A High-Altitude Digital Sonic Anemometer: Development and Deployment

A thesis submitted by

Tim J. Cheng

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

Tufts University
Medford, Massachusetts

May 2023

Advisor: Robert White

Abstract

Stratospheric sonic anemometers are an emerging technology for making *in situ* wind velocity measurements in low pressure environments. This thesis explores the design, optimization, and high-altitude testing of a digital sonic anemometer with possible uses in balloon navigation, atmospheric gravitational wave research, solar phenomena research, and Martian atmosphere characterization. The device was flown on NASA's SPARROW-3, a high-altitude balloon, out of Fort Sumner, NM, USA. The mission was successful, resulting in relative wind data at temperatures as low as -40 ºC and all the way to the balloon's peak altitude of 38 km (3.8 mbar). The device surpassed all current sonic anemometers regarding maximum operational altitude. Optimization of firmware and software resulted in a velocity resolution of 0.1 m/s (standard deviation) and a three-dimensional wind measurement every 2.25 s. Further plans are to increase performance to a resolution of 0.01 m/s and a wind sampling rate of 10 Hz.

Acknowledgements

I would like to thank: my advisor Professor Robert D. White for being a phenomenal guide in writing my master's thesis, my committee members Lecturer Felix Huang and Dr. Don Banfield for taking the time to further my education, Tufts University for providing a conducive learning environment, Dr. Chris Yoder and the NASA Balloon Programs Office for allowing us to fly aboard their balloon, my supporters from NASA for funding (partially funded under NASA-NNX16AJ24G and NASA-80NSSC20M0007), and my wife for supporting me through it all.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This thesis is an overview of Tufts' ultrasonic anemometer and its most recent revisions and tests. It includes prior art and alternative technologies within the field of low-pressure stratospheric anemometers, a detailed description of Tufts' anemometer, methods to optimize a digital sonic anemometer, descriptions of ground and airborne tests, and a brief discussion of future directions for the device.

The goals of this project included: replacing an existing analog system with a digital one to increase performance, constructing and optimizing the data processing method, and evaluating hardware, both in a bell-jar and aboard a high-altitude balloon.

1.1 Prior Art

Sonic anemometers, having the ability to measure low and high-speed winds at both low and high pressures, have been developed in stratospheric anemometry related research. Some recent achievements include their use in the SENSOR and TILDAE missions as well as their development for the Martian atmosphere.

Sonic anemometry was first conceptualized in the early 1950s by Barrett and Suomi (1949) [1] and Corby (1950) [2]. Shortly after, an exploration into signal processing, error analysis, and system design suggestions were made by Von dem Borne (1954) [3] on the subject. An analog one-dimensional system was presented in 1957 by Suomi [4] that operated with a continuous sound wave and phase shift recognition. Later revisions by Kaimal and Businger (1963) [5] included temperature as a deducible quantity from their device. However, a major step occurred in the 1980s with the

utilization of a digital system by Hanafusa (1982) [6], that could eliminate certain temperature dependencies that were challenging to avoid with past analog systems. Since then, the advances in microchip technology have led to higher wind range coverage and sample rate with a reduction in power consumption and cost for digital systems.

Table 1, Comparison of early digital anemometer to currently available product.

	Hanafusa System (1982)	Gill Systems (Consumer Product, 2023)
Measurement Range (m/s)	0 – 30	Up to 65
Resolution (cm/s)	0.5	1
Sample Frequency (Hz)	20	Up to 100
Power Usage	AC 100/115/220V±10％ 50/60Hz	~3.6 W

There are a few active groups working on high altitude sonic anemometers. The main being a group from the National Space Science Center (NSSC), Chinese Academy of Sciences. Their system's design and goals are similar to the Tufts system. This group describes a successful balloon launch to 25 km (~30 mbar) in 2019 [7]. The Stratospheric Environmental Responses to Solar Storms (SENSOR) campaign are developing an in house (Chinese Academy of Sciences) high altitude sonic anemometer for solar and climate research. They recently (2022) published an overview of a successful flight with wind measurements at and near float (25 km) [7]. The system utilizes 40 kHz transducers and employs an FPGA board to achieve 10 wind measurements per second. Another notable group that is associated with the University of Delaware launched a sonic anemometer from Antarctica in 2016 [8]. The Turbulence and Intermittency Long-Duration Atmospheric Experiment (TILDAE) flew a commercial sonic anemometer with

minor modifications from Applied Technologies, Inc. [8]. The system used 100 kHz transduces with 200 wind measurements per second and operated successfully up to an altitude of 18 km.

Our group is also working to develop a Martian sonic anemometer in conjunction with NASA and VN Instruments. Because of the similarity to Earth's stratosphere in pressure and temperature, the Martian sonic anemometer can also operate at high altitudes on Earth [9]. In prior ground testing work, earlier versions of the system operated down to 6 mbar [10] and -40^{o}C [11] in CO_2. Our collaborator, Don Banfield, successfully flew this 1 axis device to 34.5 km with wind resolutions of approximately 0.5 m/s in 2012 [11]. We argue that sonic anemometry can outperform current wind sensors, such as hot wire, by factors as large as 10 in resolution and sampling rate on Mars making it ideal to characterize turbulent eddy structures on Mars [12]. In this thesis a modified design of the Martian sonic anemometer is utilized to develop a standalone stratospheric balloon mounted anemometer.

Stratospheric anemometers have promising uses in balloon navigation [13]–[16], gravity wave detection [17]–[19], solar activity monitoring [7], [20], and planetary science [10], [11]. Balloon navigation is reliant on accurate wind field measurements due to it being the main driving force of balloon trajectory. If the wind fields are known, commonly in the form of velocity relative to the balloon, a desired balloon trajectory can be achieved. This could involve hanging a sail much lower than the balloon in a different wind field or simply changing the ballast of the balloon to raise or lower itself into a desired wind direction. Altitude control was utilized by *Loon LLC* [21] to control their internet providing high altitude balloons. Platform stabilization, for telescopes or other

sensitive instruments, is largely influenced by relative winds and could be improved through *in situ* anemometry as well [22]–[24]. Abrupt horizontal wind changes of 2 m/s can be expected at float [25], [26] leading to platform instability. Traditionally these disturbances are counteracted with some form of closed loop control, however, recently the incorporation of more complete physical balloon models are also being developed [23], [24], [27]. Adding higher quality wind measurements into either of these controllers could improve platform stability.

Gravity waves in the stratosphere, from earthquakes, explosions, or large storms, are of interest for event detection and atmospheric science [19]. Currently, experimental verifications for these event has been by satellite or radiosonde data [18], [28]. Stratospheric sonic anemometry could help verify and improve models or be used to collect real time data of gravitational wave events. Solar activity, which has a relationship to climate change [20], can also be monitored in stratospheric winds. Solar flares and proton events are notable solar activities with enough energy to disturb the Earth's stratosphere in a violent way [7]. Due to the similarity of Earth's stratosphere to that of the Martian surface, sonic anemometers have utility in measuring atmospheric characteristics such as eddies on Mars [12]. A major portion of the technology can also be used to measure gas composition instead of wind speed, with possible applications in characterizing a variety of planetary atmospheres [29]–[31]. Sonic anemometers aim to improve resolution and data frequency in low pressure environments where traditional anemometers are less capable.

1.2 Alternatives

Anemometry in the stratosphere is exceptionally challenging due to the low pressure (down to 1 mbar) environment. Low air density provides relatively low energy/inertia in atmospheric winds. Traditionally, *hot wire* anemometry has been used at these altitudes with varying success [32], [33]. There are also proposals to use *cup* anemometers near such pressure levels with successful ground testing [34], [35]. In recent years, laser doppler anemometry has been making substantial advancements with experiments on airplanes [36].

Hot wire is by far the most common high altitude wind measurement device currently used [33]. They function by measuring the resistance of thin wire while current passes through it. The name 'hot wire' comes from the increased temperature of the wire, either by constant current through the wire or more commonly a specified constant temperature for the wire. As air velocity increases convective cooling increases leading to measurable changes in resistance or current through the wire. Using Reynold's analogy, a flow rate can be estimated from this heat loss. Some of the challenges with hot wire anemometers are their calibration and favorability toward slow flow [8], [32]. Hot wire anemometry relies heavily on gathering environmental data, such as air temperature, density, and composition before accurate wind measurements can be made. These air conditions as well as radiative effects at higher altitudes are vital variables in complex nonlinear hot wire anemometer models that need to be calibrated before flight. Sonic anemometry has an advantage in this respect because it can measure air temperature and is not as affected by air density. Some of the advantages of hot wire anemometers are their long history of use, size (~1 cm wire lengths [8]), weight, and power consumption.

There are recent developments in hot wire anemometry for high altitude conditions, such as the use of optical fiber sensing with Fiber Braggs Grating (FBG) to more accurately measure heat flux [37].

Cup anemometers are a simple technology often used at sea level, but have their own challenge at low pressures [35]. Cup anemometers are a mechanical approach to wind measurement. They operate by measuring the rotation of cups around a solid axle for wind speed and a rudder to measure wind direction. These devices tend to suffer at high altitude from not enough kinetic energy from low density air to spin effectively. However, there are successful projects utilizing cup anemometers as high as 18 km aboard balloons (with simulated testing up to 25km) and ongoing research in the area [34], [35]. They do have an advantage over hot wire in that they do not need temperature measurements, just pressure. When compared to sonic anemometers, cup anemometers tend to be slightly lighter with less power requirements (Table 2) but are disadvantaged in calibration requirements and low flow conditions.

Laser Doppler Anemometry (LDA) also referred to as Doppler lidar, has been making recent advancements in atmospheric wind measurements [36], [38]. This technique works by measuring the Doppler shift in light off moving particles in the air [39]. These laser systems tend to implement high end hardware consisting of laser interferometry equipment, custom FPGA boards, and high frequency analog to digital converters (ADC). Currently, commercial LDA systems [40] are used to measure wind fields around wind turbines near sea level and are relatively heavy (~45 kg). The highest experimental demonstration of this technology was at 12 km aboard a plane with a 5 W laser [41]. In comparison to sonic anemometry, LDA currently needs more power and is

relatively untested at high altitudes. However, like sonic anemometry, LDA does not need environmental data or extensive calibration to determine wind velocity [39].

Table 2, Comparison of alternative anemometer technologies focused on high altitude relative wind measurements.

Method	Accuracy (m/s)	Range of Operation (m/s)	Altitude Reached (km)	Update Rate (Hz)	Power ** (W)	Mass** (g)	References
Sonic	0.01 - 0.1	0 - 15	25 – 38[+]	1 - 200	~1	~200	[7], [8], [10]
Cup	0.1	1.5 - 22*	17 - 25*	1	<1	~100	[34], [35], [42]
Hot Wire	Varying	Highly Variable	40	1 - 2k	<1	<100	[8], [32], [33], [43]
Laser Doppler (Lidar)	0.1	0 - 150	12	1 - 16	>3 - 45	1k - 45k	[36], [39]–[41], [44]–[46]

[+]Achieved in this thesis, 25 km in referenced studies.
*In a lab setting opposed to on a balloon.
**Records are sparse in literature, some of these are estimates based on similar hardware.

1.3 Contributions

This thesis presents the creation and testing of a high-altitude digital sonic anemometer that successfully operated at an altitude of 38 km. The system presented was the first sonic anemometer to operate this high. Custom software and hardware were also developed to produce the device.

1.4 Sonic Anemometry

Sonic anemometers function by measuring either the phase shift or more commonly the Time of Flight (ToF, or Time of Arrival, ToA) of sound wave through air [8], [34], [47]. By comparing the difference in propagation time (or phase for analog systems) in opposite directions along a given axis a velocity measurement can be

obtained. A basic single axis transducer configuration can be seen in Figure 1 with subsequent equations detailing the process for velocity extraction from ToF data.

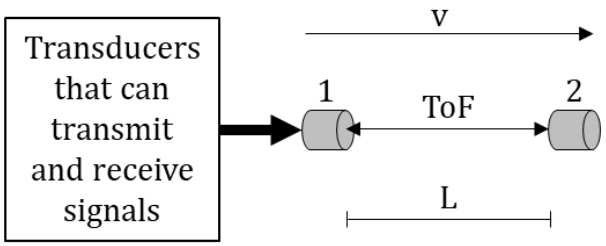


Figure 1, Basic sonic anemometer diagram.

Assuming the wind velocity $\boldsymbol{v}_{12} = -\boldsymbol{v}_{21}$,

$$ToF_{12} = \frac{L}{c + \boldsymbol{v}_{12}} \quad \textbf{(1)}$$

$$ToF_{21} = \frac{L}{c + \boldsymbol{v}_{21}} \quad \textbf{(2)}$$

$$c = \frac{L}{2}\left(\frac{1}{ToF_{12}} + \frac{1}{ToF_{21}}\right) \quad \textbf{(3)}$$

$$\boldsymbol{v}_{12} = \frac{L}{2}\left(\frac{1}{ToF_{12}} - \frac{1}{ToF_{21}}\right) \quad \textbf{(4)}$$

where c is the speed of sound. This is a general way to determine the speed of sound and wind velocity along the transducer axis from Time of Flight. Note that the wind velocity and speed of sound equations do not rely on air conditions such as pressure or temperature for sonic anemometry. This is a distinct advantage over hot wire and cup anemometers that depend on environmental conditions for accurate measurements. A major goal of this thesis is to introduce a system that replaces an existing analog system (similar to Suomi or Kaimal and Businger's anemometer) with a digital one based on ToF. With equations 1-4, a digital time domain device can deliver reliable data without a reliance on temperature as a variable just like Hanafusa's digital anemometer. The 2π ambiguity of phase shift measurements can be eliminated with ToF based calculations making in situ measurements more straightforward.

A notable challenge with high altitude sonic anemometers is a reduction in acoustic source strength and attenuation. For a given volume velocity source, acoustic source strength is proportional to density. For example, if an acoustic source moves from 1000 mbar at sea level to 4 mbar at an altitude of 38 km the resulting signal level drops 48 dB. In addition, attenuation caused by sound absorption is a function of pressure and sound frequency [7]. Lower pressure and higher frequency sound increase sound absorption leading to weaker signals and lower Signal to Noise Ratios (SNR). Similar signal strength losses arise in most anemometry methods (i.e. cup and hot wire) due to reduced density with the exception of technologies that can average single molecule measurements such as LDA [38]. If air density reduces, so does its measurability, especially in relation to physical systems that rely on thermal or kinetic energy. When the SNR of a desired signal becomes too small, interference from structural vibrations and

electrical crosstalk can cause errors in all forms of anemometry. To combat this challenge, sonic anemometers must be carefully designed to minimize noise from both electrical and structural crosstalk, and sufficiently amplify both the transmitted and received signals. Careful attention must also be paid to signal processing methods and their handling of coupled noise from crosstalk. This thesis will explore these issues on the Tufts stratospheric sonic anemometer platform.

2 The System

2.1 Hardware

The complete sonic anemometer, shown in Figure 2 and Figure 3, was comprised of an STM32 L476RG development board, a custom interfacing daughter board for the STM32, a custom amplification/multiplexor board, a Raspberry Pi 3, a custom aluminum octahedral head, *CUI* 40 kHz transducers, a SparkFun Thing Plus SAMD51, two resistive thermometers, a pressure sensor, two IMU units, a GPS unit, a ±15/5v power supply, and all the necessary wires and enclosures for connectivity and protection. The focus of this thesis is on the STM32 development board and its accompanying daughter board as they are new additions from Tufts' previous analog anemometer system. The whole system was powered by incoming 24 V from either a bench top power supply or the gondola's main LiPo battery during flight (~24V).

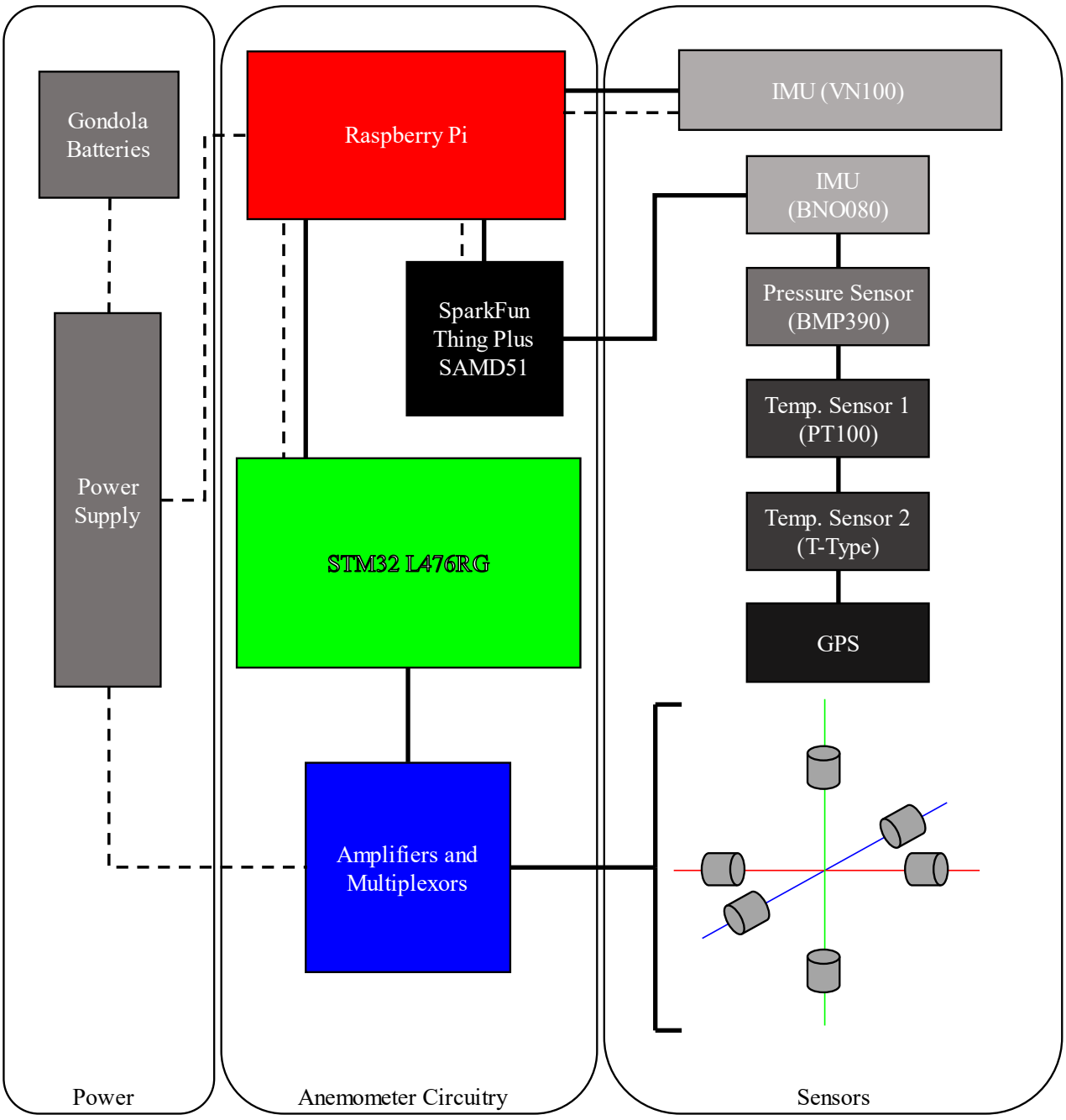


Figure 2, Diagram of entire system. Note that dashed lines represent DC power and solid lines indicate data transfer.

Figure 3, Photo of assembled system (left) and view inside dry box (right).

The STM32 L476RG is a consumer chip designed for microcontrollers with low power goals. It incorporates a 32-bit Arm Cortex-M4 processor running at up to 80 MHz with 128 KB of SRAM. This specific chip was chosen due to having multiple Audio to Digital Converters (ADCs), a Digital to Analog Converter (DAC), and multiple Direct Memory Access (DMA) controllers all on the die. Early testing with lower spec chips (L0 and F4 variants) that lacked enough ADCs and DMA controllers accentuated the need for these features. During operation of the microcontroller, the main processor was clocked at 74 MHz, the ADCs at 2.933 Msps, and the DAC at 4.111 Msps. The DMA controllers were important, as they greatly accelerated the transfer of time domain signal data from RAM to the DAC and from the ADCs to RAM. Some experimentation with DMA controlled data transfer to the Raspberry Pi was also conducted, but this was incomplete before flight and the system ultimately used a CPU based method over USB serial. See the Software section for more detailed explanation of data routing. The development

board drew ~0.5 W during operation, while the entire system consumed ~5.3 W under full load. The L476RG was powered by USB from the Raspberry Pi, and data was also handled by USB between the microprocessor and the Pi.

