```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>

#define PD4 4
volatile int opti_temp;
volatile int opti_hum;
volatile int opti_jord;
volatile int zone;
volatile int hum = 0;
volatile int nyhum;
volatile int sekunder = 0;
volatile int timmar = 0;
uint8_t RH_high, RH_low, temp_high, temp_low, checksum, dataByte = 0;
uint16_t tempDHT, humDHT;
// glöm inte optimization -01 för att delayen ska bli
precis!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!
int main(void)
{


      void start_signal()
      {
            DDRD |= (1<<PD4);                        //PB4 som output/input
(DHT22 data line)
            PORTD &= ~(1<<PD4);                      //first send low pulse
            _delay_us(1000);                 //for 1ms
            PORTD |= (1<<PD4);                       //then send high pulse
            _delay_us(40);                   //for 40us
      }

      void response_signal()
      {
            DDRD &= ~(1<<PD4);                       //pin PD4 as i/p
            while(PIND & (1<<PD4));                  //vänta på DHT22 low
pulse
            while((PIND & (1<<PD4))==0);             //vänta på DHT22 high
pulse
            while(PIND & (1<<PD4));                  //vänta på DHT22 low
pulse
      }




      void ADCenable()
      {
            ADCSRA |= (1 << ADEN ) | (1 << ADPS0 ) | (1 << ADPS1) | (1
<< ADPS2);
```

```
            // sätter prescaler till 128 bit (högsta upplösning)

            // väljer kanal (ADC5 eller PA 5 och sätter Vcc till 01 )
och most significant i ADCH9
            ADMUX |= (0 << MUX4) | (0 << MUX3) | (1 <<MUX2) | (0 <<
MUX1) | (1 << MUX0) | (0 << REFS1) | (1 << REFS0) | (0 << ADLAR);
      }

      uint16_t ADC_READ()
      {
            ADCSRA |= (1 << ADSC); // starta conversion
            while(ADCSRA & (1 << ADSC));
            return ADC;

      }

      // Write A
      void write(uint8_t k)
      {
            PORTD = 0x05;
            PORTB = k;
            PORTD = 0x01;
            PORTD = 0x05;
      }

      void clear()
      {
            PORTD = 0x04;
            PORTB = 0x01;

            PORTD = 0x00;
            PORTD = 0x04;
            _delay_ms(2);
      }

      // function set
      void functionSet()
      {
            PORTD = 0x04;
            _delay_ms(2);
            PORTB = 0x38;
            PORTD = 0x00;
            _delay_ms(2);
            PORTD = 0x04;

      }


      // Write A
      void writeGAFFEL(uint8_t k)
```

```
{
	// om 20 ska printa 2a och sen 0a.
	int tiotal = k/10;
	int ental = k % 10;

	PORTD = 0x05;
	_delay_ms(2);

	PORTB = (tiotal+48);  //  0b0011 tiotal
	_delay_ms(2);
	PORTD = 0x01;
	PORTD = 0x05;
	PORTD = 0x05;
	_delay_ms(2);
	PORTB = (ental+48);
	_delay_ms(2);
	PORTD = 0x01;
}

void writeDHT(uint16_t k)
{
	// om 20 ska printa 2a och sen 0a.

