SECTION EIGHT
INTEL ATOM PROCESSOR LAB
NUNCHUK DRIVER WITH I²C (SMBUS) DATA TRANSFER

SECTION OVERVIEW

- Learn basic programming techniques for device drivers and modules in the Linux kernel.
- Improve your knowledge of I^2C driver architecture and software.

PRELAB

Be sure to have completed these prelab tasks before your lab. They will help you perform the tasks in this lab and hopefully give you more time to experiment.

1. You will need to familiarize yourself with the C functions used to communicate with the I^2C device. Look at the following functions in the Linux source file **`i2c-dev.h`** and describe what each one does.
   a. **`i2c_smbus_read_byte()`**
   b. **`i2c_smbus_write_byte()`**
   c. **`i2c_smbus_write_byte_data()`**

2. For the second part of the lab, you will need to use the QT framework to implement a "painting" program using the Wii Nunchuk. Read the Qt documentation provided on the class website for the QPainter, QPen and QPoint classes. Define the following functions:
   a. For QPainter: **`setPen(), drawLine()`**
   b. For QPen: **`setColor()`**
   c. For QPoint: **`setX(), setY()`**

3. Read the AIMB 213 User Manual with focus on the GPIO Port pins and the JFP 1 & JFP 2 Port pins and the Wii Nunchuk guide from the class website. Define the following terms:
   a. **GPIO** – General Purpose Input/Output, **SMBus** – System Management Bus, **I^2C** – Inter-Integrated Circuit

PROCEDURE

This is the final lab! This lab will let you work with the Intel Atom Processor (AIMB 213 Embedded Developer Board). The basic architecture of the AIMB 213 board is as follows:

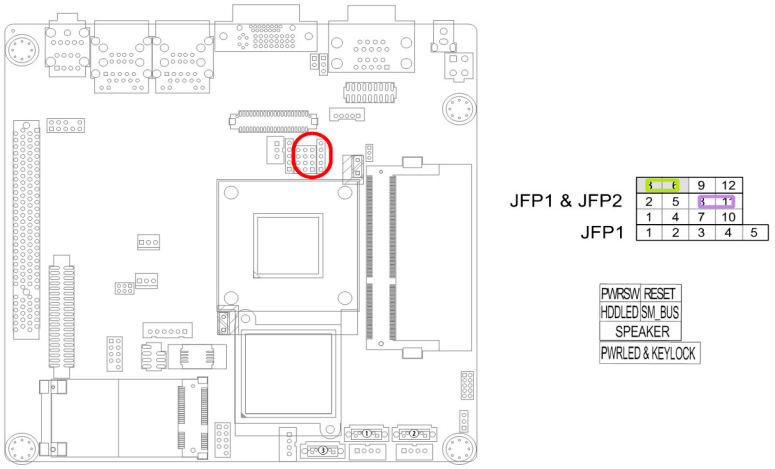

Figure 1: JFP1 & JFP2 Connector

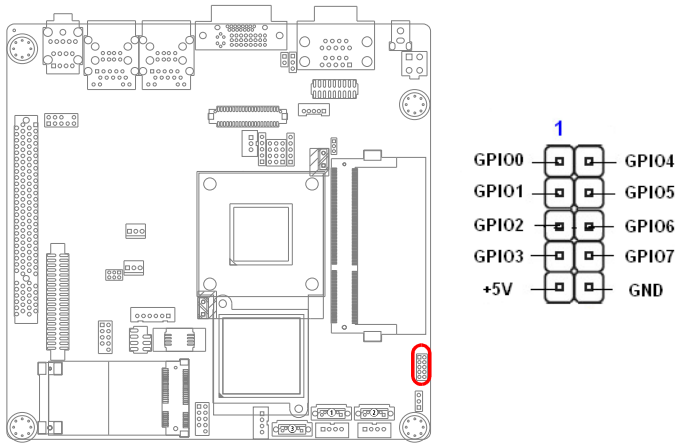

Figure 2: GPIO Connector

Specifications

1. The AIMB 213 board is powered on by shorting pins 3 and 6 of the JFP1 & JFP2 connector. You can plug in a power button to do this.

2. The Wii Nunchuk SMBus pins connect to pins 8 (SDA) and 11 (SCL) on the JFP1 & JFP2 connector. It can be powered by plugging the power and ground pins into pins 9 and 10 on the GPIO connector.

