

## Sensors and Electronics Assignment 2 – Aaron Lockett

### Task 4 – Temperature Monitor

Create a sketch which will read the analogue voltage from the LM35, convert it to a temperature in °C and print this temperature on the serial monitor. [5]

This process should repeat once per second. [2]

Display the temperature on the serial monitor in °C using 1 digit after the decimal point. [3]

Include a copy of your sketch into your submission along with output produced by the program on the serial monitor.

```
/*
Sketch to read analog voltage on pin A3 from a LM35 thermistor in order to
read the temperature of the room it is in*/
int sensorValue = 0; // value read from the pot

//These values are used to keep time intervals between readings
unsigned long previousTime = 0;
unsigned long currentTime = 0;
const int interval = 1000; // 1000 ms between actions

// These constants are used to give names to the pins used:
const int analogInPin = A3; // Analog input pin for potentiometer

//Declares variables used in the loop
float temp = 0;

//Creates a map function that works with float values
float mapfloat(long x, long inMin, long inMax, long outMin, long outMax)
{
    return (float)(x - inMin) * (outMax - outMin) / (float)(inMax - inMin) +
    outMin;
}

void setup() {
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);
}

void loop() {

    // record current time
    currentTime = millis();

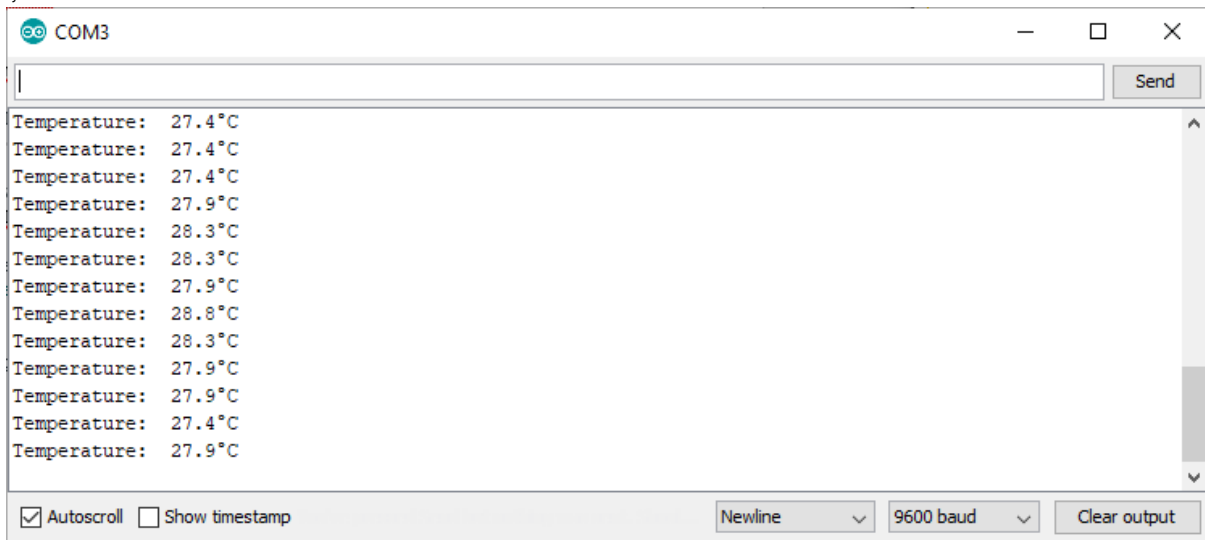
    // if statement to check if interval has elapsed
    if (currentTime - previousTime >= interval)
    {
        // update previous time
```

```
previousTime = currentTime;

// read the analog in value:
sensorValue = analogRead(analogInPin);

// convert the analogue voltage to Temperature (Celcius)
temp = mapfloat(sensorValue, 0, 1023, 0, 500);

//Print our values to the console
Serial.print("Temperature: ");
Serial.print(temp, 1);
Serial.println("°C");
} // end of if clause triggered every interval
}
```



The screenshot shows that the temperature is being read and displayed on the serial monitor. It does this every second (1000 milliseconds).

## Task 5 – Extended Temperature Monitor

Arrange your temperature monitor to have different operating modes, showing the temperature in °C, °F or K. [2]

The program should output temperatures in the current mode at the rate of 1 per second. [1]

The instrument should switch between modes depending on a character typed by the user into the serial monitor, such that C switches to Celsius, F to Fahrenheit and K to Kelvin. R or r should reset both the minimum and maximum temperatures to the current temperature. [2]

The user should be able to input the mode switch in upper or lower case.[1]

At each time step, the monitor should display the current temperature as well as the maximum and minimum temperatures reached since the program started.[2]

