<Stay Woke>

CpE 301 Final Project
May 4, 2017

Team # <17>
<Abenezer Namaga>
<Ameera Essaqi>

Abstract – <Stay woke (SW) is an alarm clock with a temperature and heart rate sensor. The results are displayed both on a four character 7-segment display and on the PuTTy terminal.>

1. Introduction

Stay woke is an integrated circuit that operates as a clock, alarm system, temperature sensor and heartbeat detector. The objective of Stay Woke is to have both a clock and an alarm system that also functions as a sleep detector to wake up the user. This is to ensure that the user would not be falling asleep during dangerous or inconvenient times, such as in class or while driving.

The system uses the Heartbeat Module (KY039), the Temperature Humidity Sensor Module (DHT11), the Active Buzzer (KY-012), the Key Switch Module (KY-004), two LEDs, 2 Mini Pushbutton Switches (COM-00097), 7-Segment Quad Digit Display, the FTDI232 and the DC vibrator motor.

The Heartbeat Module detects the user's heart rate. If the user's heart rate is within the average sleeping rate, 40-50 BPM, then they would be alerted by the vibration of the DC motor. The Temperature Humidity Sensor Module is used to detect the temperature of the environment. The Active Buzzer is used to sound the alarm. The buzzer goes off when the time matches the alarm time. The Key Switch Module is used to turn off the alarm once the alarm goes off.

The system uses two LEDs. When the user's heart rate is being read, the red LED lights up at the rate of the user's pulse. The yellow LED lights up when the alarm sounds and turns off when the alarm is turned off.

There are two Mini Pushbutton Switches, one is used to adjust the hour of the displayed clock and the other is used to adjust the minute of the clock. The 7-segment Quad Digit Display is used to display the time and the temperature.

The FTDI232 supplies 5 voltages to the entire circuit. The FTDI232 also provides a serial USART interface for the user, it displays a main menu to the screen to welcome the user and instructs them to select one of the three functions available.

2. Implementation

By pressing ‘1’ on the keyboard it allows the user to set an alarm to which the Passive Buzzer will sound once that time matches the clock time. By pressing ‘2’, the function allows the user to check their heart rate and the DC motor will begin to vibrate if their heart rate is in the sleeping range. By pressing ’3’, the last function allows the user to check the temperature of the environment and then display it on the 7-Segment Display for 5 seconds. At the end of the program, the 7-Segment Display shows the time of decided and allows the alarm to function.

We ran into difficulties when creating our clock with the 7-Segment Quad Digit Display. Since the 7-Segment uses the same input to display different numbers, it was difficult to implement because all of the four segments would display the same numerical input when you would try implementing different numbers. We were able to find a solution by introducing quick delays that could not be easily detected by the human eye. We did this by interchanging the clock speed of the Atmega and the refresh rate of the 7-Segment Display. This creates the illusion that the numbers are persistent; however we are only refreshing the segment by turning the ground pins on and off, while continuously changing the input.

Initially, we had trouble with the temperature and the heartbeat sensor with displaying the proper readings. We had to adjust our formulas in the ADC conversion to match the temperature of the room.

In order to properly get the heartbeat sensor to be accurate we had to understand how it was working. The ADC values were read through the heartbeat module through a low pass filter to get a proper reading. The sensor shines an infrared LED through your finger and gets picked up on the other side by an infrared sensor. The slight changes in light transmittance through your finger are read through the ADC. We averaged sensor reading by subtracting older values from new ones. This helped us come up with user’s heart rate.

In the beginning we had trouble implementing the button and getting the alarm to completely turn off after pressing the button. The alarm would still continue to go after pressing the alarm and we had to implement ways for the code to pick up when the user would press the button, if even for a quick second.

3. Experimental evaluation / Results

For the temperature module, we tested the correct output by being aware of the temperature of the room. This approach definitely had room for error, as we didn't know the exact temperature in the exact place of the module, and even then the module was off about 5 degrees.

When approaching the alarm, we set the alarm to one minute before it was supposed to go off and waited for the alarm go off. Following that we pressed that button to turn off the alarm and saw that it was working. We continued to test the alarm by pressing button at every stage of the alarm and saw that the alarm would turn off no matter where it was in the code.

