Tactio: A Modular Tactile Sensor Development Suite
Final Report

ECE4873 Senior Design Project

Team Touché
Advisor- Dr. Patricio Vela

Austin Keener, EE, akeener@gatech.edu
Joseph Spall, EE, jspall3@gatech.edu
Joshua Oldenburg, CMPE, joldenburg@gatech.edu
Juan Elizondo-Villasis, EE, jelizondo7@gatech.edu
Varun Madabushi, EE, vmadabushi3@gatech.edu

Submitted

December 7, 2020

Table of Contents

Executive Summary

When interacting with the world around us, humans rely heavily on their tactile senses to understand the weight, shape, and texture of the objects they hold. Robots similarly require additional sensors to understand contact forces and thus “feel” their environment.

Robotic grippers can use tactile cues to adjust their grip on objects, and locomotion systems can use force measurements to understand contact interactions with the ground. Existing tactile solutions are effective but face a number of drawbacks, such as using many wires, having large footprints relative to sensor area, or requiring complex manufacturing methods. These factors present a barrier to widespread adoption in the applications research and hobbyist spaces.

Our product, Tactio, will be composed of a series of affordable, modular tactile sensors. Each module will be capable of reporting the location and magnitude of contact forces within their sensing area using piezoresistive material. A key feature of our product is the ability to “tile” modules by placing them next to one another and wiring them up in a daisy-chain fashion, allowing use in whatever shape they wish. Key technical performance specifications include a pressure sensitivity of 2 to 240 KPa with a quantization step size of 0.806μVolts and at a special resolution 1.78 points/cm^2. Each sensor node will cost approximately \$15.58 at a volume of 5000 units at a sales margin of 30%, with the development price of \$30 per sensor at low volume.

The proof of concept will be 16 individual nodes chained together and mounted to various geometries that would interface with easy-to-use software for integrating this sensor into the end-user’s application software for visualization, processing, or control. Future work for the project would involve attaching the sensor to a robotics manipulator in order to inform a control algorithm with sensor readings.

Tactio: Modular Tactile Sensor Development Suite

1) Project Summary

This section will cover a summary of the Tactio project. We will cover the electrical, mechanical, software, firmware, testing methods, and modeling results.

a) Electrical

We set out to develop sensor units that could independently measure the force applied at multiple points on their surface. Additionally, the units would be tileable and chainable electrically so all values can be read off of a shared communication bus by a single host computer or microcontroller.

There are a few ways to solve the sensing functions. Sensing a force electronically has two main methods: resistive touch and capacitive touch. Resistive touch relies on the changing of resistance of a surface contact point; capacitive touch similarly relies on the changing of capacitance of a surface at a contact point. Based on specified interface requirements (primarily needing to interact with most materials and ease of manufacturing), a resistive touch sensor was settled on for the main sensing foundation. After experimenting with Velostat and EX-Static with various force and position measurements, Velostat was determined to be a better correlation between resistance change and force amount placed. The design consists of a rigid PCB, the Velostat, and a flexible PCB. The rigid PCB and flexible PCB have row patterns at 90deg rotational offsets, forming a grid system. The system measures resistance magnitude across the rows and grids correlate to an approximate location on the sensor surface, with a change in magnitude of the resistance change indicating force amounts.

More specifically, the sensing circuit is made-up of voltage dividers, where the Velostat is treated as the high-side resistor. In between the Velostat and the pull-down resistor of the column, an ADC takes the voltage at the grid point, allowing for the direct calculation of the Velostat's resistance. By sequentially switching the rows on and off with high-side mosfets, their respective resistance values can be isolated. The simplified circuit of a row-column pair can be seen below:

