

KHOA VÔ TUYẾN ĐIỆN TỬ  
BỘ MÔN KỸ THUẬT XUNG SỐ - VI XỬ LÝ

**HƯỚNG DẪN THÍ NGHIỆM  
TRÊN MẠCH THÍ NGHIỆM KỸ THUẬT VI XỬ LÝ  
LẬP TRÌNH NHÚNG THỜI GIAN THỰC**

**Hà Nội, tháng 11 năm 2017**

KHOA VÔ TUYẾN ĐIỆN TỬ  
BỘ MÔN KỸ THUẬT XUNG SỐ - VI XỬ LÝ

HƯỚNG DẪN THÍ NGHIỆM  
TRÊN MẠCH THÍ NGHIỆM KỸ THUẬT VI XỬ LÝ  
LẬP TRÌNH NHÚNG THỜI GIAN THỰC

CHỦ NHIỆM BỘ MÔN

NGƯỜI VIẾT

Đại tá Nguyễn Hải Dương

Thượng úy Nguyễn Khoa Sang

HỘI ĐỒNG KHOA HỌC KHOA  
CHỦ TỊCH

Đại tá Đỗ Quốc Trinh

## MỤC LỤC

CHƯƠNG 1 – TỔNG QUAN.....	5
1.1. Tổng quan chung.....	5
1.2. Các module.....	5
CHƯƠNG 2 – CÀI ĐẶT VÀ CẤU HÌNH HỆ ĐIỀU HÀNH .....	9
2.1. Giới thiệu.....	9
2.2. Cài đặt hệ điều hành trên thẻ nhớ micro SD .....	9
2.3. Cài đặt hệ điều hành trên thẻ nhớ eMMC .....	11
CHƯƠNG 3 – PHƯƠNG THỨC KẾT NỐI .....	13
3.1. Kết nối với bàn phím, chuột, USB Stick qua cổng USB .....	13
3.2. Kết nối Ethernet .....	13
3.3. Kết nối HDMI với màn hình.....	14
3.4. Kết nối từ xa thông qua SSH.....	15
3.5. Kết nối truyền nhận file qua SFTP.....	16
CHƯƠNG 4 – NGÔN NGỮ LẬP TRÌNH PYTHON.....	19
4.1. Mục tiêu.....	19
4.2. Công cụ và phần mềm.....	19
4.3. Nội dung.....	19
4.3.1. Giới thiệu về Python .....	19
4.3.2. Sử dụng Python cơ bản .....	19
4.3.4. Cài đặt Python .....	22
CHƯƠNG 5 – ĐIỀU KHIỂN CÁC NGOẠI VI CƠ BẢN TRONG HỆ THỐNG NHÚNG THỜI GIAN THỰC.....	24
5.1. Điều khiển LED đơn .....	24
5.1.1. Mục tiêu.....	24
5.1.2. Công cụ và phần mềm.....	24
5.1.3. Nội dung .....	24
5.1.4. Thực hành.....	28
5.2. Đọc giá trị từ nút nhấn.....	29
5.2.1. Mục tiêu.....	29
5.2.2. Công cụ và phần mềm.....	29

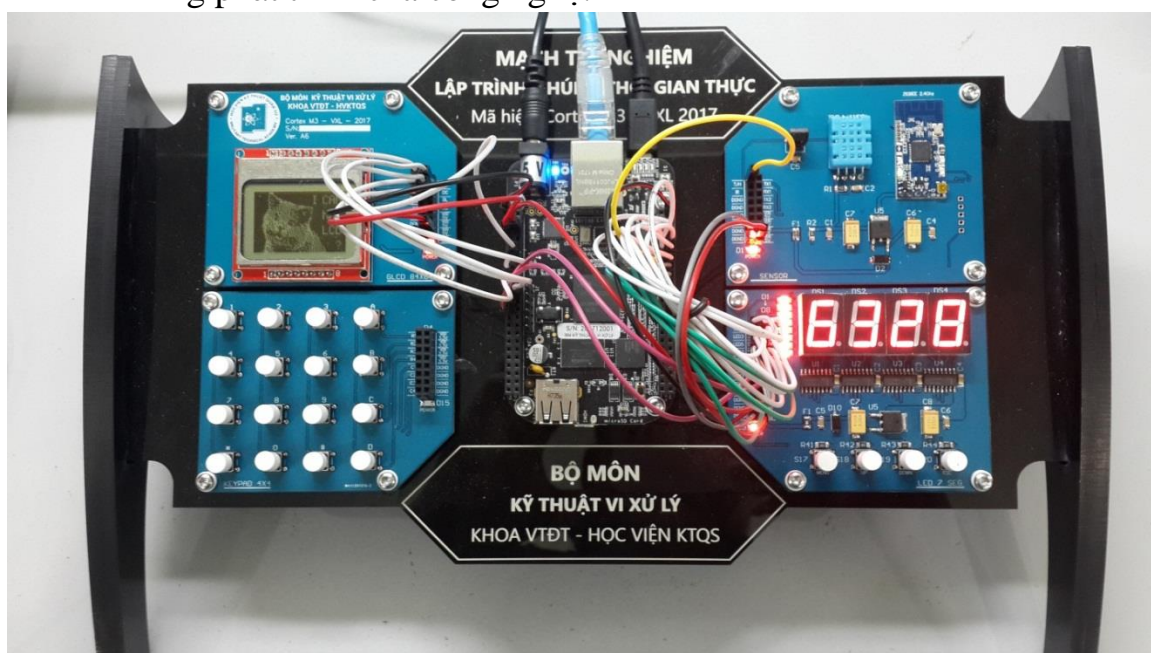
5.2.3. Nội dung.....	29
5.2.4. Thực hành.....	31
5.3. Điều khiển LED 7 thanh .....	32
5.3.1. Mục tiêu.....	32
5.3.2. Công cụ và phần mềm.....	32
5.3.3. Nội dung.....	32
5.3.4. Thực hành.....	37
5.4. Điều khiển màn hình GLCD .....	38
5.4.1. Mục tiêu.....	38
5.4.2. Công cụ và phần mềm.....	38
5.4.3. Nội dung.....	38
5.5.4. Thực hành.....	40
5.5. Giao tiếp với bàn phím hexa .....	41
5.5.1. Mục tiêu.....	41
5.5.2. Công cụ và phần mềm.....	41
5.5.3. Nội dung.....	41
5.5.4. Thực hành.....	45
CHƯƠNG 6 – ĐIỀU KHIỂN MÔ – ĐUN CẢM BIẾN TRONG HỆ THỐNG NHÚNG THỜI GIAN THỰC.....	46
6.1. Giao tiếp với cảm biến nhiệt độ và độ ẩm .....	46
6.1.1. Mục tiêu.....	46
6.1.2. Công cụ và phần mềm.....	46
6.1.3. Nội dung.....	46
6.1.4. Thực hành.....	49
6.2. Truyền thông không dây sử dụng mô đun Zibee CC2530 TI.....	50
6.2.1. Mục tiêu.....	50
6.2.2. Công cụ và phần mềm.....	50
6.2.3. Nội dung.....	50
6.2.4. Thực hành.....	60

# CHƯƠNG 1 – TỔNG QUAN

## 1.1. Tổng quan chung

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.

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.



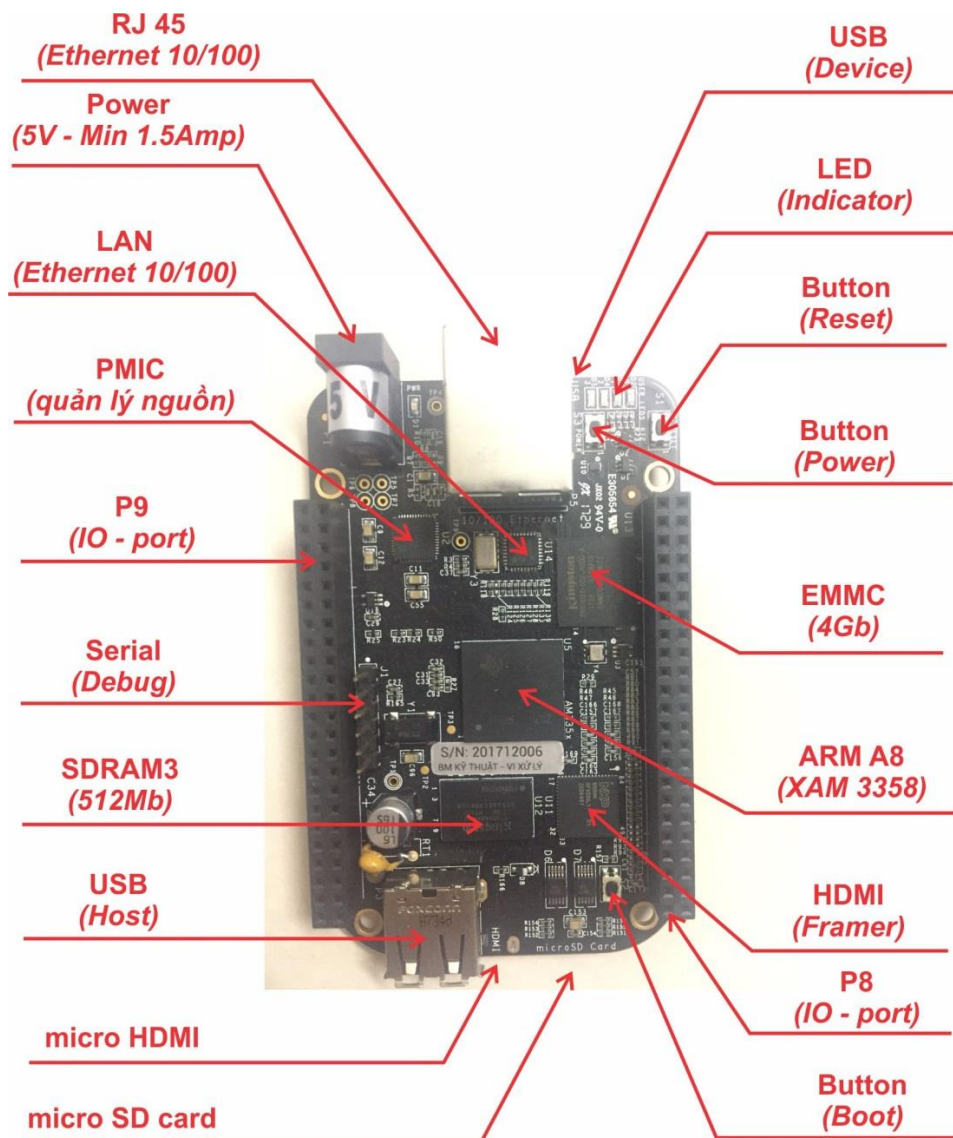
Hình 1.1. Tổng quan của KIT

## 1.2. Các module

KIT bao gồm 5 mô đun, trong đó có một mô đun Vi xử lý và 4 mô đun mở rộng

### ✚ Mô đun Vi xử lý:

- Chip xử lý AM335x 1GHz ARM® Cortex-A8
- 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
- Chạy hệ điều hành Debian, một phiên bản phân phối (distro) của hệ điều hành Linux. (có hỗ trợ hệ điều hành Android)



Hình 1.2. Mô đun Vi xử lý

#### ✚ Mô đun GLCD 84x48:

- Màn hình Graphic 84x48 điểm ảnh
- Led báo nguồn và đầu cắm giao tiếp
- Giao tiếp SPI



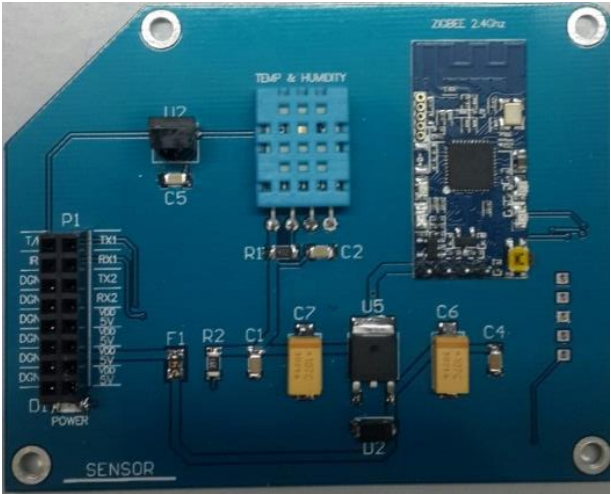
Hình 1.3. Mô đun GLCD 84x84

- Led báo nguồn và đầu cắm giao tiếp



Hình 1.4. Mô đun Keypad 4x4

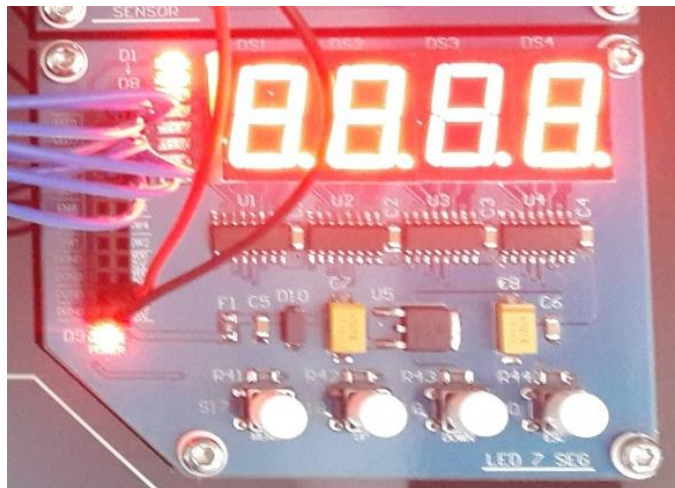
- Cảm biến nhiệt độ, độ ẩm DHT11, giao tiếp 1 dây (1-Wire)



Hình 1.5. Mô đun Sensor

- 4 Led 7 thanh đơn kết hợp với vi mạch ghi dịch 74HC595, giao tiếp SPI

- Led báo nguồn và đầu cắm giao tiếp



*Hình 1.6. Mô đun Led 7 Seg*



## CHƯƠNG 2 – CÀI ĐẶT VÀ CẤU HÌNH HỆ ĐIỀU HÀNH

### 2.1. Giới thiệu

KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017) là một hệ máy tính nhúng (Embedded Computer) hỗ trợ các phiên bản phân phối của hệ điều hành Linux như Debian, Angstrom, Android, Ubuntu và một số phiên bản hệ điều hành khác. Nhưng Debian là hệ điều hành được khuyến cáo sử dụng bởi tính ổn định, nhẹ nhàng và hỗ trợ tốt.

Debian là một phiên bản phân phối của Linux mà tối ưu cho những máy tính hoặc thiết bị nhúng cấu hình thấp, phù hợp với các thiết bị IoT. Debian được sử dụng rộng rãi trên toàn thế giới với cộng đồng sử dụng và hỗ trợ lớn mạnh. Trong KIT này, tất cả những hướng dẫn cài đặt, các phần mềm triển khai code trên KIT đều dựa trên Debian.