3. Log into the Linux desktop using the username ***root*** and the password ***root123***. The terminal program is located at ***Applications > System Tools > Terminal***.

4. To access the I²C adapters, load the ***i2c-dev*** and the ***i2c-i801*** modules in the Linux kernel using the **`modprobe`** command (for example: **`modprobe i2c-dev`**). The two modules are in **`drivers/i2c/i2c-dev.ko`** and **`drivers/i2c/busses/i2c-i801.ko`** of the Linux source tree.

5. At the terminal, try using the commands **`i2cdetect -l`**, **`ls -l /sys/class/i2c-adapter`**, and **`ls -l /sys/class/i2c-dev`** to see what is available on the system.

6. For user space access, **`#include /usr/include/linux/i2c-dev.h`** should be included in your program.

7. When ***libi2c-dev*** is installed, **`/usr/include/linux/i2c-dev.h`** is changed for user access. It contains macros for the SMBus protocol.

TASK: GUI TO SIMULATE PEN MOVEMENT USING WII NUNCHUK

Primary Goal

Design a GUI application that can be used to draw geometric shapes on a widget.

Procedure

The GUI application should consist of two sections.

1. Load the `i2c-dev.ko` and the `i2c-i801.ko` files using the `modprobe` command in the Linux root directory. (Username: ***root*** Password: ***root123***)

2. The SMBus can be accessed using pins 8 and 11 of the JFP1 & JFP2 connector.

3. The Wii Nunchuk uses a proprietary connector whose data and clock pins are connected to pins 8 and 11 of the JFP1 & JFP2 connector. 5V power supply for the Nunchuk is supplied by interfacing with the pins 9 and 10 of the GPIO connector.

4. The first section should invoke the SMBus API to read Wii Nunchuk data every 100ms.

5. It should then compute the new position of the pen by computing the X and Y-axis values of the Wii Nunchuk's analog stick to indicate the speed of pen movement.

6. The six bytes of the Wii Nunchuk output, which represents the X, Y and Z accelerometers, the X and Y-axis analog stick and the C and Z button values should be decoded (and included in your report).

7. The second section should use the X and Y-axis values of the analog stick to draw the trajectory of the pen movement on a 2D graphical window. For this task you may want to consider using the Qt `Painter` class in your program.

8. The C button should be used to change colors and the Z button to reset the graphical window.

Note: To create the GUI application use the **Qt C++ library**, which is used to build GUI applications for embedded systems.

Wii Nunchuk Information

The frequency of the Wii Nunchuk is 100KHz. The I²C slave address of the Nunchuk is 0x52.

The default values (6 bytes) expected from Nunchuk are as follows. For X and Y-axis values in the first rows, Min (Full Left) means the analog stick of the Wii Nunchuk is the to the extreme left along the axis. Medium (Center) means the stick is the default position and Max (Full right) means the stick is in the extreme right position.

Value in example	Description	Possible Nunchuk Outputs
0x7E	X-axis value of the analog stick	Min (Full Left) **0x1E** / Medium (Center) **0x7E** / Max (Full Right) **0xE1**
0x7B	Y-axis value of the analog stick	Min (Full Down) **0x1D** / Medium (Center) **0x7B** / Max (Full Up) **0xDF**
0xAF	X-axis acceleration value	Min (At 1G) **0x48** Medium (At 1G) **0x7D** Max (At 1G) **0xB0**
0x80	Y-axis acceleration value	Min (At 1G) **0x46** Medium (At 1G) **0x7A** Max (At 1G) **0xAF**
0x7A	Z-axis acceleration value	Min (At 1G) **0x4A** / Medium (At 1G) **0x7E** / Max (At 1G) **0xB1**
0x3B	Button state (Bits 0:1) Acceleration least significant bits (Bits 2:7)	Bit 0: Z-Button (0 = pressed, 1 = released) Bit 1: C-Button (0 = pressed, 1 = released) Bits 2-3: X acceleration LSB Bits 4-5: Y acceleration LSB Bits 6-7: Z acceleration LSB

The data read from the Nunchuk should be decoded as follows:

Extracted data = (Data read from Nunchuk XOR 0x17) + 0x17