Sensor Interface and Calibration Report

Harrison Gietz
Louisiana State University

December 18th, 2020

Principle of Operation:

There are two main science goals of the sensor interface report: one is to assemble and calibrate two circuits that read temperature and pressure respectively. The second goal is to develop software in the Arduino IDE that logs the output of these circuits onto an SD card. This logged data should include GPS timestamps, along with other parsed GPS information.

The pressure circuit will use a 1210 PC piezoresistive pressure sensor, which contains 4 small silicon resistors that change their resistance values depending on the mechanical strain they experience. In the case of the 1210 PC, this mechanical strain occurs from small changes in pressure. As such, the sensor is designed such that small changes in the pressure modify the voltage difference between two outputs on the sensor. It is this voltage difference that will be measured by the Arduino Mega and used to calibrate the pressure circuit.

The temperature circuit uses a 1N457 diode, which changes its resistance depending on its temperature. As stated in the LaACES sensors and signal conditioning document, the typical resistance range for this type of diode is 0.45-0.8V. Since the output of this circuit will be measured by the Arduino Mega, signal conditioning will be required to convert this signal to the Arduino's analog pin range (0-5V).

Specific goals of the report are stated below:

1. Assemble circuits for measuring both pressure and temperature.
    a. Use proper signal conditioning, ensuring that the bases of the analog readings are 0.00 ± 0.03V and the spans are 5.00 ± 0.03V (this allows for optimal precision when taking measurements over the full range of Arduino ADC values).

2. Calibrate circuits to find equations that model the relationship between the measured ADC value and the actual temperature.
    a. Take into account the uncertainty in the values given by these equations.
        i. For temperature, the calibration should be accurate within 1.0°C.
        ii. For pressure, the calibration should be accurate within 25mmHg.

3. Use Arduino IDE to design code that writes the pressure and temperature to an SD file each time a useful GPS NMEA sentence is received. The pressure and temperature should be timestamped using the parsed NMEA sentences, and various parts of the parsed NMEA sentences should be included with each data entry.
    a. This data should be recorded in a .csv file, with labelled columns at the top.
    b. A new file with a new name should be formed every 150 data entries, so that each file does not get too large.
    c. Each data entry should include the date, time, latitude, longitude, the status of the GPS fix, the number of satellites connected, and the pressure and temperature.

d. If there is no available data for a particular entry (for example, if there is no fix, there is no latitude or longitude data), the code should "skip" this entry and insert a value that clearly indicates that the data was not available.

Electrical Design:

Sensors:

The sensor used for the temperature circuit was the 1N457 diode, and the sensor used for the pressure circuit was the model 1210 PC board mountable pressure sensor.

1210 PC:

This sensor performs optimally in temperatures from 0 to 50°C, where its largest uncertainty in a reading is ±1.0%. The sensor can operate within the range of -40 to 125°C. The relationship between the pressure and voltage difference measured is linear using this sensor, and the data sheet reveals that this linear relationship is true within ±0.1%.

1N457:

Because it is a diode, this sensor is polar and must be connected to the circuit appropriately in order to function. As stated in the Diode-based Temperature Measurement document (see citations), there is a linear relationship between forward voltage and temperature, and as such, diodes can be used for effective temperature telemetry. Because 1N457 diodes are not typically used for temperature telemetry, limited information as available regarding the linearity and precision of measurements of temperature taken using this sensor. The data sheet states that a minimum storage temperature is -65°C, while a maximum operating junction temperature sits at 175°C. Based on this, it is safe to assume that the diode will be functioning properly within the range of temperature that will be measured in this calibration (~0°C to ~100°C)

Sensor Interfacing:

When using the analog pins on the Arduino Mega, the input can vary from 0V to 5V, and thus it is imperative to condition the outputs of the circuits such that they have a base at 0V and a span of 5V (see goal 1.a).

Temperature Signal Conditioning Calculations:

The diode used in the temperature circuit can create a voltage difference of 0.45V to 0.8V depending on its temperature. As such, a signal conditioning circuit must be made to adjust the output voltage of the diode. If we plot the desired voltage output vs the original voltage output, we get a function representing the transformation that the signal conditioning needs to provide (as shown in **Figure 1** on the following page):

From this, we gather that Vout = 14.286(Vin) – 6.4286

Using circuit analysis on the temperature circuit schematic in **Figure 2**, this equation can be compared to the equation given on the LaACES sensors and signal conditioning document (**Equation 1** on the following page).

