Development & Testing of New Sensors to Detect Ice Accumulation on Aircraft Wings

Senior Design Project Report, Spring 2025

Team Members: Kevin Lopez (CE), John Urban Quezada (EE), Derreck Suhul-Torres (ME), Simon Maranga (EE)

Advisor: Graham Werner

Stakeholder: Simon Clement from Pytheas Technology

Division of Engineering Programs
State University of New York at New Paltz

Expo date: May 2, 2025

Abstract

Aircrafts experience very harsh conditions when in operation. The freezing temperatures along with water vapor in the air can cause ice to accumulate over time. This accumulation can harm the plane's ability to operate correctly, which is dangerous for passengers and pilots. While planes do have deicing techniques used to help with this, there isn't any consistent way to detect ice before it becomes an issue. This project seeks to solve this problem by identifying ice accumulation in real time which will help pilots immensely and keep passengers safe.

The goal of this project is to implement an icing detection system using a piezo electronic sensor and actuator placed on the wings of an aircraft. A piezo can output a signal when it undergoes physical forces such as pressing, pushing, or change in temperature, and is read as a voltage. This was used to create a piezoelectric vibration sensor to measure the changes in dynamic behavior caused by ice accumulation on an aircraft wing. This was tested by attaching piezo electronics to an aluminum beam. The effect of ice was simulated using mass weights and melted sugar on the plate to observe the resulting changes in voltage from the piezo. From this experiment, the relationship between the plate's resonant frequency and the thickness of ice was correlated. The presence of ice shifted the resonance which decreased the voltage reading from the piezo. By determining the resonance frequency through voltage peaks, shifts in the frequency and voltage allowed for effective ice detection. A Fast Fourier Transform (FFT) was used over a repeating time cycle to find the resonance frequency. Through this, current spikes through a resistor were used to observe the thickness of the ice on the metal plate. In addition to this, ANYSYS was used to observe how ice thickness correlates to the resonance frequency. The analysis matched up with the physical tests conducted which verified the functioning of the project design.

Table of Contents

1 - Project Introduction

1.1 Motivation

At high altitudes, the moisture and low temperature of the air can cause an accumulation of ice on the wing, which decreases a plane's lift force. A decrease in a plane's lift has many consequences, one of which is a decrease in flying altitude. As of right now, a pilot must physically turn their heads and look out the window and make a judgement call before activating the deicing mechanisms. This device seeks to solve that problem by displaying a live broadcast of the amount ice on each wing on a screen in the cockpit. Since this device is small, multiple sensors can be placed along each wing, therefore giving the pilot a detailed outline of the state of the wing.

1.2 Need for Project

When planes fly, they accumulate ice on their wings, which decreases lift due to the change in airfoil surrounding the wing of the aircraft. This is dangerous because a lack of lift force could cause the plane to crash in the worst-case scenario. A damaged plane financially affects the company that owns the plane and could pose a risk of injury and/or death to the passengers aboard. Currently, there is not a form of ice detection integrated into aircraft systems apart from visually identifying ice accumulation.

1.3 Benefits of Project

The ice sensor that was developed has the potential to benefit society in many different ways. From an economic point of view, the sensor is cheap to make and can have its costs scaled down. From an environmental perspective, the sensor doesn't require much power to run, which in turn lowers power consumption from the plane. In a societal context, this device can save lives due to the live updates that it can provide to pilots. If the problem of ice accumulation can be addressed in a more efficient way, such as the way that this project does, then there is a chance for everybody to benefit.

1.4 Project Scope

We designed an ice detection system for planes using a piezo vibration sensor. It is small so that multiple can fit on a wing, and sensitive enough so that it can detect any changes. We first studied the piezo's response to different AC signals passing through it, as well the changes in its response when a mass is accumulating on it. We studied the materials used on the plane as well as materials that could be used to protect the piezo to get a better understanding of the sensitivity of the system in a real-world environment. Once completed, it was hooked up to an external display that will show live updates of ice accumulation. The sensor only detects ice and does not incorporate a method of removing the ice.

1.5 Design Objective

1.5.1 - Major Objectives

- Displays change in mass of ice in real time
- Occupies an area smaller than 1 square foot
- Consumes very little energy

1.5.2 - Functional Objectives

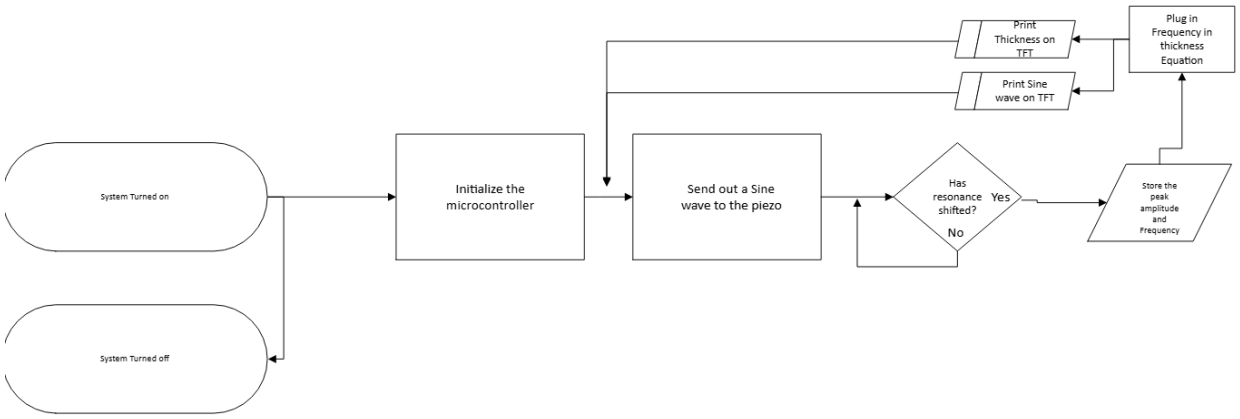
A constant sinusoidal voltage is applied to piezo that is attached to the bottom of a cantilever beam. A second piezo is attached on the other side of the beam directly above the first piezo. As the AC signal sweeps at different frequencies, the resonance frequency of the piezo causes a massive voltage spike, which gets detected by the piezo on the other side of the beam. As a mass is applied to the beam, the resonance frequency shifts downwards, and this change can be translated into thickness.