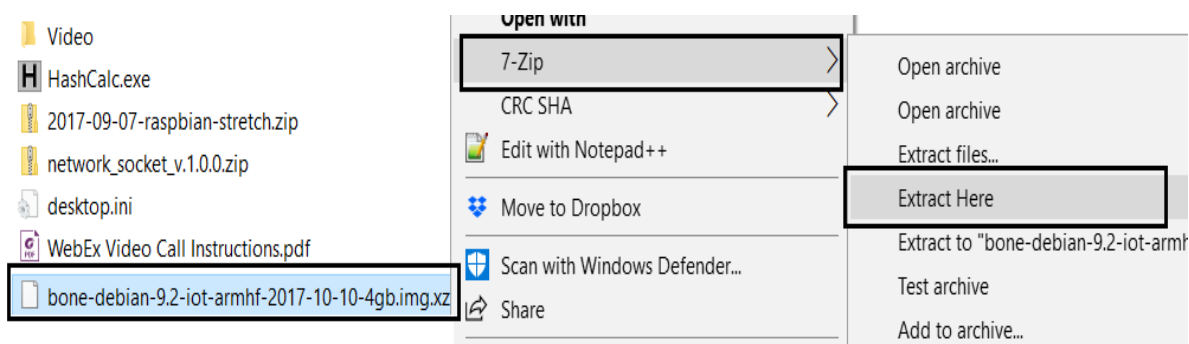
KIT được cài sẵn hệ điều hành Debian, xong cũng như các hệ điều hành khác, sẽ gặp lỗi sau một thời gian triển khai các phần mềm, hoặc đơn giản là chúng ta cần cập nhật hệ điều hành. Bài viết dưới đây sẽ hướng dẫn các bạn cách thức để cài đặt lại (hoặc nâng cấp cứng) hệ điều hành cho KIT.

### 2.2. 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 2.1. 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

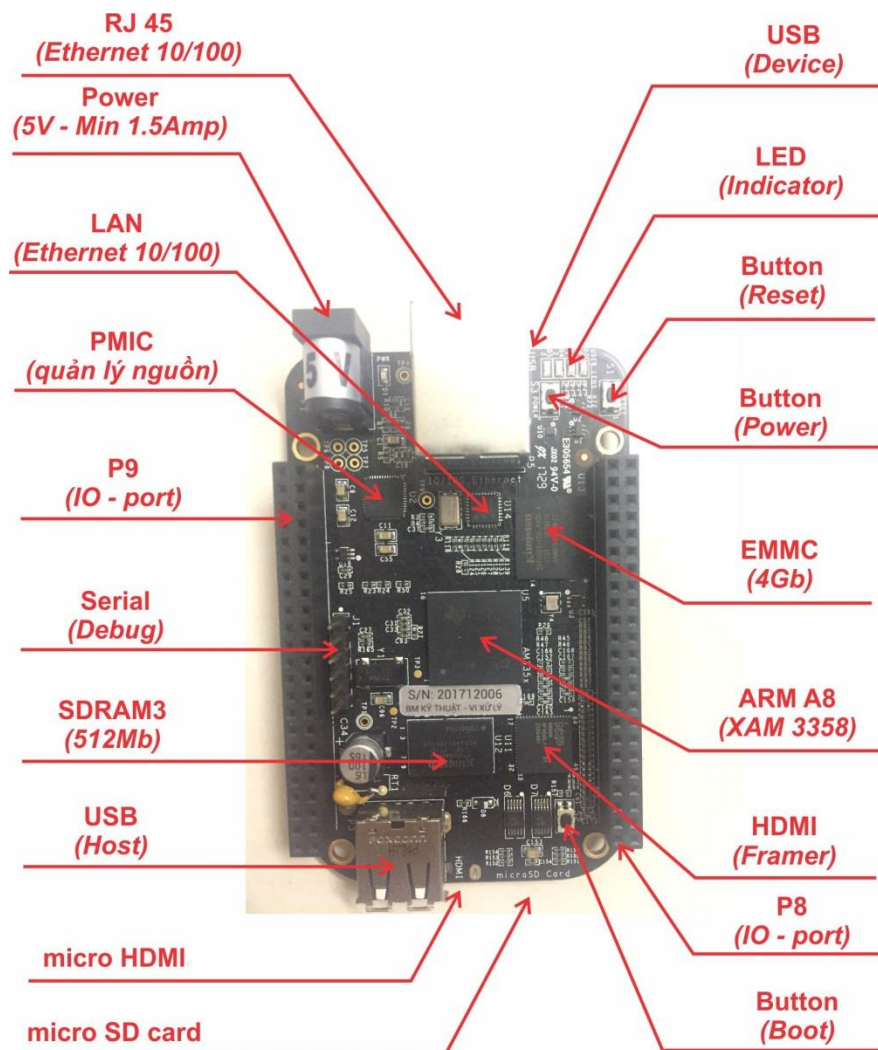


Hình 2.2. Phần mềm Etcher

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

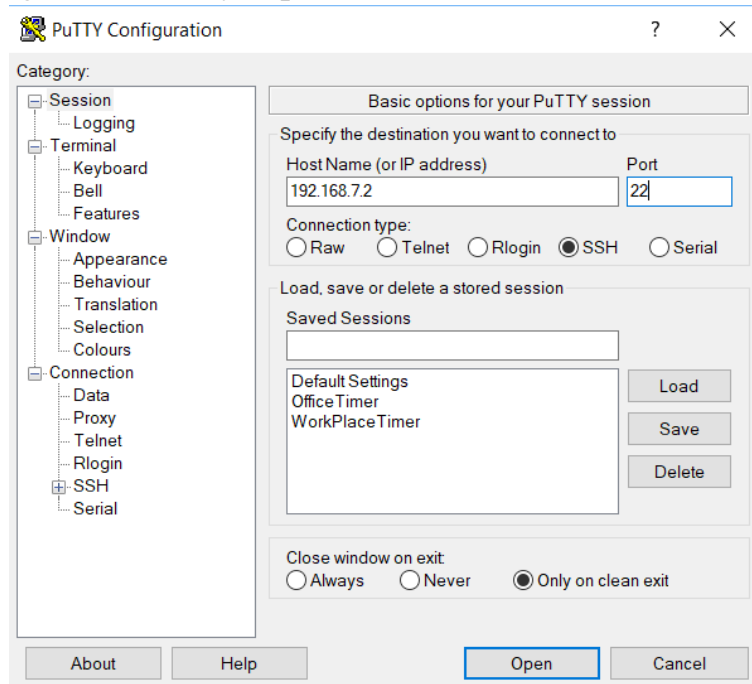


Hình 2.3 Vị trí nút nhấn BOOT và các User Led

## 2.3. Cài đặt hệ điều hành trên thẻ nhớ eMMC

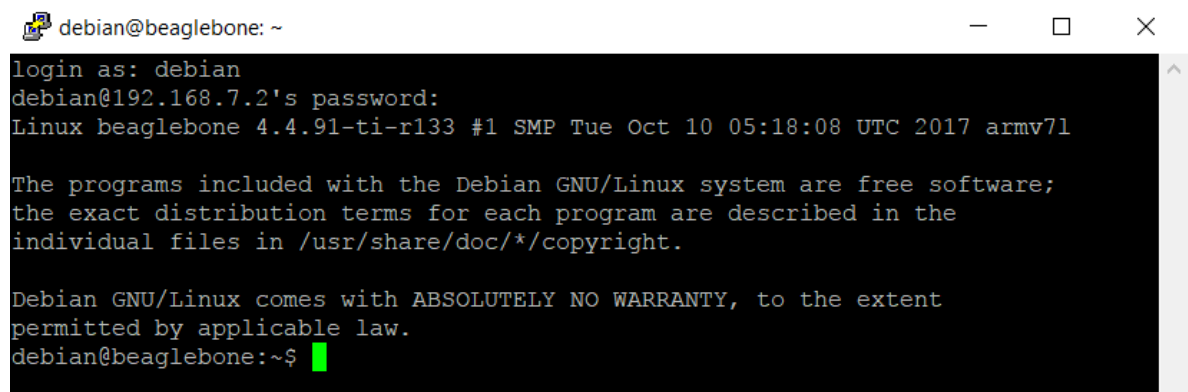
**Bước 1:** Tiến hành cài đặt hệ điều hành trên microSD trước khi cài đặt trên eMMC

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



Hình 2.4 Giao diện truy cập SSH từ PuTTY

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



Hình 2.5. Giao diện kết nối của PuTTY tới KIT

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

*sudo nano /boot/uEnv.txt*

Sửa

*##enable BBB: eMMC Flasher:*

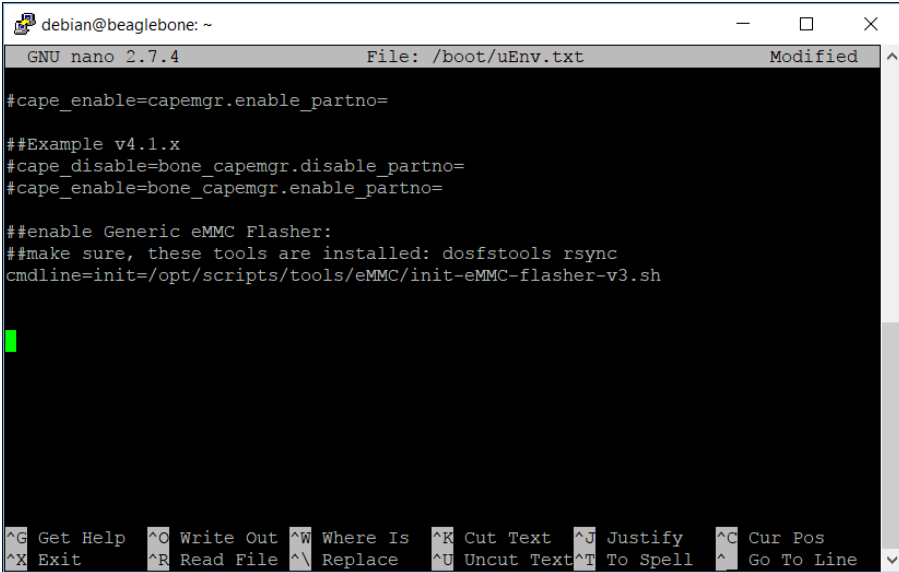
*#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh*

Thành

*##enable BBB: eMMC Flasher:*

*cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh*

Nhấn Ctrl + X và Enter để hoàn tất việc sửa.

