```
*************readPWM.c***************

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <wiringPi.h>
#include <time.h>
#include <string.h>

void clock_calibrate(int frequency);

void setOSCCAL(int value);

int OSCCAL = 0;

int main(int argc, char *argv[])
{
    int initVal;
    int next_state = 0;
    int get_new_start = 1;
    /* Initializations for using "clock_gettime()" func. to measure signal freq.  */
    struct timespec start, finish;
    double period = 0;

    /* Initialize wiringPi pin numbering scheme and set GPIO 27 (2) as input. */
    wiringPiSetup();
    pinMode(2,INPUT);

    /* Grab the initial value of incoming signal (0 or 1) */
    initVal = digitalRead(2);

    while(1) {
        /* Start timer immediately */
        if (get_new_start == 1) {
            clock_gettime(CLOCK_REALTIME, &start);
            get_new_start = 0;
        }
        /* Conditional for when on a falling or rising edge */
        if (initVal != digitalRead(2)) {
            next_state = 1;
        }
        /* If back at initial value (meaning one period gone by), stop timer */
        if ((initVal == digitalRead(2)) && (next_state == 1)){
            clock_gettime(CLOCK_REALTIME, &finish);
        }
        if (finish.tv_nsec != 0) {
            /* Cast period time to double and scale to seconds. */
            period = ((double)((finish.tv_nsec - start.tv_nsec)));
            period = period / 1000000000;
            /* Negate negative period nuance by adding one second */
            if (period < 0) {
                period += 1;
            }
            else {
                period = period;
            }
            /* Print resulting period and frequency of EVERY period of signal */
            printf("Period = %lf\n", period);
            printf("Frequency = %lf\n", 1/period);
            printf("\n\n");
            /* Reset values for next iteration */
            get_new_start = 1;
            next_state = 0;
            finish.tv_nsec = 0;
        }
    }
return 0;
```

```
}


void clock_calibrate(int frequency) {

    //IF frequency is too low, increase OSCCAL
    if (frequency < 100) {
        OSCCAL = OSCCAL + 1;
    }
    //IF frequency is too high, decrease OSCCAL
    else if (frequency > 100) {
        OSCCAL = OSCCAL - 1;
    }
    //Otherwise, do nothing
    else {
        OSCCAL = OSCCAL;
    }
    //Call setOSCCAL to reset avr registers with new value of OSCCAL
    setOSCCAL(OSCCAL);
}

void setOSCCAL(int value)
{
        FILE *fptr1, *fptr2;
        int lno, linectr = 0;
    char OSCCAL[40] = "  OSCCAL = ";
        char str[256];
    char fname[12] = "avrcode.c";
        char temp[] = "temp.txt";

        fptr1 = fopen(fname, "r");
        if (!fptr1)
        {
                printf("Unable to open the input file!!\n");
                return;
        }

        fptr2 = fopen(temp, "w");
        if (!fptr2)
        {
                printf("Unable to open a temporary file to write!!\n");
                fclose(fptr1);
                return;
        }
        //Replace line 30 with new OSCCAL offset
    lno = 30;
        while (!feof(fptr1))
        {
            strcpy(str, "\0");
            fgets(str, 256, fptr1);
            if (!feof(fptr1)) //IF we havent traversed to the end of avrcode.c
            {
                linectr++; //Increment our line counter
                if (linectr != lno)
                    {
                        fprintf(fptr2, "%s", str); //If this isnt the line we care about, r
ewrite old information
                    }
                    else
                    {
                        fprintf(fptr2, "%s%d;\n", OSCCAL, value); //If the line we specifie
d has been found, rewrite new info in place of old
                    }
                }
        }
        fclose(fptr1); //Close avrcode.c
        fclose(fptr2); //Close temp.txt
        remove(fname);
        rename(temp, fname);
    //Recompile the atmega88pa to use new specified OSCCAL offset
    system("avr-gcc -mmcu=atmega88p avrcode.c");
    system("avr-objcopy -j .text -j .data -O ihex a.out a.hex");
    system("sudo avrdude -C ~/ece477/ece477/lab5/avrdude_gpio.conf -c pi_1 -p atmega88p -U 
flash:w:a.hex:i");
  }
```

```
****************Makefile********************


TARGET = PWM
OBJS = readPWM.o
CFLAGS = -g -Wall -O2
LIBS = -lwiringPi -lpthread

all :$(TARGET)

$(TARGET): $(OBJS)
    $(CC) -o $(TARGET) $(OBJS) $(LIBS)

.PHONY:clean

clean:
    rm -f $(TARGET) $(OBJS) core*
```