FSAE Strain Gauge Amplifiers

Ernesto A. Esparza, Electrical Engineering

Mentor: Dr. Phyllis R. Nelson

Kellogg Honors College Capstone Project

BACKGROUND

As a member of the Formula SAE club on campus we research, design and present our Formula style racecar at events around the world. To do this, we need to be able to gather a ton of data about our car with numerous sensors. One of these sensors are strain gauge sensors. Strain Gauges are thin copper films which are adhered to any metallic structural member of the car in order to read how much force that member is under. The direction of the force measured depends of the placement of the strain gauges. Strain gauges only change their output slightly in response to forces, so their output needs to be amplified.

OBJECTIVE

My objective is to research, design and create a simple and inexpensive amplifier circuit. This circuit should be able to realizably and accurately read data from strain gauges and convert the analog data into digital. The digital data can then be fed into our ECU for logging purposes. (Our cars ECU is a data logger and also the computer that controls the formula car.)

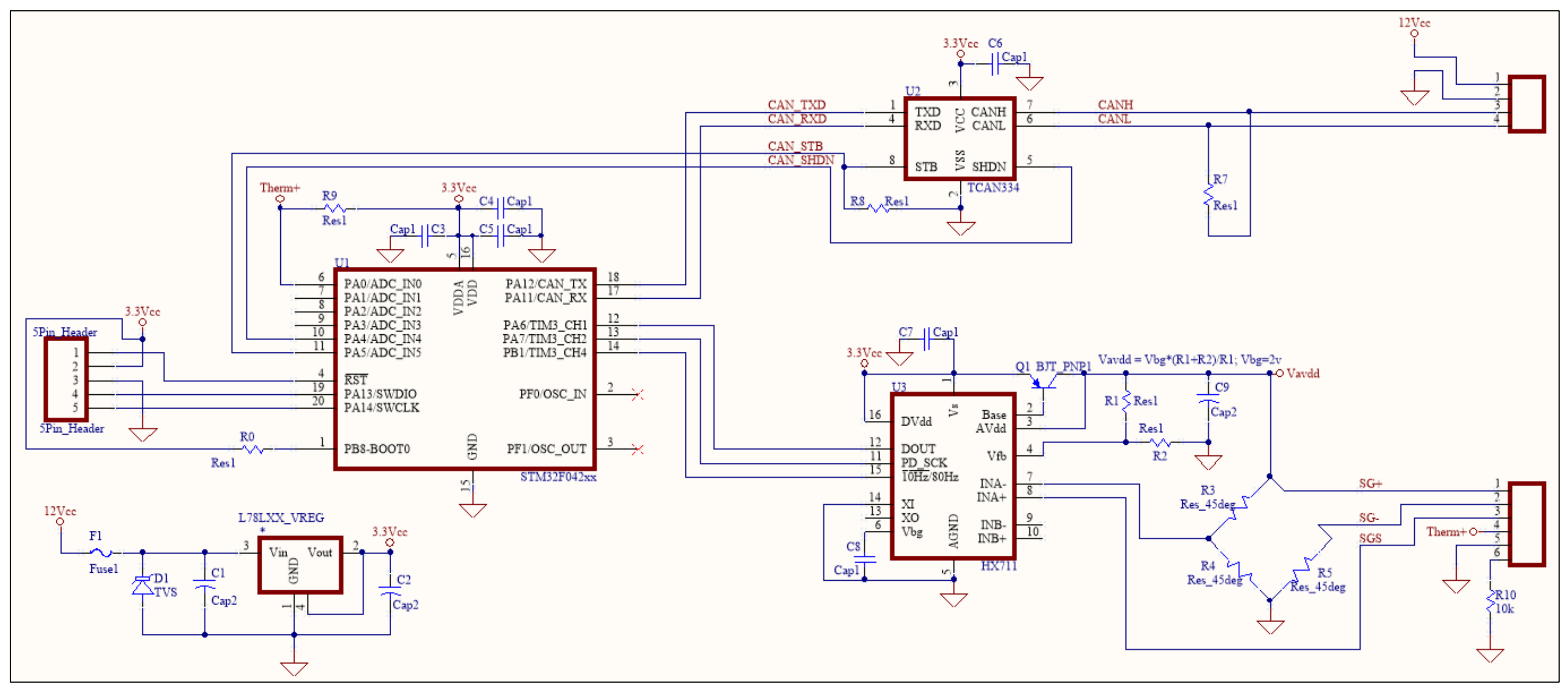
PREPARATION

In order to undertake this project I needed to quickly learn a lot of new skills. I had to learn to use an electrical CAD program (Altium Designer), create my own circuit, create my own PCB, source parts from retailers, solder tiny surface mount components, write C code for an arm based processor and also apply strain gauges on materials correctly and inexpensively.

