Lab7

2:15 Tues

Henry Fang Jack Landers

Introduction

In this lab, first, we will use a mouse to control the light dot on the LED strip by moving the mouse. Then we will use the right and left buttons to control the color of the cursor (moving LED). Lastly, we will add more color to the available color that responds to mouse clicks.

Procedure

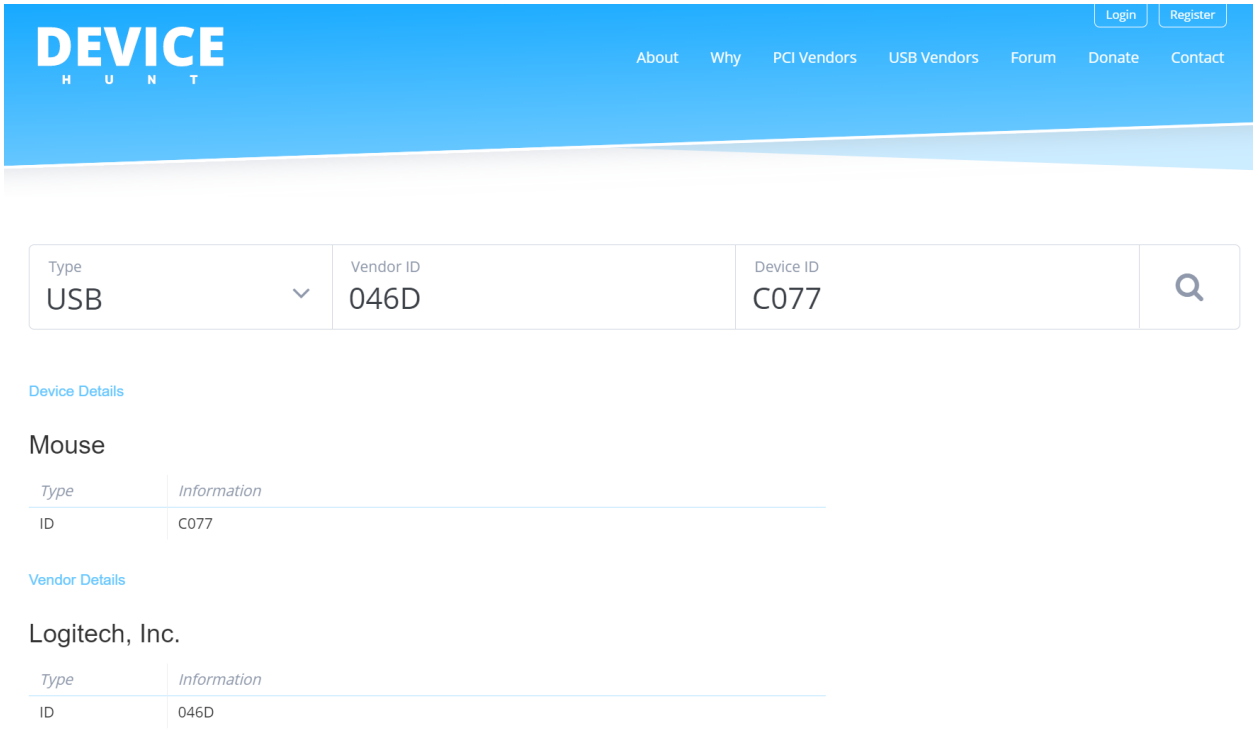
Describe what you did during the lab. The way you wired up the board, what code you wrote (don't paste your actual code here), etc…

Part 1 Import the cursor program.

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.

```
mouseInfo = USBH_HID_GetMouseInfo(&hUsbHostFS);
mouseInfo->x *= 2;
spotLocation = spotUpdate(spotLocation, fixData(mouseInfo->x));
```

2) What is the Vendor ID and the Product ID for your mouse? Also, go to devicehunt.com and see what it knows about your Vendor ID and the Product ID.