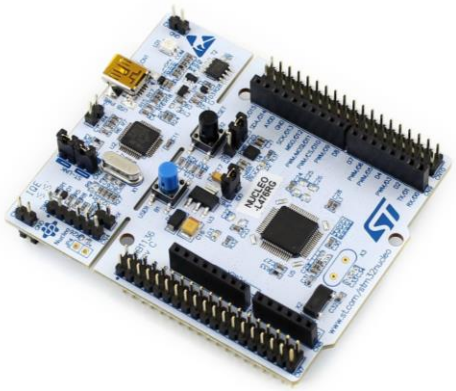


Figure 4, Image of STM32 L476RG development board courtesy of Amazon (https://a.co/d/cRKlf5k)

The interfacing daughter board was specifically made for the STM32 platform to connect with existing hardware. Based on a standard 4-layer PCB, the board was designed in *PCB123* and manufactured by *SunStone Circuits.* It was comprised of two 38 pin banks with the same dimensions as the pins aboard the STM32. These pins were used for ADC signals, signals produced from the DAC, Active Gain Control (AGC) interfacing, transducer channel selecting, and an enabling bit for the amplification boards. The analog signal traces were terminated with male SMA connections, which were the standard on the amplification/multiplexor board. The digital signal traces were terminated with 6 pin vertical headers. Traces carrying analog signal were structured in a way to reduce crosstalk by incorporating perpendicular trace intersections and grounded parallel shielding. This shielding can be seen on the “AC IN” trace of Figure 5. The overall dimensions of the board were based on a previous board design that contained analog

phase measurement chips and n Arduino. The analog board also implemented SMA connectors and 6 pin headers for its operations of external peripherals. Due to a discrepancy between the ADC requiring voltages from 0 to +3.3 V and the amplification board producing -1.5 to +1.5 V signals, a DC blocking capacitor of 100 nF in conjunction with a voltage divider comprised of two 5 kΩ were added. A mistake in the daughter board design was the neglecting of a DC blocking circuit on the DAC output side as well. This was remedied by soldering a 10 nF capacitor in line with the signal and a 90.9 kΩ resistor to ground directly onto the PCB traces after an excision was performed with an *Exacto* blade. A future revision has already been designed with this key oversight remedied and dimensional changes to improve fit in the enclosure. The daughter board was constructed by conventional means (soldering iron).

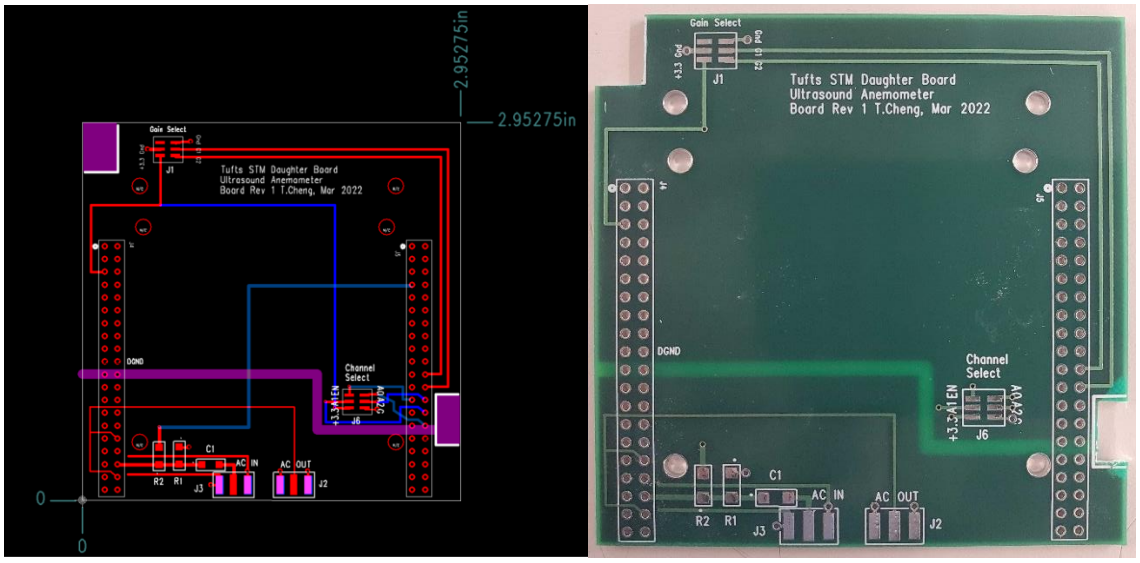


Figure 5, PCB123 daughterboard layout (left) and photo of unpopulated PCB (right).

The amplification and multiplexor boards utilized for this project were designed previously for use with the analog system. The board contained amplifiers to drive the

transducers as well as amplifiers to increase received signals. The other main function of the board was to direct signals with multiplexors so that the same amplifying circuits could be used for all 3-axis pairs in both directions. There was only one modification to the board in order to work efficiently with the new digital system, a gain change on one of the amplifiers that drove the transducers. This was achieved by replacing one of the op-amp's resistors with one of lower value. The change allowed for the full range of the microcontroller's DAC to be utilized thereby increasing the output resolution. Another approach could have been to change the reference voltage of the DAC itself to garner a similar effect. Without this gain change, the transducer amplifier was clipping and attempting to produce signals larger than ±15 V. The boards were manufactured using standard surface mount stenciling methods.

The Raspberry Pi 3 acted as the main control unit, storing information from all the sensors (i.e. temperature, GPS, IMU, etc.) and sending commands to the L476RG. The Pi was powered directly by the 5 V rail on the power supply, and automatically started recording when plugged into the gondola battery before launch. Most of the external sensors were connected to the Raspberry Pi with a USB powered microcontroller (SAMD51) which helped to distribute power and organized the incoming sensor data.

A purpose-built aluminum octahedron was utilized to house the transducers. This structure was repurposed from previous research and was not modified in any way for its use with the digital anemometer. The octahedron points were constructed from rapid prototyped aluminum, they incorporated transducer pockets and holes for aluminum connecting rods along the octahedron edges. The regular octahedron geometry allows for equal distance between three perpendicular transducer pairs allowing for wind velocity

characterization in three-dimensional space. Prior to this thesis, this design was shown to be a very rigid structure when qualitatively compared to designs that do not utilize beams between each octahedral point (see Figure 32 of the Appendix for a picture of an octahedron design without select beams between point).

The use of *CUI* transducers (CUSA-TR80-15-2000-TH) with operating frequencies around 40 kHz was chosen with hopes to reduce structural vibrations, which were a major source of crosstalk in a previous design using solid-body transducers. The *CUI* transducers were designed such that the sound producing diaphragm was only connected to the driving piezoelectric, this contrasted with previously used *PUI* drivers which had diaphragms that were attached to both the piezoelectric and the body of the transducer (see Figure 6). Prior to this thesis, a switch to *CUI* drivers came from the hypothesis that the freer floating diaphragm would be coupled to the air alone and not transmit undesirable vibrations through the rest of the octahedron. This thesis includes the first utilization of *CUI* transducers in the Tufts anemometer during flight. *40* kHz has been a common frequency for low pressure operation as it balances resolution (based on wavelength) and attenuation. At higher frequencies, one could increase wind resolution, however, higher frequencies tend to attenuate more leading to lower signal strength. Even at 40 kHz there was noticeable noise at low pressures due to amplifier noise in the electronics, see Figure 39. The transducers were driven with roughly 25 Vp-p signals from the transmission amplifier aboard the Amplification/Multiplexor board. A notable downside to using the *CUI* drivers was their narrow resonant peaks (high Q factor) as this made producing a wide band acoustic signal difficult (see Ground Test Discussion).

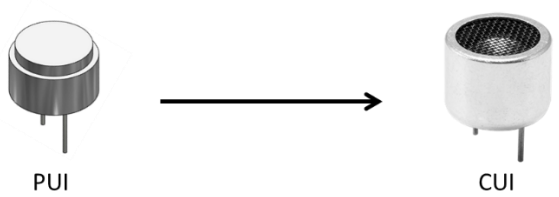


Figure 6, Example of PUI and CUI transducer construction. Note the solid construction of the PUI and the vibrating diaphragm under a wire mesh on the CUI.

A SparkFun Thing Plus SAMD51 was used as an intermediation controller between many of the systems sensors and the Raspberry Pi. The microcontroller drew power and transmitted data over USB to the Pi and connected to sensors with SparFun's *qwiic* (I^2C) system. The qwiic protocol allowed for sensors to be daisy chained together greatly reducing the need for available ports (the Pi only had 4 USB ports), see Figure 2. The only sensor not attached to the SAMD51 was a secondary IMU unit (VN100).

The system included two temperature sensors, a PT100 (resistive temperature detector) and a T-type Thermocouple. Both were placed by the octahedral head and were connected to the SparkFun SAMD51.

Two types of thermometers were utilized for their varying properties. The PT100 was chosen for its superior accuracy and the T type thermocouple for its ability to equilibrate quickly. In this iteration of the flight system, both temperature sensors were exposed to direct sunlight. Figure 23, shows a noticeable temperature bias during balloon float between the temperature sensors placed near the octahedra head and a sensor hidden from sunlight (NASA's sensor). A temperature sensor within the STM32 L467RG chip also reported data, however, this sensor was not used for any atmospheric

characterization since it resided in the electronics box and was purely utilized to monitor chip functionality.

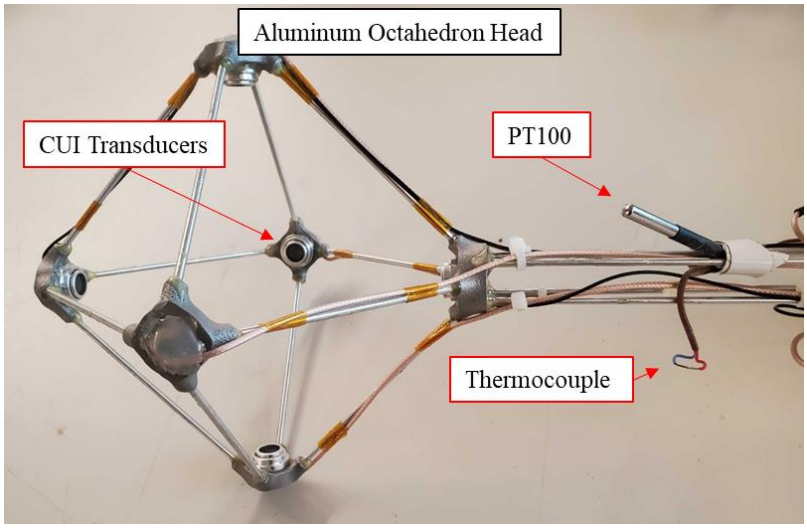


Figure 7, Photo of thermometers and CUI transducers on the custom alumium octahedron head before balloon launch. Note the large thermal mass of the PT100 compared to the Thermocouple.

A Pressure sensor on board was attached to the SAMD51 through the daisy chain. The BMP390 sensor by Bosch Sensortec had an operating range of 1250 mbar down to 300 mbar. The planned use for the system was up to an altitude above 30 km (below 11 mbar), which posed a problem for this sensor. Ultimately, a sensor from NASA's system was used for pressure measurements at high altitudes as it could operate below 300 mbar. Pressure was not a vital variable in any signal processing methods. The only case where pressure was a notable metric was in bell jar testing, where it was a substitute for altitude and derived from a lab grade capacitive manometer.

Two onboard Inertia Measurement Units (IMU) devices (VN100 and BNO080) were incorporated into the system. They were largely not used in the scope of this thesis, however their information could be useful in more extensive evaluation of the orientation

relative to world coordinates. The BNO080 was connected to the SAMD51 and the VN100 directly to the Raspberry Pi over USB.

A Global Positioning System (GPS) was connected directly to the Pi over USB. This device provided accurate time and position data to the rest of the data set and was vital in benchmarking the performance of the system. The position data could be turned directly into velocity data, thus giving a platform for comparison with the anemometer results. Flight metrics such as ascent velocity and the buoyancy frequency could be found with vertical GPS measurements.

A CUI INC. PYB30-Q48-T515-U power supply converted the gondola power (~24 V) to ±15 V for the transducer driving amps and +5 V for the Raspberry Pi. The Pi's USB hub acted as a power distribution point for many of the systems sensors and the STM32.

2.2 Software

The sonic anemometer utilized three main pieces of software; the main firmware on the STM32 which produced and recorded transducer signals, scripts on the Raspberry Pi that commanded sensors and stored their data, and a MATLAB algorithm that processed time-domain sensor data into velocity data post flight.

Since the STM32 was entirely new to the anemometer system considerable effort was placed into its code development by the author. *STM32CubeIDE 1.7.0* was used for programming and flashing of the development board. The Integrated Development Environment (IDE) worked well with a set of preconfigured settings and firmware. The device had to conduct two main operations: produce and record short segments of sound

(on the order of 10 ms) simultaneously. The production of a sound signal was handled by first the onboard synthesis, then production with the DAC. The signal was generated on startup or after any alternations to the desired start or end frequencies. It consisted of a linear chirp equation with a Tukey window function. The linear chirp was written in a for-loop and defined by,

$$x(t) = A * \cos\left(\omega_0 t + \frac{(\omega_0 - \omega_1)t^2}{2M}\right) \quad \textbf{(5)}$$

Where *A* is the amplitude, ω_0 is the staring frequency, ω_1 is the ending frequency, and *M* is the length of chirp. Since this was changeable through the command handling functions, a chirp could be altered into a pure sinusoid by setting ω_0 and ω_1 to the same value (the desired frequency). The Tukey window was also written in a for-loop and then multiplied datapoint by datapoint (*i*) in accordance with,

$$i_s = 0.5\alpha(N-1)$$

$$x(i) = \begin{cases} 0.5\left(1 + \cos\left(\pi\left(\frac{i}{i_s} - 1\right)\right)\right), & i < i_s \\ 1, & i_s < i < end - i_s \\ 0.5\left(1 + \cos\left(\pi\left(\frac{i}{i_s} - \frac{2}{\alpha} + 1\right)\right)\right), & end - i_s < i < end \end{cases} \quad \textbf{(6)}$$

Where *N* is the length of data points and α is a constant that controls window size. For the purpose of this study, an $\alpha = 0.5$ was used throughout. An example of

implementing the linear chirp and Tukey equations into C can be found in the Appendix at Figure 33.

Ensuring the timescale was accurate, and therefore producing the correct frequency, involved modifying internal timers and clocks. The discretization of the above continuous equations had to be conducted carefully to produce a desired sinusoid frequency. The DAC was clocked at 74 MHz with an 8 counter with a pre-scaler of 1 by following the DAC equation provided in the STM32 manual,

$$Freq_{OUT} = \frac{Freq_{CLOCK}}{SinePeriodBinSize(PreScale + 1)(Counter + 1)} \quad \textbf{(7)}$$

To achieve the desired waveform and ample resolution a *SinePeriodBinSize* of ~200 was used. An overall bin of 8192 allowed for a 2 ms long sinusoid and approximately 80 cycles to be produced. Using a DMA controller, the waveform was delivered to the DAC which produced the signal on one of the STM32's pins, see Figure 8. The resulting frequency was verified with an oscilloscope.

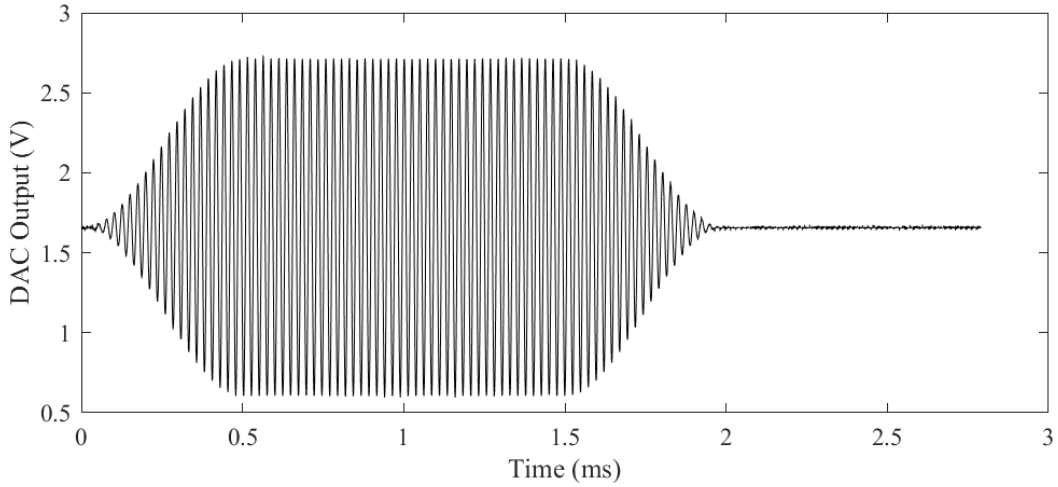


Figure 8, Graph of 41 kHz Tukey windowed burst produced on the STM32 DAC and captured with the STM32 ADC.

The signal receiving (ADC) code was simpler in relation to the clocks, however, a problem of enough RAM became a limiting challenge. With only 128 KB, storing the signal to be transmitted (8192 uint32 = 32 KB) and two received signals (2x8184 uint16 + 16368 uint16 = 66 KB) populated most of the RAM. Due to the high memory throughput of the system, RAM was the only form of memory used. The onboard flash (1 MB) was only utilized for information used in system initialization. Optimizing memory management involved increasing the system's resolution within a limited RAM space. The ADC ran at 2.933 Msps based on a 44 MHz clock and 15 clock cycles needed to resolve a single point. See Appendix Figure 34 for the system's clock configuration.

All major internal data transfers were handled with DMA controllers. This included RAM to DAC and ADC to RAM transfers. Direct Memory Access allowed the code to operate ~5-10x faster than without it. Its implementation was necessary due to the system completing a full cycle (send and receive 6 chirps with the offloading of all the received signals) in 2.25 s with DMA making it rather slow when compared to a desired goal of 10 Hz (0.1 s cycle time). Even with DMA the system could benefit from further optimization. Other onboard sensors, such as the temperature, GPS, pressure, and IMU units from the SAMD51, were less important and were run at 5 s intervals to conserve resources. Select settings from the IDE relating to DAC, ADC, UART, and DMA configuration can be found in Figure 35 of the Appendix.

Other than signal production, capture, and off-boarding; the STM32 needed a script to handle incoming commands from the Raspberry Pi. These commands were sent over serial and included: chirp frequency modifiers, chip temperature indicators (sent

over serial when requested), enabling of transducer amplifiers (handled with a GPIO pin on the STM32), selection of transducer pair (also GPIO controlled), changing of receiver gains, and the requests for time domain signal data from the board. All commands were sent directly over serial to the STM32 USB UART connection which triggered an interrupt prompting the code to read the alpha-numeric command and initiate the relevant code. This system worked well in testing but was limited to two character inputs and would report an error if the character length was inappropriate.

Table 3, Serial commands recognized by the STM32 firmware when sent over UART (USB).

Command	STM32 Action
T1, T2, T3, T4, T5, T6	Selects transducer for transmission
G1, G2, G3	Selects gain of receiver amplifiers (0 dB, 20 dB, 40 dB)
S1, S2, S3, S4 / S5, S6, S7, S8	Sends time-domain signals in 4 part chunks (S1-S4/S5-S8) for received / transmitted signals
E1, D1	Enables and disables transducer amplifiers
V1	Initiates signal production and capture
C1	Sends internal temperature of STM32 MCU
1U, 1D	Raises (1U) and lowers (1D) the chirp start frequency, ω_0, by 1 kHz
2U, 2D	Raises (2U) and lowers (2D) the chirp end frequency, ω_1, by 1 kHz

When data was passed off to the Raspberry Pi with the *S* commands it had to be broken up into smaller chunks. This was due to the limited buffer size of the Linux (Raspbian OS) kernel. Without modification, the UART/Serial buffer of the Pi had a maximum of 4 KB making it 4 times smaller than a captured signal of ~16 KB. By splitting the signal up into 4 chunks, a chunk of data could be sent over to the Pi then stored briefly in a Python script where it was then combined with the 3 subsequent parts and then saved to the system's SD card. While this workaround achieved the desired

result, an optimized transmission method (possibly changing the protocol or keeping wind velocity calculations on the STM32) could increase performance. For increased performance, the uint16 signal values were sent in binary over serial to conserve characters. For example, serial generally uses a *char* value which takes up 8 bits per character (i.e. *4094*=4 bytes or 32 bits), but when binary is utilized a value such as *4094* only uses 12 bits. This was used to increase the throughput of the 1 Mbit/s UART controller.

