxData = strRX.split(':')      # Tach gia tri luu tru thoi gian
        strRxData = filter(None, strRxData)    # Xoa cac ki tu Null
        # Chuyen doi tu mang ky tu sang mang so nguyen
        # Loi truyen sai ky tu khong phai so nguyen
        try:
            arrRxData = [int(strNumber) for strNumber in strRxData]
            print arrRxData
            #Kiem tra xem so phan tu nhan ve co bang so truyen di hay khong
            lenRX = len(arrRxData)
            if(lenRX == Str_TX_Number):
                for i in range(0, lenRX-1):
                    checksum = checksum + arrRxData[i]
                # Kiem tra checksum co bang gia tri cuoi cung khong
                if(checksum == arrRxData[Str_TX_Number - 1]):
                    #Kiem tra Node_ID, nap du lieu vao vi tri tuong ung trong mang
                    if (arrRxData[0] <= Node_Sum):
                        Node_ID_Rx = arrRxData[0]

```

```

        print Node_ID_Rx
        Data_Array[3*(Node_ID_Rx-1)+1] = arrRxData[1]
        Data_Array[3*(Node_ID_Rx-1)+2] = arrRxData[2]
    print Data_Array
else:
    print("Ma kiem tra loi khong khop")
else:
    print("Qua trinh truyen nhan bi loi")
except ValueError:
    print("Du lieu nhan duoc xay ra dot bien")
    print strRxData
return

#----- Nhan du lieu tu mang PAN khac -----
def processRX5():
    global Data_Array, strRX, Node_Sum
    checksum = 0
    Node_ID_Rx = 0
    if (ser5.inWaiting()):
        strRX = ser5.readline()      # Doc chuoi nhan ve
        strRX = strRX.replace("\n", "")
        # Chuyen ky tu xuong dong "\n" thanh ":" truoc khi tach
        strRxData = strRX.split(':')      # Tach gia tri luu tru thoi gian
        strRxData = filter(None, strRxData)      # Xoa cac ki tu Null
        # Chuyen doi tu mang ky tu sang mang so nguyen
        # Loi truyen sai ky tu khong phai so nguyen
    try:
        arrRxData = [int(strNumber) for strNumber in strRxData]
        print arrRxData
        #Kiem tra xem so phan tu nhan ve co bang so truyen di hay khong
        lenRX = len(arrRxData)
        if (lenRX == Str_TX_Number):
            for i in range(0, lenRX-1):

```

```

        checksum = checksum + arrRxData[i]
    # Kiem tra checksum co bang gia tri cuoi cung khong
    if(checksum == arrRxData[Str_TX_Number - 1]):
    #Kiem tra Node_ID, nap du lieu vao vi tri tuong ung trong mang
        if (arrRxData[0] <= Node_Sum):
            Node_ID_Rx = arrRxData[0]
            print Node_ID_Rx
            Data_Array[3*(Node_ID_Rx-1)+1] = arrRxData[1]
            Data_Array[3*(Node_ID_Rx-1)+2] = arrRxData[2]
        if (arrRxData[0] > Node_Sum):
            Data_Array[3*Node_Sum] = arrRxData[0]

            Data_Array[3*Node_Sum+1] = arrRxData[1]
            Data_Array[3*Node_Sum+2] = arrRxData[2]
        print Data_Array
    else:
        print("Ma kiem tra loi khong khop")
    else:
        print("Qua trinh truyen nhan bi loi")
except ValueError:
    print("Du lieu nhan duoc xay ra dot bien")
    print strRxData
return

```

#=====Doc thoi gian hien tai=====

```

def processTime():
    global Second, Minute, Hour
    curTime=datetime.datetime.now()          # Lay gio cua he thong
    Hour = curTime.hour
    Minute = curTime.minute
    Second = curTime.second
    print("Thoi gian hien tai: %s:%s:%s" % (Hour, Minute,Second))

```



