Software code

/*Code for data logging for *Arduino* with Atmega328 chip and 3 differential pressure sensors MPX5100DP mounted on tensiometers for monitoring water potential in soils.

After powering, the tensiometer station serial connection can be established either from within the *Arduino* IDE (serial monitor) or using a third-party software (e.g. Hyper Terminal). Set the baud rate to 9600 bps. A user menu will appear when typing the character "x", listing the following options, which can be selected by typing the appropriate letter:

s (status) --> displays momentary tensiometer values and various programme parameters

t (transfer data) --> start data transfer from *Arduino* to PC

e (erase memory) --> erase all data stored in the *Arduino* EEPROM

h (set time) --> set the time of the *Arduino* software clock

d (set date) --> set the date of the *Arduino* software calendar

i (set recording interval) --> set the time interval at which tensiometer recording should take place

b (set blink threshold) --> set the tensiometer value threshold at or above which the LED should start blinking

*/

#include <EEPROM.h> // this library is needed for writing/reading data to/from the atmega's EEPROM

```
const int powerpins[]={10,11,12}; //this array lists the digital pins used for supplying power to the pressure sensors – avoid using digital pins 0 and 1.
const int tensiopins[]={0,1,2}; //this array specifies the analogue pins which measure sensor output. Current is provided to the sensors from the digital pins listed in the same sequential order as in the array "powerpins". A maximum of 6 sensors can be connected (AD ports 0-5).
const int offset=10; //this corresponds to the typical value of 0.2V for the calibration offset of the MPX5100DP sensor.
/*
This is an example code for 3 tensiometers. To change the number of tensiometers, simply declare less or more digital and analoueg pins in the respective arrays:
 example 1: a 2-tensiometer configuration:
 int powerpins[]={3,4};
 int tensiopins[]={0,1};

 example 2: a full configuration with 6 tensiometers:
 const int powerpins[]={3,4,5,6,7,8};
 const int tensiopins[]={0,1,2,3,4,5};
 */
int mem_counter;  //counter for available memory
int time;
int _MINUTE=0;
int _HOUR=0;
int value;
long minute_counter;
```

```
int _DAY=1;
int _MONTH=1;
const int month_days[12]={
  31,28,31,30,31,30,31,31,30,31,30,31}; //this array contains the length (no. of days) of each month
const int tensionumber=(sizeof(tensiopins)/2); //determine the number of tensiometers from the number of ports listed in the array "tensiopins"
const int led_pin=13; // Pin 13 is connected to a LED on most Arduino boards
int threshold=300; //variable for threshold of soil water tension at or above which the LED will start blinking; default = 300 hPa
int interval; // variable for the time interval between succesive tensiometer recordings
byte blinkflag=0;  //flag indicating whether tensiometer threshold has been reached


void setup()
{
  for (int i=0; i < tensionumber; i++){  //set the digital pins as output
    pinMode(powerpins[i], OUTPUT);
  }
   pinMode(led_pin, OUTPUT);
  minute_counter=(millis() / 60000);  //set minute counter to initial value
  delay(1000);
  Serial.begin(9600);
  delay(1000);
  mem_count_retrieve(); //retrieve the last stored value of the memory counter from EEPROM
  check_interval(); //look up the logging interval from EEPROM
```

```
}

void loop()
{
  time = _HOUR;
  while(time==_HOUR) {
      if (blinkflag==1) {  //if the tensiometer threshold has been reached, start blinking the LED
     blinkloop();
   }
    simple_clock();
    if (Serial.available() > 0) {
      checkrx();
    }

  }
    tensio_record();
}

void simple_clock(){

  if (millis() / 60000 != minute_counter) {
    minute_counter=(millis() / 60000);
    _MINUTE++;
```

```
/*60000 millis make a minute; however there might be variations due to differences between
individual quartzes/resonators mounted on the boards. If the clock runs too fast, increase
this value in the 2 programme lines above, if the clock runs too slow, reduce it
*/



  if(_MINUTE>59) {
    _MINUTE=_MINUTE-60;
    _HOUR++;
  }

  if(_HOUR>23)
  {
    _HOUR=0;
    _DAY++;