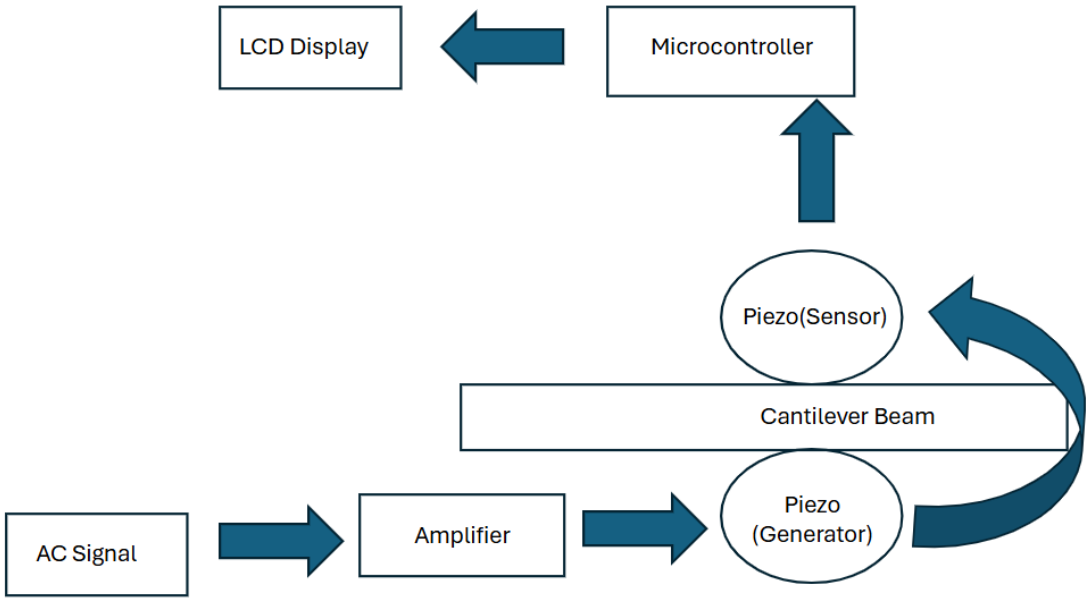
Function	Qualification
Detect changes in piezo's resonance frequency	The microcontroller will be able to trigger an output (in this case turn on an LED) if the resonance frequency goes outside of a predetermined range of +/- 15% from the baseline.
Adequate insulation	The circuit should not fail if tested after being left in the freezer for 12 hours
Accurate mass readings	Given the change in frequency, the microcontroller should be able to accurately determine the mass added to the sensor within 15% error and display it on a set of 7 segment displays
Stable connections	When mounted, the piezo should stay firm and not disconnect from the circuit or fall from its holding place. Conditions it needs to withstand are high speed winds (~12 mph) and approximately 3 pounds of ice

Quick response time	The sensor should display any changes within 1 second (+/- 0.25 sec)
Compact design	The entire sensor should fit in a 0.25 square inch area

Since this sensor is supposed to go on a plane's wing, we also wanted to rate it for weather conditions that a plane typically experiences, including high speed winds and freezing temperatures. Since we wanted to have an entire network of these sensors lining the plane wing, we wanted to take up as little space as possible with the design. In addition, we wanted to implement some kind of display that shows all the relevant information that a pilot should know about the wings.

2 – Functional Description





An AC signal is amplified and later used as an input to the piezo that acts as a generator. The piezo directly above it mimics the frequency response of the first piezo as mass applied on the beam changes. This data is interpreted by the microcontroller and is displayed on the LCD screen.

3 – Preliminary Designs

3.1 – Prototyping Considerations

For different prototype considerations, the main component that was changed was the type of piezo that was used for testing. Initially, the piezo used was not easy to work with. It required a large input wave to function which made finding the resonance difficult. Because of this, a different type of piezo was used which ended up being the ones used in the final design. The new piezos used were piezos utilized in guitar tuning. This meant that they were more sensitive and required less input to observe their behavior. The design for the project did go through many changes but switching the type of piezo used was the main alternative in terms of using different components.

3.2 – First Prototype

3.2.1 – Overview of Design

The first prototype was simply just a singular piezo on a breadboard that had an input sine coming in on one side while the other side was grounded. The voltage was measured across the piezo in an attempt to find resonance.

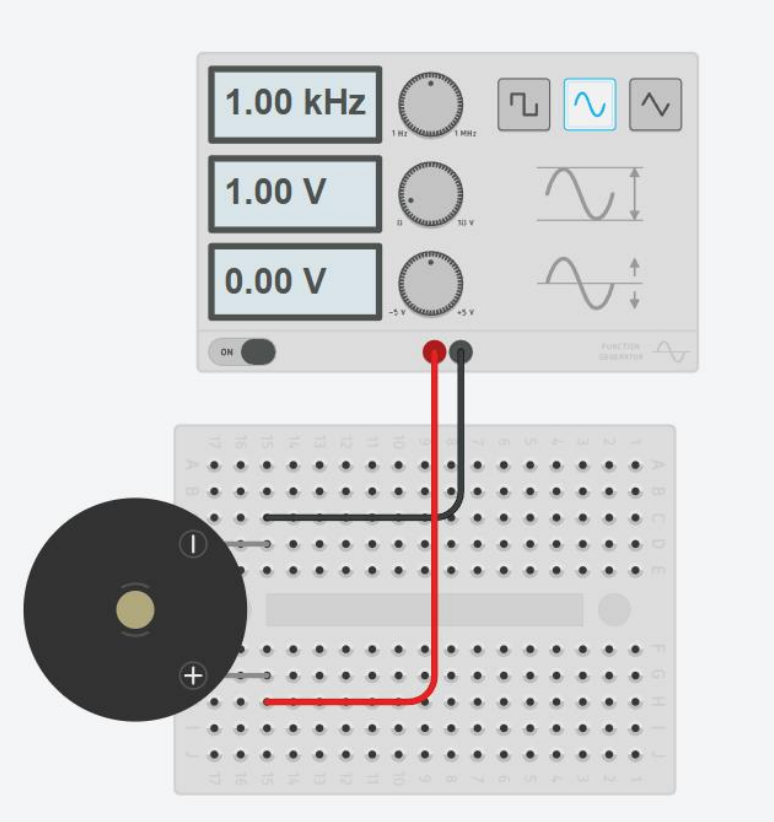


Figure 1: First Prototype

3.2.2 –Design and Integration

The test conducted with the initial design seen in the figure above did not produce adequate results. This was due to many different factors. The main issue was that the type piezo used at first was difficult to work with due to its insensitivity. It required a large AC voltage and frequency to observe any audible sounds and the resonance. At the time, the only method of producing a sine wave was using the software Agilent. Because Agilent could not produce significant voltage ranges, the piezo was barely audible making finding resonance difficult. Another issue was the fact that initially, voltage was measured across the piezo itself. Due to the large resistance of the piezo, the current was incredibly low meaning that the voltage was too low to read consistently.

From these issues, a new type of piezo(a guitar piezo) was used and new circuit was constructed to account for the issues mentioned above. The new circuit was exactly the same, but a resistor was added in series to the piezo that connected it to ground. The resistor was simply used to measure voltage which gave better results for finding resonance. From here, the piezo was attached to a metal plate as opposed to just hanging in the air. This metal plate was used to simulate the wing of an aircraft. The new piezo was more sensitive making it easier to work with

the low voltage inputs given by Agilent. However, attaching it to the metal plate still caused problems with the input signal being too weak. This lead to another addition to the circuit which was crucial in getting the system to operate consistently.

An op amp was used on the input signal which allowed for higher voltage. This made finding resonance even easier. In addition to this, a second piezo was attached to the metal plate. The first piezo would act as the actuator to vibrate the metal plate and the second piezo was simply a measurer used to find resonance. This setup is what allowed for experimentation of finding resonance and detecting "ice" through shifts in that resonance.

3.2.3 – Experimentation

For the experiment conducted, melted sugar was put onto the metal plate to simulate ice. The amount of melted sugar added was measured in "scoops". The first test involved finding the resonance by manually changing the frequency of the input sine wave and observing at which frequency the voltage across the measuring piezo peaked. These values were recorded in excel with frequency on the x axis and voltage on the y axis. The resulting graph showed clearly where resonance was happening as seen here in one of the first tests conducted.

