Report: 40
Demo: 40
Prelab: 10
Delay: 0

Total: 90

Lab1-Report

Haoxiang Zhang

September 12, 2019

Contents

Part I

Prelab

1 Week-2

1.1 mmap()

Argument

Mapping or project the address from virtual space or disk address to process with starting address and the length of the memory, then we could use pointer to operate this memory. The return value is the pointer of the space. It is efficient to read and write.

1.2 ioremap()

Argument

The purpose of ***ioremap()*** is that project a size of space that has a start I/O address and specific size from RAM of I/O device to CPU as a virtual space. ***ioremap()*** will return an address pointer that is the address of the starting address.

1.3 Bit Masking

Because in order to change one I/O PIN or several but not all of them, it is better to use Bit Masking to tell the process to change the specific I/O PINs.

For example:

We want to change only PIN 2, PIN 6, and PIN9, just do this:
0000 0000 0000 0000 0000 0000 0000 0000 ← the most right one is 0
0000 0000 0000 0000 0000 0010 0100 0100 ← set PINs
0x244 Then use iowrite32(0x244, GPSET0) to set the PINs as 1 ← change the binary to Hex

40/50

Part II

Postlab

2 Week-1

2.1 Assignment-1

Objectives and Lab Description

5/5

Turn on and off two of the LEDs on the auxiliary board, for every one second, the lights should be like this, one light is ON and the other is OFF, next one second, the light ON one OFF, and the OFF one ON.

Implementation

I wrote two functions, ***lightAllOn()*** and ***lightAllOff()*** to control the lights. First of all, in order to use the functions that could control PINs, we have to include header file *"wiringPi.h"*. Secondly, use ***wiringPiSetup()*** to make sure the the functions could use the functions from wiringPi.h. Thirdly, set PINs as INPUT or OUTPUT with function ***pinMode(PIN_num, Operator)***. Fourthly, use ***lightAllOff()*** as initialization. Fifthly, I declared int turn =0. In the while loop that set param as endless, if turn equal 0, then one light ON one light OFF, then set turn as 1. If turn equal 1 then switch the light and set turn back to 0. To control the light output, use ***digitalWrite(pin_num, 1 or 0)***, when it is 1 then the pin set as high, light on, when it is 0, otherwise. Last, set ***delay(1000)*** at the bottom in the while loop. ***delay(1000)*** meas delay 1000 ms.

10/10

Experiments and Results

At the beginning, I set all the lights on and off at every second. It worked very well. Later, I followed the task description to set red LED (pin8) and blue LED(pin21) to alternately on and off every one second. It worked well too. At last, I use Ctl + z to end the program.

15/15

2.2 Assignment-2

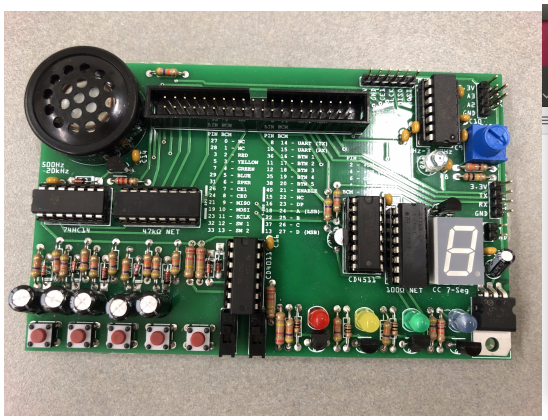
Objectives and Lab Description

Create a square wave and send to speaker. On the Auxiliary Board, there are five red buttons on the left-bottom of the board. User will enter a number from 1-5, then only could put the button that the number just enter. For example, if user enter number 3, then only the third button works. after press the button, the speaker is making sound constantly.

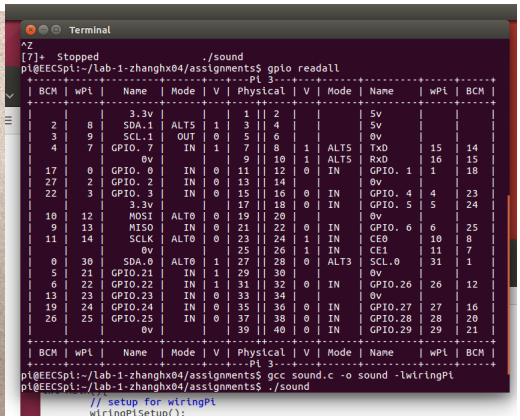
Implementation

At first, declare a function ***biu(int *freq)*** that could make sound with freq frequency. Then, setup wiringPi for supporting the wiringPi functions and set up PINs as well. After that, set the input from user. Then according to the input to set the relative pin number and frequency. Finally, give a while loop and set if the button is pressed, then set the button's PIN keeping HIGH with function ***pullUpDnControl(pin_num, PUD_UP)***.