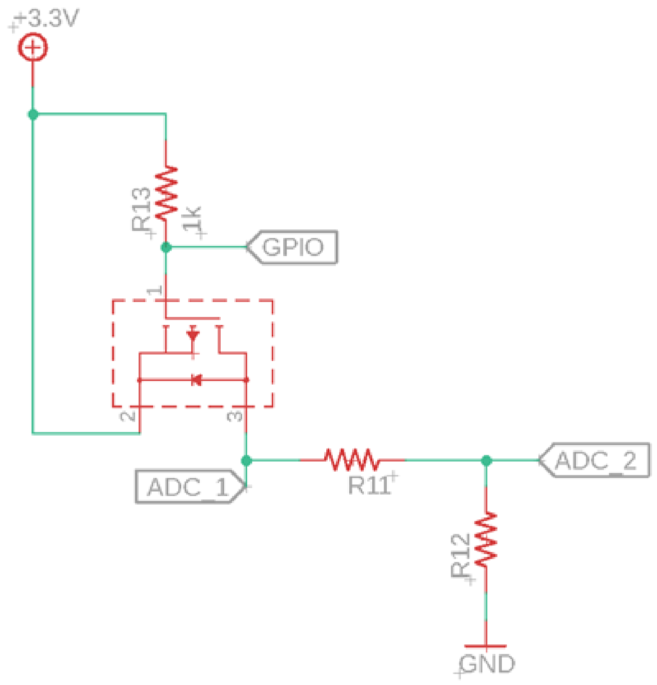


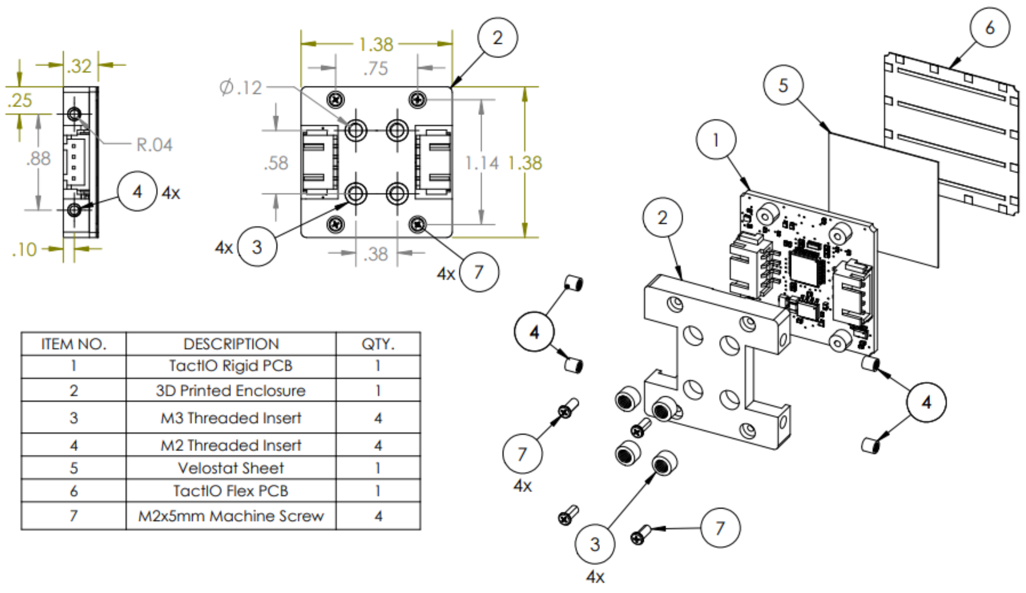
Figure 1. Simplified resistance measurement circuit.

For the communications solution we looked into some protocols for addressing communication schemes that are often used by microcontrollers, including I2C, CAN bus, and Ethernet. Some important aspects include high through-put, ease of implementation, minimal hardware infrastructure, commonality for implementation in embedded systems, flexibility for the sensor's specific use-case, and tolerance over long wires. CAN bus has all necessary aspects with the significant tradeoff of needing additional hardware for addressing and communication transceivers. However, this was the final choice.

