Introduction to Embedded Software

Practice #4

Linux device model for an I2C driver

Dongkun Shin
Embedded Software Laboratory
Sungkyunkwan University
http://nyx.skku.ac.kr/

Objective

- Add an I2C device to a device tree.
- Find your driver and device in /sys.
- Implement basic i2c device control program.

Day Practice

1) Python Program
   - Make function of each component
   - Print variation of each component with written function

Add a New Device on Device Tree

Modify Device Tree

- /arch/arm/boot/dts

```
// host PC
$ cd linux/arch/arm/boot/dts/
$ vi bcm2710-rpi-3-b.dts
```

```
&i2c1 {
    pinctrl-names = "default";
    pinctrl-0 = <&i2c1_pins>;
    clock-frequency = <100000>;

    status = "okay";

    wiichuk: nunchuk@52 {
        compatible = "nintendo, nunchuk";
        reg = <0x52>;
    };
};
```

Rebuild Kernel

```
// host PC
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
    zImage modules dtbs -j32
$ scp arch/arm/boot/dts/*.dtb  pi@ip:/home/pi
$ scp arch/arm/boot/dts/overlays/*.dtb*  pi@ip:/home/pi
```

```
// raspberry-pi
$ sudo mv *.dtb /boot
$ sudo mv *.dtb* /boot/overlays
$ sudo reboot
```

Check Device Tree

- You can see nunchuk device is registered that you modified

```
// raspberry-pi
$ cd /sys
$ sudo find . -name "*nunchuk*"
```

```
pi@raspberrypi:/sys $ sudo find . -name "*nunchuk*"
./firmware/devicetree/base/soc/i2c@7e804000/nunchuk@52
```

Device Tree Compiler

- **You can see structure of device tree using DTC**

```
// raspberry-pi
$ dtc -I fs /proc/device-tree
```

```
compatible = "raspberrypi,3-model-b", "brcm,bcm2837";
serial-number = "00000000b88f3334";
model = "Raspberry Pi 3 Model B Rev 1.2";
memreserve = <0x3b400000 0x4c00000>;
interrupt-parent = <0x1>;
#address-cells = <0x1>;
#size-cells = <0x1>;

clocks {
        compatible = "simple-bus";
        #address-cells = <0x1>;
        #size-cells = <0x0>;

        clock@3 {
                compatible = "fixed-clock";
                #clock-cells = <0x0>;
                phandle = <0x4>;
                reg = <0x3>;
                clock-output-names = "osc";
                clock-frequency = <0x124f800>;
        };

        clock@4 {
                compatible = "fixed-clock";
                #clock-cells = <0x0>;
                phandle = <0x18>;
                reg = <0x4>;
                clock-output-names = "otg";
                clock-frequency = <0x1c9c3800>;
        };
};
```