NOTE: for he PIN number, first, check the PIN number table in the center of **Auxiliary Board**, then find the **BCM number** on the gpio table in terminal that use *"gpio readall"* to show to get wPi number.



(a) Auxiliary Board



(b) GPIO readall

Experiments and Results

After run the program, first, I enter 5 in terminal. After that, when I press button1 to button4, they did not work. When I press button5, the speaker is making sound. Press Ctl+z to stop the program, and run it again. The other button makes sound with different frequency.

2.3 Discussion and Postlab Questions

1. Imagine that in the user-space program you want to terminate the sound by pressing a key on the keyboard. How could you do that? Are there any implementation difficulties? You do NOT have to implement it; only discuss it in the report.

answer: My idea is that put code at begining of while loop:

```
if(_kbhit()){
    break;
}
```

use ***_kbhit()*** to check whether the keyboard is pressed or not. ***_kbhit()*** return the **bool** value.

5/15

2. How do you think you could solve the frequency issues to get a "nice", consistent sound in the speaker?

answer: I think just set the delay between HIGH wave and LOW wave as quick as possible. When I set delay(1), the sound is really nice and consistent. It is not ugly like set delay(50).

Discussion for part-1 ??

3 Week-2

3.1 Assignment-1

Objectives and Lab Description

Access the GPIO port on terminal directly. Turn On and Off all the LEDs on the auxiliary board. And try to get the value of buttons (both with pressing and without pressing).

Implementation

There are Three ways to implement it:

Method 1:

First of all go to folder */**sys/class/gpio***, then for example enter **"echo 2 > export"**, which tell export to make a file named *"gpio2"*. Secondly, enter **"echo out > gpio2/direction"**, which set PIN2's direction as output. Thirdly, enter **"echo 1 > gpip2/value"** to set PIN2 as **HIGH** or enter **"echo 0 > gpio2/value"** to set PIN2 as **LOW**.

Method 2:

This method is to set GPIO with **wiringPi numbering**. First, enter **"gpio mode <pin> in/out"** to set pin as input or output. Then enter **"gpio write <pin> 1/0"** to set pin as **HIGH** or **LOW**.

Method 3:

This method is to set GPIO with **BCM_GPIO numbering**. First enter **"gpio export <pin> in/out"** to set pin as input or output. Then enter **"gpio -g write <pin> 1/0"** to set pin as **HIGH** or **LOW**.

To read the value of button, enter **"gpio -g read <pin>"** to get the value of pin.

Experiments and Results

The method 1, when I echo pin number 2 to export, the under the folder */gpio/* will create a new file */gpio2/*. after I echo the HIGH or LOW to the value, the light is ON or OFF. For the method 2 and method3. The detail is in the ***Figure 1***. To catch in PIN's infomation, enter ***"gpio exports"*** to get all the PINs that exported. When I read the value of button, it return 0 if I did not press the button and it return 1 if I was pressing the button.

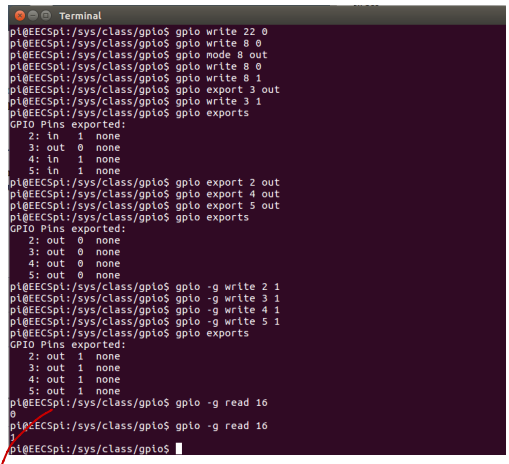


Figure 1: GPIO Operation

3.2 Assignment-2

Objectives and Lab Description

This task, we will not be allowed to use any library such as wiringPi. We will create a kernel module and turn all the LEDs when the module is installed, and turn all the LEDs when the module is removed. For implementing it, we should access the GPIO ports directly through GPSET, GPCLR, GPLEV register.