For the heartbeat we checked the average heartbeat for an adult and compared it to our own. There was room for error in this one due to not having a quick way to test our own heart rate against the module.

4. Conclusion

The goal of the project was to make an alarm and clock that accepted all of its user input through USART by using the PuTTy terminal, and create an alarm that sounded once the user was beginning to fall asleep.

We were able to properly get mostly everything working, but had some trouble setting up the alarm and time on the clock. Due to having problems with USART, we ended up using buttons to implement the time instead.

To improve the project, upgrades made would include utilizing the user input through USART to set the clock time. There is also room to maximize the precision of the temperature module and heartbeat module.

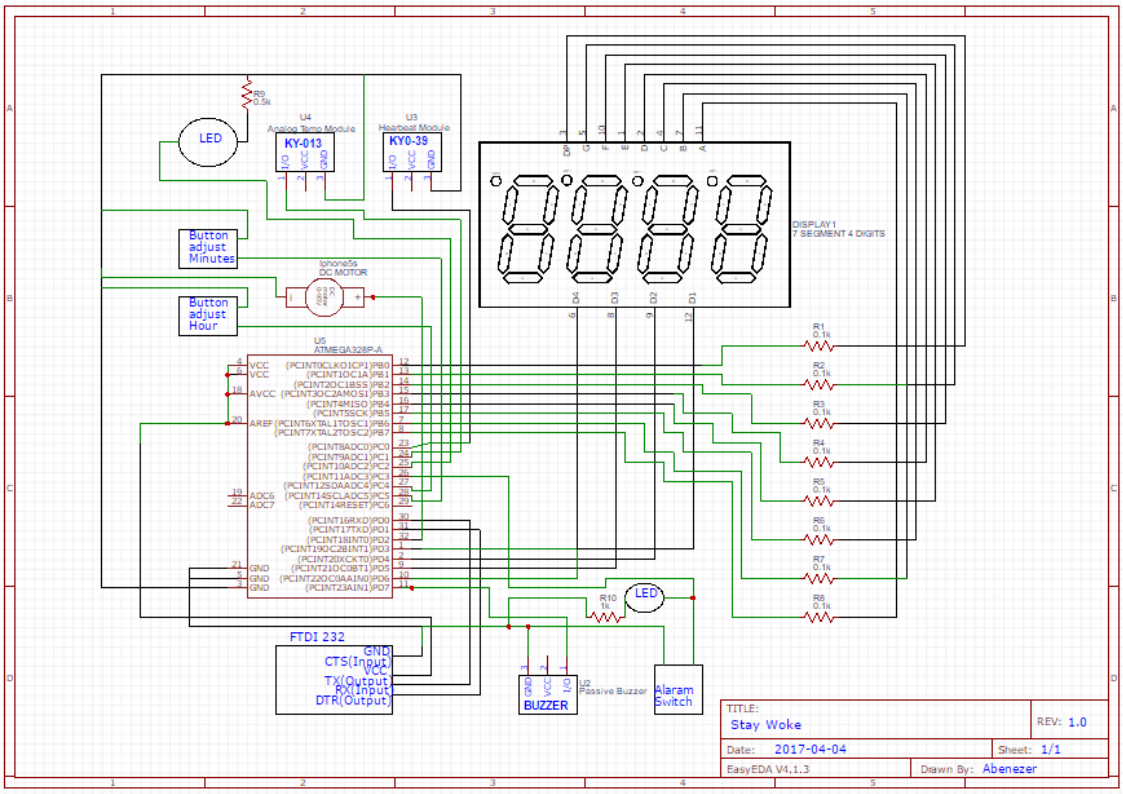
After working on the Stay Woke project, we gained a better understanding on how and when to implement interrupts. We also learned not to get fixated on a single problem for too long, and to tackle one problem at a time.