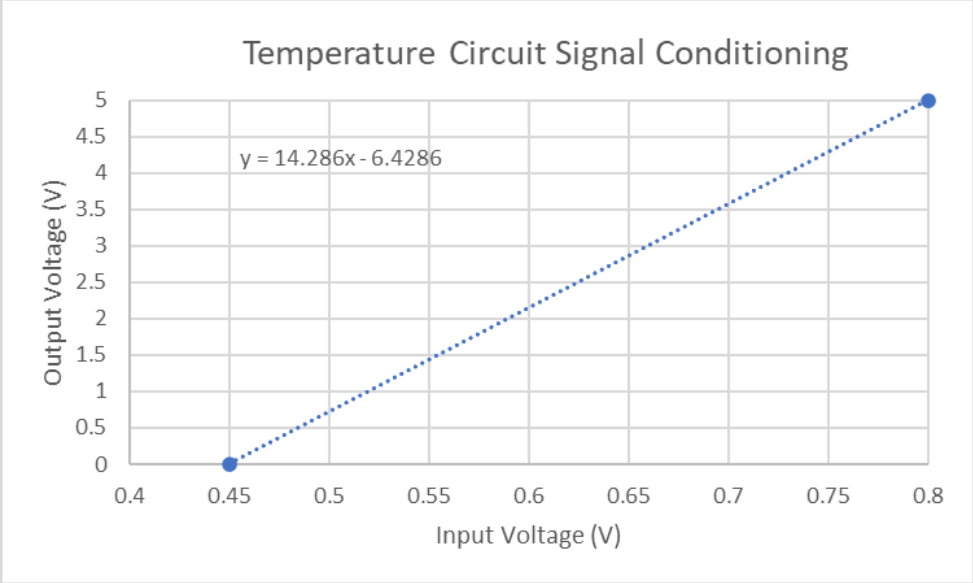


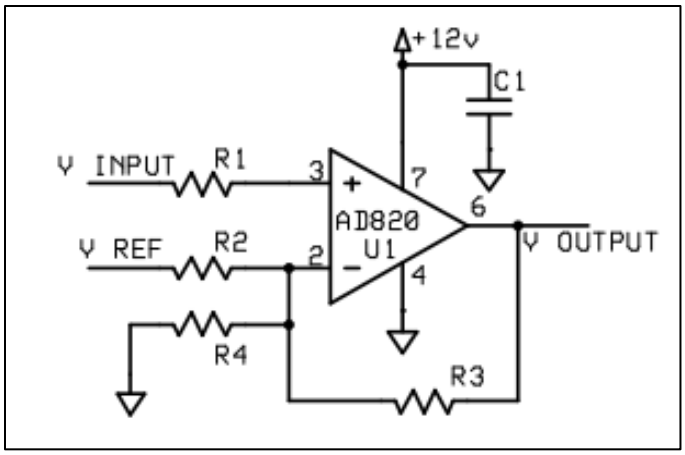
Figure 1 (above): The signal conditioning graph for the temperature circuit. This is the ideal signal conditioning that circuit will provide, converting voltages from a range of 0.45V-0.8V to a range of 0.0V-5.0V. Note that this does not reflect what the outcome of the signal conditioning circuit turned out to be, but is just the projected goal of the circuit.

Equation 1

$$Vout = Vin\left(1 + R3\left(\frac{1}{R2} + \frac{1}{R4}\right)\right) - Vref\left(\frac{R3}{R2}\right)$$

In this case, we can substitute R4 to be 10kΩ since this is a common resistor type, and Vref will be 3.3V from the Arduino Mega. As such:

Figure 2 (right): A diagram of the temperature circuit. V input is adjusted based on the temperature of the 1N457 diode (see **Figure 3**). V ref is the 3.3V output from the Arduino Mega, and V output plugs into the Analog pin of the Arduino Mega for taking measurements. This circuit converts the 0.45-0.8V signal of V input to a 0-5V signal for the Arduino to read, provided the correct resistors are used.



$$14.286 = 1 + R3\left(\frac{1}{R2} + \frac{1}{R4}\right)$$

$$13.286 = R3\left(\frac{1}{R2} + \frac{1}{10k}\right)$$

As well:

$$Vref\left(\frac{R3}{R2}\right) = 6.4286$$

$$R3 = R2(1.94806)$$

Solving this series of equations for R3 and R2, we get

R2 = 60.2kΩ

R3 = 117.3kΩ

Since these are not exact resistor quantities, we can use the potentiometers along with resistors to get the exact value we need. Note that the potentiometers need to be connected such that the middle and a side are used (not two opposite sides). This makes it so that turning the knob on the potentiometers directly affects the resistance between the two sides, rather than dividing the voltage.

For R2, a 47kΩ resistor was used in series with the 10kΩ potentiometer. It was originally thought that the resistance required for R2 was lower, so the max resistances from this setup was only 57kΩ. However, as we will see on page 6, the circuit was still relatively accurate. Unfortunately, with remote working, this issue could not be addressed once it was discovered.

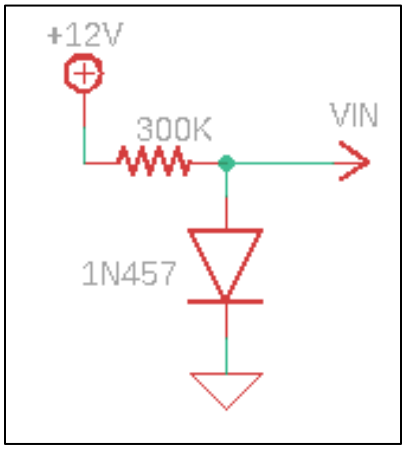


Figure 3 (above): The schematic for the setup of the V input of the temperature circuit (**Figure 2**). Note that during assembly, the 100kΩ potentiometer was used in place of the diode in order to test different resistances that the diode might provide at various temperatures.

For R3, a 100kΩ resistor was used in series with the 20kΩ potentiometer, so this resistance value could be adjusted to the exact value needed.

Pressure Signal Conditioning Calculations:

To interpret the two outputs of the 1210 PC pressure sensor, their voltage difference must be used. The schematics of the 1210 PC sensor and the pressure circuit can be seen in **Figure 4** and **Figure 5** respectively on the following page. The equation relating the output and the inputs of the circuit in **Figure 5** is given by circuit analysis, and found in the LaACES sensors and signal conditioning document once again. This is given as **Equation 2** on the following page.

NOTE: This sample report includes discussion of a pressure sensor and corresponding amplifying circuit that is no longer used as part of the LaACES materials. Students should only be working with a temperature sensor in the current program.

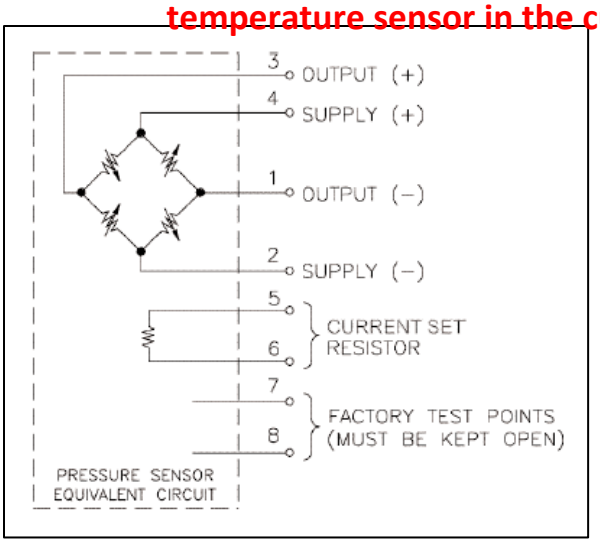


Figure 4 (above): The schematic for the 1210 PC pressure sensor. In the case of the circuit set up in **Figure 5**, INPUT V-A was provided by "OUTPUT+" on this diagram, and INPUT V-B was provided by "OUTPUT-". The "SUPPLY+" on this diagram was provided by the 12V+ power supply, and the "SUPPLY-" was connected to the ground.

