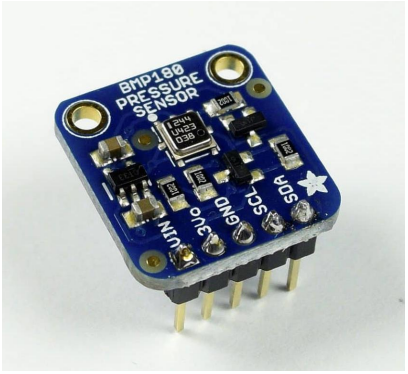
Temperature & Pressure Monitoring project

SBC-21-1037 Haolin Zhuang
SBC-21-1026 Yuxuan Wang

Abstract

This paper details the creation of a monitoring system using an Arduino board, tracking temperature and pressure and displaying the data on seven-segment displays. The use of the BMP180 sensor, combined with a clear library, ensures accurate and swift data presentation. The document discusses the programming and circuit design, emphasizing careful sensor and circuit choices. A thorough demonstration confirms the system's effectiveness in real-time temperature and pressure monitoring.



Content

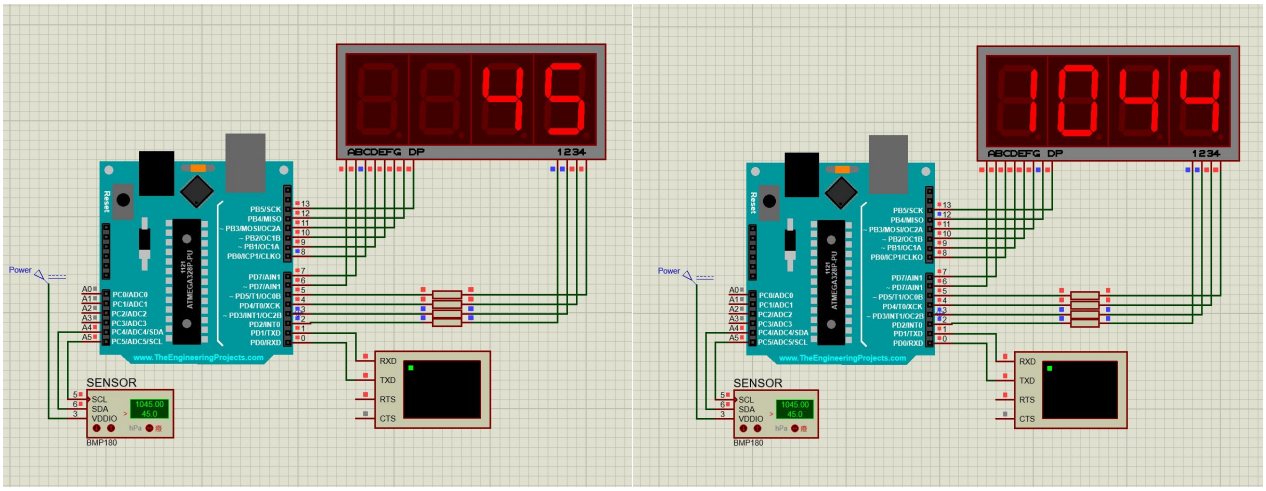
Program code

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include "SevSeg.h"

Adafruit_BMP085 bmp;
SevSeg sevseg;                                  //Call all library files and mark their usage.

byte numDigits = 6;                             // To display the full pressure on the 4-bit 7-segment displays,
set the digit count to 6.
byte digitPins[] = {2, 3, 4, 5};                //Connect the common anode of the seven-segment display to the
Arduino pin.
byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13 }; //The pins corresponding to the seven-segment display segments
a, b, c, d, e, f, g, and dp
bool resistorsOnSegments = false;               //  Resistors are connected to the digital pins.
byte hardwareConfig = COMMON_ANODE;             // A common anode 7-segment display is being used.
bool updateWithDelays = false;                  // Default 'false' is Recommended
bool leadingZeros = false;                      // No need to keep leading zeros.
bool disableDecPoint = false;                   //Utilize the decimal point functionality.

int i;
volatile bool enterInterrupt = false;           //Set variable parameters.



void setup() {
//Initialize the seven-segment display.
  sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments,
  updateWithDelays, leadingZeros, disableDecPoint); //Add the previously set parameters.
  sevseg.setBrightness(6);                      //Set the brightness of the seven-segment display.
 if (!bmp.begin()) {
    while (1) {}                                //begin bmp180
  }
}