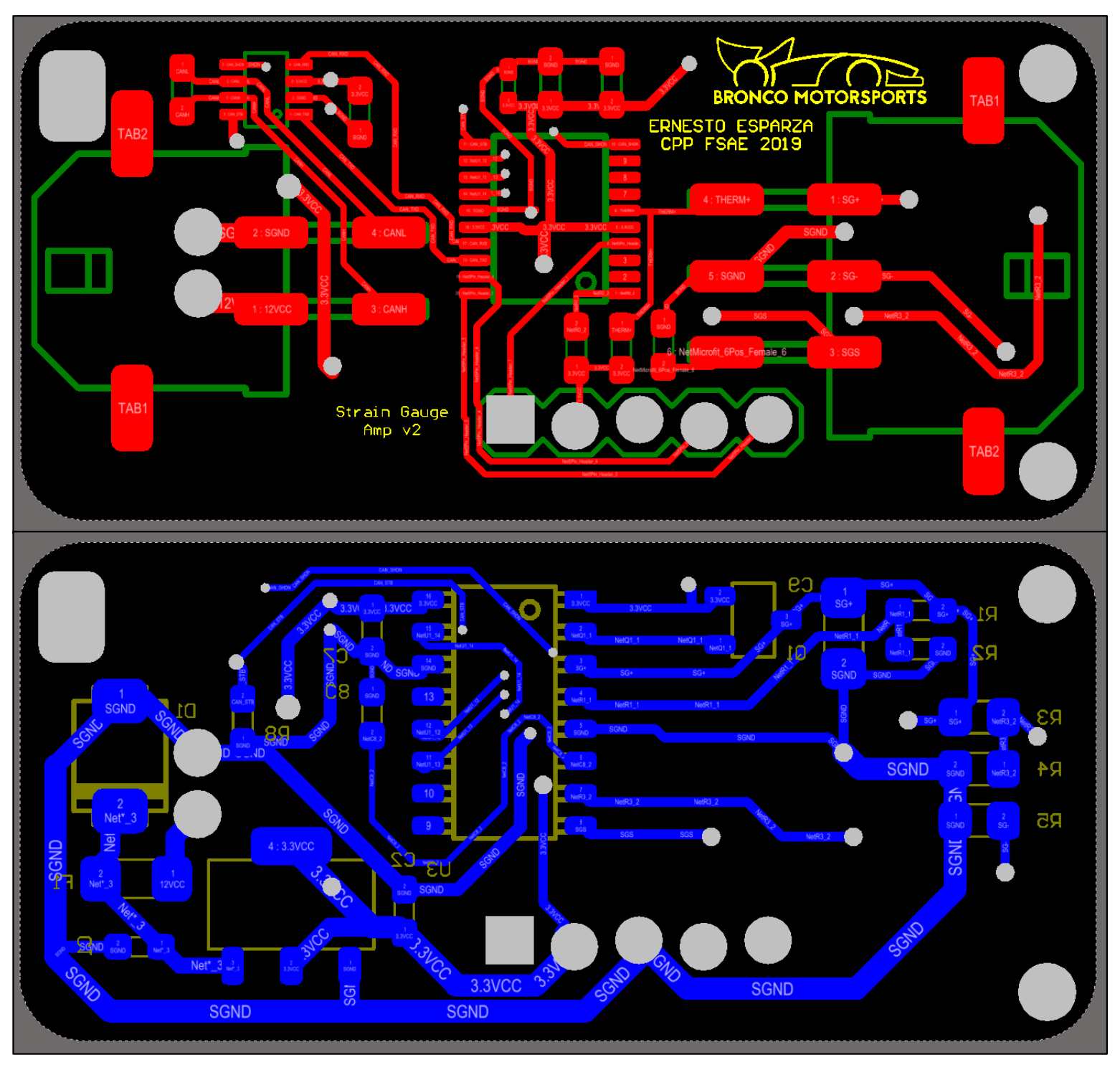
DESIGN (PCB)