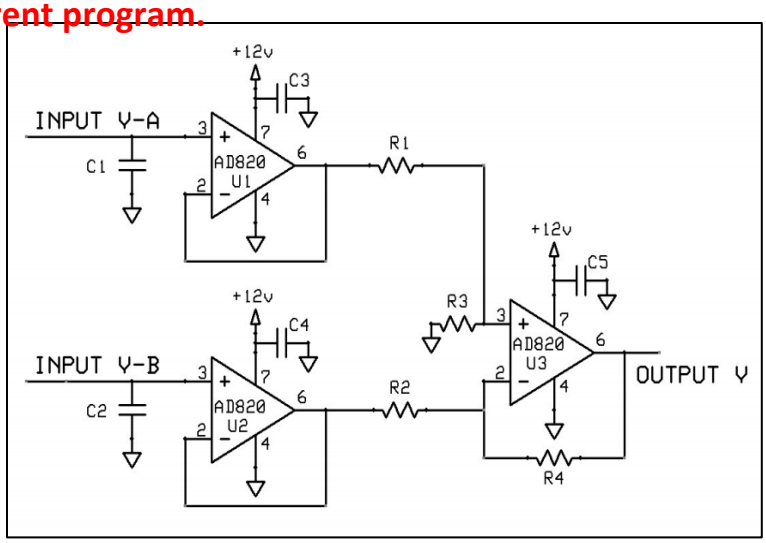


Figure 5 (above): The circuit used for the pressure reading. The two amplifiers on the left side are voltage followers, designed so that the output at 6 is the same as the input entering at 3. The purpose of these is to prevent any feedback from affecting the pressure sensor, as small changes in voltage can have a significant impact on the resistors seen in **Figure 4**. The amplifier on the right is designed to yield an output directly proportional to the difference between the voltages A and B.

Equation 2

$$Vout = \left(1 + \frac{R4}{R2}\right)\left(\frac{R3}{R1 + R3}\right)(VA) - \left(VB\frac{R4}{R2}\right)$$

If we set R1 = R2 and R3 = R4, then this equation simplifies to

Equation 3

$$Vout = \frac{R4}{R2}(VA - VB)$$

This means that the output will be directly related to the difference of VA and VB.

Temperature Signal Conditioning Process:

1. Approximately 24 hours ahead of step 2, the diode was soldered to two wires, covered in liquid electrical tape, and left to set. Because the diode is polar.
2. Using **Figure 2** and **Figure 3**, the temperature circuit was assembled accordingly.
3. Using a multimeter to measure resistance, R2 and R3 were set to the appropriate values by adjusting the potentiometers for each.

4. In temporary place of the diode in **Figure 3**, a 100kΩ potentiometer was used. This allowed the resistance to be changed to test various input voltages.
5. A multimeter was set up to measure the input voltage at the same time that another was used to measure output voltage.
6. The potentiometer in place of the diode was adjusted such that the input multimeter read 0.793V (as close to 0.8V as was feasible), and the output multimeter read 4.98V.
7. The potentiometer was then adjusted for the input voltage to read 0.464V, and the output voltage read 0.401V
8. The latter number was not within the ±0.03V range that was desired for goal 1.a., so the resistance of R3 was slightly decreased by adjusting the potentiometer there.
9. New multimeter measurements yielded an output of 5.01V when the input was at 0.801V, and 0.0045V when the input was at 0.454V. These outputs were well within the range of 5±0.03V and 0±0.03V, so the criteria for goal 1.a. was met and the signal conditioning for the temperature circuit was successful.

Pressure Signal Conditioning Process:

1. Using **Figure 4** and **Figure 5**, the pressure circuit was assembled accordingly.
2. Using the multimeter, the potential difference between VA and VB was measured to be 190mV at ambient pressure, which was 30.31inHg (767mmHg).
3. In order to be able to find easily obtainable resistance values, this value was approximated to be 200mV. This value is represented in **Equation 3** as VA-VB.
4. Assuming a minimum value of VA-VB as 0V (the data sheet gives the zero-pressure output difference to be 0±0.002V), we can use a directly proportional relationship to determine the resistance values.
5. Using **Equation 3**, the ratio of R4/R2 was found to be 25 in order to make the maximum value of (VA-VB) 5V.
6. R1 and R2 were set to 3kΩ and R3 and R4 set to 75kΩ, as these were the smallest resistance sizes found that have a perfect 25:1 ratio.
7. After using the resistance values, the output at ambient pressure was found to be 4.78V. This is likely due to the approximation of 190mV ≈ 200mV, and because of assuming the base was exactly 0V when the pressure was 0.
8. This ratio did not meet the signal conditioning criteria discussed in goal 1.a, but the circuit almost used the entire span of 5V. Thus, this circuit could still be used in the calibration to find a relationship between V out and pressure, and was not modified after this test.

Software Design:

Data Format and Storage:

The data obtained by the telemetry circuits and Adafruit GPS logger shield was formatted in a .csv file on a micro-SD card that was plugged into the shield. Each time a new row of data

was logged, the GPS timestamp was recorded first, followed by the latitude and longitude, the GPS fix status, the number of satellites, and finally the pressure and temperature readings that were obtained from the circuits. If a data value was not available for whatever reason while data was being recorded to the SD card, then an easily-discernable value of -999 was used in place of the data to indicate this.

In regards to the data acquisition rate, a new NMEA sentence of each type requested (RMC and GGA in this case) was obtained by the GPS logger shield every second. Data would be added to the .csv file every time a new set of NMEA sentences was picked up.

