Development of a Raspberry Pi kernel module and an application code to interface a 7 segment display and a button to implement a digital die.

Submitted by,

Glenn Elish Peter	4NM19EC055
Nishan N.	4NM19EC106
Sudheshna Rao	4NM19EC178
Vishaka Pai K.	4NM19EC198

Description

The objective of this project is to develop a kernel module and an application code for a Raspberry Pi to interface a 7 segment display and a button to implement a digital die. The hardware included a Raspberry Pi 3, a 7 segment display, IC 7447, a 4 pin push button, and jumper cables.

The kernel code is developed from an example code built for LEDs, and modified to configure four GPIO pins as output pins. These four pins are given as input to the 7447 IC , which is a 4x7 decoder to translate 4 bit binary BCD to 7 pin configuration of the 7 segment display.

The logic for the design is such that whenever the pin connected to the push button grounded, a random number between 1 to 6 is generated and displayed on the 7 segment display.

The application code interfaces a button and generates random numbers between 1 to 6, which is then displayed on the 7 segment display by writing a number into a device file. The kernel module reads this device file and displays the corresponding number on the 7 segment display.

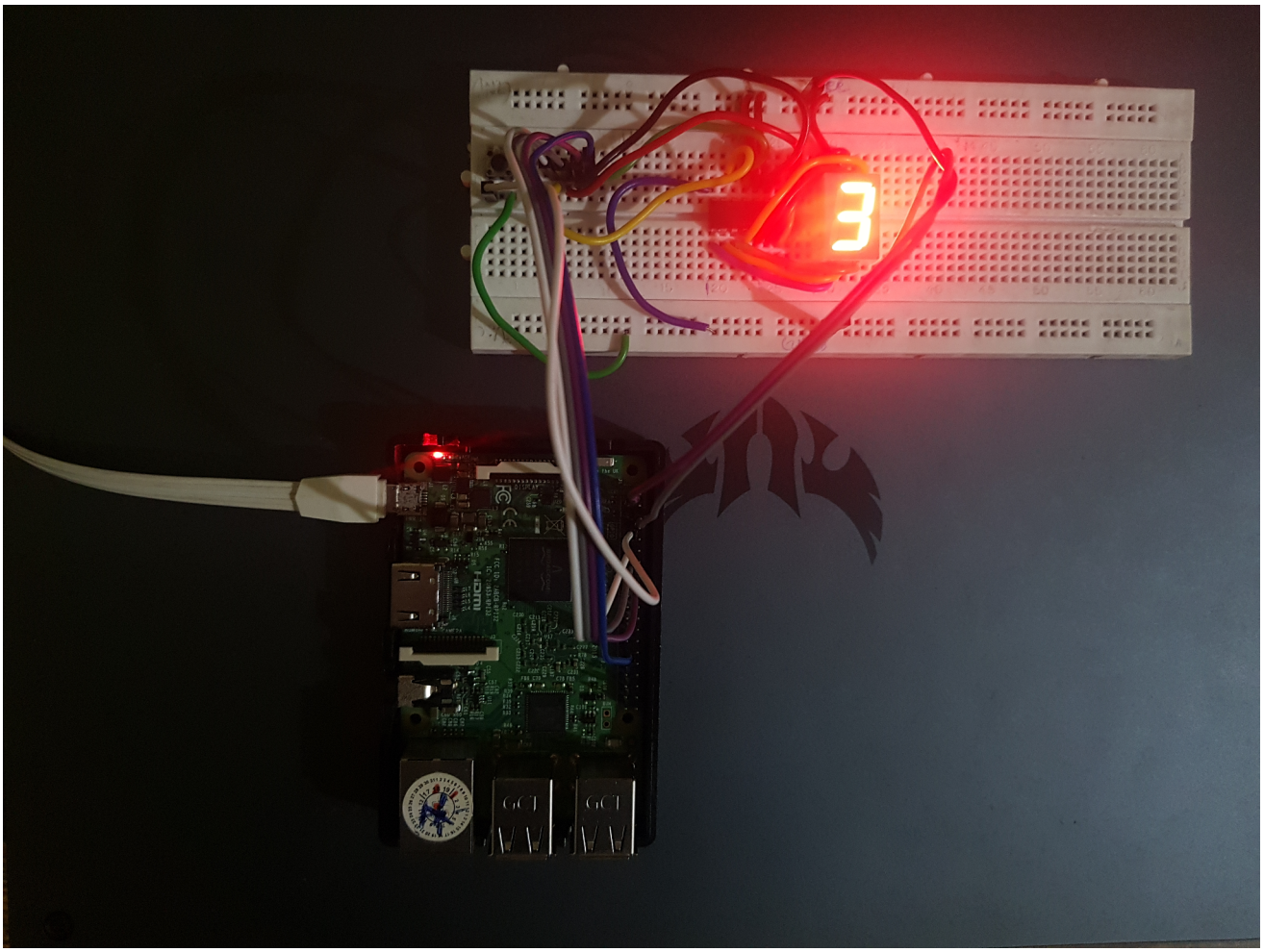

Fig. 1 Hardware setup

Code

7seg.c

```
#include <linux/init.h>         //LKM-Library
#include <linux/module.h>       //LKM-Library
#include <linux/gpio.h>         //Library for the GPIO´s
#include <linux/fs.h>           //Library for the file_operation Definitionen
#include <asm/uaccess.h>        //Library for the copy_from_user Funktion
#include <linux/slab.h>         //Library for the Speicherreservierung
#include <linux/kernel.h>       //Library for KERN_INFO



#define Sa 17 // Pin 17 (7SEG a)
#define Sb 27 // Pin 27 (7SEG b)
#define Sc 22 // Pin 12 (7SEG c)
#define Sd 5  // Pin 5 (7SEG d)

#define SUCCESS 0
#define DEVICE_NAME "sevenseg"
#define BUF_LEN 1

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Glenn Elish Peter, Nishan N, Sudheshna Rao, Vishaka
Pai K");
MODULE_DESCRIPTION("7-Segment Driver");
MODULE_SUPPORTED_DEVICE("Raspberry Pi 3 Model B V1.2");
MODULE_VERSION("0.1");



static ssize_t seven_write(struct file *, const  char *, size_t, loff_t *);
```

```
struct file_operations seven_fops = {

        write : seven_write,

};

static char *msg = NULL;
static int Major;



void seven_status(int display) {

        switch (display) {
        case 1:
                gpio_set_value(Sa, 1);
                gpio_set_value(Sb, 0);
                gpio_set_value(Sc, 0);
                gpio_set_value(Sd, 0);
                break;
        case 2:
                gpio_set_value(Sa, 0);
                gpio_set_value(Sb, 1);
                gpio_set_value(Sc, 0);
                gpio_set_value(Sd, 0);
                break;
        case 3:
                gpio_set_value(Sa, 1);
                gpio_set_value(Sb, 1);
                gpio_set_value(Sc, 0);
```

```
        gpio_set_value(Sd, 0);
        break;
    case 4:
        gpio_set_value(Sa, 0);
        gpio_set_value(Sb, 0);
        gpio_set_value(Sc, 1);
        gpio_set_value(Sd, 0);
        break;
    case 5:
        gpio_set_value(Sa, 1);
        gpio_set_value(Sb, 0);
        gpio_set_value(Sc, 1);
        gpio_set_value(Sd, 0);
        break;
    case 6:
        gpio_set_value(Sa, 0);
        gpio_set_value(Sb, 1);
        gpio_set_value(Sc, 1);