5. **Appendix** (not included in 5 page limit)

Video Link

- https://youtu.be/Hq6ZpUKLHvQ

- **Detailed Schematics**



- **Bill of Materials**

Final Project: Team 17		4-May-17						
Stay Woke								
Description	**Manufacture**	**Distributor Part Numbe**	**Distributo**	**Web Link**	**Quantity**	**Cost (USD**	**Notes**	**Cost per Kit (USD)**
Buzzer	CEM-1203(42	102-1153-ND	Digikey	http://www.digikey.c	1.00	0.78		0.78
LCD 1X8	EA DOGM081	1481-1063-ND	Digikey	http://www.digikey.c	1.00	10.06		10.06
Breadboard	BB-32621	377-2094-ND	Digikey	http://www.digikey.c	1.00	3.32		3.32
Jumper wires (male-fema	MIKROE-512	1471-1231-ND	Digikey	http://www.digikey.c	0.20	2.7	Kit price: kit include	0.54
Jumper wires (male-male	BC-32625	377-2093-ND	Digikey	http://www.digikey.c	0.31	3.94	Kit price: kit include	1.21
Temperature and Humidity Sensor		2213368	DFRobot	http://www.jameco.c	1.00	5.19		5.19
Heartbeat Sensor		#00903402	KEYES	http://www.miniinth	1.00	1.99		1.99
LEDs	COM-11450			https://www.sparkfu	2.00	0.75		1.50
FTDI232	BOB-12731	1568-1195-ND		https://www.digikey.	1.00	14.95		14.95
							Cost per kit	**39.54**