A screenshot of a terminal window showing the nano text editor. The window title is 'debian@beaglebone: ~'. The editor is editing the file '/boot/uEnv.txt'. The content of the file includes configuration for the BeagleBone Black, such as enabling the capemgr, setting the cmdline to 'init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh', and enabling the Generic eMMC Flasher. The nano editor's status bar at the bottom shows various keyboard shortcuts like ^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, and ^\_ Go To Line.

```
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 2.6. Giao diện chỉnh sửa file cấu hình boot từ eMMC

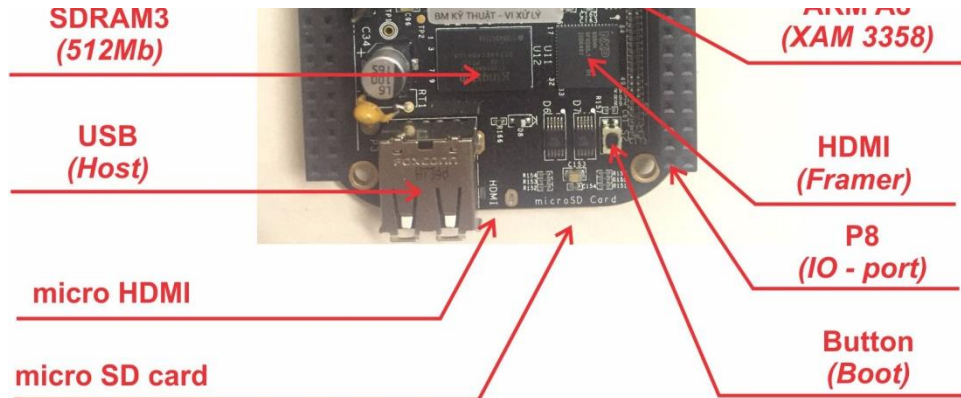
**Bước 4:** Rút nguồn của mô đun và cắm lại, lúc này hệ điều hành sẽ được ghi vào thẻ nhớ eMMC

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

## CHƯƠNG 3 – PHƯƠNG THỨC KẾT NỐI

### 3.1. 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 3.1. 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

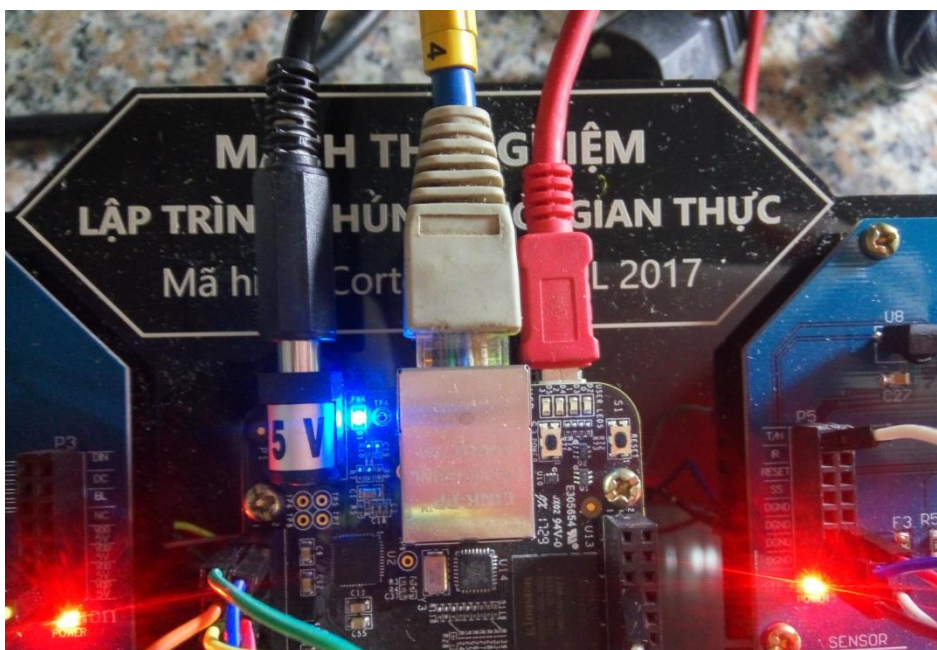


Hình 3.2. Bộ mở rộng cổng USB có hỗ trợ nguồn bên ngoài

### 3.2. Kết nối Enthernet

- Cổng Enthernet 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 Enthernet. Để kết nối, ta cắm dây mạng Cat 5e (hoặc Cat 6) từ KIT tới Switch chia mạng hoặc Router





Hình 3.3. Kết nối vào mạng thông qua cổng Ethernet

### 3.3. 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



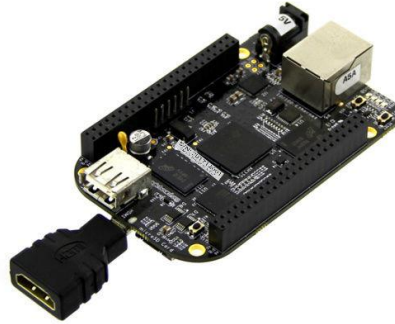
Hình 3.4. Cổng chuyển đổi microHDMI sang HDMI

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



Hình 3.5. Cổng chuyển đổi từ microHDMI sang VGA

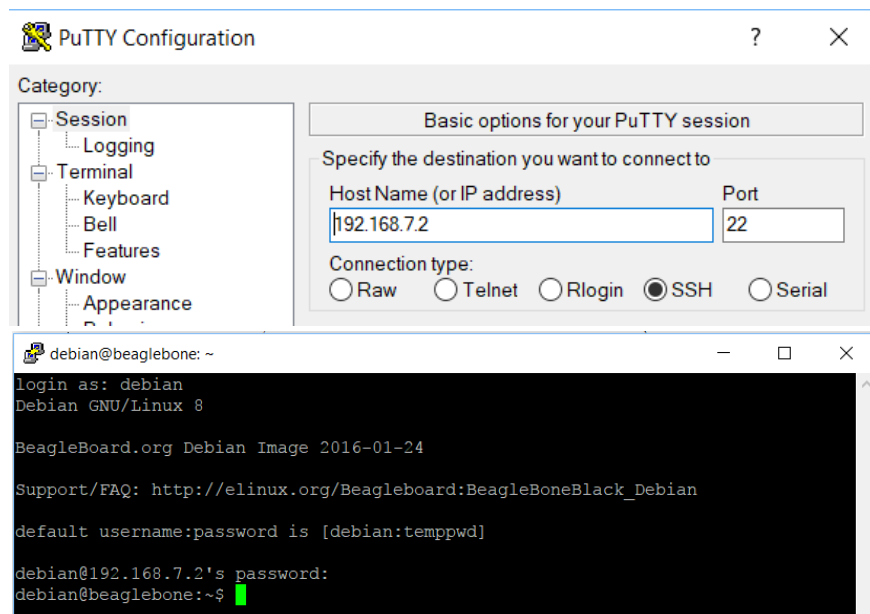
## Vị trí cắm nối



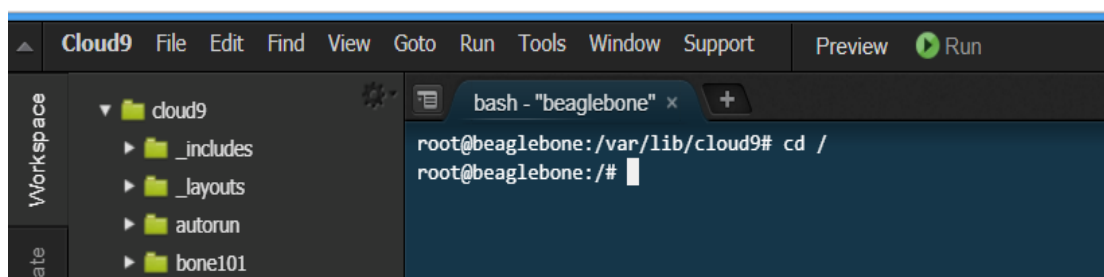
Hình 3.6. Vị trí cắm cổng HDMI từ mô đun vi xử lý cho màn hình

### 3.4. 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

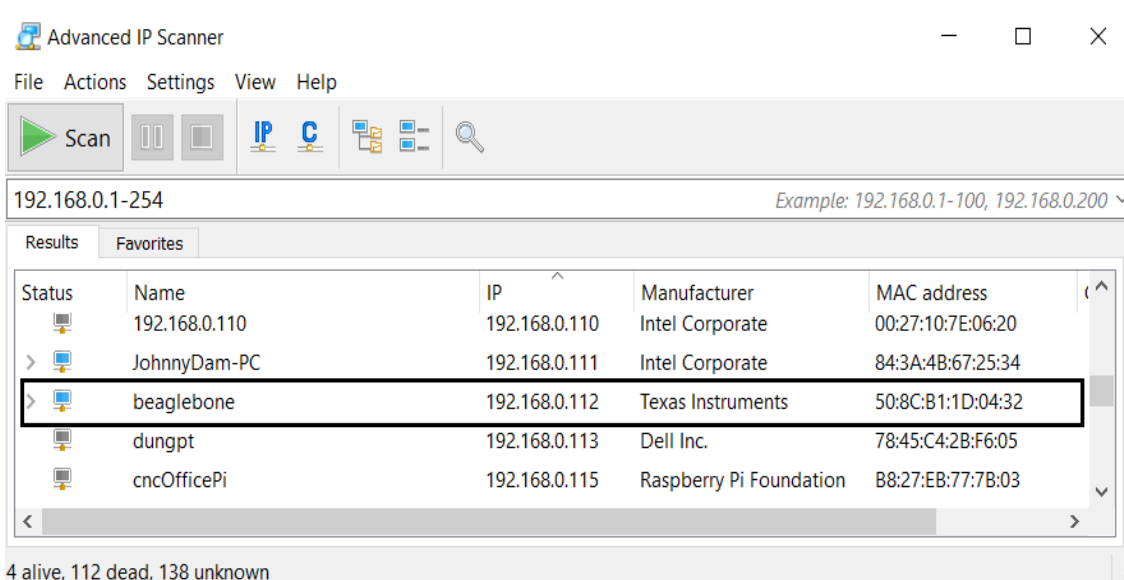


Hình 3.7. Giao diện của phần mềm PuTTY



Hình 3.8. Giao diện của Cloud 9

- 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)



Hình 3.9. Địa chỉ IP của KIT được quét bởi phần mềm Advanced IP Scanner

### 3.5. 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

Trong tài liệu này, 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

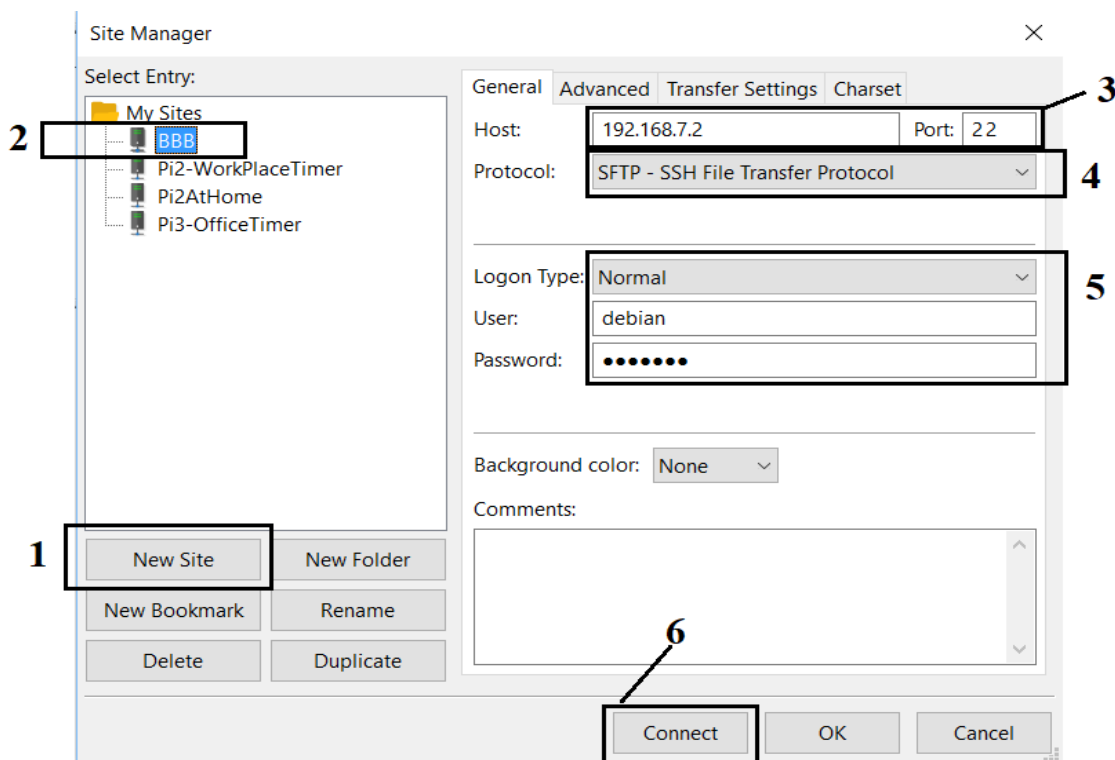
Các bước cấu hình và sử dụng:



**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

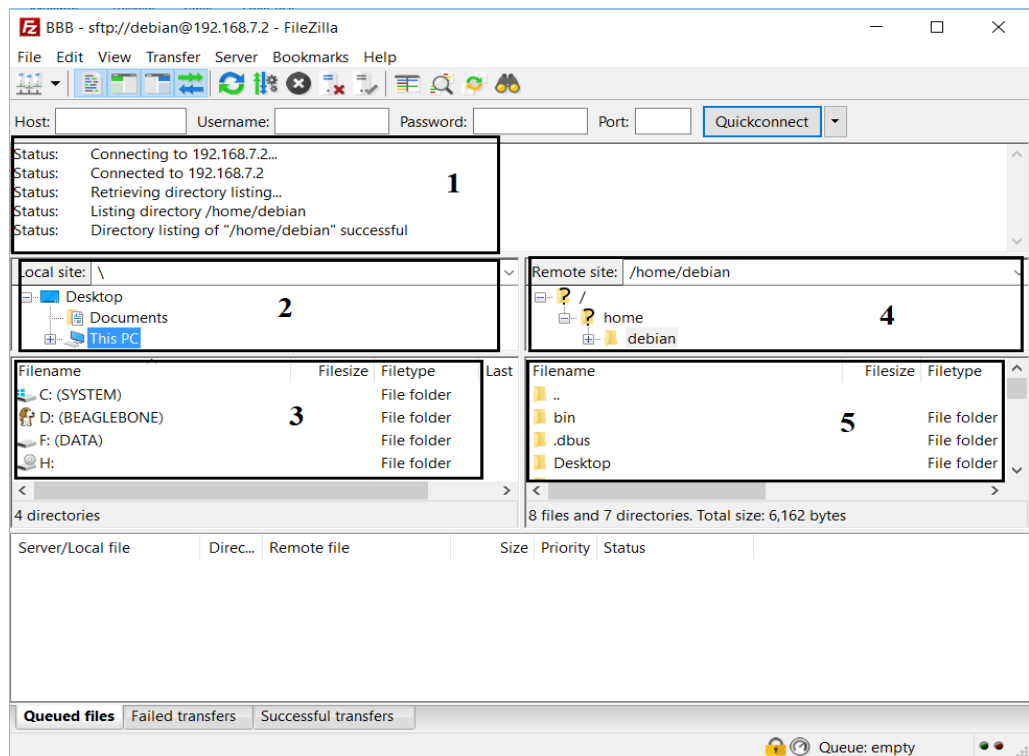


Hình 3.10. Thiết lập các thông số kết nối SFTP từ FileZilla

- ✚ 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

**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)



Hình 3.11. Giao diện kết nối của chương trình FileZilla

#### 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)

## CHƯƠNG 4 – NGÔN NGỮ LẬP TRÌNH PYTHON

### 4.1. Mục tiêu

- Làm quen khái niệm về trình thông dịch Python
- Làm quen với cấu trúc chương trình Python
- Cài đặt, nâng cấp Python cho KIT

### 4.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017) có sẵn kết nối Internet
- Phần mềm PuTTY (cho kết nối từ xa theo giao thức SSH)
- Phần mềm Advanced Scan IP để quét địa chỉ IP của KIT

### 4.3. Nội dung

#### 4.3.1. 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.



Hình 4.1. Logo trình thông dịch Python

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.

#### 4.3.2. Sử dụng Python cơ bản

In ra dòng chữ bất hủ trong giới lập trình "Hello Word" với câu lệnh sau, lưu ý, ngôn ngữ Python không sử dụng dấu (;) để kết thúc câu lệnh

```
>>> print('Hello world')
```

#### ❖ THỤT ĐẦU DÒNG TRONG PYTHON

Một số ngôn ngữ sử dụng dấu ngoặc nhọn { và } để bọc mã nguồn bên trong các dòng code đều thụt đầu dòng để hiển thị lồng trực quan. Tuy nhiên, Python không sử dụng dấu ngoặc nhọn nhưng đòi hỏi phải thụt đầu dòng vào trong. Ví dụ như một vòng lặp for trong Python:

```
>>> for i in range(10):
```

```
print("Hello")
```

khi dùng lệnh thứ 2 thụt vào thì nó sẽ thuộc trong vòng lặp for, như vậy nó sẽ in ra 10 dòng chữ Hello.

Đối với nhiều dòng lệnh hơn thì chúng ta cũng làm tương tự như thế, ví dụ:

```
>>>for i in range(2):  
    print("A")  
    print("B")
```

Kết quả in ra là:

A  
B  
A  
B

Một ví dụ khác

```
>>> for i in range(2):  
    print("A")  
    print("B")
```

Kết quả là

A  
A  
B

## ❖ SỬ DỤNG BIẾN

Để lưu giá trị vào biến bạn làm như sau:

```
name = "Bob"  
age = 15
```

Việc khai báo biến trong Python không cần quan tâm đến kiểu dữ liệu, Python sẽ xử lý tự động để thay đổi kiểu dữ liệu trong quá trình hoạt động.

## ❖ GHI CHÚ MÃ NGUỒN (COMMENT)

Trong Python, chúng ta sử dụng dấu (#) để tạo ghi chú

```
age = 15  
age = age+1 #Tang tuoi len mot don vi  
print(age)
```

## ❖ DANH SÁCH (LIST)

LIST hoặc mảng trong các ngôn ngữ khác là tập hợp các giá trị có kiểu bất kỳ, các giá trị được đặt trong dấu móc vuông ([ và ]):

```
numbers = [1, 2, 3]
```

## ❖ ITERATION

Một số kiểu dữ liệu là iterable, có nghĩa là bạn có thể lặp qua các giá trị mà nó có. Ví dụ: một danh sách:

```
numbers = [1, 2, 3]
for number in numbers:
    print(number)
```

Câu lệnh trên sẽ in ra từng giá trị trong mảng numbers

1  
2  
3

Tôi sử dụng từ number (bạn có thể sử dụng từ nào tùy ý) để chỉ đến từng phần tử trong mảng numbers.

Ví dụ khác, in ra từng ký tự trong 1 chuỗi

```
dog_name = "BINGO"
for char in dog_name:
    print(char)
```

Kết quả nhận được

B  
I  
N  
G  
O

## ❖ RANGE

Kiểu số nguyên thì không thể duyệt qua từng phần tử nếu không chúng sẽ sinh lỗi. Ví dụ:

```
for i in 3:
    print(i)
```

Nó sẽ sinh lỗi

***TypeError: 'int' object is not iterable***

Và chúng ta sẽ sử dụng hàm range() để khắc phục

