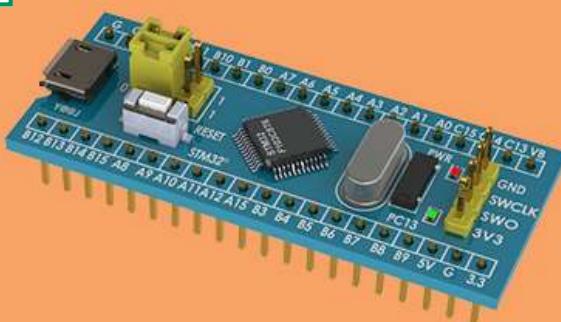


**LẬP TRÌNH STM32**

Bài 17: Lập trình STM32 USB HID chuột máy tính

POSTED ON 17/03/2021 BY KHUÊ NGUYỄN

17
Th3**Khuê Nguyễn Creator**

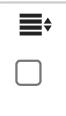
Bài 17: STM32 USB HID giả lập chuột máy tính

Các thiết bị USB HID rất gần gũi với chúng ta ví dụ như chuột máy tính, bàn phím, sound card... Vì điều khiển STM32 hỗ trợ giao thức HID Device giúp chúng ta có thể lập trình tạo ra các sản phẩm giống như những thiết bị đó. Bài hôm nay mình sẽ nói đến USB HID Mouse, làm thế nào để nó có thể truyền nhận dữ liệu với máy tính, hãy cùng tìm hiểu nhé!

Bài 17 trong Serie Học lập trình STM32 từ A tới Z

Mục Lục

- 1. USB HID là gì?
- 2. Cách giao tiếp với tất cả các thiết bị USB HID
 - 2.1. Human Interface Device Class hoạt động như thế nào?
 - 2.2. Cấu trúc của Report Descriptor
- 3. Các tool dùng trong lập trình STM32 USB HID
- 4. Cách biến Kit Bluepill thành chuột máy tính với USB HID
- 5. Lập trình STM32 USB HID như 1 con chuột (Mouse) điều khiển con trỏ máy tính
 - 5.1. Cấu hình STM32 USB HID trên Cube MX
 - 5.2. Lập trình STM32 USB HID trên Keil C
 - 5.3. Kết nối và chạy thử USB HID
 - 5.4. Tinh chỉnh code (Calibration)
- 6. Kết
 - 6.1. Related posts:



USB HID là gì?

HID (viết tắt của Human Interface Device) là một tiêu chuẩn cho các thiết bị máy tính được vận hành bởi con người. Tiêu chuẩn này cho phép dễ dàng sử dụng các thiết bị này mà không cần bất kỳ phần mềm hoặc trình điều khiển bổ sung nào.

HID là một tiêu chuẩn được tạo ra nhằm đơn giản hóa quá trình cài đặt các thiết bị đầu vào thông qua từng giao thức cụ thể cho từng thiết bị như chuột, bàn phím,... Một thiết bị tuân thủ HID bao gồm “gói dữ liệu” có thể chứa tất cả các hành động của thiết bị.

Ví dụ: Bàn phím có thể có một phím để điều chỉnh âm lượng. Khi nhấn phím đó, “bộ mô tả HID” sẽ cho máy tính biết mục đích của hành động đó được lưu trữ trong các gói tin ở đâu và lệnh đó sẽ được thực thi.

Cách giao tiếp với tất cả các thiết bị USB HID

Trước tiên nếu bạn chưa bao giờ điều khiển một thiết bị USB thì hãy quay lại Bài 16 để đọc bài viết và tài liệu USB in a Nutshell để có cái nhìn tổng quát nhất.

Thiết bị HID và Host (máy chủ) giao tiếp với nhau qua kiểu Control Transfer (hay Endpoint 0). Sử dụng Ngắt tại chiều IN và tùy chọn ở chiều Out. Đặc tả của lớp HID cho phép chúng có thể truyền dữ liệu ở cả tốc độ low speed , full speed và high speed.

Human Interface Device Class hoạt động như thế nào?

Trước khi máy chủ có thể nói chuyện với thiết bị, nó cần biết cách sử dụng hoặc ứng dụng của thiết bị này là gì? Dữ liệu của nó được tổ chức như thế nào? và Dữ liệu thực sự đo lường điều gì?

Lấy ví dụ: Nếu thiết bị của bạn là một con chuột máy tính, các nút bấm và tọa độ sẽ điều khiển Pointer trên màn hình. Sự kiện click hoặc right click sẽ làm gì, scroll sẽ làm gì. Để tất cả các sự kiện đó được xảy ra, trình điều khiển lớp HID phải biết rõ:

- Thiết bị HID kết nối sẽ điều khiển Pointer của máy tính
- Có 3 nút trên thiết bị và khi nhấn tương ứng với các hàm của Pointer
- Có 2 byte dữ liệu tọa độ sẽ thay đổi tọa độ của Pointer

Tất cả các thông tin này sẽ được mô tả trong phần Report Descriptor. Khi trình điều khiển phân tích cú pháp của Report Descriptor nó sẽ hiểu được khi thiết bị chuột máy tính truyền dữ liệu lên, dữ liệu nào sẽ thuộc ứng dụng nào của máy tính. (Tương tự bạn phân luồng dữ liệu UART vậy).

Khi một thiết bị HID được kết nối, Host sẽ tạo ra 1 Request đó là GET_DESCRIPTOR, sau khi hoàn tất quá trình. Chuột máy tính và máy tính sẽ giao tiếp với nhau mà không cần thêm driver gì cả.

Cấu trúc của Report Descriptor

Bộ Report Descriptor được mô tả bởi chuỗi các mục, các mục này mô tả dữ liệu sẽ truyền đi khi thiết bị USB HID device truyền hoặc nhận. Mỗi mục bắt đầu bằng tiền tố là 1 Byte quy định vai trò của mục và độ dài dữ liệu của nó.

Mỗi mục chia làm 3 loại thẻ chính:

- **Main:** Mô tả thực tế dữ liệu truyền đi và nơi dữ liệu được sử dụng. Các thẻ Global và Local có chức năng bổ nghĩa cho Main
- **Global:** Mô tả thuộc tính của tất cả các thẻ Main phía sau nó, cho đến khi có 1 thẻ Global khác xuất hiện
- **Local:** Thẻ này mô tả thuộc tính của thẻ Main phía sau nó.

Mỗi loại thẻ bao gồm một số loại chính như:

Với Main item:

- **Input:** Mô tả dữ liệu truyền từ thiết bị lên host như sự kiện nhấn nút, dữ liệu cảm biến, dữ liệu của nhà phát hành muốn gửi
- **Output:** Mô tả dữ liệu từ Host truyền về thiết bị như điều khiển led, động cơ
- **Feature:** Mô tả dữ liệu được truyền đi được sử dụng để cấu hình cài đặt thiết bị như, tăng giảm tốc độ nháy của led, tốc độ động cơ ...
- **Collection và End Collection:** Mỗi thiết bị HID phải có 1 bộ sưu tập ứng dụng (Application Collection), để trình xử lý có thể biết được dữ liệu đang sử dụng trong ứng dụng nào. Ví dụ: 1 thiết bị có 3 tín năng chuột, bàn phím và nút nguồn thì phải có 3 Application Collection để phân biệt dữ liệu gửi đi và trả về.