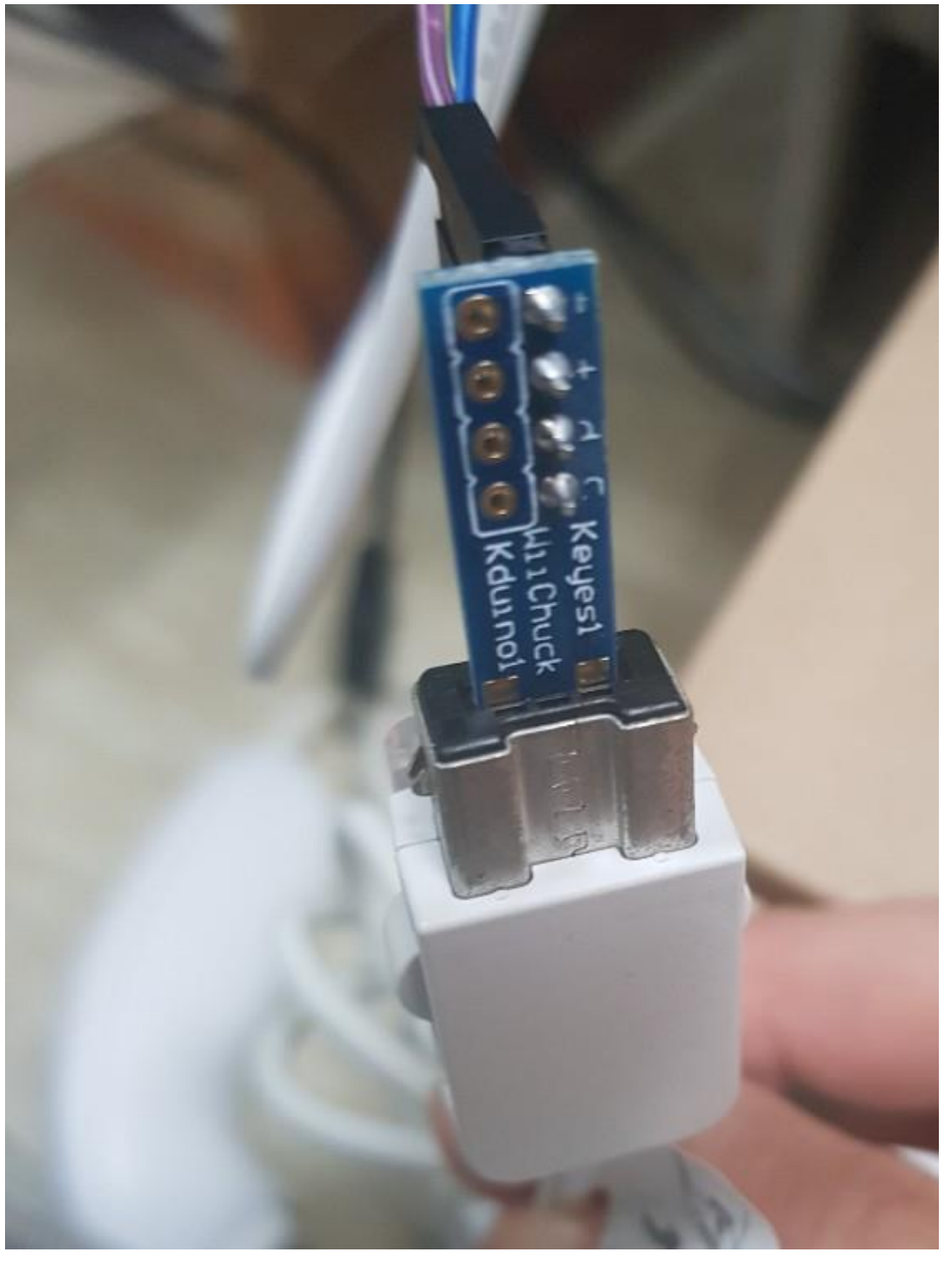
Implement i2c Device Control Program

Kernel Level vs User level

- **We can get a sensor data both in kernel level and user level**
- **In this lecture, we first try to get a sensor data with user level program**
- **In user level, we use SMbus protocol to get data**