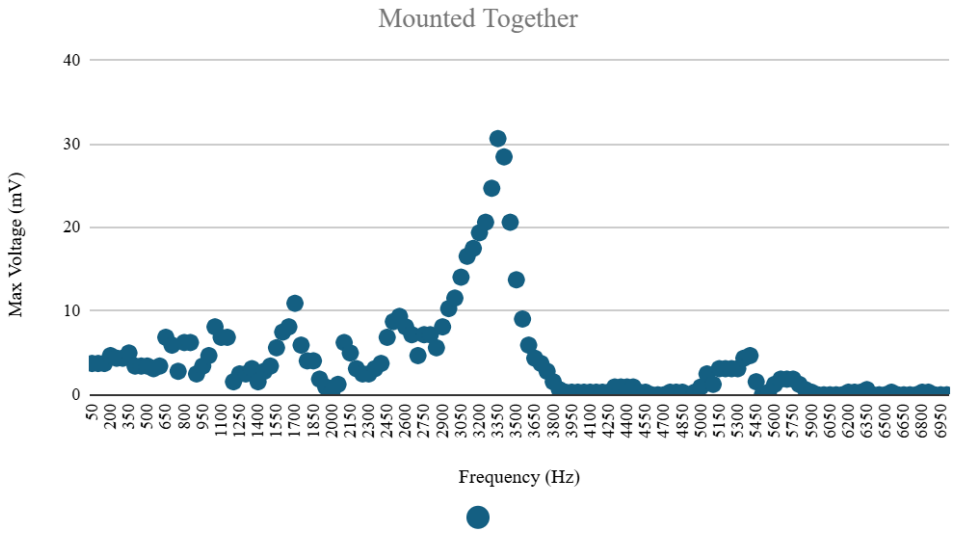


Figure 2: Finding Resonance

From here, the melted sugar was added to the plate to see if there were any changes in the resonance. The exact same test was conducted but with half a scoop of melted sugar added. The resulting graph showed that the resonance shifted downwards as seen here.

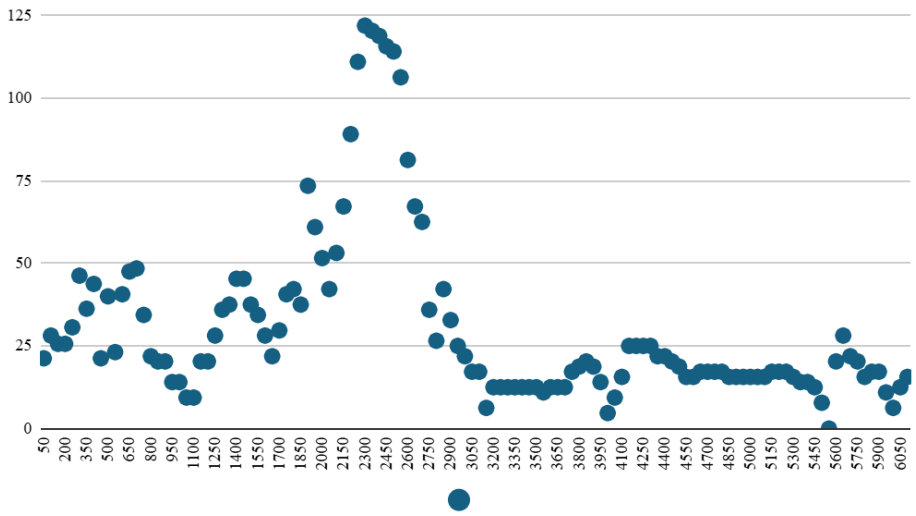


Figure 3: Finding Resonance(0.5 scoops of sugar)

This verified the circuit design as viable for accurately measuring resonance and observing any additions of “ice” to the metal plate. This test was also done using current as opposed to voltage as seen here.

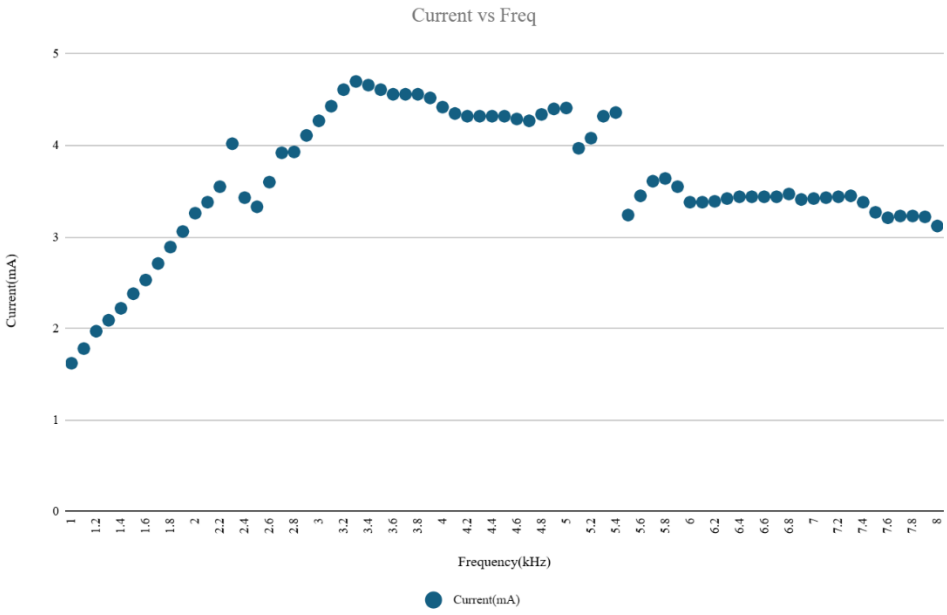


Figure 4: Finding Resonance Using Current

The final step for this was using an Arduino to automatically find resonance by searching for the peak voltage and its given frequency. It did this through using a sine sweep to look at each voltage and frequency. If the peak voltage occurs at a different frequency, the system knows that ice has been added and uses the shift in frequency to calculate the thickness of the ice. Some

changes were made to the design so the Arduino could produce the sine wave itself without using Agilent. A different metal plate was also used in addition with mass weights instead of sugar to simulate ice.

3.2.4 - Results and Lessons Learned

The circuit had to be reconstructed multiple times to actually find resonance. The reason it did not work before was simply because the sine wave coming through the piezo was not strong enough so the resonance could not be found. In addition, Fast Fourier Transforms were utilized to try and find resonance through the Arduino but this did not function as intended. Chirping was the method used to find resonance instead.

3.3 Various Iterations Prototype

One of the starting microcontroller prototypes was setting up a Piezo to behave as a pressure sensor