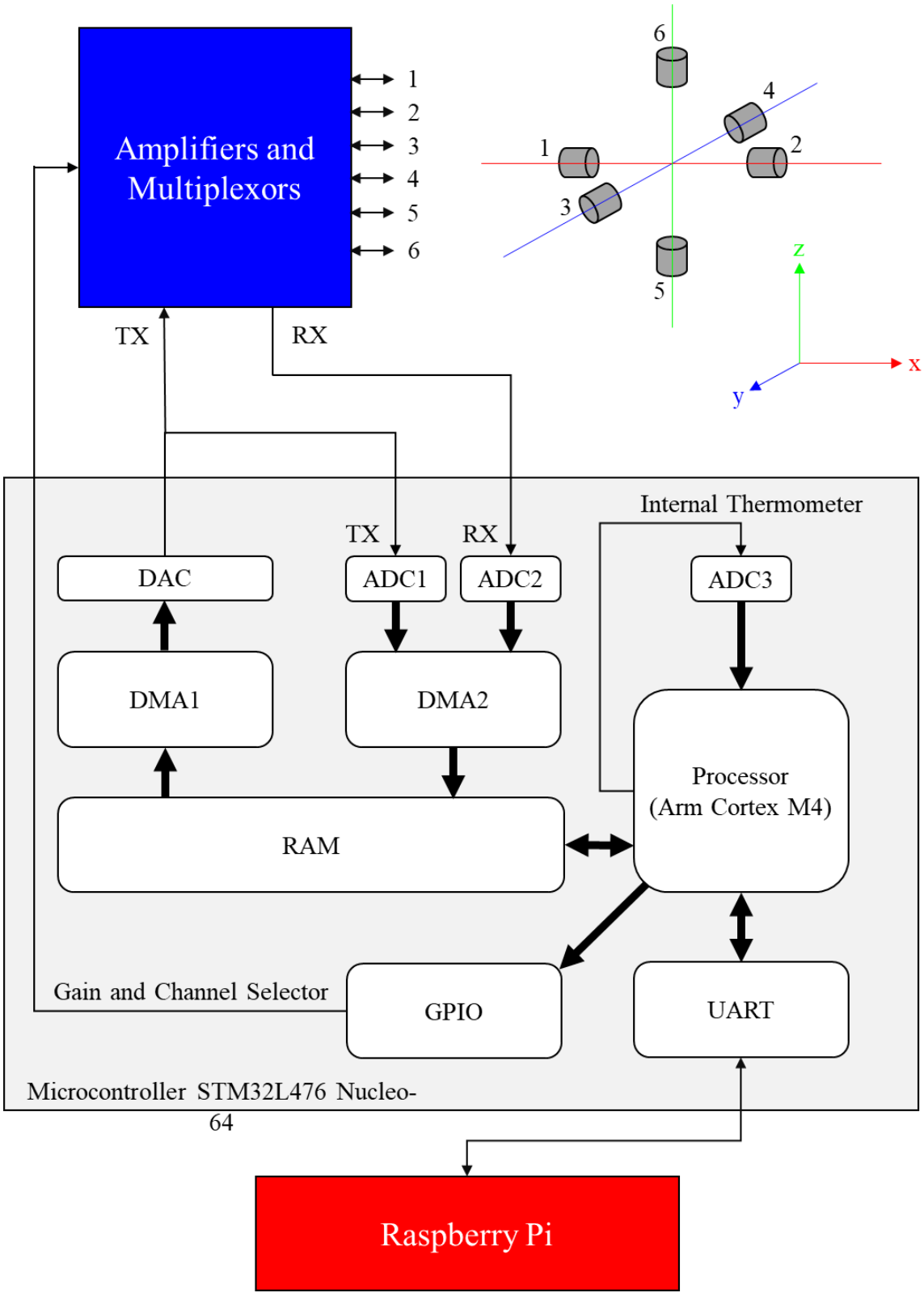


Figure 9, Diagram of STM32 data transfer structure.

The Raspberry Pi began running Python scripts when it was turned on by placing a file in the boot directory. This insured full coverage of any flights because gondola power was only activated right before flight. The scripts had one main goal, intake sensor information and save it to the Pi's internal storage. One program took data from the IMU, pressure, GPS, and temperature sensors every 5 s and wrote them to a *.csv* file. A second program took data from an additional IMU sensor (VN100) and wrote it to a *.csv* file. The final Python script interfaced with the STM32 to produce a *.csv* file for organization, internal temperature, and set gain. In addition, the production of time domain signal files created and stored with the *.bin* extension. This last program sent out commands to the STM32 guiding it to systematically produce signals on all 6 of its transducers and send the resulting data back after each received signal. It took the aforementioned chunks and combined them into a full continuous received waveform. The Active Gain Control (AGC) was also handled in this script. By finding the maximum magnitude of the signal the program would command the STM32 to change gains when the incoming signal was too low or too high. During initialization, frequencies for both the beginning and end of the chirp were set by sending the appropriate commands to the STM32.

Post processing of the flight data was conducted exclusively in MATLAB. This included telemetry, environmental, and time domain anemometry data handling code. The telemetry and environmental data processing was straightforward, usually resulting in a graph. Conversely, the anemometry data processing methods were undefined before this study and needed to built up and verified.

3 Ground Test

Before flight, the system needed to be evaluated for mechanical strength, power consumption, weight, and environmental readiness. The sensor head was modeled in FEA and physically tested by applying a 10 *g* (10 times static load) lateral load to ensure sufficient mechanical strength. The sensor head by itself massed 180 g including frame, transducers, and internal cables. The telemetry sensors and datalogging and control electronics massed 960 g. The enclosure (Polycase hardbox) massed 1.4 kg, and 2 m of cables between the sensor head and electronics massed 500 g. Therefore, the total mass of the system was 3.0 kg, although more than half of this was the enclosure and cables. Total power draw for the system was 5.3 W. The microcontroller by itself consumed 300 mW under full load. The amplification boards consumed 1.2 W, and the Raspberry Pi and telemetry sensors consumed 2.5 W for a 4.0 W total. The DC-DC power converter was 75% efficient resulting in 5.3 W of required power. A post processing method was needed to convert time domain signals into accurate wind velocity. This was accomplished by using data gathered with the sensor head in a bell jar and MATLAB. Comparisons of physical model (transfer function) use, chirp bandwidth, interpolation, and correlation methods were made to determine an optimal approach to post flight data processing. These are described in the remainder of the chapter.

3.1 Transducer Model

This thesis explores the implementation of a transducer model to evaluate its usefulness in increasing accuracy and reducing ambiguity. A spring-mass-damper model

was used to fit the acoustic transmission mode measurement of the transducers. This resulted in a transfer function of,

$$TF(s) = \frac{Output}{Input} = \frac{\omega_n^2}{s^2 + \frac{\omega_n}{Q} + \omega_n^2} \quad (8)$$

Where ω_n is the natural frequency and Q is the Q-factor (akin to the inverse of damping) of the transducers. The model was applied twice to the reference signal, once for the transmitting transducer and once for the receiving transducer. The modified reference signal was then used for correlation with the received signal. The goal was to remove any effective delays produced by the phase lag in the transducers themselves. A similar procedure could be used to correct for the amplifier transfer functions and diffraction, but was not implemented in this work. The implementation of the transfer function in MATLAB can be found in Figure 38 of the Appendix.

3.2 Correlation Methods

Research on different ways to obtain ToF from time domain waveforms is quite vast [47]–[52]. Many include Fast-Fourier-Transform (FFT) for implementation flexibility. For this high-altitude anemometer, a few common methods were chosen for comparison: FFT Cross-Correlation (FFT-CC), Generalized Cross-Correlation with Phase Transform weighting (GCC-PHAT), and Phase Shift (PS).

3.2.1 FFT-CC

By taking the Fourier Transform of both the transmitted, $s_{TX}(t)$, and received, $s_{RX}(t)$, signals then multiplying them together creates a cross-spectrum in the frequency

domain [50]. An inverse Fourier Transform brings the cross-spectrum into the time domain and a max correlation position can be found for ToF. Note that the cross-correlation method proposed in this study is identical to calculating the discrete covariance of the two signals.

$$S_{RX}(\omega) = FFT\big(s_{RX}(t)\big) \tag{9}$$

$$S_{TX}(\omega) = FFT(s_{TX}(t)) \tag{10}$$

$$S_{CC}(\omega) = S_{TX}^{*}(\omega) * S_{RX}(\omega) \tag{11}$$

$$s_{CC}(t) = IFFT(S_{CC}(\omega)) \tag{12}$$

$$ToF = t \ @ \ Max(s_{CC}(t)) \tag{13}$$

This method greatly resembles the functions: “xcorr” in MATLAB [53] or “arm_correlate” in the CMSIS DSP library [54] for ARM processors. The latter being a library native on the STM32 platform. Since the FFT algorithm is widely utilized in computation, FFT-CC is a well-established method for convolution. However, it has some limitations in ToF calculations. Without subsampling, it is limited to a resolution equal to the sampling rate of the input waveforms. The method also benefits from complex signals with diverse frequency content, which can limit hardware choices and

increase signal production time. Figure 37 of the Appendix shows the MATLAB code used for FFT-CC in this thesis.

3.2.2 GCC-PHAT

GCC-PHAT follows a similar procedure to FFT-CC with the addition of a weighting factor in the frequency domain.

$$S_{CC}(\omega) = \frac{S_{TX}^{*}(\omega) * S_{RX}(\omega)}{|S_{TX}^{*}(\omega) * S_{RX}(\omega)|} \quad \textbf{(14)}$$

this was accomplished in MATLAB with “gccphat” [55]. GCC-PHAT has been shown to reduce variance in ToF calculations in situations where the received signal contains additional signals (usually from reflections or alternate transmission paths) that arrive close in time to the main signal [50]. Computationally it is similar to the standard FFT-CC with only the addition of a division before the inverse FFT. However, the algorithm is more sensitive to noise and can perform poorly when the received signal has an SNR below 7 dB [49], [50]. There are other weighting factors such as SCOT, ROTH, Weiner, Eckhart, etc. used in ToF measurements. Some of these methods aim to improve performance in low SNR conditions but are outside the scope of this thesis.

3.2.3 PS

The phase shift method can be thought of as an equivalent ToF for a fluctuation in phase [5]. The flow velocity can be found with equations (1) and (2) and the following,

$$\Delta ToF = \frac{\Delta\varphi}{\omega} \tag{15}$$

$$\Delta\varphi = \omega(ToF_{12} - ToF_{21}) \tag{16}$$

$$\Delta\varphi = \omega\left(\frac{L}{c + \boldsymbol{v}_{12}} - \frac{L}{c - \boldsymbol{v}_{12}}\right) \tag{17}$$

$$\Delta\varphi = \frac{2\omega L\boldsymbol{v}_{12}}{(c + \boldsymbol{v}_{12})(c - \boldsymbol{v}_{12})} \tag{18}$$

$$if\ \boldsymbol{v}_{12} \ll c, then \tag{19}$$

$$v_{12} \approx \left(\frac{\varphi_{12} - \varphi_{21}}{2\omega L}\right)c^2$$

where ω is the drive frequency of the transducer and φ is the phase found from the FFT of the received signal. The speed of sound can be found through the summing received signal phases. Although an offset terms must be added to account for total initial phase shift in the system under a no flow condition at a known speed of sound, φ_0 and c_0.

$$\varphi_{12} + \varphi_{21} = \frac{-2\omega Lc}{(c + \boldsymbol{v}_{12})(c - \boldsymbol{v}_{12})} + \varphi_0 \tag{20}$$

$$if\ \boldsymbol{v}_{12} \ll c, then \tag{21}$$

$$\varphi_{12} + \varphi_{21} \approx \frac{-2\omega L}{c} + \varphi_0$$

$$\frac{1}{c} \approx \frac{-(\varphi_{12} + \varphi_{21} - \varphi_0)}{2\omega L} + \frac{1}{c_0} \tag{22}$$

Note that this method needs an initial speed of sound, c_0, and phase shift. The φ signals are the change in received phase from the reference phase measured at $c = c_0$. There is a challenge in using phase which is the 2π ambiguity of single tone phase measurements [5], [6]. Resolving the 2π ambiguity requires either multiple frequencies to be used or a time domain unwrapping method [1], [5].

3.3 Interpolation Methods

Interpolation in cross correlation methods is quite common when evaluating ToF [51], therefore a comparison of interpolating the time domain signals and/or FFT-CC data was conducted. Both cubic spline [7], [56] and lowpass [57] interpolation were also compared. Cubic spline was used by other sonic anemometer researchers [7] and lowpass is the native MATLAB algorithm for interpolating data ("interp" function [57]). Since the signals and FFT-CC results are sinusoidal, subsampling on a cubic or frequency bases is assumed to be an appropriate approach with minimal bias errors [51]. Lowpass interpolation is also referred to as 'reconstruction interpolation' or 'zero-padding (in the

frequency domain)' as it uses the frequency make up of the waveform to build a higher resolution version of itself.

3.4 Test Methods

Ground testing was performed with the sensor head inside an 18 inch diameter, 30 inch tall glass bell jar. To reduce variability during ground testing, pressure, gas composition, and temperature were closely monitored. Internal gasses were evacuated with a rotary vane pump to a base pressure of 0.1 mbar. Both a capacitive monometer and thermocouple vacuum gauge were used to monitor pressure. Dry air (20-22% O_2, 78-80% N_2, <7 ppm water vapor) was used for back filling. The temperature of the air was monitored using three T-type thermocouples at various heights. Temperature could be reduced using a liquid nitrogen cold plate.

3.4.1 Bell Jar Conditions

The bell jar was prepped by pumping down to less than 0.1 mbar, then purging the chamber 10 times with dry air, pumping down to 1 mbar after each purge. A final pressure in dry air was held at 10 mbar for most of the testing. Temperature inside the bell jar was nominally left at room temperature (~20 °C) apart from a varying temperature experiment to compare the change in speed of sound as a function of temperature from the sonic anemometer to a theoretical model, as discussed below.

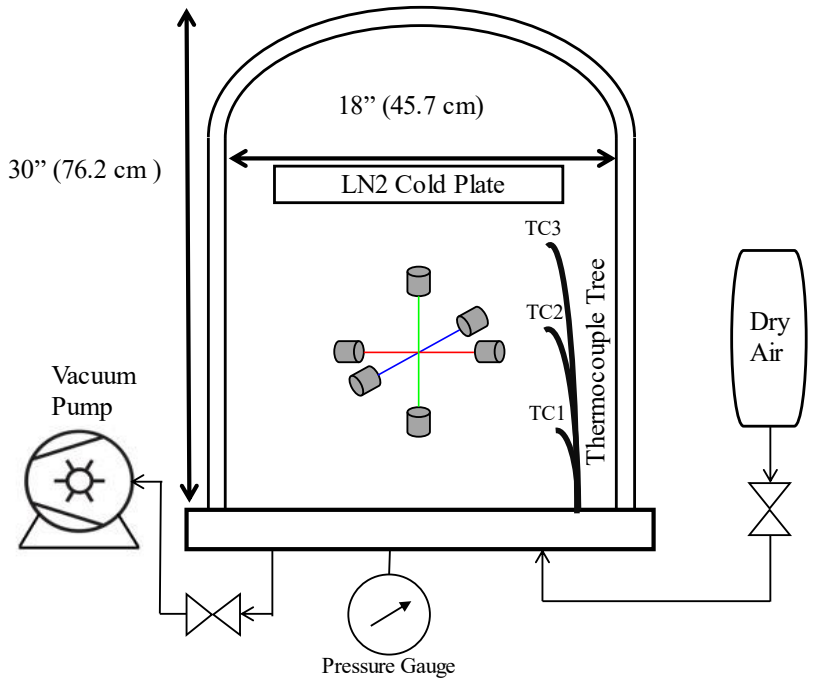


Figure 10, Bell jar diagram. Note the thermocouple tree made up of 3 T-type thermocouples at ascending heights.

3.4.2 Software

The sample rate of the ADC was set to 2.933 Msps (44 MHz / 15 cycles per point). This was a good balance between resolution and window size with a limiting amount of RAM. The microcontroller produced an adjustable linear chirp with a Tukey window (r=0.5) and ~2 ms duration (see RAM limitation and chirp production code in The System, Software for more detail). The beginning and end frequencies were varied around the resonant frequency of the transducer to find the best band. The adjustability also allowed for bursts of a single frequency to be tested. The desired waveform was sent through a DAC and into the amplifier as well as directly back into an ADC to record reference signals. Resulting time domain data (RX/TX) from the microcontroller was sent

to a Raspberry Pi where it was stored for retrieval. The system could retrieve bidirectional 3-axis data at a rate of 0.44 Hz. This meant that 6 time-domain waveforms were sent every 2.25 s. Data on board the microcontroller was handled with DMA when being moved between RAM, DAC, and ADCs.

All post processing was done in MATLAB, including transducer model corrections, interpolation, and correlation functions. The correlation functions utilized were FFT-CC, GCC-PHAT, and PS. This provided a comparison of algorithms. A few other methods were considered, such as GCC-SCOT [58] and L2WR [59], however, their performance differences were not large enough to warrant further testing. The FFT-CC method utilized a Prior Estimate (TDPE [60]) approach to help eliminate peak ambiguity and improve processing time. A ToF from FFT-CC was found by determining the maximum of cross-correlated (FFT-CC) data (example in Figure 12). Spline and lowpass interpolation were compared in three configurations applied to the RX and TX time domain data, applied to the time domain cross correlated data, and applied to the RX, TX, and correlation data. Each subsampling was done at a 4x resolution increase, meaning latter configuration ended with an 8x resolution increase resulting in an effective 42.6 ns data spacing (or 0.03 m/s resolution when c = 340 m/s, L = 0.12m, and v_{12}=10 m/s).

An offset when measuring the speed of sound was subtracted from the data to account for hardware time delays (or hardware phase shifts) from the amplification and multiplexer circuitry. This was found by estimating the speed of sound based on temperature and subtracting a portion of the ToF (from the sonic anemometer data) until the theoretical speed of sound and the anemometer's measurements agreed. The offset calibration factor was used both for ground tests and for the balloon flight.

3.5 Results

3.5.1 Transducer Models

A proposed method to improve accuracy in this thesis was the use of a physical model (transfer function) to account for the signal passing through two transducers (TX and RX). The models were applied to a reference signal captured from an ADC prior to performing the cross correlation with the received waveform. Physical property estimates of ω_n=39 kHz and Q=30 were used based on transducer impedance and transmission mode testing, see Figure 11. A comparison was made with and without the model using a 36-42 kHz chirp and a 41 kHz burst (a 41 kHz burst was used on the balloon flight to 38 km). This test resulted in a preference toward using the transducer model when a burst was used, and *not* using the model when a chirp was used. With the sharpest overall envelope peak being a chirp without a model, see Figure 13.

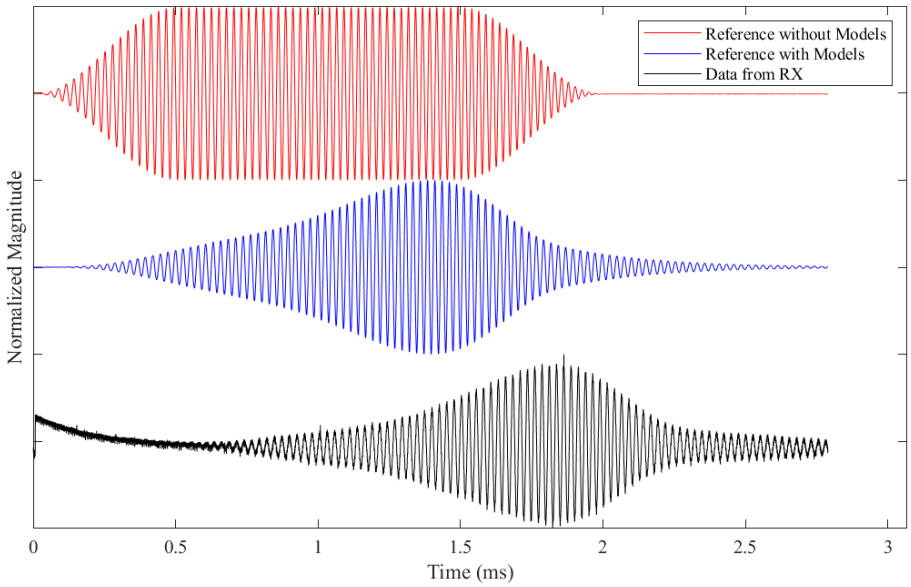


Figure 11, Time domain graph showing TX Reference signal (red) without and with a physical model (blue) as well as an RX signal (black) after passing through both transducers. Note the DC bias and noise at the beginning of the RX data which could interfere with GCC-PHAT.

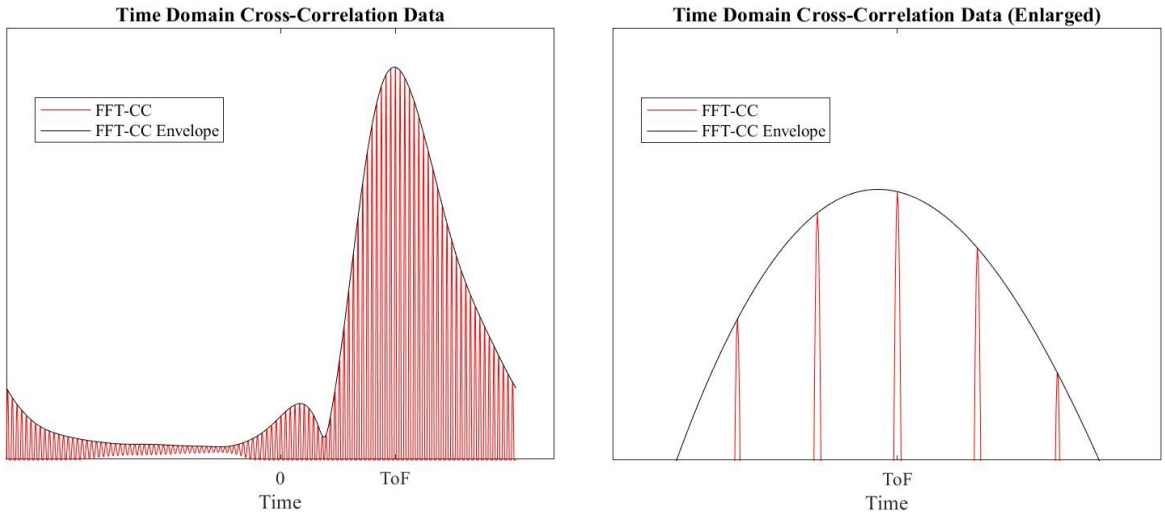


Figure 12, FFT-CC applied to an 2 ms windowed chirp from 36-42 kHz at 10 mbar and 20C. Note the possibility for peak ambiguity of 'FFT-CC' (red) as the envelope peak widens. Values of ToF were taken from peaks of the red line.

