```
#define F_CPU 8000000UL

#include <avr/   io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define RW 5
#define Enable 6
#define RS 1
#define DA 2
#define DHT11 4

volatile char val;
volatile int timer;
volatile int newChoice;
volatile int seconds = 5;
volatile int minutes = 33;
volatile int hours = 12;
volatile uint8_t data = 0;
volatile uint8_t humidity_H;
volatile uint8_t humidity_D;
volatile uint8_t temp_H;
volatile uint8_t temp_D;
volatile uint8_t checkSum;
volatile uint8_t toggleTH;
volatile uint8_t toggleMM;
volatile int limitMaxTemp = 40;
volatile int limitMinTemp = 0;
volatile int limitMaxHum = 40;
volatile int limitMinHum = 0;
volatile int maxTemp;
volatile int minTemp;
volatile int maxHum;
```

```
volatile int minHum;
volatile int tempBool = 0;

volatile int limitBool = 0;
volatile int limitBool2 = 0;
volatile int highSelect = 0;
volatile int minSelect = 0;

volatile int toggleClk = 0;
volatile int toggleClkHM = 0;
volatile int onOrOff = 0;

void SendCmd(unsigned char command) { //Send commands to display
        PORTD |= (1<<Enable);
        PORTD &= ~ (1 << RW | 1 << RS);
        PORTB = command;
        _delay_ms(5);
        PORTD &= ~ (1<<Enable);
        PORTB = 0;
}

void SendData(unsigned char data) { //Send data to display(Display characters)
        PORTD |= (1<<Enable);
        PORTD &= ~ (1 << RW);
        PORTD |=   (1 << RS);
        PORTB = data;
        _delay_ms(5);
        PORTD &= ~ (1<<Enable);
        PORTB = 0;
}

void clearDisplay(){
        SendCmd(0x01);
```

```
	_delay_ms(5);
}

void displayOn(){
	SendCmd(0b00001100);
	clearDisplay();
	SendCmd(0x3C);
}

void displayOff(){
	SendCmd(0b00001000);
}

void SendText(char text[]){ //Display text
	int i = 0;
	while(text[i] != '\0'){
		SendData(text[i]);
		i++;
	}
}

void SendNbr(int nbr){  //Display numbers
	char talet[sizeof(nbr) * 4 + 1];
	sprintf(talet, "%d", nbr);
	SendText(talet);
}

void initTimer() {
	TCCR1B |= (1 << WGM12); // Configure timer 1 for CTC mode
	TIMSK1 |= (1 << OCIE1A); // Enable CTC interrupt

	OCR1A = 31249;// Set CTC compare value to 1Hz at 8 MHz AVR clock, with a prescaler of 256
	TCCR1B |= (1 << CS12); // Start timer at Fcpu/256
```

```
	sei();
}

void startSensor(){
	//REQUEST
	DDRD |= (1<<DHT11); //DHT11 PIN to OUTPUT
	PORTD &= ~(1 << DHT11); //Set pin to LOW
	_delay_ms(25); //At least 18ms delay
	PORTD |= (1 << DHT11); //Set pin to HIGH

	//RESPONSE
	DDRD &= ~(1 << DHT11); //DHT11 PIN to INPUT
	while(PIND & (1<<DHT11)); //Checks DHT11 PIN, WAIT UNTIL SIGNAL GETS LOW
	while((PIND & (1<<DHT11))==0); //Checks DHT11 PIN, WAIT UNTIL SIGNAL GETS HIGH
	while(PIND & (1<<DHT11));//Checks DHT11 PIN, WAIT UNTIL SIGNAL GETS LOW

}

uint8_t receiveData(){
	for(int i = 1; i <= 8; i++){  //8 bits at a time
		while((PIND & (1<<DHT11))==0); //Wait until DHT11 is high
		_delay_us(30);
		if(PIND & (1<<DHT11)){ //If DHT11 is still high after delay, data bit is '1'
			data = (data<<1)|(0b00000001);
		}
		else { //data bit is '0'
			data = (data<<1);
		}
		while(PIND & (1<<DHT11)); //Wait until DHT11 is low, starting to transmit new bit
	}
	return data;
}
```

```
void measure(){ //Receive value of temperature and humidity
	humidity_H = receiveData();
	humidity_D = receiveData();
	temp_H = receiveData();
	temp_D = receiveData();