The lowest number of characters in each row of data on the .csv file would occur if each data entry only used a single digit, while the highest number of characters would occur if each data column used the most possible digits (i.e. if the longitude reading were -108, it would be using 4 characters, which is the highest possible number of characters for a longitude reading). It is also important to consider the commas separating each value in the .csv file. Based on the data format seen on page 10, the minimum number of characters (and thus the minimum numbers of bytes) recorded during a span of one NMEA sentence would be 28, while the highest number of characters would be 56. This highest number of characters would occur if temperature and pressure readings were both in the form of xxx.xx (using 6 bytes), if there was no latitude or longitude readings (thus they were -999), and if every component of the timestamp contained 2 digits.

Because of this, the rate of data acquisition is between 28 and 56 bytes per set of NMEA sentences, or 28-56 bytes/second. To be conservative, only the maximum value was used for calculations.

Considering that the titles of the columns require a total of 142 characters/bytes, the maximum number of bytes in a file containing 150 rows of data would be given by **Equation 4**.

Equation 4

$$MaxBytes = 142 + 150(56) = 8542\ bytes$$

Assuming that a typical balloon flight is roughly 22 hours, there would be 79200 seconds worth of data from a flight. At 56 bytes/sec, the required storage space for collecting this data on a typically flight would be a maximum of

$$(79200\ sec) * (56\ bytes/sec)\ =\ 4.4352\ mB$$

This is well below the total number of bytes that the micro-SD card can hold, so there is no concern for adjusting data storage.

Operating Software:

Flowcharts of the software's functions are shown below, beginning with the Setup function and followed by the Loop and GPS interrupt functions.

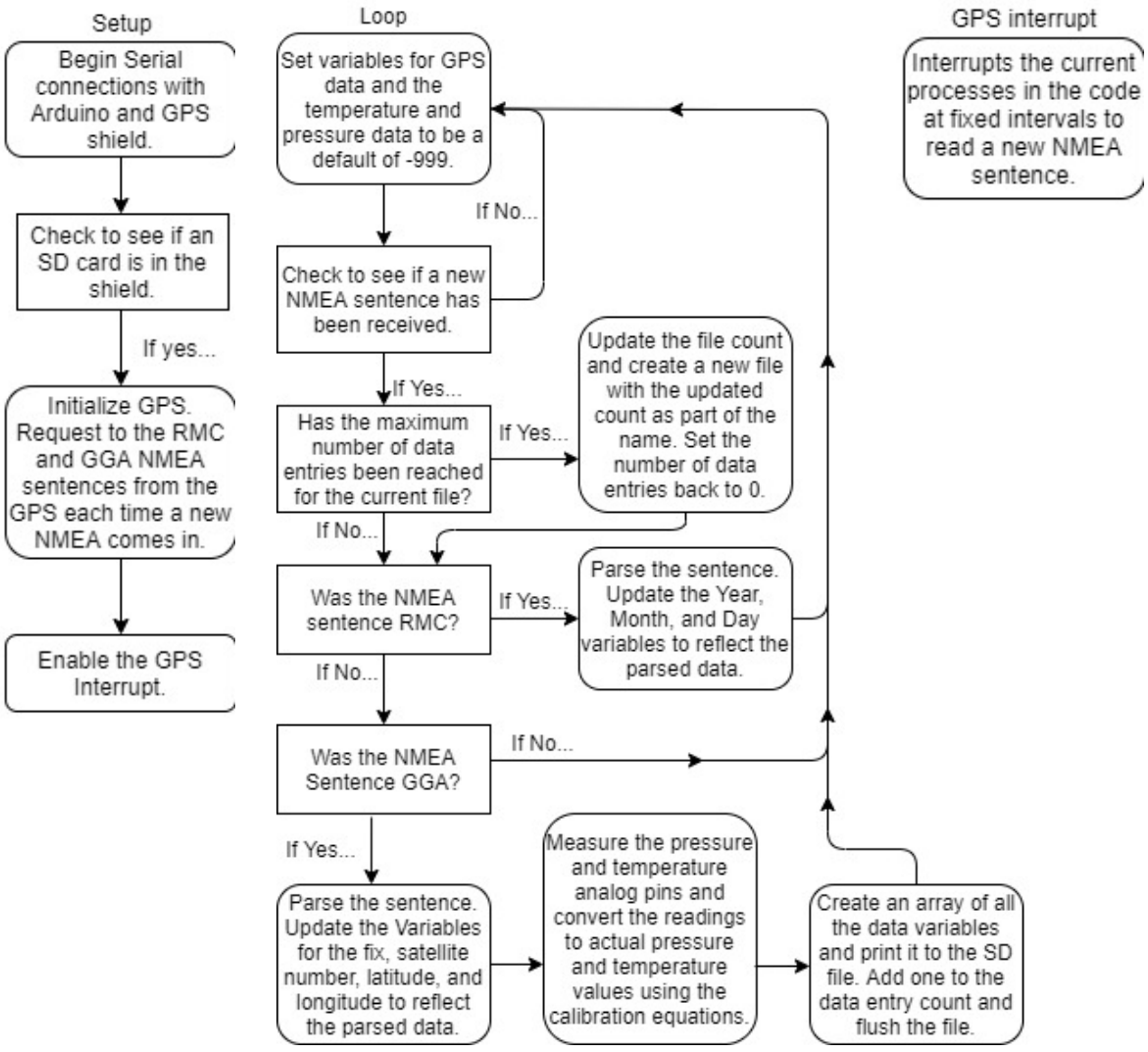
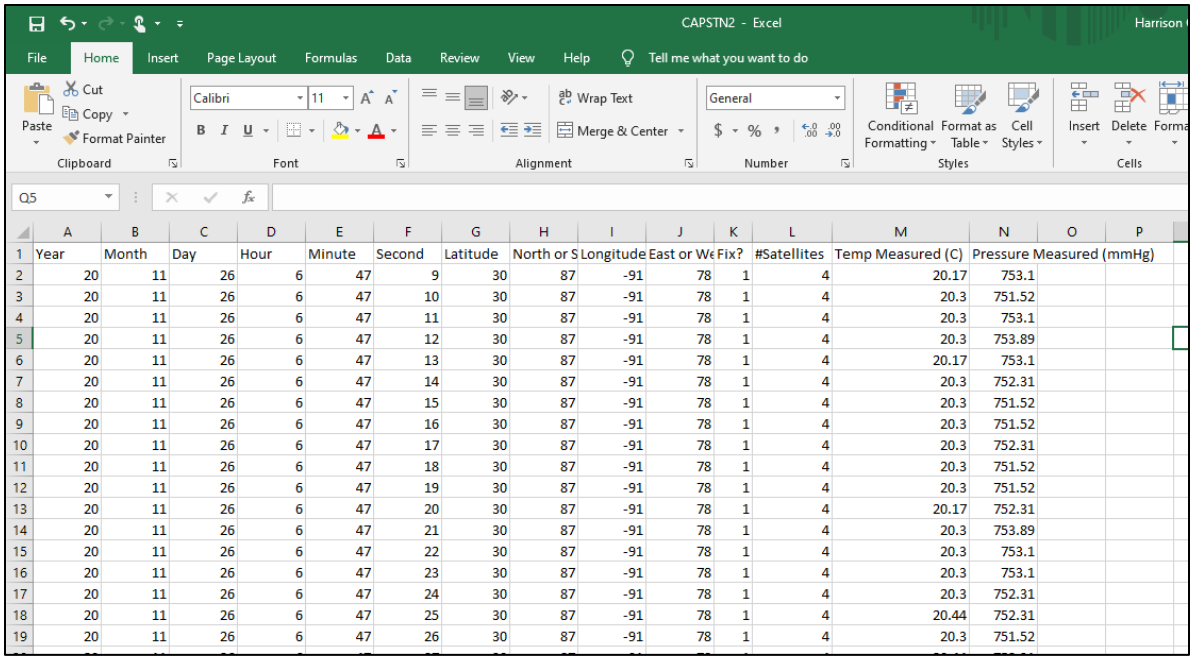