Với Global Item:

- **Usage Page:** Mô tả danh mục cao nhất của thiết bị như Generic Desktop Controls (điều khiển thiết bị để bàn), Game control, điện thoại
- **Logical Minimum:** Giá trị số nguyên nhỏ nhất của Main Item
- **Logical Maximum:** Giá trị số nguyên lớn nhất của Maih Item

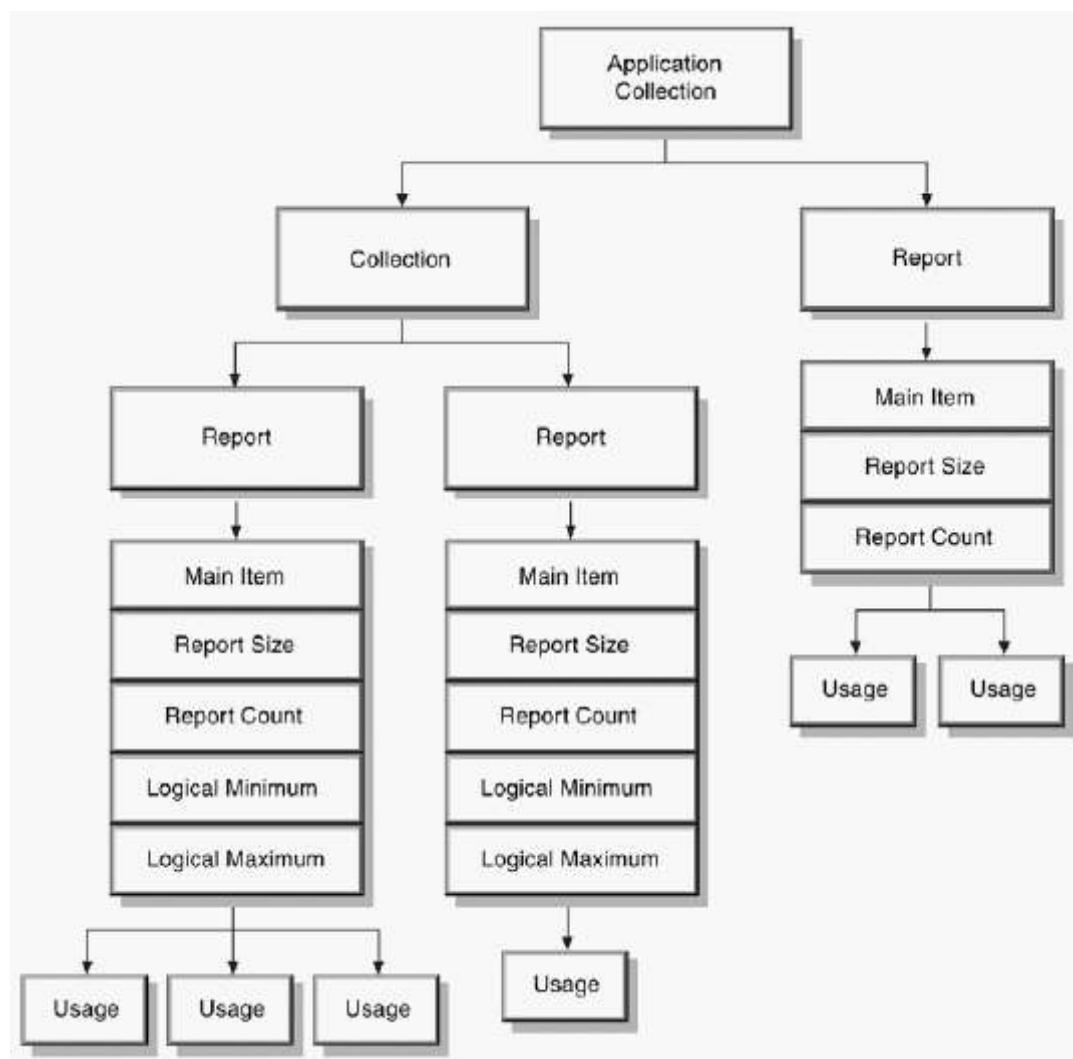
- **Report Size:** Kích thước của Main Item (tính theo Bit)
- **Report Count:** Số lượng Main Item

Với Local Item:

- **Usage:** Mô tả nhỏ hơn về lớp Usage Page: Ví dụ: như Usage Page là Generic Desktop Controls thì Usage có thể là *System Control hoặc Application control*.

Mỗi thẻ mục sẽ được phân loại tương ứng với 1 mã từ 0 – 255 (1 Byte).

Cấu trúc của Report Descriptor như sau:



Lấy ví dụ về Report Descriptor cho chuột máy tính như sau:

```

/*********************  

 * Class specific descriptor - HID Report descriptor  

*****  

const uint8_t hid_rpt0[] =  

{  

    0x05, 0x01, /* Usage Page (Generic Desktop) */  

    0x09, 0x02, /* Usage (Mouse) */  

    0xA1, 0x01, /* Collection (Application) */  

    0x09, 0x01, /* Usage (Pointer) */  

    0xA1, 0x00, /* Collection (Physical) */  

    0x05, 0x09, /* Usage Page (Buttons) */  

    0x19, 0x01, /* Usage Minimum (01) */  

    0x29, 0x03, /* Usage Maximum (03) */  

    0x15, 0x00, /* Logical Minimum (0) */  

    0x25, 0x01, /* Logical Maximum (1) */  

    0x95, 0x03, /* Report Count (3) */  

    0x75, 0x01, /* Report Size (1) */  

    0x81, 0x02, /* Input (Data, Variable, Absolute) */  

    0x95, 0x01, /* Report Count (1) */  

    0x75, 0x05, /* Report Size (5) */  

    0x81, 0x01, /* Input (Constant) ;5 bit padding */  

    0x05, 0x01, /* Usage Page (Generic Desktop) */  

    0x09, 0x30, /* Usage (X) */  

    0x09, 0x31, /* Usage (Y) */  

    0x15, 0x81, /* Logical Minimum (-127) */  

    0x25, 0x7F, /* Logical Maximum (127) */  

    0x75, 0x08, /* Report Size (8) */  

    0x95, 0x02, /* Report Count (2) */  

    0x81, 0x06, /* Input (Data, Variable, Relative) */  

    0xC0, 0xC0
};  


```

1st byte
of HID

2nd & 3rd
bytes of HID
Report

Phần khoanh đỏ cấu hình 3 nút nhấn của chuột, phần khoanh xanh cấu hình tọa độ của chuột.

Phần Usage Page và Usage xác định kiểu thiết bị đó là Mouse và thuộc máy tính để bàn (Generic Desktop).

Các bạn có thể tham khảo link này để phân tích 1 RD:

https://www.crifan.com/files/doc/docbook/usb_hid/release/webhelp/hid_report_exampl

Các tool dùng trong lập trình STM32 USB HID

Phần mềm gửi và nhận dữ liệu HID Terminal:

Link download:[HID Terminal](#)

Tools giải mã report descriptor.