To bring it all together, we chose a microcontroller (STM32F0) that had CAN support, enough ADCs, was low cost, and had enough memory. We added the MCP2562 chip to connect sensors to the CAN network. And we added a voltage regulator taking an input of 5V and outputting 3.3V to separate the CAN power from the sensing power, along with a resettable fuse and a polarity protection mosfet. Additionally, we added indicator LEDs for debugging.The final schematic is located in Appendix 1.

b) Mechanical

The mechanical solution utilized aims to solve the design goals set forth of flexible mounting configurations of the sensing node as well as provide a simple assembly method with minimal number of parts. This solution utilizes a few key components to satisfy these design goals. First, the rigid PCB contains soldered-on threaded inserts, which allows for the attachment of the node enclosure using machine screws. This enclosure protects the underlying electronics on the sensor, and it also serves as the mechanism for mounting the sensor to a surface by incorporating threaded inserts set into the plastic. These threaded inserts can be used either for direct attachment to planar surfaces, or the sensor can be attached to a custom mounting bracket using the threaded inserts to mount the sensor to non-planar or irregular surfaces. This solution allows the sensors to be mounted to essentially any possible surface, which facilitates the ability to cover large surfaces with an array of smaller sensing nodes. The remaining components of the sensing stackup, mainly the Velostat sheet and the upper flex PCB, are secured to the rest of the node by soldering exposed copper pads on the upper PCB to matching pads on the lower PCB, which effectively sandwiches the Velostat sheet in between without requiring adhesives.



ITEM NO.	DESCRIPTION	QTY.
1	TactIO Rigid PCB	1
2	3D Printed Enclosure	1
3	M3 Threaded Insert	4
4	M2 Threaded Insert	4
5	Velostat Sheet	1
6	TactIO Flex PCB	1
7	M2x5mm Machine Screw	4

Figure 2. Tactio Node Assembly

c) Software

The user-facing aspects of the Tactio system required the development of custom software. The software subsystem was necessary to provide a bridge between low-level aspects of the Tactio chain (sensor commands, raw data) and a user-written application running on a PC or other embedded system. In addition, it provides an easy PC interface to visualize and command the sensor chain.

Bridging was accomplished through code running on the Network Controller unit. This unit, consisting of an MBed microcontroller and a CAN Transceiver module, can send or read commands from any of the sensors in the chain. We used this, along with the microcontroller's built-in UART Transceiver, to convert messages from CAN to UART and vice versa. The UART communication system is documented and open-sourced so users can easily write Tactio libraries for their own embedded applications.

To interface a PC with the Tactio chain, we utilize the USB-UART converter built into the MBed microcontroller. This allows the same UART protocol to be sent over USB to a user computer. A Python library was developed to wrap the required sequences of serial bytes into convenient function calls – the resulting code can now be imported into any user libraries for easy interfacing with Tactio.

Finally, we developed one such user application – a Tactio control center and visualization software. This software can project the data from a chain of any length onto the screen and visualizes it as a series of color-changing squares. Applying pressure to a certain point on a sensor causes the corresponding square in the visualization to change color. To adapt to any user configuration, the sensor arrangement can be specified in an XML file. This allows the user to arrange visual elements to convey the sensor data in whatever way they deem fit. The application also allows the user to send commands to individual sensors, such as “Run Calibration Sequence” or “Flash Onboard LED.”

