

EASY TO USE TAILORED AUTOMATIC DATA LOGGER DESING USING MICROCONTROLLERS IN A TWO-STAGE GASIFICATION EXPERIMENT

ANDRÁS ARNOLD KÁLLAY¹–VIVIEN VINCZE²–GÁBOR NAGY³

Microcontrollers have been around for almost five decades now. They have been and are still used in almost every consumer electronics since their first appearances in the calculator war of the 70's and 80's between the biggest names in electronics. However, their use was limited as their programming language was difficult and one had to be an expert in electronics and had to spend a large amount of funds just for basic programming equipment. Their cost and accessibility overcame their advantages for use in prototyping and research. In the last decade though, a platform emerged, with all the key features needed to make it popular and accessible. It is easy to program, accessible and very low priced. One of the most popular of these microcontroller boards are the Arduinos. In this paper, a working example of the implementation in our two-stage pyrolysis experiment will be presented discussing in detail the hardware and the software setup.

Keywords: data logger, microcontroller, Arduino, automatization

INTRODUCTION

Data logging is essential in every experiment of every single research branch. Besides making the process easier, automatization of data logging also speeds up the measurements. With a faster measurement, more frequent measurements can be made throughout the experiment. Thus, a more detailed dataset can be logged at the end of the experiment. The details of the current experiment are presented in the papers *The analysis of the solid and liquid phase products of two-stage pyrolysis* and *Two-stage pyrolysis of Hungarian brown coal to reduce hydrocarbons within synthesis gas*. The temperature of an experiment can have a profound effect on the synthesis gas produced during pyrolysis and gasification [1–3]. To precisely follow the processes taking place in the two-stage gasification experiment, the temperatures had to be measured in six different places of the two furnaces used in the experiment. This setup is also required in order to separate the high from the low temperature processes and assure that heat is not transferred from one process to the other. The experiment itself, including the low and high temperature pyrolysis, is usually no longer then a few hours. However, the temperature must be recorded frequently as the temperature can reach a heating rate of 20 °C/min. There are commercially available solutions for temperature logging. The YTC YC-74UD Data

¹ University of Miskolc, Department of Combustion Technology and Thermal Energy
Miskolc-Egyetemváros 3515, Hungary
tuzaak@uni-miskolc.hu

² University of Miskolc, Department of Combustion Technology and Thermal Energy
Miskolc-Egyetemváros 3515, Hungary

³ University of Miskolc, Department of Combustion Technology and Thermal Energy
Miskolc-Egyetemváros 3515, Hungary

logger thermometer as an example, which is capable of measuring temperatures from four different thermocouples and has an on-board memory to log the measurements [4].

These tools are relatively expensive and their use is rather cumbersome as it needs special cable and software to acquire the logged data. However, in the last decade a series microcontrollers have gained popularity with their user-friendly programming interface and a wide variety of sensor breakout boards that can be customised to fit the requirements of the users. In our case, this was temperature measurement from 6 thermocouples, real time data displaying and logging to a more commonly available and used data storage device. These series of microcontroller boards are commercially known as Arduinos [5], their board is based around *Atmel AVR microcontrollers* [6], their programming language is based on *Wiring* [7] and their software is based on *Processing* [8]. All these mean that the microcontroller boards are inexpensive, as they are open source and can be purchased with no licence fee. Therefore, they can be acquired through online web shops directly from the Chinese manufacturers for prices that often cost less delivered, than their domestic shipment would cost in Hungary.

Their developed software (Arduino IDE [9]) is cross-platform that works on all major operating systems, and has a simple and clear programming environment. Easy to use for beginners but flexible enough for more advanced users. The software is also extensible, the language is expanded through C++ libraries which is a key feature and also the reason it gained such high popularity over the years. Almost every single sensor breakout board that is available has a library written for it and the data can be acquired with a few simple lines of codes within the IDE. In this paper, the data logger we created for our research will be presented as an example to demonstrate how easily automated data logging microcontrollers can be used in your project. The data logger can easily be extended, modified and tailored as the research progresses. Therefore it is also useful for speeding up the process of the development.