4 – Final Design

4.1 – Overview of Design

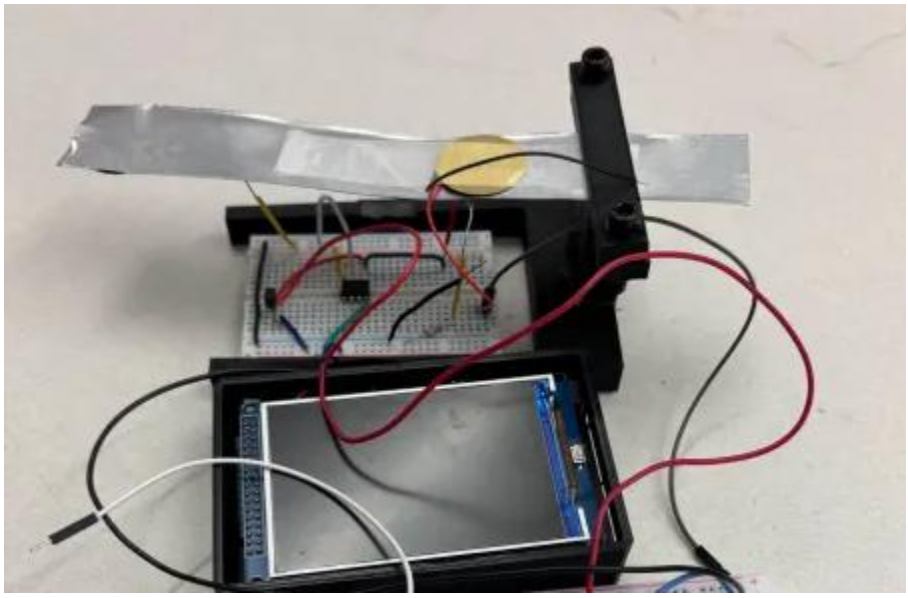


Figure 5: Final Design to determine changes in resonance

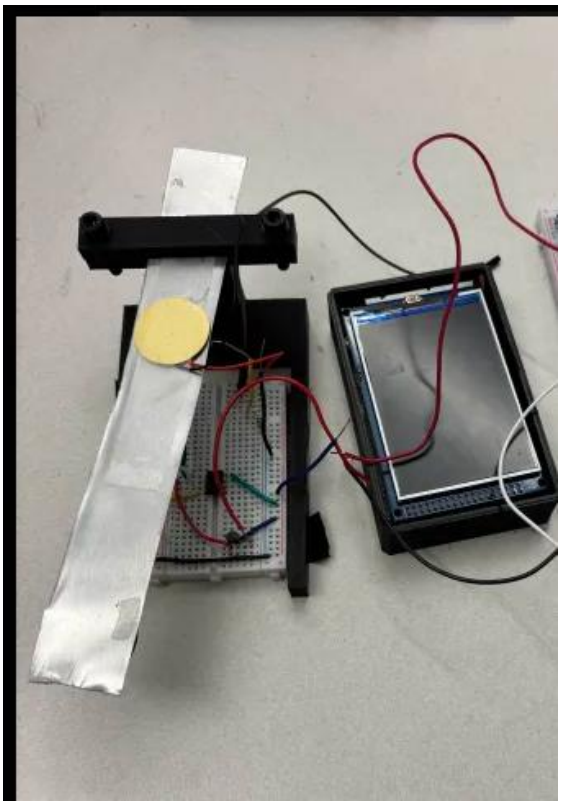
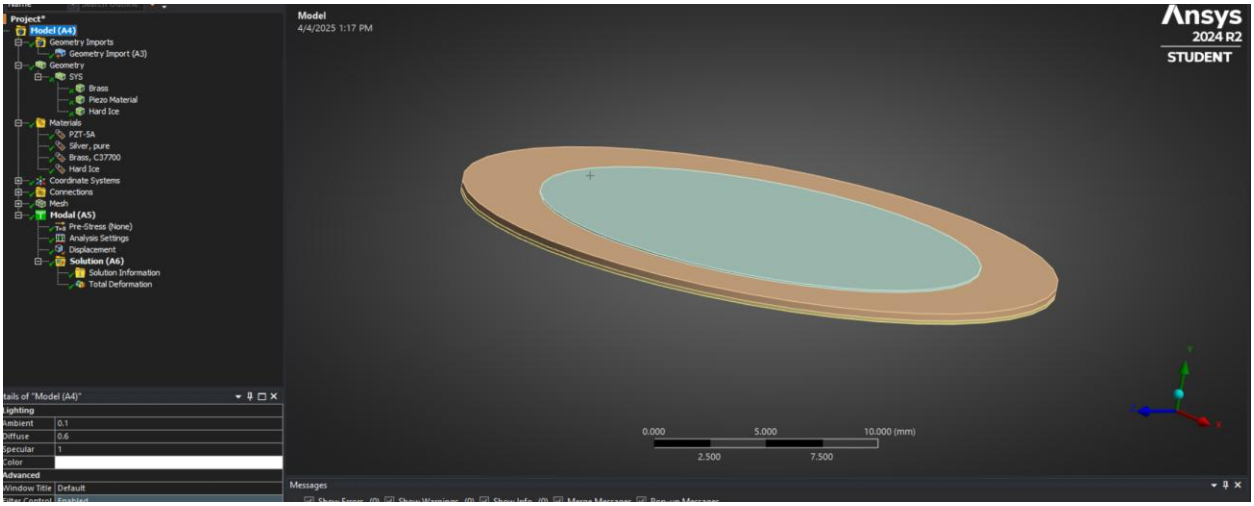


Figure 5: Final Design to determine changes in resonance(Different Angle)

The main prototype involved using a simple aluminum plate/support system. On one end of the underside of the structure was attached one piezo (using superglue). On the other end of the underside was attached another piezo. The first piezo was directly attached to an amplified AC signal, while the other end was connected to an oscilloscope. The other protype involved two piezos being directly attached to each other back-to-back. For both designs, we needed a spur sticky substance that could hold the piezos up against the force of gravity. For the final design, we chose that substance to be JB Weld epoxy. One piezo received an amplified AC signal, while the other was connected to an oscilloscope. For the final design, we 3D printed a mount because it provided a more polished look to the design, while at the same time allowing us to swap out different aluminum plates. Below the support, there was a small space that housed the breadboard the circuit sat on. The circuit contained an N-type silicon MOSFET, an operational amplifier (LM358), a small resistor (100 Ω), a few wires, and of course the piezos. The amplifier was needed because the input signal to circuit was not large enough to provide the second piezo with any relevant information. The MOSFET was needed to increase the current flowing through the circuit and make the data “smoother”. The resistor was included to stabilize the circuit and decrease the runoff. In addition to the supplies required to create the circuit data, we needed to buy a microcontroller with enough memory to and strength to handle frequency sweeps across the piezo. For this task, the Arduino DUE was chosen. To display the results of the sweep, an LCD was chosen, specifically a TFT LCD.

4.2 – Design and Integration

For the final design, we chose to have three different aluminum plates, with the piezo mounted in different locations on each. On the first plate, the piezos were attached close to the location of where the beam would be attached to the support system. On the second plate, the piezos were attached in the middle of the aluminum plate. On the third plate, the piezos were attached on the opposite side relative to where the plate would be attached to the support. We decided to have two piezos on opposite ends because after all our preliminary tests, we found that a single piezo was not good at being a generator and a sensor at the same time. In addition, we found that having the piezos too far away (opposite ends) was not effective because it gave noise a huge opportunity to be introduced, as well as the signal could not be kept entirely intact. Having the piezos directly on top of each other was not ideal as it would require the ceramic of one piezo to be attached to the metal of the other, and that would muffle the signal, if not kill it entirely.