Figure 6 (above): These flow charts display the setup, loop, and GPS interrupt functions for the software used. The software was used to parse GPS data, take pressure and temperature readings from the Arduino Mega, and upload these readings with timestamps and other GPS data onto a micro-SD card.

Data Retrieval Software:

The data retrieval process was simple, as the information was stored in .csv files, which are readable by Microsoft Excel. Once data was uploaded onto the micro-SD, the card was taken out of the logger shield and plugged into a computer with Excel installed. From here, the data could be read and displayed as in **Figure 7** below.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Year	Month	Day	Hour	Minute	Second	Latitude	North or S	Longitude	East or We	Fix?	#Satellites	Temp Measured (C)	Pressure Measured (mmHg)		
2	20	11	26	6	47	9	30	87	-91	78	1	4	20.17	753.1		
3	20	11	26	6	47	10	30	87	-91	78	1	4	20.3	751.52		
4	20	11	26	6	47	11	30	87	-91	78	1	4	20.3	753.1		
5	20	11	26	6	47	12	30	87	-91	78	1	4	20.3	753.89		
6	20	11	26	6	47	13	30	87	-91	78	1	4	20.17	753.1		
7	20	11	26	6	47	14	30	87	-91	78	1	4	20.3	752.31		
8	20	11	26	6	47	15	30	87	-91	78	1	4	20.3	751.52		
9	20	11	26	6	47	16	30	87	-91	78	1	4	20.3	751.52		
10	20	11	26	6	47	17	30	87	-91	78	1	4	20.3	752.31		
11	20	11	26	6	47	18	30	87	-91	78	1	4	20.3	751.52		
12	20	11	26	6	47	19	30	87	-91	78	1	4	20.3	751.52		
13	20	11	26	6	47	20	30	87	-91	78	1	4	20.17	752.31		
14	20	11	26	6	47	21	30	87	-91	78	1	4	20.3	753.89		
15	20	11	26	6	47	22	30	87	-91	78	1	4	20.3	753.1		
16	20	11	26	6	47	23	30	87	-91	78	1	4	20.3	753.1		
17	20	11	26	6	47	24	30	87	-91	78	1	4	20.3	752.31		
18	20	11	26	6	47	25	30	87	-91	78	1	4	20.44	752.31		
19	20	11	26	6	47	26	30	87	-91	78	1	4	20.3	751.52		

Figure 7 (above): This figure shows one of the .csv files that was uploaded onto the micro-SD from the GPS logger shield. Note that the variables for the North/South aspect of the latitude and East/West aspect of the longitude are in number form, giving the numerical ASCII representation of the letters N, S, E, or W.

As seen in **Figure 7**, the year is represented by the last two digits of the current year. The timestamp gives the time in UTC (coordinated universal time), while the latitude and longitude are given in degrees. The direction of the latitude and longitude are given by the North/South and East/West variables. The “Fix?” column, displays a 1 if there is a GPS fix, and a 0 if there is none. The “# satellites” column indicates the total number of satellites that the GPS shield had contact with at the time of the NMEA sentence, and the temperature and pressure columns indicate the pressure and temperature that was measured in the Arduino’s analog pins from the two circuits.

Proof of Performance:

The calibration of each circuit was needed in order to provide an understanding of how the ADC value and the actual temperature or pressure were related. As such, a thermometer and pressure gage were used in order to compare the ADC values to the varying temperatures/pressures experienced by the sensors. The materials used for the calibrations are listed below, followed by a description of the calibration process for each circuit.