```
for i in range(3):
    print(i)
```

Hàm range(5) chứa các số 0, 1 và 2 (3 là tổng số)

## ❖ TÌM CHIỀU DÀI GIÁ TRỊ

Ta dùng hàm len():

```
ame = "Jamie"
```

```
print(len(name)) # 5
names = ["Bob", "Jane", "James", "Alice"]
print(len(names)) # 4
```

### ❖ CÂU LỆNH IF

Có cấu trúc như ví dụ sau:

```
name = "Joe"
if len(name) > 3:
    print("Nice name,")
    print(name)
else:
    print("That's a short name,")
    print(name)
```

### ❖ CÁC TẬP TIN PYTHON TRONG IDLE

Ta chọn File > New File để mở file mới, trong đó bạn có thể nhập các câu lệnh Python, lưu lại và chạy nó, kết quả sẽ hiển thị trong 1 cửa sổ khác.

Ví dụ:

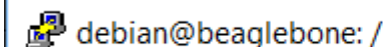
```
n = 0
for i in range(1, 101):
    n += i
print("The sum of the numbers 1 to 100 is:")
print(n)
```

Lưu file File > Save hoặc Ctrl + S, chạy file (Run > Run Module) hoặc nhấn F5 và xem kết quả ở cửa sổ khác

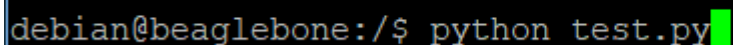
### ❖ THỰC THI FILE PYTHON TỪ DÒNG LỆNH

Ta sử dụng lệnh *cd*, *ls* và *python* để chạy file python có đuôi mở rộng là py.

Ví dụ:



```
debian@beaglebone: /
```



```
debian@beaglebone:/$ python test.py
```

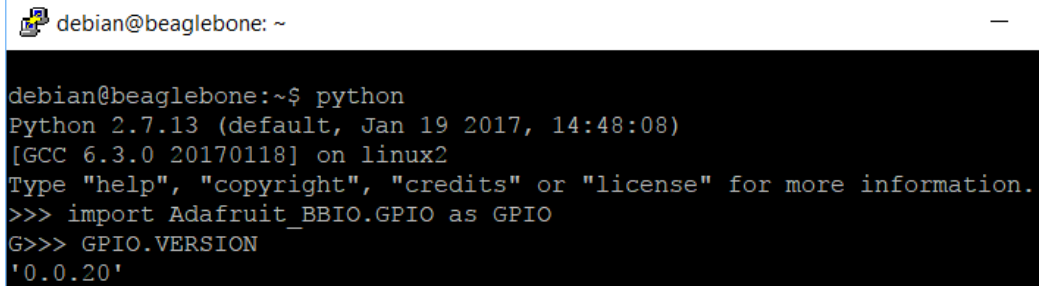
Hình 4.2. Thực thi tập tin python trong LXTerminal

#### 4.3.4. Cài đặt Python

Phiên bản mới nhất của hệ điều hành Debian đã được cài đặt sẵn thư viện giao tiếp phần cứng RPi.GPIO. Chúng ta có thể kiểm tra xem phiên bản RPi.GPIO bằng lệnh trong Terminal như sau:

```
$sudo python
```

```
>>>import Adafruit_BBIO.GPIO as GPIO
>>>GPIO.VERSION
```

A terminal window titled 'debian@beaglebone: ~' with a dark background. It shows the execution of a Python script. The prompt is 'debian@beaglebone:~\$'. The user enters 'python', which starts a Python 2.7.13 shell. The prompt changes to 'Python 2.7.13 (default, Jan 19 2017, 14:48:08)'. The user enters '[GCC 6.3.0 20170118] on linux2'. The user enters 'Type "help", "copyright", "credits" or "license" for more information.'. The user enters '>>> import Adafruit\_BBIO.GPIO as GPIO'. The prompt changes to 'G>>>'. The user enters 'GPIO.VERSION'. The output is ''0.0.20''.

```
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 4.3. 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
sudo apt-get upgrade
```

## CHƯƠNG 5 – ĐIỀU KHIỂN CÁC NGOẠI VI CƠ BẢN TRONG HỆ THỐNG NHÚNG THỜI GIAN THỰC

### 5.1. Điều khiển LED đơn

#### 5.1.1. Mục tiêu

- Làm quen với môi trường phát triển ứng dụng Cloud 9
- Làm quen với ngôn ngữ lập trình Python cho KIT
- Làm quen với quy tắc đặt địa chỉ và chế độ output của GPIO
- Kết nối được mô đun vi xử lý trung tâm tới mô đun Led đơn
- Lập trình tắt, mở, nhấp nháy led đơn

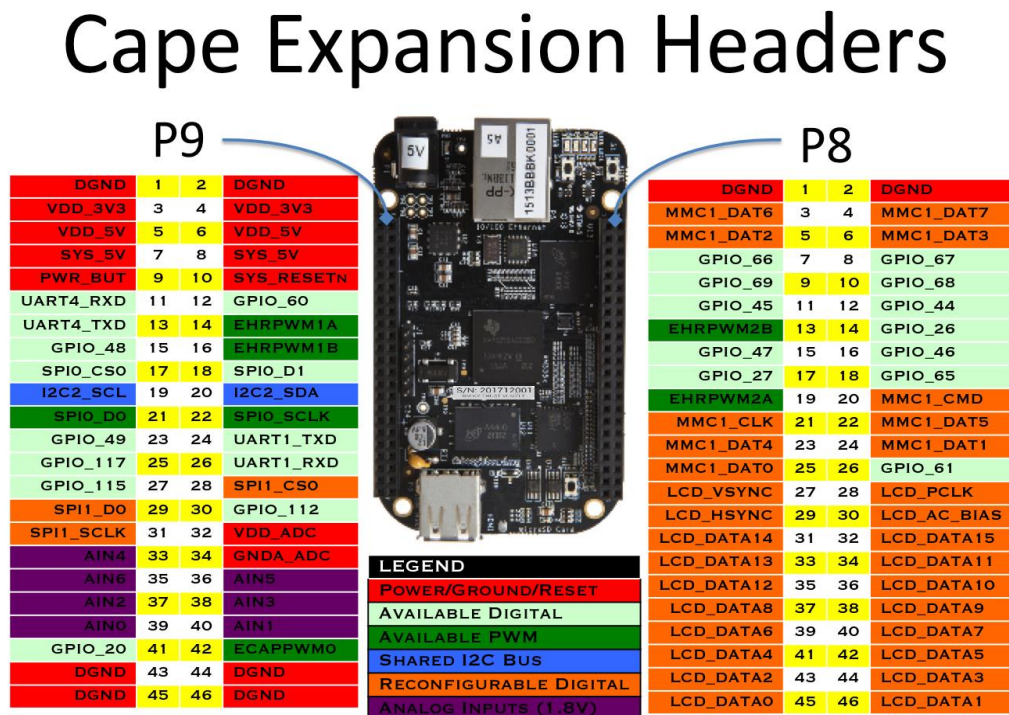
#### 5.1.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9

#### 5.1.3. Nội dung

##### Các cổng vào ra cơ bản (GPIO):

- KIT được thiết kế với 65 GPIO sử dụng như cổng vào hoặc ra số (digital)
- Các cổng GPIO được phân bố đều theo 2 Header là P8 và P9



Hình 5.1. Sơ đồ bố trí Header P8 và P9



# 65 possible digital I/Os

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUTTON	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

Hình 5.2. Sơ đồ bố trí các GPIO

## 🚦 Quy tắc đặt tên GPIO

- Trong lập trình bằng Python cho KIT, khi muốn chỉ định (đánh địa chỉ) một cổng thì ta sử dụng theo cấu trúc “P9\_số thứ tự chân” hoặc “P8\_số thứ tự chân” với Header P9 hoặc P8 tương ứng

- Ví dụ:

Muốn chỉ định GPIO\_50, ta sẽ ghi là “P9\_14”

Muốn chỉ định GPIO\_32, ta sẽ ghi là “P8\_25”

## 🚦 Môi trường phát triển Cloud 9

- Cloud 9 là môi trường phát triển đa ngôn ngữ chạy trên nền Web, tại đây ta có thể soạn thảo, triển khai code xuống KIT
- Cloud 9 hỗ trợ ngôn ngữ nodejs, python, ruby và arduino
- Để khởi chạy vào Cloud9, ta chạy trên trình duyệt với địa chỉ IP của KIT tại cổng 3000

- + Kết nối KIT với máy tính qua cổng USB thì địa chỉ Cloud 9 là:

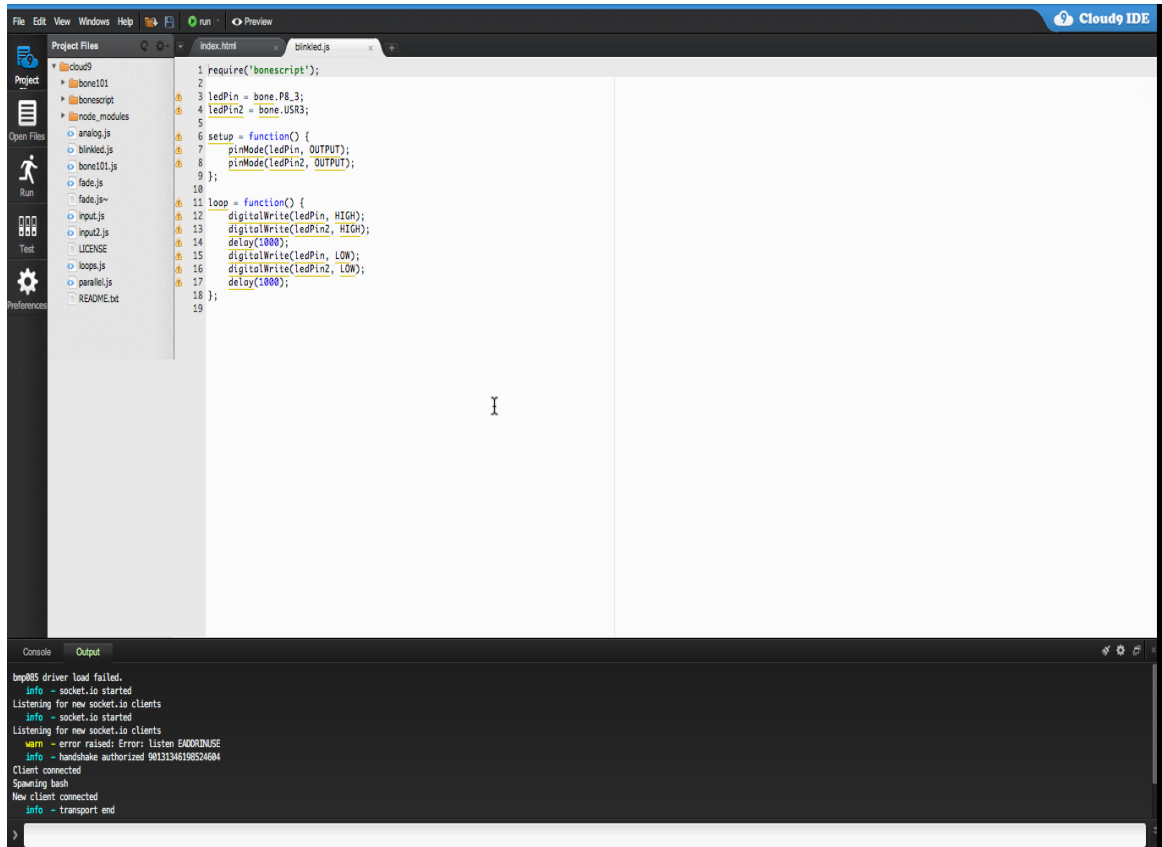
192.168.7.2 :3000

- + Kết nối KIT với cổng Ethernet, địa chỉ Cloud 9 sẽ là IP của KIT với cổng 3000

Ví dụ :

IP : 192.168.1.123

Địa chỉ Cloud 9 : 192.168.1.123 :3000



Hình 5.3. Môi trường phát triển Cloud 9

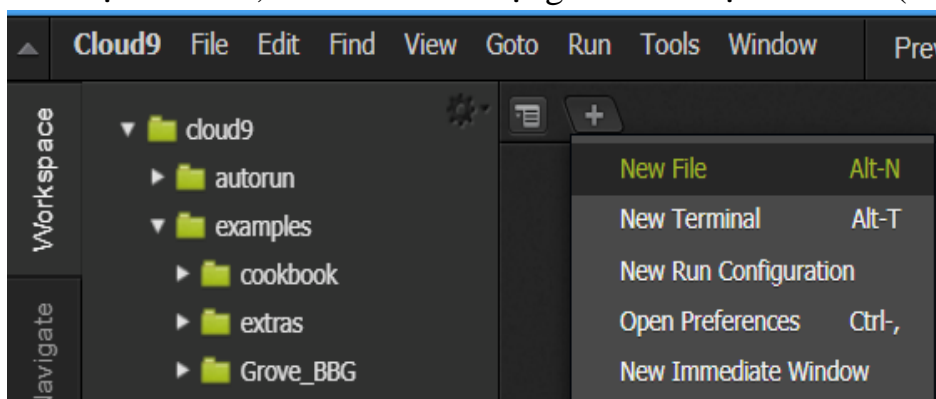
### **Lập trình điều khiển LED đơn**

**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)

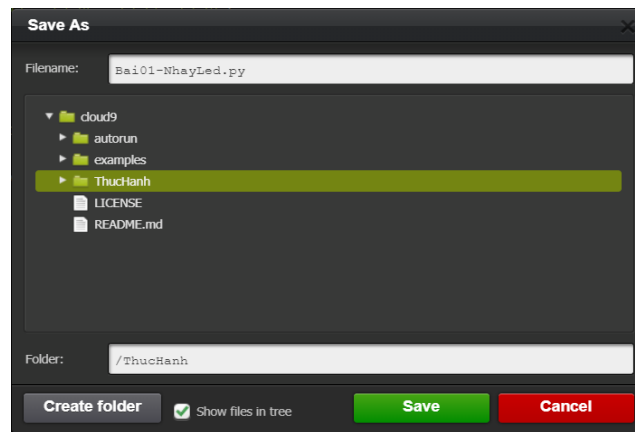
**Bước 3:** Truy cập vào địa chỉ **192.168.7.2 :3000** để mở Cloud 9

**Bước 4:** Tại Cloud 9, tích vào biểu tượng dấu + để tạo file mới (Alt-N)



Hình 5.4. Tạo file mới trong Cloud 9

**Bước 5:** Nhấn **Ctrl-S**, khung hội thoại **Save as** hiện ra, tạo thư mục **ThucHanh** và lưu tên file là: **Bai01-NhayLed.py** vào thư mục vừa tạo



Hình 5.5. Tạo và lưu tên file

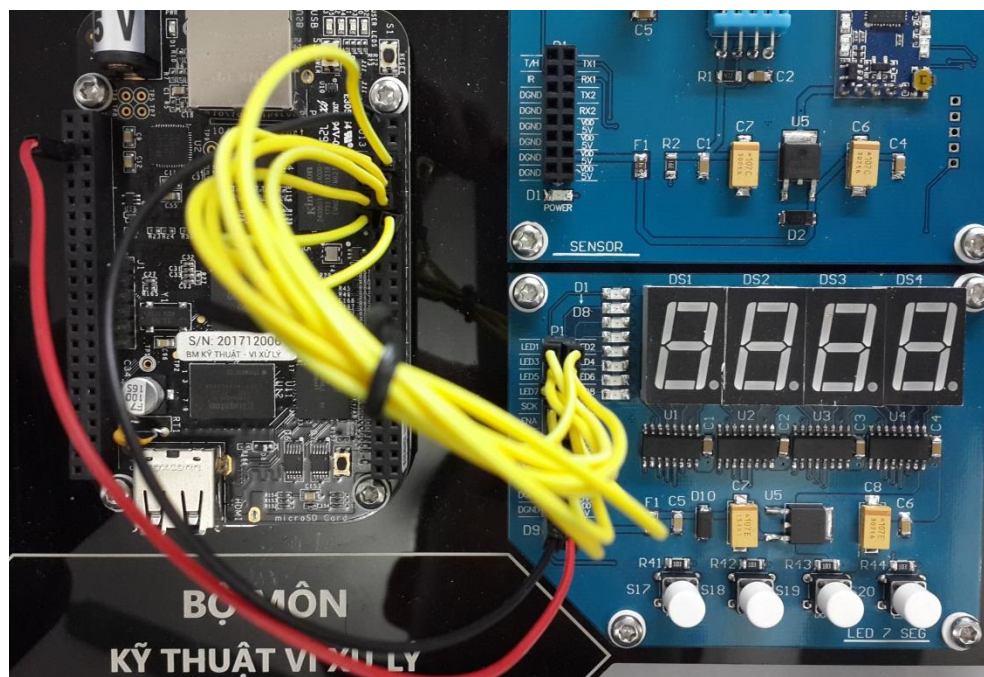
**Bước 6:** Đánh code sau vào khung soạn thảo của Cloud 9