1. MATERIALS AND METHODS

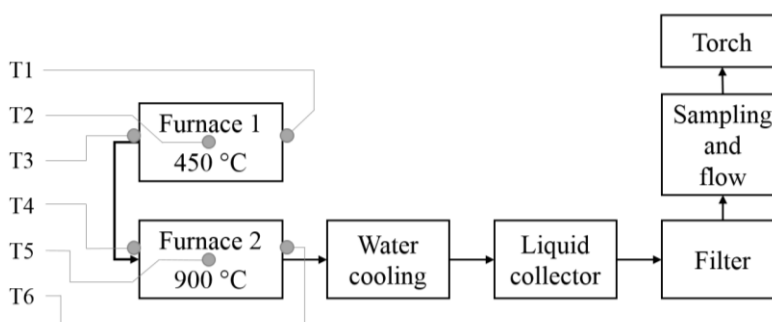


Figure 1

The schematic of the furnace setup for the gasification experiment, grey spots marking the temperature measurement points

The experiment was set up using two tube furnaces, in which the temperature was measured at the entry points of the heat resistant steel pipes placed inside the furnaces and halfway between the two entry points inside the furnace, as it is shown in the schematics on *Figure 1*.

The thermocouples used in the experiments are K-type thermocouples with MAX6675 modules. To display the temperature measurement in real time a 5110 LCD module was used with SPI protocol. The temperature data was logged with a MicroSD Card Adapter board. The microcontroller board used in this experiment was an Arduino Pro Mini with 5V logic level and an Atmel Atmega328P, 16MHz microcontroller. The components were placed on a prototyping board and the connection soldered to place as it is shown in the schematic, the details of the schematic and the placement will be detailed in the “Results and discussion” section of the paper.

2. RESULTS AND DISCUSSION

The discussion of the results will be sectioned in two parts, the first part describing the hardware. In the hardware section the potentials behind of this platform and their use in other data logging and capability to fulfil unique needs of individual research areas will also be discussed. The second part will discuss the software written for this experiment and details of how it can be expanded to other measurements and automated controlling, and finally an example of the measured data set will be presented.

2.1. Hardware setup for the data logger

The main controller used in this experiment for data logging is an Arduino Pro Mini using the Atmel Atmega328P microcontroller. There are many variation of microcontroller boards, differing in size, I/O pin numbers and features. One must consider the amount of data that has to be measured and/or the number of outputs that must be controlled to choose a microcontroller with the required number I/O pins and memory. Every single expansion board comes with its own specific chip and the required components installed on its own breakout-board, the user is only required to connect the wires according to the schematics of the board and upload the example code which is usually provided as a download.

The communication between the breakout-boards (board containing the microchip and the passive electric components) and the microcontroller are achieved through SPI or I²C protocols in case of more advanced boards as a display, clock module, micro SD card board and libraries, as their implementation is also provided. The input from the switches can be read through digital inputs as high (5 volts) or low (ground, 0 volts). The analogue sensors can be read through the analogue inputs of the microcontroller, with signals between 0 and 5 volts with a 10-Bit resolution (1024 steps). Relays, switches or other modules can be controlled through the digital outputs of the microcontroller, either driving the pins of the Arduino high (5 volts) or low (ground, 0 volts,) or through a PWM signal with an average voltage between 0 and 5 volts and with an 8-Bit resolution (256 steps). To read or write analogue signals with resolutions higher than those provided by the microcontroller, ADC and DAC breakout-boards are also available.