All 3 temperatures (current, min and max) should be displayed in the current temperature mode with 1 digit after the decimal point and a units indicator (C, F or K).[2]

```
/*
Sketch to read analog voltage on pin A3 from a LM35 thermistor in order to
read the temperature of the room it is in
*/

// These constants are used to give names to the pins used:
const int analogInPin = A3; // Analog input pin for potentiometer
unsigned long previousTime = 0;
unsigned long currentTime = 0;
const int interval = 1000; // 1000 ms between actions
//Declares variables used in the loop
float temp = 0;
String inString = "C";
String pastInString = "C";
float maxTemp; //Stores the maximum recorded temperature since last reset
float minTemp; //Stores the minimum recorded temperature since last reset
int sensorValue = 0; // value read from the pot

void setup() {
// initialize serial communications at 9600 bps:
Serial.begin(9600);
previousTime = millis();

//Calls the temperature reading and resetting functions once at the start
to initialise the min & max temp values
Readtemp();
tempReset();
}

void loop() {

    // Determines if an input has been sent through the serial monitor,
    converts it to upper case, preparing it to be interpreted
```

```

// by the tempFinder function to select the appropriate unit
if (Serial.available() > 0)
{
  inString = Serial.readString()[0];
  inString.toUpperCase();
}

// Determines if the serial monitor input is 'R' and, if so, resets the
maximum & minimum temperatures to the current temperature
if (inString.indexOf('R') != -1)
{
  tempReset();
  inString = pastInString;
  Serial.println("\nTemperature has been reset");
}
currentTime = millis(); // record current time

// if statement to check if interval has elapsed
if (currentTime - previousTime >= interval)
{
  previousTime = currentTime; // update previous time
  Readtemp();
  tempFinder(inString);
} // end of if clause triggered every interval
}

//Reads the Analogue value sent to the Arduino from the thermistor to
calculate the temperature in Celcius & compares it to the
//current max & min temperature values
void Readtemp()
{
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // convert the analogue voltage to Temperature (Celcius)
  temp = mapfloat(sensorValue, 0, 1023, 0, 500);
  minTempChecker(temp);
  maxTempChecker(temp);
}

//Function which outputs the readings for: Current, minimum & maximum
temperatures in the specified units
void tempFinder(String inString)
{
  //Serial.print(inString);
  Serial.print("\nTemperature: ");

  //Label for goto function in the switch
  switchStart:

  // Converts the first letter of the serial monitor input to a single char
value to be used with the switch statement
char command = inString.charAt(0);

// Switch to convert temperature reading to the selected unit
switch(command)
{
  //Switch unit to Celcius
  case 'C':
    //Serial.print(command);

```

```

        cel(temp);
        Serial.print(" Minimum; ");
        cel(minTemp);
        Serial.print(" Maximum: ");
        cel(maxTemp);
        pastInString = 'C';

        break;
    //Switch unit to Kelvin
    case 'K':
        //Serial.print(command);
        kel(temp);
        Serial.print(" Minimum; ");
        kel(minTemp);
        Serial.print(" Maximum: ");
        kel(maxTemp);
        pastInString = 'K';
        break;
    //Switch unit to Fahrenheit
    case 'F':
        //Serial.print(command);
        fah(temp);
        Serial.print(" Minimum; ");
        fah(minTemp);
        Serial.print(" Maximum: ");
        fah(maxTemp);
        pastInString = 'F';
        break;

    //Switch unit back to Celcius in the event of an unknown value being
    passed
    default:
        //Serial.print(command);
        command = 'C';

        //returns to just before the start of the Switch statement
        goto switchStart;
        break;
    }
}

// Function to convert the temp value to Fahrenheit & output temperature
values in Fahrenheit to one decimal place
void fah(float temp)
{
    float fahTemp = (temp*(9/5)+32);
    Serial.print(fahTemp, 1);
    Serial.print("°F");
}

// Function to convert the temp value to Kelvin & output temperature values
in Kelvin to one decimal place
void kel(float temp)
{
    float kelTemp = (temp + 273.15);
    Serial.print(kelTemp, 1);
    Serial.print(" K");
}

// Function to output temperature values in degrees Celcius to one decimal
place
void cel(float temp)