Materials:

- Arduino Mega with USB cable for Serial connection
- HP laptop model 15-da0xxx with Arduino IDE installed
- 12V power supply
- 1N457 Diode
- Model 1210 PC board mountable pressure sensor
- 500mL beaker
- Water
- Ice (in small chunks)
- Thermolyne Cimarec 3 Hot Plate
- Heat resistant gloves
- Pasco scientific digital meter (model SF-9616) (set to "thermometer" and set to the "<200 °C" mode)
- US General manual pressure pump
- Lab notebook and pen
- Assembled Pressure and Temperature Circuits

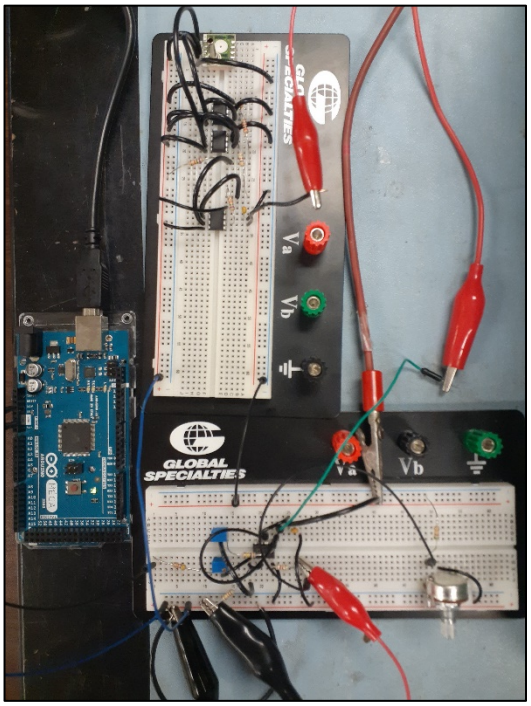


Figure 8 (above): A depiction of the telemetry circuits used, both connecting to the Arduino Mega on the left. The pressure circuit is constructed on the top breadboard, while the temperature circuit is on the bottom breadboard. In this image, the temperature circuit is using a potentiometer (the cylindrical shape on the bottom right) in place of the diode, as this image was taken while testing the signal conditioning.

Temperature Calibration Process:

1. The 100kΩ potentiometer in the temperature circuit was replaced by the diode, now covered in liquid electrical tape so that it could be submerged in water without shorting.
2. In Arduino IDE, a code was written to read the analog pin that the output of the temperature circuit was plugged into.
3. The beaker was filled with 300mL of water, and it was heated on the hotplate.
4. The 12V+ power supply was turned on to power the temperature circuit.
5. Once the water in the beaker was boiling, the beaker was removed and placed such that both the diode and the thermometer could reach into it at the same time (see **Figure 9**).
6. The thermometer was held in the center of the beaker, not touching the edges or bottom of the beaker.
7. After the temperature reading was allowed to stabilize, the diode was submerged next to the thermometer and held there. The ADC readings from the diode were read by the Arduino and displayed on the serial monitor.
8. Five ADC values were recorded at various temperatures, starting at 96.4°C and going through 14 temperatures roughly spaced by about 7 degrees each. The lowest temperature at which ADC values were recorded was 1.3°C.

9. To cool off the water in the beaker between temperatures, ice was added and mixed in until dissolved using the thermometer.
10. ADC values were only used if the temperature reading on the thermometer was constant for all 5 ADC measurements, in order eliminate any error from changing temperature.

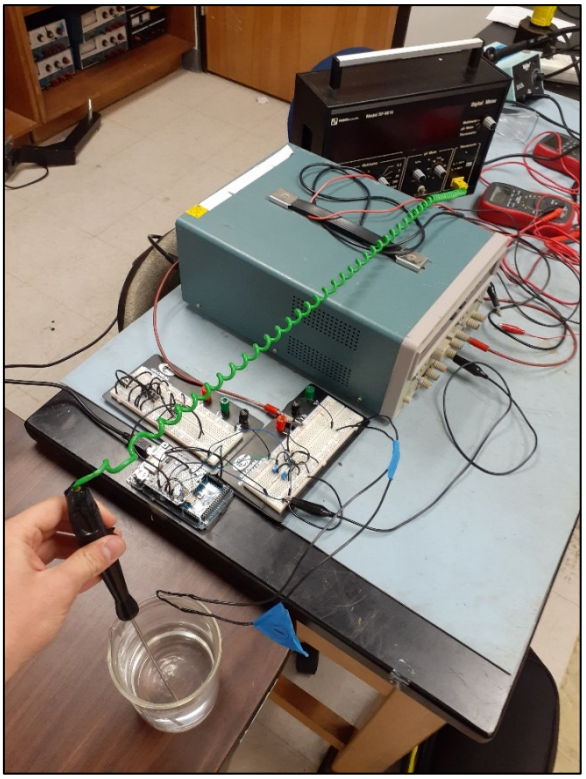
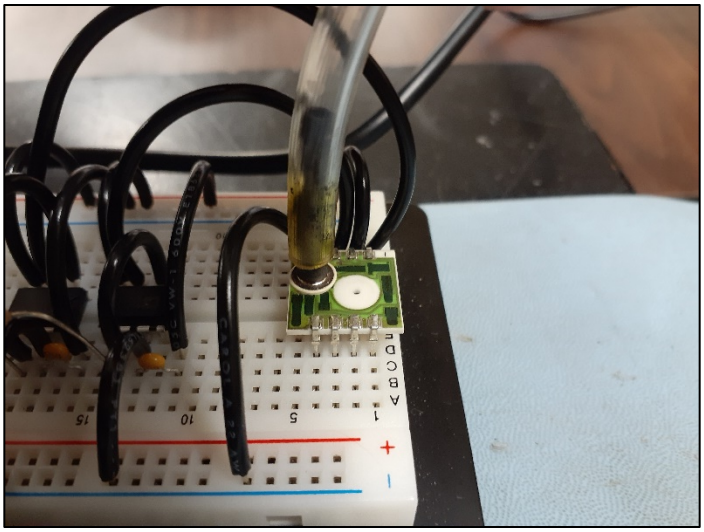


Figure 9 (right): The setup used while calibrating the temperature circuit. The black box at the top of the frame is the thermometer, not turned on in this image. Below that is the power supply, followed by the two circuits and the Arduino Mega. The diode can be seen resting on the edge of the beaker, and was inserted next to the thermometer probe when measurements were being taken.