  }

  if(_DAY>month_days[_MONTH-1]) //check if end of the month has been reached
  {
    _DAY=1;
    _MONTH++;
  }

  if(_MONTH>12)
```

```
    {
      _MONTH=1;
    }
  }
}

void checkrx()
{
  if (Serial.read()==120) {  //if the character is "x", display the menu
    menu();
  }
  else {
    Serial.println("no access");
  }
  Serial.flush();  //empty serial buffer
}

void menu()
{
  Serial.println("******************");
  Serial.println("s - status");
  Serial.println("t - transfer data");
  Serial.println("e - erase memory");
  Serial.println("h - set time");
  Serial.println("d - set date");
  Serial.println("i - set logging interval");
```

```
Serial.println("b - set blink threshold");
Serial.println("******************");
while (Serial.available() == 0){
}  //wait for command over serial port
if (Serial.available()>0) {
  value = Serial.read();
}
if (value == 'h') { //h-set time
  set_time();
}
if (value == 'd') { //d-set date
  set_date();
}
if (value == 't') {//t-transfer data
  read_memory();
}
 if (value == 's') {  //s-status
  actual_read();
}
if (value == 'e') {  //e-erase memory
  clear_memory();
}
if (value == 'b') {  //b-blink threshold
  set_threshold();
}
if (value == 'i') { //i-set logging interval
```

```
    set_interval();
  }
  }

void tensio_record() {   //for each tensiometer take 10 readings and calculate the mean
  blinkflag=0;
  for (int i=0; i <tensionumber; i++){
    digitalWrite(powerpins[i], HIGH);  //power up sensor
    delay(50); //warming up of MPX5100DP
    int value = 0;
    for (int j=0; j <= 9; j++){  //take 10 readings and calculate mean value
      value=value + analogRead(tensiopins[i]);
      delay(100);
    }
    digitalWrite(powerpins[i], LOW);  //power off
    value=value/40; //mean value divided by 4 to reduce from 10 to 8 bit

   if((((value-offset)*5.00/256)/45*10000)>=threshold) {   //if tensiometer reading in hPa is
equal or higher than the threshold value (both as absolute values), start blinking the LED
     blinkflag=1;
     }
    if (mem_counter < 1024) {  //back to loop if memory space is exhausted
    interval=EEPROM.read(4); //retrieve value of interval from EEPROM
    if (_HOUR%interval==0 || _HOUR==0) { //if the set time interval has been reached, store
tensiometer readings in EEPROM
    EEPROM.write(mem_counter, value);
```

```
mem_counter = mem_counter+1;
    mem_count_write();
   }
  }
 }
}

void set_time(){
 Serial.print("hour(00-23): ");
 make_number();
  if (value>=0 && value<24) {
 _HOUR = value;
 }
 Serial.print("minute(00-59): ");
 make_number();
  if (value>=0 && value<60) {
 _MINUTE = value;
 }
 write_time();
}

void set_date(){
 Serial.print("day(01-31): ");
 make_number();
 if (value>=0 && value<=31) {
 _DAY = value;
```

```
  }
  Serial.print("month(01-12): ");
  make_number();
  if (value>=0 && value<=12) {
  _MONTH = value;
  }
   write_time();
}

void set_threshold(){
  Serial.print("threshold: ");
  Serial.println(threshold);
  Serial.print("new threshold/10: (00-60) "); //e.g. to set the threshold to 500, enter 50
  make_number();
  if (value>=0 && value<=60) { //possible threshold range is 0 - 600 hPa
  threshold = value*10;
  }
  Serial.println(threshold);
  }

void read_memory(){
  Serial.println("start logging (dd/mm h:min):");
  Serial.print(int(EEPROM.read(2)));
  Serial.print("/");
  Serial.print(int(EEPROM.read(3)));
  Serial.print("  ");
```

```
 Serial.print(int(EEPROM.read(0)));
 Serial.print(":");
 if((int(EEPROM.read(1)))<10) {
  Serial.print("0");
 }
 Serial.println(int(EEPROM.read(1)));
 Serial.print("logging interval (h): ");
 interval=EEPROM.read(4); //retrieve the last stored value of the logging interval
 Serial.println(interval);


 Serial.println("tensiometer readings:");
 for (int i=1; i<=tensionumber; i++){
   Serial.print("T");
   Serial.print (i);
   if (i%tensionumber==0) {              //one column for each tensiometer
     Serial.println("");
   }
   else {
     Serial.print("\t");
   }
 }
 mem_count_retrieve();
 int j=0;
 for (int i=7; i<=(mem_counter-1);i++){
   Serial.print(int(((EEPROM.read(i)-offset)*5.00/256)/45*10000));     //read from memory
and transform to hPa
```

```
    j++;
    if (j%tensionumber==0) {