```

```

{
    Serial.print(temp, 1);
    Serial.print("°C");
}

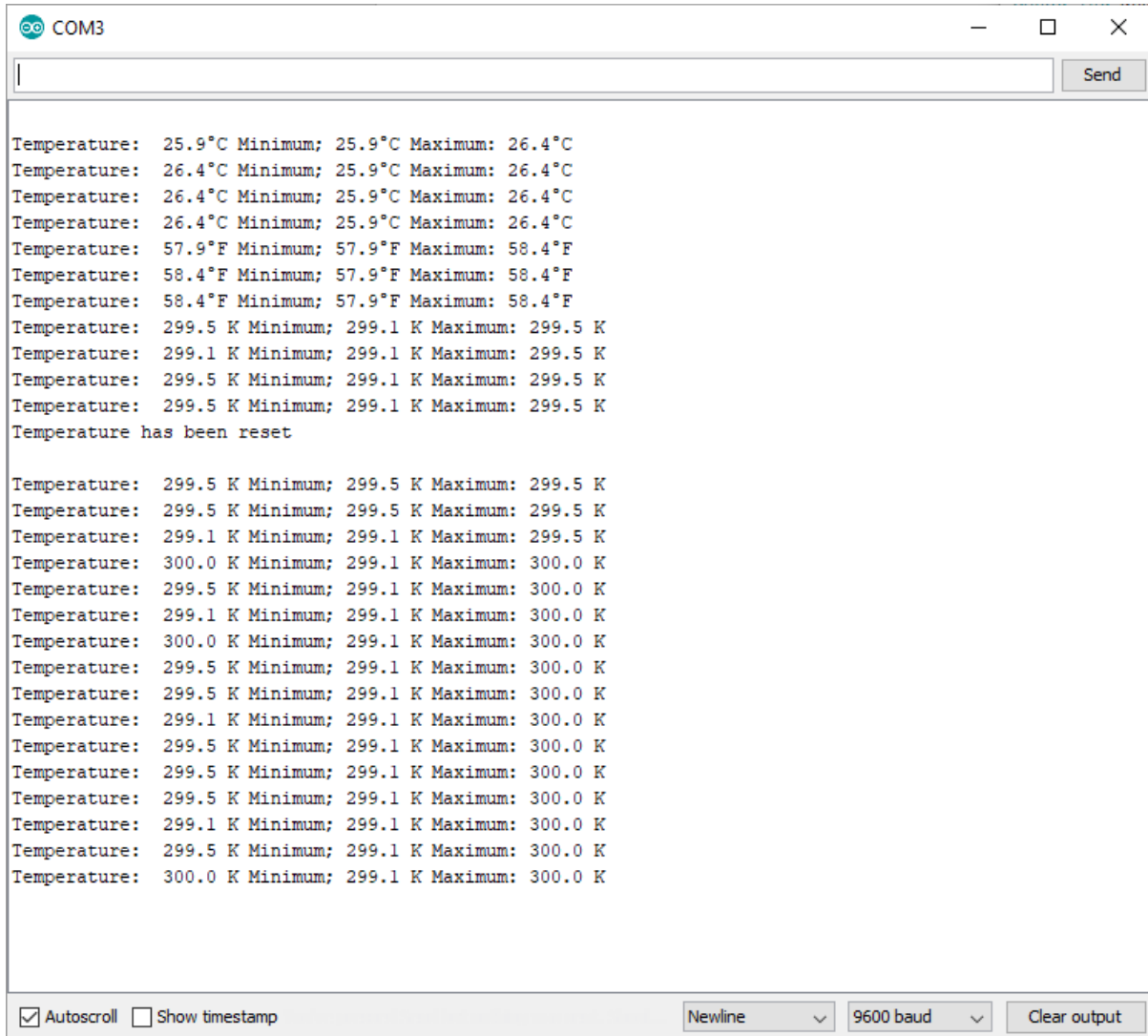
// Checks if temperature read is lower than minTemp and sets minTemp to
// equal temp if temp was lower
void minTempChecker(float temp)
{
    if (minTemp > temp)
    {
        minTemp = temp;
    }
    return;
}

// Checks if temperature read is higher than maxTemp and sets maxTemp to
// equal temp if temp was higher
void maxTempChecker(float temp)
{
    if (maxTemp < temp)
    {
        maxTemp = temp;
    }
    return;
}

// Resets the maximum & minimum temperature values when the function is
// called
void tempReset()
{
    maxTemp = temp;
    minTemp = temp;
    return;
}

//Creates a map function that works with float values
float mapfloat(long x, long inMin, long inMax, long outMin, long outMax)
{
    return (float)(x - inMin) * (outMax - outMin) / (float)(inMax - inMin) +
    outMin;
}

```



```
COM3
Temperature: 25.9°C Minimum; 25.9°C Maximum: 26.4°C
Temperature: 26.4°C Minimum; 25.9°C Maximum: 26.4°C
Temperature: 26.4°C Minimum; 25.9°C Maximum: 26.4°C
Temperature: 26.4°C Minimum; 25.9°C Maximum: 26.4°C
Temperature: 57.9°F Minimum; 57.9°F Maximum: 58.4°F
Temperature: 58.4°F Minimum; 57.9°F Maximum: 58.4°F
Temperature: 58.4°F Minimum; 57.9°F Maximum: 58.4°F
Temperature: 299.5 K Minimum; 299.1 K Maximum: 299.5 K
Temperature: 299.1 K Minimum; 299.1 K Maximum: 299.5 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 299.5 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 299.5 K
Temperature has been reset
Temperature: 299.5 K Minimum; 299.5 K Maximum: 299.5 K
Temperature: 299.5 K Minimum; 299.5 K Maximum: 299.5 K
Temperature: 299.1 K Minimum; 299.1 K Maximum: 299.5 K
Temperature: 300.0 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.1 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 300.0 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.1 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.1 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 299.5 K Minimum; 299.1 K Maximum: 300.0 K
Temperature: 300.0 K Minimum; 299.1 K Maximum: 300.0 K
```

☒ Autoscroll ☐ Show timestamp Newline 9600 baud Clear output

This shows the temperature being displayed in different units depending on what input was entered into the serial monitor. It also shows the minimum and maximum temperatures being reset when “R” or “r” was entered into the serial monitor. This continues to print every second.