Our requirements were 6 temperature measurements realised through K-type thermocouples and MAX6675 breakout-boards, a MicroSD Card Adapter and display attached to show the real-time temperature measurements, in this case a 5110 LCD display (*Figure 2*). The microcontroller chosen for the data logger is an Arduino Pro Mini as it has the required number of input and output pins, the required amount of memory and it can be mounted on a prototyping board due to its small size. All chosen parts for the data logger communicate with the microcontroller through SPI protocol which communicates with 4 wires in a master-slave configuration. Two wires are used for the data communication: master

in slave out (MISO), master out slave in (MOSI), serial clock (SCLK) and slave select (SS). One of the main advantages of the SPI communication is that if more than one slave is required, as in our case for the temperature sensors for each additional device, only one additional wire (SS) is required, the other wires can be shared between the devices. A disadvantage of this protocol is that more than one clock signal triggers exist (different polarity and phase) for data transmission depending on the used protocol. From the user's point of view this means that while the same type of sensors, displays or other modules can share the same MOSI, MISO and SCLK wires, others can vary. This is the reason behind the three different SPI bus generated for our data logger, which can be seen on *Figure 3*. The MAX6675 sensor board shares the same data out lines (DO same as MISO), SCK lines and the power lines and has separate pins set up for the chip select lines (CS same as SS). For the LCD, another SPI bus was generated with all the pins required and for the micro SD card adapter as well. In addition to the SPI lines, a toggle switch and an LED were implemented in the circuit as well, for turning on the data writing and a visual confirmation if the temperature data writing has started on to the micro SD card.

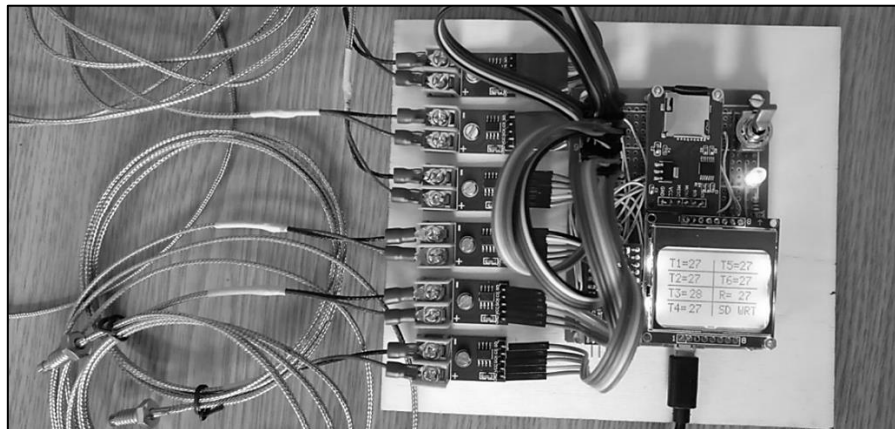


Figure 2

The temperature data logger fully wired, with the K-type thermocouples attached to the MAX6675 sensor boards

2.2. Software for the temperature data logger

The Arduino software environment (Arduino IDE) can be downloaded from the developers' page [9] as a free download and is available for the most popular operating systems, including Windows, Mac OS and Linux. The Arduino environment is written in Java and based on processing and other open-source software. After the software is installed, the required libraries for each breakout board must be installed in the Arduino's library folder. The software written within the Arduino IDE is called sketch. After the sketch has been written, it can be uploaded to the microcontroller board through USB cable or, in the case of the Arduino Pro Mini, using USB to Serial converter board. The written sketch is translated to machine language by the Arduino software and after uploading, the uploaded sketch is ran by the microcontroller continuously.