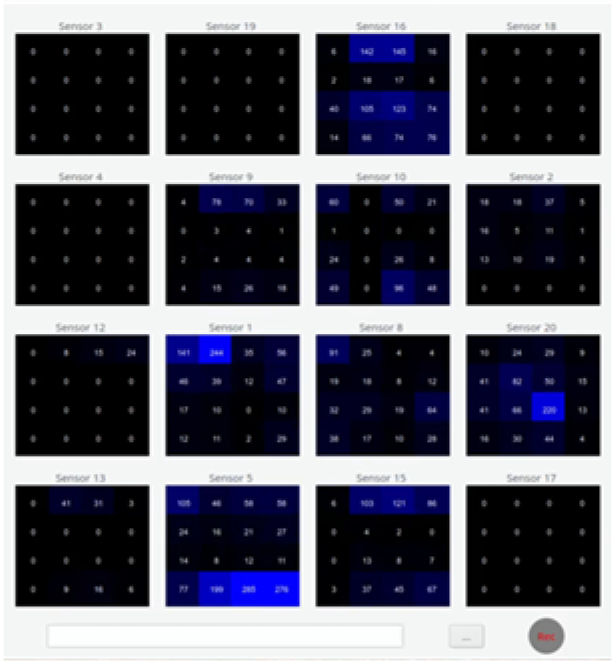


Figure 3. A view of the visualization software showing a 4x4 array of sensor units

d) Firmware

There are two devices in the project with separate firmware: the network controller and the nodes. There were many nodes in the network, but all of them ran identical firmware. All communication between devices happens over a standard, shared CANbus network.

The nodes handle all data processing, including reading the values from the ADCs, bias calibration to remove any pressure offset introduced by manufacturing variation, and formatting the data to be sent to the end-user software. The nodes also support various commands useful for debugging, like turning on and off the LEDs. This firmware is written using the STM32Cube hardware abstraction layer, provided by the manufacturer of the microcontroller, ST Microelectronics. The node's main firmware loop is spent by continuously reading data from the ADCs and running it through the bias calibration process. When a message is received on the CAN bus, it is handled entirely in the interrupt service routine. This entirely asynchronous flow keeps the code very performant.

The network controller firmware runs four threads to handle its four main tasks: it constantly requests updated sensor data from the nodes, handles messages received on the CAN bus, sends the sensor data to the attached computer, and handles messages received from the computer. Each of those threads happen roughly in parallel using the mbed real time operating system. The firmware uses the mbed hardware abstraction layer so it can be easily ported to a wide array of microcontrollers. This will allow us to easily switch to a custom PCB for the controller if we choose to in the future.

When the program starts, the controller cannot do anything as nodes have not been discovered yet. The controller receives self-identification messages from the nodes and adds them to its internal list of nodes. It also sends the node a command to set its node address by matching the node's unique device identifier with a preconfigured node address. As nodes get added to the network they are dynamically added to the array, regardless of the state of the program. Therefore, nodes can be added to the network at any point.

The controller continually requests sensor data from each node in the network in sequence, with delays built in to give the nodes time to reply. It refreshes its internal copy of the data when the reply is received from the node. A separate thread continually sends updated data to the PC. Both of those threads run on a compile-time configurable delay. A final thread manages commands received from the PC that are passed on to the nodes or handled internally to the controller. The PC communications protocol is fully documented publicly. The PC could be replaced by a standard 3.3V UART with a simple compile-time configuration change.

e) Testing Methods

Our testing methods for the project typically revolved around physical prototypes of designs. The main categories include sensor prototyping, circuit prototyping, and weight / ADC data collection.

Sensor prototyping focused on informing feasibility of building the sensor in the first place. We made two initial prototypes as scene in Figure 4: a plastic flex set PCB and an etched set PCB.