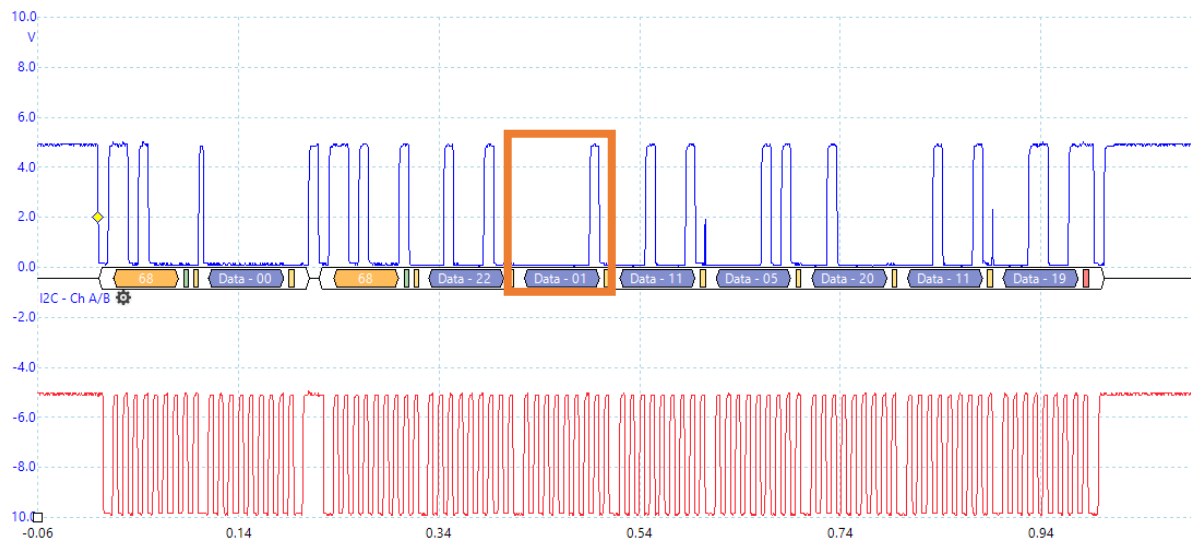
## Task 6 – Communications With The Real Time Clock

Use the Picoscope edit menu to copy the scope display as an image and paste it into your submission. [3]

On the image of the waveform in your document, highlight the byte of information that corresponds to the minutes being transferred from the real time clock to the Arduino. You can insert a shape from Word on top of the image in order to do this. [1]

Write down the hexadecimal address of the real time clock on the I2C bus. [1]

**Hexadecimal Address: 68**



To get the hexadecimal address we hovered over the data being sent and the program said the address was 68 (In hexadecimal). We believe that area inside the shape is the minutes data being sent as, as the program was running, that area of data corresponded to the real time clock.



## Task 7 – Weather Station

Construct a weather station display using the BME280 and the LCD keypad display to give a display similar to that shown in Figure 1, showing the temperature in degrees C to 1dp, the humidity in % and the atmospheric pressure in milliBar (mB);

The display should update once per second. Note that the BME280 returns pressure in Pascals and the display should show the pressure in mBar, so you will need to do that conversion.

```
#include <stdint.h>
#include "SparkFunBME280.h"

#include "Wire.h"
#include "SPI.h"

#include <LiquidCrystal.h>

//Global sensor object
BME280 mySensor;

unsigned long previousTime = 0;
unsigned long currentTime = 0;
const int interval = 1000; // 1000 ms between actions

//Creates the variables used in the program
float Temperature;
float Pressure;
float Humidity;

LiquidCrystal lcd(8,9,4,5,6,7);

void setup()
{

    //Initializes the BME280 sensor to run using an I2C protocol
    mySensor.settings.commInterface = I2C_MODE;
    mySensor.settings.I2CAddress = 0x77;

    //***Operation settings*****//
    mySensor.settings.runMode = 3; // 3, Normal mode
    mySensor.settings.tStandby = 0; // 0, 0.5ms
    mySensor.settings.filter = 0; // 0, filter off
    //tempOverSample can be:
    // 0, skipped
    // 1 through 5, oversampling *1, *2, *4, *8, *16 respectively
    mySensor.settings.tempOverSample = 1;
```

```

//pressOverSample can be:
// 0, skipped
// 1 through 5, oversampling *1, *2, *4, *8, *16 respectively
mySensor.settings.pressOverSample = 1;
//humidOverSample can be:
// 0, skipped
// 1 through 5, oversampling *1, *2, *4, *8, *16 respectively
mySensor.settings.humidOverSample = 1;

// Initializes the LCD screen & tells the Arduino the dimensions of the
screen
lcd.begin(16, 2);

delay(10); //Make sure sensor had enough time to turn on. BME280
requires 2ms to start up.

// Gets the sensor to start taking readings
mySensor.begin();

// variable used to help keep track of time so readings can be made every
second
previousTime = millis();
}

void loop()
{
currentTime = millis(); // record current time

// if statement to check if 1 second interval has elapsed
if (currentTime - previousTime >= interval)
{
previousTime = currentTime; // update previous time
//Calls the function which reads data from the BME280 sensor, formats it,
and writes the formatted data to the LCD
SensorDataReadWriter();

}
}

// Function which reads data from the BME280 sensor, formats it, and writes
the formatted data to the LCD
void SensorDataReadWriter()
{
//Reads the Temperature, Pressure & Humidity from the BME280 Sensor
Temperature = mySensor.readTempC();
Pressure = (mySensor.readFloatPressure())/100; // converts the reading in
Pascals to Millibar
Humidity = mySensor.readFloatHumidity();

// sets the LCD to print the next set of data from the top left corner of
the screen
lcd.setCursor(0, 0);
// prints a formatted Temperature string to 1 decimal place &
concatenates '°C' to the end
lcd.print(String(Temperature, 1) + (char)223 + "C");

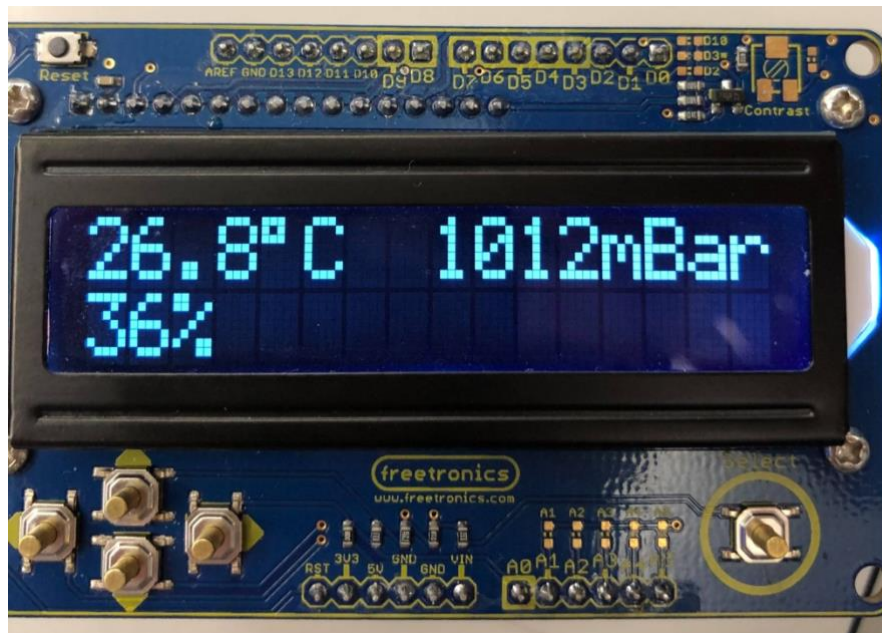
// sets the LCD to print the next set of data from the middle of the top
part of the screen
lcd.setCursor(8, 0);

