Lab Assignment 4

Communication between the Human Interface Device and the Platform

The goal of this lab is to provide an introduction to a Human Interface Device (HID), and learn how to use software to carry out communication between the HID and the platform. You will be using a Wiimote as the device and the ZedBoard as the platform. You will be learning how to connect the Wiimote to the ZedBoard through Bluetooth. You will also be learning how to read data from the Wiimote and communicate with it on the ZedBoard. You will also learn a little bit about writing scripts.

Instructions

- Each lab assignment consists of a set of pre-lab questions, the actual time in the lab, and a lab report.
- The pre-lab assignments prepare you for the challenges you will be facing in the actual lab. So that each lab member receives the same benefits, pre-lab assignments have to be solved individually. Feel free to discuss problems with class and lab mates, but complete the assignment on your own.

Reading List

The following reading list will help you to complete the pre-lab assignment. The readings will also help you in subsequent lab assignments. Please complete the readings that are marked ***[Required]*** before attempting the pre-lab assignments.

[1] Materials on the Bash Tutorial ***[Required]***
- http://www.ansatt.hig.no/erikh/tutorial-bash/bash-notes.pdf

[2] Materials on the protocol Bluetooth
- https://en.wikipedia.org/wiki/Bluetooth

Pre-Lab Assignment

We have only very limited time for the lab available. To make efficient use of the lab time, you will need to prepare for it. First, go through the assigned reading above and get an overview of the schematics and the manuals mentioned. Second, approach the questions below to apply your knowledge. Please submit the solution to the pre-lab assignments in PDF format via blackboard.

Code Review

Pre 4.1) Write a bash script that asks the user's name (e.g., David), and says hello to the user (e.g. "Hello David"). The user can either input the name when prompted or read it as an input argument.

Pre 4.2) Look at the code `wiimote_btns.c` program provided on Blackboard. Describe what the program does in your own words, and discuss how it implements this functionality. Note that just copying the file header is not sufficient.

Pre 4.3) Look at the code `wiimote_accl.c` program provided on Blackboard. Describe what the program does in your own words, and include how it implements this functionality. Note that just copying the file header not sufficient.

Pre 4.4) How can the modularity of the code for these two programs be improved? Please note that we will use the Wiimote as an input device in the future.

Lab Assignments

This lab continues to familiarize you with the Zedboard as an embedded computing device and C programming. In this lab you will become more accustomed to working in Linux, write more complex C programs and learn how to use different options with the gcc compiler. Moreover, you will learn how to manipulate different data types in C. You will develop functions to increase the modularity of your code, making it easier to debug.

This laboratory is divided into four parts. In 4.1, you will establish a Bluetooth connection between the Wiimote and the ZedBoard. In 4.2, you will explore the drivers associated with the Wiimote and read packets sent from the Wiimote. In 4.3, you will write programs to control the LEDs on the ZedBoard through the Wiimote buttons. Finally, in 4.4 you will add functionality to the program you developed in 4.3 to control the LEDs through the Wiimote accelerometers.

Connecting to ZedBoard

1. Login into the Windows desktop PC (we will refer to this system as the host) using your myneu credentials.
2. Make sure your Zedboard is powered (plugged in and the power switch on the board is in the ON position), and has been given time to boot up (generally takes a few minutes). Then connect the Zedboard to the PC host via Ethernet to the RJ45 connector.
3. Make an SSH connection to the Zedboard, where the IP address is 192.168.1.10, and port number of 22.