	int tiotal = k/10;
	int ental = k% 10;
	int decimalH = k;
	int decimalL = k;
	PORTD = 0x05;
	_delay_ms(2);

	PORTB = (decimalL + 48);  //  tusental
	_delay_ms(2);
	PORTD = 0x01;
	PORTD = 0x05;

	PORTD = 0x05;
	_delay_ms(2);

	PORTB = (decimalH + 48);  //  hundratal
	_delay_ms(2);
	PORTD = 0x01;
	PORTD = 0x05;
	// printa punkt
	_delay_ms(2);

	PORTB = (0x2E);  //  hundratal
	_delay_ms(2);
	PORTD = 0x01;
	PORTD = 0x05;

	PORTD = 0x05;
```

```
            _delay_ms(2);

            PORTB = (ental + 48);  //  0b0011 tiotal
            _delay_ms(2);
            PORTD = 0x01;
            PORTD = 0x05;
            PORTD = 0x05;
            _delay_ms(2);
            PORTB = (tiotal + 48);
            _delay_ms(2);
            PORTD = 0x01;
      }
      void registerOn()
      {
            DDRB = 0xFF; // sätter igång registerna jag behöver
            DDRD = 0x07;
      }
      void displayON()
      {
            PORTD = 0x04; // sätt E till 1
            _delay_ms(2);
            PORTB = 0x0F; //
            PORTD = 0x00;
            _delay_ms(2);
            PORTD = 0x04;
      }
      void writeTEMPERATURE()
      {
            _delay_ms(100);
            write(0x54);     // T
            _delay_ms(1);
            write(0x45);      // E
            _delay_ms(1);
            write(0x4D);      // M
            _delay_ms(1);
            write(0x50);      // P
            _delay_ms(1);
//          write(0x45);      // E
//          _delay_ms(1);
//          write(0x52);     // R
//          _delay_ms(1);
//          write(0x41);     // A
//          _delay_ms(1);
//          write(0x54);      // T
//          _delay_ms(1);
//          write(0x55);      // U
//          _delay_ms(1);
//          write(0x52);     // R
//          _delay_ms(1);
//          write(0x45);     //E
      }
```

```
	void writeHUMIDITY()
	{
		_delay_ms(100);
		write(0x48);			//H
		_delay_ms(1);
		write(0x55);			//U
		_delay_ms(1);
		write(0x4D);			//M
		_delay_ms(1);
//		write(0x49);			//I
//		_delay_ms(1);
//		write(0x44);			//D
//		_delay_ms(1);
//		write(0x49);			//I
//		_delay_ms(1);
//		write(0x54);			//T
//		_delay_ms(1);
//		write(0x59);			//Y
	}

	void writeDAY()
	{
		_delay_ms(1);
		write(0x44);			//D
		_delay_ms(1);
		write(0x41);			//A
		_delay_ms(1);
		write(0x59);			//Y
	}

	void writeNIGHT()
	{
		_delay_ms(1);
		write(0x4E);		//N
		_delay_ms(1);
		write(0x49);		//I
		_delay_ms(1);
		write(0x47);		//G
		_delay_ms(1);
		write(0x48);		//H
		_delay_ms(1);
		write(0x54);		//T
	}

	void writeMODE()
	{
		_delay_ms(100);
		write(0x4D);		//M
		_delay_ms(1);
```

```
            write(0x4F);           //O
            _delay_ms(1);
            write(0x44);           //D
            _delay_ms(1);
            write(0x45);        // E
            _delay_ms(1);
            write(0x3A);     // :
      }

      void setDataRam(uint8_t d)
      {
            PORTD = 0x04;
            PORTB = d;
            _delay_ms(2);
            PORTD = 0x00;
            _delay_ms(2);
            PORTD = 0x04;
      }



      void humidityToDisplay()
      {
            nyhum = hum/10;
            for(int i = 0; i <= 95; i++)
            {
                  if(nyhum == i)
                  {
                        writeGAFFEL(nyhum);
                        break;
                  }

            }
      }


      uint8_t read_DHT22()
      {
            for(uint8_t i=0; i<8; i++)
            {
                  while((PIND& (1<<PD4))==0);        //detect data bit
(high pulse)
                  _delay_us(50);

//---------------------------------------------------------------------
                  if(PIND & (1<<PD4)) dataByte = (dataByte<<1)|(0x01);
                  else dataByte = (dataByte<<1);      //store 1 or 0 in
dataByte

//---------------------------------------------------------------------
```

```
                while(PIND & (1<<PD4));             //wait fot DHT22
low pulse
            }
            return dataByte;
      }

      void medelhav(){
            setDataRam(0xD6);
            _delay_us(150);
            write(0x4D);     // M
            _delay_us(150);
            write(0x45);      // E
            _delay_us(150);
            write(0x44);      // D
            _delay_us(150);
            write(0x45);      // E
            _delay_us(150);
            write(0x4c);      // L
            _delay_us(150);
            write(0x48);     // h
            _delay_us(150);
            write(0x41);     // A
            _delay_us(150);
            write(0x56);      // v
            _delay_us(150);
            write(0x45);      // E