```
Bai01-NhayLed.py x +
1 import Adafruit_BBIO.GPIO as GPIO
2 import time
3
4 leds1 = ["P8_7", "P8_8", "P8_9", "P8_10"];
5 leds2 = ["P8_11", "P8_12", "P8_13", "P8_14"];
6 for i in leds1:
7     GPIO.setup(i, GPIO.OUT)
8 for i in leds2:
9     GPIO.setup(i, GPIO.OUT)
10
11 while True:
12     for i in leds1:
13         GPIO.output(i, GPIO.HIGH)
14         time.sleep(0.5)
15     for i in leds2:
16         GPIO.output(i, GPIO.LOW)
17         time.sleep(0.5)
18     for i in leds1:
19         GPIO.output(i, GPIO.LOW)
20         time.sleep(0.5)
21     for i in leds2:
22         GPIO.output(i, GPIO.HIGH)
23         time.sleep(0.5)_
```

Hình 5.6. Code mẫu điều khiển Led đơn

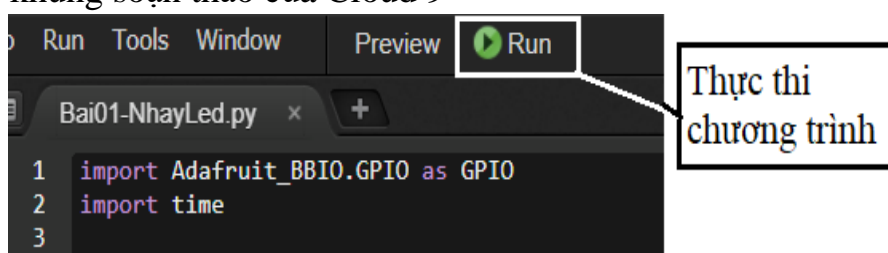
**Bước 7:** Kết nối mô đun Vi xử lý với mô đun Led đơn

- Nối lần lượt các chân từ 7 đến 14 của P8 (Mô đun vi xử lý) sang từ LED1 đến LED8 (Mô đun LED 7 thanh)
- Nối chân 1 (DGND) của P8 (Mô đun vi xử lý) sang chân DGND (Mô đun LED 7 thanh)
- Nối chân 6 (VDD\_5V) của P9 (Mô đun vi xử lý) sang chân VDD5V (Mô đun LED 7 thanh)



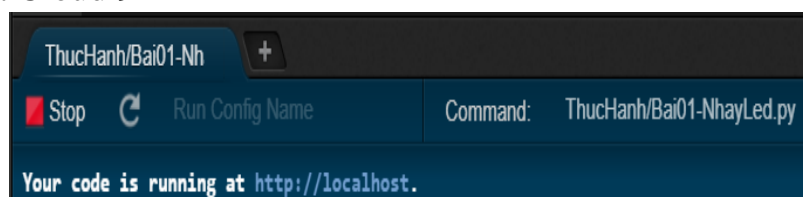
Hình 5.7. Kết nối mô đun Vi xử lý đến mô đun Led

**Bước 8:** Thực thi chương trình bằng cách nhấn nút Start màu xanh phía trên khung soạn thảo của Cloud 9



Hình 5.8 Nhấn nút RUN để thực thi chương trình trên KIT

**Bước 9:** Dừng chương trình bằng cách nhấn nút Stop phía dưới khung soạn thảo của Cloud 9



Hình 5.9. Vị trí nút Stop để dừng thực thi chương trình

#### 5.1.4. Thực hành

- Bài 1: Chuyển dây nhảy sang P9 và thực hiện lại chương trình nháy led
- Bài 2: Thực hiện cho nháy Led với các tần số khác nhau
- Bài 3: Tạo hiệu ứng Led sáng đuôi, sáng dần, sáng quay vòng, hai đèn đuổi nhau, sáng đếm nhị phân.



## 5.2. Đọc giá trị từ nút nhấn

### 5.2.1. Mục tiêu

- Làm quen với môi trường phát triển ứng dụng Cloud 9
- Làm quen với ngôn ngữ lập trình Python cho KIT
- Làm quen với quy tắc đặt địa chỉ và chế độ input của GPIO
- Kết nối được mô đun vi xử lý trung tâm tới mô đun nút nhấn và Led đơn
- Lập trình tắt, mở, nhấp nháy led đơn bằng cách nhấn nút tương ứng

### 5.2.2. Công cụ và phần mềm


- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9

### 5.2.3. Nội dung

#### Các cổng vào ra cơ bản (GPIO):

- KIT được thiết kế với 65 GPIO sử dụng như cổng vào hoặc ra số (digital)
- Các cổng GPIO được phân bố đều theo 2 Header là P8 và P9

## Cape Expansion Headers

P9					P8			
DGND	1	2	DGND		DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3		MMC1_DAT6	3	4	MMC1_DAT7
VDD_5V	5	6	VDD_5V		MMC1_DAT2	5	6	MMC1_DAT3
SYS_5V	7	8	SYS_5V		GPIO_66	7	8	GPIO_67
PWR_BTN	9	10	SYS_RESETN		GPIO_69	9	10	GPIO_68
UART4_RXD	11	12	GPIO_60		GPIO_45	11	12	GPIO_44
UART4_TXD	13	14	EHRPWM1A		EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B		GPIO_47	15	16	GPIO_46
SPI0_CS0	17	18	SPI0_D1		GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA		EHRPWM2A	19	20	MMC1_CMD
SPI0_D0	21	22	SPI0_SCLK		MMC1_CLK	21	22	MMC1_DAT5
GPIO_49	23	24	UART1_TXD		MMC1_DAT4	23	24	MMC1_DAT1
GPIO_117	25	26	UART1_RXD		MMC1_DAT0	25	26	GPIO_61
GPIO_115	27	28	SPI1_CS0		LCD_VSYNC	27	28	LCD_PCLK
SPI1_D0	29	30	GPIO_112		LCD_HSYNC	29	30	LCD_AC_BIAS
SPI1_SCLK	31	32	VDD_ADC		LCD_DATA14	31	32	LCD_DATA15
AIN4	33	34	GNDA_ADC		LCD_DATA13	33	34	LCD_DATA11
AIN6	35	36	AIN5		LCD_DATA12	35	36	LCD_DATA10
AIN2	37	38	AIN3		LCD_DATA8	37	38	LCD_DATA9
AIN0	39	40	AIN1		LCD_DATA6	39	40	LCD_DATA7
GPIO_20	41	42	ECAPPWM0		LCD_DATA4	41	42	LCD_DATA5
DGND	43	44	DGND		LCD_DATA2	43	44	LCD_DATA3
DGND	45	46	DGND		LCD_DATA0	45	46	LCD_DATA1

**LEGEND**

POWER/GROUND/RESET
AVAILABLE DIGITAL
AVAILABLE PWM
SHARED I2C BUS
RECONFIGURABLE DIGITAL
ANALOG INPUTS (1.8V)

Hình 5.10. Sơ đồ bố trí Header P8 và P9

# 65 possible digital I/Os

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BTN	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

Hình 5.11. Sơ đồ bố trí các GPIO

## ✚ Môi trường phát triển Cloud 9

Để khởi chạy vào Cloud9, ta chạy trên trình duyệt với địa chỉ IP của KIT tại cổng 3000

- + Kết nối KIT với máy tính qua cổng USB thì địa chỉ Cloud 9 là:

192.168.7.2 :3000

- + Kết nối KIT với cổng Ethernet, địa chỉ Cloud 9 sẽ là IP của KIT với cổng 3000

Ví dụ :

IP : 192.168.1.123

Địa chỉ Cloud 9 : 192.168.1.123 :3000

## ✚ Lập trình đọc giá trị của nút nhấn và điều khiển LED đơn

**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, tích vào biểu tượng dấu + để tạo file mới (Alt-N)

**Bước 5:** Nhấn **Ctrl-S**, khung hội thoại **Save as** hiện ra, tại thư mục **ThucHanh** và lưu tên file là: **Bai012-DocNutNhan.py** vào thư mục vừa tạo

**Bước 6:** Đánh code sau vào khung soạn thảo của Cloud 9