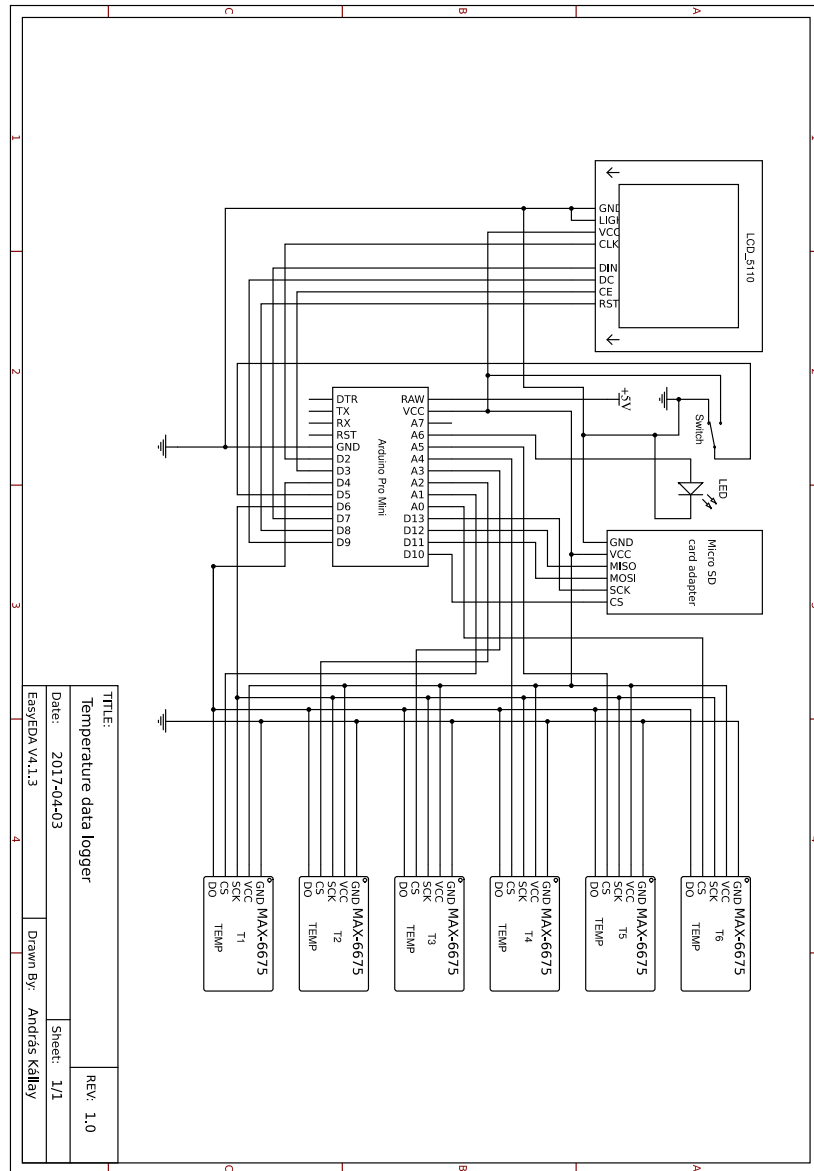


Figure 3

Schematic of the temperature data logger wiring diagram

The sketch is usually divided to two sections. The first is the setup section and the second is the loop section. The inclusion of the libraries required, the declaration of the variables and constants and the assignment of the pin function are contained by the setup section of the code.

All codes presented in this paper are shown as they appear in the written sketch for the temperature data logger in the Arduino IDE, broken down into smaller segments for the comfortable explanation and reading. In our temperature data logger sketch, the included

libraries (*Sketch segment 1*) for the three types of boards used in our setup are contained in the first part.

```
#include "max6675.h"
#include "SdFat.h"
#include "LCD5110_Graph.h"
```