            _delay_us(150);
            write(0x54);     // T

      }

      void tropisk(){
            setDataRam(0xD6);
            _delay_us(150);
            write(0x54);     // T
            _delay_us(150);
            write(0x52);      // R
            _delay_us(150);
            write(0x4f);      // O
            _delay_us(150);
            write(0x50);      // P
            _delay_us(150);
            write(0x49);      // I
            _delay_us(150);
            write(0x53);     // S
            _delay_us(150);
            write(0x4b);     // K
            _delay_us(150);
            write(0x54);      // T
            _delay_us(150);
            write(0x20);      //
```

```
            _delay_us(150);
            write(0x20);      //
      }
      void update_zone(){

            switch(zone){
                  case(1):
                  opti_temp = 25;
                  opti_hum = 70;
                  opti_jord = 30;
                  medelhav();
                  break;
                  case(2):
                  opti_temp = 30;
                  opti_hum = 80;
                  opti_jord = 35;
                  tropisk();
                  break;
//                default:
//                medelhav();
//                break;

            }
      }
      void lamp_update(){

            if((tempDHT/10) < opti_temp-5){
                  PORTA |= (2);
            }
            if(tempDHT/10 > opti_temp+3){
                  PORTA &= ~(2);
            }

            if(humDHT/10 < opti_hum-10){
                  PORTA |= (1);
            }
            if(humDHT/10 > opti_hum+5){
                  PORTA &= ~(1);
            }
            if(nyhum < opti_jord-5){
                  setDataRam(0x9e);
                  _delay_us(150);
                  write(0x2a);      // vattning

            }
            if(nyhum > opti_jord+10){
                  setDataRam(0x9e);
                  _delay_us(150);
                  write(0x20);
            }
      }
```

```
void clock_update()
{
      setDataRam(0x96);
      _delay_ms(1);
      write(timmar/10 + 48);
      _delay_ms(1);
      write(timmar%10 + 48);
      _delay_ms(1);
      write(0x3A); // kolon mannen
      _delay_ms(1);
      write(sekunder/10 + 48);
      _delay_ms(1);
      write(sekunder%10 + 48);
}

void dayOrNight()
{
      if(timmar % 2 == 0)
      {
            setDataRam(0x90);
            writeDAY();
            _delay_ms(1);
            write(0x20);
            _delay_ms(1);
            write(0x20);
            _delay_ms(1);
            PORTA |= (4);
      }else{
      setDataRam(0x90);
      _delay_ms(1);
      writeNIGHT();
      PORTA &= ~(4);
      }
}

registerOn();
displayON();
functionSet(); // sätt cursor till default.
clear();
// writeTEMPERATUR();
DDRA = 7;
PINA = 32;
setDataRam(0x02);
writeTEMPERATURE();
write(0x3A);

setDataRam(0xC0);
//SET DDRAM ADRESS - Second Line = 0xC0

writeHUMIDITY();
```

```
	write(0x3A);

	setDataRam(0x90);
	//SET DDRAM ADRESS - Third Line = 0x90

	writeDAY(); //OR Night
	setDataRam(0xD0);
	//SET DDRAM ADRESS - Fourth Line = 0xD0
	writeMODE();
	write(0x3A);



	while (1)
	{
		ADCenable();
		hum = ADC_READ();

//		humidityToDisplay();

		_delay_ms(2000);				//read from DHT22
every 2s

		start_signal();				//send start signal to
sensor
		response_signal();			//receive response
from sensor
		RH_high = read_DHT22();			//read high byte
humidity
		RH_low = read_DHT22();			//read low byte
humidity
		temp_high = read_DHT22();		//read high byte temp
		temp_low = read_DHT22();		//read low byte temp
		checksum = read_DHT22();		//read checksum
		humDHT  = (RH_high << 8) | RH_low;	//get 16-bit value
of humidity
		tempDHT = (temp_high << 8) | temp_low;	//get 16-bit value
of temp

		registerOn();
		functionSet();

	setDataRam(0x8C);
	// uppdatera temperatur
	setDataRam(0xC5);
	write(0x4A); // J
	setDataRam(0xc6);
	write(0x3A); // :
	humidityToDisplay(); // HUM I JORDEN
	setDataRam(0xC9);
	write(0x25); //  %
```

```
	setDataRam(0xCB);
	write(0x4C); //L
	setDataRam(0xCC); // kolon
	write(0x3A);
	writeGAFFEL(humDHT/10);
	setDataRam(0xCF);
	write(0x25); //  %
	write(0x25);
	setDataRam(0x86);
	writeGAFFEL(tempDHT/10);
	setDataRam(0x88);
	write(0xDF); // krumelur
	setDataRam(0x89);
	write(0x43); // bokstaven C


	if((PINA & (1<<3)) > 0){
		while((PINA & (1<<3)));
			if((PINA & (1<<3))==0){
				zone++;}

			if(zone>2){
				zone=1;
				}
				update_zone();
	}
	DDRA =7;
	lamp_update();

		sekunder +=2;
		if(sekunder == 60)
		{
			timmar++;
			sekunder = 0;


		}
		clock_update();
		dayOrNight();
	}


}
```