ANSYS modeling of piezoelectric sensor

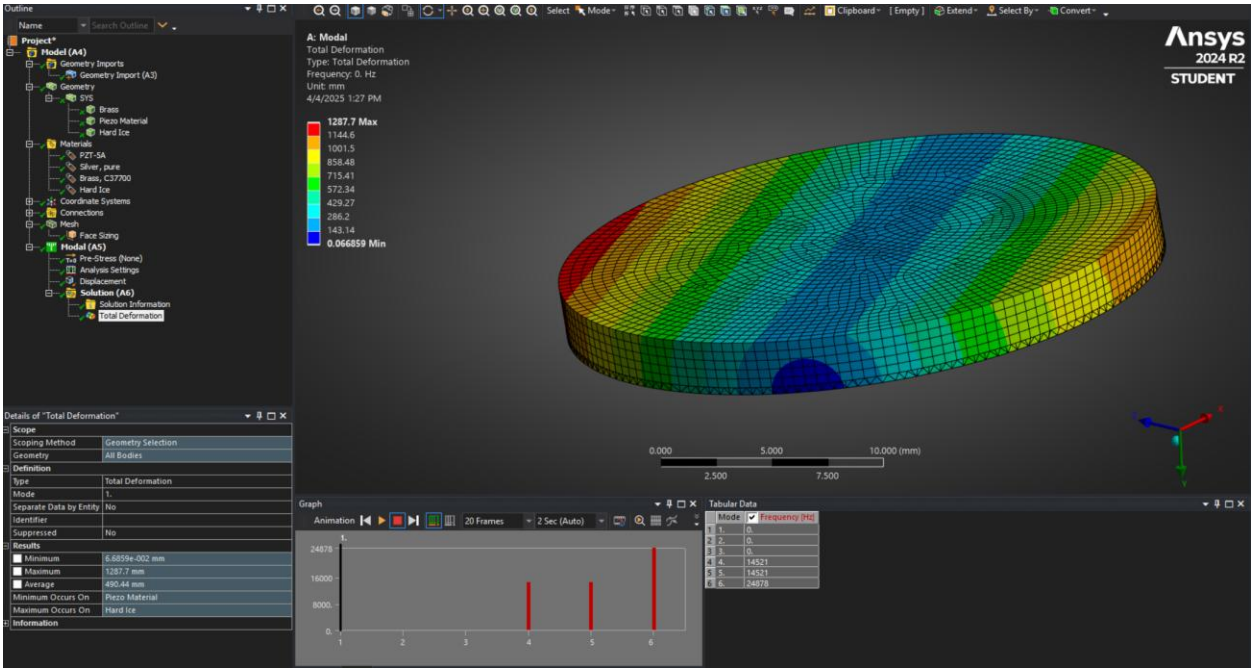
The piezoelectric sensor itself was modeled for modal analysis. The reason behind this was that this was the model that gave the most similar resonance frequency that matched what was being recorded during experimentation. The 2 materials chosen for modeling the sensor were brass and piezoelectric material. These were both found within the material library within the software. Ice was not a preset material in ANSYS, so the material properties for it had to be found using online sources and assigned accordingly. The boundary condition of the sensor was placed such that the back of the sensor (brass only) was fixed, as it was supposed to represent being attached to the wing.

The purpose of conducting finite element analysis on the sensor was to develop a calibration equation that can relate the resonance frequency of the plate to the thickness of ice that built up on it. This was done by modeling an additional layer of ice and running the simulation in increments of 0.01mm until it had reached 2mm. This allowed us to be able to know the approximate resonance frequency that the system should be at as ice builds up.

Material Used for Piezo Sensor Modeling			
Layer 1	Brass	Thickness	0.33 mm
Layer 2	PZT-5A (piezoelectric)	Thickness	0.08 mm
Layer 3	Hard Ice (sticks to wing)	Thickness Range	0 - 0.2mm

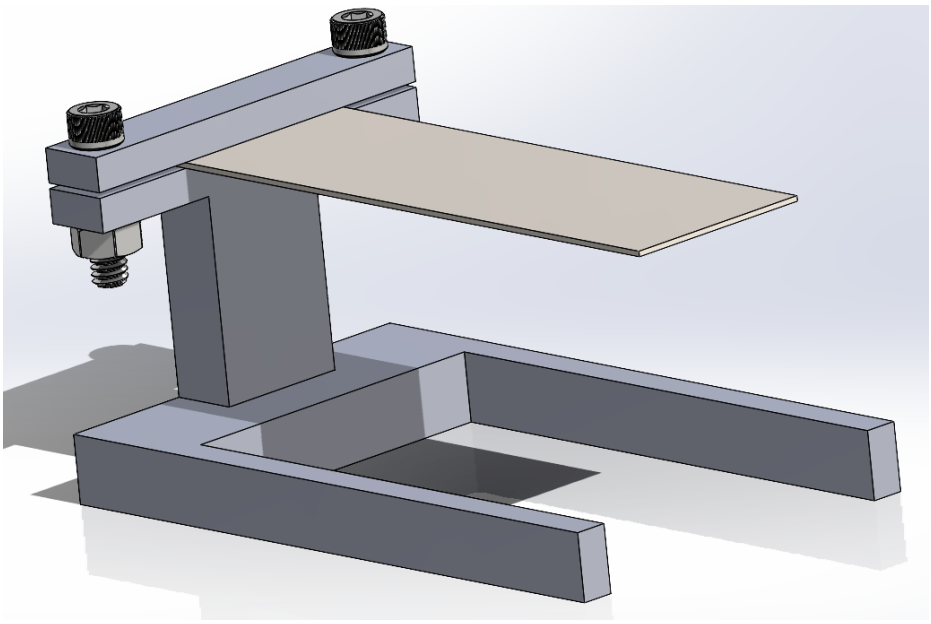
Material Properties Used for Ice Simulation		
Density	917	kg/m^3
Elasticity	10.00	GPa
Poisson Ratio	0.3	

Material Properties and dimensions used in ANSYS



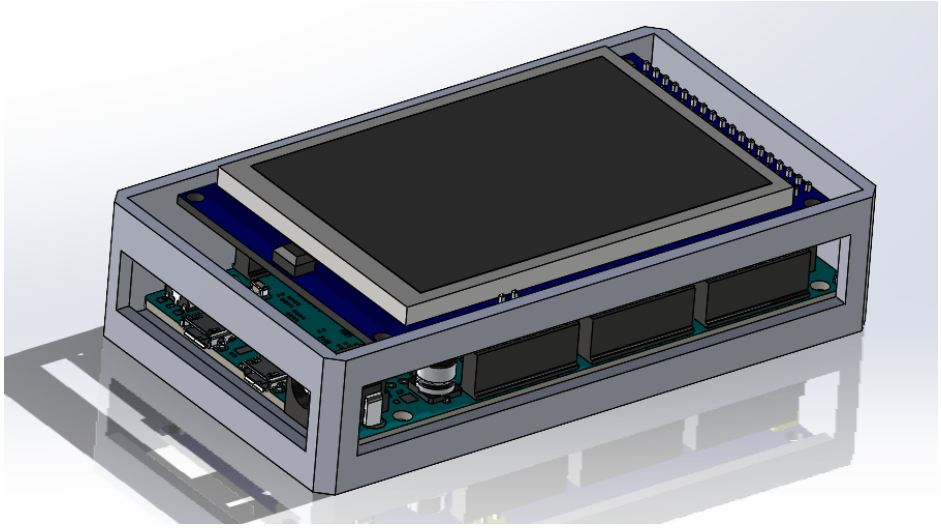
Modal results of piezoelectric sensor with 2mm ice thickness

4.3 – Overview of Design



Apparatus setup for cantilever beam

The modeling for the apparatus involved breaking up the CAD part into 3 separate components: The base plate, the T-section and the top bracket. The goal of this apparatus was to provide a cleaner cantilever beam for the sensor to lay on while running tests. The bottom plate was extruded out to make space for the circuit, as well as the wiring for the piezo sensor. The bracket was mounted using two M5-20 bolts and nuts, and an aluminum plate was used as our medium for the sensor to pick up on the resonance frequency. This set up was 3D printed using PLA.