Sketch segment 1

Code written for the inclusion of libraries required

The inclusion of libraries is followed by the assignment of the pins for each board (*Sketch segment 2*). The text written after the double slash signs are comment placed by the writer of the sketch. The assignment of each pins for the “SdFat” library is predefined, except for the chip select (SS) pin that can be changed by the user. The pins of the other two boards are arbitrarily chosen by the user and placed within the code as it is shown in the example. In addition to the assignment of pins, the font is also assigned in this section for the LCD. The following lines of code (*Sketch segment 3*) contain the assignment of the pins for the LED and the toggle switch, as well as the initialisation of the “millis” function that is necessary to be able to measure the time, and the micro SD card file name. Next is the “void setup” part of the code where the LCD and the micro SD card are initiated (*Sketch segment 4*).

```
//SdFat pins: MOSI - pin 11, MISO - pin 12, CLK - pin 13, CS - pin 10
SdFat SD;
const int chipSelect = 10;

//Pin associated with the LCD module: SCK - Pin 2, MOSI - Pin 7, DC - Pin 9, RST - Pin 8, CS - Pin 3
LCD5110 myGLCD(2,7,9,8,3);

//Font assigned for the LCD
extern unsigned char SmallFont[];

//Pins associated with the thermocouples
int thermoDO = 4;
int thermoCLK = 6;
int thermoCS1 = A1;
int thermoCS2 = A2;
int thermoCS3 = A3;
int thermoCS4 = A4;
int thermoCS5 = A5;
int thermoCS6 = A0;
MAX6675 T1(thermoCLK, thermoCS1, thermoDO);
MAX6675 T2(thermoCLK, thermoCS2, thermoDO);
MAX6675 T3(thermoCLK, thermoCS3, thermoDO);
MAX6675 T4(thermoCLK, thermoCS4, thermoDO);
MAX6675 T5(thermoCLK, thermoCS5, thermoDO);
MAX6675 T6(thermoCLK, thermoCS6, thermoDO);
```

Sketch segment 2

Code for the assignment of the pins to the breakout-boards

After the setup section of the sketch is followed by the loop section. In the loop section of the sketch, the code that will cycle through the microcontroller after the upload is contained. The microcontroller follows the code line by line and the task in the code are performed. The setup of the LCD screen, divided into separate sections, for the six temperature measurement data and the micro SD card state (*Sketch segment 5*) are contained in the first part of the code. The insertion of the data read from the sensors in each segment set up on the LCD is also included in the command.

```
//Initialising switch and led pins and state of the switch
int Switch = 5;
int SwitchState = 0;
int LedPin = A6;

//Millis fucntion required for time measurement
unsigned long previousMillis = 0;
unsigned long interval = 100;

//File name decalration for the SD card
File TimeFile;
```

Sketch segment 3

Code for the initialisation of the switch, LED, time and micro SD card functions.

```
void setup() {

  pinMode(Switch, INPUT);
  pinMode(LedPin, OUTPUT);
  SD.begin(chipSelect);

  myGLCD.InitLCD();
  myGLCD.setFont(SmallFont);
}
```

Sketch segment 4

Code written for “void setup”

```
//Main loop
void loop() {

  //Setup LCD to show if the SD card has initialized correctly(if yes = ON, if not= OFF)
  myGLCD.clrScr();
  if (!SD.begin(chipSelect)) {
    myGLCD.clrScr();
    myGLCD.print("SD OFF", 48, 36);
  }
  else{
    myGLCD.print("SD ON", 48, 36);
  }

  //LCD setup for alignment of the grid and displaying the read thermocouples and flowmeter data
  myGLCD.drawLine(42,0,42,45);

  myGLCD.print("T1=", LEFT, 0);
  myGLCD.printNumF(T1.readCelsius(), 0, 18, 0);
  myGLCD.drawLine(0,9,84,9);

  myGLCD.print("T2=", LEFT, 12);
  myGLCD.printNumF(T2.readCelsius(), 0, 18, 12);
  myGLCD.drawLine(0,21,84,21);

  myGLCD.print("T3=",LEFT, 24);
  myGLCD.printNumF(T3.readCelsius(), 0, 20, 24);
  myGLCD.drawLine(0,33,84,33);

  myGLCD.print("T4=", LEFT, 36);
  myGLCD.printNumF(T4.readCelsius(), 0, 20, 36);

  myGLCD.print("T5=", 48, 0);
  myGLCD.printNumF(T5.readCelsius(), 0, 66, 0);
  myGLCD.drawLine(0,9,84,9);

  myGLCD.print("T6=", 48, 12);
  myGLCD.printNumF(T6.readCelsius(), 0, 66 , 12);
  myGLCD.drawLine(0,21,84,21);
```