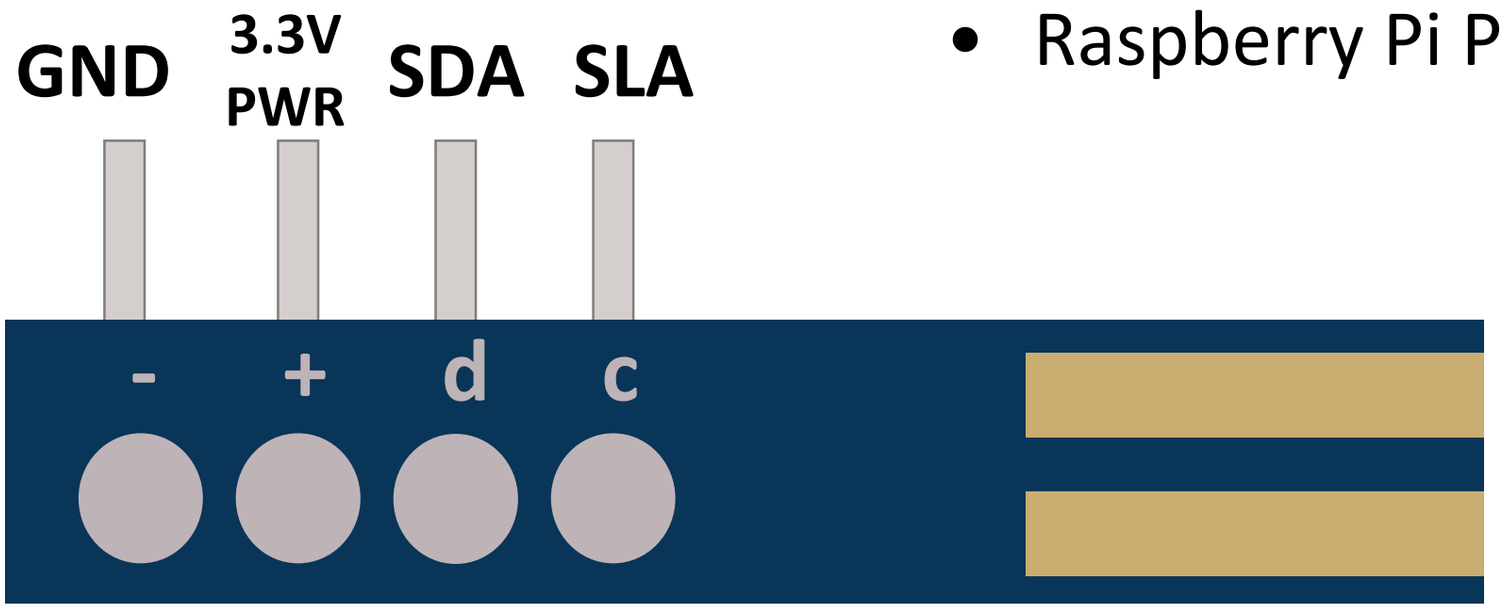
Connecting the nunchuk

- **Connect numchuk to adapter**



Connecting the nunchuk

- Raspberry Pi Pin Layout



- Wii WiiChuck Nunchuck Adapter

i2c Device Detect

```
// raspberry-pi
$ sudo apt-get install i2c-tools
$ sudo i2cdetect -y 1
```

```
pi@raspberrypi:~/gwang $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- 52 -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Requirements

```
// raspberry-pi
$ sudo apt-get install python-smbus
$ sudo apt-get install python3-smbus
```

Wii Nunchuk Output

Data byte receive								Address
Joystick X								0x00
Joystick Y								0x01
Accelerometer X (bit 9 to bit 2 for 10-bit resolution)								0x02
Accelerometer Y (bit 9 to bit 2 for 10-bit resolution)								0x03
Accelerometer Z (bit 9 to bit 2 for 10-bit resolution)								0x04
Accel. Z bit 1	Accel. Z bit 0	Accel. Y bit 1	Accel. Y bit 0	Accel. X bit 1	Accel. X bit 0	C-button	Z-button	0x05

Byte 0x00 : X-axis data of the joystick

Byte 0x01 : Y-axis data of the joystick

Byte 0x02 : X-axis data of the accellerometer sensor

Byte 0x03 : Y-axis data of the accellerometer sensor

Byte 0x04 : Z-axis data of the accellerometer sensor

Byte 0x05 : bit 0 as Z button status - 0 = pressed and 1 = release

bit 1 as C button status - 0 = pressed and 1 = release

bit 2 and 3 as 2 lower bit of X-axis data of the accellerometer sensor

bit 4 and 5 as 2 lower bit of Y-axis data of the accellerometer sensor

bit 6 and 7 as 2 lower bit of Z-axis data of the accellerometer sensor

Base Code

- Using base code, implement your own python program

```
from smbus import SMBus
import RPi.GPIO as rpi
import time as time

bus = 0

class nunchuck:

  def __init__(self,delay = 0.05):
    self.delay = delay
    i2c_bus = 1
    self.bus = SMBus(i2c_bus)
    self.bus.write_byte_data(0x52,0x40,0x00)
    time.sleep(0.1)

  def read(self):
    self.bus.write_byte(0x52,0x00)
    time.sleep(self.delay)
    temp = [(0x17 + (0x17 ^ self.bus.read_byte(0x52))) for i in range(6)]
    return temp
```