Code:

```
#include <avr/io.h>
#include <stdlib.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <time.h>
#include <stdbool.h>

#define F_CPU 8000000       // Clock Speed
#include <util/delay.h>
#define BAUD  9600

#define SegOne 0xF7;        //PD3
#define SegTwo 0xEF;        //PD4
#define SegThree 0xDF;      //PD5
#define SegFour 0xBF;       //PD6

volatile char i;
volatile char hours = 8;
volatile char minutes = 55;
volatile char seconds = 0;
unsigned char amorpm;
volatile char alarm_h = 8;
volatile char alarm_m = 55;
volatile char alarm_s = 0;
char clocktime[6];
char alarmtime[6];

unsigned char DigitTo7SegEncoder(unsigned char digit, unsigned char common);
ISR(TIMER1_COMPA_vect);

void initUART();
void inittime();
void getheartbeat();
void gettemperature();
void MainMenu();
void UserInput();
void writeChar(unsigned char c);
void writeChar1();


int main(void)
{
        DDRB = 0xFF; // Set portb pins to output
        DDRD = 0xFB; // Set portd pins to output PD3 - PD6
        PORTD = 0xFB;
```

```
DDRC = 0xF3; //
PORTC = 0xF3;

// Create a delay of 1 second
TCCR1B = (1<<CS12|1<<WGM12);
OCR1A = 31250-1;

//global interrupts are enabled, micro will jump to the compare A
//interrupt vector
TIMSK1 = 1<<OCIE1A;
sei();

while(1)
{
	initUART();
	inittime();
	MainMenu();
	UserInput();

// Set minutes when pin5 button is pressed
	if((PINC & 0x20) == 0 )
	{
		_delay_ms(200);
		if(minutes < 59)
		minutes++;
		else
		{
			minutes = 0;
			hours++;
		}
		int i = 3;
		if(i == 0)
		{
			PORTB = 0x02;
			_delay_ms(200);
			i--;
		}

	}
	// Set Hours when Pin 4 is Pressed
	if((PINC & 0x10) == 0 )
	{
		_delay_ms(200);
		if(hours < 12)
		hours++;
		else
		hours = 1;
	}

		PORTB = DigitTo7SegEncoder(amorpm*10,1);
```

```
                        PORTD = SegFour;
                        PORTB = DigitTo7SegEncoder(minutes%10,1);
                        PORTD = SegFour;
                        PORTB = DigitTo7SegEncoder(minutes/10,1);
                        PORTD = SegThree;
                        PORTB = DigitTo7SegEncoder(hours%10,1);
                        PORTD = SegTwo;
                        PORTB = DigitTo7SegEncoder(hours/10,1);
                        PORTD = SegOne;

                //      getheartbeat();
                        //gettemperature();
        }

        return 0;
}

void initUART() {
        unsigned int baudrate;

        // Set baud rate:  UBRR = [F_CPU/(16*BAUD)] -1
        baudrate = ((F_CPU/16)/BAUD) - 1;
        UBRR0H = (unsigned char) (baudrate >> 8);
        UBRR0L = (unsigned char) baudrate;

        UCSR0B |= (1 << RXEN0) | (1 << TXEN0); // Enable receiver and transmitter
        UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Set data frame: 8 data bits, 1 stop bit,
no parity
}

void inittime(){
        hours = 0;
        seconds = 0;
        minutes = 0;
}
unsigned char DigitTo7SegEncoder(unsigned char digit, unsigned char common)
{
        unsigned char SegVal;

        //Holds segment values to be displayed as decimal digits from 0-9.
        /*

                __
                11(a)
        |10(f)  7(b)|

                ___
                5(g)
        |1(e)    4(c)|

                ____
                2(d)    dp(3)
        */
```

```
	switch(digit)
	{
		case 0:if(common == 1)		SegVal = 0xEB;  // Prints 0
		else				SegVal = ~0xEB;
		break;
		case 1:if(common == 1)		SegVal = 0x28;  // Prints 1
		else				SegVal = ~0x28;
		break;
		case 2:if(common == 1)		SegVal = 0xB3;  // Prints 2
		else				SegVal = ~0xB3;
		break;
		case 3:if(common == 1)		SegVal = 0xBA;  // Prints 3
		else				SegVal = ~0xBA;
		break;
		case 4:if(common == 1)		SegVal = 0x78; // Prints 4
		else				SegVal = ~0x78;
		break;
		case 5:if(common == 1)		SegVal = 0xDA; // Prints 5
		else				SegVal = ~0xDA;
		break;
		case 6:if(common == 1)		SegVal = 0xDB; // Prints 6
		else				SegVal = ~0xDB;
		break;
		case 7:if(common == 1)		SegVal = 0xA8; // Prints 7
		else				SegVal = ~0xA8;
		break;
		case 8:if(common == 1)		SegVal = 0xFB; // Prints 8
		else				SegVal = ~0xFB;
		break;
		case 9:if(common == 1)		SegVal = 0xF8; // Prints 9
		else				SegVal = ~0xF8;
		break;
		case 10: if(common == 1) SegVal = 0x04; // Prints a decimal point
		else				SegVal = ~0x04;
		break;
		case 11: if(common == 1) SegVal = 0xF0; // prints Degree
		else				SegVal = ~0xF0;
		break;
		case 12: if(common == 1) SegVal = 0xAB; // prints B
		else				SegVal = ~0xAB;
		break;
		case 13: if(common == 1) SegVal = 0xF1; // Prints P
		else				SegVal = ~0xF1;
	}
	return SegVal;
}


ISR(TIMER1_COMPA_vect)
{
```

```
while(1)
{
	if(((alarm_h - hours) && (alarm_m - minutes)) == 0)
	{
		DDRC = 0x08;
		PORTC = 0x08;
		DDRD= 0x80;
		PORTD =0x80;

		for(int i=25;i>=0;i--) //if time and while buttion isn't pressed
		{
			PORTD=255;
			_delay_ms(1000);
			_delay_ms(1000);
			PORTD=0x7F;
			_delay_ms(1000);
			_delay_ms(1000);

			if(!(PINC & (1<<3)))
			{
				PORTD=0x7F;
				PORTC = 0;
				break;
			}
		}

		if(!(PINC & (1<<3)))
		{
			PORTD=0x7F;
			break;
		}

		for(int i=25;i>=0;i--) //if time and while button isn't pressed