```
Bai02-DocNutNhan.py x +
1 import Adafruit_BBIO.GPIO as GPIO
2 import time
3
4 BTN1 = "P9_23"
5 BTN2 = "P9_25"
6 LED = "P8_13"
7
8 GPIO.setup(LED, GPIO.OUT) #Thiet lap dau ra
9 GPIO.setup(BTN1, GPIO.IN) #Thiet lap dau vao
10 GPIO.setup(BTN2, GPIO.IN) #Thiet lap dau vao
11
12 GPIO.output(LED, GPIO.HIGH) #Trang thai ban dau cua LED la tat
13
14 status = 0
15
16 while True:
17     if (GPIO.input(BTN1) == 0):
18         status = 1
19     if (GPIO.input(BTN2) == 0):
20         status = 2
21
22     if(status == 1):
23         GPIO.output(LED, GPIO.LOW)
24         time.sleep(0.5)
25         GPIO.output(LED, GPIO.HIGH)
26         time.sleep(0.5)
27     if(status == 2):
28         GPIO.output(LED, GPIO.HIGH)
29
```

Hình 5.12. Code mẫu điều khiển Led đơn

**Bước 7:** Kết nối mô đun Vi xử lý với mô đun Led đơn và nút nhấn

- Nối lần lượt các chân từ 23 đến 25 của P9 (Mô đun vi xử lý) sang từ SW1, SW2 ; chân 13 của P8 sang chân LED7 (Mô đun LED 7 thanh)
- Nối chân 1 (DGND) của P8 (Mô đun vi xử lý) sang chân DGND (Mô đun LED 7 thanh)
- Nối chân 6 (VDD\_5V) của P9 (Mô đun vi xử lý) sang chân VDD5V (Mô đun LED 7 thanh)

**Bước 8:** Thực thi chương trình bằng cách nhấn nút Start màu xanh phía trên khung soạn thảo của Cloud 9

**Bước 9:** Dừng chương trình bằng cách nhấn nút Stop phía dưới khung soạn thảo của Cloud 9

#### 5.2.4. Thực hành

Bài 1: Thay đổi nút nhấn và thực hiện lại chương trình

Bài 2: Lập trình với 4 nút nhấn và 8 Led. Mỗi nút nhấn sẽ điều khiển bật tắt một led riêng biệt.

## 5.3. Điều khiển LED 7 thanh

### 5.3.1. Mục tiêu

- Làm quen với ngôn ngữ lập trình Python cho KIT
- Phân loại và thực hiện được phương pháp giải mã Led 7 thanh
- Hiểu được cách thức hoạt động và điều khiển IC ghi dịch 74HC595
- Kết nối được mô đun vi xử lý trung tâm tới mô đun Led 7 thanh
- Lập trình hiển thị số trên màn hình Led 7 thanh

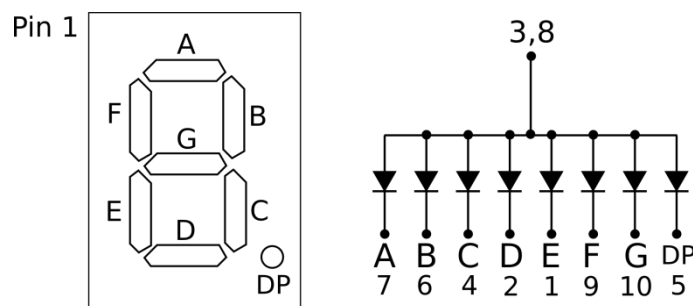
### 5.3.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9

### 5.3.3. Nội dung

#### Led 7 thanh

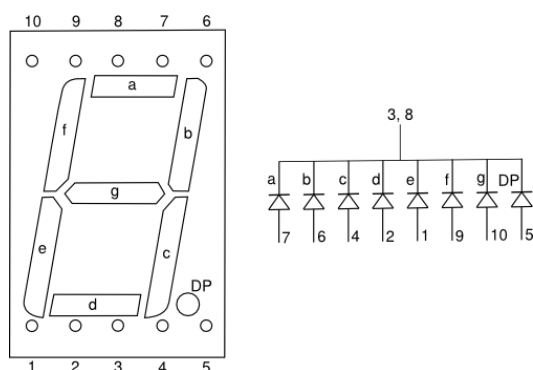
- LED 7 thanh có 2 loại:
  - + Chung cực dương (Anode chung): Mỗi đèn LED có 2 chân (1 dương 1 âm). Ở loại LED 7 đoạn này tất cả cực dương sẽ được nối chung cực dương. Để làm các đèn LED trong LED 7 đoạn sáng thì ta chỉ cần cấp cực âm vào các chân của đèn.



Hình 5.13 Cấu tạo của LED 7 thanh chung cực dương

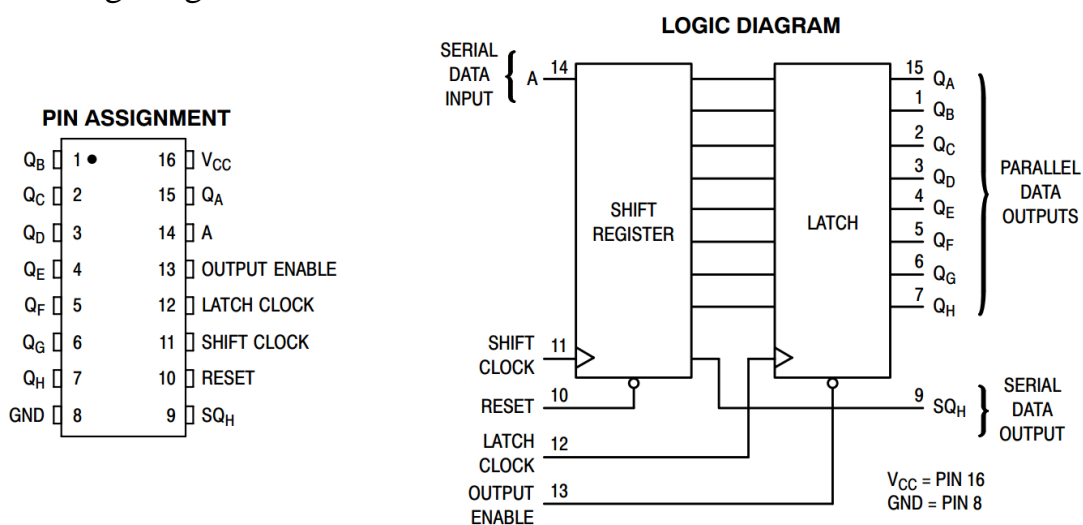
- + Chung cực âm (Cathode chung): Tương tự nhưng ngược lại và bạn cần đến 8 điện trở cho các chân dương của LED.
- Nguyên lý căn bản của LED 7 đoạn đó là cấp nguồn là nó sáng. Để nó sáng theo các số quy định và đơn giản trong lập trình, ta phải tiến hành tạo mã Led 7 thanh và đưa vào một mảng. Chỉ số của phần tử mảng chính là số hiển thị của LED.





Common Anode									
Decimal	Individual Segments Illuminated								
Digit	dp	g	f	e	d	c	b	a	HEX
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	0	1	0x99
5	1	0	0	1	0	0	1	0	0x92
6	1	0	0	0	0	0	1	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	0	0	0	0	0x90

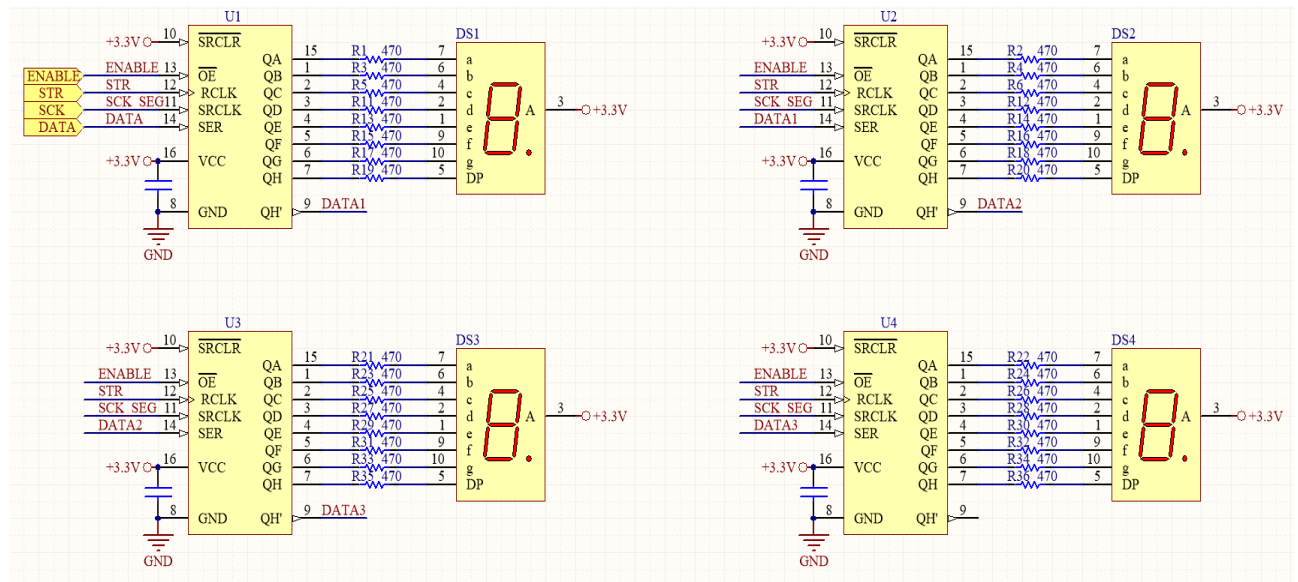
## Vi mạch ghi dịch 74HC595



Do đặc tính vào nối tiếp, nên để giao tiếp thì chỉ cần 3 chân của Vi xử lý nối với chân Shift Clock, Latch Clock và Data Input của 74HC595.

Do đặc tính ra song song, nên ta sử dụng các chân đầu ra từ QA cho đến QH nối với chân từ chân a cho đến chân h của Led 7 thanh.

Nếu có hơn một Led 7 thanh cần hiển thị, việc đơn giản chúng ta chỉ cần nối chân SQH của 74HC595 trước với chân Data In của 74HC595 sau.



Hình 5.17 Sơ đồ nguyên lý khối hiển thị LED 7 thanh có sử dụng 74HC595

### 🔧 **Lập trình hiển thị số đếm từ 0000 đến 9999**

**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 Bai03-Led7vsHC595.py và lưu vào thư mục ThucHanh

**Bước 5:** Đánh code sau vào khung soạn thảo của Cloud 9

```
import Adafruit_BBIO.GPIO as GPIO
import time
import datetime

STR    = "P8_16"
SCK    = "P8_17"
DATA   = "P8_18"
ENA    = "P8_26"

#Khai bao ma LED
ledCode = [0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90]

# Khai bao bien
soDem = 0
```

```

# ===== Thiet lap GPIO
=====
GPIO.setup(SCK, GPIO.OUT)
GPIO.setup(STR, GPIO.OUT)
GPIO.setup(DATA, GPIO.OUT)
GPIO.setup(ENA, GPIO.OUT)

GPIO.output(SCK,GPIO.LOW)
GPIO.output(STR,GPIO.LOW)
GPIO.output(DATA,GPIO.LOW)
GPIO.output(ENA,GPIO.LOW)

# ===== Xay dung ham con dieu khien LED
=====
def shiftingOut(sdata = 0):
    i = 0
    buffer = 0
    temp = 0
    for i in range(0,8):
        buffer = sdata
        sdata = sdata << 1
        temp = buffer & 0x80
        if(temp):
            GPIO.output(DATA,GPIO.HIGH)
        else:
            GPIO.output(DATA,GPIO.LOW)

        GPIO.output(SCK,GPIO.HIGH)
        time.sleep(0.0001)
        GPIO.output(SCK,GPIO.LOW)
    return

def shiftingLatch():
    GPIO.output(STR,GPIO.HIGH)
    time.sleep(0.0001)
    GPIO.output(STR,GPIO.LOW)
    return

def display(number = 0):
    ng = number/1000
    tr = (number%1000)/100
    ch = (number%100)/10
    dv = number%10

    shiftingOut(ledCode[dv])
    shiftingOut(ledCode[ch])
    shiftingOut(ledCode[tr])

```

```

shiftingOut(ledCode[ng])

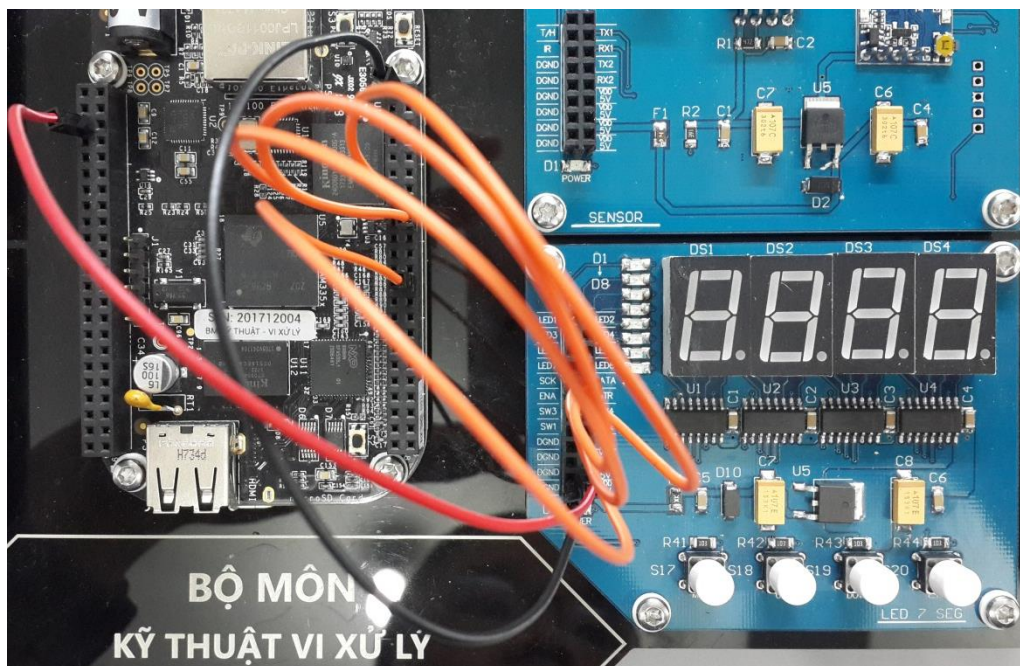
shiftingLatch()
return

while True:
    display(soDem)
    time.sleep(0.001)
    soDem = soDem + 1
    if(soDem >= 9999):
        soDem = 0

```

### Bước 6: Đấu nối

- Nối lần lượt các chân từ 16,17,18 và 26 của P8 của Mô đun vi xử lý sang lần lượt các chân STR, SCK, DATA, ENA của Mô đun LED 7 thanh
- Nối chân 2 (DGND) của P9 của Mô đun vi xử lý sang chân DGND của Mô đun LED 7 thanh
- Nối chân 6 (VDD\_5V) của P9 của Mô đun vi xử lý sang chân VDD5V của Mô đun LED 7 thanh



Hình 5.18 Kết nối mô đun Vi xử lý đến mô đun Led 7 thanh

### Bước 7: Thực thi chương trình

- Sử dụng nút **Start** để bắt đầu thực thi chương trình, theo dõi hoạt động của LED 7 thanh
- Sử dụng nút **Stop** để dừng chương trình

#### **5.3.4. Thực hành**

Bài 1: Viết lại chương trình với số đếm lùi từ 9999 về 0000

Bài 2: Viết chương trình hiển thị số lần bấm nút. Nếu nhấn nút nhấn 1 thì số đếm tiến lên một đơn vị, nếu nhấn nút nhấn 2 thì số đếm lùi đi một đơn vị. Số đếm nằm trong khoảng từ 0000 đến 9999.

## 5.4. Điều khiển màn hình GLCD

### 5.4.1. Mục tiêu

- Nhớ được nguyên lý điều khiển của màn hình GLCD
- Làm quen với giao tiếp SPI trên KIT
- Làm quen với thao tác điều khiển từ xa qua SSH
- Cài đặt được các thư viện hỗ trợ lập trình Python cho GLCD
- Sử dụng được thành thạo môi trường phát triển Cloud9
- Sử dụng Python để lập trình vẽ hình đồ họa và chữ lên màn GLCD

### 5.4.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9
- Kết nối vào Internet thông qua cổng Ethernet
- Phần mềm PuTTY để kết nối với KIT qua giao thức SSH
- Phần mềm truyền nhận file FileZilla qua giao thức SFTP

### 5.4.3. Nội dung

#### *Lập trình hiển thị trên màn hình GLCD*

**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 Bai04-GLCD.py và lưu vào thư mục ThucHanh

**Bước 5:** Đánh code sau vào khung soạn thảo của Cloud 9

```
import time
import Adafruit_Nokia_LCD as LCD
import Adafruit_GPIO.SPI as SPI

from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

# Beaglebone Black software SPI config:
DC    = "P9_15"
RST   = "P9_12"
SCLK  = "P9_11"
DIN   = "P9_14"
```

```

CS    = "P9_13"

# Software SPI usage (defaults to bit-bang SPI interface):
disp = LCD.PCD8544(DC, RST, SCLK, DIN, CS)

# Initialize library.
disp.begin(contrast=40)

# Clear display.
disp.clear()
disp.display()

# Create blank image for drawing.
# Make sure to create image with mode '1' for 1-bit color.
image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a white filled box to clear the image.
draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT),outline=255,
fill=255)

# Draw some shapes.
draw.ellipse((2,2,22,22), outline=0, fill=255)
draw.rectangle((24,2,44,22), outline=0, fill=255)
draw.polygon([(46,22), (56,2), (66,22)], outline=0, fill=255)
draw.line((68,22,81,2), fill=0)
draw.line((68,2,81,22), fill=0)

# Load default font.
font = ImageFont.load_default()

# Alternatively load a TTF font.
# Some nice fonts to try: http://www.dafont.com/bitmap.php
#font = ImageFont.truetype('Minecraftia.ttf', 8)
#font = ImageFont.load("arial.ttf")

# Write some text.
draw.text((8,30), "Hello World!", font=font)

# Display image.
disp.image(image)
disp.display()

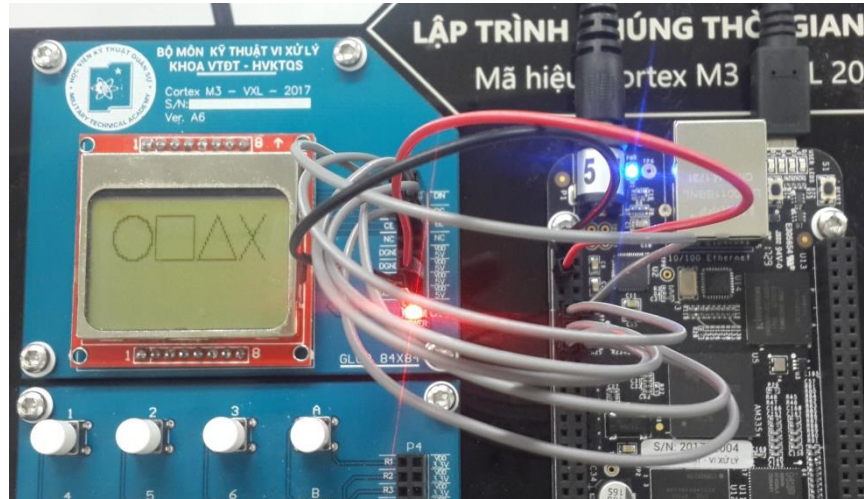
print('Press Ctrl-C to quit.')
while True:

```