<http://eleccelerator.com/usbdescreqparser/>

Cách biến Kit Bluepill thành chuột máy tính với USB HID

Để sử dụng kit stm32f103c8t6 Bluepill thành chuột máy tính chúng ta cần phân tích dữ liệu truyền lên máy tính của 1 con chuột.

Cấu trúc dữ liệu truyền lên bao gồm:

- 3 phím bấm: Right, Left, Scroll với 2 mức logic 0 và 1
- 2 biến tọa độ X và Y với mức logic từ -127 đến 127

Chúng ta sẽ sử dụng Joystick đọc ADC 2 kênh X Y và nút nhấn trên nó sẽ tương ứng với phím Left Click.



Mỗi khi có sự kiện nhấn nút hoặc di chuyển, chúng ta sẽ gửi dữ liệu theo gói thông qua USB HID.

Lập trình STM32 USB HID như 1 con chuột (Mouse) điều khiển con trỏ máy tính

Trong bài này mình sẽ sử dụng Joystick đọc giá trị ADC biểu thị trục X, Y. Và nút nhấn trên Joystick biểu thị cho Left_Button trên chuột máy tính.

Cấu hình STM32 USB HID trên Cube MX

Mở CubeMx, chọn chip STM32f103C8T6, trong System Core

SYS Mode and Configuration

Mode

Debug Serial Wire

System Wake-Up

Timebase Source SysTick

Configuration

⚠ Warning: This IP has no parameters to be configured.

SYS chọn Debug Serial Wire: cho phép debug qua stlink

RCC Mode and Configuration

Mode

High Speed Clock (HSE) Crystal/Ceramic Resonator

Low Speed Clock (LSE) Disable

Master Clock Output

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

System Parameters

- VDD voltage (V) 3.3 V
- Prefetch Buffer Enabled
- Flash Latency(WS) 1 WS (2 CPU cycle)

RCC Parameters

RCC chọn HSE: Thạch anh ngoại

The screenshot shows the STM32CubeMX software interface. On the left, there's a tree view of peripheral categories: System Core, DMA, GPIO, IWDG, NVIC, RCC, SYS, and WWDG. Under Analog, ADC1 and ADC2 are listed. In the main area, the Configuration tab is selected, showing a table for pin PA2. The table includes columns for Pin Name, Signal on, GPIO output, GPIO mode, GPIO Pull-up/Pull-down, Maximum value, User Label, and Modified. PA2 is configured as an input with pull-up. A search bar and a checkbox for 'Show only Modified Pins' are also present.

Pin Name	Signal on	GPIO output	GPIO mode	GPIO Pull-up/Pull-down	Maximum...	User Label	Modified
PA2	n/a	n/a	Input mode	Pull-up	n/a	Left_button	<input checked="" type="checkbox"/>

PA2 Configuration :

- GPIO mode: Input mode
- GPIO Pull-up/Pull-down: Pull-up
- User Label: Left_button

GPIO chọn PA2 là nút nhấn Pull Up

Trong Tab Analog ta sẽ cấu hình ADC1 để đọc giá trị Joystick

ADC1 Mode and Configuration

Mode

- IN0
- IN1
- IN2
- IN3

Configuration

Reset Configuration

- NVIC Settings
- DMA Settings
- GPIO Settings
- Parameter Settings
- User Constants

Search (Ctrl+F)

Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Number Of Conversion	2
External Trigger Conversion Source	Regular Conversion launched by software
Rank	
Channel	Channel 0
Sampling Time	239.5 Cycles
Rank	
Channel	Channel 1
Sampling Time	239.5 Cycles

Chọn 2 kênh In1 và In2. Bật chế độ scan và cont. Nhớ chọn Num of Con là 2 và chỉnh các channel về đúng 0 và 1

ADC1 Mode and Configuration

Mode

IN0
IN1
IN2
IN3

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low

Add Delete

DMA Request Settings

Mode: Circular

Increment Address:

Peripheral:

Memory:

Data Width: Half Word

Half Word

Add thêm DMA từ ADC về Memory, chế độ Circular và Width là Half Word

Thiết lập USB

USB Mode and Configuration

Mode

Device (FS)

Configuration

Reset Configuration

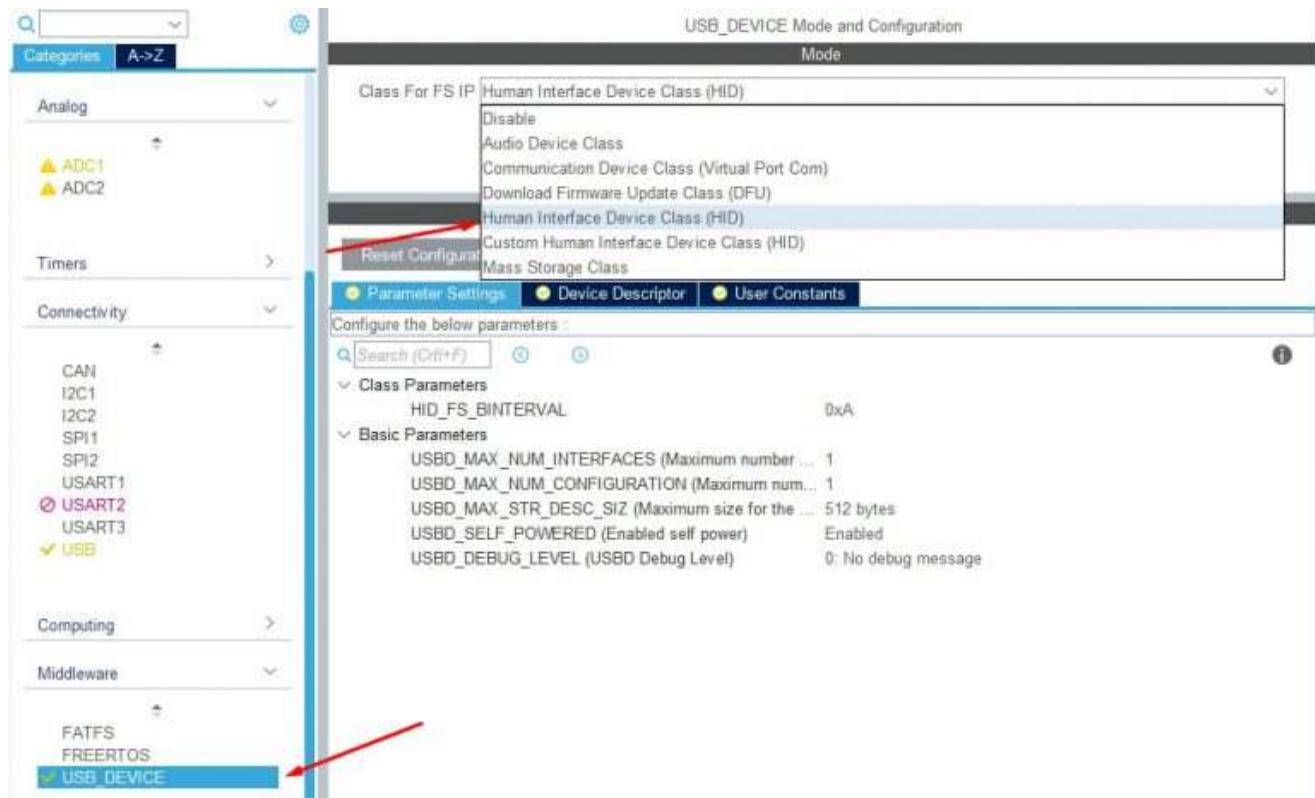
Parameter Settings User Constants NVIC Settings GPIO Settings