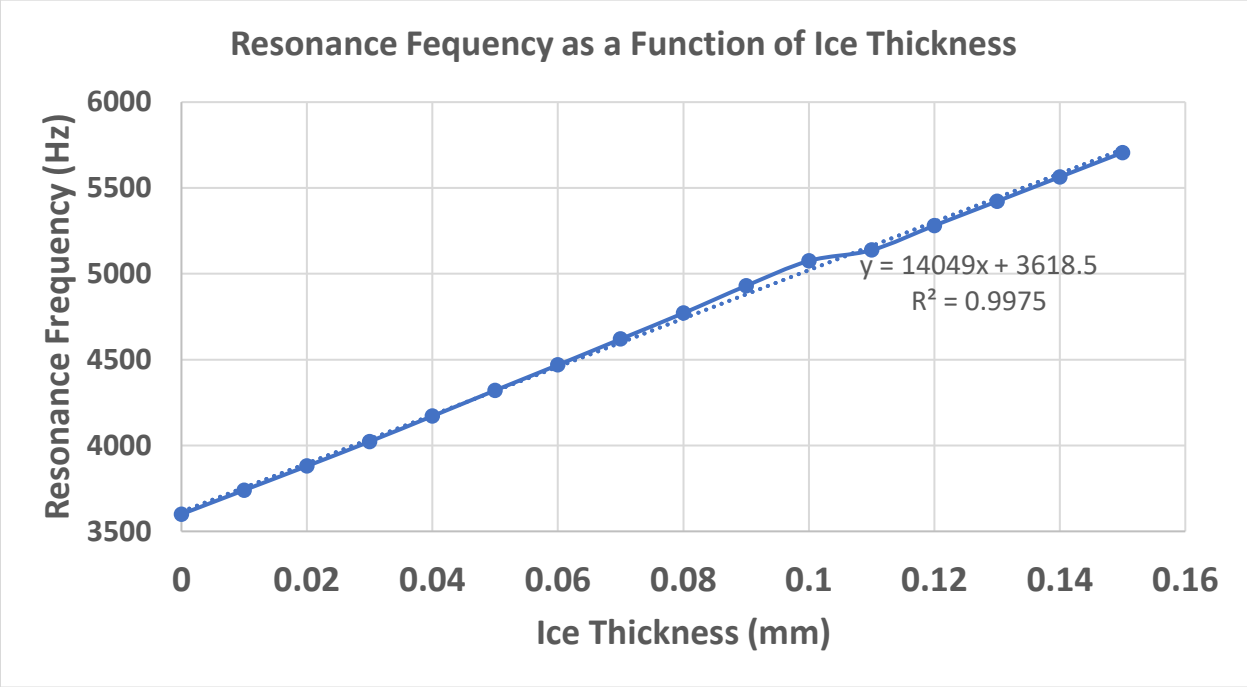


Case for the TFT LCD and the Arduino DUE

The case for the Arduino was also 3D printed using PLA. The case was created to ensure that the Arduino was not damaged or became unplugged when the sensor was being presented, as well as providing a cleaner look for the LCD as it was being presented.

4.3 – Experimentation

To verify that the system worked as anticipated, we added mass in the form of weighted discs on top of the beam. Then we compared what the LCD display was telling us to what the model that was found in ANSYS should have given us. We found that the sensor was the most accurate when the weights were added near or close to the second piezo. In addition to the location of the weights, the most accurate measurements also occurred when the piezo was attached close to where the aluminum plate was attached to the stand.



Plot demonstrating resonance frequency as a function of ice accumulation

Plotting the ANSYS results, we can establish the calibration equation for the sensor as ice builds up on it. We can see that it is a linear relationship between the ice and resonance frequency. Interestingly, we find as ice builds up, the stiffness plays a bigger influence than mass. This is observed since ANSYS demonstrates that the slope is positive, indicating that resonance frequency goes up as ice accumulates.

However, since we were unable to have actual ice buildup on the plate during presentation, this calibration equation needed to be adapted for other situations. In our case, we used brass mass to simulate ice being built up on the plate. This means that the resonance frequency would go down as more mass is applied, since they are inversely related. Had we kept the original equation, we would see an output of negative thickness as mass is applied, which is impossible.

5 – Final Design Performance

Function	Qualification	Achievement	Performance as Tested
Detect changes in piezo's resonance frequency	The microcontroller will be able to trigger an output (in this case turn on an LED) if the resonance frequency goes outside of a predetermined range of +/- 15% from the baseline.	Exceeded	Microcontroller captures and translates the new resonance frequency into an ice thickness
Adequate insulation	The circuit should not fail if tested after being left in the freezer for 12 hours	Not tested	Works well when tested at room temperature. Not tested at other temperatures besides room temperature
Accurate mass readings	Given the change in frequency, the microcontroller should be able to accurately determine the mass added to the sensor within 15% error and display it on a set of 7 segment displays	Unmet	Instead of mass, the LCD displays ice thickness at a maximum accuracy of 83%. Worst case accuracy of 24%
Stable connections	When mounted, the piezo should stay firm and not disconnect from the circuit or fall from its holding place. Conditions it needs to withstand are high speed winds (~12 mph) and approximately 3 pounds of ice	Exceeded/ not tested	Although the system looks rugged, all connections are tightly made and will not fail unless they are thrown across the room. Not tested at high speed winds or with physical ice
Quick response time	The sensor should display any changes within 1 second (+/- 0.25 sec)	Not met	The LCD updates its readings every 10-30 seconds

Compact design	The entire sensor should fit in a 0.25 square inch area	Not met	The circuit/stand alone occupy about half a square foot of space

The final product met or was close to meeting the most important specifications, which were stability, speed of results, and close to tolerable ice thickness measurements. Although it is far from being a usable product in a physical setting, it is a good baseline for future iterations.

6 – Recommendations for Future Work

The limitations of this project are packaging, accuracy of data displayed, external noise, and speed at which results are displayed. Since this sensor is meant to be connected to a plane's power systems, it is difficult to design a final package. However, for demonstration purposes, the packaging could be improved by putting all the circuity on a printed circuit board (PCB) and replacing the bulky power supply with a DC battery. In addition, the aluminum can be replaced with a more elegant and firmer metallic structure that can provide better results. The accuracy of the data can be fixed by doing more laboratory tests and ANSYS tests until a better model can be inserted into the microcontroller. The external noise can be reduced by using a much better operational amplifier. The amplifier we used had a good common mode rejection ratio for academic purposes but is not good enough for industrial purposes. In addition, we could also open the possibility of placing the circuit in a soundproofed container and experiment with that. The speed at which the results are displayed will be difficult to overcome because the results can only ever be displayed once an adequate sine sweep is conducted over a period. This limitation can only begin to be addressed when the other limitations are overcome.

For the expansion of this project, we could implement and train an artificial intelligence (AI) module such that it can automatically activate deicing mechanisms once the microcontroller produces results outside of a tolerable range of ice thickness. In addition, we can scale down the whole circuit by using smaller components that are used in the electronics industry.

7 – Conclusion

Overall, the project was a success because it can detect change when an external force is applied to the sensor using very little energy. Although the product was far from meeting some of the specifications we initially wanted to implement (such as speed of results and package size), it met the most important one which was using a piezo's frequency response in order determine the amount of ice that has accumulated on an external surface. Despite this goal being achieved, it can be improved upon further by scaling the size down, integrating more complicated systems such as AI, using better equipment, and more rigorous testing. Furthermore, this project is not

good enough to be used in a real-world scenario, but with further experimentation and improvement it can one day.

References

[1] "In-Flight Icing | Federal Aviation Administration," *Faa.gov*, 2016. https://www.faa.gov/nextgen/programs/weather/awrp/ifi

[2]"Standards," *SA Main Site*. https://standards.ieee.org/standards

8 - Additional Information

8.1 Disciplines Used During Project

Below, identify which courses were used to generate a design solution. If a course you used is not listed, please add it.