Implementation

Because of the new RaspberryPi, the GPIO base address is changed. the GPIO base address is **0x3F200000**. First of all, we should give the permission to install module. with **MODULE_LICENSE("GPL");**. Then we declare a unsigned long pointer *vaddr*. When we install the module, will run the code ***init_module()***, and when we remove the module, will run the function cleanup_module(). Thus, secondly, give vaddr the virtual address which is the return value of function ***ioremap(start address, size)***. The purpose of ioremap() function is to project a space from memory of device to CPU start from the given start address with specific size that given. Thirdly, checked the table in page 90 in the PDF **BCM2837-ARM-Peripherals.pdf**. Set the 3bit of binary of the position that the LED located as 100. So if set all four LEDs, then the hex is **0x9240**. Use **iowrite32(0x9240, vaddr)** in order to tell the CPU to set all four LEDs' PIN as output. Fourthly, use **iowrite32(0x3c, vaddr + 7);** to set PIN as HIGH which means turn the light up. vaddr+7 is the address of GPSET0 which set PIN to 1. Lastly, in the function ***cleanup_module()***, use ***iowrite32(0x3c, vaddr + 10);*** to set PINs to 0 which means turn light off.

After finish the code, in the termial, we should use **make** to run the *makefile* to create **.ko file.** Then enter *sudo insmod filename.ko* to install the module which run the *init_module()*, then enter *sudo rmmod modulename* which runs *cleanup_module()*. And, we can use ***dmesg | tail*** to get the log.

Experiments and Results

When install the module, all the lights turn ON; and when remove the module, all the lights turn OFF.

Discussion for part-2 ??

Part III

Comments: 5/5

Code Section

4 Lab-1 Week-1 Task-1

Switch two LEDs ON and OFF every one second

Code: 40/40

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

/*  Because we are using wiringPi
so we should use wiringPi numbering for PINs

 PIN name       BCM_GPIO     wiringPi
    red lED:         2           8
yellow LED:         3           9
 green LED:         4           7
  blue LED:         5           21
*/
void lightAllOn(){
        // turn all LEDs ON
        digitalWrite(8,1);
        digitalWrite(9,1);
        digitalWrite(7,1);
        digitalWrite(21,1);
}
void lightAllOff(){
        // turn all LEDs OFF
        digitalWrite(8,0);
        digitalWrite(9,0);
        digitalWrite(7,0);
        digitalWrite(21,0);
}
int main(){
        // setup wiringPi
        wiringPiSetup();
        // set goal pins to operatable
        pinMode(8,OUTPUT);      // red
        pinMode(9,OUTPUT);      // yellow
        pinMode(7,OUTPUT);      // green
        pinMode(21,OUTPUT);     // blue
        // set all lights off
        lightAllOff();
```

```
        /* part of here
         * shining the light per second */
        int timer = 0;
        int turn = 0;
        while(1){
                if(turn == 0){
                        // red light ON and blue light OFF
                        digitalWrite(8,1);
                        digitalWrite(21,0);
                        turn = 1;
                }else{
                        // red light OFF and blue light ON
                        digitalWrite(8,0);
                        digitalWrite(21,1);
                        turn = 0;
                }
                // delay one second
                delay(1000);
        }
        return 0;
}
```

5 Lab-1 Week-1 Task-2

Make sound by press the button that entered in terminal

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

void biu(int *freq){
        /*
         for this function,
         set the signal of pin 22 to High
         then delay it for giving time for processing data
         set the sighal of pin 22 to Low
         then delay it for giving time for processing data
                figure is like:
                high(wait) low (wait) hight (wait) low (wait)
                ----------|_________|----------|_________|-
        */
        digitalWrite(22, 1);
        delay(*freq);
        digitalWrite(22, 0);
        delay(*freq);
}

// Set All Button as OFF
void init(){
        pullUpDnControl(27, PUD_OFF);
        pullUpDnControl(0, PUD_OFF);
        pullUpDnControl(1, PUD_OFF);
        pullUpDnControl(24, PUD_OFF);
        pullUpDnControl(28, PUD_OFF);
}

int main(){
        // setup for wiringPi
        wiringPiSetup();
        // setup pins
        pinMode(27,INPUT);      // btn1
        pinMode(0,INPUT);       // btn2
        pinMode(1,INPUT);       // btn3
        pinMode(24,INPUT);      // btn4
        pinMode(28,INPUT);      // btn5
        pinMode(22, OUTPUT);    // spkr
        // main
```

```
int input;
int pin_num;
int freq;
int mode;

// set enter
while(1){
        printf("Enter 0 for normal sound, Enter 1 for fancy sound:\n");
        scanf("%d",&mode);
        printf("Enter the Button Number: \n");
        scanf("%d",&input);
        if (input >= 1 && input <= 5 && (mode == 0 || mode == 1)){
                break;
        }
}
//init
init();
// change the input number to pin number
//and set the frequency of wave
switch(input){
        case 1:// button 1
                pin_num = 27;
                freq = 100;
                break;
        case 2:// button 2
                pin_num = 0;
                freq = 50;
                break;
        case 3:// button 3
                pin_num = 1;
                freq = 30;
                break;
        case 4:// button 4
                pin_num = 24;
                freq = 15;
                break;
        case 5:// button 5
                pin_num = 28;
                freq = 1;
                break;
        default:
                break;
}

// mode 0 works well and making an ugly sound :)
```

```
        if (mode == 0){
                while(1){
                        if(digitalRead(pin_num)){
                                // set the pin on, so the speaker will be sounding wit
                                pullUpDnControl(pin_num, PUD_UP);
                                biu(&freq);
                        }
                }
        // mode 1 supposed to making a nice alarm sound, but does not implement well
        } else if (mode == 1){
                while(1){
                        if(digitalRead(pin_num)){
                                // set the pin on, so the speaker will be sounding wit
                                pullUpDnControl(pin_num, PUD_UP);
                                break;
                        }
                }
                int s;
        /*
         for the alarm sound,
         the tone goes from high frequency to low frequency
       */
                for (s = 1; s <1000; s++){
                        if (s >= 995){
                                s = 1;
                        }
                        biu(&s);
                }
        } else {
                printf("you are amazing!");
        }

        return 0;
}
```