Configure the below parameters :

Search (Ctrl+F)	Basic Parameters	Speed: Full Speed 12MBit/s
	Power Parameters	Low Power: Disabled
		Link Power Management: Disabled
		Battery Charging: Disabled

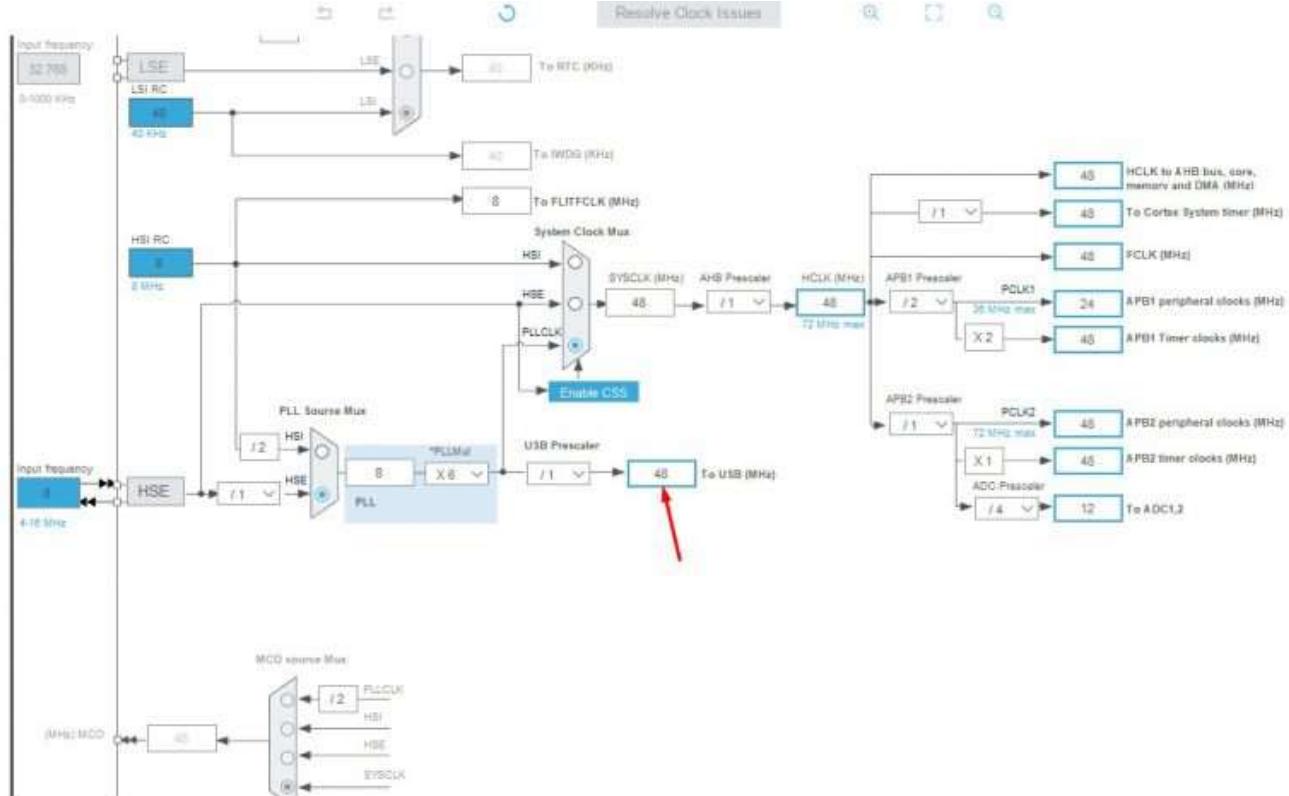
USB

Chọn Usb Device (FS), USB chế độ full speed

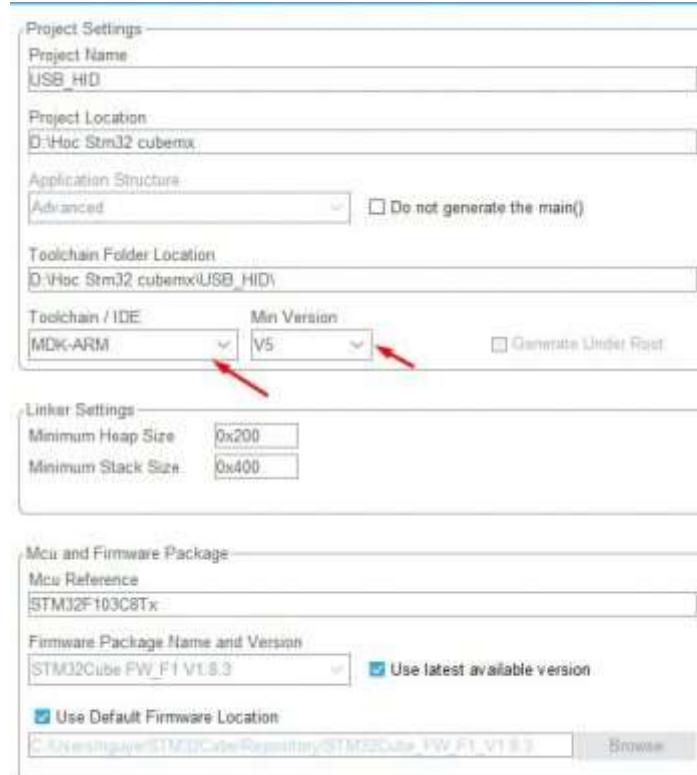


Chọn HID trong Class của Middleware

Thiết lập Clock cho USB là **48Mhz** (Bắt buộc)



Đặt tên bài học rồi Gen code như tất cả các bài trước



Lập trình STM32 USB HID trên Keil C

Open bằng Keil C. Đầu tiên chúng ta sẽ thêm thư viện Hid vào main để dễ dàng thao tác bằng lệnh **#include "usbd_hid.h"**

```

18  /*
19  /* USER CODE END Header */
20  /* Includes -----
21  #include "main.h"
22  #include "usb_device.h"
23
24  /* Private includes -----
25  /* USER CODE BEGIN Includes */
26  #include "usbd_hid.h"
27  /* USER CODE END Includes */
28
29  /* Private typedef -----
30  /* USER CODE BEGIN PTD */
31
32  /* USER CODE END PTD */
33

```

Sau đó extern biến chứa giá trị cài đặt của USB vào main

```

-- /* Private function prototypes -----
52  void SystemClock_Config(void);
53  static void MX_GPIO_Init(void);
54  static void MX_DMA_Init(void);
55  static void MX_ADC1_Init(void);
56
57  /* USER CODE BEGIN PFP */
58  extern USBD_HandleTypeDef hUsbDeviceFS;
59  /* USER CODE END PFP */
60
61  /* Private user code -----