	checkSum = receiveData();
}

void initPins(){
	DDRB = 0b11111111; // pins B to output
	DDRD |= (1<<Enable) | (1<<RW) | (1<<RS);
	DDRA = 0b00000011;
}

void printTH(){ //Print temperature and humidity
	startSensor();
	measure();
	if((humidity_D + humidity_H + temp_D + temp_H) != checkSum){
		//Error, wrong value
	}
	else{
		SendCmd(0b10000000);
		SendText("T:");
		SendNbr(temp_H);
		SendText(" C");
		SendCmd(0b10001001);
		SendText("H:");
		SendNbr(humidity_H);
		SendText(" %");
	}
}
```

```
void calcTime() {
	if (timer == 1) { //Timer gets the value "1" through the interrupt that occurs every second
		seconds++;
		hours %= 24;
		minutes %= 60;
		if (seconds == 60) {
			minutes++;
			seconds = 0;
		}
		if (minutes == 60) {
			hours++;
			minutes = 0;
		}
		if (seconds < 10) {
			SendCmd(0b10101110);
			SendNbr(0);
			SendNbr(seconds);
		}
		if (minutes < 10) {
			SendCmd(0b10101011);
			SendNbr(0);
			SendNbr(minutes);
			SendText(":");
		}
		if (hours < 10) {
			SendCmd(0b10101000);
			SendNbr(0);
			SendNbr(hours);
			SendText(":");
		}
		if (seconds >= 10) {
			SendCmd(0b10101110);
			SendNbr(seconds);
```

```
                }
                if (minutes >= 10) {
                        SendCmd(0b10101011);
                        SendNbr(minutes);
                        SendText(":");
                }
                if (hours >= 10) {
                        SendCmd(0b10101000);
                        SendNbr(hours);
                        SendText(":");
                }
                printTH(); //Print temperature and humidity every second.
                timer = 0;
                SendCmd(0b10000000);
        }
}

void displayLimit(){ //Display the limits
        if(toggleTH == 1){ //Temperature selected
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Limit Temperature");
                SendCmd(0b10101000);
                SendText("Max:");
                SendNbr(limitMaxTemp);
                SendCmd(0b10110001);
                SendText("Min:");
                SendNbr(limitMinTemp);
                _delay_ms(1000);
                clearDisplay();
        }
        else if(toggleTH == 0){ //Humidity selected
                clearDisplay();
```

```
                SendCmd(0b10000000);
                SendText("Limit Humidity");
                SendCmd(0b10101000);
                SendText("Max:");
                SendNbr(limitMaxHum);
                SendCmd(0b10110001);
                SendText(" Min:");
                SendNbr(limitMinHum);
                _delay_ms(1000);
                clearDisplay();
        }
}

void calcLimit(int amount){ //Increase or decrease the limits
        if(toggleClk == 1){ //If "Adjust time" button is pressed
                if(toggleClkHM == 1) { //Hours selected
                        if((hours + amount) < 0){
                                hours = 24 + (hours + amount);
                                seconds = -1; // Seconds++ before time is calculated in calcTime()
                        }
                        else{
                                hours+= amount;
                                seconds = -1;
                        }
                }
                else if(toggleClkHM == 0) { //Minutes selected
                        if((minutes + amount) < 0){