```
time.sleep(1.0)
```

#### **Bước 6:** Đấu nối

- Nối lần lượt các chân từ 11 đến 15 của P9 của Mô đun vi xử lý sang lần lượt các chân CLK, RST, CS, DIN, DC của Mô đun GLCD
- Nối chân 2 (DGND) của P9 của Mô đun vi xử lý sang chân DGND của Mô đun GLCD
- Nối chân 6 (VDD\_5V) của P9 của Mô đun vi xử lý sang chân VDD5V của Mô đun GLCD



*Hình 5.19 Kết nối mô đun Vi xử lý đến mô đun GLCD*

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

- Sử dụng nút **Start** để bắt đầu thực thi chương trình, theo dõi hoạt động của GLCD
- Sử dụng nút **Stop** để dừng chương trình

#### **5.5.4. Thực hành**

Bài 1: Lập trình hiển thị giá trị nút nhấn lên màn GLCD

Bài 2: Lập trình hiển thị một hình ảnh lên màn GLCD.



## 5.5. Giao tiếp với bàn phím hexa

### 5.5.1. Mục tiêu

- Nhớ được nguyên lý hoạt động của bàn phím ma trận
- Nhớ được các phương pháp quét bàn phím ma trận
- Kết nối được mô đun vi xử lý với mô đun bàn phím
- Sử dụng được thành thạo môi trường phát triển Cloud9
- Sử dụng được ngôn ngữ Python để lập trình đọc được giá trị của phím nhấn và hiển thị giá trị lên màn hình LED7 thanh

### 5.5.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9

### 5.5.3. Nội dung

#### **Lập trình đọc giá trị nút nhấn từ bàn phím ma trận**

**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 **Bai05-HexaKeypad.py** và lưu vào thư mục **ThucHanh**

**Bước 5:** Đánh code sau vào khung soạn thảo của Cloud 9

```
import Adafruit_BBIO.GPIO as GPIO
import time

# Khai bao hang, cot cua ban phim
KEYPAD_ROW1 = "P9_11"
KEYPAD_ROW2 = "P9_12"
KEYPAD_ROW3 = "P9_13"
KEYPAD_ROW4 = "P9_14"

KEYPAD_COL1 = "P9_15"
KEYPAD_COL2 = "P9_16"
KEYPAD_COL3 = "P9_17"
KEYPAD_COL4 = "P9_18"

#Thiet lap che do vao ra
GPIO.setup(KEYPAD_ROW1, GPIO.OUT)
GPIO.setup(KEYPAD_ROW2, GPIO.OUT)
```

```

GPIO.setup(KEYPAD_ROW3, GPIO.OUT)
GPIO.setup(KEYPAD_ROW4, GPIO.OUT)

GPIO.setup(KEYPAD_COL1, GPIO.IN)
GPIO.setup(KEYPAD_COL2, GPIO.IN)
GPIO.setup(KEYPAD_COL3, GPIO.IN)
GPIO.setup(KEYPAD_COL4, GPIO.IN)

#Khoi tao gia tri ban dau cho hang
GPIO.output(KEYPAD_ROW1, GPIO.HIGH)
GPIO.output(KEYPAD_ROW2, GPIO.HIGH)
GPIO.output(KEYPAD_ROW3, GPIO.HIGH)
GPIO.output(KEYPAD_ROW4, GPIO.HIGH)

#Ham quet phim
def keypad_readkey():
    keyPressed = 255 #Phim duoc nhan

    #-----#
    GPIO.output(KEYPAD_ROW1, GPIO.LOW)
    GPIO.output(KEYPAD_ROW2, GPIO.HIGH)
    GPIO.output(KEYPAD_ROW3, GPIO.HIGH)
    GPIO.output(KEYPAD_ROW4, GPIO.HIGH)

    if(GPIO.input(KEYPAD_COL1) == 0):
        while (GPIO.input(KEYPAD_COL1) == 0):
            nop()
            keyPressed = 1
    elif (GPIO.input(KEYPAD_COL2) == 0):
        while (GPIO.input(KEYPAD_COL2) == 0):
            nop()
            keyPressed = 2
    elif (GPIO.input(KEYPAD_COL3) == 0):
        while (GPIO.input(KEYPAD_COL3) == 0):
            nop()
            keyPressed = 3
    elif (GPIO.input(KEYPAD_COL4) == 0):
        while (GPIO.input(KEYPAD_COL4) == 0):
            nop()
            keyPressed = 10

    #-----#
    GPIO.output(KEYPAD_ROW1, GPIO.HIGH)
    GPIO.output(KEYPAD_ROW2, GPIO.LOW)
    GPIO.output(KEYPAD_ROW3, GPIO.HIGH)
    GPIO.output(KEYPAD_ROW4, GPIO.HIGH)

```

```

if(GPIO.input(KEYPAD_COL1) == 0):
    while (GPIO.input(KEYPAD_COL1) == 0):
        nop()
        keyPressed = 4
elif (GPIO.input(KEYPAD_COL2) == 0):
    while (GPIO.input(KEYPAD_COL2) == 0):
        nop()
        keyPressed = 5
elif (GPIO.input(KEYPAD_COL3) == 0):
    while (GPIO.input(KEYPAD_COL3) == 0):
        nop()
        keyPressed = 6
elif (GPIO.input(KEYPAD_COL4) == 0):
    while (GPIO.input(KEYPAD_COL4) == 0):
        nop()
        keyPressed = 11

#-----#
GPIO.output(KEYPAD_ROW1, GPIO.HIGH)
GPIO.output(KEYPAD_ROW2, GPIO.HIGH)
GPIO.output(KEYPAD_ROW3, GPIO.LOW)
GPIO.output(KEYPAD_ROW4, GPIO.HIGH)

if(GPIO.input(KEYPAD_COL1) == 0):
    while (GPIO.input(KEYPAD_COL1) == 0):
        nop()
        keyPressed = 7
elif (GPIO.input(KEYPAD_COL2) == 0):
    while (GPIO.input(KEYPAD_COL2) == 0):
        nop()
        keyPressed = 8
elif (GPIO.input(KEYPAD_COL3) == 0):
    while (GPIO.input(KEYPAD_COL3) == 0):
        nop()
        keyPressed = 9
elif (GPIO.input(KEYPAD_COL4) == 0):
    while (GPIO.input(KEYPAD_COL4) == 0):
        nop()
        keyPressed = 12

#-----#
GPIO.output(KEYPAD_ROW1, GPIO.HIGH)
GPIO.output(KEYPAD_ROW2, GPIO.HIGH)
GPIO.output(KEYPAD_ROW3, GPIO.HIGH)
GPIO.output(KEYPAD_ROW4, GPIO.LOW)

if(GPIO.input(KEYPAD_COL1) == 0):

```

```

        while (GPIO.input(KEYPAD_COL1) == 0):
            nop()
            keyPressed = 42
    elif (GPIO.input(KEYPAD_COL2) == 0):
        while (GPIO.input(KEYPAD_COL2) == 0):
            nop()
            keyPressed = 0
    elif (GPIO.input(KEYPAD_COL3) == 0):
        while (GPIO.input(KEYPAD_COL3) == 0):
            nop()
            keyPressed = 35
    elif (GPIO.input(KEYPAD_COL4) == 0):
        while (GPIO.input(KEYPAD_COL4) == 0):
            nop()
            keyPressed = 13

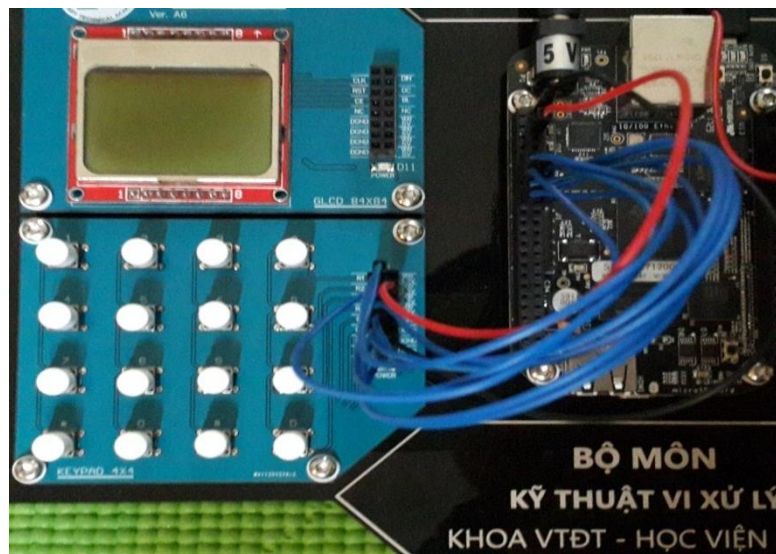
    return keyPressed

while True:
    key = keypad_readkey()
    print(key)

```

### **Bước 6: Đầu nối**

- Nối lần lượt các chân từ 11 đến 18 của P9 của Mô đun vi xử lý sang lần lượt các chân R1, R2, R3, R4, C1, C2, C3, C4 của Mô đun Keypad 4x4
- Nối chân 2 (DGND) của P9 của Mô đun vi xử lý sang chân DGND của Mô đun Keypad 4x4
- Nối chân 4 (VDD\_3V3) của P9 của Mô đun vi xử lý sang chân VDD5V của Mô đun Keypad 4x4



Hình 5.20 Kết nối mô đun Vi xử lý đến mô đun Keypad4x4

**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

**5.5.4. Thực hành**

Bài 1: Đọc giá trị của phím nhấn hiển thị lên GLCD

Bài 2: Đọc giá trị của phím nhấn hiển thị lên LED 7 thanh

## CHƯƠNG 6 – ĐIỀU KHIỂN MÔ – ĐUN CẢM BIẾN TRONG HỆ THỐNG NHÚNG THỜI GIAN THỰC

### 6.1. Giao tiếp với cảm biến nhiệt độ và độ ẩm

#### 6.1.1. Mục tiêu

- Nhớ được nguyên lý hoạt động của cảm biến nhiệt độ, độ ẩm DHT11
- Làm quen với giao tiếp 1-Wire trên KIT
- Làm quen với thao tác điều khiển từ xa qua SSH
- Cài đặt được các thư viện hỗ trợ lập trình Python cho DHT11
- Sử dụng được thành thạo môi trường phát triển Cloud9
- Sử dụng Python để lập trình đọc nhiệt độ, độ ẩm, hiển thị lên màn hình Led 7 thanh, màn GLCD hoặc màn hình máy tính.

#### 6.1.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9

#### 6.1.3. Nội dung

##### *Lập trình đọc giá trị nhiệt độ, độ ẩm*

**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 **Bai05-HexaKeypad.py** và lưu vào thư mục **ThucHanh**

**Bước 5:** Đánh code sau vào khung soạn thảo của Cloud 9

```
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_DHT
import time

# Khai báo chân của module LED 7 thanh
STR    = "P8_16"
SCK    = "P8_17"
DATA   = "P8_18"
```

```

ENA    = "P8_26"

#Khai bao chan cua cam bien DHT11
sensorPin = "P8_15"  #T/H Pin

#Khai bao ma cua Led 7 thanh
ledCode = [0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90]

#Khai bao loai cam bien
sensorType = Adafruit_DHT.DHT11

#          ===== Thiet lap GPIO
=====
GPIO.setup(SCK, GPIO.OUT)
GPIO.setup(STR, GPIO.OUT)
GPIO.setup(DATA, GPIO.OUT)
GPIO.setup(ENA, GPIO.OUT)

GPIO.output(SCK,GPIO.LOW)
GPIO.output(STR,GPIO.LOW)
GPIO.output(DATA,GPIO.LOW)
GPIO.output(ENA,GPIO.LOW)

# ===== Xay dung ham con dieu khien LED =====
def shiftingOut(sdata = 0):
    i = 0
    buffer = 0
    temp = 0
    for i in range(0,8):
        buffer = sdata
        sdata = sdata << 1
        temp = buffer & 0x80
        if(temp):
            GPIO.output(DATA,GPIO.HIGH)
        else:
            GPIO.output(DATA,GPIO.LOW)

        GPIO.output(SCK,GPIO.HIGH)
        time.sleep(0.0001)
        GPIO.output(SCK,GPIO.LOW)
    return

```



```

def shiftingLatch():
    GPIO.output(STR,GPIO.HIGH)
    time.sleep(0.0001)
    GPIO.output(STR,GPIO.LOW)
    return

def display(number = 0):
    ng = number/1000
    tr = (number%1000)/100
    ch = (number%100)/10
    dv = number%10

    shiftingOut(ledCode[dv])
    shiftingOut(ledCode[ch])
    shiftingOut(ledCode[tr])
    shiftingOut(ledCode[ng])

    shiftingLatch()
    return

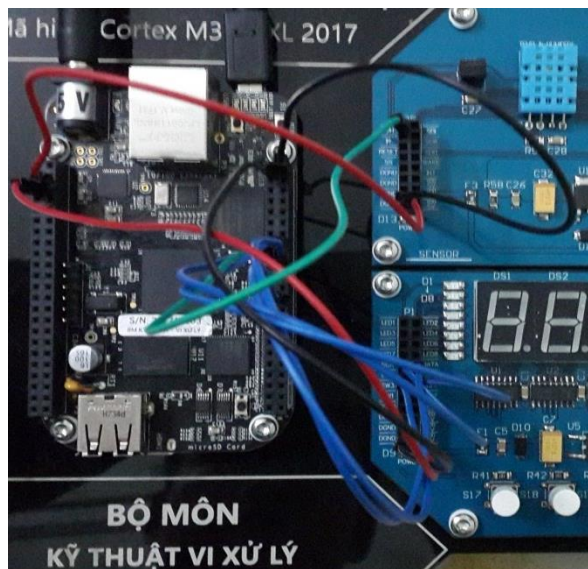
while True:
    # Doc Do am va Nhiet do tu cam bien thong qua thu vien
    Adafruit_DHT
    # Ham read_retry se doc gia tri Do am va Nhiet do cua
    cam bien
    # neu khong thanh cong se thu 15 lanj moi lan each nhau
    2 giay
    humidity,temperature=Adafruit_DHT.read_retry(sensorType,
    sensorPin)