Since I knew that the most important part of the design was the amplifier for the strain gauges, I began to search for that part first. Eventually I came across a relatively popular and inexpensive 24 bit amplifier and ADC, the HX711. This part was perfect since it has a lot of resolution (24-bits) and can read at rates up to 80 samples per second. Then I moved onto selecting an appropriate microcontroller. For a microcontroller I needed one that could collect the data from the amplifier, scale it (to convert it to a force from a voltage), then output it to CAN (a digital data format that cars use). I had multiple options from STM, to PIC and even just a plain old Arduino UNO. However a friend of mine had experience with the STM32 line so I decided to go with those. I found a small one that had built-in CAN support and was relatively inexpensive. I settled on a STM32F042. With the main parts out of the way, I could search for all the supporting circuitry. I found an appropriate LDO voltage regulator, PTC resettable fuse. transient voltage suppression diode, a couple very accurate resistors for the wheatstone bridge and a CAN transceiver. The strain gauge is essentially a super thin 'stretchable' metal film. Therefore, to read a voltage off it we need to connect the strain gauge to a very accurate wheatstone bridge. The circuit and the bridge are pictured below.

The design of the printed circuit board was very straight forward. However, given that this was my very first PCB design, there was a couple small hiccups. The main goal in designing the PCB was to make is as compact as possible. This is because the smaller the board is, the easier it will be to tuck it in somewhere on the formula car. Therefore I decided to make a standard 2 layer PCB. I read a lot Of books on PCB design prior to attempting to design my PCB, but there was nothing too critical about my deign. There was no high frequency signals or high current devices. I began by grouping Everything into sections (ex: microcontroller, ADC, CAN Transceiver, Power Regulation). Then I wired everything up, taking care to minimize the amount of vias created (the holes that connect one layer of the board to the other). I ended up with a board that is about 1.7 inches by 0.8 inches. The photos below are 4 times actual size.