                                minutes = 60 + (minutes + amount);
                                seconds = -1;
                        }
                        else{
                                minutes+= amount;
                                seconds = -1;
```

```
                }
            }
        }
        else if(toggleTH == 1 && toggleMM == 1){ //Temperature and max limit selected
            limitMaxTemp += amount;
            clearDisplay();
            SendCmd(0b10000000);
            SendText("Limit Max T: ");
            SendNbr(limitMaxTemp);
            _delay_ms(1000);
            clearDisplay();
        }
        else if(toggleTH == 1 && toggleMM == 0){ //Temperature and min limit selected
            limitMinTemp += amount;
            clearDisplay();
            SendCmd(0b10000000);
            SendText("Limit Min T: ");
            SendNbr(limitMinTemp);
            _delay_ms(1000);
            clearDisplay();
        }
        else if(toggleTH == 0 && toggleMM == 1){ //Humidity and max limit selected
            limitMaxHum += amount;
            clearDisplay();
            SendCmd(0b10000000);
            SendText("Limit Max H: ");
            SendNbr(limitMaxHum);
            _delay_ms(1000);
            clearDisplay();
        }
        else if(toggleTH == 0 && toggleMM == 0){ //Humidity and min limit selected
            limitMinHum += amount;
            clearDisplay();
```

```
                SendCmd(0b10000000);
                SendText("Limit Min H: ");
                SendNbr(limitMinHum);
                _delay_ms(1000);
                clearDisplay();
        }
}

void limitCheck(){
        //A bit complicated, two LEDS to show if max/min limits of temperature/humidity are exceeded.
        //Red LED shows the state of the MAX limit, Green LED shows the state of the MIN limit.

        if(limitBool == 0){ //Code for handling MAX limit. Warn ONE time if you exceed a limit
        if(temp_H > limitMaxTemp && humidity_H > limitMaxHum){
                limitBool = 1;
                highSelect = 1; //1 = combination of temp&hum over both Max limits
                PORTA |= (1<<PINA0);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("HIGH TEMP & HUM");
                _delay_ms(1000);
                clearDisplay();
        }
        else if(temp_H > limitMaxTemp){
                limitBool = 1;
                highSelect = 2; //2 = temp over MAX limit
                PORTA |= (1<<PINA0);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
```

```
                SendText("HIGH TEMP");
                _delay_ms(1000);
                clearDisplay();
        }
        else if(humidity_H > limitMaxHum){
                limitBool = 1;
                highSelect = 3; //3 = humidity over MAX limit
                PORTA |= (1<<PINA0);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("HIGH Humidity");
                _delay_ms(1000);
                clearDisplay();
        }
                else{
                        PORTA &= ~(1<<PINA0); //TURN OFF RED LIGHT
                }
}

        if(limitBool == 1) { //If MAX limit has been exceeded
                if(highSelect == 1 && temp_H < limitMaxTemp && humidity_H < limitMaxHum){
//If temp&hum was over MAX limits, but now both are under the MAX.