```

```
// prints a formatted Pressure string to 0 decimal places and
concatenates the units 'mBar' to the end
lcd.print(String(Pressure, 0) + "mBar");

// sets the LCD to print the next set of data from the bottom left part
of the screen
lcd.setCursor(0, 1);
// prints a formatted Humidity string to 0 decimal places and
concatenates a % symbol to the end
lcd.print(String(Humidity, 0) + "%");

}
```



This picture shows the LCD monitor displaying the necessary data. It displays the temperature, pressure and humidity in the correct format.

## Task 8 – Datalogging Weather Station

This exercise builds on the weather station introduced in Exercise 7 but extends it to record a timestamp alongside the weather conditions as well displaying the parameters on the LCD screen. The simple solution will be to write this to the serial monitor. It will also write to an SD card.

Before programming the Arduino for complex projects such as this, make a note of the pin allocations used in the project. Copy the following table into your submission and complete the missing fields.

The data should be written to the serial monitor (and, optionally, the SD Card) every 10 seconds. The program does not need to take any input from the serial monitor. Note that the LCD display should still be refreshed every second.

The data should be written to a file named DATALOG.TXT or (for extra marks) DATAxxxx.TXT where xxxx is a sequence number, picked such that the program writes to the first unused filename in the sequence.

Add a feature to adjust the backlight on the display using the up and down keys on the LCD keypad.

Add a feature to change the units for the LCD temperature display using the left arrow key.

Add a feature to display minimum and maximum temperatures on the LCD display. These should take the place of the humidity and clock displays on the second line of the display

The minimum and maximum temperature displays should be shown in the current temperature mode (C,F or K). Pressing the right arrow key should toggle between display of min/max temperature on line 2 of the LCD and display of humidity and time.

	Communication Bus used	Arduino Chip Select pin or I2C address
SD Card	SPI	Pin 10
Realtime Clock	I2C	0x68
BM280	I2C	0x77
Display	Direct Connection 8,9,4,5,6,7	Enable = 9 Backlight = 3

Table with the chips we used as well as the type of communication bus they use.

```

#include <stdint.h>

#include "SparkFunBME280.h"

#include "Wire.h"
#include "SPI.h"
#include "RTCLib.h"
#include <LiquidCrystal.h>
#include <SD.h>

//Global sensor object
BME280 mySensor;

unsigned long previousTime = 0;
unsigned long currentTime = 0;
const int interval = 1000; // 1000 ms between actions
int i = 0;

//Creates the variables used in the program
float Temperature;
float displayTemp;
float maxTemp;
float minTemp;
String displayString;
float displayMinTemp;
float displayMaxTemp;
float Pressure;
float Humidity;
String Hour;
String Minute;
String Second;
int counter = 1;
int counterTwo = 1;
String unit;

File dataFile;

char fileName[16];

const int RIGHT_KEY = 1; // define some symbolic names for the keys
const int UP_KEY = 2;
const int DOWN_KEY = 3;
const int LEFT_KEY = 4;
const int SELECT_KEY = 5;

int backlight = 127;
bool flag = true;

const int chipSelect = 10;

RTC_DS1307 rtc;

LiquidCrystal lcd(8,9,4,5,6,7);

```

```

void setup()
{
  Serial.begin(9600);

  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.print("Initializing SD card...");

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    while (1);
  }
  Serial.println("card initialized.");

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }
  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was
    compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to
    set
    // November 12, 2019 at 3pm you would call:
    // rtc.adjust(DateTime(2019, 11, 12, 15, 24, 0));
  }

  FileBuilder();

  //Initializes the BME280 sensor to run using an I2C protocol
  mySensor.settings.commInterface = I2C_MODE;
  mySensor.settings.I2CAddress = 0x77;

  /***Operation settings*****/
  mySensor.settings.runMode = 3; // 3, Normal mode
  mySensor.settings.tStandby = 0; // 0, 0.5ms
  mySensor.settings.filter = 0; // 0, filter off
  //tempOverSample can be:
  // 0, skipped
  // 1 through 5, oversampling *1, *2, *4, *8, *16 respectively
  mySensor.settings.tempOverSample = 1;
  //pressOverSample can be:
  // 0, skipped
  // 1 through 5, oversampling *1, *2, *4, *8, *16 respectively
  mySensor.settings.pressOverSample = 1;
  //humidOverSample can be:
  // 0, skipped
  // 1 through 5, oversampling *1, *2, *4, *8, *16 respectively
  mySensor.settings.humidOverSample = 1;