		{
			PORTD=255;
			_delay_ms(1000);
			PORTD=0x7F;
			_delay_ms(1000);
			if(!(PINC & (1<<3)))
			{
				PORTD=0x7F;
				break;
			}
		}
		if(!(PINC & (1<<3)))
		{
			PORTD=0x7F;
			break;
		}
```

```
                        for(int i=1000;i>=0;i--)
                        {
                                _delay_ms(50);
                                PORTD=255;
                                _delay_ms(50);

                                if(!(PINC & (1<<3)))
                                {
                                        PORTD=0x7F;
                                        break;
                                }
                        }
                        if(!(PINC & (1<<3)))
                        {
                                PORTD=0x7F;
                                break;
                        }

                }
        }

        seconds++;

        if(seconds == 60)
        {
                seconds = 0;
                minutes++;
        }
        if(minutes == 60)
        {

                minutes = 0;
                hours++;
        }
        if(hours > 12)
        hours = 1;
}

getheartbeat()
{
        unsigned int baudrate;
        static int BeatsPerSec = 0;
        int BeatsPerMin = 0;
        int adc_temp,rawvalue;
        char Heart_Rate[8];
        float change = 0.0;
        float alpha = 0.75;
        float maxvalue = 0.0;
```

```
        static float oldvalue = 1009;
        // Set baud rate:  UBRR = [F_CPU/(16*BAUD)]
        baudrate = ((F_CPU/16)/BAUD) - 1;
        UBRR0H = (unsigned char) (baudrate >> 8);
        UBRR0L = (unsigned char) baudrate;
        UCSR0B |= (1 << RXEN0) | (1 << TXEN0); // Enable receiver and transmitter
        UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Set data frame: 8 data bits, 1 stop bit,
no parity

        //for(i=0; i < 40; i++)
        //{
                DDRC = 0x0;    // Make portc an input
                ADCSRA = 0x87; // ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS0);
clk/128
                // ADEN: Set to turn on ADC, by default it is turned off
                ADCSRB = 0x0;  // Free running mode
                ADMUX=0x0;     // Reference = Aref, ADC0 (PC.0) used as analog input
                // data is right-justified

                ADCSRA |= (1<<ADSC); // Start conversion
                while ((ADCSRA & (1<<ADIF)) == 0); // wait for conversion to complete
                adc_temp = ADC;

                change = alpha*oldvalue +  (1 - alpha)* adc_temp;
                rawvalue = abs(change - oldvalue);

                BeatsPerMin = ((rawvalue*BeatsPerSec)/5);
        //}
        if(BeatsPerSec == 60)
        {
                BeatsPerSec == 0;
        }

        BeatsPerSec++;
        int i;
        for(i=0; i <25; i++){

        if(BeatsPerMin < 45)
                {


                        _delay_ms(1000);
                        DDRD = 0;
                        PORTD = 0; //x00;
                }

        if(BeatsPerMin >= maxvalue)
        {
                DDRC = 0x04;
                PORTC = 0x04;
```

```
                DDRD = 0x04;
                        PORTD = 0x04;
                maxvalue = BeatsPerMin;

        //itoa(BeatsPerMin,Heart_Rate,10);
                sprintf(Heart_Rate,"%dBPM",BeatsPerMin);
                int i;
                for(i =0; i<5; i++)   // Run through function
                {
                        if(Heart_Rate[i] != '\0') //Write characters and terminate at null
                        {
                                UDR0 = Heart_Rate[i];
                                _delay_ms(1000);
                        }
                        else
                        break; //Break out of loop
                }
                PORTC = 0x00;
                PORTD = 0x00;
                UDR0= ' '; //Output a space after a string of statements
                _delay_ms(1000);  //Delay for a second
                maxvalue = maxvalue * 0.98;
                }
        }
}

gettemperature()
{
        ADCSRA = ((1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0));// ADC
prescaler 128
         ADMUX = ((1<<REFS1)|(1 << REFS0)|(1 << MUX1));
         int adc_temp; //stores ADC temporarily
         float adc_tempf; //float for calculations
         int adc_tempi; //integer part
         int adc_tempd; //decimal part

         //read ADC
         ADCSRA |= (1<<ADSC); //start conversion
         while((ADCSRA &(1<<ADIF)) == 0); //wait for conversion to finish
         adc_temp = ADC;      //save ADC value

         adc_tempf = (float)adc_temp * (1.1 / 1024) / 0.01; //(ADC*res/.01) (lm34 sf =
10mv/degF)
         adc_tempi = (int)adc_tempf; //integer part
         adc_tempf = adc_tempf - adc_tempi; //remove integer part of float temp
         adc_tempd = (int)(adc_tempf * 100); //store 2 decimal bits as float
         char TmpTemp[7];

         //sprintf here is used to "print" to TempInt string and then dot is added at end
         sprintf(TmpTemp, "%d.%d C", adc_tempi,adc_tempd);
```

```