Pressure Calibration Process:

1. A code was written in Arduino IDE to read the analog pin that the output of the pressure circuit was plugged into.
2. The US general manual pressure pump was attached to the 1210 PC.
3. At ambient pressure (767mmHg), five ADC values were recorded. This ambient pressure was obtained from the barometer index on LocalConditions.com (see citations).
4. The pump was used to reduce the pressure by 100mmHg, and 5 more ADC measurements were taken at this new pressure.
5. A slight increase in the ADC value was measured over time at pressures below ambient pressure. This indicates some leakage, but was not large enough to cause any detectable shift in the reading on the pump.
6. Step 5 was repeated 5 more times, with ADC values being measured every 100mmHg from 767mmHg to 167mmHg.
7. The final -100mmHg after 167mmHg was too low of a pressure to be achieved by the manual pump, so the pressure was reduced by only 80mmHg. Five more ADC values were recorded at p = 87mmHg.

Figure 10 (right): A view of part of the pressure circuit built on the solderless breadboard. The 1210 PC pressure sensor in the middle of the image can be seen attached to the tube that is part of the US General pressure pump.



Temperature Calibration Results:

Once data was collected for the temperature calibration, a mean ADC value (m) was found for each temperature, and the uncertainty in the mean σ_m was calculated using **Equation 5** below.

Equation 5

$$\sigma_m = \frac{\sigma}{\sqrt{N}}$$

Where σ is the standard deviation given by **Equation 6** and N is the total number of ADC measurements taken for one temperature.

Equation 6

$$\sigma = \sqrt{\frac{\sum(ADC - m)^2}{N - 1}}$$

Where "ADC" is one individual ADC measurement taken at the temperature.

A graph of the data was then plotted (**Figure 11** below), and the slope was used to determine the relationship between the ADC value and temperature.

As seen in the top of the **Figure 11**, the equation representing the temperature with respect to ADC value is:

Equation 7

$$T = -0.1366(ADC) + 47.898$$

The uncertainty of any one temperature obtained by using this formula can be expressed using partial derivatives, as seen in **Equation 8** below.

Equation 8

$$\sigma_T = \sqrt{\sum\left[\left(\frac{\partial T}{\partial x_i}\right)^2 (\sigma_{xi})^2\right]}$$

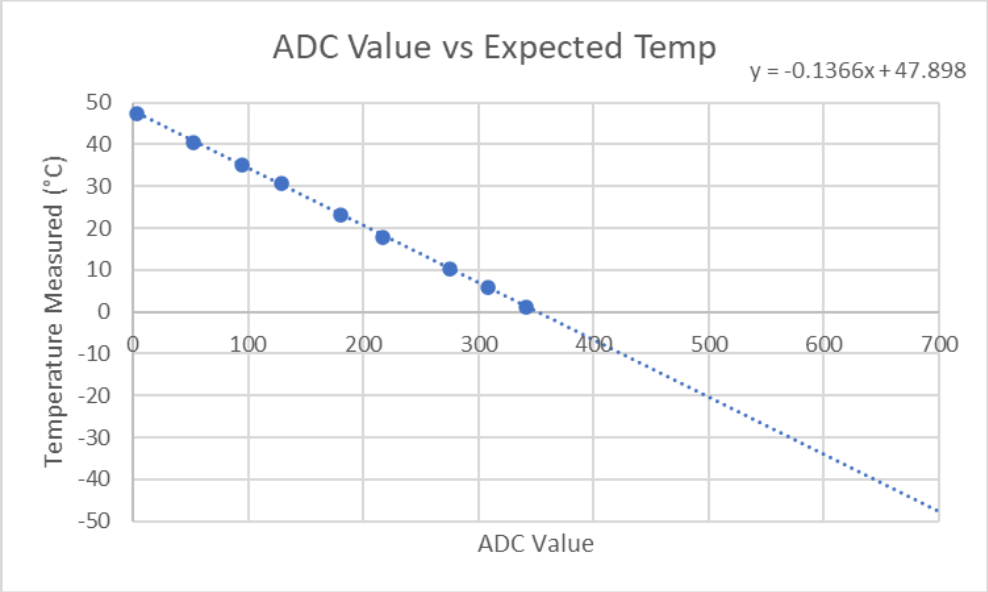


Figure 11 (above): Average ADC value picked up by the temperature circuit vs temperature measured by the thermometer. Note that error bars were included, but were too small to see. The error in the temperature reading was ±0.5°C, taken from the thermometer data sheet. The average σ_m for the ADC values at each temperature was ±0.4. Since an ADC value can only be an integer, this uncertainty must round up to 1, since we cannot know the ADC value with 100% certainty.

Because excel cannot give the uncertainty in the line of best fit, we assume that it is constant, making the slope and intercept constant as well. As such, the uncertainty expressed by **Equation 8** can be simplified to

$$\sigma_T = \sqrt{(-0.1366)^2(\sigma_{ADC})^2}$$

Where the minimum value of the uncertainty in the ADC (σ_{ADC}) is ±1. This simplifies σ_T to an uncertainty of ±0.1366°C, which is far less than the uncertainty in the thermometer reading (±0.5°C) from the data sheet. Because of this, to keep estimates of uncertainty conservative the uncertainty in a temperature obtained by using **Equation 7** was the data sheet value of ±0.5°C. This uncertainty reaches the requirement set by goal 2.a.i.

It should be noted that the x-axis range displayed in **Figure 11** does not extend to the full 1023 potential ADC values. The line of best fit extrapolates out to around -90°C if the ADC value was maxed. However, in the case of ballooning payloads, the typical temperature range would likely be smaller than -90°C to 50°C. As such, the relationship between ADC value and temperature could be further optimized by decreasing the slope of the graph. This could be done by excluding the consideration of higher input voltages in the signal conditioning graph (**Figure 1**), which would allow for the full ADC span to cover only the higher temperatures, or by excluding the lower input voltage values, which would allow the ADC span to cover only the lower temperatures. Doing the latter would be more difficult to calibrate, since there is limited access to calibration equipment that can measure below 0°C.