Strain Gauge Amplifier Circuit Schematic

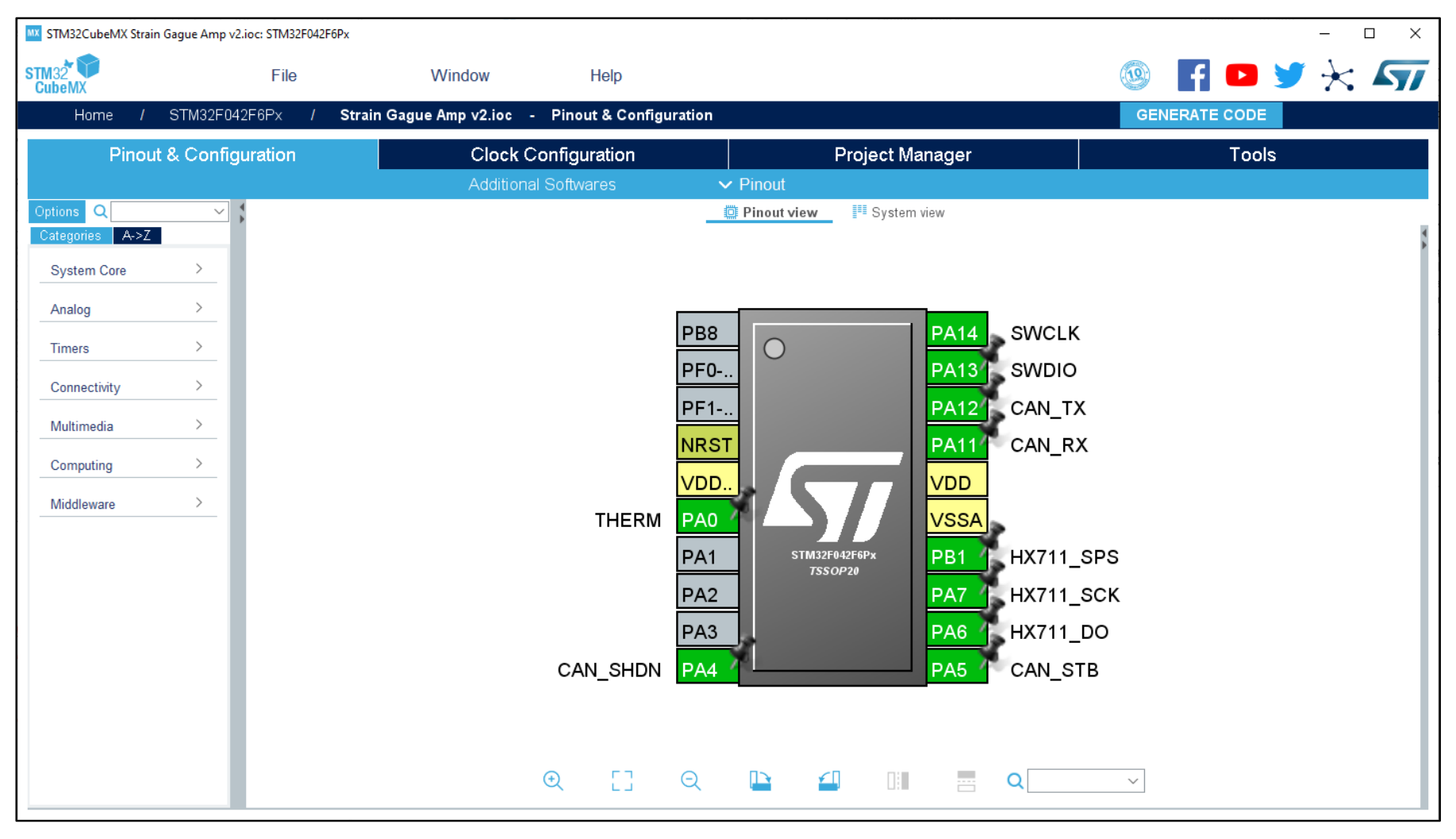


Strain Gauge Circuit PCB

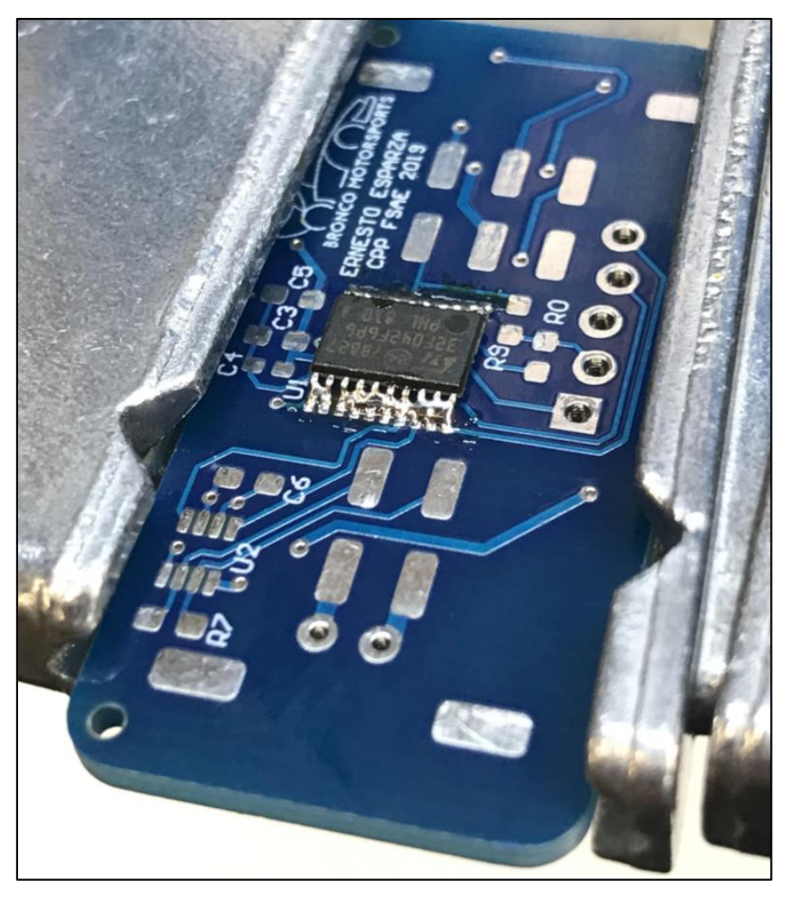
DESIGN (Program and Strain Gauge Application)