6 Lab-1 Week-2 Task-1

Turn all LEDs OFF and ON directly with COMMAND LINE and get the value of button

```
# turn red light on
# the BCM of Red light is 2

Method-1:
$ echo 2 > /sys/class/gpio/export                          // select port number
$ echo out > /sys/class/gpio/gpio2/direction               // set port as output
$ echo 1 > /sys/class/gpio/gpio2/value           // set port as HIGH (turn on)
$ echo 0 > /sys/class/gpio/gpio2/value                     // set port as LOW (turn off)

Method-2:
in command line:
using wiringPi numbering:
gpio mode <pin> in/out          // set wiringPi pin as input or output
gpio write <pin> 1/0            // set wiringPi pin as 1(HIGH) or 0(LOW)

using BCM_GPIO numbering:
gpio export <pin> in/out        // set BCM_GPIO pin as input or output
gpio -g write <pin> 1/0         // set BCM_GPIO pin as 1(HIGH) or 0(LOW)

read the value of button:
gpio -g read 16                 // read the value of pin16(button1)

when using command line without pressing button, the value is 0
when using command line  with   pressing button, the value is 1
```

7 Lab-1 Week-2 Task-2

Operate the PINs directly by GPIO to turn ON and OFF the LEDs

```
#ifndef MODULE
#define MODULE
#endif
#ifndef __KERNEL__
#define __KERNEL__
#endif
#include <linux/module.h>
#include <linux/kernel.h>
#include <asm/io.h>
#include <linux/init.h>
#include <linux/types.h>

#define GPIOBASE 0x3F200000
MODULE_LICENSE("GPL");          // to give the permission to install module
unsigned long *vaddr;

// Old Style
int init_module(){
        printk("Load the module\n");
        vaddr = (unsigned long*)ioremap(GPIOBASE, 4096);
        //    vaddr: 0xf3200000
        // vaddr+1: 0xf3200004
        //GPFSEL0  set RED      PIN 2 as output 0b_          0100 0000  0x40
        //GPFSEL0  set YELLOW   PIN 3 as output 0b_     0010 0000 0000  0x200
        //GPFSEL0  set BLUE     PIN 4 as output 0b_0001 0000 0000 0000  0x1000
        //GPFSEL0  set GREEN    PIN 5 as output 0b_1000 0000 0000 0000  0x8000
        //                             SET ALL: 0b_1001 0010 0100 0000  0x9240
        iowrite32(0x9240, vaddr);       //GPFSEL0

        printk("%x \n",vaddr);
        printk("%x \n", vaddr + 7);

        // 0000 0000 0000 0000 0000 0000 0011 1100 --> 0x3c
        iowrite32(0x3c, vaddr + 7);     //GPSET0

        printk("light it up\n");
        return 0;
}

void cleanup_module(){
        iowrite32(0x3c, vaddr + 10);    //GPCLR0
```

```
        printk("Unloaded the module\n");
}

/*
// New Style
static int __init myStart(void){
    printk(KERN_INFO "Loading the moduel\n");
    return 0;
}
static void __exit myEnd(void){
    printk(KERN_INFO "Goodbye Mr.\n");
}
module_init(myStart);
module_exit(myEnd);
*/
```