        //output degrees Fahrenheit
         initUART();
        int i;
        for(i =0; i<8; i++)   // Run through function
        {
                if(TmpTemp[i] != '\0')
                {
                        UDR0 = TmpTemp[i];
                        _delay_ms(100);
                }
                else
                break; //Break out of loop
        }
        UDR0= ' '; //Output a space after a string of statements
        _delay_ms(100);  //Delay for a second
}

void MainMenu()
{


        UDR0=hours;
        UDR0=minutes;
        UDR0=seconds;
        UDR0= amorpm;
        UDR0=alarm_h;
        UDR0= alarm_m;
        UDR0= alarm_s;


        char array[40] = "Welcome to Stay Woke";
        char
blank[200]="
                                                ";
        char menu1[50]= "**Main Menu**";
        char menu2[50]="(1)  set time";
        char menu3[50]="(2)  set alarm";
        char menu4[50]="(3)  check temperature";
        char menu5[50]="(4)  check heart rate";
        char laststar[25]="********************";

        int i;
        for(i=0;i<60;i++)
        {
                UDR0=blank[i];
                _delay_ms(10);
        }

        writeChar1();
```

```
for(i = 0; i <20; i++ )
{
        UDR0 = array[i];
        _delay_ms(25);
}
writeChar1();

for(i=0;i<169;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}
UDR0='*';

for(i=0;i<27;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

for(i=0;i<50;i++)
{
        UDR0=menu1[i];
        _delay_ms(10);
}

for(i=0;i<26;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<169;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<26;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

for(i=0;i<50;i++)
{
```

```
        UDR0=menu2[i];
        _delay_ms(10);
}


for(i=0;i<27;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<169;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<26;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

for(i=0;i<50;i++)
{
        UDR0=menu3[i];
        _delay_ms(10);
}

for(i=0;i<26;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<169;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<26;i++)
```

```
{
        UDR0=blank[i];
        _delay_ms(10);
}


for(i=0;i<50;i++)
{
        UDR0=menu4[i];
        _delay_ms(10);
}


for(i=0;i<18;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<169;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<26;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

for(i=0;i<50;i++)
{
        UDR0=menu5[i];
        _delay_ms(10);
}

for(i=0;i<19;i++)
{
        UDR0=blank[i];
        _delay_ms(10);
}

UDR0='*';

for(i=0;i<169;i++)
```

```
        {
                UDR0=blank[i];
                _delay_ms(10);
        }

        writeChar1() ;
        writeChar1() ;
        for(i=0;i<25;i++)
        {
                UDR0=laststar[i];
                _delay_ms(10);
        }

        for(i=0;i<169;i++)
        {
                UDR0=blank[i];
                _delay_ms(10);
        }

}

void updatetime()
{

        bool morningtime=0;

        if(clocktime[5]=='a')
        morningtime='1';

        char ch1[2];
        char cm1[2];


        ch1[0]=clocktime[0];
   ch1[1]=clocktime[1];
        cm1[0]=clocktime[2];
        cm1[1]=clocktime[3];

        int ch=atoi(ch1);
        int cm=atoi(cm1);


        hours=ch;
        minutes=cm;
        seconds=0;

        //UDR0=hours;
        //UDR0=minutes;
}
```

```
void updatealarm()
{
		bool morningalarm=0;

		if(alarmtime[5]=='a')
		morningalarm=1;


		char ah1[2];
		char am1[2];

		int ah;
		int am;
		ah1[0]=alarmtime[0];
		ah1[1]=alarmtime[1];
		am1[0]=alarmtime[3];
		am1[1]=alarmtime[4];

		ah=atoi(ah1);
		am=atoi(am1);

		alarm_h=ah;
		alarm_m=am;
		alarm_s=0;

}

void UserInput()
{
	initUART();
  char
blank[200]="
                                                ";
	unsigned char ch;
	while(1)
	{
		while(UCSR0A & (1<<RXC0))
		{
			int x;
			ch=UDR0;

			if(ch=='1')
			{
				UDR0=ch;
				int i;
				for(i=0;i<5;i++)
				{
					UDR0=blank[i];
					_delay_ms(5);
				}
```

```
                                        char array1[86]=" Please enter current time in the format
hh:mm am/pm. For example: 03:10pm or 05:20am ";
                                        for(x=0;x<86;x++)
                                        {
                                                UDR0=array1[x];
                                                _delay_ms(50);
                                        }

                                        i=0;
                                while(ch!='m')
                                {
                                        while(UCSR0A & (1<<RXC0))
                                        {
                                                ch=UDR0;

        if(ch=='0'||ch=='1'||ch=='2'||ch=='3'||ch=='4'||ch=='5'||ch=='6'||ch=='7'||ch=='8'||ch=='9'||ch
==':'||ch=='a'||ch=='p')
                                                {
                                                        UDR0=ch;
                                                        clocktime[i]=ch;
