

**SANTA CLARA UNIVERSITY**  
**Electrical and Computer Engineering Department**

**Real-Time Embedded Systems - ECEN 121**  
**Lab 7**

*Dylan Thornburg & Kai Hoshide*

**USB Mouse**

*Andrew Wolfe*

**Pre Lab:**

Read the [STM32Cube USB host library manual](#).

**1) When using a basic mouse as an HID class device, how many device endpoints are used and what are they called?**

There are two endpoints; the control endpoint and the interrupt endpoint.

**2) What is the format of the data structure used to report mouse motion and button information?**

The data structure is a struct that has an 8-bit int x,y, and array [3] for the buttons.

**3) What function is used to get this data about mouse motion and button information?**

USBH\_HID\_GetMouseInfo

**4) What source code file includes the HID mouse subclass handler?**

usbh\_hid\_mouse.c

**5) How many total USB pipes are supported by this library without modification?**

15

**6) Does the existing USB library support USB hubs?**

NAH

**Lab Procedure:**

1. Connect the LED strip the same way you had it in Lab 3.
2. Run one of your old LED strip programs to make sure the LED strip works.
3. Connect a wired USB mouse (The Logitech B100) through the adapter to the USB OTG connector
4. Download the project Mouse Blue Dot 60LED from the Lab Handouts directory. Unzip it into the same directory as your other CubeMX-based projects. Open the project in CubeMX and generate code. Then open it in Keil, build it, and download it to your board. This should work.

5. Reset the board. The green light near the USB port should go on to indicate USB power. After a few seconds, the regular red LED should come on indicating a mouse is detected. Also, an LED in the middle of the light strip should turn on blue.
6. Move the mouse left and right. The blue LED should move.

### Part 1:

By reading the code and running it with the debugger, you need to answer the following questions. For each question, take a screen shot showing the code and/or data that contains your answer, highlight the relevant information, and include it in your report.

- 1) What value in the code needs to be changed to alter the amount of LED movement for a given amount of mouse movement? Identify the value, then adjust it to work well with your mouse.

It is SPOTFRAC. Several functions use it to determine location of the spot and changing it to one thousand made the spot move a lot less. We will choose 500 as we believe this is a sensible sensitivity.

```
#define SPOTFRAC 100    // fractional representation of the spot location
#define SPOTFRAC 1000  // fractional representation of the spot location
```

- 2) What is the Vendor ID and the Product ID for your mouse? Also go to [devicehunt.com](http://devicehunt.com) and see what it knows about your Vendor ID and the Product ID.

Using the debugger and dev\_desc struct, we can see that the Vendor ID is 0x046D=1133 and the Product ID is 0xC077= 49271. Also devicehunt.com knew we had a mouse and which company made it based on the info we gave it. It did not have drivers for us though.

```
dev_desc->idVendor = LE16(buf + 8);
dev_desc->idProduct = LE16(buf + 10);
```

dev_desc->idVendor	0x046D	ushort
dev_desc->idProduct	0xC077	ushort

DEVICE

H U N T

[About](#)
[Why](#)
[PCI Vendors](#)
[USB Vendors](#)
[Forum](#)
[Donate](#)
[Contact](#)

[Login](#)
[Register](#)

Type

USB

▼

Vendor ID

046D

Device ID

C077

Q

Device Details

Mouse

Type	Information
ID	C077

Vendor Details

Logitech, Inc.

Type	Information
ID	046D

Drivers

🚗

Sorry, no drivers found for this device.