Sketch segment 5

Code for the LCD setup

The micro SD card adapter is set up with a condition statement (*Sketch segment 6*). The function of the condition statement, the writing function, can be turned on with an input from the user, in our case with a toggle switch. In addition to this functionality, the availability of the micro SD card is also checked, if it has been inserted and it can be successfully written in the file defined in the sketch. The file produced by the code is a text file saved on the micro SD card. The temperature values from each sensor are written in the same line, including a time stamp from the “millis” function. If the data writing is successful, the wired led is lit up and the LCD screen presents a “WRT” status in the section of the SD card state defined in the LCD section. If the card is not inserted, the card state section of the SD card shows “OFF” as a written feedback. If the SD card is inserted but the toggle switch is not in the writing position, the “ON” written feedback is shown on the LCD’s SD card state section. In the last part of the loop section, the LCD is refreshed and a delay of 1000 milliseconds is inserted before the loop section starts again.

```
//setup of the switch to start the data logging in the SD card
SwitchState = digitalRead(Switch);
if (SwitchState == HIGH){
  unsigned long currentMillis = millis(); //millis function for time stamping the logged data
  if (currentMillis - previousMillis >= interval){
    previousMillis = currentMillis;
    TimeFile = SD.open("DATALOG.txt", FILE_WRITE);//creating file on the SD card
    if (TimeFile) {
      myGLCD.print("SD WRT", 48, 36); //displaying if the file opened succesfully
      analogWrite(LedPin, 255); //indicating the file opening with a LED
      TimeFile.print(currentMillis/1000);
      TimeFile.print(", ");
      TimeFile.print(T1.readCelsius());
      TimeFile.print(", ");
      TimeFile.print(T2.readCelsius());
      TimeFile.print(", ");
      TimeFile.print(T3.readCelsius());
      TimeFile.print(", ");
      TimeFile.print(T4.readCelsius());
      TimeFile.print(", ");
      TimeFile.print(T5.readCelsius());
      TimeFile.print(", ");
      TimeFile.println(T6.readCelsius());
      //TimeFile.print(", ");
      //TimeFile.println(R6.readCelsius());
      TimeFile.close();
    }
  }
}
else {
  //turning of the LED if the data logging is switched off.
  digitalWrite(LedPin, LOW);
}
myGLCD.update(); //necesary refresh of the LCD

delay(1000);
}
```

Sketch segment 6
The micro SD card loop section

2.3. Experiment using the data logger

With the sketch setup presented, the loop runs in every second and returns the temperature measured with the K-type thermocouples. The measured data is presented in real time on the LCD and if the data logging is turned on, a file is created and the data is also written to a text file at every second of the experiment. Each line of the text file contains a time stamp and the 6 temperatures registered by the MAX6675 sensors, separated with commas. The text file can be easily imported in to MS excel data sheets, from where it can be further processed as desired. The data logging speed presented here is limited by the delay and it is obvious that it can be increased and decreased if necessary. In our case, the temperature logged every second is more than the required data.

Using the described hardware setup and software, the exact temperatures of the various data logging points could be measured automatically using the data logger, the result of which can be seen in *Figure 4*. The presented experiment was 95 minutes long. Increasing temperatures can be observed on the presented figure up to the 95-minute period point, when the furnace was turned off, after which the furnace temperatures started to decrease as the furnace continuously cooled down.