```

Tìm hiểu HID_MOUSE_ReportDesc

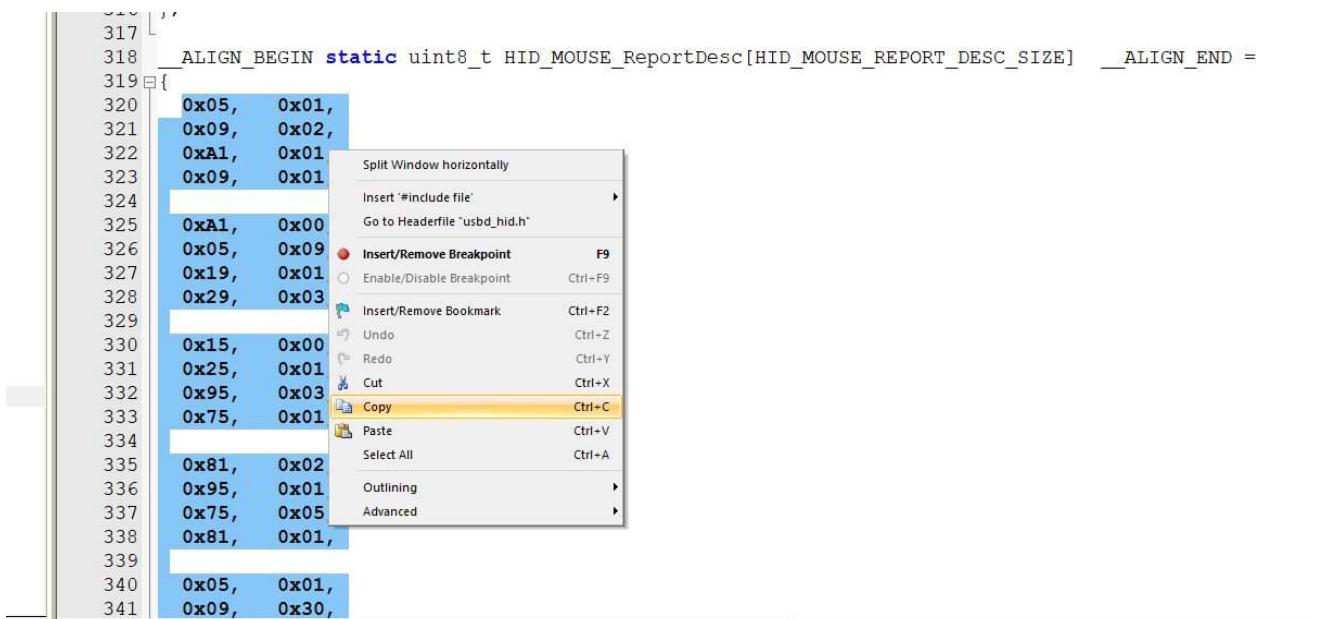
Khi chọn chế độ USB HID, CubeMX đã mặc định chọn thiết bị HID đó là chuột máy tính. Chúng ta cùng phân tích Report Descriptor mà CubeMx đã cho sẵn nhé. Các bạn vào USBD_HID.c và tìm dòng code như sau:

```

307     USB_DESC_TYPE_DEVICE_QUALIFIER,
308     0x00,
309     0x02,
310     0x00,
311     0x00,
312     0x00,
313     0x40,
314     0x01,
315     0x00,
316   };
317
318   _ALIGN_BEGIN static uint8_t HID_MOUSE_ReportDesc[HID_MOUSE_REPORT_DESC_SIZE] _ALIGN_END =
319   {
320     0x05, 0x01,
321     0x09, 0x02,
322     0xA1, 0x01,
323     0x09, 0x01
324
325     0xA1, 0x00
326     0x05, 0x09
327     0x19, 0x01
328     0x29, 0x03
329
330     0x15, 0x00
331     0x25, 0x01
332     0x95, 0x03
333     0x75, 0x01
334
335     0x81, 0x02
336     0x95, 0x01
337     0x75, 0x05
338     0x81, 0x01,
339
340     0x05, 0x01,
341     0x09, 0x30,

```

Copy đoạn dữ liệu và paste vào công cụ phân tích mình vừa mới nêu trên:
<http://eleccelerator.com/usbdescreqparser/>



Kết quả phân tích sẽ như sau:

```

0x05, 0x01,          // Usage Page (Generic Desktop Ctrls)
0x09, 0x02,          // Usage (Mouse)
0xA1, 0x01,          // Collection (Application)
0x09, 0x01,          //   Usage (Pointer)
0xA1, 0x00,          //   Collection (Physical)
0x05, 0x09,          //     Usage Page (Button)
0x19, 0x01,          //     Usage Minimum (0x01)

```

```

0x29, 0x03,          // Usage Maximum (0x03)
0x15, 0x00,          // Logical Minimum (0)
0x25, 0x01,          // Logical Maximum (1)
0x95, 0x03,          // Report Count (3)
0x75, 0x01,          // Report Size (1)
0x81, 0x02,          // Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No
0x95, 0x01,          // Report Count (1)
0x75, 0x05,          // Report Size (5)
0x81, 0x01,          // Input (Const,Array,Abs,No Wrap,Linear,Preferred State,
0x05, 0x01,          // Usage Page (Generic Desktop Ctrl)
0x09, 0x30,          // Usage (X)
0x09, 0x31,          // Usage (Y)
0x09, 0x38,          // Usage (Wheel)
0x15, 0x81,          // Logical Minimum (-127)
0x25, 0x7F,          // Logical Maximum (127)
0x75, 0x08,          // Report Size (8)
0x95, 0x03,          // Report Count (3)
0x81, 0x06,          // Input (Data,Var,Rel,No Wrap,Linear,Preferred State,No
0xC0,                // End Collection
0x09, 0x3C,          // Usage (Motion Wakeup)
0x05, 0xFF,          // Usage Page (Reserved 0xFF)
0x09, 0x01,          // Usage (0x01)
0x15, 0x00,          // Logical Minimum (0)
0x25, 0x01,          // Logical Maximum (1)
0x75, 0x01,          // Report Size (1)
0x95, 0x02,          // Report Count (2)
0xB1, 0x22,          // Feature (Data,Var,Abs,No Wrap,Linear,No Preferred State,
0x75, 0x06,          // Report Size (6)
0x95, 0x01,          // Report Count (1)
0xB1, 0x01,          // Feature (Const,Array,Abs,No Wrap,Linear,Preferred State,
0xC0,                // End Collection

// 74 bytes

```



Ta thấy rằng trình tự các byte gửi như sau:

1 byte nút nhấn -> 3 byte X, Y, Whell -> 2byte Wakeup -> 1 byte kết thúc

Số byte này bạn chỉ cần đếm trong các đoạn **Report Count** theo thứ tự từ trên xuống dưới. Còn ý nghĩa của các từ mình đã giải thích bên trên rồi nhé.