Table 8.1: Electrical Engineering (EE) Disciplines and Courses within discipline in the EE program

Discipline	Course in the discipline	Check if used
Computers	EGC251 C/C++ Programming	
	EGE331 Computer Simulation (MATLAB)	
	EGC331/EGC332 Microprocessors + Lab	
Analog Electronics	EGE200/EGE201 Circuit Analysis + Lab	
	EGE320/EGE322 Electronics I + Lab	x
	EGE321/EGE323 Electronics II + Lab	
Signals/ Systems	EGE311 Signals and Systems	x
	EGE416 Control Systems	
	EGE417 Digital Control Systems	

	EGE412 Communication Systems Theory
	EGE493 Applied Digital Signal Processing
Electromagnetism	EGE340 Applied Electromagnetics
	EGE445 Antenna Systems
	EGE493 Intro to MEMS
Energy Systems	EGE350/EGE351 Electric Energy Systems + Lab EGE452 Electric Power Systems

Table 8.2: Computer Engineering (CE) Disciplines and Courses within Discipline in the CE program

Discipline D1) Computers	**Course in the discipline**	**Check if used**
Software	EGC251 C/C++ Programming	X
	CPS210 Computer Science I	
	CPS310 Computer Science II (Data Structures) CPS353 Software Engineering	
Computer Systems	EGC331 Microprocessors + EGC332 Microprocessors Lab	X
	EGC433 Embedded Systems	X
	EGC442 Computer Architecture	
	EGC451 Real-Time Systems	
Digital Systems	EGC220 Digital Logic Fundamentals + EGC221 Digital Logic Lab EGC320 Digital System Design	
	EGC441 System On Chip	
	EGC445 VLSI Design + EGC446 VLSI Lab EGC447 Functional Verification	X

Table 8.3: Mechanical Engineering (ME) Discipline and Courses within Discipline in the ME program

Discipline	Course in the discipline	Check if used
Computers	EGE331 Computer Simulations	
	EGM302 Finite Element Analysis	X
	EGM393 Advanced Computer Aided Design	X
Mechanics & Machines	EGM211 Statics	
	EGM212 Dynamics	
	EGM311 Kinematics of Machines	
	EGM312 System Dynamics	X
	EGM393 Biomechanics	
Thermodynamics/	EGM331 Thermodynamics	
Fluid Dynamics	EGM332 Fluid Mechanics +	
	EGM333 Fluid Mechanics Lab	
	EGM334 Heat Transfer	
	EGM335 Thermo-systems Design	
Materials	EGM221 Engineering Materials	
	EGM322 Mechanics of Materials +	X
	EGM323 Materials Lab	
	EGM393 Composite Materials	
	EGM393 Design of Machine Elements	

8.2 – Design Constraints Driving Project

The table below shows the design constraints that drove this project.

Design constraints that drove this project

Design Constraint	**Check if applied to this project**
Economic	x
Manufacturability	
Social and Political	
Environmental	
Ethical	
Sustainability	
Health and Safety	
Other (explain)	Must be implemented using piezos

Since the school only provided our team with $200, it greatly limited our exploration of different components technologies that might have been better. For example, an amplifier with a better slew rate (rate at which it changes states) and common mode rejection ratio (noise cancellation) could have benefitted in the processing of signals. In addition to the economic constraint involved with research and design, the final product had to have been relatively cheap to manufacture since a plane wing would ideally have multiple sensors on each wing.

Since this was a research-based project for our stakeholder PYTHEAS technology, we were limited to using piezoelectric components in the development and implementation of our sensor. This was a constraint because there were a few other methods of designing a sensor that could meet the same specifications without the piezo (i.e. a water sensor).

8.3 – Use of Engineering Standards

Engineering Standard	Where it was used in the Project:
IEEE 1451 - Smart Transducer Interface	Setting up the line of communication between the piezo and the Arduino DUE
IEEE 1057 – Digitizing Waveform Controllers	The processing of analog signals from the piezo by the microcontroller
IEEE 829 – Software Test Documentation	The testing of the microcontroller's software response

8.4 - New Knowledge Acquired

We learned about the concept of resonance frequency, and how it can be used within a circuit. We were first introduced to the concept by our project advisor, and we did a lot of reading about it online. The most insightful piece of information we got on the subject was an old video of a bridge rocking back and forth because it was resonating with its environment. Through testing, we learned certain properties about an item's (specifically a piezo) frequency response. We learned the most about these things when we did our initial tests of burning sugar on the aluminum plate and physically seeing/hearing the resonance frequency of the piezo shift downwards. In addition, we learned about (Fast) Fourier Transforms and how they can be used to process signal information. Specifically, we used Fourier Transforms to determine where the resonance frequency was when different masses were added.

8.5 - Application of Engineering Principles

Electrical Engineers

- Signal amplification – In order to hear the piezo functioning, we had to amplify the signal using an operational amplifier. Different iterations of the amplifier using feedback were conducted to see the piezo's different responses
- Noise cancellation – Although not implemented in the final design, the removal/cancellation of outside noise was attempted using RC circuits and BJT differential pairs. Similar principles were used when trying to make a hyper-selective frequency filter as a prototype.

Computer Engineer

- Learned how to effectively perform a sine sweep from a microcontroller using Pulse Width Modulation (PWM)
- Learned how to perform a Fast Fourier Transform to extract the resonance frequency
- Learned how to do "chirping" to extract the value of the voltage spike from the resonance frequency

Mechanical Engineer

- Vibration Analysis – The fundamental principle of the sensor involved utilizing the resonance frequency of the system and observing the influence of ice on the natural frequency as it accumulates.
- Solid mechanics – The piezoelectric sensors needed to be placed in locations of the beam that would subject it to the most bending, allowing for the output signal to be its greatest.
- Finite element modeling – Conducting a modal analysis using ANSYS was critical for developing the calibration equations

