**ARDUINO BASED**

**HEAT AND PRESSURE SENSOR ARRAY**

For:

Dr. Orban, Physics Department

OSU Marion

By:

Joshua Simmeron

01MAY15

Table of Contents

[Summary 1](#_Toc421719868)

[Temperature Sensor 2](#_Toc421719869)

[Summary 2](#_Toc421719870)

[Thermal Resister 2](#_Toc421719871)

[Thermocouple 2](#_Toc421719872)

[Calibration 3](#_Toc421719873)

[Pressure Sensor 3](#_Toc421719874)

[Software 3](#_Toc421719875)

[ADC Readings 3](#_Toc421719876)

[FIR Filters 3](#_Toc421719877)

[Code 4](#_Toc421719878)

[Wiring Diagrams 7](#_Toc421719879)

[Pressure Sensor 7](#_Toc421719880)

[Temperature Sensor 7](#_Toc421719881)

[Parts List 8](#_Toc421719882)

[Drawings 9](#_Toc421719883)

[Case Body 9](#_Toc421719884)

[Case End 10](#_Toc421719885)

# Summary

The project goal was to build an Arduino based sensor array that could be used to measure both the temperature and pressure for the Physics 1250 - Heat Engine Lab.

The initial design utilized a thermal resister due to the simplicity of wiring it. An error in the parts listing led to ordering of a thermocouple instead. It was decided to proceed with the thermocouple because it was the only sensor in its price range with an appropriate package for the application desired. The switch from the thermal resister to a thermocouple necessitated the addition of a whetstone bridge integration that was then routed through an operational amplifier.

After the first build the accuracy on the pressure sensor was found to be insufficient so the MPX5500 pressure sensor was replaced with a MPX5010. Providing the resolution needed for the lab.

On the final build both the pressure and temp sensors were showing noise that resulted in fluctuating readings. The issue was traced to fluctuating power from the USB power supply. Filter capacitors were added to reduce the noise and finally a moving average FIR filter was implemented through software.

A serial interface was selected to request and obtain readings from the sensor. This allows the sensors to be accesses on either a PC utilizing the freeware application PuTTY or a Mac using the built in application terminal. For either application the user will initiate a serial communication line at 9600 baud.

The final build displays temperature in Celsius from -18 to 102 in tenth of a degree increments and pressure in kPa to a hundredth. The pressure response is near instant. The temperature sensor has a noticeable lag when measuring air temperature. Lag is mostly due to the housing holding heat additionally by higher than ideal current through the sensor. This lag is imperceptible when the sensor was used to measure water temp, due to water’s higher heat transfer rate. As built the temperature sensor requires a 10 min warm up period before it will accurately read room temp. This period is not needed if measuring water temperature.

# Temperature Sensor

### Summary

Thermal resisters were initially selected for the project due to the simplicity of the hardware implementation at the expense of more complicated software. While sourcing parts, only one was in an appropriate package for the application, for a reasonable price. Upon delivery it was discovered that the part was in fact a thermocouple. It was decided to utilize the thermocouple because it has a more linear response curve and the price of adding the required op-amp was lower than that of a new sensor. This decision required changing the hardware from a voltage divider to an amplified whetstone bridge. In hindsight it would have been wiser to stick with the original design implementing a thermal resister.

### Thermal Resister

Thermal resistors are implemented by using a voltage divider with the output run directly to one of the Arduino’s ADC ports. Software is based off the β parameter equation which is a simplified version for the Stein-Hart equation.

Stein-Hart equation:

β parameter equation:

The β parameter equation is then used to find the value of B for calibration and then used with the B value to determine temperature in operation.

### Thermocouple

Over the temperature range required for this experiment the thermocouple provides a *near* linear response curve and higher sensitivity than a thermal resister. This is done with the added complications of having to implement a Wheatstone bridge and the then amplifying the signal before routing it to the Arduino’s ADC. Care must also be taken to limit current through the sensor to < 1 mA to limit self-heating of the sensor. That limit was not obtained in this implementation exacerbating both the warm up period and slow response time of the sensor as built. The software is an implementation of the formula of a line using the values from the calibration.

### Calibration

Calibration of temperature sensor is done utilizing two producible known temperatures. Boiling point of water (adjusted for elevation) is found by boiling distilled water. And freezing point of water is found by making a crushed ice bath of distilled water (water level must be below crushed ice level). Leave the sensor in each bath until it stabilizes and then record the readings. Both readings are then used with the formula provided to calibrate the sensor.

The current temperature range is from -20 to 102 Celsius in tenths of a degree. The temp range can be expanded or contracted by varying the gain resister on the op-amp. Any increase or decrease to the temperature range will have a reverse effect on the sensitivity.

# Pressure Sensor

 Initially a MPX5500 was implemented for the pressure sensor. While functional it had limited range at the pressures needed for the lab so it was substituted with a MPX5010. Software is implemented using the formula below with [offset = zero pressure reading].

|  |  |
| --- | --- |
|  | Signal output |
|  | Supply voltage (5V) |
|  | Pressure in kPa |
|  |  at zero pressure |

For greater ranges a MPX5050 would expand the range by a factor of five, or the MPX 5500 would expand the range by a factor of ten. Both would reduce sensitivity by a similar factor.

# Software

### ADC Readings

Sensor outputs are analog and final output is digital, necessitating Analog to Digital Converters or ADC’s. ADC output is a function of the signal in the formula below. Note: Areff is set to 3.3 in this design

### FIR Filters

Moving average Finite Impulse Response Filters were utilized to manage signal noise. The program continuously reads sensor values into a revolving array of the 150 most recent readings. When an output is called for the program averages all the values in the array to produce a stable output

# Code

/\*

Written By: Joshua Simmerson simmerson.4@buckeyemail.osu.edu

Date: JUN15

Summary:

Display temperature and pressure reading, while performing OSU's Phyisics 1250 heat engine lab.