```

```

    // Initializes the LCD screen & tells the Arduino the dimensions of the
screen
    lcd.begin(16, 2);

    delay(10); //Make sure sensor had enough time to turn on. BME280
requires 2ms to start up.

    // Gets the sensor to start taking readings
mySensor.begin();

    Serial.print("Welcome to the hyper advanced weather monitor\n\n");
    Serial.print("Temperature: \tPressure: \tHumidity:\n\n");

    SensorDataReader();
    minTemp = Temperature;
    maxTemp = Temperature;
    // variable used to help keep track of time so readings can be made every
second
    previousTime = millis();
}

void loop()
{
    currentTime = millis(); // record current time

    // if statement to check if 1 second interval has elapsed
    keyPad();
    if (currentTime - previousTime >= interval)
    {
        previousTime = currentTime; // update previous time
        //Calls the functions which reads data from the BME280 sensor, formats
it, and writes the formatted data to the LCD
        SensorDataReader();
        TimeReader();

        //will check to see if a new minimum or maximum temperature has been
recorded
        minTempChecker();
        maxTempChecker();

        //calls the temperature conversion function
        tempConverter();

        //calls the function to write to the LCD monitor
        LCDWriter();
        i++;
        if (i == 10)
        {
            i = 0;
            //prints the recorded values to the serial monitor
            Serial.print(String(Temperature) + "°C\t\t");
            Serial.print(String(Pressure) + "mBar\t");
            Serial.print(String(Humidity) + "%\n");
            //calls the fileWriter function
            FileWriter();
        }
    }
}

```

```

    //sets the LED backlight to the backlight value
    analogWrite(3, backlight);
}

// Function which formats data from the Arduino modules, and writes the
formatted data to the LCD
void LCDWriter()
{
    // sets the LCD to print the next set of data from the top left corner of
the screen
    lcd.setCursor(0, 0);
    // prints a formatted Temperature string to 1 decimal place &
concatenates '°C' to the end
    lcd.print((String(displayTemp, 1)));
    if(counter != 3){
        //char 223 is used to print the degrees symbol to the LCD monitor if
units are not kelvin
        lcd.print(char(223));
    }
    //will print the specified unit after the temperature is displayed
    lcd.print(unit);

    // sets the LCD to print the next set of data from the middle of the top
part of the screen
    lcd.setCursor(8, 0);
    // prints a formatted Pressure string to 0 decimal places and
concatenates the units 'mBar' to the end
    lcd.print(String(Pressure, 0) + "mBar");

    // sets the LCD to print the next set of data from the bottom left part
of the screen
    lcd.setCursor(0, 1);
    if(counterTwo == 1){
        // prints a formatted Humidity string to 0 decimal places and
concatenates a % symbol to the end
        lcd.print(String(Humidity, 0) + "%");

        lcd.setCursor(8, 1);
        //prints the time to the LCD monitor
        lcd.print(Hour + ":" + Minute + ":" + Second);
    } else {
        //prints the maximum and minimum temperatures
        lcd.print((String(displayMinTemp, 1)) + "/" + (String(displayMaxTemp,
1)) + " ");
    }
}

//Reads the Temperature, Pressure & Humidity from the BME280 Sensor
void SensorDataReader()
{
    Temperature = mySensor.readTempC();
    Pressure = (mySensor.readFloatPressure())/100; // converts the reading in
Pascals to Millibar
    Humidity = mySensor.readFloatHumidity();
}

//will add the correct amount of zeros for the time recorded
String TimeFormatter(String arg)
{
    if (arg.length() == 1)

```



```

    {
        arg = "0" + arg;          //will add a zero beforehand if the number is
one digit long
    }
    return arg;
}

//will get the current time and then format it correctly
void TimeReader()
{
    DateTime now = rtc.now();
    Hour = String(now.hour());
    Hour = TimeFormatter(Hour);

    Minute = String(now.minute());
    Minute = TimeFormatter(Minute);

    Second = String(now.second());
    Second = TimeFormatter(Second);
}

//will write the date,time, temperature, pressure and humidity to the file
created
void FileWriter()
{
    DateTime now = rtc.now();
    dataFile = SD.open(fileName, FILE_WRITE);
    if (dataFile) {
        dataFile.print(now.day(), DEC);
        dataFile.print('/');
        dataFile.print(now.month(), DEC);
        dataFile.print('/');
        dataFile.print(now.year(), DEC);
        dataFile.print(',');
        dataFile.print(now.hour(), DEC);
        dataFile.print(':');
        dataFile.print(now.minute(), DEC);
        dataFile.print(':');
        dataFile.print(now.second(), DEC);
        dataFile.print(',');
        dataFile.print(Temperature, 2);
        dataFile.print(',');
        dataFile.print(Pressure, 2);
        dataFile.print(',');
        dataFile.print(Humidity, 0);
        dataFile.print("\n");
        dataFile.close();
    }
}

//creates sequential file that the data will be written to
void FileBuilder()
{
    int count = 0;
    do
    {
        sprintf(fileName, "DATA%04d.TXT", count);
        count++;
    }
    while(SD.exists(fileName));          //if file exists, add one to
counter and try again