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

3 - The other pipes are all filled with zeros

Pipes	0x200004BC	uint[16]
[0]	0x00008000	uint
[1]	0x00008080	uint
[2]	0x00008081	uint
[3]	0x00000000	uint
[4]	0x00000000	uint
[5]	0x00000000	uint
[6]	0x00000000	uint
[7]	0x00000000	uint
[8]	0x00000000	uint
[9]	0x00000000	uint
[10]	0x00000000	uint
[11]	0x00000000	uint
[12]	0x00000000	uint
[13]	0x00000000	uint
[14]	0x00000000	uint
[15]	0x00000000	uint

4) Does USBH_HID_SOFProcess() ever get called in this program? How do you know?
No, in the projects SOFProcess is a function that is declared in the HID class but is never called when we search for instances of it in the project.

5) What does USBH_UsrLog() do?

```
#if (USBH_DEBUG_LEVEL > 0U)
#define  USBH_UsrLog(...)   do { \
                    printf(__VA_ARGS__); \
                    printf("\n"); \
} while (0)
#else
#define USBH_UsrLog(...) do {} while (0)
#endif
```

This looks like a log generator to print input of this function to some logs when the USBH_DEBUG_LEVEL is greater than 0U…
This sends the status of the USB.

6) What does fixData() do?

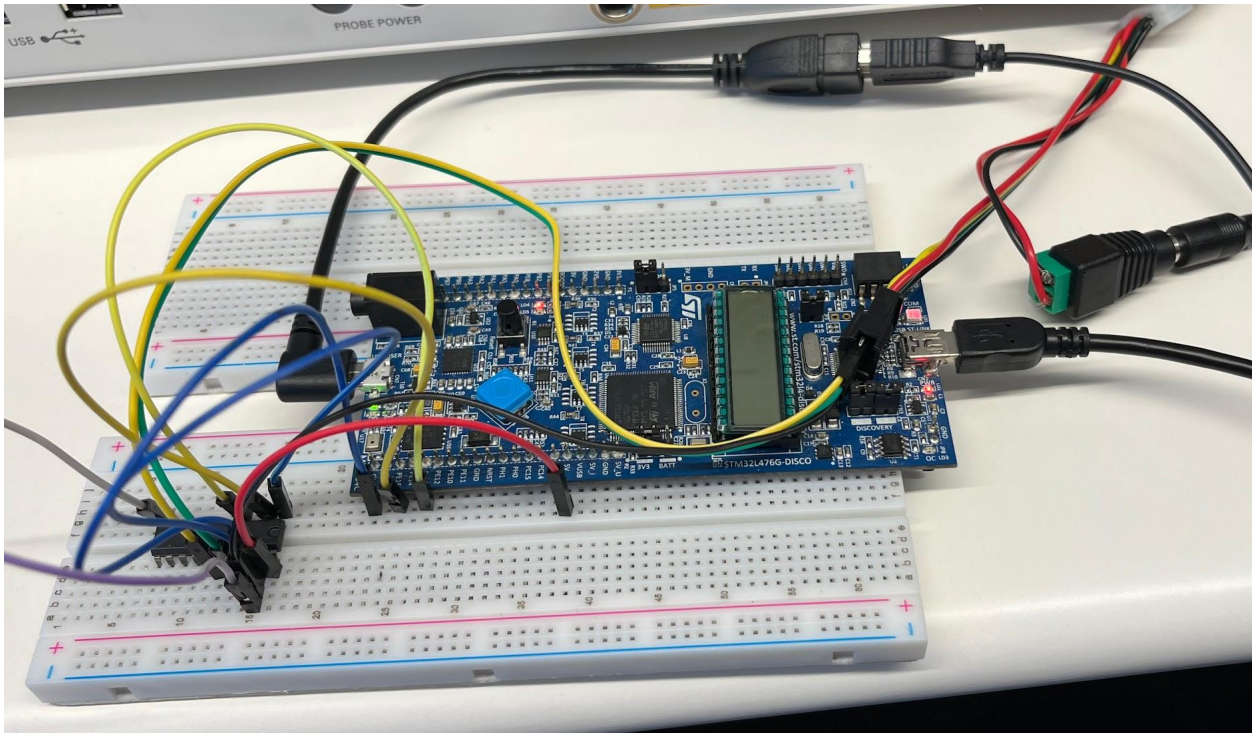
fixData() is doing the sign extension for the input 8-bit unsigned integer d, and outputs the signed integer.

