Embedded Des Enabling Robotics
Lab 7 report

Shiyu Wang
Hao Jiang
Fall 2016
10th November

Introduction

The goal of this lab is to control the robotic arm using software running on the ARM microprocessor on the ZedBoard. We sent PWM signals to the Robotic arm using Pmod connector on the ZedBoard. This lab implements knowledge of General Purpose I/O interfacing and PWM signal.

Lab Preparation

We first logged onto the computer using our credentials. We then connected the Zedboard to the PC host via Ethernet to the RJ45 connector. The ZedBoard was boosted after connected to the computer. We made an SSN connection to the ZedBoard where the IP is 192.168.1.10, and port 22.

7.1 - 7.3

We firstly learned how to initiate the GPIO and use provided GPIO_init.sh file to configure the required settings.

```
FILEPATH='/sys/class/gpio'
echo 13 > $FILEPATH/unexport 2>/dev/null     #2>/dev/null redirect stderr to /dev/null
echo 10 > $FILEPATH/unexport 2>/dev/null
echo 11 > $FILEPATH/unexport 2>/dev/null
echo 12 > $FILEPATH/unexport 2>/dev/null
echo 0 > $FILEPATH/unexport 2>/dev/null
echo 13 > $FILEPATH/export
echo 10 > $FILEPATH/export
echo 11 > $FILEPATH/export
echo 12 > $FILEPATH/export
echo 0 > $FILEPATH/export
echo out > $FILEPATH/gpio13/direction
echo out > $FILEPATH/gpio10/direction
echo out > $FILEPATH/gpio11/direction
echo out > $FILEPATH/gpio12/direction
echo out > $FILEPATH/gpio0/direction
```

```
root@localhost:/sys/class/gpio/gpio13# ls
active_low  device  direction  power  subsystem  uevent  value
root@localhost:/sys/class/gpio/gpio13#
```

It basically set the direction of the reserved pin, input and output. Then we read the GPIO pin and write values.

Then we wrote the program servoPos.c that takes two integers to move selected servo to the desired position.

```c
int main()
{

    printf("\n-----------  Generate PWM Signal:  ------------\n\n");

    // file descriptor for 13
        int fdNum;
    printf("Enter a servo number:\n");
    scanf("%d", &fdNum);
    printf("Enter a position between 600 and 2400\n");
    int pos;
    scanf("%d", &pos);
    int fd;
    // open device file on Linux file system
    fd = open_GPIO(fdNum);

    // generate PWM signal with 20ms period and 1.5ms on time
    // generate 400 periods, this will take 20ms * 400 iterations = 8s
    PWM_gen(20000, pos, 400, fd);

    sleep(1); // wait for an additional second

    close(fd);    //Close Linux GPIO interface file before exiting

    return 0;
}
```

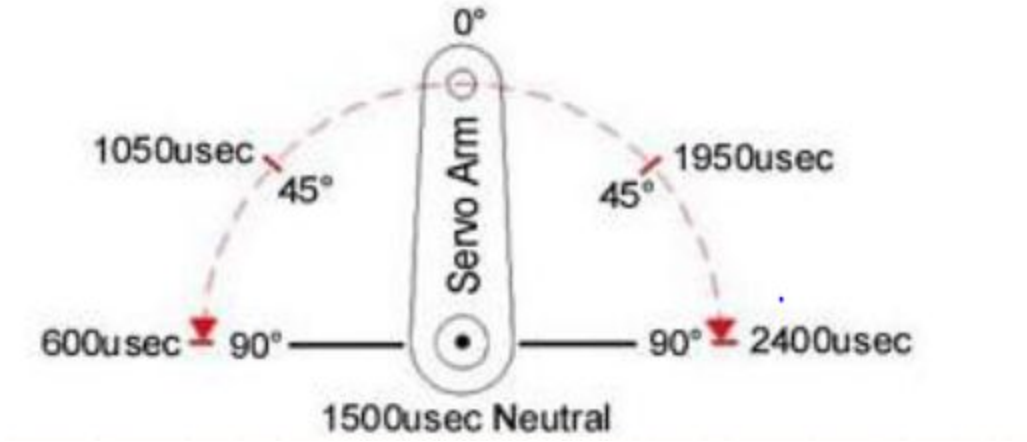
The function asks for two variables: fd is the servo number which is used to select servo.

Table 1. PMOD JE (on ZedBoard) Mapping of robotic arm

PIN Name	PIN Location	Port Number in Linux	PWM Signal for Servo
JE1	Upper JE1	13	Base
JE2	Upper JE2	10	Bicep
JE3	Upper JE3	11	Elbow
JE4	Upper JE4	12	Wrist
JE7	Lower JE1	0	Gripper
JE8	Lower JE2	9	
JE9	Lower JE3	14	
JE10	Lower JE4	15	

This table shows the PWM signal for each servo.

Another variable, “Pos” is used to determine the desired move-to position.



This chart shows how its value relate to angels.

The full code is attached.

Lab 8.4

We made a new program called ervoPosSpeed.c. It basically does 4 things:

a. servo number
b. start position for 4 seconds
c. end position for 4 seconds
d. rotational speed [degree/second]

```c
int main()
{

    printf("\n-----------  Generate PWM Signal:  -------------

    // file descriptor for 13
        int fdNum, pos1, pos2, speed;
    printf("Enter a servo number:\n");
    scanf("%d", &fdNum);

    printf("Enter a start position between 600 and 2400\n");
        scanf("%d", &pos1);

        int fd;
        // open device file on Linux file system
        fd = open_GPIO(fdNum);

    printf("Enter a position between 600 and 2400\n");

    //move to end position by given speed
        printf("Enter a rotational speed (degree/second)\n");
    scanf("%d", &speed);

    printf("Enter an end position between 600 and 2400\n");
    scanf("%d", &pos2);

        //move to start position
    PWM_gen(20000, pos1, 200, fd);

        int n = pos2 - pos1;
    int next;
        if(n >0) {
        next = speed / 50;
            while (n > 0) {
            PWM_gen(2000, pos1 + next, 50, fd);
            next = next + speed /50;
            n = n -speed /50;
        }
    }
    if (n < 0) {
        next = -(speed / 50);
        while (n <0) {
            PWM_gen(2000, pos1 + next, 50, fd);
                        next = next - speed /50;
                        n = n - speed /50;

        }
    }
    else{
        //do nothing;
    }

    //stay for 4 sec at end position
    PWM_gen(2000, pos2, 200, fd);


    sleep(1); // wait for an additional second

    close(fd);    //Close Linux GPIO interface file before exi
```

In the main function, we ask the user to sequentially enter start position, rotational speed and end position. Then we use PWM_gen to make the servo stay at start position for 4 seconds. Then we use end position - start position to find the distance of the move. With while loop, we ask the servo to move step by step based on given speed until it gets to the end position. Then

we stay there for 4 seconds. Btw, staying period: 4 s = generate 200 periods, this will take 20ms * 200 iterations = 4s.

The full code is attached