      Serial.println("");
    }
    else {
      Serial.print("\t");
    }
    delay (30); //delay for reducing speed of data transfer to avoid serial overflow
  }
  Serial.println("data transfer complete");
  write_time();
}



void residual_memory(){
  mem_count_retrieve();
  interval=EEPROM.read(4); //retrieve the last stored value of the logging interval
  value=(1023-mem_counter)/tensionumber/(24/interval);
  Serial.print("remaining memory: ");
  Serial.print(value);
  Serial.print(" days  ");
  value=(((1023-mem_counter)/tensionumber)%(24/interval));
  Serial.print(value);
  Serial.println(" hours");
}
```

```
void actual_read() {
  Serial.println("actual values (hPa):");
  for (int i=0; i <tensionumber; i++){
    digitalWrite(powerpins[i], HIGH);
    delay(50);
    int value = 0;
    for (int j=0; j <= 9; j++){
      value=value + analogRead(tensiopins[i]);
      delay(100);
    }
    digitalWrite(powerpins[i], LOW);
    value=value/40;
    value=value-offset;
     if(value<0)  //avoiding negative numbers
     {value=0;
     }
    Serial.print(int((value*5.00/256)/45*10000));  //transformation of reading into hPa
according to calibration function
    if ((i+1)%tensionumber==0) {
      Serial.println();
    }
    else {
      Serial.print("\t");
    }
  }
  Serial.print("blink threshold (hPa): ");
```

```
  Serial.println(threshold);
  Serial.print("logging interval (h): ");
  interval=EEPROM.read(4); //retrieve the last stored value of the logging interval
  Serial.println(interval);
  residual_memory(); //display the availabe memory for data logging
  write_time();
}


void make_number(){
  int a;
  int b;
  while (Serial.available() == 0){
  }  //wait for data from serial port
  if (Serial.available()>0) {
    a = Serial.read();
  }
  Serial.print(a-48);

  while (Serial.available() == 0){
  }
  if (Serial.available()>0) {
    b = Serial.read();
  }
  Serial.println(b-48);
  value=((a-48)*10+b-48);
```

```
}

void write_time(){
  Serial.print("date: ");
  Serial.print(_DAY);
  Serial.print("/");
  Serial.println(_MONTH);
  Serial.print("time: ");
  Serial.print(_HOUR);
  Serial.print(":");
  if (_MINUTE<10) {
    Serial.print("0");
  }
  Serial.println(_MINUTE);
}

void mem_count_retrieve() {
  mem_counter=EEPROM.read(5)*256+EEPROM.read(6);
}

void clear_memory() {
  Serial.println("sure? y/n");
  while (Serial.available() == 0){
  }  //wait for data from serial port
  if (Serial.read()==121) { //y
    mem_counter=7;  //the first 7 bytes of the EEPROM are reserved for storing variables
```

```
    mem_count_write();
    EEPROM.write(0,_HOUR);  //save hour to address 0
    EEPROM.write(1,_MINUTE); //save minutes to address 1
    EEPROM.write(2,_DAY);  //save day to address 2
    EEPROM.write(3,_MONTH); //save month to address 3
    EEPROM.write(4,interval); //save logging interval to address 4
    Serial.println("memory cleared");
  }
  Serial.flush();  //empty serial buffer
}

void mem_count_write() { //write value of memory counter to EEPROM
  EEPROM.write(5, mem_counter/256);
  EEPROM.write(6, mem_counter%256);
}

void blinkloop() {
  digitalWrite(led_pin, HIGH);   // set the LED on
  delay(500);              // wait
  digitalWrite(led_pin, LOW);    // set the LED off
  delay(1500);              //wait
}

void set_interval(){ //a newly set interval will only become effective after clearing the memory with the appropriate function in the menu
 Serial.print("interval (h): ");
```

```
  Serial.println(interval);
  Serial.println("new interval (01-02-03-04-06-08-12-24) ");
  make_number();
  if (value==1 || value==2 || value==3 || value==4 || value==6 || value==8 || value==12 ||
value==24){
  interval = value;
  }
  Serial.print("new interval: ");
  Serial.println(interval);
  Serial.println("clear memory to make change effective!");
 }

 void check_interval() {
 value=EEPROM.read(4); //retrieve the last stored value of the logging interval
 if (value==1 || value==2 || value==3 || value==4 || value==6 || value==8 || value==12 ||
value==24){
  interval = value;
  }
  else {     //if no value for the logging interval is set, use 1h as default
  interval=1;
  EEPROM.write(4,interval); //save logging interval to address 4
  }
 }
```