\*/

// Variables

int indexFIR = 0; // Location index for FIR filters

const int num = 150; // Number of samples for analog reads and FIR filter

float sample = 0; // Return value of readAvg for temp and pressure readings

byte serialRead = 0; // Input from terminal

double timer = millis(); // Timer to cycle outputs

#define tPin A1 // Temp pin

float temp = 999; // Final temperature value

int tFIR[num]; // Temp FIR filter array

#define pPin A3 // Pressure pin

float pZoffset = 0; // Pressure zero offest

float pressure = 999; // Final pressure value

int pFIR [num]; // Pressure FIR filter array

void setup()

{

 Serial.begin(9600); // Start serial communications

 analogReference(EXTERNAL); // Aref pin tied to 3.3V

 delay(40); // Wait for pressure sensor to stabilize

 sample = 0;

 for (int i = 0; i < num; i++) // Calculate average zero offset for pressure sensor

 {

 sample += analogRead(pPin);

 }

 sample /= num;

 pZoffset = sample; // Set offset to zero pressure reading

}

void loop()

{

 /\*

 This section is for a serial interface

 if (Serial.available()) // Check for input from terminal

 {

 serialRead = Serial.read(); // Read input

 if (serialRead==49) // Check for flag to execute, 49 is asci for 1

 {

 calcPrint(); // Calculate moving averages and print results to terminal

 }

 }

\*/

 if (millis() > timer) { // Print every 0.5 seconds

 calcPrint(); // Calculate moving averages and print results to terminsl

 timer += 500; // Increment timer

 }

 analogRead(tPin); // Pre-read

 tFIR[indexFIR] = analogRead(tPin); // Update temp FIR

 analogRead(pPin); // Pre-read

 pFIR[indexFIR] = analogRead(pPin); // Update pressure FIR

 indexFIR = (++indexFIR) % num; // Reset FIR index back to 0 when end of array is reached

}

//Average values in array to generate moving average Finite Impule Response Filter

float avgFIR(int FIR[])

{

 float avg = 0;

 for (int i = 0; i < num; i++)

 {

 avg += FIR[i];

 }

 avg /= num;

 return avg;

}

void calcPrint()

{

 // calculate pressure

 sample = avgFIR(pFIR);

 pressure = ((sample - pZoffset) \* .007161458)+100; // 100 is estimated atmospheric pressure

 // Calculate temp

 sample = avgFIR(tFIR);

 temp = .118132 \* sample - 18.6736;

 // Print

 Serial.print('\n');

 Serial.print("Temerature: ");

 Serial.print(temp, 1);

 //Serial.print(analogRead(tPin));

 Serial.print("C - Pressure: ");

 Serial.print(pressure);

 Serial.print("kpa");

 Serial.println('\n');

}

# Wiring Diagrams

### Pressure Sensor



# Temperature Sensor



# Parts List

**Arduino Uno** $28.30

<http://www.digikey.com/product-detail/en/A000066/1050-1024-ND/2784006>

**Temp Probe** $10

[http://www.amazon.com/AGPtek-Stainless-Thermistor-Sensor-Temperature/dp/B008YP1D04/ref=sr\_1\_1?ie=UTF8&qid=1426199574&sr=8-1&keywords=Thermistor+Sensor+Probe](http://www.amazon.com/AGPtek-Stainless-Thermistor-Sensor-Temperature/dp/B008YP1D04/ref%3Dsr_1_1?ie=UTF8&qid=1426199574&sr=8-1&keywords=Thermistor+Sensor+Probe)

**Proto-Board** $15

[http://www.amazon.com/Adafruit-Proto-Shield-Arduino-Kit/dp/B00N426HCG/ref=sr\_1\_1?ie=UTF8&qid=1426200319&sr=8-1&keywords=Adafruit+Proto+Shield](http://www.amazon.com/Adafruit-Proto-Shield-Arduino-Kit/dp/B00N426HCG/ref%3Dsr_1_1?ie=UTF8&qid=1426200319&sr=8-1&keywords=Adafruit+Proto+Shield)

**Pressure sensor - MPX5010DP-ND** $12.83

<http://www.digikey.com/product-detail/en/MPX5010DP/MPX5010DP-ND/464054>

**750 ohm resistor** $0.10

<http://www.digikey.com/product-detail/en/MFR-25FBF52-750R/750XBK-ND/12987>

**.33uf capacitor** $0.32

<http://www.digikey.com/product-detail/en/FK24X7R1H334K/445-5263-ND/2256743>

**USB cable** $4.67

[http://www.amazon.com/AmazonBasics-USB-2-0-Cable--Male/dp/B001TH7GUA/ref=sr\_1\_1?s=pc&ie=UTF8&qid=1426206153&sr=1-1&keywords=usb+ab+cable](http://www.amazon.com/AmazonBasics-USB-2-0-Cable--Male/dp/B001TH7GUA/ref%3Dsr_1_1?s=pc&ie=UTF8&qid=1426206153&sr=1-1&keywords=usb+ab+cable)

**OP-AMP: INA126PA-ND** $3.33

<http://www.digikey.com/product-detail/en/0/INA126PA-ND>

**RES 100 OHM 1/4W 0.1% AXIAL** $0.46 x3

<http://www.digikey.com/product-detail/en/0/100ADCT-ND>

# Drawings

### Case Body



### Case End