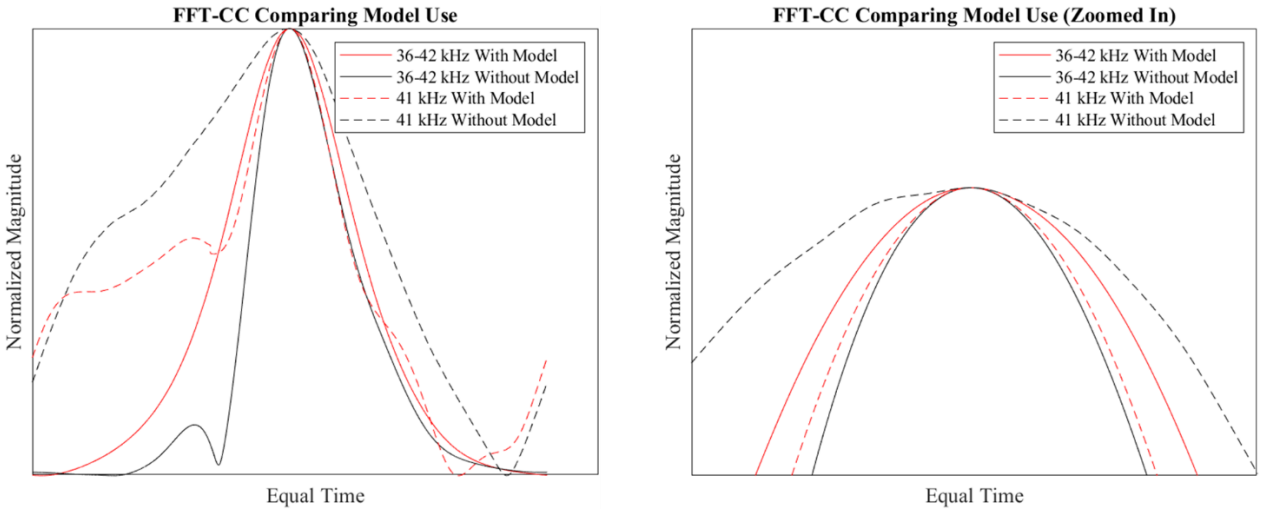


Figure 13, FFT-CC envelope comparison of chirps and busts with and without the use of a physical model. Note that the model makes the burst peak sharper and the chirp peak wider.

Observing the sharpness of peaks was not the only method to characterize the performance of the anemometer and signal processing method. It can only be utilized with FFT-CC and the main benefits of a sharper peak was a reduction in peak ambiguity, which was largely mitigated with a different strategy of previous estimates [60]. In order to evaluate any further methods, such as interpolation or PS, the standard deviation of the flow velocity was used as a way of comparison. Comparing the peak sharpness did help characterize the frequency response of the transducers, helping to determine an appropriate start and end chirp frequency for further testing.

3.5.2 Comparing Linear Chirp Ranges

According to theory [50], the wider band of frequencies a transducer can produce the better correlation between signals can be found due to a more distinguished ToF peak, see Figure 12. Since this device utilized narrowband transducers (Q-factor ≈ 30), an exploration into different bands was conducted, see Figure 14, to evaluate the capabilities of the CUI transducers to produce off resonant frequencies. An optimal frequency chirp range appeared to be around 36-42 kHz, with any further widening of frequency band leading to no further sharpness. This was likely due to the narrowband nature of the transducer; it simply lacked the ability to effectively produce frequencies lower or higher than the 36-42 kHz range. Overall, the chirps had sharper peaks than the bursts which was expected, and the 36-42 kHz range appeared to be an optimal waveform and was used for further optimization strategies. Additional testing could be conducted with chirps not centered around the resonant frequency. However, the 36-42 kHz range

appears to be near the frequency production limits of the transducers based on manufacturer specifications (see Figure 36 of the Appendix).

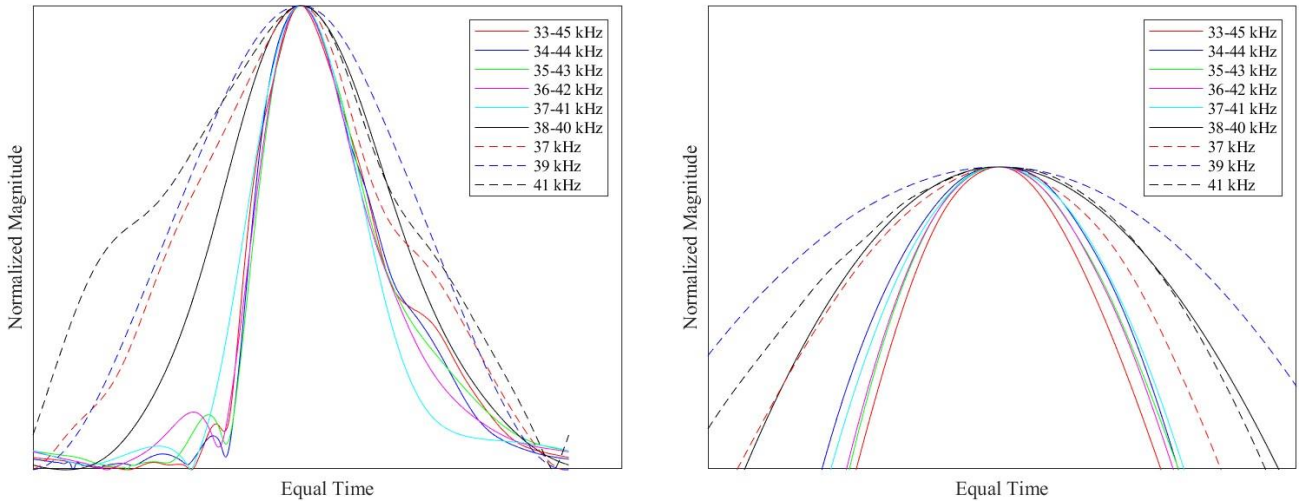


Figure 14, Comparing effects of chirp and burst frequencies on cross-correlation. Note that the bursts tend to have wider envelope peaks and the chirp peaks do not change much after 37-41 kHz (light blue). These tests did not include physical models or interpolation.

3.5.3 Interpolation

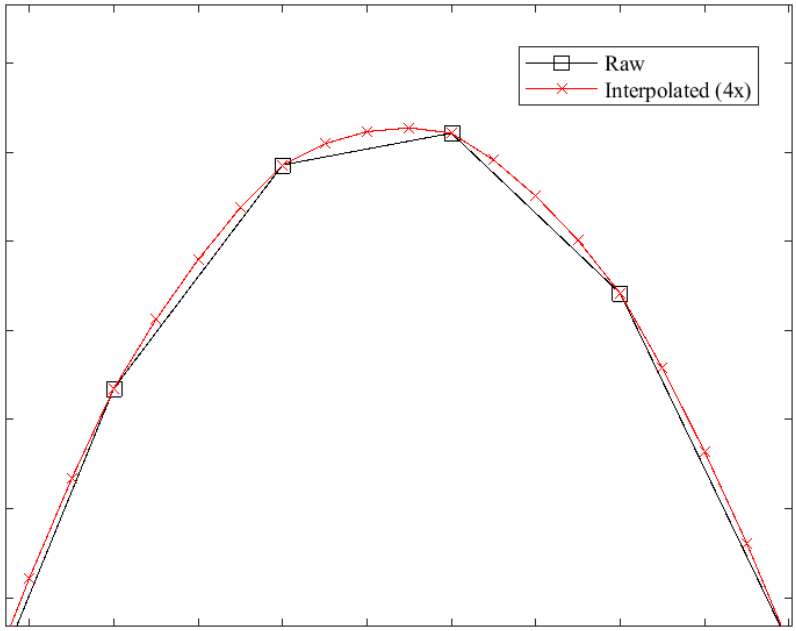


Figure 15, Example of lowpass interpolation (subsampling) of data at a 4x resolution increase. Note that when peaks are between data points this can improve precision of ToF.

Interpolation was conducted in 3 different ways: on the RX/TX waveforms, on the cross-correlation results from FFT-CC (CC interpolation), and a combination of applying interpolation to both the RX/TX and the cross-correlation waveforms. Table 4, shows the resulting standard deviation of wind velocity within the bell jar. These results show minimal improvements when applying interpolation to data collected from this device. Notably, applying transducer models helped the constant frequency burst regardless of interpolation but had minimal effect on the chirp velocity data. This was likely a result of the peak sharpening phenomena that can reduce ambiguity as to which cross correlation peak was the maximum (most correlated). The comparison of Lowpass and Cubic Spline interpolation shows no appreciable differences. CC interpolation appeared to be less effective as RX/TX interpolation but could reduce processing power as the subsampling happened after FFT and was applied to a single waveform (TX/RX interpolation required the subsampling of both the TX and the RX waveform). The CC interpolation could be used with truncated data about the peak of interest, further reducing processing time.

Overall, the data did not improve much when adding interpolation to the signal processing which was uncharacteristic according to expectations from the literature. This, and a similarity of wind measurements even after interpolation, see Figure 16, led to an exploration into the possible noise and error sources within the system.

The most impactful improvement was observed with bursts, therefore a recommendation to use interpolation with balloon data (captured with 41 kHz bursts) can be made. The wind data of Table 4 also shows the benefits of using a transducer model

when a burst was being processed. Note that the values of Table 4 are not zero, meaning there was considerable noise in the system, see Error Sources for further exploration into this phenomenon.

Table 4, Standard deviation of wind velocity (m/s) after applying interpolation different ways. Lower is better. There is no noticeable difference between Lowpass and Cubic Spline interpolation when considering RMS deviation.

	36-42 kHz Chirp, FFT-CC, No Models				41 kHz Burst, FFT-CC, No Models	
	No Interpolation	TX/RX Interpolation	CC Interpolation	Combined Interpolation	No Interpolation	TX/RX Interpolation
Lowpass	0.1159 (0.1031*)	0.1098 (0.1164*)	0.1100	0.1098	0.1219 (0.1052*)	0.1086 (0.0884*)
Cubic Spline	0.1159 (0.1031*)	0.1098	0.1103	0.1098	0.1219 (0.1052*)	0.1089

*With Transducer Models

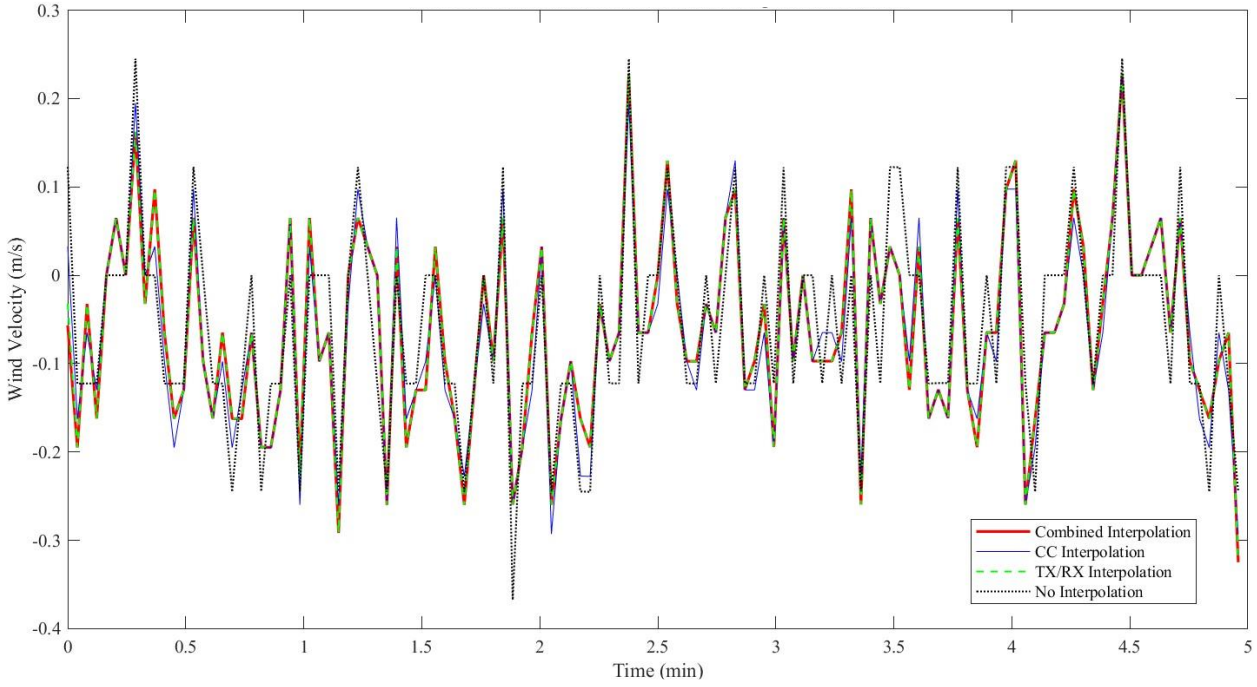


Figure 16, Graph of wind data (122 data pt.) collected with 36-42 kHz chirps at 10 mbar and FFT-CC. Note how similar the data points are even after interpolation.

3.5.4 PS and GCC-PHAT

Until this point, FFT-CC (a direct cross correlation equivalent) was the main method for determining ToF. However, other algorithms that could potentially improve wind measurements were investigated including PS and GCC-PHAT. PS was found to be a viable method, see Table 5, with a disadvantage of phase ambiguity that needed an estimation of speed of sound (found from temperature) to operate. GCC-PHAT failed to determine reliable ToF data, likely do to a low signal to noise ratio [50], [61], possibly from internal noise from the amplification electronics. Further exploration into GCC-PHAT can be found in the Error Floor analysis. All the methods appeared to trend toward a lower limit, about 0.1 m/s, when evaluating standard deviation indicating some sort of noise in the system prompting an exploration into Error Sources.

Table 5, Comparison of correlation algorithms and their resulting effect on wind velocity standard deviation (m/s).

		FFT-CC		PS		GCC-PHAT	
		36-42 kHz	41 kHz	36-42 kHz	41 kHz	36-42 kHz	41 kHz
No Interpolation	No Model	0.1159	0.1219	0.1641	0.1108	SNR to Low	
	Model	0.1031	0.1052				
RX/TX Interpolation	No Model	0.1098	0.1086				
	Model	0.1164	0.0884				

3.5.5 Temperature Variation

An 8-hour temperature varying test was conducted from 10 to 40 mbar with a temperature variation of 0 to 20 °C. This test helped verify that the sonic anemometer was operating correctly within the bell jar. For this test, 2 ms long bursts of 39 kHz were used. Both PS and FFT-CC with all its optimizations worked well. The speed of sound

measured by the sonic anemometer fit well with theoretical expectations from air temperature as seen in Figure 17. The ideal gas model used for theoretical speed of sound followed,

$$c = \sqrt{\frac{\gamma R T}{M}} \tag{23}$$

where $\gamma = 1.401$ is the ratios of specific heats, R is the ideal gas constant 8.3145 $8.3145\,\frac{\mathrm{J}}{\mathrm{K*mol}}$, T is the temperature in Kelvin, and $M = 28.85\,\frac{g}{mol}$ is the molar mass of dry air.

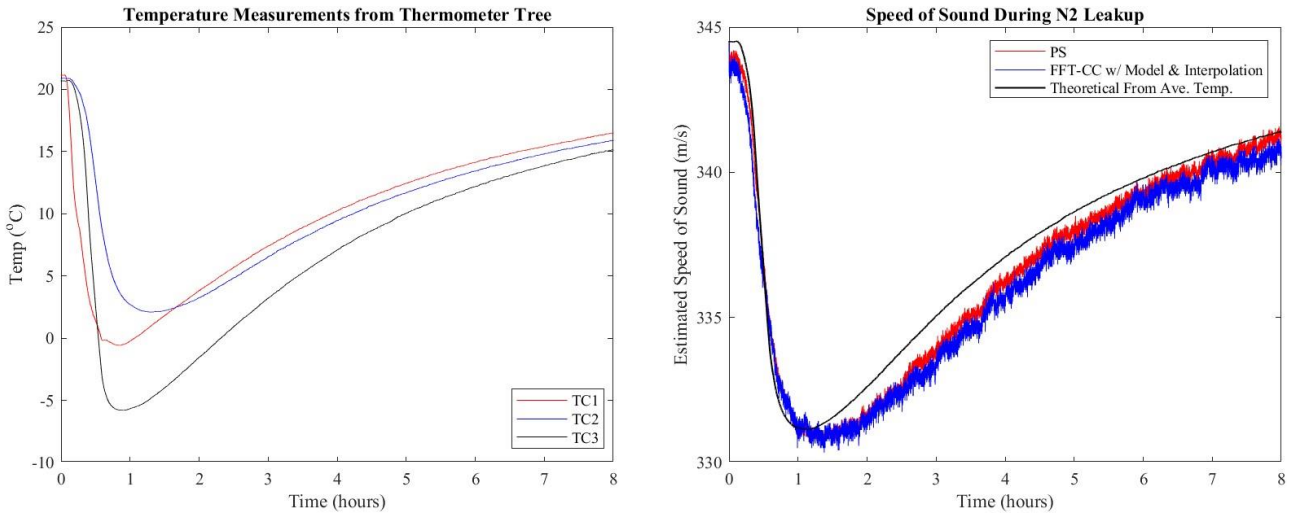


Figure 17, Bell jar temperature readings from the thermometer tree of sensors at different heights (left). Comparison of speed of sound measured from sonic anemometer data and from expectations based on temperature (right).

3.6 Error Sources

3.6.1 Estimation of Errors

Table 6, Table of expected error sources and their impact on the system.

Type of Error	Estimated Error Size (m/s)
Sampling Frequency Quantization (without interpolation)	$3x10^{-1}$ (0.316 for c, or 0.153 for v)
Bell Jar Flow*	$\sim10^{-3}$ (0.001)
ADC Clock Fluctuations**	$5x10^{-5}$ (0.00005)

*Based on typical order of magnitude under natural convective flow of a heated vertical plate when T1 = 21.8 °C, T2 = 22.4 °C, and 8 mbar.
**Error of 0.25% (specified in developer board data manual) on 44MHz ADC clock. This was propagated through speed of sound equations based on ToF, c = 340 m/s, v = 0, and L = 0.12 m.

Estimations of specific error magnitudes were made, see Table 6, to investigate possible contributions to error. This shows a large portion of error could come from the quantization of digital data. This is likely why interpolation is commonly used. The ADC clock fluctuations were an estimation based on the manufacturer's claims on clock accuracy. Clock jitter and internal ADC noise were not estimated but could have been a large source of error.

3.6.2 Error Floor

To explore sources of error, a test was performed which isolated the microcontroller from its accompanying multiplexors, transducer amplifiers, and transducers. This was done by connecting the output of the TX DAC directly to the RX ADC of the development board. It was assumed that any error found with this method could be an ideal target for the current system as a whole (i.e. with amplifiers and transducers present). This limit acted as a floor that was inherent to the digital system.

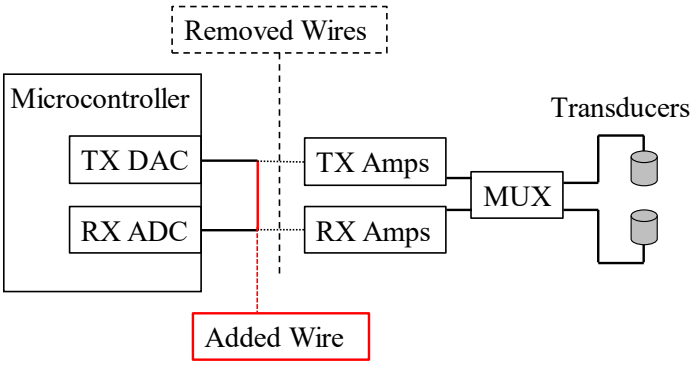


Figure 18, Diagram of isolated microcontroller by removal wires to external hardware and an added short.

The approach to isolate just the microcontroller to find its error range was conducted over 5 min of data taking with a 36-42 kHz chirp to match previous testing parameters. This resulted in an error floor not much lower than data collected within the bell jar with true acoustic signals (std. of ~0.06 m/s when isolated compared to ~0.09 m/s in the bell jar with all acoustic hardware active).

In addition, GCC-PHAT was tested on the data collected with the isolated STM32 to see whether its instability to perform on real data was due to SNR problems from the rest of the system. GCC-PHAT requires an operating range above a 5-10 dB SNR [50], [61]. Since the RX waveform an had SNR of approximately 9 dB within the bell jar (found with MATLAB's "snr" function) it appears that the noise level could have been too large for GCC-PHAT leading to poor results. When tested under an isolated microcontroller GCC-PHAT functioned as expected, see Table 7.