Pressure Calibration Results:

Once data was collected for the pressure calibration, a mean ADC value was found for each pressure, and the uncertainty in the mean σ_m was calculated using **Equation 5**.

Note that the ADC reading was slowly increasing with each measurement as a result of leaking, but the pressure reading on the analog pump did not change noticeably while this happened. As such, the systematic error from leakage was significantly smaller than the error from the precision of the manual pump, and thus was not taken into consideration during calculations. However, the pump measured in intervals of 20mmHg, so the uncertainty in each measurement of pressure was ±10mmHg (1/2 of the smallest marked interval on the pump). This uncertainty from the pump was the value of uncertainty assigned to each pressure measurement.

As with temperature, a graph of the data was plotted (**Figure 12**), and the slope of the line of best fit was used to determine the relationship between the ADC value and pressure.

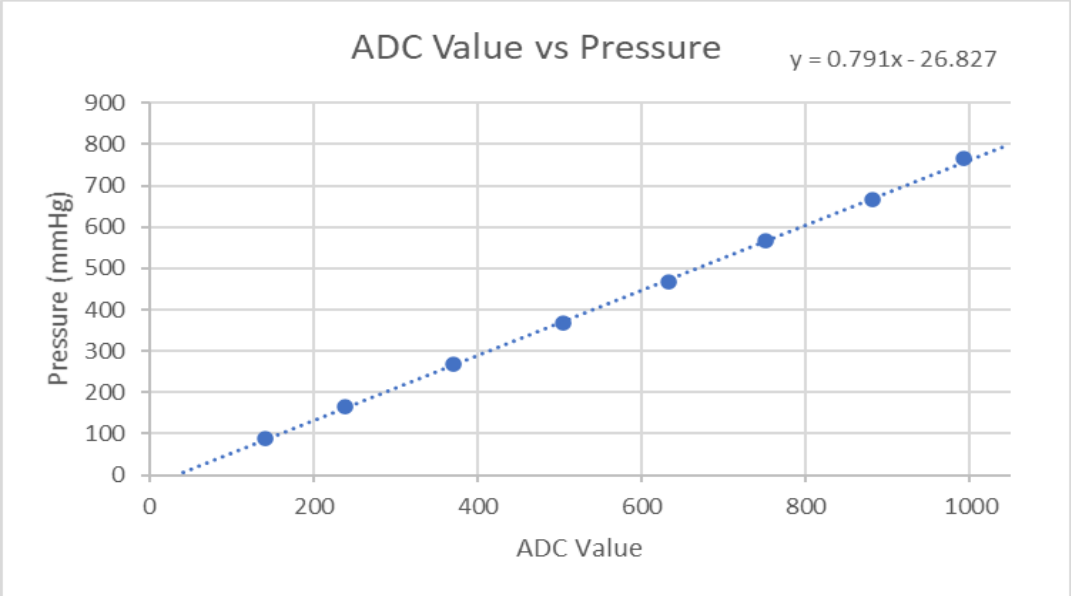


Figure 12 (above): Average ADC value measured vs pressure measured by the hand pump. Note that error bars were included, but were too small to see. The error in the pressure reading was ±10mmHg, as stated above. The average σ_m for the ADC values at each pressure was ±1.7.

As seen at the top of **Figure 12**, the equation representing the temperature with respect to ADC value is:

Equation 6

$$P = 0.791(ADC) + 26.827$$

In the same way as the analysis done on **Equation 8**, the uncertainty in a pressure measurement obtained using **Equation 6** can be expressed and simplified in terms of partial derivatives. This yields an uncertainty of ±1.34mmHg (by using σ_{ADC}= ±1.7).

This uncertainty was significantly smaller than the uncertainty found from the intervals on the manual pump, and since the total uncertainty cannot be smaller than the error of the device taking measurements, the pump's error of ±10mmHg was used in place of this calculated error. This uncertainty value meets the criteria of goal 2.a.ii.

Conclusion:

Ultimately, the Sensor Interface and Calibration report was successful, as all but one of the outlined goals were met. The one goal that was not met (1.a, signal conditioning of the pressure circuit) did not have a large impact on any other aspect of the calibration or software development, so it was not a major setback. The code functioned as intended, and the necessary data was uploaded and formatted correctly onto the micro-SD card.

One possible improvement of the software could come from using alphabet characters rather than ASCII numbers to indicate the direction of the latitude and longitude on the .csv files, as this would be easier to comprehend. Also, to save memory, the code could be modified to check for RMC sentences less frequently, as the RMC NMEA sentences were only used for updating the day, month and year, which did not change very frequently.

Citations:

Ambient Pressure: https://www.localconditions.com/weather-baton-rouge-louisiana/70801/#:~:text=Baton%20Rouge%2C%20LA%20Weather&text=The%20barometric%20pressure%20is%2030.19,hour%2C%20gusting%20at%202%20mph.

1210 PC Pressure Sensor Data Sheet: https://laspace.lsu.edu/laaces/wp-content/uploads/2020/08/R16.03_ICS1210_Datasheet.pdf

LaACES Sensors and Signal Conditioning Document: https://laspace.lsu.edu/laaces/wp-content/uploads/2020/11/A16.01_Sensors_and_Signal_Conditionings.pdf

Operational Amplifier Data Sheet: https://laspace.lsu.edu/laaces/wp-content/uploads/2020/08/R16.04_AD820_Op_Amp_Datasheet.pdf

1N457 Diode Data Sheet: https://www.mouser.com/datasheet/2/149/1N457-888325.pdf

Diode-based Temperature Measurement: https://www.ti.com/lit/an/sboa277a/sboa277a.pdf?ts=1607973667840&ref_url=https%253A%252F%252Fwww.google.com%252F