Vậy nên chúng ta sẽ tạo 1 mảng chứa giá trị của các byte gửi lên với 1byte kết thúc là mặc định nên ko cần thêm vào. Mình tạo mảng mouse_report[5], và mảng lưu 2 giá trị ADC1 truyền qua DMA

```

-- , ----- , -----
33
34 /* Private define ----- */
35 /* USER CODE BEGIN PD */
36 /* USER CODE END PD */
37
38 /* Private macro ----- */
39 /* USER CODE BEGIN PM */
40 int8_t mouse_report[5];
41 uint16_t ADC_val[2];
42 /* USER CODE END PM */
43
44 /* Private variables ----- */
45 ADC_HandleTypeDef hadc1;
46 DMA_HandleTypeDef hdma_adc1;
47
48 /* USER CODE BEGIN PV */
49
50 /* USER CODE END PV */
51
52 /* Private function prototypes ----- */

```

Trước Main ta sẽ lập trình như sau

Tạo một hàm tên là map để chuyển đổi giá trị ADC từ 0 – 4096 thành -127,127

```

60
61 /* Private user code ----- */
62 /* USER CODE BEGIN 0 */
63 long map(long x, long in_min, long in_max, long out_min, long out_max) // chuyen doi gia tri min max
64 {
65     long value;
66     value = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
67     return value;
68 }
69 void Mouse_GetAction(void) // doc gia tri nut nhan va toa do
70 {
71     static uint8_t button = 0;
72     int8_t XPos,YPos = 0;
73     if(HAL_GPIO_ReadPin(Left_button_GPIO_Port,Left_button_Pin) == 0)
74     {
75         button = 1;
76     }
77     else
78     {
79         button = 0;
80     }
81     XPos = map(ADC_val[0],0,4096,-127,127); // chuyen doi
82     YPos = map(ADC_val[1],0,4096,-127,127);
83
84     mouse_report[0] = button; // cac nut nhan
85     mouse_report[1] = XPos; // toa do
86     mouse_report[2] = YPos;
87     mouse_report[3] = 0; //wheel
88     mouse_report[4] = 0; //wakeups
89
90 }

```

Tạo hàm đọc giá trị nút nhấn, thực hiện chuyển đổi và ghi vào mảng mouse_report

Trước While cho bắt đầu chuyển đổi ADC DMA, Trong While ta đọc giá trị của mouse và gửi qua cổng USB HID

```

119  /* Initialize all configured peripherals */
120  MX_GPIO_Init();
121  MX_DMA_Init();
122  MX_ADC1_Init();
123  MX_USB_DEVICE_Init();
124  /* USER CODE BEGIN 2 */
125  HAL_ADC_Start_DMA(&hadc1, (uint32_t *)ADC_val, 2);
126  /* USER CODE END 2 */
127
128  /* Infinite loop */
129  /* USER CODE BEGIN WHILE */
130  while (1)
131  {
132      /* USER CODE END WHILE */
133
134      /* USER CODE BEGIN 3 */
135      Mouse_GetAction(); ←
136      USBD_HID_SendReport(&hUsbDeviceFS, (uint8_t *)mouse_report, 5); ←
137  }
138  /* USER CODE END 3 */
139 }
140
141 /**
142 * @brief System Clock Configuration
143 * @retval None
144 */

```

Kết nối và chạy thử USB HID

Nhấn F7 để Build và F8 để nạp Code vào Kit.

Kết nối Joystick vào Kit theo hướng dẫn:

5V → 3.3V

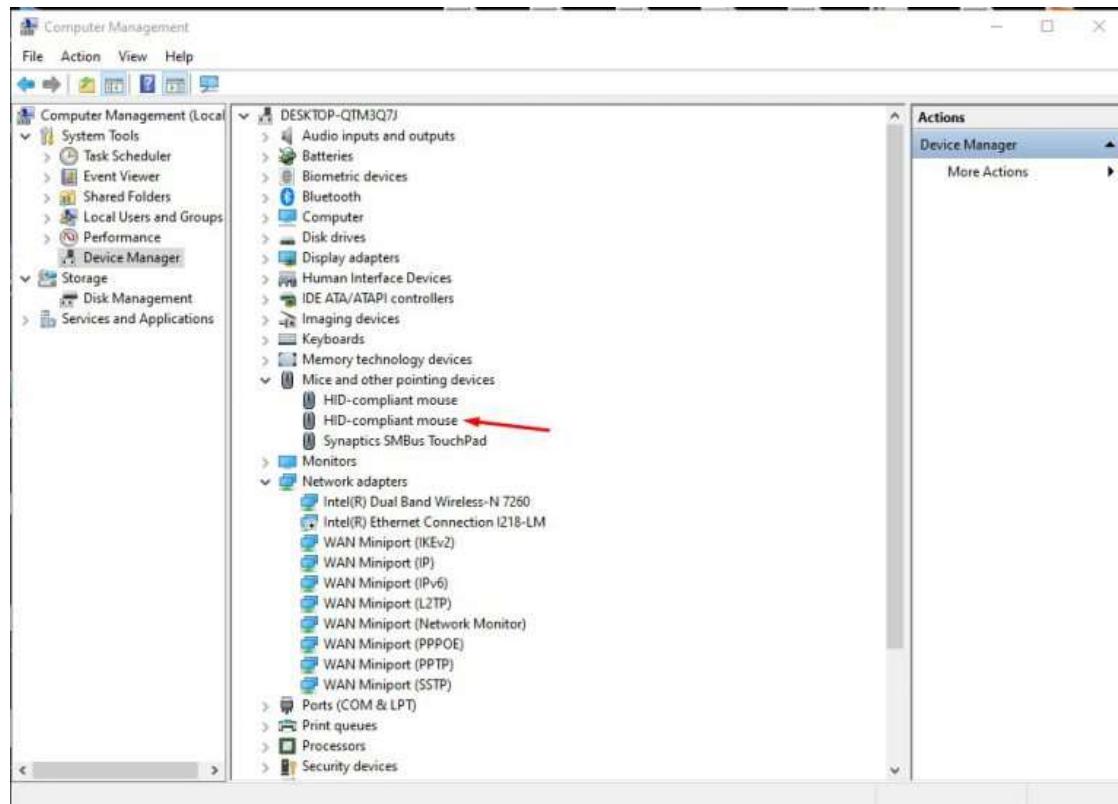
GND → GND

VRX → PA0

VRY → PA1

SW → PA2

Cắm dây Micro USB vào mạch và cắm đầu còn lại vào máy tính. Mở Manager ra xem bạn sẽ thấy thêm 1 thiết bị Mouse



Mở Paint Sử dụng Joystick điều khiển thử.



Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển

Chia sẻ

Facebook Watch

Ta thấy rằng chuột chạy quá nhanh, không thể kiểm soát và khi không di chuyển chuột cũng vẫn tự chạy. Lý do là điểm cân bằng của Joystick không

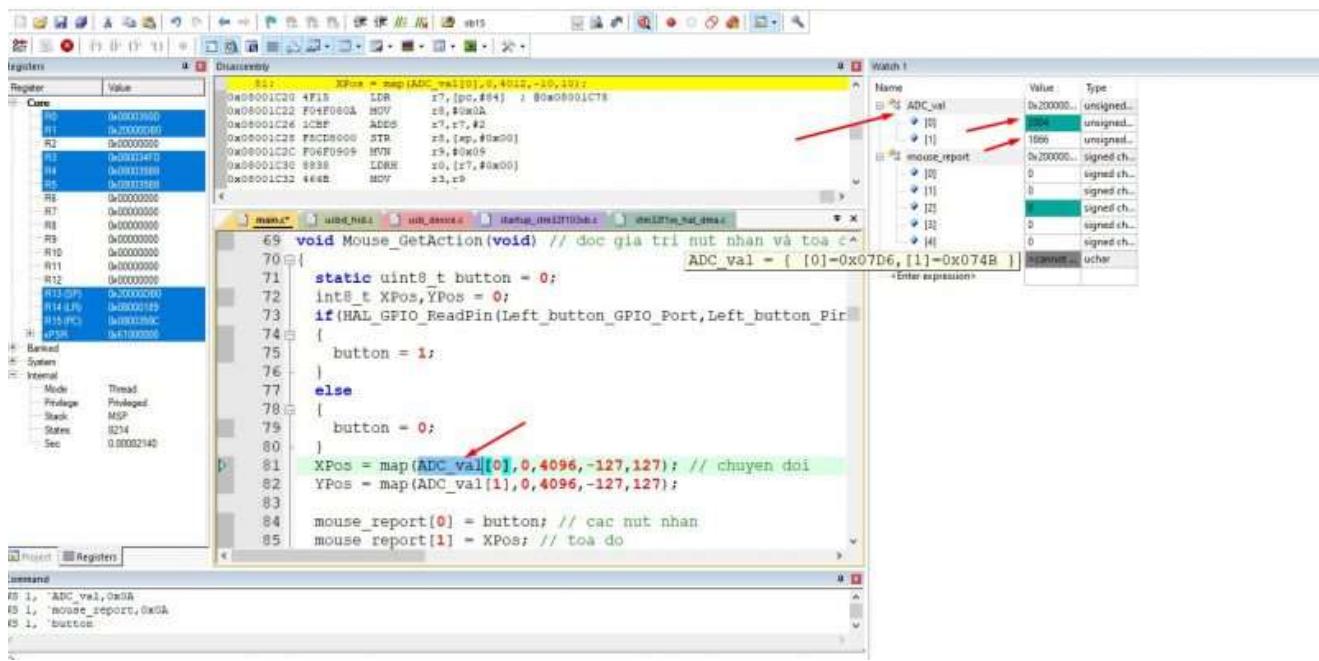
giống lý thuyết đó là giá trị 2048.

Và chuột chạy quá nhanh do mạch sẽ gửi các giá trị từ -127 đến 127 cực nhanh khi giữ Joystick, các tọa độ này khiến chuột di chuyển rất nhanh. Vậy làm ntn để hiệu chỉnh hai thứ đó

Tinh chỉnh code (Calibration)

Rút dây cắm HID từ chuột ra, chạy chế độ debug.

Trong debug ta add giá trị ADC_Val vào Watch 1 và nhấn chạy chương trình. Chúng ta thấy rằng: giá trị cân bằng khác nhau dẫn tới chuột luôn luân di chuyển.



Sửa lại trong hàm Get Action, như sau:

```

74     {
75         button = 1;
76     }
77     else
78     {
79         button = 0;           2004x2
80     }
81     XPos = map(ADC_val[0], 0, 4008, -10, 10); // chuyen doi
82     YPos = map(ADC_val[1], 0, 3932, -10, 10);   1966x2
83     |
84     mouse_report[0] = button; // cac nut nhan
85     mouse_report[1] = XPos; // toa do
86     mouse_report[2] = YPos;
87     mouse_report[3] = 0;    //wheel
88     mouse_report[4] = 0;    //wakeup
89
90 }
91 /* USER CODE END 0 */
92

```

Sửa -127, 127 thành -10 và 10 sẽ giúp chuột di chuyển chậm hơn.

Nạp và chạy thử kết quả



Khuê Nguyễn Creator - Học Lập Trình Vi Điều Khiển

Chia sẻ

Facebook Watch

Kết

STM32 USB HID được sử dụng rất rộng rãi khi muốn giao tiếp với máy tính, điện thoại, game pad một cách đơn giản nhất. Về cơ bản tất cả các device sử dụng USB HID đều làm việc giống nhau, sự khác nhau của chúng là cấu trúc gói tin truyền và đích đến sẽ được định nghĩa hết trong Report Descriptor.

Nếu thấy bài viết này có ích hãy like và chia sẻ cho người khác, đừng quên vào nhóm [Anh Em Nghiện Lập Trình](#) để giao lưu nhé các bạn

4.3/5 - (6 bình chọn)

Related Posts:

1. [Lập trình STM32 đọc nhiệt độ với DS18b20 giao tiếp OneWire](#)
2. [Hướng dẫn download và sử dụng tài liệu Lập trình STM32](#)
3. [Bài 18: Lập trình STM32 USB HID Keyboard bàn phím máy tính](#)
4. [Bài 16: Lập trình STM32 USB CDC truyền nhận dữ liệu qua cổng COM ảo](#)
5. [Bài 9: Lập trình STM32 ADC nhiều kênh với DMA](#)
6. [Bài 2: Tổng quan về KIT STM32F103C8T6 Blue Pill](#)



KHUÊ NGUYỄN

Chỉ là người đam mê điện tử và lập trình. Làm được gì thì viết cho anh em xem thôi. :D

2 THOUGHTS ON “BÀI 17: LẬP TRÌNH STM32 USB HID CHUỘT MÁY TÍNH”



Hải says:

A ơi e có tải phần code của a trên github ý nạp vào stm32 thì lúc chạy nó chỉ chạy 1 đường chéo chứ ko dùng analog được q a giải đáp giùm e q

11/05/2021 AT 9:54 CHIỀU

TRẢ LỜI



Khuê Nguyễn says:

em phải calib (cân bằng) lại nhé. A hướng dẫn rồi đó

12/05/2021 AT 8:18 SÁNG

TRẢ LỜI

Trả lời

Email của bạn sẽ không được hiển thị công khai. Các trường bắt buộc được đánh dấu *

Bình luận *

Tên *

Email *

Trang web

PHẢN HỒI

Fanpage

 Khuê Nguyễn Creator - Họ...
2.754 lượt thích

Đã thích **Chia sẻ**

**Khuê Nguyễn Creator - Học
Lập Trình Vi Điều Khiển**
khoảng một tháng trước

Lý do thời gian gần đây mình không viết bài và làm thêm gì cả là đây 😊)
Chính thức ra mắt sản phẩm định vị thông minh vTag.
Đây là một sản phẩm định vị đa năng với 3 công nghệ định vị WIFI, GPS, LBS kết hợp với sóng NB-IOT dành riêng cho các sản phẩm IOT.
Chỉ với 990.000đ chúng ta đã có thể có sản phẩm để:
- Định vị trẻ em, con cái... [Xem thêm](#)