Part 2 Add click response to the cursor program.

Part 3 Add more available colors to click response.

Results

Part 1 Import the cursor program.



Part 2 Add click response to the cursor program.

Part 3 Add more available colors to click response.

Maybe google slide is a better way to do a report? Then we can put video in there.

Conclusion

Short paragraph talking about the takeaways from this lab.
In this lab we learned how to use the usb communication to read HIDs like a mouse. In part 1, we got an example program. We read through the code and adjust some variables to see how the program works after adjustment. We find a variable that respects the sensitivity of the mouse. And we looked up the IDs of this mouse in the device structure. In part 2, we tried to grab the click response from the mouse to change the color of the LED cursor. In part 3, we tried to suppress the update of color from changing positions while holding the right button. To achieve this goal we add a lock to the program, while we hold the right button, the lock is on,

and the update of color will check the lock, then changing will be suppressed if the right mouse button is holded. After we release the right mouse button, the lock is off, then the color can be updated by clicking the right button.

Appendix

Place code and other data here

Part 2

Private variable:

```
unsigned int CursorColor;
```

While(1) loop:

```
while (1)
  {
    setDot(colors, NUM_LEDS, spotLocation, CursorColor);
          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) {
                          if(mouseInfo->buttons[0] != 0){
                               CursorColor=KRED;
                          }
```

```
                        if(mouseInfo->buttons[1] != 0){
                              CursorColor=KPURPLE;
                        }
                        if(mouseInfo->buttons[2] != 0){
                              CursorColor=KGREEN;
                        }
                        spotLocation = spotUpdate(spotLocation,
fixData(mouseInfo->x));
                     }
               }
          }
          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);
          }
  }
```

Part3

Private variable:

```
unsigned int *CursorColor;
unsigned int ColorList[8]={KRED, KYELLOW, KPURPLE, KGREEN, KBLUE,
KPINK, KWHITE, KORANGE};
```

Main:

```
int main(void)
{
  /* USER CODE BEGIN 1 */
     CursorColor=&ColorList[4];
     int flag0 = 0;
     int flag1 = 0;
     int flag2 = 0;
  /* 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 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    setDot(colors, NUM_LEDS, spotLocation, *CursorColor);
          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) {
                            if(mouseInfo->buttons[0] == 0 && flag0 ==
1)
                            {
                                  flag0=0;
                            }
                            if(mouseInfo->buttons[1] == 0 && flag1 ==
1)
                            {
                                  flag1=0;
                            }
                            if(mouseInfo->buttons[2] == 0 && flag2 ==
1)
                            {
                                  flag2=0;
                            }
                            if(mouseInfo->buttons[0] != 0 && flag0 ==
0){
                                  if(CursorColor==&ColorList[7])
                                  {
                                        CursorColor=ColorList;
                                  }
                                  else{CursorColor++;}
                                  flag0 = 1;
                            }
                            if(mouseInfo->buttons[1] != 0 && flag1 ==
0){
                                  if(CursorColor==&ColorList[0])
                                  {
                                        CursorColor=&ColorList[7];
                                  }
                                  else{CursorColor--;}
                                  flag1 = 1;
                            }
                            if(mouseInfo->buttons[2] != 0 && flag2 ==
0){
                                  //CursorColor=KGREEN;
                                  flag2 = 1;
                            }
                            spotLocation = spotUpdate(spotLocation,
fixData(mouseInfo->x));
                      }
                }
```

```
            }
            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 */
}
```