void loop() {

  if (bmp.readTemperature() >= 50 && bmp.readPressure() >= 105000){
  //Enter the interrupt if the current temperature and pressure are greater than the set values of 50 and 105000,
respectively.
    enterInterrupt = true;
    } else{
  //Otherwise, enter the normal detection loop.
```

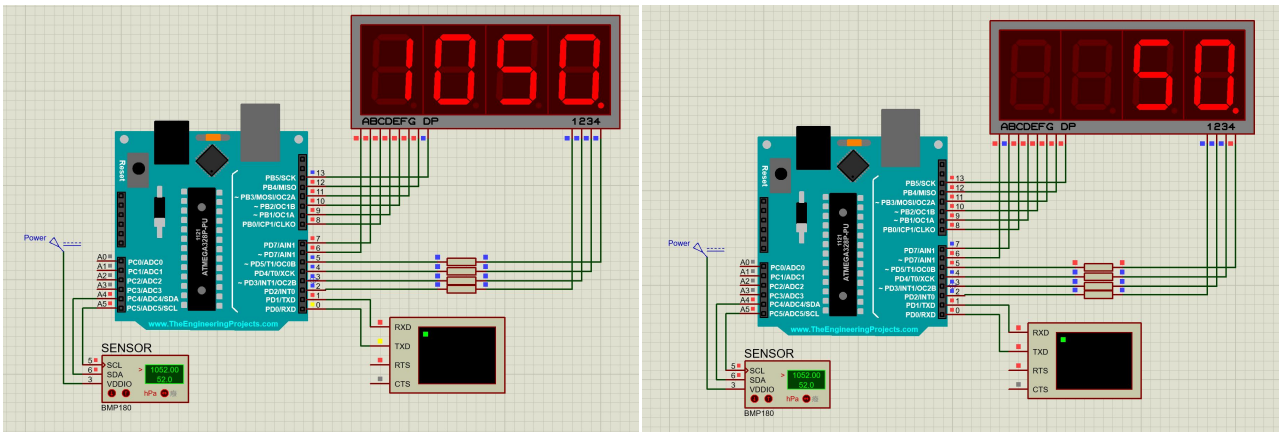
```
    enterInterrupt = false;
    }

  if (!enterInterrupt){
  //Non-interrupt normal display loop.
    i=bmp.readTemperature();
    i=i*100;                                    //Extract the current temperature value and multiply it by 100
to display it correctly on the seven-segment display.
    sevseg.setNumber(i);                        //Use the seven-segment display to show the data.
  for(int k=0;k<32000;k++){
    sevseg.refreshDisplay();                    //Rapidly refresh the 7-segment display multiple times to ensure
proper display on Proteus.
    }

    sevseg.setNumber(bmp.readPressure());        //Extract and display the current pressure value.
  for(int k=0;k<32000;k++){
    sevseg.refreshDisplay();                    //refresh the 7-segment display
    }
  }

  if (enterInterrupt){
  //Interrupt loop content.
    sevseg.setNumber(5000,2);                   //Display the set maximum alarm temperature, indicating it with
a decimal point at the end.
  for(int k=0;k<32000;k++){
    sevseg.refreshDisplay();                    //refresh the 7-segment display
    }

   sevseg.setNumber(105000,2);                  //Display the set maximum alarm pressure
  for(int k=0;k<32000;k++){
    sevseg.refreshDisplay();                    //refresh the 7-segment display
    }

  if (bmp.readTemperature() < 50 || bmp.readPressure() < 105000){
  //Exit the interrupt loop if either the temperature or pressure falls below the set values.
    enterInterrupt = false;
    }
  }

}
```

Circuit design & simulate with Proteus

Due to the necessity of connecting a 4 bit 7-segment display to pin 12 on the Arduino for output, we have opted for an integrated temperature and pressure sensor. This sensor simplifies the setup, requiring only two interfaces and a power supply. Utilizing a library compatible with Arduino, we can effortlessly employ the read function to obtain accurate temperature and pressure readings.



Through the connection of code and simulated circuitry, the entire device consistently displays temperature and atmospheric pressure at regular intervals upon startup. As you adjust the temperature and pressure by manipulating the button on the BMP180 sensor, the seven-segment display promptly responds, providing real-time updates on the latest temperature and pressure values. When both temperature and pressure surpass the predetermined thresholds, the display swiftly cycles to showcase the set threshold temperature and pressure alternately. Moreover, the decimal point on the last digit illuminates momentarily, offering users a clearer indication.



During the programming of the seven-segment display, the SevSeg library was employed to significantly reduce the length of the code. The use of simple setNumber statements within this library allowed for the convenient display of the desired data on the seven-segment display.

Problem and solution

However, several issues arose in the process:

1)The incompatibility between the SevSeg library and the use of delay. When utilizing the SevSeg library to streamline the code, the rest of the code cannot include delay. Therefore, alternative methods such as using `int i` were employed for countdowns, relying on the time required for the code itself to execute as a form of delay.

2)Due to the operational principle of a 4-digit seven-segment display, where each digit rapidly blinks in sequence to achieve the desired display, it is essential to simulate this effect in Proteus. However, the persistence of vision effect cannot be replicated in the simulation. Therefore, the mentioned system delays and adjustments to the trigger time of the seven-segment display are necessary to achieve accurate representation.

Conclusion

This experiment aimed to design an Arduino-based temperature and pressure monitoring system with real-time data displayed on a 4 bit 7-segment display. To simplify the code, the SevSeg library was utilized, allowing for easy data display on the seven-segment display. However, this introduced an issue with the incompatibility between the SevSeg library and the delay function. To address this, a clever workaround was implemented, simulating delays by counting down within the runtime of the code.

The key to the entire system was the use of the BMP180 sensor, requiring only two interfaces and a power supply to effortlessly obtain temperature and pressure data. Additionally, to simulate the operational principle of the 7-segment display in Proteus, system delays and adjustments to the trigger time of the seven-segment display were introduced for an accurate representation of the blinking effect.By integrating these techniques, a robust monitoring system was successfully established, displaying real-time temperature and pressure. When reaching predefined temperature and pressure thresholds, the system rapidly responds, refreshing the seven-segment display to clearly alert the user.