Once the board was complete, then I had to create a custom program to read data from the Amplifier/ADC and output it to CAN. I used a program called STM32CubeMX to simplify my programming work. This program sets up a base layer of code that initializes each pin of the microcontroller to whatever you want. From then I just used any C compiler to write the logic and upload it to the STM32 microcontroller. In order to ensure perfect function I used an oscilloscope to see every single bit being sent and verify that everything is correct. (The data transfer format and protocol is on the datasheet for the Amplifier/ADC I used, HX711).

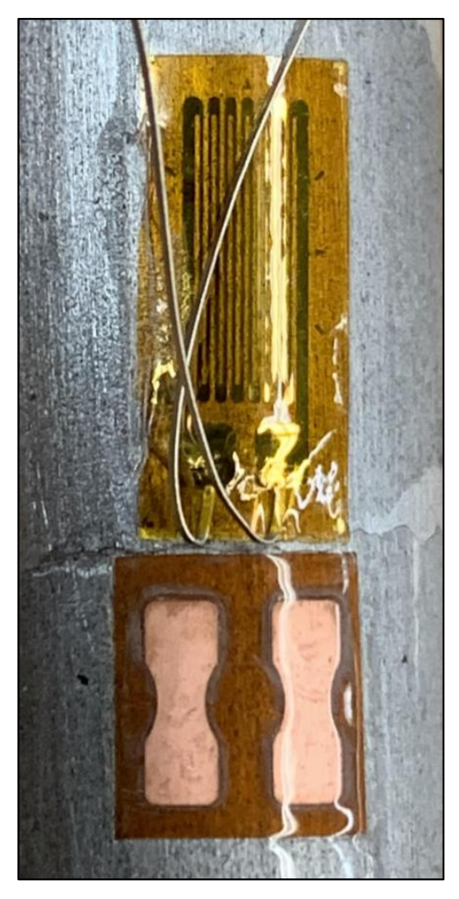
Finally, after all that designing I focused onto applying the thin, fragile, sensitive strain gauge films to different metallic surfaces. Most places I found use an array of hard to find and expensive chemical solutions to prepare the surface and adhere the strain gauge. Due to the steep cost I decided to forgo some measurement accuracy in favor of a much more inexpensive solution. My steps are as follows (for a steel surface). First, clean surface thoroughly with acetone or a degreaser if heavily soiled. Then sand with 120 grit sandpaper. After that, wet sand with 300 grit sandpaper (sanding is Mostly to create a relatively flat surface with a bit of scratches to help adhesion). Then clean again with acetone (acetone is great because it is neutral on the pH scale so it will not harm the strain Gauge). Tape the strain gauge to the steel and fold it back. Apply a small amount of cyanoacrylate glue catalyst to the back of the gauge, let air dry. Apply a small thin line of cyanoacrylate glue along the edge of the gauge and the steel. Fold the gauge over and spread out the glue under it into a thin even layer. Apply pressure with your thumb for 2 minutes, until the glue dries.



STM32CubeMX Program Interface



Strain Gauge Amplifier PCB Soldering



Applied Strain Gauge Film

RESULTS AND NEXT STEPS

Since I was not able to fully test my circuit due to the school closing and losing access to the laboratories, I do not have a lot of data. From the small amount of data I was able to generate, the readings seem to be too noisy to be useful. When I place a weight onto my testing rig and remove it, the readings do not return back to zero. I have discussed this with others, and I believe that this is because I only used a quarter bridge design (only one strain gauge and three fixed resistors). I should have used a half bridge design where there are two strain gauges and two fixed resistors. For future versions I would use a half bridge design, maybe use a better ADC that can read faster, implement a zeroing potentiometer (make one of the fixed wheatstone bridge resistors into a potentiometer), and deign the PCB so that is single sided. A 2-sided PCB is great for minimizing the size of the board, but it makes the board much more complex than needed. A single sided board would only be a bit larger, but it would simplify the process of soldering it together and finding a place for it on the car (because all the fragile components are only on one side).