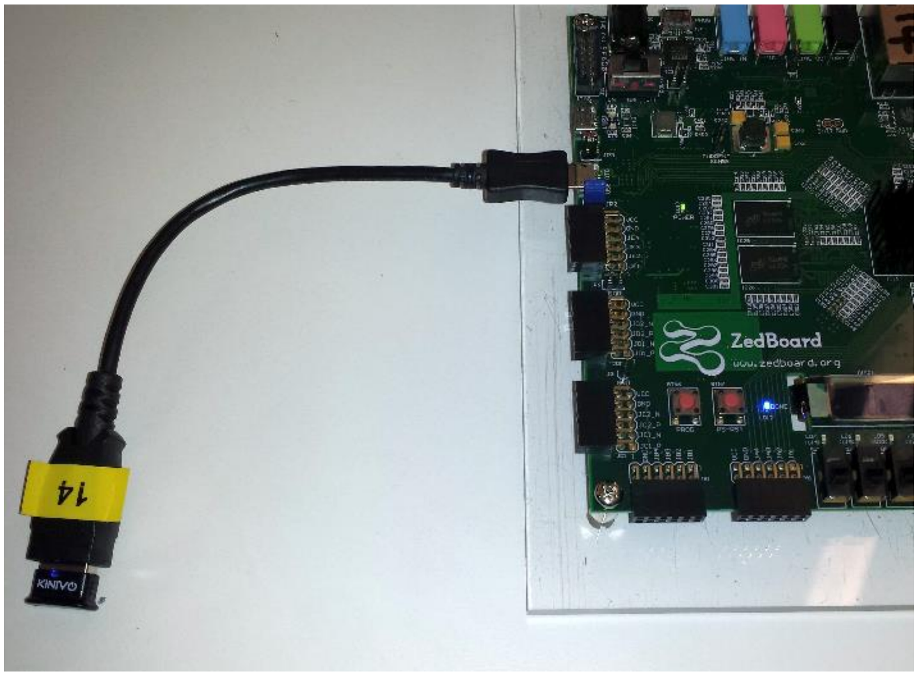
   Don't work as root, you already created your user accounts, use that!

Lab 4.1 Establishing Bluetooth Connection

You will use a USB Bluetooth dongle to establish a Bluetooth connection between the Wiimote and the ZedBoard. The Linux on the ZedBoard uses the BlueZ stack (http://ww.bluez.org), which is an implementation of the Bluetooth standard.

Note that connecting with the Wiimote requires some tricks, which luckily are possible with Linux. It however takes a few steps and some patience. The following instructions guide you through the necessary steps.

1. Making the USB Bluetooth Dongle Operational
   a. Connect the USB Bluetooth dongle (through the standard USB-to-micro USB adapter) to the ZedBoard in the micro USB port on the ZedBoard named "USB OTG" (the label is on the board).



b. Issue:
```
hciconfig –a
```
on the ZedBoard to see the status of the Bluetooth adapter on the platform. Report what you see and try to interpret the information reported.

c. If the device is DOWN (it should be UP by default), issue:
```
sudo hciconfig hci0 up
```
which turns the adapter on. Issue:
```
hciconfig –a
```
again to make sure the device is UP.

Please record the MAC address of the Bluetooth adapter (e.g., XX:XX:XX:XX:XX:XX) connected to the Zedboard. The address is labeled “BD Address:”.

2. Identify Your Wiimote via Bluetooth
   a. Place the Wiimote in pairing mode so that it can be found by the adapter on the ZedBoard.
      i. Remove the battery cover and push the red sync button.
      ii. The four blue LEDs on the Wiimote should be blinking, indicating the device is in pairing mode.
      iii. Pairing mode is active for ~20 seconds (push the red button again if you miss the window)
   b. Issue:
```
hcitool scan
```
The Wiimote should show up as “Nintendo RVL-CNT-01” and a MAC address (e.g., XX:XX:XX:XX:XX:XX). Note, you will see any Wiimote that is trying to sync at the

same time. So, make sure you find yours by checking the MAC address on the Wiimote's box.