Appendix

Appendix A Ice Sensor code

```
/*************************************************
 * Ice Detection / IceSensor.c
 *
 * DAC signal generating, TFT setup, ADC reading
 * for product sensor
 *
 * Author: Kevin Lopez
 * Date Last Modified: 5/2/2025
 *
 *************************************************/

#i#include "arduinoFFT.h"
#include <Arduino.h>
#include <UTFT.h>
#include <math.h>



// Fonts from UTFT's DefaultFonts.c
extern uint8_t SmallFont[];
extern uint8_t BigFont[];

// ILI9486, RS=38, WR=39, CS=40, RST=41 as in your working demo
UTFT  myGLCD(ILI9486, 38, 39, 40, 41);

#define NUM_SAMPLES   4096
#define SAMPLE_FREQ   40000     // 20 kHz ADC

#define GRAPH_HEIGHT  200       // vertical graph area
#define GRAPH_WIDTH   319       // horizontal graph are
//Chirp parameter
const float F_START = 1000.0f;   // start freq (Hz)
const float F_END   = 10000.0f;  // end   freq (Hz)
const float DURATION = 5; // seconds for one sweep
const int TABLE_SIZE = 256;

// buffer for time-domain samples
double vReal[NUM_SAMPLES];
double vImag[NUM_SAMPLES];
```

```
 40.
 41. ArduinoFFT<double> FFT = ArduinoFFT<double>(vReal, vImag, NUM_SAMPLES, SAMPLE_FREQ);
 42. uint16_t sineTable[TABLE_SIZE];
 43.
 44. void setup() {
 45.   Serial.begin(115200);
 46.   analogReadResolution(12);     // Due's ADC → 0…4095
 47.   analogWriteResolution(12);   // DAC → 0…4095
 48. //sine table DAC0
 49.   for (int i = 0; i < TABLE_SIZE; i++) {
 50.   float phase = (2 * PI * i) / TABLE_SIZE;
 51.   sineTable[i] = (uint16_t)(2047.5 + 2047.5 * sin(phase));
 52. }
 53.
 54.   myGLCD.InitLCD();
 55.   myGLCD.clrScr();
 56.   myGLCD.setBackColor(0,0,0);
 57.
 58.   myGLCD.setColor(0,255,0);
 59.   myGLCD.setColor(0,255,0);
 60.   myGLCD.drawLine(0, GRAPH_HEIGHT, GRAPH_WIDTH, GRAPH_HEIGHT); // X axis
 61.   myGLCD.drawLine(0, 0, 0, GRAPH_HEIGHT);                      // Y axis
 62.
 63.   // 2) Print static labels
 64.   myGLCD.setFont(SmallFont);
 65.   myGLCD.setBackColor(0,0,0);
 66.   myGLCD.setColor(255,255,0);
 67.   myGLCD.print("Min:", 10, GRAPH_HEIGHT+5);
 68.   myGLCD.print("Max:", 160, GRAPH_HEIGHT+5);
 69.   myGLCD.print("Thickness:", 10, GRAPH_HEIGHT+25);
 70.   myGLCD.print("By Kevin Lopez", 360, GRAPH_HEIGHT-170);
 71.   myGLCD.print(" John Urban", 360, GRAPH_HEIGHT-150);
 72.   myGLCD.print(" Derreck Suhul-Torres", 320, GRAPH_HEIGHT-130);
 73.   myGLCD.print(" Simon Maranga", 360, GRAPH_HEIGHT-110);
 74.   myGLCD.print("and Graham Werner", 340, GRAPH_HEIGHT-90);
 75. }
 76.
 77. void loop() {
 78.   // --- run a 5 s chirp (200 000 samples) into a rolling 4096-point buffer ---
 79. const uint32_t totalSamples = uint32_t(SAMPLE_FREQ * DURATION);  // 40000 Hz * 5 s = 200000
 80. float phase = 0.0f;
 81.
 82. for (uint32_t i = 0; i < totalSamples; i++) {
 83.   // compute instantaneous freq
 84.   float t    = float(i) / SAMPLE_FREQ;
 85.   float f    = F_START + (F_END - F_START) * (t / DURATION);
 86.   float dphi = 2 * PI * f / SAMPLE_FREQ;
 87.   phase    += dphi;
 88.   if (phase >= 2*PI) phase -= 2*PI;
 89.
 90.   // DAC output
 91.   uint16_t idx = (uint16_t)((phase / (2*PI)) * TABLE_SIZE) & (TABLE_SIZE - 1);
 92.   analogWrite(DAC0, sineTable[idx]);
 93.
 94.   // ADC sample and center
 95.   double raw = analogRead(A0) * (3.3 / 4095.0);
 96.   vReal[i % NUM_SAMPLES] = raw;
 97.   vImag[i % NUM_SAMPLES] = 0.0;
 98.
 99.   delayMicroseconds(1000000 / SAMPLE_FREQ);
100. }
101.
102. // --- envelope over full 200 k samples ---
103. float env = 0.0f, maxEnv = 0.0f;
104. uint32_t maxIdx = 0;
```

```
105. const float alpha = 0.02f;
106. for (uint32_t i = 0; i < totalSamples; i++) {
107.   float a   = fabs(vReal[i % NUM_SAMPLES]);
108.   env       = alpha * a + (1 - alpha) * env;
109.   if (env > maxEnv) {
110.     maxEnv = env;
111.     maxIdx = i;
112.   }
113. }
114.
115.   myGLCD.setColor(0,0,0); // background
116.   myGLCD.fillRect(0, 0, GRAPH_WIDTH-1, GRAPH_HEIGHT-1);
117.
118.   int centerY = GRAPH_HEIGHT/2;
119.   myGLCD.drawHLine(0, centerY, GRAPH_WIDTH);
120.   myGLCD.setColor(255,192,203);
121.   // --- plot as a continuous polyline ---
122.   int prevX = 0;
123.   // compute and constrain the first point
124.   double val0 = vReal[0];
125.   int y0 = centerY - (int)((val0/1.65)*(GRAPH_HEIGHT/2));
126.   y0 = constrain(y0, 0, GRAPH_HEIGHT);
127.
128.   for (int x = 1; x <= GRAPH_WIDTH; x++) {
129.     // pick the sample corresponding to this x
130.     int idx = (long)x * NUM_SAMPLES / (GRAPH_WIDTH + 1);
131.     double val = vReal[idx];
132.     int y = centerY - (int)((val/1.65)*(GRAPH_HEIGHT/2));
133.     y = constrain(y, 0, GRAPH_HEIGHT);
134.
135.     // draw a line from the previous point
136.     myGLCD.drawLine(prevX, y0, x, y);
137.
138.     // advance
139.     prevX = x;
140.     y0 = y;
141.   }
142.
143.   // optional: show min/max voltage
144.   double minV = vReal[0], maxV = vReal[0];
145.   for (int i = 1; i < NUM_SAMPLES; i++) {
146.     minV = min(minV, vReal[i]);
147.     maxV = max(maxV, vReal[i]);
148.   }
149.
150.
151.    int peakIndex = 1;
152.    double maxVal = vReal[1];
153.    for (int i = 2; i < NUM_SAMPLES / 2; i++) {
154.      if (vReal[i] > maxVal) {
155.     //Serial.print(vReal[i]);
156.     maxVal = vReal[i];
157.     //Serial.print(" ");
158.     //Serial.print(maxVal);
159.     peakIndex = i;
160.   }
161.  }
162.
163.   FFT.windowing(FFTWindow::Hamming, FFTDirection::Forward); // Apply Hamming window
164.   FFT.compute(FFTDirection::Forward);                       // Compute FFT
165.   FFT.complexToMagnitude();                                 // Convert complex numbers to
magnitude spectrum*/
166.
167.
168.   char buf[32];
```

```
169.    snprintf(buf, sizeof(buf), "%6.3f V", minV);
170.    myGLCD.setFont(SmallFont);
171.    myGLCD.setColor(255,255,0);
172.    myGLCD.print(buf, 50, GRAPH_HEIGHT + 5);
173.    snprintf(buf, sizeof(buf), "%6.3f V", maxV);
174.
175.     myGLCD.print(buf, 210, GRAPH_HEIGHT + 5);
176.    float tPeak = maxIdx / (float)SAMPLE_FREQ;
177.    float fRes  = F_START + (F_END - F_START)*(tPeak / DURATION);
178.    Serial.println(fRes);
179.    Serial.println(vReal[peakIndex]);
180. // calibration: f = 14049*x + 3618.5  =>  x = (f - 3618.5)/14049
181.    float thickness = ((fRes+2000) - 3618.5f) / -(14049.0f);
182.    myGLCD.setFont(SmallFont);
183.    myGLCD.setColor(255, 255, 0);
184.
185.    snprintf(buf, sizeof(buf), "%5.4f mm", thickness);
186.     myGLCD.print(buf, 100, GRAPH_HEIGHT+25);
187.
188.    delay(50);
189. }
190.
```