Table 7, Standard deviation of wind velocity (m/s) with an isolated microcontroller.

36-42 kHz Chirp, Isolated Microcontroller					
FFT-CC		PS		GCC-PHAT	
Raw	TX/RX Interpolated	Raw	TX/RX Interpolated	Raw	TX/RX Interpolated
0.0824	0.0623	0.0569	0.0571	0.0809	0.074±0.005*

*Depended on where TX/RX window was made before GCC-PHAT algorithm.

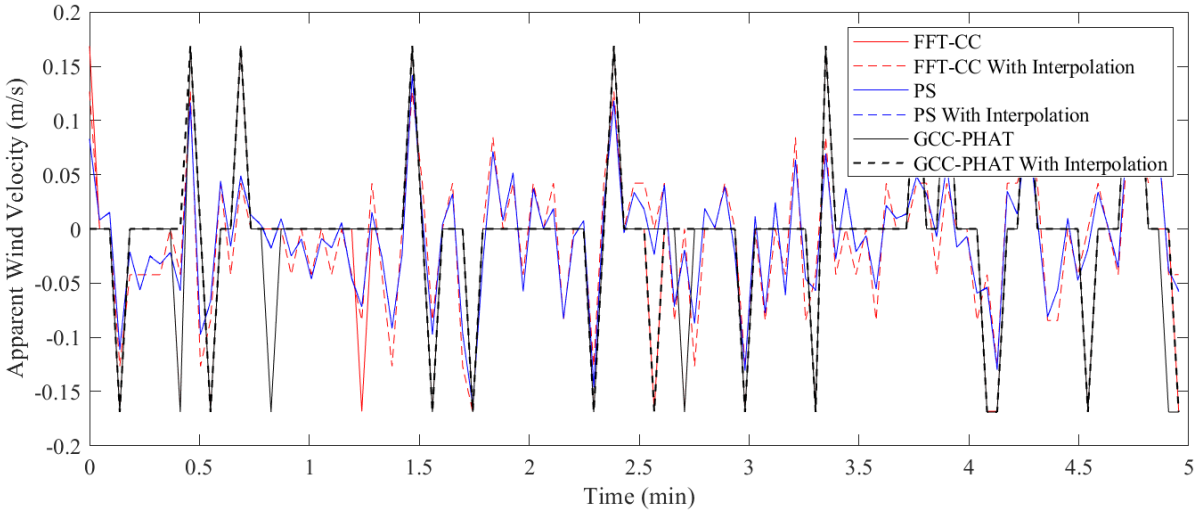


Figure 19, Apparent wind velocity variations found through different algorithms with an isolated STM32 board (no acoustic hardware). Note that GCC-PHAT and PS do not visually improve with interpolation, and that FFT-CC (without interpolation) and GCC-PHAT produce similar results. This level of variation is present purely in the digital hardware and is not due to the amplification/multiplexer electronics, transducers, or acoustics.

3.7 Ground Test Discussion

Other ToF methods were considered but not tested. For example, using simulated reference waveforms or forgoing reference waveforms altogether could help improve results for windspeed but were not attempted. Further exploration into higher interpolation subsampling, reducing sampling rate (<3 Msps), shorter signals, other GCC weighting strategies (not PHAT), and reference wave elimination are all possible avenues moving forward. However, the error floor analysis indicates a limiting factor being the internal noise of the STM32L476RG development board. The microcontroller was by no means high end (~$15 USD) or designed for precision data acquisition. The development board did not implement noise reduction strategies, such as ground shielding of analog

signal on the PCB or in the die. Future progress could be made to shift off a consumer development board to a purpose-built device for higher performance. Future iterations could also include wider band transducers which could reduce the peak ambiguity phenomena.

Overall, ground testing indicated that the system using pure tone bursts near 41 kHz or chirps in the 36-42 kHz band could resolve wind and speed of sound with an approximate resolution under 0.05 m/s and standard deviation of 0.1 m/s. Both the PS and FFT-CC methods worked. However, GCC-PHAT did not perform well due to SNR limitations. Interpolation and transducer model preprocessing did not have substantial impacts on random variance in measured velocities. The FFT-CC method needed a prior estimates approach [60] to greatly reduce peak ambiguity in the correlation data. 2pi phase wrapping ambiguities were a weakness for PS processing and were mitigated by incorporating external speed of sound values and use of MATLAB's *unwrap* function.

4 Balloon Flight

A stratospheric flight was conducted on a NASA balloon out of New Mexico in August 2022.

4.1 Method

The device operated at 41 kHz with 4ms bursts of sound for the duration of the flight. As intended, time domain data was captured and saved to the system's Raspberry Pi. All wind calculations were conducted with MATLAB post-flight with a cross-correlation algorithm that included a physical model, previous estimates, lowpass interpolation (4x), and FFT-CC; this method followed from recommendations found in bell jar testing during the ground test. When applicable, a transform matrix was applied to switch from anemometer coordinates to gondola coordinates. The physical layout and coordinate systems are shown in Figure 20.

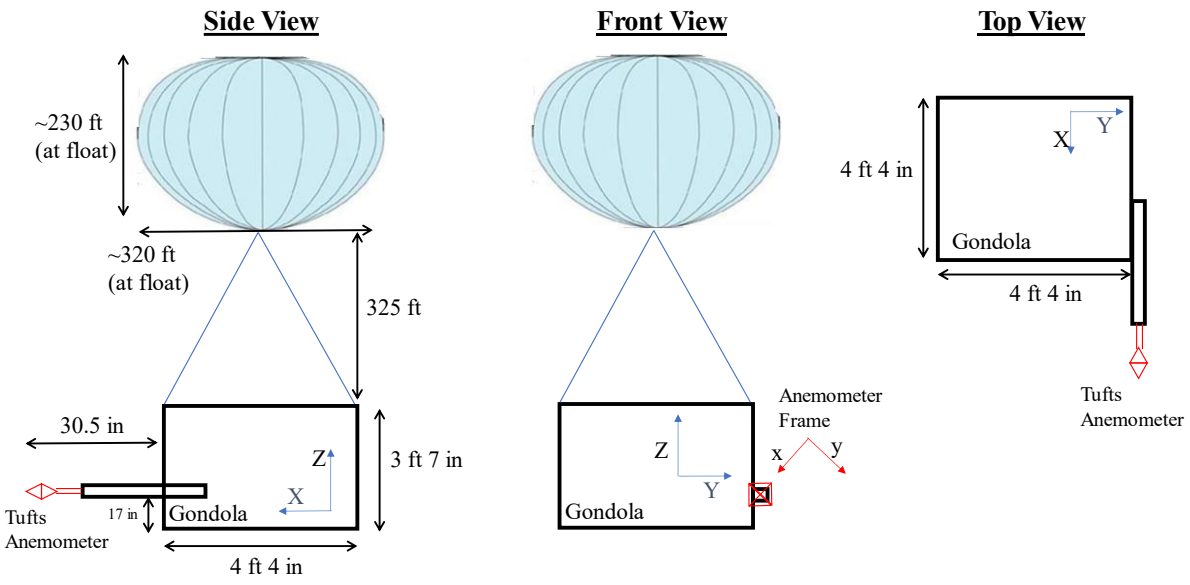


Figure 20, Diagram of anemometer on gondola showing the difference between the anemometer and the gondola coordinate systems.

To verify the accuracy of wind velocity data, a comparison to ascent velocity was made. The onboard GPS unit provided altitude data which was converted to an ascent rate using a numerical derivative. This was compared to vertical wind (Z in gondola coordinates) from the anemometer. Assuming minimal environmental vertical wind (ground coordinates), the ascent velocity from GPS and vertical wind velocity from the anemometer should be comparable. Since a speed of sound can be measured from the anemometer, a comparison of speed of sound from the anemometer to speed of sound based on air temperature (ideal gas) was made. A final benchmark for the anemometer was an investigation into the spectra of vertical wind at float and a comparison to the spectra from vertical GPS velocity. The dominating altitude changes are the result of

buoyancy forces visible in a buoyancy frequency. Ideally, this frequency should appear in both the GPS data and the sonic anemometer data.

4.2 Result

Launching from Fort Sumner, New Mexico (34° 52.73’ N, 105° 25.03’ W) on August 23, 2022, SPARROW-3 (NASA) rose to a float altitude of 38 km above sea level where it stayed for 3 hours. Fig Y describes the flight trajectory and Fig X shows a photo taken from the gondola at float. The total flight lasted 6 h and touched down 127 km away at 34° 52.20’ N, 105° 19.20’ W. The balloon experienced pressures from 100 mbar down to 3.9 mbar and temperatures ranging from 20 C down to -60 C.

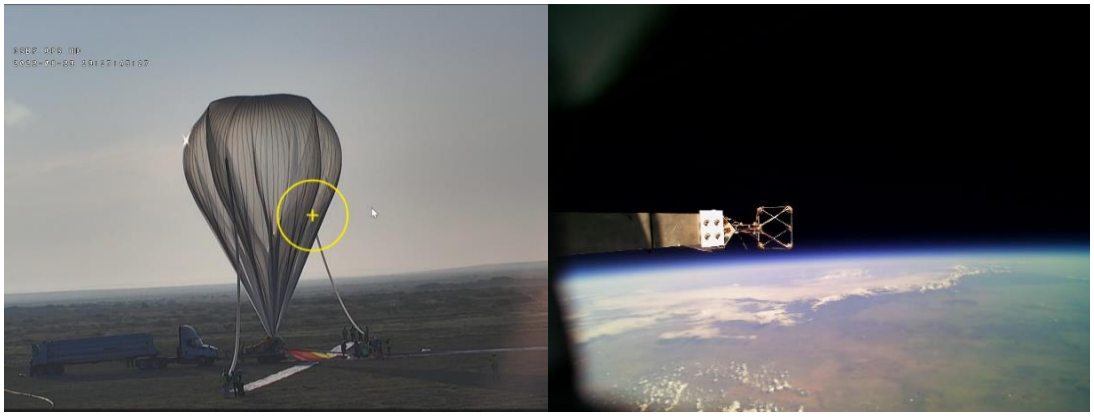


Figure 21, Preflight (left) and float (right) photos from August 23, 2022 balloon flight.

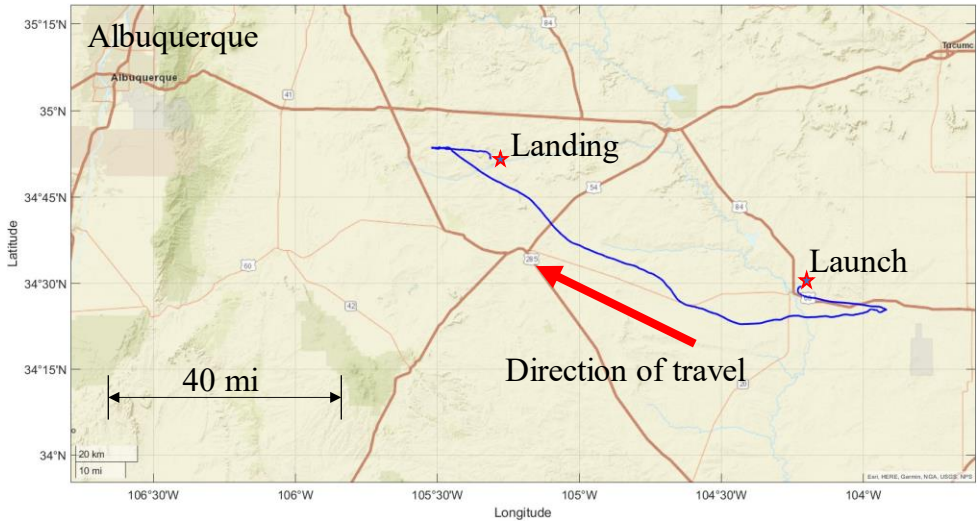


Figure 22, Map of flight trajectory.

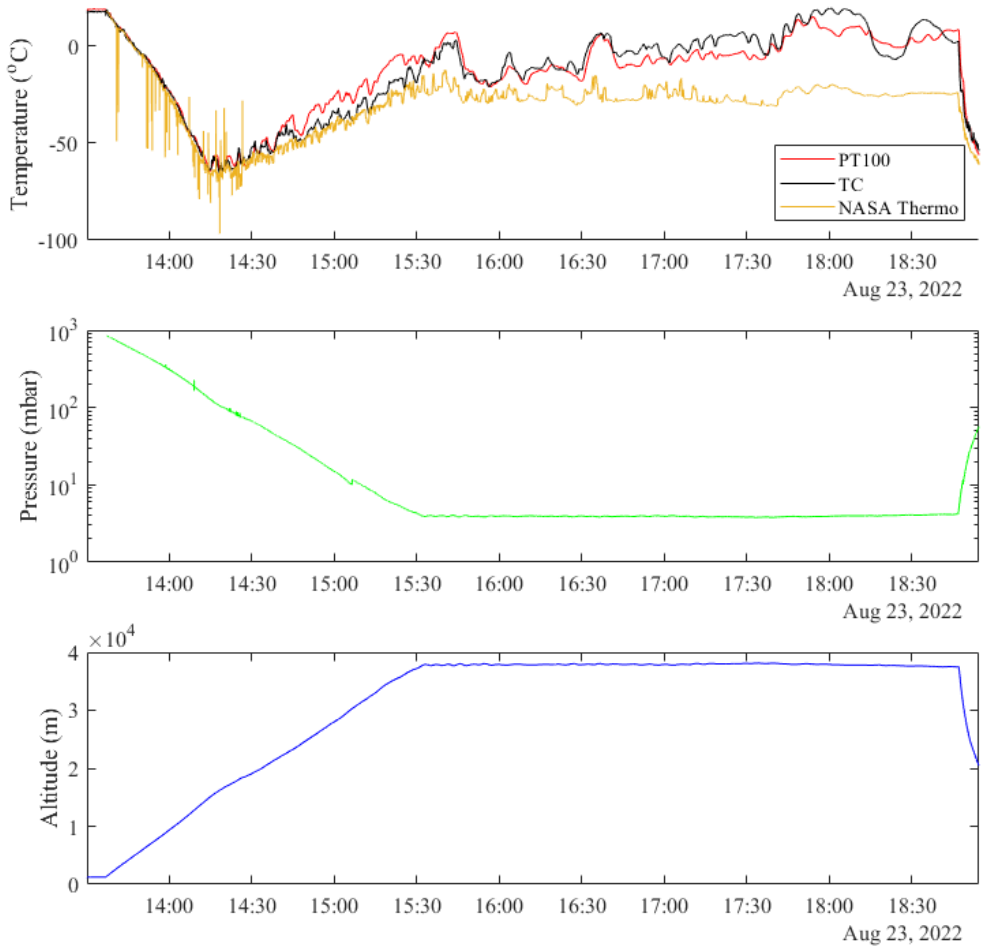


Figure 23, Environmental data (temperature, pressure, and altitude) during flight. Note the discrepancy between NASA's thermometer (yellow) and the anemometer's (PT100 and TC) during float. This was attributed to solar radiation on the Tufts sensors.

The IMU data captured had a clear resolution difference between the VN100 and the BSO080, with the VN100 having higher resolution in all metrics (i.e. accelerometer, gyroscope, and magnetometer). Figure 24 and Figure 25 show graphs of IMU data just after launch to before float and during float. The IMU data was not utilized in any further comparisons or studies. However, comparing some of the IMU metrics with anemometer

wind velocities could be done. For example, during ascent the gondola rotated about the z axis (see Figure 24) which could be compared to x or y axis wind data from the anemometer. In addition, in future work the IMU frame of reference can be used to convert the gondola coordinates into a world frame.

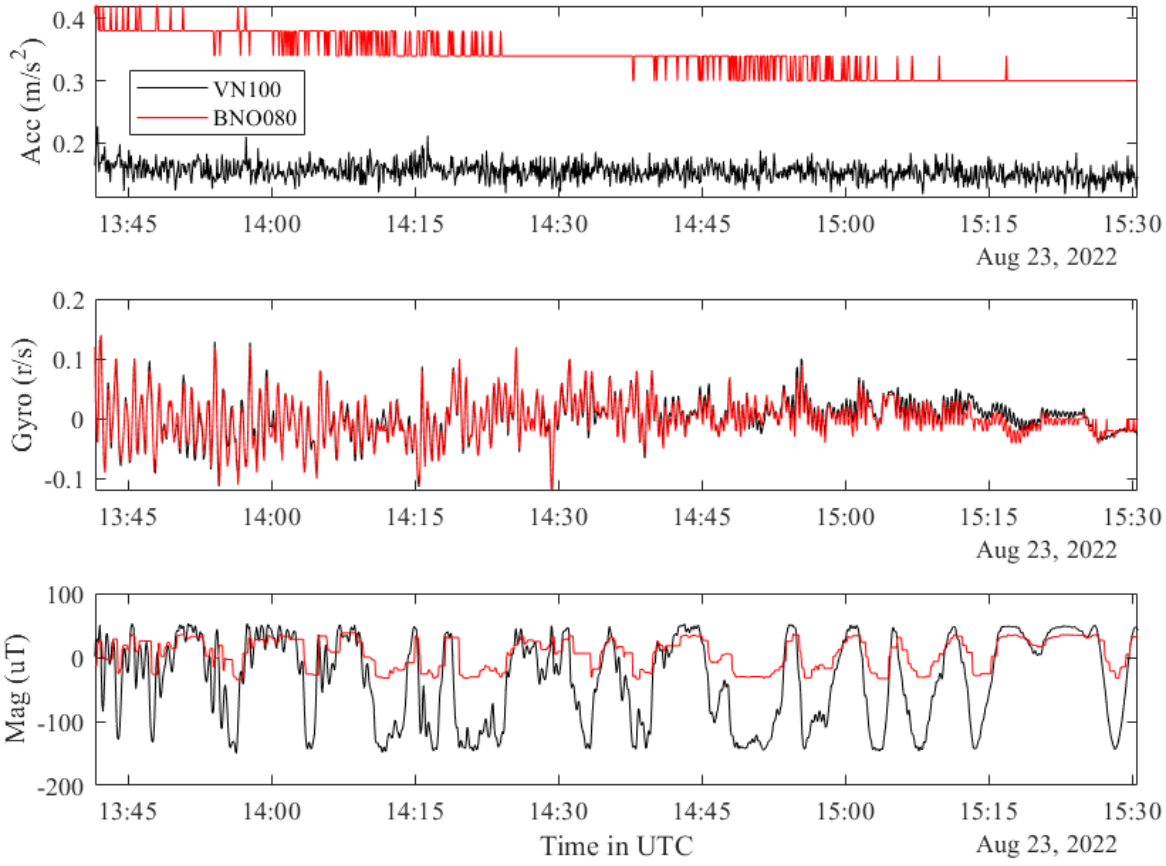


Figure 24, Graphs of IMU data from balloon ascent. Acceleration was compared on the x axis, Gyro on the z, and Mag on the x.

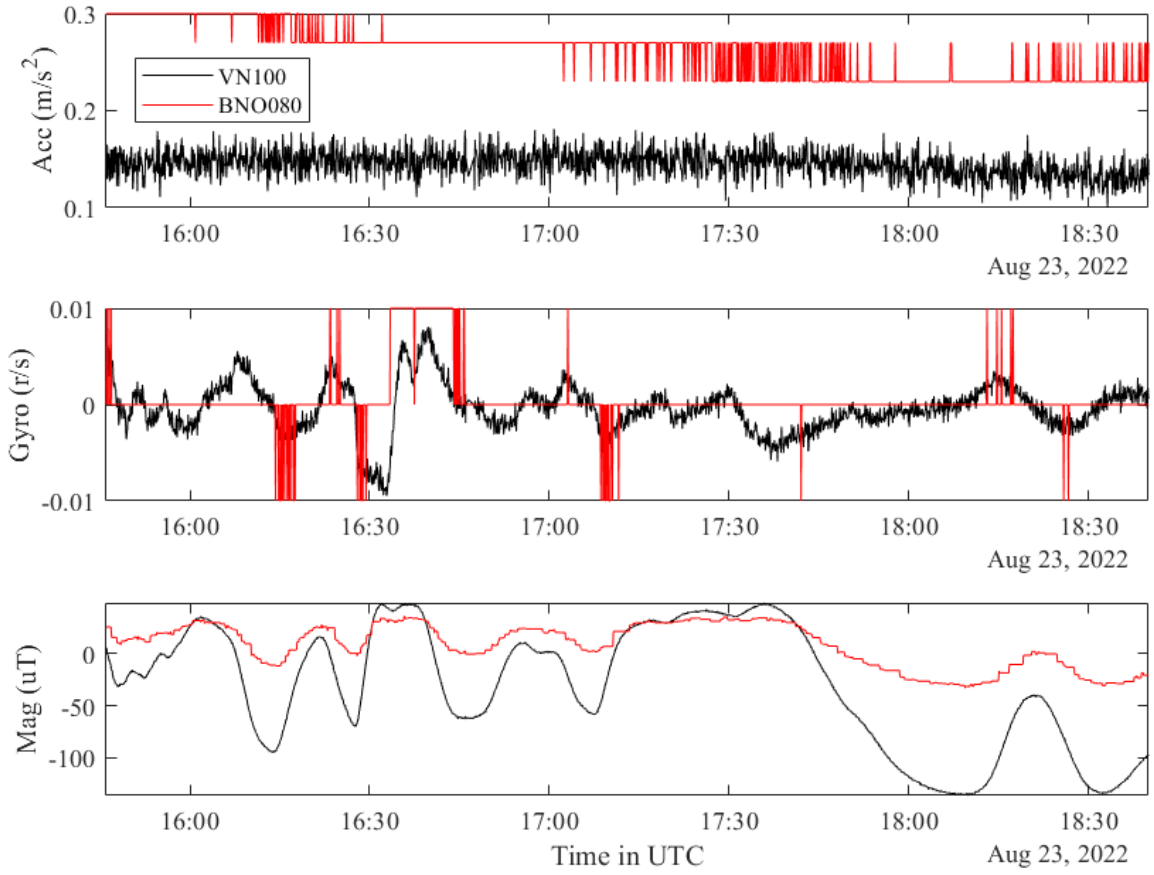


Figure 25, Graphs of IMU data from balloon float. Acceleration was compared on the x axis, Gyro on the z, and Mag on the x. Note the large quantization of the BNO080 sensor.