←


Ads by Google

Send feedback

Why this ad? ▶

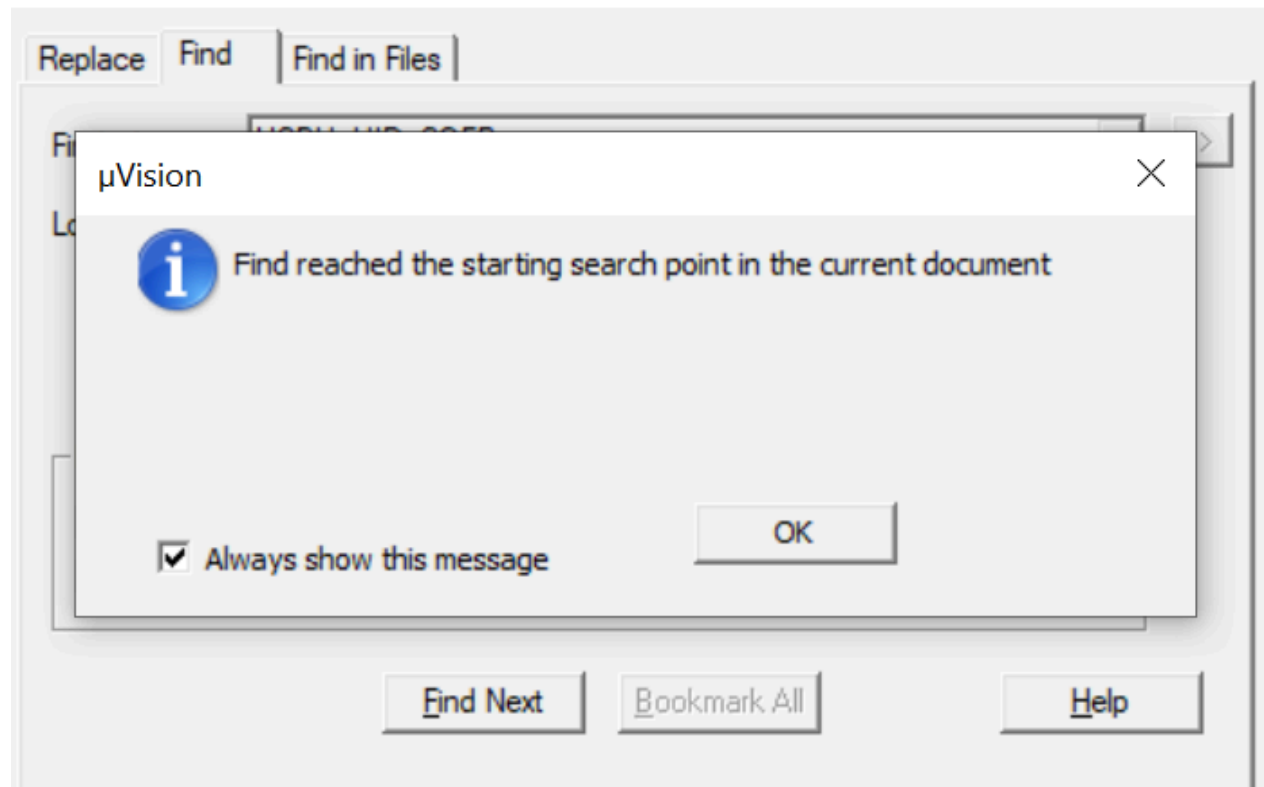
### 3) How many USB pipes are open to your mouse? How do you know?

There are three used but one is a default pipe. There appears to be 16 declared though. It may be 15 though.

	<b>Pipes</b>	0x200004BC	uint[16]
	[0]	0x00008000	uint
	[1]	0x00008080	uint
	[2]	0x00008081	uint
#define USBH_MAX_PIPES_NBR			15U

### 4) Does USBH\_HID\_SOFProcess() ever get called in this program? How do you know?

No it does not. We can see this if we scan the project to find every instance of the function. It was never used.



##### 5) What does USBH\_UsrLog() do?

USBH\_UsrLog() logs what's happening with the USB using strings.

```

/*Decode Bootclass Protocol: Mouse or Keyboard*/
if (phost->device.CfgDesc.Itf_Desc[interface].bInterfaceProtocol == HID_KEYBRD_BOOT_CODE)
{
    USBH_UsrLog("KeyBoard device found!");
    HID_Handle->Init = USBH_HID_KeybdInit;
}
else if (phost->device.CfgDesc.Itf_Desc[interface].bInterfaceProtocol == HID_MOUSE_BOOT_CODE)
{
    USBH_UsrLog("Mouse device found!");
    HID_Handle->Init = USBH_HID_MouseInit;
}
else
{
    USBH_UsrLog("Protocol not supported.");
    return USBH_FAIL;
}

```

```

#define USBH_UsrLog(...) do { \
    printf(__VA_ARGS__); \
    printf("\n"); \
} while (0)
#else
#define USBH_UsrLog(...) do {} while (0)
#endif

#if (USBH_DEBUG_LEVEL > 1U)

```

#### 6) What does fixData() do?

Fix data allows x to be negative by comparing the x location value to the most significant bit in a byte. It then ors the x coordinate with leading ones. (sign extension)

```

int fixData(uint8_t d) {
    if ((d & 0x80) != 0)
        return 0xffffffff00 | (int) d;
    else
        return (int) d;
}

/* USER CODE END 0 */

```

#### Part 2:

Copy the project. Modify the copied version so that:

- When you press the left button, the moving LED turns red and stays red until you hit another button.
- When you press the right button, the moving LED turns purple and stays purple until you hit another button.
- When you press the center button (under the scroll wheel), the moving LED turns green and stays green until you hit another button.
- If you hit multiple buttons, something sensible should happen.

**Demo** and include your code modifications in the report.

#### CODE (MAIN):

```

int main(void)
{
    /* USER CODE BEGIN 1 */

```

```

/* USER CODE END 1 */

/* MCU Configuration-----*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN Sysinit */

/* USER CODE END Sysinit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USB_HOST_Init();
/* USER CODE BEGIN 2 */
    //DECLARE PRIVATE VARIABLES FOR USE
    int flags[3] = {0,0,0};
    int array[4] = {KBLUE, KRED, KPURPLE, KGREEN};
    int index =0;
    int toggle[3] = {0,0,0};
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    setDot(colors, NUM_LEDS, spotLocation, array[index]);
    send_array(colors);
    /* USER CODE END WHILE */

    MX_USB_HOST_Process();

/* USER CODE BEGIN 3 */
    if (hUsbHostFS.gState == HOST_CLASS) {

#ifdef KBMOUSECOMBO
        hUsbHostFS.device.current_interface = 1;
#endif

        devType = USBH_HID_GetDeviceType(&hUsbHostFS);
        if (devType == HID_MOUSE) {
            HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_SET);
        }
        else {
            HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_RESET);
        }
        if (devType == HID_KEYBOARD) {
            HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_SET);
        }
        else {
            HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_RESET);
        }
    }
}