3. To make your first connection:
   a. Put the Wiimote into pairing mode (see above to make it visible).
   b. Issue the command:
   ```
   sudo bluez-simple-agent hci0 XX:XX:XX:XX:XX:XX
   ```
   where XX:XX:XX:XX:XX:XX is the Wiimote MAC address (written on the Wiimote's box).
   c. You should issue the previous command while the Wiimote is in paring mode. Make sure the four blue LEDs on the Wiimote are blinking when you issue the command. If not, push the red sync button and issue the command again.
   d. If you complete the previous step correctly, you may be asked for a PIN Code (the PIN Code may be already set for your Wiimote, so you can jump to step 5d). Press "Enter" without inputting any PIN code to cancel pairing. This will generate a message that "Creating device failed…..". This is expected.
4. Some tricks to avoid needing to input the PIN each time:
   a. The directory /var/lib/bluetooth/ lists all Bluetooth adapters known to the operating system. Change the directory (using cd) to /var/lib/bluetooth/<BT_Adapter_MAC>. Replace <BT_Adapter_MAC> with the MAC address found in 1c. List the directory contents. The directory contains files regarding Bluetooth devices that are (or have been connected) to this adapter.
   ```
     cd /var/lib/bluetooth/XX:XX:XX:XX:XX:XX
   ```
   where XX:XX:XX:XX:XX:XX is the Bluetooth Adapter MAC address found in 1c.
   b. Modify the contents of the file "did", to be able to connect to the Wiimote without entering a PIN code. Open the "did" file in an editor, e.g. "sudo vi did". Find the line that resembles "XX:XX:XX:XX:XX:XX FFFF 0000 0000 0000", (where XX:XX:XX:XX:XX:XX is the Wiimote MAC address. Change the last 16 characters, starting with FFFF in that line with "0002 057E 0306 8500". Assuming your Wiimote MAC address is "8C:56:C5:40:00:D0", the line would be "8C:56:C5:40:00:D0 0002 057E 0306 8500". Save and exit the file.
5. Pair and create the connection
   a. Put the wiimote in pairing mode (i.e., push the red button).
   b. As you did in 3b, Issue the command:
   ```
    sudo bluez-simple-agent hci0 XX:XX:XX:XX:XX:XX
   ```
   where XX:XX:XX:XX:XX:XX is the Wiimote MAC address (written on the Wiimote's box).
   c. This should pair the Bluetooth devices without entering a PIN code.
   d. Issue:
   ```
   sudo bluez-test-device trusted XX:XX:XX:XX:XX:XX yes
   ```

to place the Wiimote on the trusted devices list. Again, XX:XX:XX:XX:XX:XX is the Wiimote MAC address.

e. Put the Wiimote into pairing mode one last time. To establish the connection, issue:
   `sudo bluez-test-input connect XX:XX:XX:XX:XX:XX`

   Make sure to use the Wiimote MAC address.

The Wiimote should now have one of the blue lights on all the time, indicating that it is connected to the Zedboard.

6. We will be using the Wiimote in other labs as well. However, after each system restart, the Wiimote needs to be connected again. To simplify the process of reconnection, you will develop a bash script.
   a. Write a bash script (wiimoteConnect.sh). This bash script should define a local variable WII_MAC with your own Wiimote MAC address.
   b. Then, call the three commands: `bluez-simple-agent, bluez-test-device, bluez-test-input.` Please see the instructions in the manual steps above for the right command line parameters. Add echo statements to instruct the user (you) exactly what to do.
   c. Validate that the script works: First, push the red sync button to put the Wiimote into pairing mode. Wait (~20 seconds) until all blue LEDs are off. Now, the Wiimote is disconnected from the ZedBoard. Then, call the script to connect again.

Lab 4.2 Looking at the drivers associated with the Bluetooth connection

1. Issue:
   `dmesg | grep bluetooth`
   and report what you see. Describe in your writeup what the “dmesg” command is used for and how it works. Also discuss what the command “grep bluetooth” is doing.
2. The 5 lines that are output (careful, some lines may wrap on the display) describe the device drivers associated with the Wiimote. You will find a long device path that includes the USB and Bluetooth addresses (which may appear cryptic). You will also notice text labels describing accelerators and IR inputs. Each line ends with “inputX”, where “X” corresponds to the events that we will discuss next.