The comparison of vertical velocity from the anemometer and the GPS was very favorable. The anemometer indicated a velocity of ~6 m/s and the GPS ascent velocity at a similar ~6 m/s (Figure 26). Both the investigations into speed of sound and buoyancy frequency resulted in expected outcomes. The speed of sound measured from the anemometer followed closely with that calculated from temperature. There was a subtle separation of speed of sound as temperature dropped, this could be due to the temperature sensor being open to solar radiation possibly heating it more than expected. As the

pressure dropped the radiative heating factor may have become more prevalent. When comparing the spectra at float from the anemometer to the GPS, there is a similar hump around 2-3 mHz, which is in line with expectations of the buoyance frequency. The results of these tests indicated successful operation of the sonic anemometer during ascent up to an altitude of 15 km with vertical flow velocity matching closely with ascent rate and speed of sound measurements matching well with expectations. The instrument continued to perform at float (altitude of 38 km), however it was more difficult to determine whether the measured wind values were accurate as we had no expectations or other sonsors to compare against.

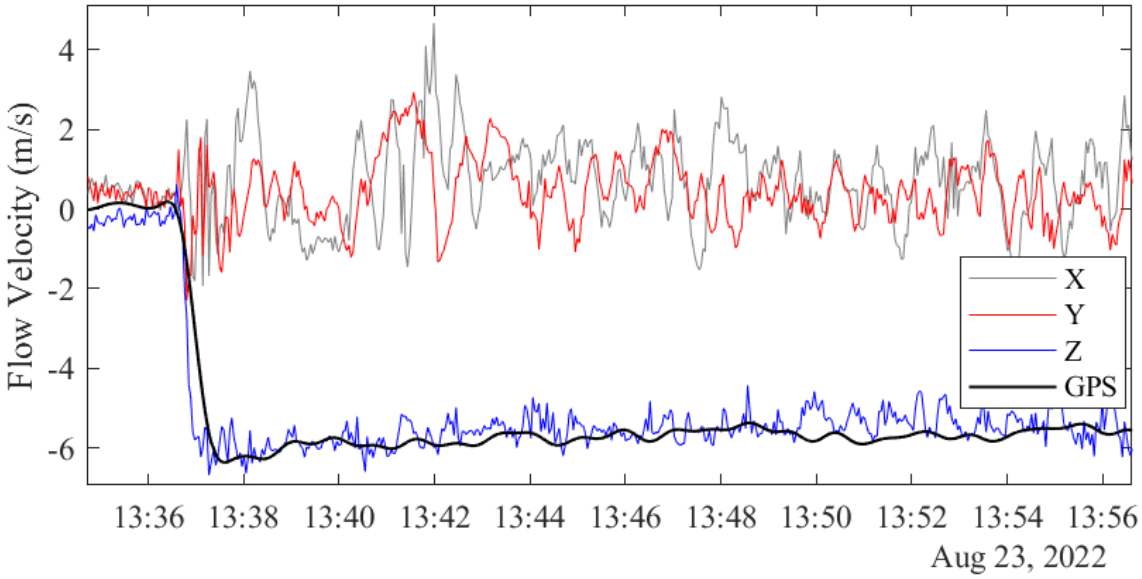


Figure 26, Comparison of gondola velocity found with GPS measurements to anemometer wind velocities. Note the agreement between GPS and the anemometer.

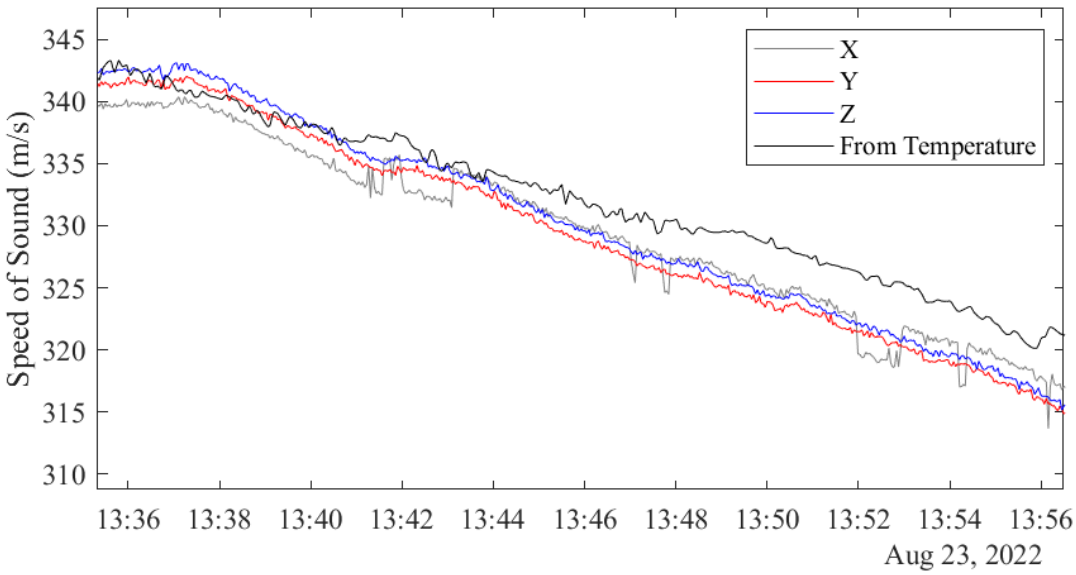


Figure 27, Comparison of speed of sound from temperature and from each anemometer axis during balloon ascent. Note that the X axis' (blue) unit shifts are likely from peak ambiguity in the correlation data, see Figure 12.

The anemometer did experience some limitations. It was fully operational above -40 ºC, however, below that temperature produced signal amplitudes larger than the hardware could handle (see Appendix Figure 39). As the air heated up with ascension into the stratosphere, this overloading of the hardware disappeared. Another fault occurred above 24 km where one of the axes (Figure 40) had a substantially lower signal to noise ratio (SNR). This led to less than expected wind velocities from this singular axis at float (see Figure 30). Excluding an axis at float meant having to make some assumptions before a comparison of GPS vertical velocity could be compared with the anemometer values. Since the anemometer axes were mounted 45 deg off with respect to the gondola's Z-axis and one of the anemometer axes that contributes to the gondola Z-axis was faulty a float, an assumption of small X and Y wind in the gondola's coordinate system was made. If this is true then a projection of the GPS velocity onto the

anemometer's usable axis can be made (Figure 28) and a spectra taken (Figure 29). However, without any other instrument to compare to, the small wind approximation could not be independently verified.

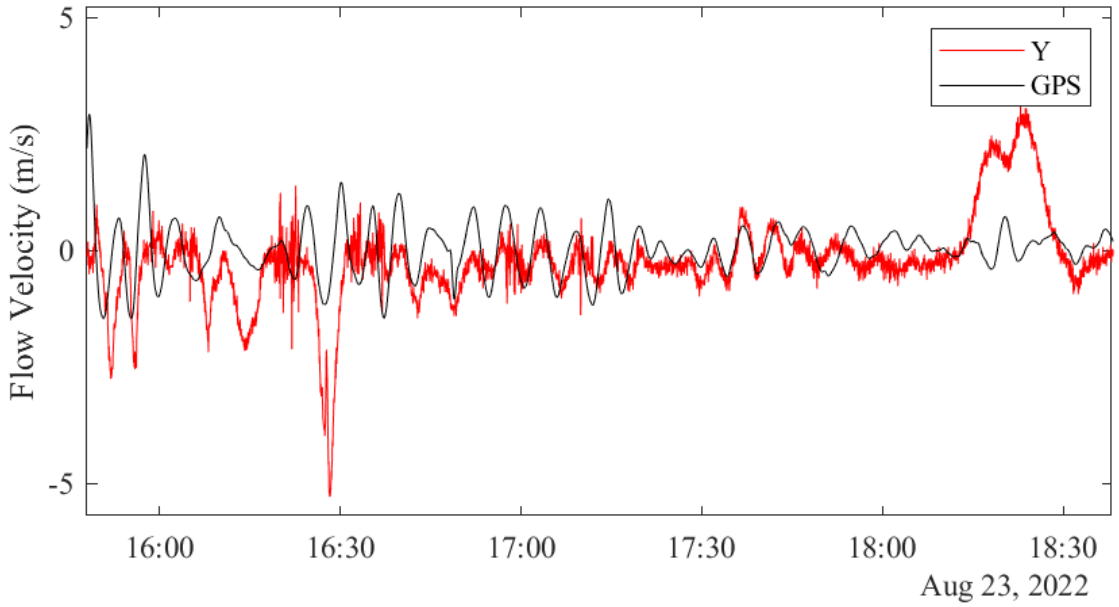


Figure 28, Vertical GPS velocity projected onto working anemometer axis (Y) at float.

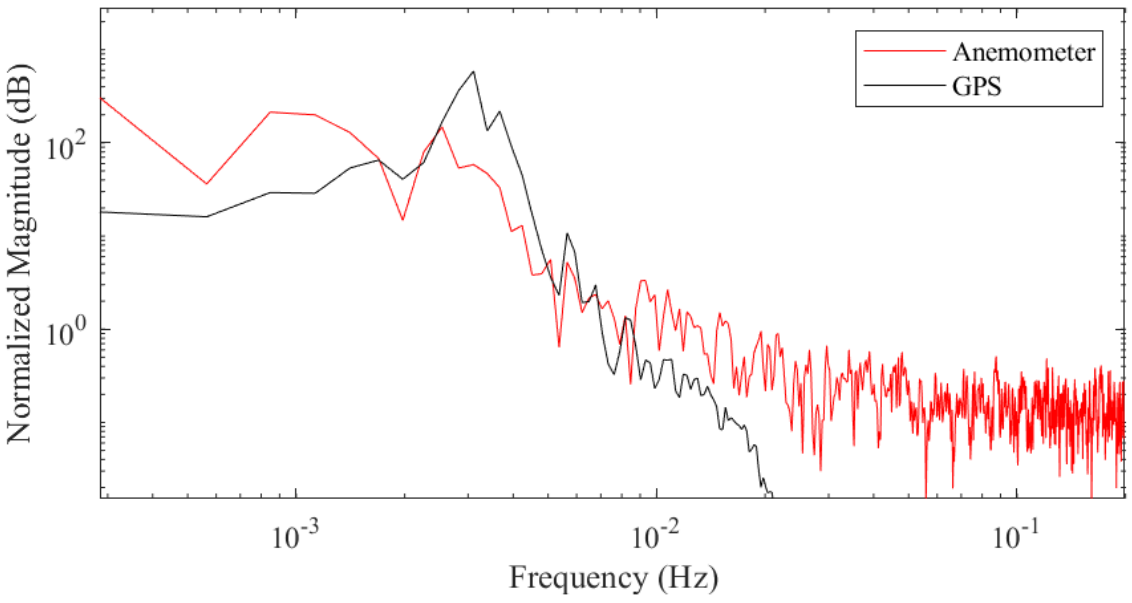


Figure 29, Spectrum of working anemometer axis and projected GPS velocity at float.

When looking at the wind data from float there appears to be no major abnormalities. The relative wind of the anemometer rarely peaked above 5 m/s which was expected for the altitude. There were some prominent features, both at the beginning of float (15:30 – 16:30 UTC) and toward the end of float (18:15 – 18:30 UTC). These features were larger than expected from the buoyancy frequency movement or subtle rotations of the gondola. They could be gusts of wind that changed relative to the gondola's position and could be important in gravity wave or solar phenomena research. However, there was no way to independently versify the accuracy, since this was the first sonic instrument to operate at this altitude.

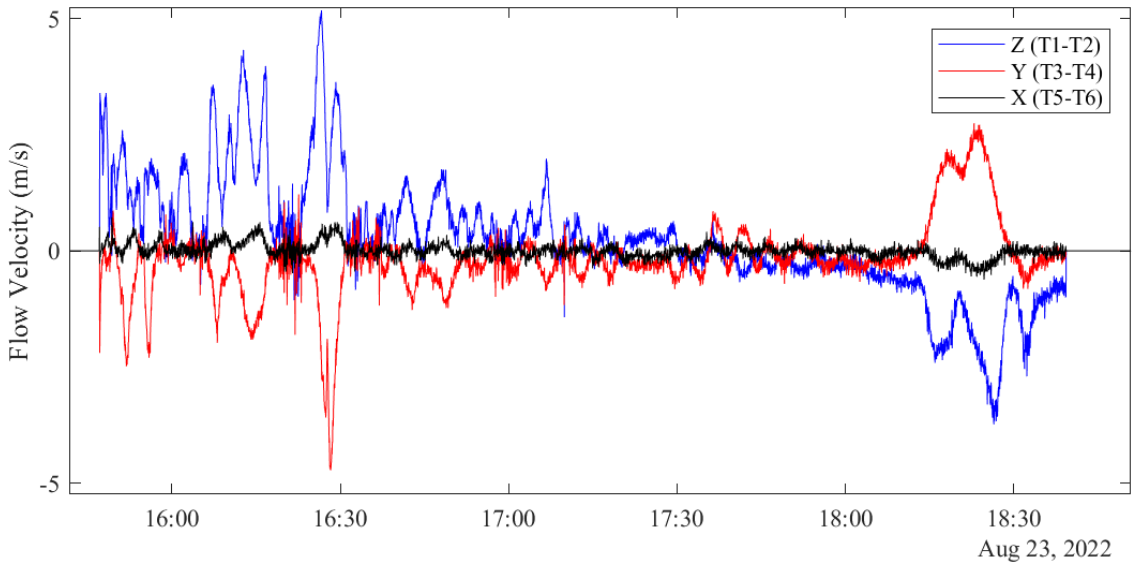


Figure 30, Graph of wind velocity at float from the sonic anemometer. Note that this is in the anemometer's coordinate system and that the X axis (black) is reporting low wind values likely from a low SNR.

4.3 Flight Discussion

Overall, the system including the 3-axis sonic anemometer and all other sensors operated successfully throughout the 6 hour flight and recorded all major features including ascent, float, and decent withing expectations. There were some findings that could be addressed in future revisions. The most pressing concerns were the one noisy axis at float, and the large signal strength at low temperatures (<-40 ºC). The higher noise levels on the x-axis were observed during ground testing before flight, but the cause was not isolated and the issue could not be corrected in time for delivery. In future iterations, new boards, transducers, and head will be used and this issue will be isolated and repaired. The difficulty with operation at temperatures below -40 ºC could have been caused by a shift in transducer resonant frequency resulting in a larger than intended signal (we operated off resonance at 20 ºC). This issue would be best addressed by identifying a broadband transducer and operating over a wider and to avoid resonant effects. A broadband signal would also reduce peak ambiguity with the FFT-CC algorithm. Other updates to consider in the future include upgrades to the digital hardware to increase sample length, increase time domain signal storage speed, and reduce noise within the digital electronics to improve accuracy and resolution.

5 Conclusion and Future Work

An STM32 L476RG based development board successfully replaced analog phase measurement chips in a high altitude (low pressure) sonic anemometer. Its implementation involved the design and creation of a daughterboard that acted as an adapter to existing hardware, as well as ample program creation to facilitate the capture and processing of time domain signal data. The anemometer took 3-axis velocity data, enough to fully characterize air velocity in three-dimensional space, every 2.25 seconds (0.44 Hz) with a possible resolution of 0.05 m/s (using FFT-CC and interpolation). Internal noise, primarily from digital hardware limitations in the ADC, clock, etc. produced a resolution limitation of 0.1 m/s in terms of velocity standard deviation. The new hardware had little effect on the system's power consumption (<5% out of 5.3 W total) and performed well for ground testing and a subsequent balloon flight test. Firmware and programs were written to operate the new microcontroller, most notably were the data capturing firmware of the STM32 in C# and the post processing method to determine wind velocity in MATLAB. The creation of the firmware emphasized the need for DMA and high data throughput, while processing of the time domain data yielded recommendations for optimal correlation techniques.

Successful ground testing of the system cleared the way for flight aboard SPARROW-3, a NASA high altitude zero pressure balloon. A bell jar was used to emulate a low-pressure environment for ground testing of the anemometer. This allowed for optimization of post processing methods and validation that the system could operate before flight. This resulted in successful tests down to 10 mbar and 0 ºC. The

anemometer was subsequently sent to Fort Sumner, New Mexico where it flew on SPARROW-3 with launch on August 23, 2022. The flight lasted 6 hours from launch to touchdown and reached an altitude of 38 km. The anemometer functioned throughout the flight with only minor issues, namely an oversaturation of signal below -40 ºC and signal to noise problems with one of the axes at float. Resulting wind velocities indicated the balloon's ascent velocity during travel upward to float and the buoyancy frequency of the balloon at float. Possible gusts on the order of 5 m/s relative wind were also observed during float.

5.1 Future Work

While the system operated successfully, points of improvement became apparent during the study. The STM32 L476RG operated at its functional limit with respect to RAM and data throughput. A switch to higher performance hardware could aid in a future goal of a 10 Hz sampling rate, a frequency desired for studying turbulent boundary layer eddies on Mars and high-altitude balloon navigation. The mechanical hardware also has room for improvement. A redesign of the head structure and inclusion of wider band transducers could positively impact resolution and low-pressure operation. For example, removing select octahedral edges (Figure 32) could help eliminate structural vibration transmissions and wider band transducers would reduce peak ambiguity in ToF acquisition (Figure 14). There was also data from IMU units that was not explored in this thesis. Comparing IMU data to anemometer data could be significant in modeling balloon physics for platform stability or converting wind velocity into a global coordinate frame.

The velocity data has yet to be analyzed for features related to gravitational waves, solar events, etc. and could be another avenue for future research.

References

[1] E. W. Barrett and V. E. Suomi, “PRELIMINARY REPORT ON TEMPERATURE MEASUREMENT BY SONIC MEANS,” *J. Atmospheric Sci.*, vol. 6, no. 4, pp. 273–276, Aug. 1949, doi: 10.1175/1520-0469(1949)006<0273:PROTMB>2.0.CO;2.

[2] C. R. E, “Acoustic Anemometer-Anemoscope,” *Electronics*, vol. 23, no. 1, p. 88, 1950.

[3] A. Wieser, F. Fiedler, and U. Corsmeier, “The Influence of the Sensor Design on Wind Measurements with Sonic Anemometer Systems,” *J. Atmospheric Ocean. Technol.*, vol. 18, no. 10, pp. 1585–1608, Oct. 2001, doi: 10.1175/1520-0426(2001)018<1585:TIOTSD>2.0.CO;2.

[4] V. E. Suomi, “Energy budget studies and development of the sonic anemometer for spectrum analysis.”.

[5] J. C. Kaimal and J. A. Businger, “A Continuous Wave Sonic Anemometer-Thermometer,” *J. Appl. Meteorol. Climatol.*, vol. 2, no. 1, pp. 156–164, Feb. 1963, doi: 10.1175/1520-0450(1963)002<0156:ACWSAT>2.0.CO;2.

[6] T. Hanafusa, T. Fujitani, Y. Kobori, and Y. Mitsuta, “A new type sonic anemometer-thermometer for field operation.,” *Pap. Meteorol. Geophys.*, vol. 33, no. 1, pp. 1–19, 1982, doi: 10.2467/mripapers.33.1.

[7] L. Song, X. Hu, F. Wei, Z. Yan, Q. Xu, and C. Tu, “Development of a Balloon-Borne Acoustic Anemometer to Measure Winds for SENSOR Campaign,” *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022, doi: 10.1109/TIM.2022.3189629.

[8] B. A. Maruca *et al.*, “Overview of and first observations from the TILDAE High-Altitude Balloon Mission,” *Atmospheric Meas. Tech.*, vol. 10, no. 4, pp. 1595–1607, Apr. 2017, doi: 10.5194/amt-10-1595-2017.

[9] R. D. White and D. Banfield, “Sonic anemometry on a high altitude balloon,” *J. Acoust. Soc. Am.*, vol. 151, no. 4, pp. A185–A185, Apr. 2022, doi: 10.1121/10.0011044.