    # Kiem tra gia tri tra ve tu cam bien khac NULL
    if (do_am is not None) and (nhiet_do is not None):
        print ("Nhiet Do = {0:0.1f} Do Am =
{1:0.1f}\n").format(temperature, humidity);
        display(temperature)
        time.sleep(2)
        print ("RASPI.VN cho 2 giay de tiep tuc do ...\n");
        display(humidity)
        time.sleep(2)
    else:
        # Loi :(
        print("Loi khong the doc tu cam bien DHT11 :(\n")

```

### Bước 6: Đấu nối

- Nối lần lượt các chân từ 16,17,18 và 26 của P8 của Mô đun vi xử lý sang lần lượt các chân STR, SCK, DATA, ENA của Mô đun LED 7 thanh
- Nối chân 15 của P8 sang chân số 1 (T/H) của khối Mô đun cảm biến nhiệt độ, độ ẩm
- Nối chân 1, 2 (DGND) của P8 của Mô đun vi xử lý sang chân DGND của Mô đun LED 7 thanh và Mô đun cảm biến nhiệt độ, độ ẩm
- Nối chân 5, 6 (VDD\_5V) của P9 của Mô đun vi xử lý sang chân VDD5V của Mô đun LED 7 thanh và Mô đun cảm biến nhiệt độ, độ ẩm.



Hình 6.1. Kết nối mô đun Vi xử lý đến mô đun Led 7 thanh  
và mô đun cảm biến nhiệt độ, độ ẩm

### Bước 7: Thực thi chương trình

- Sử dụng nút **Start** để bắt đầu thực thi chương trình, theo dõi hoạt động của LED 7 thanh
- Sử dụng nút **Stop** để dừng chương trình.

#### 6.1.4. Thực hành

Bài 1: Đọc nhiệt độ và độ ẩm, hiển thị lên màn GLCD

Bài 2: Đọc nhiệt độ, độ ẩm, cài đặt khoảng cảnh báo nhiệt độ, độ ẩm bằng bàn phím hexa, hiển thị lên màn GLCD.

## 6.2. Truyền thông không dây sử dụng mô đun Zibee CC2530 TI

### 6.2.1. Mục tiêu

- Nhớ được các bước đặt ban đầu cho mô đun Zigbee CC2530
- Nhớ được các bước kích hoạt truyền thông UART trên KIT
- Sử dụng SSH để truy cập vào KIT và cài đặt được các thư viện cần thiết
- Sử dụng được thành thạo môi trường phát triển Cloud9
- Sử dụng Python để lập trình truyền nhận dữ liệu từ mô đun không dây Zigbee CC2530 theo giao thức P2P, hiển thị giá trị truyền nhận lên màn hình máy tính.

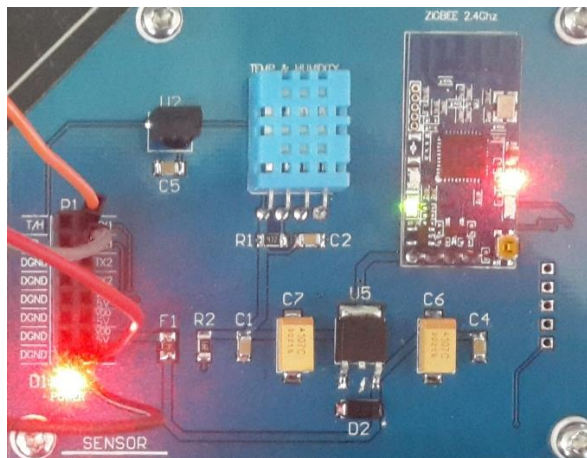
### 6.2.2. Công cụ và phần mềm

- KIT thực hành lập trình nhúng thời gian thực (Cortex M3 – VXL2017)
- Cáp USB kết nối với máy tính
- Dây nhảy để kết nối các mô đun
- Môi trường lập trình nền Web Cloud 9
- Phần mềm PuTTY và FileZillar.

### 6.2.3. Nội dung

#### Giới thiệu về mô đun Zigbee CC2530

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...

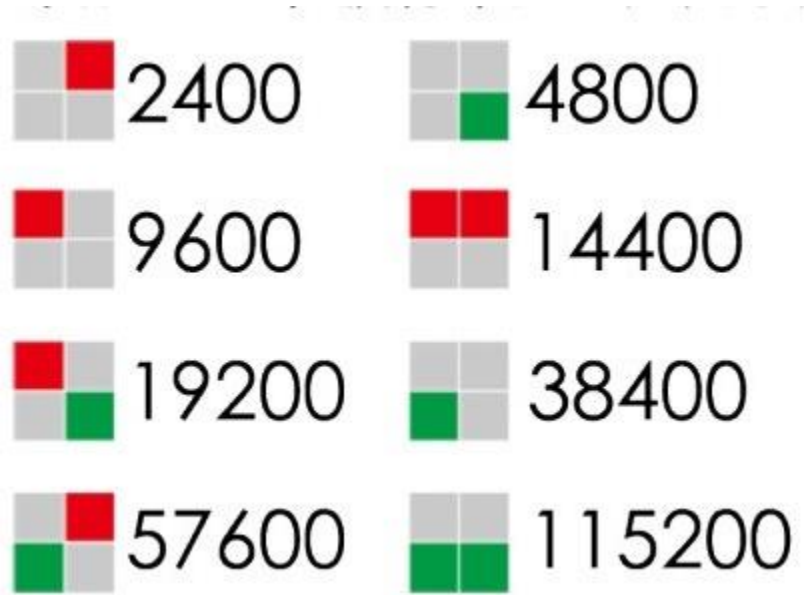


Hình 6.2. Mô đun Zibee CC253

## **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 6.3. 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.

#### **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

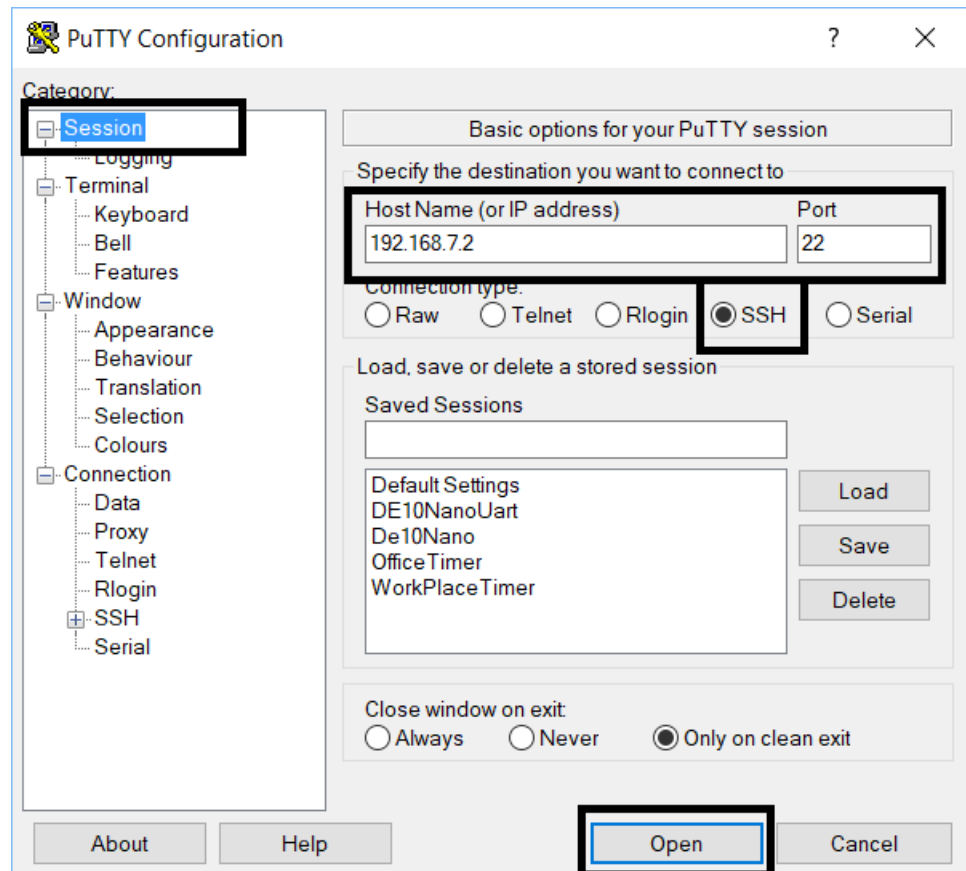
UART	RX	TX	CTS	RTS	Device
UART1	P9_26	P9_24	P9_20	P9_19	/dev/ttyO1 hoặc /dev/ttyS1
UART2	P9_22	P9_21			/dev/ttyO2 hoặc /dev/ttyS2
UART3		P9_42	P8_36	P8_34	/dev/ttyO3 hoặc /dev/ttyS3
UART4	P9_11	P9_13	P8_35	P8_33	/dev/ttyO4 hoặc /dev/ttyS4
UART5	P8_38	P8_37	P8_31	P8_32	/dev/ttyO5 hoặc /dev/ttyS5

Bảng 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 6.4. 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 v3.8.x
#cape_disable=capemgr.disable_partno=
#cape_enable=capemgr.enable_partno=

##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 6.5. Sửa file cấu hình hệ thống uEnv.txt để kích hoạt UART

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

```
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

```
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 6.6. 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 6.7. Vô hiệu hóa uboot\_overlays để có thể truy cập vào slots

### Cài đặt các thư viện cần thiết

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
```

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

**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 **Bai08-ZigbeeUart.py** và lưu vào thư mục **ThucHanh**

**Bước 5:** Đánh code sau vào khung soạn thảo của Cloud 9

```
# 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 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
#Khai bao chan cua cam bien DHT11
sensorPin = "P8_15" #T/H Pin
#Khai bao frame truyen cua UART
uartTX = [0, 0, 0, 0]
uartRX = [0,0,0,0,0,0,0,0,0,0]
strRX = ""
#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 loai cam bien
sensorType = Adafruit_DHT.DHT11

# ===== 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=47)
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!")

# ===== Ham xu ly doc nhiet do =====
def processDHT11():
    # Su dung bien Global
    global t1, t1Old, h1, h1Old

    #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()
            time.sleep(1)
        else:
            print("Loi khong the doc tu cam bien DHT11 :(\n")
            return

# ===== Ham xu ly truyen =====
def processTX(indexTX = 0):
    # Su dung bien global
    global uartTX
    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():
    # Cho phép thay doi gia tri cua bien toan cuc (Global)
    global h2, t2, strRX
    checksum = 0
    # Doc chuoai nhan ve
    if(ser.inWaiting()):
        strRX = ser.readline()
    # Chuyen ky tu xuong dong "\n" thanh ":" truoc khi tach

```

```

strRX = strRX.replace("\n","")

# Tach gia tri luu tru thoi gian
strRxData = strRX.split(':')
# Xoa cac ki tu Null
strRxData = filter(None, strRxData)
# 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)
    # print lenRX
    if(lenRX == numOfSent):
        # Tinh checksum
        for i in range(0, lenRX-1):
            checksum = checksum + arrRxData[i]
        # Kiem tra checksum co bang gia tri cuoi cung
khong
        if(checksum == arrRxData[numOfSent - 1]):
            #Kiem tra IndexRX
            if(arrRxData[0] == indexRX):
                t2 = arrRxData[1]
                h2 = arrRxData[2]
                nokiaTHDisplay()
            else:
                print("Ma kiem tra loi khong khop")
                time.sleep(0.2)
        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 phép thay đổi giá trị của biến toàn cục (Global)
    global t2Old, h2Old
    #Kiểm tra nhiệt độ, độ ẩm có thay đổi so với giá trị cũ
    hay không, nếu không thì không cập nhật lên LCD Nokia
    if((t2 != t2Old) or (h2 != h2Old)):
        t2Old = t2
        h2Old = h2
        # Xóa giá trị cũ để ghi vào giá trị mới
        draw.rectangle((58,18,72,38), outline=255,
fill=255)
        disp.image(image)
        disp.display()

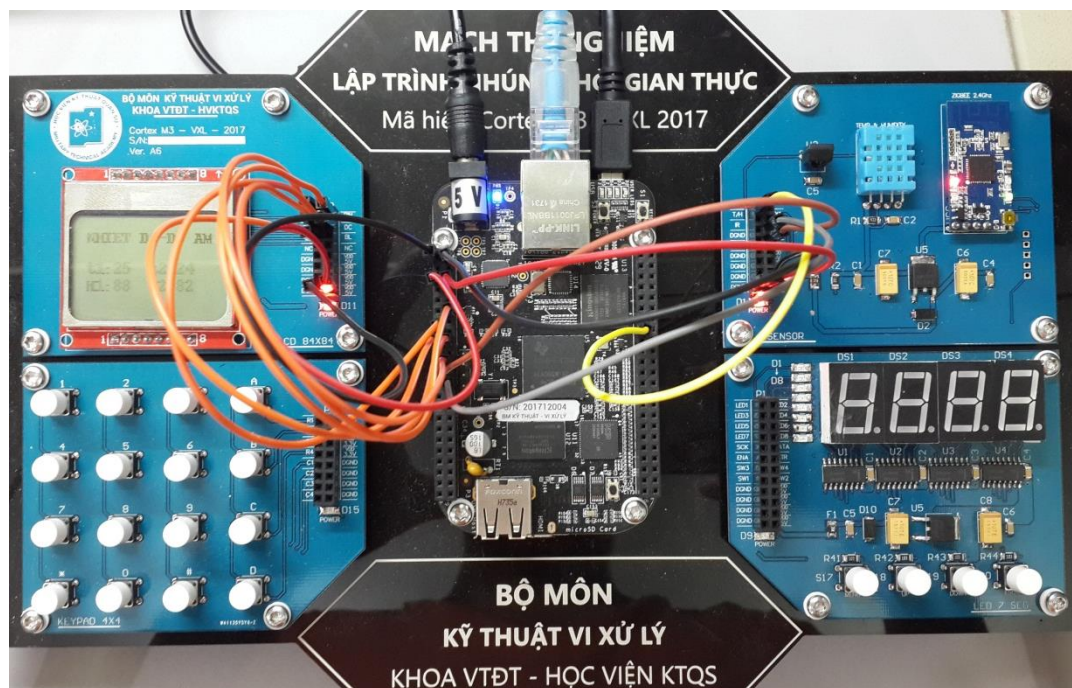
        print ("t2 = {0:0.1f}  H2 = {1:0.1f}\n").format(t2, h2);
        draw.text((42,18), 't2:%0.2s' %(t2), font=font)
        draw.text((42,28), 'H2:%0.2s' %(h2), font=font)
        disp.image(image)
        disp.display()
        return

# ===== Chương trình chính =====
# Địa chỉ của thiết bị
indexRX = 4
# Số dữ liệu truyền đi
numOfSent = 4
while True:
    processDHT11()
    time.sleep(0.5)
    processTX(5)
    time.sleep(0.5)
    processRX()
    time.sleep(1)
    strRX = ""
    t1 = 0
    h1 = 0
    t2 = 0
    h2 = 0

```

### 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 Sensor
- Nối chân 15 của P8 sang chân số 1 (T/H) của khối Mô đun cảm biến nhiệt độ, độ ẩm
- 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 Sensor
- Nối chân 5, 6 (VDD\_5V) của P9 của Mô đun vi xử lý sang chân VDD5V của khối GLCD và Mô đun Sensor.



Hình 8. Sơ đồ kết nối sử dụng Mô đun Zibee CC253

### 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

#### 6.2.4. Thực hành

Bài 1: Thực hiện truyền nhận Broadcast cho từ 3 KIT trở lên

## Bài 2: Thiết lập mạng Zigbee trên các KIT.

Mạch thí nghiệm Kỹ thuật Vi xử lý lập trình nhúng thời gian thực ngoài những ứng dụng được đề cập ở trên còn có rất nhiều ứng dụng nâng cao khác có thể khai thác, nghiên cứu, đáp ứng được nhu cầu nghiên cứu khoa học của cán bộ giáo viên trẻ và sinh viên. Các hướng nâng cao như: Điều khiển thiết bị thông qua giao diện Web, đọc dữ liệu cảm biến nhiệt độ, độ ẩm hiển thị qua giao diện Web, truyền nhận và điều khiển thiết bị thông qua Internet hay lập trình Arduino thông qua Mạch thí nghiệm này.