```

```

    }

//function to apply button functionality
void keyPad()
{
    int A0Input = analogRead(A0); // temporary storage for analog input
    if (flag == true)
    {
        //Determine which button was pressed and applies their functionality
        switch (A0Input)
        {
            case 0 ... 100 :
                // RIGHT key pressed;
                counterTwo++;
                //identifies which display
                format to use on the bottom line
                if(counterTwo > 2){
                    counterTwo = 1;
                }
                flag = false;
                return;
            case 101 ... 160:
                // UP key pressed
                backlight = constrain(backlight +
                16,0,255); //increases brightness of screen
                flag = false;
                return ;
            case 161 ... 340:
                // DOWN key pressed
                backlight = constrain(backlight -
                16,0,255); //decreases brightness of screen
                flag = false;
                return ;
            case 341 ... 510:
                // LEFT key pressed
                counter++;
                if(counter > 3)
                    //identifies which unit of
                    temperature to display in
                {
                    counter = 1;
                }
                flag = false;
                return ;
            case 511 ... 750:
                // SELECT key pressed
                return ;
            default:
                return;
        }
    }
    //if no button is pressed flag is set to true so that buttons are not
    repeatedly triggered with one push
    if (A0Input > 800)
    {
        flag = true;
        return 0;
    }
}

//function to convert temperature into kelvin
float kel(float temp)