                        limitBool = 0; //CAN WARN AGAIN
                }
                else if(highSelect == 1 && temp_H > limitMaxTemp && humidity_H < limitMaxHum)
{
 //If temp&hum was over MAX limits, but now humidity is under the MAX
                        limitBool = 0;
                }
                else if(highSelect == 1 && temp_H < limitMaxTemp && humidity_H > limitMaxHum)
{
```

```
//If temp&hum was over MAX limits, but now temp is under the MAX
                        limitBool = 0;
                }
                else if(highSelect == 2 && temp_H < limitMaxTemp) {
 //If temp was over MAX limit, but now temp is under the MAX
                        limitBool = 0;
                }
                else if(highSelect == 2 && temp_H > limitMaxTemp && humidity_H > limitMaxHum)
{
//If temp was over MAX limit, but now humidity is also over MAX
                        limitBool = 0;
                }
                else if(highSelect == 3 && humidity_H < limitMaxHum){
 //If humidity was over MAX limit, but now humidity is under MAX
                        limitBool = 0;
                }
                else if(highSelect == 3 && humidity_H > limitMaxHum && temp_H  >
limitMaxTemp){
//If humidity was over MAX limit, but now temp is also over MAX
                        limitBool = 0;
                }
        }

        if(limitBool2 == 0){ //Code for handling MIN limit. Warn ONE time if you exceed a limit
        if(temp_H < limitMinTemp && humidity_H < limitMinHum){
                limitBool2 = 1;
                minSelect = 1;
                PORTA |= (1<<PINA1);
                clearDisplay();
                SendCmd(0b10000000);
                SendText("WARNING");
                SendCmd(0b10101000);
                SendText("LOW TEMP & HUM");
```

```
            _delay_ms(1000);
            clearDisplay();
    }
    else if(temp_H < limitMinTemp){
            limitBool2 = 1;
            minSelect = 2;
            PORTA |= (1<<PINA1);
            clearDisplay();
            SendCmd(0b10000000);
            SendText("WARNING");
            SendCmd(0b10101000);
            SendText("LOW TEMP");
            _delay_ms(1000);
            clearDisplay();
    }
    else if(humidity_H < limitMinHum){
            limitBool2 = 1;
            minSelect = 3;
            PORTA |= (1<<PINA1);
            clearDisplay();
            SendCmd(0b10000000);
            SendText("WARNING");
            SendCmd(0b10101000);
            SendText("LOW HUMIDITY");
            _delay_ms(1000);
            clearDisplay();
    }
    else{
            PORTA &= ~(1<<PINA1); //Turn OFF Green light
    }
    }
            if(limitBool2 == 1) {
```

```
                        if(minSelect == 1 && temp_H > limitMinTemp && humidity_H >
limitMinHum){
                                limitBool2 = 0;
                        }
                        else if(minSelect == 1 && temp_H > limitMinTemp && humidity_H <
limitMinHum) {
                                limitBool2 = 0;
                        }
                        else if(minSelect == 1 && temp_H < limitMinTemp && humidity_H >
limitMinHum) {
                                limitBool2 = 0;
                        }
                        else if(minSelect == 2 && temp_H > limitMinTemp){
                                limitBool2 = 0;
                        }
                        else if(minSelect == 2 && temp_H < limitMinTemp && humidity_H <
limitMinHum) {
                                limitBool2 = 0;
                        }
                        else if(minSelect == 3 && humidity_H > limitMinHum){
                                limitBool2 = 0;
                        }
                        else if(minSelect == 3 && humidity_H < limitMinHum && temp_H <
limitMinTemp){
                                limitBool2 = 0;
                        }
                }
}

void calcTempHum(uint8_t temp, uint8_t hum){
//Calculate the highest & lowest temperature and humidity measured
        if(tempBool == 0 && temp != 0){ //First measured temp and hum gets set as max and min
                tempBool = 1;
```

```
                maxTemp = temp;
                minTemp = temp;
                maxHum = hum;
                minHum = hum;
        }
        if(temp > maxTemp){
                maxTemp = temp;
        }
        if(temp < minTemp){
                minTemp = temp;
        }
        if(hum > maxHum){
                maxHum = hum;
        }
        if(hum < minHum){
                minHum = hum;
        }
}

ISR(TIMER1_COMPA_vect) //Interrupt every second
{
        timer = 1;

}

ISR(INT0_vect) //Keypad interrupt
{
        val= PINA & 0b11110000;
        newChoice = 1;
}

void keyConfig()
{
```

```
	EICRA |= (1 << ISC00) | (1 << ISC01);
	EIMSK |= (1 << INT0);
	DDRD &= ~(1 << DA);
}

int main(void) {
	initPins();
	displayOff();
	sei();
	initTimer();
	keyConfig();

	while (1) {
		calcTime();
		limitCheck();
		calcTempHum(temp_H, humidity_H);
		if (newChoice == 1) {
			if (val == 0xc0) { //1 DISPLAY ON
				if (onOrOff == 0) {
					onOrOff++;
					displayOn();
					} else {
					onOrOff--;
					displayOff();
				}
			}
			if (val == 0x40) { // 2 //Toggle between hours and minutes(clock)
				if (toggleClkHM == 0) {
					toggleClkHM++;
					clearDisplay();
					SendCmd(0b10000000);
					SendText("Timmar vald");
					_delay_ms(1000);
```

```
                clearDisplay();
                } else {
                toggleClkHM--;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Minuter vald");
                _delay_ms(1000);
                clearDisplay();
            }

        }
        if (val == 0x80) { // 3 //Press to be able to change the clock, press again when
done.