```
#===== Lay thoi gian chenh lech so voi moc Moc 0 (tinh bang s) =====
def Timeline():
```

```
    curTime=datetime.datetime.now()
    desTime=curTime.replace(hour=0,minute=0,second=0)
    # lay moc time: 00: 00:00 s
    delta=curTime-desTime
    delta_time = (24*60*60+delta.total_seconds())%(24*60*60)
    print "Thoi gian phai cho %d giay" % delta_time
    return delta_time          # tra lai khoang cach time so voi moc cai dat
```

```
#===== Xet khe thoi gian bat dau de phat du lieu lan dau =====
def Trans_data_start():
```

```
    global Node_ID, TX_times
    Time_distance = Timeline() # Doc thoi gian chenh lech voi Moc cai dat
    Time_node = Time_distance % (Node_Sum*Node_time_distance) # Thoi
gian so voi moc phat du lieu
    print("Khoan cach thoi gian: %d seconds" % (Time_distance))
    print("Cach diem moc thoi gian: %d seconds" % (Time_node))
    if((Time_node >= (Node_time_distance*(Node_ID-1))) and (Time_node
<= (Node_time_distance*(Node_ID-0.6)))):
        print("Cach diem moc thoi gian: %d seconds" % (Time_node))
        print("Phat du lieu_Node: %s" % (Node_ID))
        print("Trans_data: is first time")
        processTX1()
        # Thuoc moc thoi gian cua Node, bat dau phat du lieu
        TX_times = 1
        # Danh dau da tim duoc khe phat du lieu cua Node hien hanh
        Trans_data_timer() # Kich hoat Timer de dinh thoi gian phat
    return
```

```
#===== Phat du lieu dua tren moc phat lan 1 =====
```

```

def Trans_data_timer():
    print('Trans_data continuous')
    global TX_times, Node_Sum, Node_time_distance
    processTX1()
    if (Node_ID == 1):
        processTX5()
        # Neu la Node dau ra cua mang PAN thi phat du lieu cho mang PAN khac
    TX_times = 2
    timer_trans =
threading.Timer(Node_Sum*Node_time_distance,Trans_data_timer)
    timer_trans.start()    # Khoi dong dinh thoi phat du lieu cua Node hien hanh
    print TX_times
    return

#==== Khoi tao mang dong va dinh gio reset lai noi dung:600s mot lan =====
def Reset_data_array_timer():
    print('Data reset')
    global Data_Array, Node_Sum
    for i in range(1, Node_Sum + 2):
        Data_Array[3*i-1] = 0
        Data_Array[3*i-2] = 0
        Data_Array[3*i-3] = i
    print Data_Array
    timer_Data_array = threading.Timer(600,Reset_data_array_timer)
    timer_Data_array.start()
    return

# ===== Ham xu ly doc nhiet do =====
def processDHT11():
    global Sensor_temp, Sensor_humi
    # Su dung bien Global
    #Doc nhiet do, do am tu cam bien DTH11

```

```

    Sensor_humi, Sensor_temp = Adafruit_DHT.read_retry(Sensor_Type,
Sensor_Pin)
    #Cap nhat thong tin doc duoc vao mang du lieu
    Data_Array[3*Node_ID-2] = Sensor_temp
    Data_Array[3*Node_ID-1] = Sensor_humi
    print Data_Array
    # Kiem tra gia tri tra ve tu cam bien (do _am va nhiet_do) khac NULL
    if (Sensor_humi is not None) and (Sensor_temp is not None):
        print ("Sensor_temp = {0:0.1f}  Sensor_humi =
{ 1:0.1f}\n").format(Sensor_temp, Sensor_humi);
    else:
        print("Loi khong the doc tu cam bien DHT11 :(\n")
    return

#==== Cap nhat nhiet do, do am Node hien hanh:60s mot lan =====
def DHT11_timer():
    print('Cap nhat DHT11')
    processDHT11()
    timer_DHT11 = threading.Timer(60,DHT11_timer)
    timer_DHT11.start()
    return