[10] White, R. D., Neeson, I., Schmid, E. S., Merrison, J., Iversen, J. J., & Banfield, D., “Flow Testing of a Sonic Anemometer for the Martian Environment,” *AIAA SciTech Forum*, doi: 10.2514/6.2020-0712.

[11] D. Banfield, D. W. Schindel, S. Tarr, and R. W. Dissly, “A Martian acoustic anemometer,” *J. Acoust. Soc. Am.*, vol. 140, no. 2, pp. 1420–1428, Aug. 2016, doi: 10.1121/1.4960737.

[12] G. A. McBean, “Instrument Requirements for Eddy Correlation Measurements,” *J. Appl. Meteorol. Climatol.*, vol. 11, no. 7, pp. 1078–1084, Oct. 1972, doi: 10.1175/1520-0450(1972)011<1078:IRFECM>2.0.CO;2.

[13] K. Garg, “Wind Based Navigation for Zero-Pressure Stratospheric Balloons Using Reinforcement learning,” *Acta Astron.*, 2019, Accessed: Aug. 16, 2022. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-76542

[14] M. G. Bellemare *et al.*, “Autonomous navigation of stratospheric balloons using reinforcement learning,” *Nature*, vol. 588, no. 7836, Art. no. 7836, Dec. 2020, doi: 10.1038/s41586-020-2939-8.

[15] E. van Wynsberghe and A. Turak, “Station-keeping of a high-altitude balloon with electric propulsion and wireless power transmission: A concept study,” *Acta Astronaut.*, vol. 128, pp. 616–627, Nov. 2016, doi: 10.1016/j.actaastro.2016.08.017.

[16] S. S. Ramesh, J. Ma, K.-M. Lim, H. P. Lee, and B. C. Khoo, “Numerical evaluation of station-keeping strategies for stratospheric balloons,” *Aerosp. Sci. Technol.*, vol. 80, pp. 288–300, Sep. 2018, doi: 10.1016/j.ast.2018.07.010.
[17] T. J. Dunkerton, “Inertia–Gravity Waves in the Stratosphere,” *J. Atmospheric Sci.*, vol. 41, no. 23, pp. 3396–3404, Dec. 1984, doi: 10.1175/1520-0469(1984)041<3396:IWITS>2.0.CO;2.
[18] Y. He, Z. Sheng, and M. He, “Spectral Analysis of Gravity Waves from Near Space High-Resolution Balloon Data in Northwest China,” *Atmosphere*, vol. 11, no. 2, Art. no. 2, Feb. 2020, doi: 10.3390/atmos11020133.
[19] R. Fovell, D. Durran, and J. R. Holton, “Numerical Simulations of Convectively Generated Stratospheric Gravity Waves,” *J. Atmospheric Sci.*, vol. 49, no. 16, pp. 1427–1442, Aug. 1992, doi: 10.1175/1520-0469(1992)049<1427:NSOCGS>2.0.CO;2.
[20] L. J. Gray *et al.*, “Solar Influences on Climate,” *Rev. Geophys.*, vol. 48, no. 4, 2010, doi: 10.1029/2009RG000282.
[21] “Loon,” *X, the moonshot factory*. https://x.company/projects/loon/ (accessed Apr. 12, 2023).
[22] E. Kassarian, F. Sanfedino, D. Alazard, H. Evain, and J. Montel, “Modeling and stability of balloon-borne gondolas with coupled pendulum-torsion dynamics,” *Aerosp. Sci. Technol.*, vol. 112, p. 106607, May 2021, doi: 10.1016/j.ast.2021.106607.
[23] R. Fesen and Y. Brown, “A method for establishing a long duration, stratospheric platform for astronomical research,” *Exp. Astron.*, vol. 39, no. 3, pp. 475–493, Oct. 2015, doi: 10.1007/s10686-015-9459-9.
[24] E. F. Young *et al.*, “Sub-arcsecond performance of the ST5000 star tracker on a balloon-borne platform,” in *2012 IEEE Aerospace Conference*, Mar. 2012, pp. 1–7. doi: 10.1109/AERO.2012.6187179.
[25] T. Abe, T. Imamura, N. Izutsu, and N. Yajima, *Scientific Ballooning*. New York, NY: Springer New York, 2009. doi: 10.1007/978-0-387-09727-5.
[26] R. F. Woodman and A. Guillen, “Radar Observations of Winds and Turbulence in the Stratosphere and Mesosphere,” *J. Atmospheric Sci.*, vol. 31, no. 2, pp. 493–505, Mar. 1974, doi: 10.1175/1520-0469(1974)031<0493:ROOWAT>2.0.CO;2.
[27] E. Kassarian, F. Sanfedino, D. Alazard, J. Montel, and C.-A. Chevrier, “Robust line-of-sight pointing control on-board a stratospheric balloon-borne platform.” arXiv, Dec. 20, 2021. Accessed: Aug. 17, 2022. [Online]. Available: http://arxiv.org/abs/2112.10458
[28] R. A. Vincent and M. Joan Alexander, “Gravity waves in the tropical lower stratosphere: An observational study of seasonal and interannual variability,” *J. Geophys. Res. Atmospheres*, vol. 105, no. D14, pp. 17971–17982, 2000, doi: 10.1029/2000JD900196.
[29] A. Petculescu, B. Hall, R. Fraenzle, S. Phillips, and R. M. Lueptow, “A prototype acoustic gas sensor based on attenuation,” *J. Acoust. Soc. Am.*, vol. 120, no. 4, pp. 1779–1782, Oct. 2006, doi: 10.1121/1.2336758.
[30] T. Leighton and A. Petculescu, “Sounds in space: the potential uses for acoustics in the exploration of other worlds,” *Hydroacoustics*, vol. Vol. 11, pp. 225–239, 2008.
[31] A. Petculescu and R. M. Lueptow, “Atmospheric acoustics of Titan, Mars, Venus, and Earth,” *Icarus*, vol. 186, no. 2, pp. 413–419, Feb. 2007, doi: 10.1016/j.icarus.2006.09.014.
[32] C. A. Roseman, J. L. Pointer, B. Argrow, and D. A. Lawrence, “Experimental and Numerical Calibration of Hotwire Anemometers for the Study of Stratospheric Turbulence,” in *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2021-0828.

[33] C. A. Roseman and B. M. Argrow, “Low-Speed DSMC Simulations of Hotwire Anemometers at High-Altitude Conditions,” *Fluids*, vol. 6, no. 1, Art. no. 1, Jan. 2021, doi: 10.3390/fluids6010020.
[34] D. Alfonso-Corcuera, M. Ogueta-Gutiérrez, A. Fernández-Soler, D. González-Bárcena, and S. Pindado, “Measuring Relative Wind Speeds in Stratospheric Balloons with Cup Anemometers: The TASEC-Lab Mission,” *Sensors*, vol. 22, no. 15, Art. no. 15, Jan. 2022, doi: 10.3390/s22155575.
[35] Á. Ramos-Cenzano, E. López-Núñez, D. Alfonso-Corcuera, M. Ogueta-Gutiérrez, and S. Pindado, “On cup anemometer performance at high altitude above ground,” *Flow Meas. Instrum.*, vol. 79, p. 101956, Jun. 2021, doi: 10.1016/j.flowmeasinst.2021.101956.
[36] O. Kliebisch and P. Mahnke, “Real-time laser Doppler anemometry for optical air data applications in low aerosol environments,” *Rev. Sci. Instrum.*, vol. 91, no. 9, p. 095106, Sep. 2020, doi: 10.1063/5.0014389.
[37] M. Güçyetmez, S. Keser, and Ş. E. Hayber, “Wind speed measurement with a low-cost polymer optical fiber anemometer based on Fresnel reflection,” *Sens. Actuators Phys.*, vol. 339, p. 113509, Jun. 2022, doi: 10.1016/j.sna.2022.113509.
[38] M. Harris, G. N. Pearson, K. D. Ridley, C. J. Karlsson, F. Å. A. Olsson, and D. Letalick, “Single-particle laser Doppler anemometry at 1.55 µm,” *Appl. Opt.*, vol. 40, no. 6, pp. 969–973, Feb. 2001, doi: 10.1364/AO.40.000969.
[39] F. Durst, A. Melling, and J. H. Whitelaw, “Principles and practice of laser-Doppler anemometry,” *NASA STIRecon Tech. Rep. A*, vol. 76, p. 47019, Jan. 1976.
[40] “Wind Energy WindCube,” *Vaisala*. https://www.vaisala.com/en/wind-lidars/wind-energy/windcube (accessed Aug. 18, 2022).
[41] S. M. Spuler, D. Richter, M. P. Spowart, and K. Rieken, “Optical fiber-based laser remote sensor for airborne measurement of wind velocity and turbulence,” *Appl. Opt.*, vol. 50, no. 6, pp. 842–851, Feb. 2011, doi: 10.1364/AO.50.000842.
[42] S. Pindado, J. Cubas, and F. Sorribes-Palmer, “On the harmonic analysis of cup anemometer rotation speed: A principle to monitor performance and maintenance status of rotating meteorological sensors,” *Measurement*, vol. 73, pp. 401–418, Sep. 2015, doi: 10.1016/j.measurement.2015.05.032.
[43] X. Ren, Y. Liu, Z. Cai, and Y. Zhang, “Observations of Dynamic Turbulence in the Lower Stratosphere over Inner Mongolia Using a High-resolution Balloon Sensor Constant Temperature Anemometer,” *Adv. Atmospheric Sci.*, vol. 39, no. 3, pp. 519–528, Mar. 2022, doi: 10.1007/s00376-021-1233-5.
[44] X. Bai, H. Jiang, and Y. Hu, “Research and Application of Airborne Optical Atmospheric Data System,” in *2021 IEEE 15th International Conference on Electronic Measurement & Instruments (ICEMI)*, Oct. 2021, pp. 259–263. doi: 10.1109/ICEMI52946.2021.9679546.
[45] P. Feneyrou *et al.*, “Frequency-modulated multifunction lidar for anemometry, range finding, and velocimetry–1. Theory and signal processing,” *Appl. Opt.*, vol. 56, no. 35, pp. 9663–9675, Dec. 2017, doi: 10.1364/AO.56.009663.
[46] B. Augere, B. Besson, D. Fleury, D. Goular, C. Planchat, and M. Valla, “1.5 μ m lidar anemometer for true air speed, angle of sideslip, and angle of attack measurements on-board Piaggio P180 aircraft,” *Meas. Sci. Technol.*, vol. 27, no. 5, p. 054002, May 2016, doi: 10.1088/0957-0233/27/5/054002.

[47] H. Dong and Y. Jun, “High Accuracy Time of Flight Measurement for Ultrasonic Anemometer Applications,” in *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, Sep. 2013, pp. 61–64. doi: 10.1109/IMCCC.2013.21.
[48] M. Azaria and D. Hertz, “Time delay estimation by generalized cross correlation methods,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 32, no. 2, pp. 280–285, Apr. 1984, doi: 10.1109/TASSP.1984.1164314.
[49] Ritu and S. K. Dhull, “A Comparison of Generalized Cross-Correlation Methods for Time Delay Estimation,” *IUP J. Telecommun.*, vol. 8, no. 4, pp. 31–46, Nov. 2016.
[50] Md. O. Khyam, S. S. Ge, X. Li, and M. R. Pickering, “Highly Accurate Time-of-Flight Measurement Technique Based on Phase-Correlation for Ultrasonic Ranging,” *IEEE Sens. J.*, vol. 17, no. 2, pp. 434–443, Jan. 2017, doi: 10.1109/JSEN.2016.2631244.
[51] I. Cespedes, Y. Huang, J. Ophir, and S. Spratt, “Methods for Estimation of Subsample Time Delays of Digitized Echo Signals,” *Ultrason. Imaging*, vol. 17, no. 2, pp. 142–171, Apr. 1995, doi: 10.1006/uimg.1995.1007.
[52] C. Knapp and G. Carter, “The generalized correlation method for estimation of time delay,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 24, no. 4, pp. 320–327, Aug. 1976, doi: 10.1109/TASSP.1976.1162830.
[53] “Cross-correlation - MATLAB xcorr.” https://www.mathworks.com/help/matlab/ref/xcorr.html (accessed Aug. 16, 2022).
[54] “Correlation.” https://www.keil.com/pack/doc/CMSIS/DSP/html/group__Corr.html (accessed Aug. 16, 2022).
[55] “Generalized cross-correlation - MATLAB gccphat.” https://www.mathworks.com/help/phased/ref/gccphat.html?searchHighlight=GCC-PHAT&s_tid=srchtitle_GCC-PHAT_1 (accessed Aug. 16, 2022).
[56] “Cubic spline data interpolation - MATLAB spline.” https://www.mathworks.com/help/matlab/ref/spline.html (accessed Aug. 15, 2022).
[57] “Interpolation — increase sample rate by integer factor - MATLAB interp.” https://www.mathworks.com/help/signal/ref/interp.html#bqijs12-1 (accessed Aug. 15, 2022).
[58] G. C. Carter, A. H. Nuttall, and P. G. Cable, “The smoothed coherence transform,” *Proc. IEEE*, vol. 61, no. 10, pp. 1497–1498, Oct. 1973, doi: 10.1109/PROC.1973.9300.
[59] L. Svilainis, “Review on Time Delay Estimate Subsample Interpolation in Frequency Domain,” *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 66, no. 11, pp. 1691–1698, Nov. 2019, doi: 10.1109/TUFFC.2019.2930661.
[60] “Motion Estimation in Ultrasound Images Using Time Domain Cross Correlation With Prior Estimates | IEEE Journals & Magazine | IEEE Xplore.” https://ieeexplore.ieee.org/abstract/document/1703750 (accessed Aug. 15, 2022).
[61] H.-G. Kang, M. Graczyk, and J. Skoglund, “On pre-filtering strategies for the GCC-PHAT algorithm,” in *2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, Sep. 2016, pp. 1–5. doi: 10.1109/IWAENC.2016.7602964.

Appendix

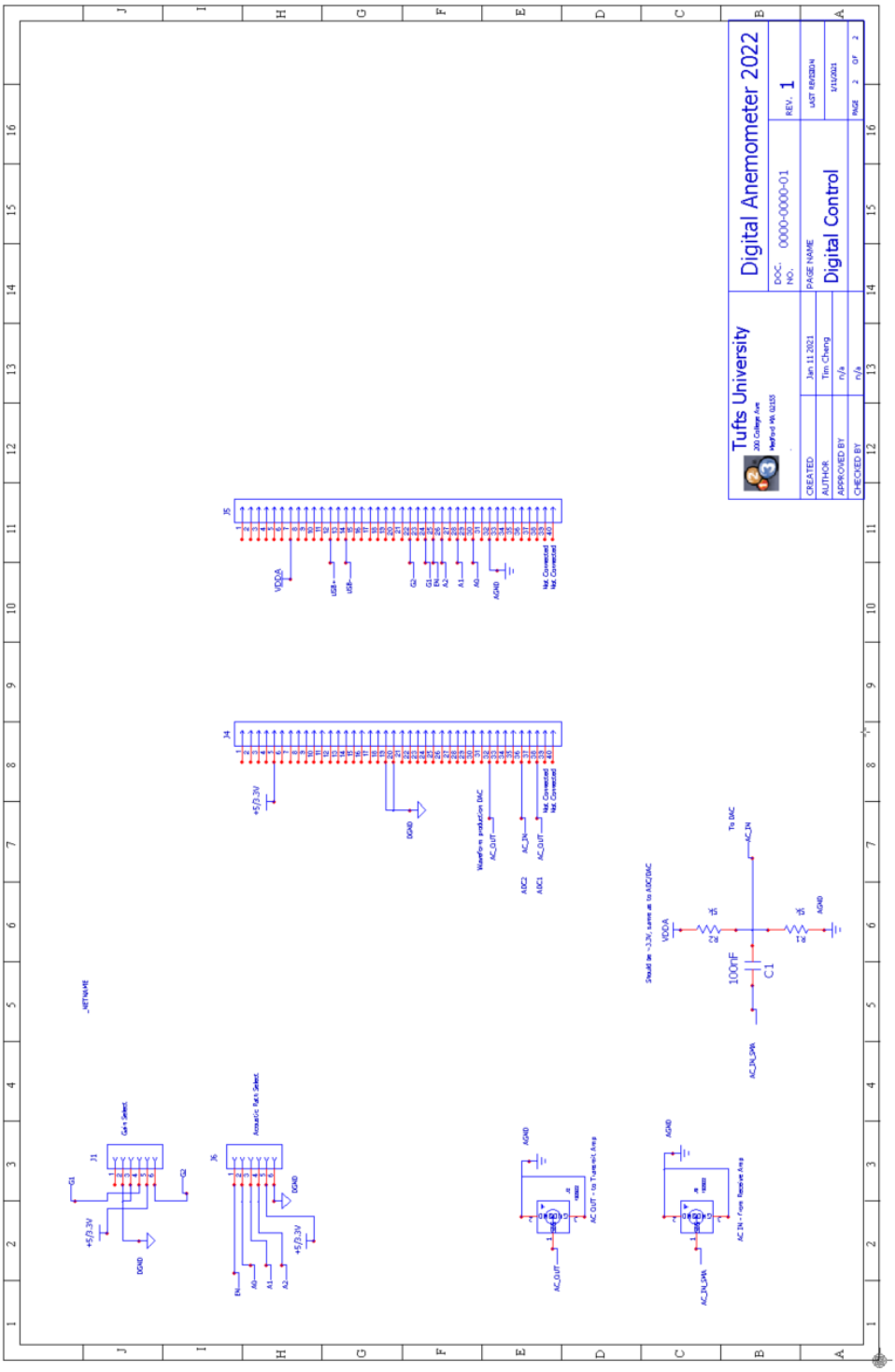


Figure 31, PCB123 daughterboard schematic.

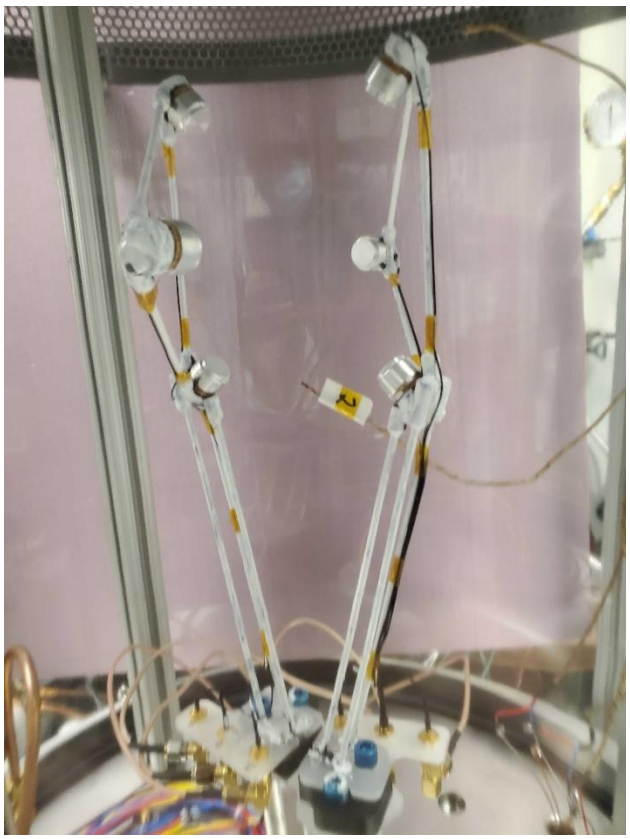


Figure 32, Bell jar testing of an octahedral without every edge.

Chirp Production Code in C

```
//Tukey Window
    double tukey (int i, int N, float alpha) {
      double anm12 = 0.5*alpha*(N-1);

      if( i <= anm12 ) {
        return 0.5*(1+cos(M_PI*(i/anm12 - 1)));
      }
      else if ( i < (N-1)*(1-0.5*alpha) ) {
        return 1;
      }
      else {
        return 0.5*(1+cos(M_PI*(i/anm12 - 2/alpha + 1)));
      }
    }

//Chirp Equation
      double Chirp(double w1, double w2, double A, double M, double time)
         {
         double res;
         res=A*cos(w1*time+(w2-w1)*time*time/(2*M));
         return res;
         }


  width = dur/0.002*8192;
  angularStart = 2*M_PI*f1;
  angularEnd = 2*M_PI*f2;
  timeratio = dur/width;


// Building Waveform

for(int x = 0; x < NS; x++)
{
    if(x<width)
    {
    waveform[x] = 2048 + Chirp(angularStart, angularEnd, amp, dur, x*timeratio)*tukey(x,width,alph);    //Multiplies Chirp and Tukey Window
    }
    else
    {
    waveform[x] = 2048;
    }

}
```

Figure 33, Code used to produce a linear chirp with a Tukey window.