```

```

{
    float kelTemp = (temp + 273.15);
    return kelTemp;
}

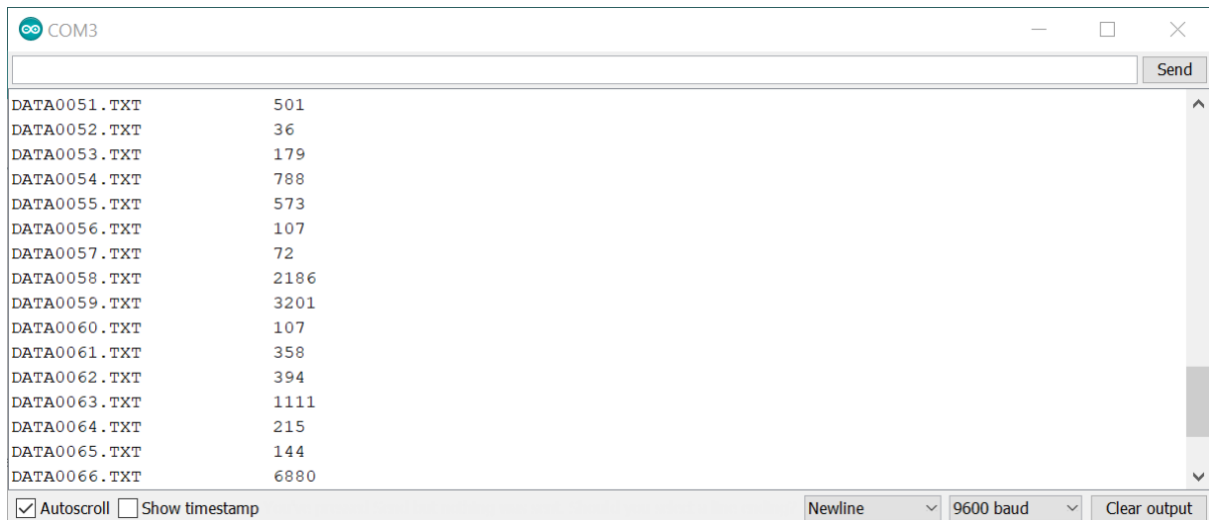
//function to convert temperature into fahrenheit
float fah(float temp)
{
    float fahTemp = (temp*(9/5)+32);
    return fahTemp;
}

//Does temperature conversion for selected temperature scale
void tempConverter(){
    switch(counter){
        case 1:                                //for celcius
            displayTemp = Temperature;
            displayString = (String(displayTemp, 1));
            unit = ("C");
            displayMinTemp = minTemp;
            displayMaxTemp = maxTemp;
            break;
        case 2:                                //for fahrenheit
            displayTemp = fah(Temperature);
            displayString = (String(displayTemp, 1));
            unit = ("F");
            displayMinTemp = fah(minTemp);
            displayMaxTemp = fah(maxTemp);
            break;
        case 3:                                //for kelvin
            displayTemp = kel(Temperature);
            displayString = (String(displayTemp, 1));
            unit = ("K");
            displayMinTemp = kel(minTemp);
            displayMaxTemp = kel(maxTemp);
            break;
    }
}

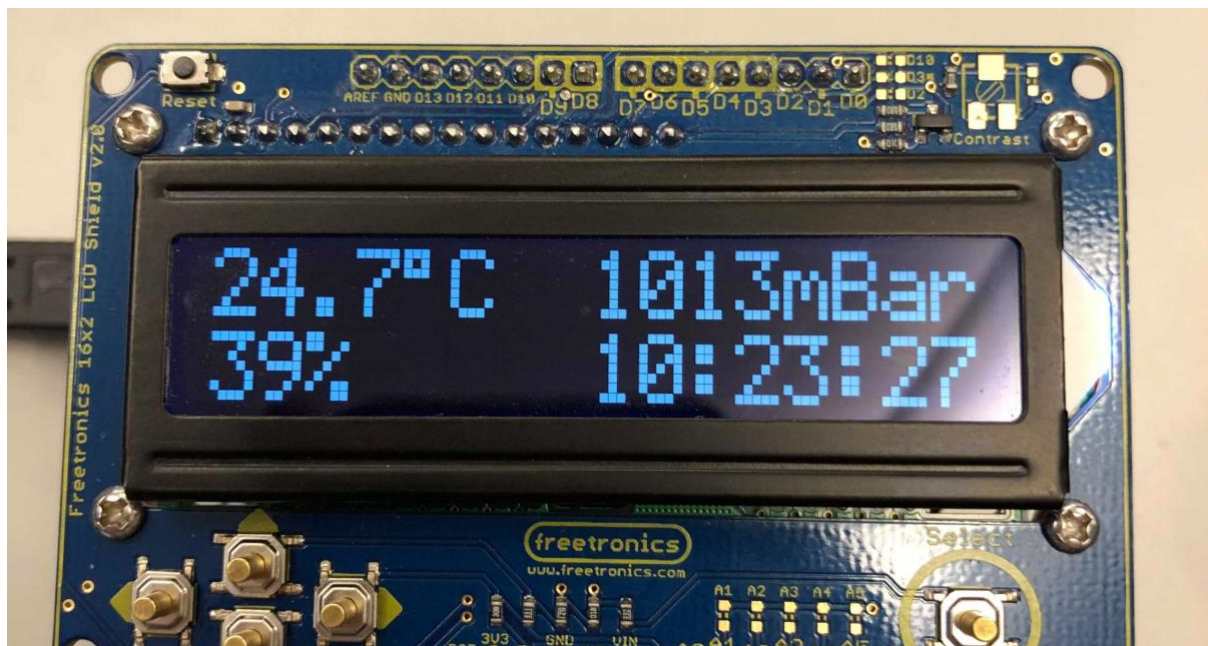
// Checks if temperature read is lower than minTemp and sets minTemp to
// equal temp if temp was lower
void minTempChecker()
{
    if (minTemp > Temperature)
    {
        minTemp = Temperature;
    }
    return;
}

// Checks if temperature read is higher than maxTemp and sets maxTemp to
// equal temp if temp was higher
void maxTempChecker()
{
    if (maxTemp < Temperature)
    {
        maxTemp = Temperature;
    }
    return;
}

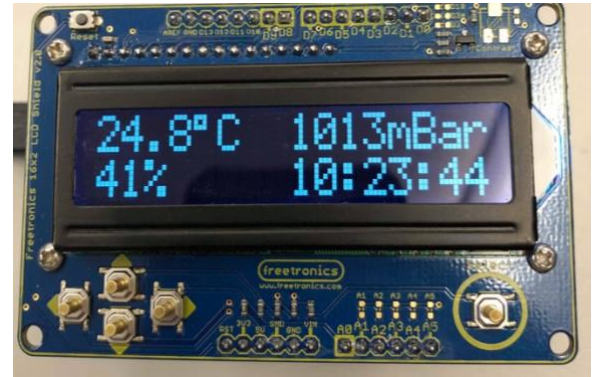
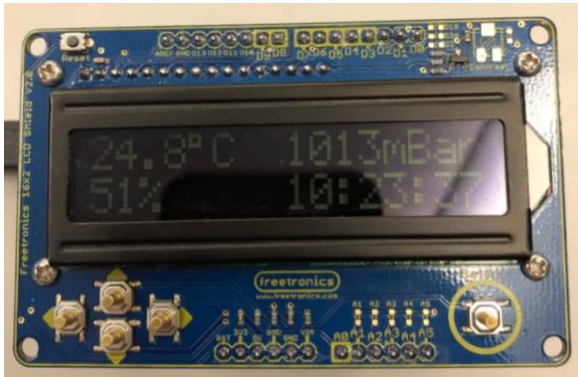
```



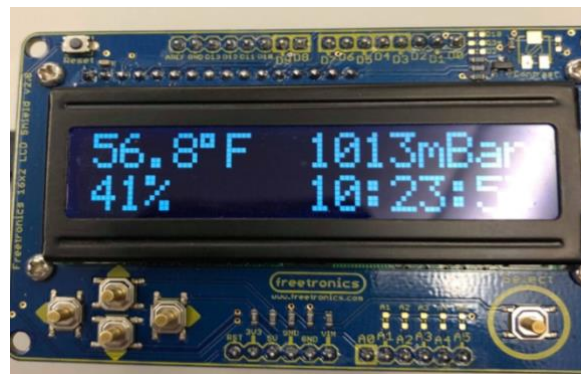
A program running which shows that a new data file is being opened and written to every time the program is run.



Picture showing the data recorded on the LCD screen. This will update every second along with the clock.



These two images show that the LCD screen changes in brightness if either the up or down button is pressed. The image on the left is when the screen is at minimum brightness and the image on the right is when it is at maximum brightness.



This shows the temperature switching to a different unit (Fahrenheit in this case) when the left button is pressed. If the button is continuously pressed it will keep looping around the different temperature units.



This shows the LCD monitor switching to show the minimum and maximum temperature that has been recorded so far. These values will update straight away if they change and will also change what unit they are in if the left button is clicked.