Gigi Patmore and Pranav Chainani
ECEN 121 Lab

Lab 7

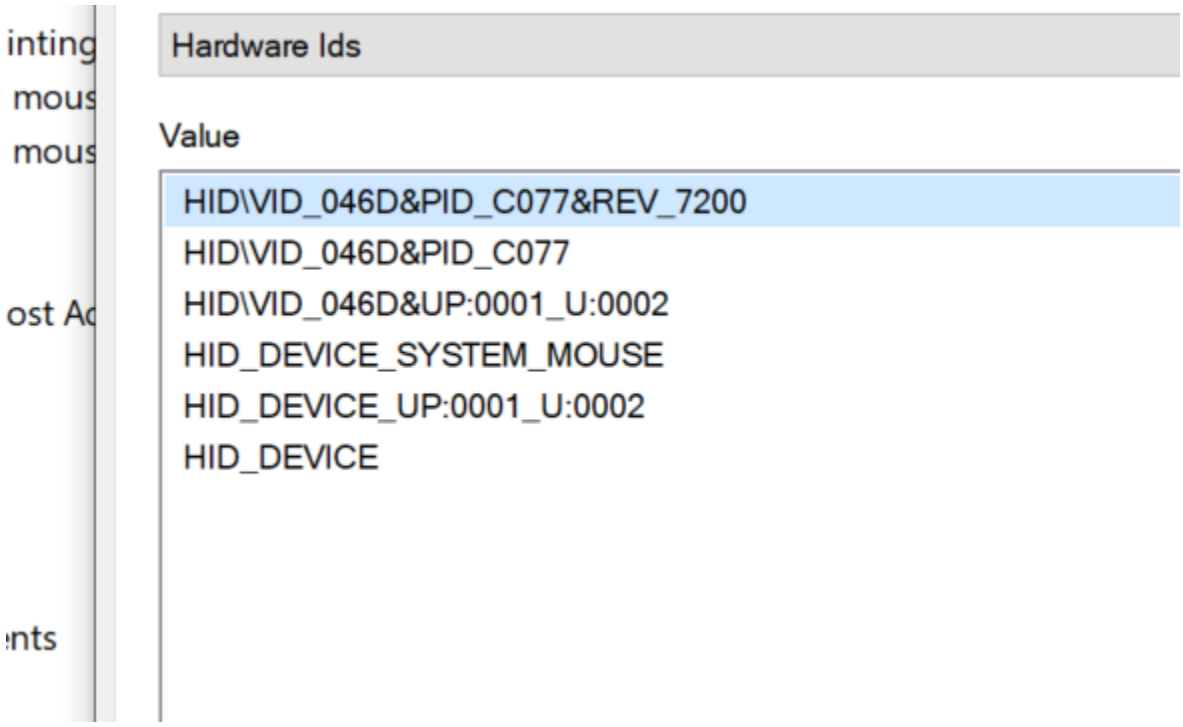
Part 1

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.

Value is spotfrac, smaller is makes spot move faster for the same amount of movement.

```
132
133
134 void setDot(unsigned int *color_array, int color_array_size, int spot, unsigned int color) {
135   for (int i=0; i< color_array_size; i++) {
136     color_array[i] = KBLACK;
137   }
138   color_array[spot/SPOTFRAC] = color;
139 }
140
141 int spotUpdate(int oldspot, int motion) {
142   oldspot += motion;
143   if (oldspot <= 0)
144     return 0;
145   if (oldspot >= ((NUM_LEDS-1) * SPOTFRAC))
146     return ((NUM_LEDS-1) * SPOTFRAC);
147   return oldspot;
148 }
149
```

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.



Type	Vendor ID	Device ID
USB	046D	C077

Device Details

Mouse

Type	*Information*
ID	C077

Vendor Details

Logitech, Inc.

Type	*Information*
ID	046D

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

```
#ifndef USBH_MAX_PIPES_NBR
#define USBH_MAX_PIPES_NBR                                      15U
#endif /* USBH_MAX_PIPES_NBR */
```

4) Does USBH_HID_SOFProcess() ever get called in this program? How do you know?
The function is defined but never gets called.

5) What does USBH_UsrLog() do?
Outputs the action of user. Keeps track (keeps a log) of user actions.

```
  if (interface < phost->device.CfgDesc.bNumInterfaces)
  {
    phost->device.current_interface = interface;
    USBH_UsrLog("Switching to Interface (#%d)", interface);
    USBH_UsrLog("Class    : %xh", phost->device.CfgDesc.Itf_Desc[interface].bInterfaceClass);
    USBH_UsrLog("SubClass : %xh", phost->device.CfgDesc.Itf_Desc[interface].bInterfaceSubClass);
    USBH_UsrLog("Protocol : %xh", phost->device.CfgDesc.Itf_Desc[interface].bInterfaceProtocol);
  }
```

6) What does fixData() do?

Fix data formats the mouse motion report so that it can be used as a motion value to show how much to move spot for the spotUpdate() function.

```
    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));
    }
  }
}
```

```
39  }
40
41  int spotUpdate(int oldspot, int motion) {
42    oldspot += motion;
43    if (oldspot <= 0)
44      return 0;
45    if (oldspot >= ((NUM_LEDS-1) * SPOTFRAC))
46      return ((NUM_LEDS-1) * SPOTFRAC);
47    return oldspot;
48  }
49
50  int fixData(uint8_t d) {
51    if ((d & 0x80) != 0)
52      return 0xffffff00 | (int) d;
53    else
54      return (int) d;
55  }
```

Part 2

```
while (1)
 {
   setDot(colors, NUM_LEDS, spotLocation, led_color);
                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) {
                                        flag =1;
                                        if(flag==1) {
                                                        led_color = KRED;

                                        }
                                }


                                if (mouseInfo->buttons[1] != 0) {
                                        flag = 2;
                                }
                                        if(flag==2) {
                                                led_color = KPURPLE;
                                        }

                                if (mouseInfo->buttons[2] != 0) {
                                        flag =3;
                                        if(flag==3) {
                                                led_color = KGREEN;
                                        }
                                }
```

```
                        }
                    }



                }
```

Part 3

```
int i =0;
unsigned int led_colors[8] = {KRED, KYELLOW, KPURPLE, KGREEN, KBLUE, KINDIGO,
KWHITE, KORANGE};

 while (1)
  {
    setDot(colors, NUM_LEDS, spotLocation, led_color);
                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) {
                                                flag =1;
                                        }

                                        if(flag==1 && mouseInfo->buttons[0]!=1) {
                                                        led_color = led_colors[i];
                                                        i++;
                                                        flag=0;
                                        }


                                        if(i>7) {
                                                i = 0;
                                        }

                                }
                        }

                }
```