Select Board Configuration Settings

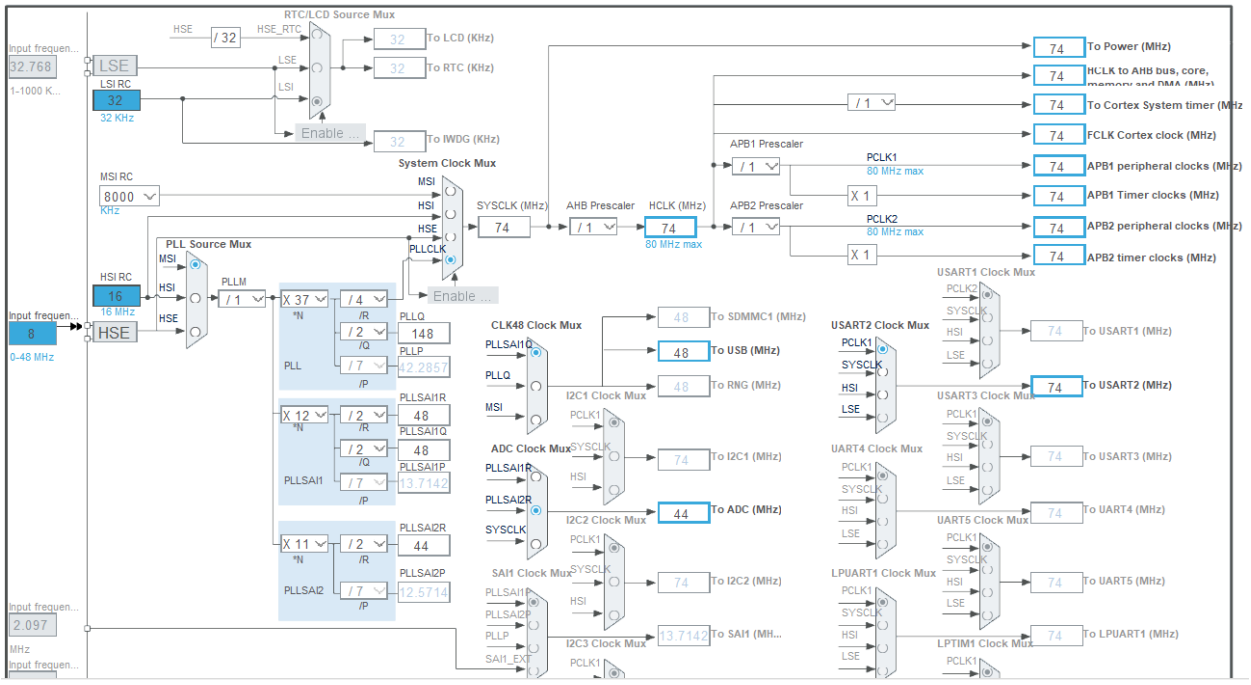
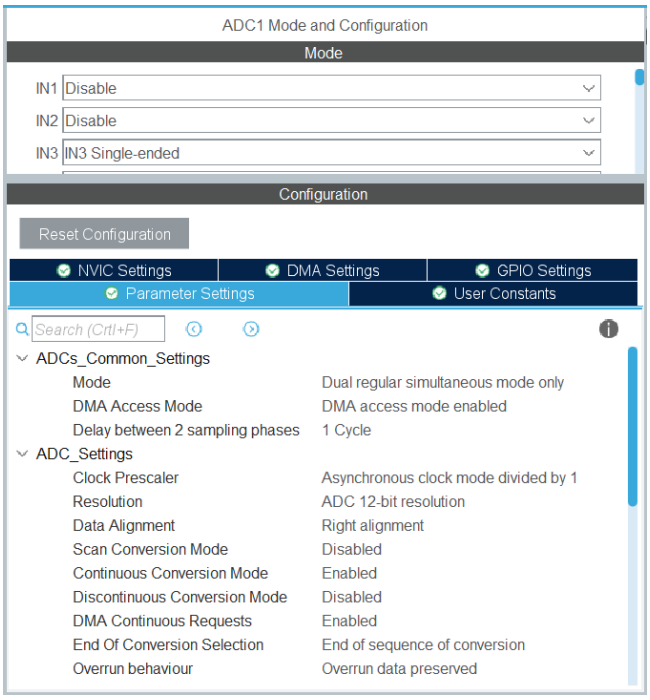
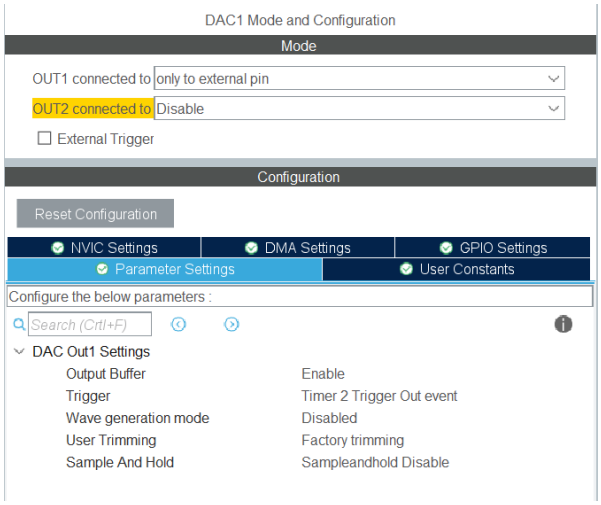


Figure 34, STM32 Clock Configuration.


ADC1 Mode and Configuration
Mode
IN1 Disable
IN2 Disable
IN3 IN3 Single-ended
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Search (Crtl+F)
ADCs_Common_Settings
Mode Dual regular simultaneous mode only
DMA Access Mode DMA access mode enabled
Delay between 2 sampling phases 1 Cycle
ADC_Settings
Clock Prescaler Asynchronous clock mode divided by 1
Resolution ADC 12-bit resolution
Data Alignment Right alignment
Scan Conversion Mode Disabled
Continuous Conversion Mode Enabled
Discontinuous Conversion Mode Disabled
DMA Continuous Requests Enabled
End Of Conversion Selection End of sequence of conversion
Overrun behaviour Overrun data preserved


DAC1 Mode and Configuration
Mode
OUT1 connected to only to external pin
OUT2 connected to Disable
External Trigger
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Crtl+F)
DAC Out1 Settings
Output Buffer Enable
Trigger Timer 2 Trigger Out event
Wave generation mode Disabled
User Trimming Factory trimming
Sample And Hold Sampleandhold Disable

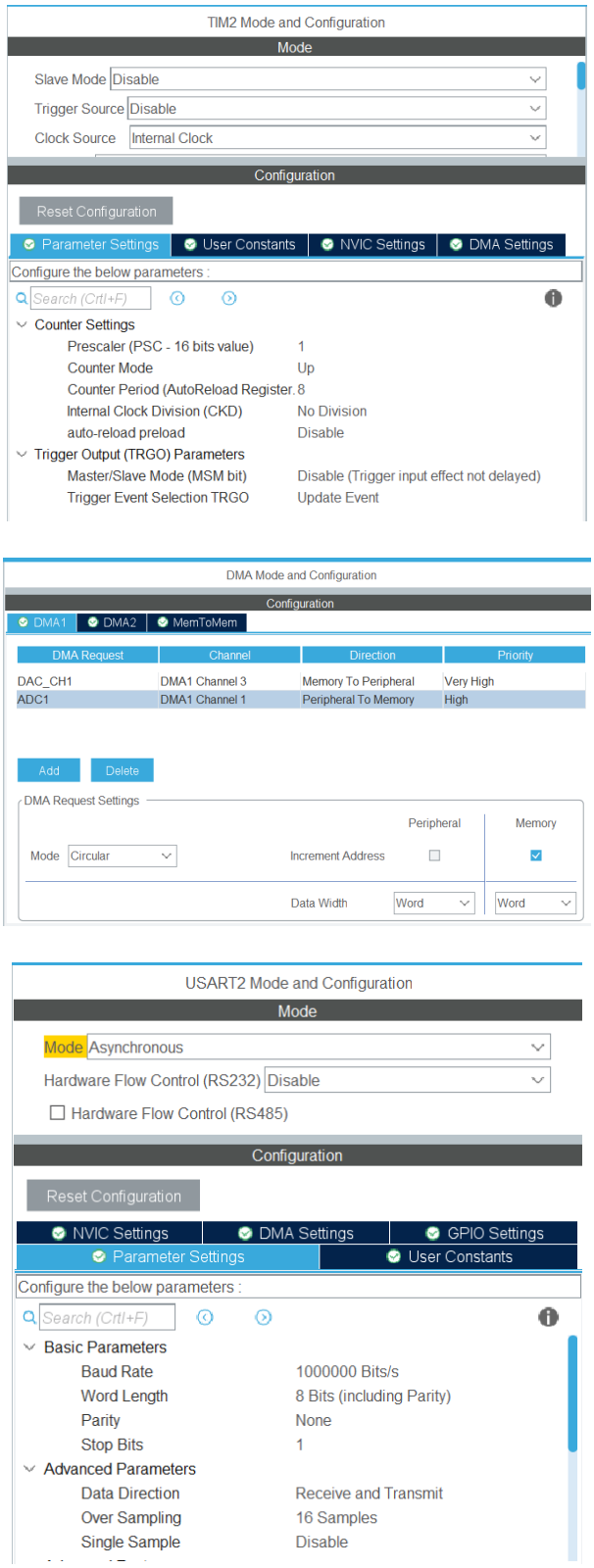


Figure 35, Select settings from ADC, DAC, TIM2 (Timer), DMA, and UART control menus.

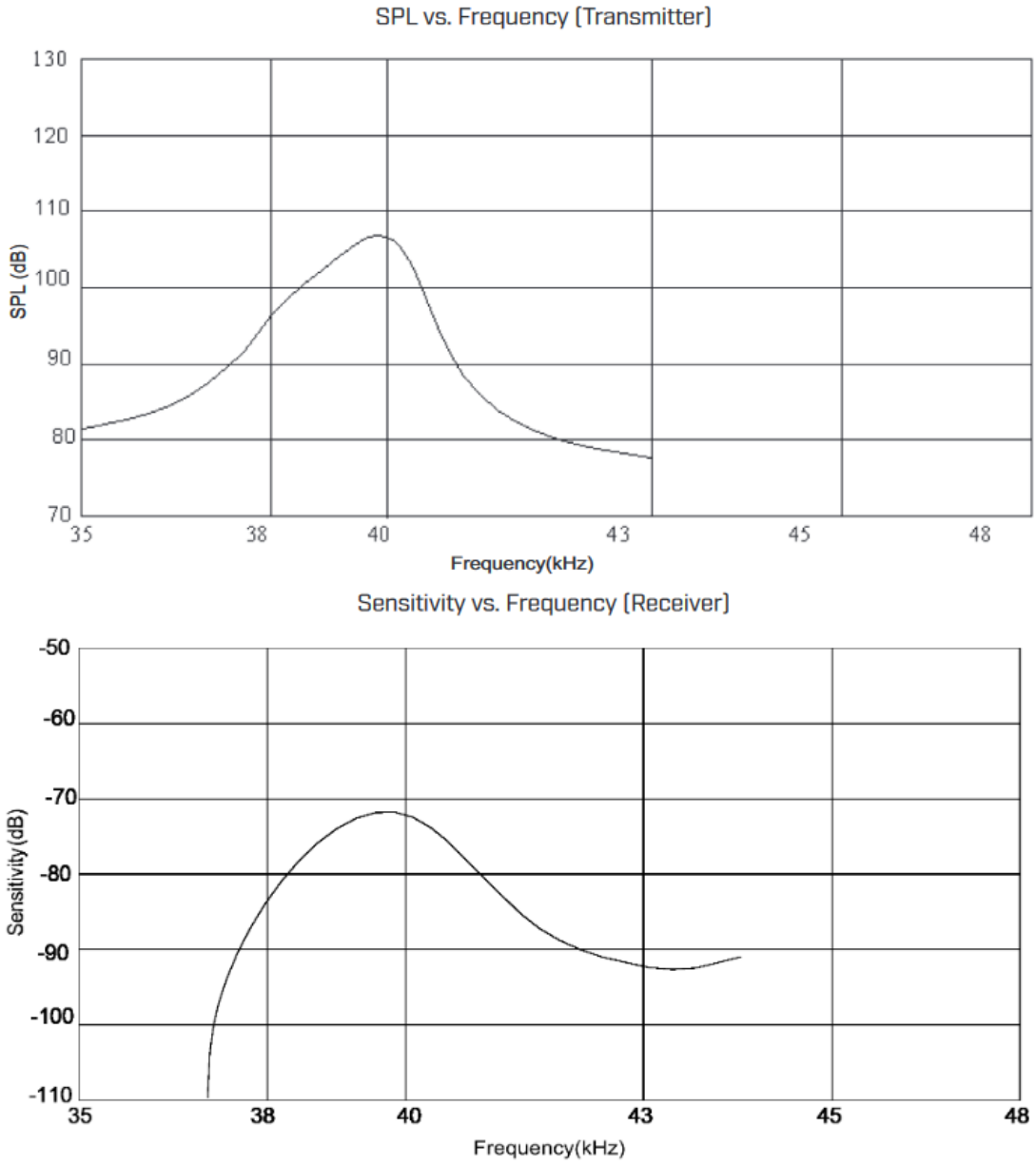


Figure 36, Frequency responses of the CUI CUSA-TR80-15-2000-TH in transmission and receiving configurations. Note the resonant peaks around 40 kHz, and a more linear region above 41 kHz that has less power/sensitivity. This was likely taken at 1 atm bringing its applicability into question.

```
DATA_AN = DATA_RAW(current_index:current_index+NUM_REF-1)-2048;  %intakes single axis from 6 axis dataset and adds 3.3/2V offset
DATA_AN = normalize(DATA_AN); %normalizing, may not be needed
DATA_AN(1:1300)=0; %truncates early transmissions that could be structural vibrations (faster than the speed of sound in air)

% Interpolation of RX
DATA_AN_CC=interp(DATA_AN,4);

TX_FFT = fft(DATA_REF);
RX_FFT = fft(DATA_AN_CC);
C_f = conj(TX_FFT).*RX_FFT;
FFT_CORR = ifft(C_f);


%Windows correlated data based on previous values of correlation peak, and finds new peak
MAX_CORR_I_LAST(cnt2) = round(CORR_LAST(cnt2));
FFT_CORR_windowed = FFT_CORR(MAX_CORR_I_LAST(cnt2)-env_size:1:MAX_CORR_I_LAST(cnt2)+env_size);
[MAX_CORR_M,MAX_CORR_I] = max(FFT_CORR_windowed);
MAX_CORR_I = MAX_CORR_I_LAST(cnt2) - env_size + MAX_CORR_I;
CORR_LAST(cnt2)=MAX_CORR_I;


mydelta = MAX_CORR_I/4;%229.5;

%Record time delay for this time and axis:
MAX_XCORR_RESULTS(time_index,cnt2)=mydelta*dt;
```

Figure 37, Select MATLAB code for FFT-CC with Prior Estimates.

```
% Apply transducer transfer function to get estimate of what we would
% expect to measure with no time delay
wn=39000*2*pi; %Transducer natural frequency (rad/sec)
Q=30; %Transducer Q
sys=tf(wn^2,[1 wn/Q wn^2]); %Transducer simplified transfer function
DATA_REF2=lsim(sys,DATA_REF,t_REF); %Apply once for TX
DATA_REF3=lsim(sys,DATA_REF2,t_REF); %Apply once for RX
%interpolation of TX
DATA_REF=interp(DATA_REF3,4);
```

Figure 38, MATLAB code used to apply the transducer transfer function twice to the reference signal.

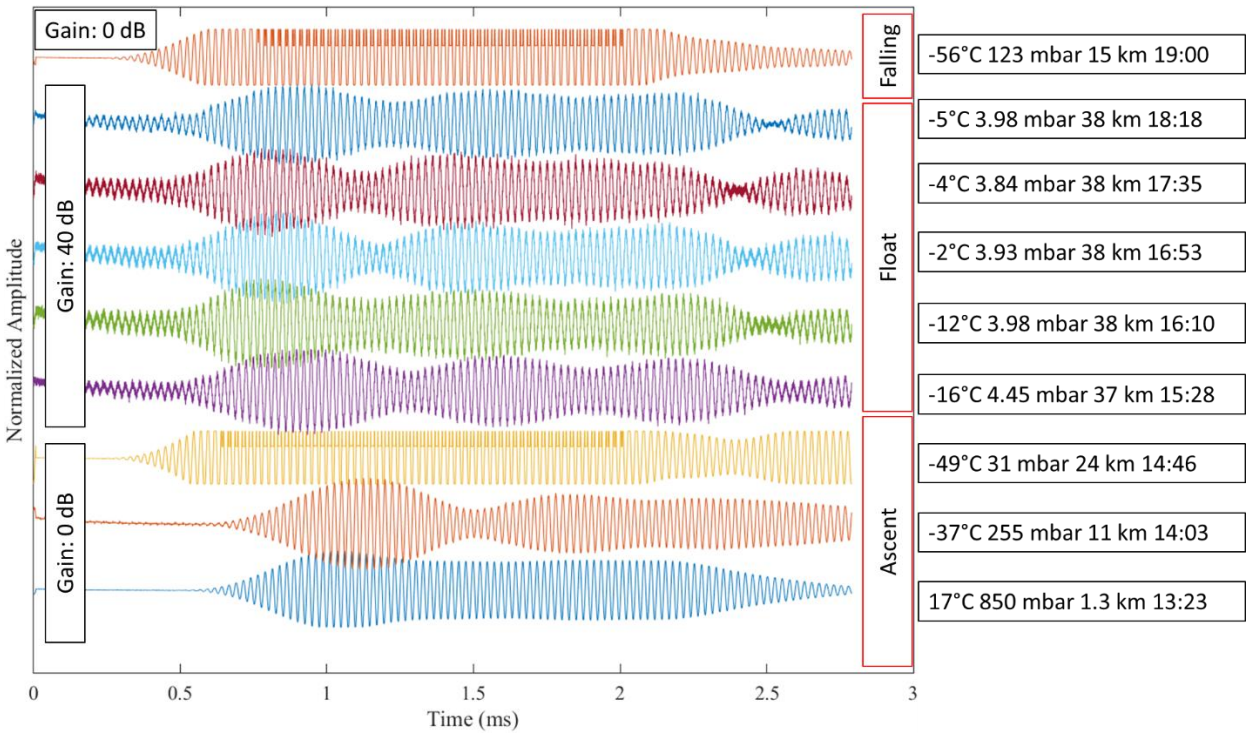


Figure 39, Time domain data taken from anemometer axis not overwhelmed with noise at float. Note the overwhelimg of the signal at temperatures below -40 ºC.

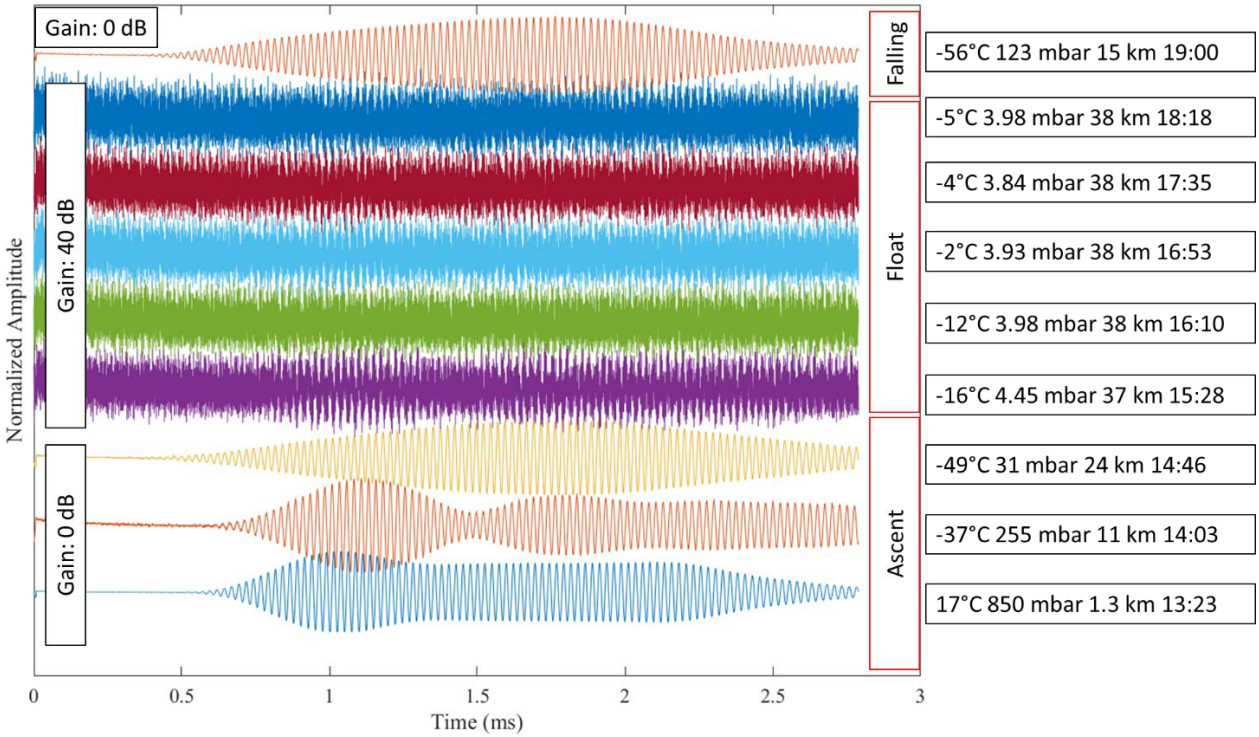


Figure 40, Time domain data taken from anemometer axis with substantial noise at float.