   Note, each time you connect the device, you will see an additional 5 lines. The additional lines will not affect the rest of the steps in this lab.
3. For each of the inputX, the Linux kernel creates device file in /dev/input. Change the directory so you are looking at the /dev/input directory. List the files. You should see five files (event0 – event4). As seen in the dmesg report, input0 is associated with the Wiimote’s accelerometer and input2 is associated with the Wiimote. File event0 is associated with input0 and file event2 is associated with input2.
4. Issue:
   `sudo hexdump event2`

The command is waiting for input. Now push the buttons on the Wiimote (but not the power button) and report what you see. Issue control-c to exit hexdump.

5. The Wiimote sends a 32-byte packet to event2 file every time a button is pressed. The event value contains the button code, button state (1 for pushed and 0 for released), and some arbitrary values. Report an example sample and state the data format.
6. Do the same experiment, this time for event0 to see the packets for accelerometer values. The receiving packets might be so fast when you move the wiimote. If you fix the Wiimote position (put it on the table) you will be able to see the packets better. The code for X value is 03, the code for Y value is 04, and the code for Z value is 05. The bytes after the codes are the angle values.

Lab 4.3 Wiimote Buttons

In this part, you will control the LEDs on the ZedBoard using the Wiimote.

1. Copy the `wiimote_btns.c` and `wiimote_accl.c` programs to the ZedBoard. These programs read values from the event0 and event2 files, decode the packets and print the actions on the screen. You will use these programs as the base for your programs. Compile and run `wiimote_btns.c` to read values from the event2 file. Then push the buttons on the Wiimote one by one and try to find the code associated with each button. Complete Table 1.

Table 1 Wiimote Button Codes

Button	Code
↑	
↓	
←	
→	
A	
+	
-	
Home	
1	
2	
B	

2. Write a program (`wiimote_led_button.c`) that turns on/off LEDs on the ZedBoard using the buttons on the Wiimote. You should choose 9 buttons (8 buttons for the 8 LEDs and the other for exiting the program). When you push a button, the related LED should turn on and when you release it, the LED should turn off. In your report, record the specification of button assignments. Implementation hints:

a. Make one #define for each of the buttons used. For example, WIIMOTE_BTN_A.
b. Create a switch case statement to evaluate evt2Code. Define a new variable ledNr that contains the LED number to turn on or off.
c. Then, use an if-statement to distinguish if LED should be turned on or off based on *evt2Value*.
d. For terminating the while loop, use define a flag “stayInLoop” set this flag to 1 at beginning. If the key for end is received, set the stayInLooop variable to 0, and the while loop exits.

Lab 4.4 WiiMote Accelerometer

3. Write another program (`wiimote_led_accel.c`) that displays the X-axis acceleration on the bank of LEDs. Base this program on the downloaded `wiimote_accl.c` from blackboard. Assume that the X-axis acceleration is between -100 and 100. With -100, all LEDs should be off. As X-axis acceleration increases, the LEDs should turn on one by one with an equal spacing. With X=100, all 8 LEDs should be on.
   a. Add a condition to only look at the X-axis acceleration events. Hint, use a #define.
   b. Compute the acceleration value out of the high byte *evt0ValueH* and the low byte *evt0ValueL*. Store the acceleration (a signed value) in the variable *accel*. Change the print statement from using “%x%x” (which prints two hexadecimal bytes) to print a signed decimal with the computed acceleration value. Validate the results by compiling and running the program.
   c. Compute how many LEDs should be turned on based on the computed acceleration value. The computed value should be between 0 and 8. Expand the printf statement to show the numbers of LEDs to be turned on.
   d. Write a function `userio_ledBar`( ) that visualizes the LED lights, with a 0 lighting no LEDs and a 8 lighting all of them. Compute the X-axis acceleration value to determine which LEDs should be lit.

Laboratory Report

You should follow the lab report outline provided on Blackboard. Your report should be developed on a wordprocessor (e.g., OpenOffice or LaTeX), and should include graphics when trying to present a large amount of data. Submit all programs developed in the appendix of your report.