# ===== Ham xu ly hien thi Nokia_GLCD =====

def Nokia_Display(i):
    global Data_Array, Firt_Display, Node_Sum, PAN_ID
    draw.rectangle((1,10,78,47), outline=255, fill=255)
    disp.image(image)
    disp.display()          # Xoa gia tri cu de ghi vao gia tri moi
    print("Vao hien thi: %s" % (Data_Array[3*i-3]))
    print ("NHIET DO = {0:0.1f}  DO AM =
{ 1:0.1f}\n").format(Data_Array[3*i-2], Data_Array[3*i-1]);

```

```

# Trang thai hien thi khi chuyen hien thi giua cac Node
if(Firt_Display == 1) and ((i < Node_Sum + 1) or (PAN_ID != 0)):
    if(i == Node_Sum + 1):
        draw.text((12,18), "CHUYEN PAN", font=font)
    else:
        draw.text((12,18), "CHUYEN NODE", font=font)
Firt_Display = 1
disp.image(image)
disp.display()
time.sleep(2.0)
# Xoa gia tri cu de ghi vao gia tri moi
draw.rectangle((1,10,78,47), outline=255, fill=255)
disp.image(image)
disp.display()
if(i <= Node_Sum):
    draw.text((12,10), 'TAI NODE:%0.2s' %(Data_Array[3*i-3]),
font=font)
    if(Data_Array[3*i-2] == 0):
        draw.text((6,23), 'OFF OR ERROR', font=font)
    else:
        draw.text((10,23), 'NHIET DO:%0.2s' %(Data_Array[3*i-2]),
font=font)
        draw.text((22,36), 'DO AM:%0.2s' %(Data_Array[3*i-1]),
font=font)
    disp.image(image)
    disp.display()
if(i > Node_Sum) and (PAN_ID != 0):
    if(Data_Array[3*i-3] == i):
        draw.text((10,18), "NO CONNECT", font=font)
    else:
        draw.text((12,10), 'TAI PAN:%0.2s' %(Data_Array[3*i-3]),
font=font)

```

```

        if(Data_Array[3*i-2] == 0):
            draw.text((6,23), 'OFF OR ERROR', font=font)
        else:
            draw.text((10,23), 'NHIET DO:%0.2s' %(Data_Array[3*i-
2]), font=font)
            draw.text((22,36), 'DO AM:%0.2s' %(Data_Array[3*i-1]),
font=font)
        disp.image(image)
        disp.display()

    if(i > Node_Sum) and (PAN_ID == 0):
        draw.text((12,18), "NOT PAN OUT", font=font)
        disp.image(image)
        disp.display()
    return

#===== Ham cai dat thoi gian thay doi hien thi Nokia_Timer =====
def Nokia_timer():
    print('Hello timer LCD')
    global Node_Display
    Node_Display=Node_Display +1
    if (Node_Display > Node_Sum + 1):
        Node_Display = 1
    Nokia_Display(Node_Display)
    timer = threading.Timer(10,Nokia_timer)    #Cai dat Timer
    timer.start()
    return

#===== Chuong trinh chinh =====
if __name__ == '__main__':
    Reset_data_array_timer()                #Tao mang dong
    processTime()                            #Doc thoi gian hien tai

```

```

DHT11_timer()                #Kich hoat cap nhat nhiet do, do am
Nokia_Display(Node_ID)       #Hien thi nhiet do, do am Node hien hanh
time.sleep(10)
Nokia_timer()
#Kich hoat time tu dong hien thi lan luot thong tin tung Node

#----- Vong lap while cua chuong trinh chinh -----
while True:
    if (TX_times == 0):
        Trans_data_start()
    if (TX_times == 1):
        print("Qua trinh phat da duoc kich hoat bang Timer")
        TX_times = 2
    time.sleep(10)
    processRX1()
    processRX5()
    strRX = ""

```