Bài viết khác

Lập trình 8051 - AT89S52



**Bài 1: Tổng quan về 8051
và chip AT89S51 - 52**



Khuê Nguyễn Creator



PROTEUS

Tổng quan về 8051

8051 là một dòng chip nhập môn cho lập trình viên nhúng, chúng được sử...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator

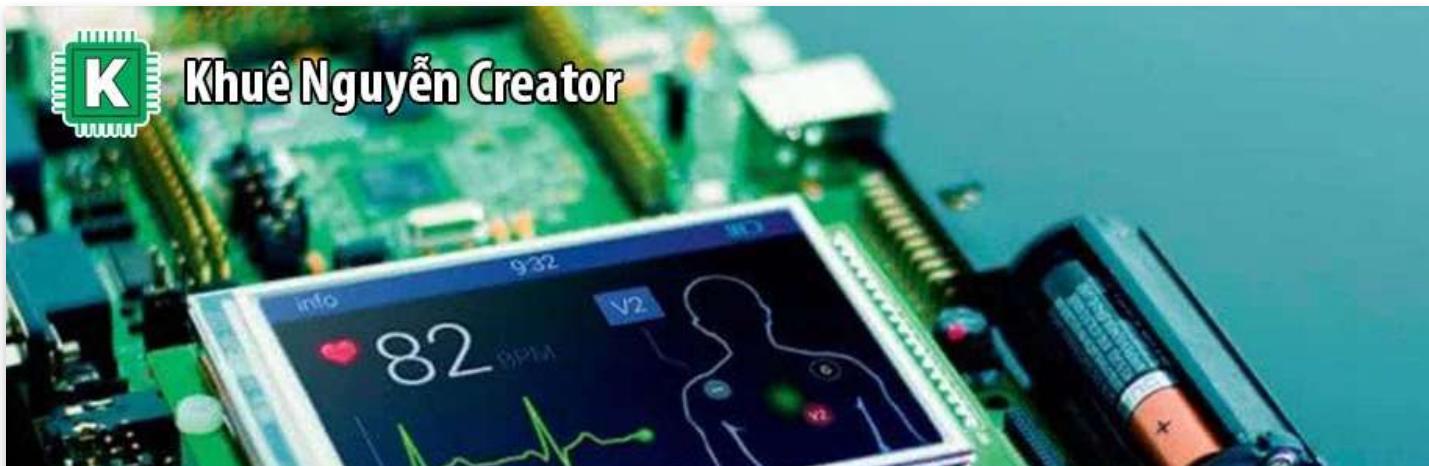


Lập trình STM32 HID Host giao tiếp với chuột và bàn phím

Lập trình STM32 USB HID Host giao tiếp với chuột và bàn phím máy tính

Trong bài này chúng ta sẽ cùng học STM32 HID Host, biến STM32 giống như...

[ĐỌC THÊM](#)





Lộ trình học lập trình nhúng từ A tới Z

Lộ trình học lập trình nhúng từ A tới Z

Lập trình nhúng là một ngành có cơ hội nhưng cũng đòi hỏi nhiều kiến...

3 COMMENTS

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Lập trình STM32F407 SDIO đọc dữ liệu thẻ nhớ

Lập trình STM32 SDIO đọc ghi dữ liệu vào thẻ nhớ SD card

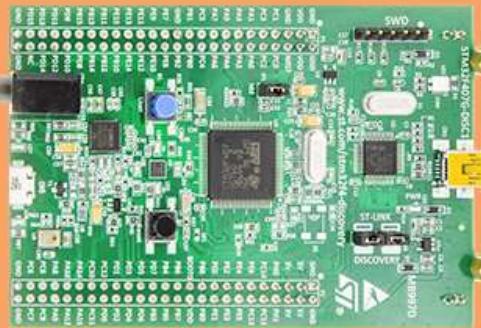
Trong bài này chúng ta cùng học cách lập trình STM32 SDIO, một chuẩn giao...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Lập trình STM32F407 DAC chuyển đổi số sang tương tự

Lập trình STM32 DAC tạo sóng hình Sin trên KIT STM32F407 Discovery

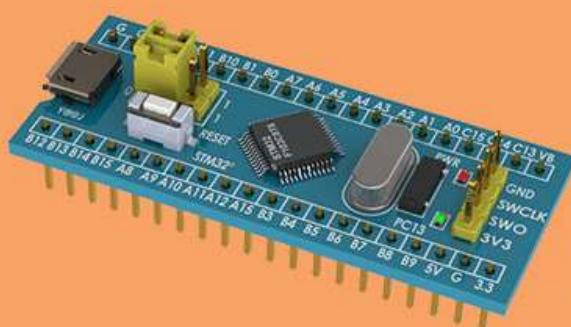
Trong bài này chúng ta sẽ cùng nhau tìm hiểu STM32 DAC với KIT STM32F407VE...

[ĐỌC THÊM](#)

Lập trình STM32 và CubeMX



Khuê Nguyễn Creator



Sử dụng hàm printf để in Log khi Debug trên STM32

Hướng dẫn sử dụng printf với STM32 Uart để in Log trên Keil C

Trong bài này chúng ta sẽ học cách retarget hàm printf của thư viện stdio...

3 COMMENTS

[ĐỌC THÊM](#)

ESP32 và Platform IO



Khuê Nguyễn Creator



Bài 9 WIFI: Lập trình ESP32 OTA nạp firmware trên Internet

Lập trình ESP32 FOTA nạp firmware qua mạng Internet với OTA Drive

Trong bài này chúng ta sẽ học cách sử dụng ESP32 FOTA (Firmware Over The...

4 COMMENTS

[ĐỌC THÊM](#)

Lập trình Nuvoton



Khuê Nguyễn Creator

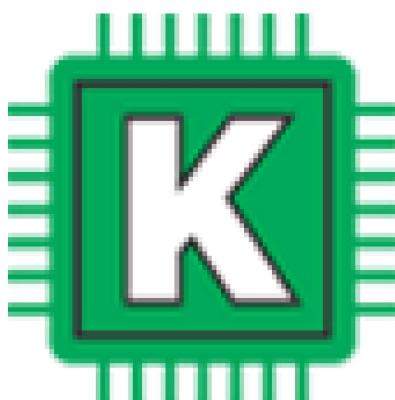


Cài đặt SDC Complier và Code:Blocks IDE

Hướng dẫn cài đặt SDCC và Code::Blocks lập trình Nuvoton

Ở bài này chúng ta sẽ cài đặt các công cụ cần thiết cho việc...

[ĐỌC THÊM](#)



KHUÊ NGUYỄN CREATOR
Chia sẻ đam mê

Blog này làm ra để lưu trữ tất cả những kiến thức, những câu chuyện của mình. Đôi khi là những ý tưởng nhất thời, đôi khi là các dự án tự mình làm. Chia sẻ cho người khác cũng là niềm vui của mình, kiến thức mỗi người là khác nhau, không hẳn quá cao siêu nhưng sẽ có lúc hữu dụng.

Liên Kết

Nhóm: Nghiên Lập Trình

Fanpage: Khuê Nguyên Creator

My Shop

Thông Tin

Tác Giả

Chính Sách Bảo Mật



Copyright 2022 © Khuê Nguyễn