            if (toggleClk == 0) {
                toggleClk++;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Justera klockan");
                _delay_ms(1000);
                clearDisplay();
                } else {
                toggleClk--;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("Klar!");
                _delay_ms(1000);
                clearDisplay();
            }
        }
        if (val == 0x00) { // 4 Toggle between Temperature and Humidity
            if (toggleTH == 0) {
                toggleTH++;
                clearDisplay();
```

```
                SendCmd(0b10000000);
                SendText("TEMPERATUR VALD ");
                _delay_ms(1000);
                clearDisplay();
                } else {
                toggleTH--;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("FUKTIGHET VALD ");
                _delay_ms(1000);
                clearDisplay();
        }
}
if (val == 0xe0) { // 5 Increase limit by +1
        calcLimit(1);
}
if (val == 0x60) { // 6 Increase limit by +5
        calcLimit(5);
}
if (val == 0xa0) { // 7 Increase Limit by +10
        calcLimit(10);
}
if (val == 0x20) { // 8 Toggle between Max Limit and Min Limit
        if (toggleMM == 0) {
                toggleMM++;
                clearDisplay();
                SendCmd(0b10000000);
                SendText("MAX VALD ");
                _delay_ms(1000);
                clearDisplay();
                } else {
                toggleMM--;
                clearDisplay();
```

```
                SendCmd(0b10000000);
                SendText("MIN VALD ");
                _delay_ms(1000);
                clearDisplay();
            }
        }
        if (val == 0xd0) { // 9 Decrease limit by -1
            calcLimit(-1);
        }
        if (val == 0x50) { // 10 Decrease limit by -5
            calcLimit(-5);
        }
        if (val == 0x90) { // 11 Decrease limit by -10
            calcLimit(-10);
        }
        if (val == 0x10) { // 12 Show Max/Min limit without changing it
            displayLimit();
        }
        if (val == 0xf0) { // 13 Show the highest measured humidity
            clearDisplay();
            SendCmd(0b10000000);
            SendText("MaxHum Measured:");
            SendCmd(0b10101000);
            SendNbr(maxHum);
            SendText(" %");
            _delay_ms(1000);
            clearDisplay();
        }
        if (val == 0x70) { // 14 Show the lowest measured humidity
            clearDisplay();
            SendCmd(0b10000000);
            SendText("MinHum Measured");
            SendCmd(0b10101000);
```

```
                    SendNbr(minHum);
                    SendText(" %");
                    _delay_ms(1000);
                    clearDisplay();
                }
                if (val == 0xb0) { // 15 Show the highest measured temperature
                    clearDisplay();
                    SendCmd(0b10000000);
                    SendText("MaxTemp Measured");
                    SendCmd(0b10101000);
                    SendNbr(maxTemp);
                    SendText(" C");
                    _delay_ms(1000);
                    clearDisplay();
                }
                if (val == 0x30) { // 16 Show the lowest measured temperature
                    clearDisplay();
                    SendCmd(0b10000000);
                    SendText("MinTemp Measured");
                    SendCmd(0b10101000);
                    SendNbr(minTemp);
                    SendText(" C");
                    _delay_ms(1000);
                    clearDisplay();
                }
                newChoice = 0;
            }
        }
    }
```