```

```

}
if (devType == HID_MOUSE) {
    mouseInfo = USBH_HID_GetMouseInfo(&hUsbHostFS);
    if (mouseInfo != NULL) {
        spotLocation = spotUpdate(spotLocation, fixData(mouseInfo->x));
        //Code goes beyond specifications and allows toggle back to blue
        //if you click the same button twice in a row. If you didn't want this,
        //just delete toggle and all its instances.

        //handles left click to red
        if(mouseInfo->buttons[0]==1)
        {
            toggle[1] = 0;
            toggle[2] = 0;
            if (flags[0] ==0)
            {
                flags[0]=1;
            }
        }
        if(mouseInfo->buttons[0]==0 && flags[0] ==1)
        {
            index = 1;
            flags[0] = 0;
            toggle[0]++;
        }

        //handles right click to purple
        if(mouseInfo->buttons[1]==1)
        {
            toggle[0] = 0;
            toggle[2] = 0;
            if (flags[1] ==0)
            {
                flags[1]=1;
            }
        }
        if(mouseInfo->buttons[1]==0 && flags[1] ==1)
        {
            index = 2;
            flags[1] = 0;
            toggle[1]++;
        }

        //handles middle click to green
        if(mouseInfo->buttons[2]==1)
        {
            toggle[0] = 0;
            toggle[1] = 0;
            if (flags[2] ==0)
            {
                flags[2]=1;
            }
        }
        if(mouseInfo->buttons[2]==0 && flags[2] ==1)
        {

```

```

        index = 3;
        flags[2] = 0;
        toggle[2]++;
    }

    //handles the toggle back to blue
    if(toggle[0] >=2 || toggle[1] >= 2 || toggle[2] >=2)
    {
        index = 0;
        toggle[0] = 0;
        toggle[1] = 0;
        toggle[2] = 0;
    }
}

}

else if (hUsbHostFS.gState == HOST_IDLE) {
    HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_RESET);
}

}
/* USER CODE END 3 */
}

```

### Part 3:

Copy the original project. Modify the copied version so that:

- You have a list of 8 colors, red, yellow, purple, green, blue, indigo, white, and orange – in that order.
- The moving LED starts blue
- When you click the left button, the moving LED switches to the next color in the list. After orange, start again with red. One color change per click.

**Demo** and include your code modifications in the report.

### CODE (MAIN):

```

int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

```



```

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN Sysinit */

/* USER CODE END Sysinit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USB_HOST_Init();
/* USER CODE BEGIN 2 */
    int flag = 0;
    int array[8] = {KRED, KYELLOW, KPURPLE, KGREEN, KBLUE, KINDIGO, KWHITE, KORANGE};
    int index=4;
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    setDot(colors, NUM_LEDS, spotLocation, array[index]);
    send_array(colors);
    /* USER CODE END WHILE */

    MX_USB_HOST_Process();

/* USER CODE BEGIN 3 */
    if (hUsbHostFS.gState == HOST_CLASS) {

#ifdef KBMOUSECOMBO
        hUsbHostFS.device.current_interface = 1;
#endif

        devType = USBH_HID_GetDeviceType(&hUsbHostFS);
        if (devType == HID_MOUSE) {
            HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_SET);
        }
        else {
            HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_RESET);
        }
        if (devType == HID_KEYBOARD) {
            HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_SET);
        }
        else {
            HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_RESET);
        }
        if (devType == HID_MOUSE) {
            mouseInfo = USBH_HID_GetMouseInfo(&hUsbHostFS);
            if (mouseInfo != NULL) {
                spotLocation = spotUpdate(spotLocation, fixData(mouseInfo->x));
                if(mouseInfo->buttons[0] != 0)
                {
                    if (flag ==0)
                    {
                        flag=1;
                    }
                }
                if(mouseInfo->buttons[0] != 1 && flag ==1)
                {

```

```

        index= (index+1) % 8;
        flag= 0;
    }
}
}
}
else if (hUsbHostFS.gState == HOST_IDLE) {
    HAL_GPIO_WritePin(LD_G_GPIO_Port, LD_G_Pin,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(LD_R_GPIO_Port, LD_R_Pin,GPIO_PIN_RESET);
}
}
/* USER CODE END 3 */
}

```

We had a problem with part three in which our left click wasn't working correctly. We had to put our code in the "if (mouseInfo != NULL)" statement to make it work for the left click, but not the middle or right click. This is because the left click is the start of the array and is affected by movement changes as both data are sent around the same time with how Wolfe constructed his part of the code. Our change stops the code from getting two signals around the same time, and instead the data is updated one after the other when any mouseInfo change is detected. By putting the code in a section that activates if any mouseInfo changes, we solve for the left click issue without affecting right or middle click outputs.