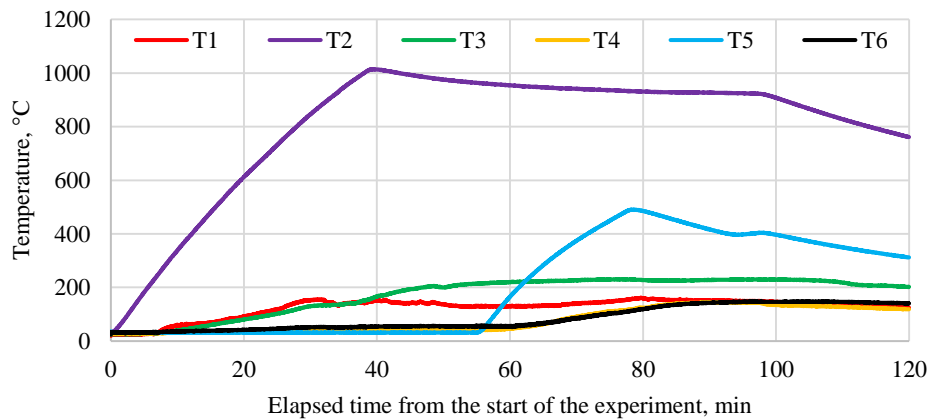


Figure 4
Temperatures collected by the data-logging system during an experiment

CONCLUSION

A temperature data logger setup has been presented, controlled by a microcontroller registering temperatures and with data rate speed set up by the user within the sketch running on the microcontroller. Setting up the hardware is relatively easy following this paper with step by step description, and connecting the right pins on the microcontroller to the appropriate pins on the breakout boards of the sensor. The written sketch is built up from the examples provided with the libraries of the breakout boards and are just pieced together. With this setup, the temperature data logger can show real time temperature measurements on an LCD screen and create a text file on a micro SD card with the data being registered every second with comma separated values. The text files created can be further processed in spreadsheet handling applications, like MS Excel. Furthermore, the setup can be easily

customised to fit special setups, and the whole setup can cost ten times less compared to the commercially available temperature data loggers with just four thermocouple inputs.

To extend the data logger, measuring temperatures from more thermocouples is just a question of attaching more MAX6675 breakout boards to the microcontroller and including them in the sketch. If more input pins are required, there are larger microcontroller boards available. To increase or decrease the frequency of the measurement, a simple adjustment is needed in the sketch written. As almost every commercial sensor for data measurement, an analogue output or input function that operates at 5V logic level is included. The data sent by the sensor can be read, registered, displayed, converted by the microcontroller, while the analogue and digital write functions of the microcontrollers allow precise control of motors, controllers, relays and solid state switches operating high powered applications, etc. For higher resolution, analogue read/write function breakout boards are also available. The opportunities opened by this microcontroller platform are almost endless and including it in our research has undeniable advantages in the measurements and controlling of the experiments.

ACKNOWLEDGEMENT

The authors are grateful to Mária Ambrus for her advice.

REFERENCES

- [1] WANG, Q.H.–ZHANG, R.–LUO, Zy Y.–FANG, M. X. –CEN, K. F.: Effects of Pyrolysis Atmosphere and Temperature on Coal Char Characteristics and Gasification Reactivity. *Energy Technology*, Vol. 4, Issue 4 (2016), 543–550.
- [2] ZHANG, H.X.–GUO, X.W.–ZHU, Z.P.: Effect of temperature on gasification performance and sodium transformation of Zhundong coal. *Fuel*, Vol. 189 (2017), 301–311.
- [3] BEHNIA, I.–YUAN, Z. S.–CHARPENTIER, P.–XU, C.B.: Production of methane and hydrogen via supercritical water gasification of renewable glucose at a relatively low temperature: Effects of metal catalysts and supports. *Fuel Processing Technology*, Vol. 143 (2016), 27–34.
- [4] Webshop, YC-747UD Hand-held Data Logger 4ch. Thermocouple Thermometer with USB PC Interface. <http://www.tmswebshop.co.uk/buy/yc-747ud-hand-held-data-logger-thermometers-with-usb-pc-interface-yc747dlogger.html>. (Downloaded 2017. 04. 05.)
- [5] Arduino - Home, 2017. <https://www.arduino.cc/>. (Downloaded 2017. 04. 04.)
- [6] Atmel AVR 8-bit and 32-bit Microcontrollers, 2017. <http://www.atmel.com/products/microcontrollers/avr/default.aspx>. (Downloaded 2017. 04. 06.)
- [7] BARRAGN: H., Wiring, 2017. <http://wiring.org.co/>. (Downloaded 2017. 04. 05.)
- [8] P. Foundation, Processing.org, 2017. <https://processing.org/>. (Downloaded 2017. 04. 04.)
- [9] Arduino – Software, 2017. <https://www.arduino.cc/en/Main/Software>. (Downloaded 2017. 04. 04.)