        gpio_set_value(Sd, 0);
        break;

    default:
        gpio_set_value(Sa, 1);
        gpio_set_value(Sb, 1);
        gpio_set_value(Sc, 1);
        gpio_set_value(Sd, 1);
        break;

    }
```

```
}


int seven_dev_init(void)                                    //Called when the
module is loaded
{
        Major = register_chrdev(0, DEVICE_NAME,
&seven_fops);                   //Registration of a Chardevice


        if (Major < 0) {
                printk(KERN_ALERT "SEVENSEG: Registering char device failed
with %d\n", Major);
                return Major;
        }
        printk(KERN_INFO "SEVENSEG: sevenseg driver loaded with major %d\
n", Major);
        printk(KERN_INFO "SEVENSEG: >> $ mknod /dev/%s c %d 0\n",
DEVICE_NAME, Major);  //Info which node file should be created with which
parameters


        msg = (char *)kmalloc(32, GFP_KERNEL);
//Memory reservation for the transfer from kernel space to user space
        if (msg != NULL)
                printk("malloc allocator address: 0x%p\n", msg);
        return SUCCESS;
}


void seven_dev_exit(void)                                    //Called when the
module is unloaded
{
```

```
	unregister_chrdev(Major, "sevenseg");				//Chardevice is logged off
	printk(KERN_INFO "SEVENSEG: device file closed.\n");


}



static ssize_t seven_write(struct file *filep, const char *buffer, size_t len, loff_t * offset)
{
	short count;


	memset(msg, 0, 32);



	count = copy_from_user(msg, buffer, len);   // Kopieren des Strings aus dem User-Space


	if (msg[0] == '1') {
		seven_status(1);
	}
	else if (msg[0] == '2') {
		seven_status(2);
	}
	else if (msg[0] == '3') {
		seven_status(3);
	}
	else if (msg[0] == '4') {
		seven_status(4);
	}
	else if (msg[0] == '5') {
```

```
        seven_status(5);
    }
    else if (msg[0] == '6') {
        seven_status(6);
    }
    else
    {
        seven_status('default');
    }
    return len;

}

void seven_gpio_init(void) {                    //Initializing The GPIO-Pins  (gpio_request)
    printk(KERN_INFO "SEVENSEG: starting gpio...");
    gpio_request(Sa, "Sa");
    gpio_request(Sb, "Sb");
    gpio_request(Sc, "Sc");
    gpio_request(Sd, "Sd");

    gpio_direction_output(Sa, 0);                   //Specifies that the pin can be addressed as an output pin (gpio_direction_output)
    gpio_direction_output(Sb, 0);
    gpio_direction_output(Sc, 0);
    gpio_direction_output(Sd, 0);
    printk(KERN_INFO "SEVENSEG: starting gpio done.");
}
```

```
void seven_gpio_exit(void) {                                    //When
unloading the LKM, the pins are released again (gpio_free)
        printk(KERN_INFO "SEVENSEG: stopping gpio...");
        gpio_free(Sa);
        gpio_free(Sb);
        gpio_free(Sc);
        gpio_free(Sd);

        printk(KERN_INFO "SEVENSEG: stopping gpio done.");
}



/* Method functions and summary of all individual initialization elements */


static int __init seven_init(void) {
        printk(KERN_INFO "SEVENSEG: starting...");
        seven_gpio_init();
        seven_dev_init();
        printk(KERN_INFO "SEVENSEG: starting done.");
        return 0;
}


static void __exit seven_exit(void) {
        printk(KERN_INFO "SEVENSEG: stopping...");
        seven_gpio_exit();
        seven_dev_exit();
        printk(KERN_INFO "SEVENSEG: stopping done.");
}
```

```
module_init(seven_init);                                    //All functions in
led_init are loaded in the module

module_exit(seven_exit);
```