                                                        i++;
                                                }

                                        }
                                }
                                        updatetime();
                        }
                        if(ch=='2')
                        {
                                UDR0=ch;
                                int i;

                                for(i=0;i<5;i++)
                                {
                                        UDR0=blank[i];
                                        _delay_ms(5);
                                }
                                char array2[150]=" Please enter the time you would like to set your
alarm in the format hh:mm am/pm. For example: 03:10pm or 05:20am ";
                                        int x;
                                        for(x=0;x<150;x++)
                                        {
                                                UDR0=array2[x];
                                                _delay_ms(50);
                                        }

                                        i=0;
                                        while(ch!='m')
                                        {
                                                while(UCSR0A & (1<<RXC0))
```

```
                                        {
                                                ch=UDR0;

        if(ch=='0'||ch=='1'||ch=='2'||ch=='3'||ch=='4'||ch=='5'||ch=='6'||ch=='7'||ch=='8'||ch=='9'||ch
==':'||ch=='a'||ch=='p')
                                                {
                                                        UDR0=ch;
                                                        alarmtime[i]=ch;
                                                        i++;
                                                }

                                        }
                                }
                                updatealarm();
                                _delay_ms(100);
                        }

                        if(ch=='3')
                        {
                                UDR0=ch;
                                int i;
                                        for(i=0;i<5;i++)
                                        {
                                                UDR0=blank[i];
                                                _delay_ms(5);
                                        }
                                char array3[50]=" The current temperature is: ";
                                int x;
                                for(x=0;x<29;x++)
                                {
                                        UDR0=array3[x];
                                        _delay_ms(50);
                                }

                                gettemperature();
                        }

                        if(ch=='4')
                        {

                                UDR0=ch;
                                int i;
                                        for(i=0;i<5;i++)
                                        {
                                                UDR0=blank[i];
                                                _delay_ms(5);
                                        }
                                        char array3[93]=" Please place your finger on the heart
rate monitor...10...9...8...7...6...5...4...3...2...1 ";
                                        char array4[21]=" Your heart rate is: ";
```

```
                                        int x;
                                        for(x=0;x<51;x++)
                                        {
                                                UDR0=array3[x];
                                                _delay_ms(100);
                                        }
                                        for(x=51;x<92;x++)
                                        {
                                        UDR0=array3[x];
                                        _delay_ms(300);
                                        }
                                        for(x=0;x<21;x++)
                                        {
                                                UDR0=array4[x];
                                                _delay_ms(50);
                                        }
                                        getheartbeat();
                                }

                        }
                }
        }

void writeChar1()
{
        int filler = '*'; /* setfill('#') */
        int width = 25;   /* setw(10)     */
        int target = 1;

        int s = snprintf(NULL, 0, "%d", target);
        int i;
        for(i = 0; i < width - s; i++)
        {

                UDR0 = filler; // printf("%d", filler);
                _delay_ms(10);

        }
}

void writeChar(unsigned char value)
{
        UDR0 = value;// Display character on serial (i.e., PuTTY) terminal
        _delay_ms(100); // Delay for a 100ms
}
```