app.c

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


#define SHELLSCRIPT1 "\
#/bin/bash \n\
echo 1 >> /dev/sevenseg \n\
"

#define SHELLSCRIPT2 "\
#/bin/bash \n\
echo 2 >> /dev/sevenseg \n\
"

#define SHELLSCRIPT3 "\
#/bin/bash \n\
echo 3 >> /dev/sevenseg \n\
"

#define SHELLSCRIPT4 "\
#/bin/bash \n\
echo 4 >> /dev/sevenseg \n\
"

#define SHELLSCRIPT5 "\
#/bin/bash \n\
echo 5 >> /dev/sevenseg \n\
"

#define SHELLSCRIPT6 "\
#/bin/bash \n\
echo 6 >> /dev/sevenseg \n\
"

int main()
{

        int lower, upper, num;

        const int button = 7;
        wiringPiSetup();
        pinMode(button, INPUT);
```

```
        lower =1;
        upper =6;

        while(1){
      if (digitalRead(button) == LOW) {

        num = (rand() % (upper - lower + 1)) + lower;

        switch(num)
        {
                case 1: system(SHELLSCRIPT1);
                        break;
                case 2: system(SHELLSCRIPT2);
                        break;
                case 3: system(SHELLSCRIPT3);
                        break;
                case 4: system(SHELLSCRIPT4);
                        break;
                case 5: system(SHELLSCRIPT5);
                        break;
                case 6: system(SHELLSCRIPT6);
                        break;

        }//switch


        }//while
        };

   return 0;
}
```

Makefile

```
obj-m := 7seg.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

setup.sh

```
make all
sudo insmod 7seg.ko
sudo mknod /dev/sevenseg c 240 0
sudo chmod 777 /dev/sevenseg
gcc app.c -o app -lwiringPi
./app
```