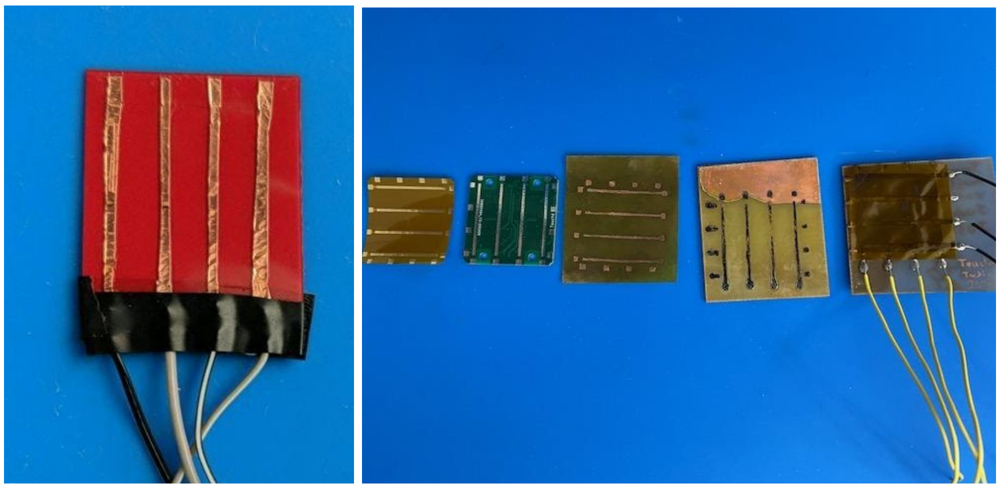


Figure 4. On the left is the initial flex PCB made of plastic and on the right manually etched rigid and flex PCB using copper plates.

The first prototype sensor was made of two flexible sheets of plastic, copper tape in rows / columns, and a piece of Velostat sandwiched between. The copper tape was cut to the planned size of the produced PCBs. This configuration was also used to verify the EX-STATIC material, which proved ineffective for pressure magnitude readings. We then pressed on the sensors with weights, our hands, etc. to measure the response. This configuration produced a significant change in ADC values, indicating a very positive indicator for the initial design. Retrospectively, both sheets being flexible allowed for greater strain on the Velostat material, leading to a much greater response then was seen on further prototypes and the initial final design.

The second prototype sensor was made of a copper manually etched FR4 rigid board, copper manually etched SF305 flexible board, and Velostat sandwiched again. Again, the width of the rows and columns were fitting the soon to be manufactured version. As mentioned, there was a much lower response to the pressure on the ADC values, but this was at the time blamed on the very poor manufacturing quality of the etching. Because of the rudimentary process, many of the traces where not continuous due to copper not designed to be etched away being etched.

The final prototype sensor was the manufactured sensor with a professionally manufactured rigid PCB and flexible PCB from our designs and Velostat sandwiched between once again. The initial version of the board saw a similarly poor magnitude response as the second prototype. Because of the professional manufacturing no longer being a variable, it was determined it was most likely the grid width of the conductive elements relative to the non-conductive elements. This is most likely due to the changing resistance of the Velostat being so localized, which serves great for the localization precision capabilities but makes the deadzones incredibly large. This was remedied with an update to the final prototype: the addition of 1/8th inch wide copper tape. This copper tape had a conductive adhesive which allowed for application directly on top of the existing pads to effectively increase the surface area of the rows and columns. This did resolve the pressure magnitude problem, but it also added another significant variance attribute for the sensor assembly.

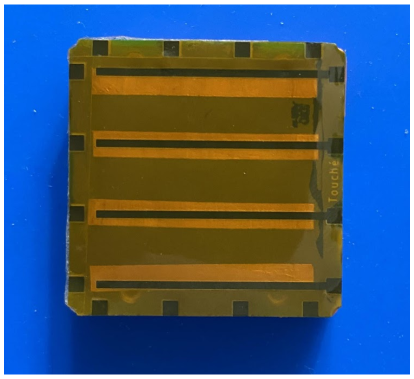


Figure 5. Example assembled sensor with copper tape visible through the flex material.

The next main testing procedure involved the circuit prototyping. This involved representing the microcontroller, sensor driving circuit, and CAN Bus circuit prior to final PCB manufacturing. The microcontroller was represented using the Nucleo-F042K6, which uses the same chip as the final board design. This allowed us to design and verify firmware prior to receiving the finished board. The sensor driving circuit was represented using discrete MOSFET chips, 390Ω resistors, and the prototyped sensors version 1 and 2 as previously mentioned. This was connected to the Nucleos in order to verify the individual row activation, timing, and ADC measurements. The final circuit component prototyped was the CAN Bus using the MCP2551 CAN Transceiver breakout board, 120Ω resistors terminating resistors, and the same Nucleos. This allowed for the Tactnet CAN library to be designed and verified, occurring at the same time as the sensor prototyping.

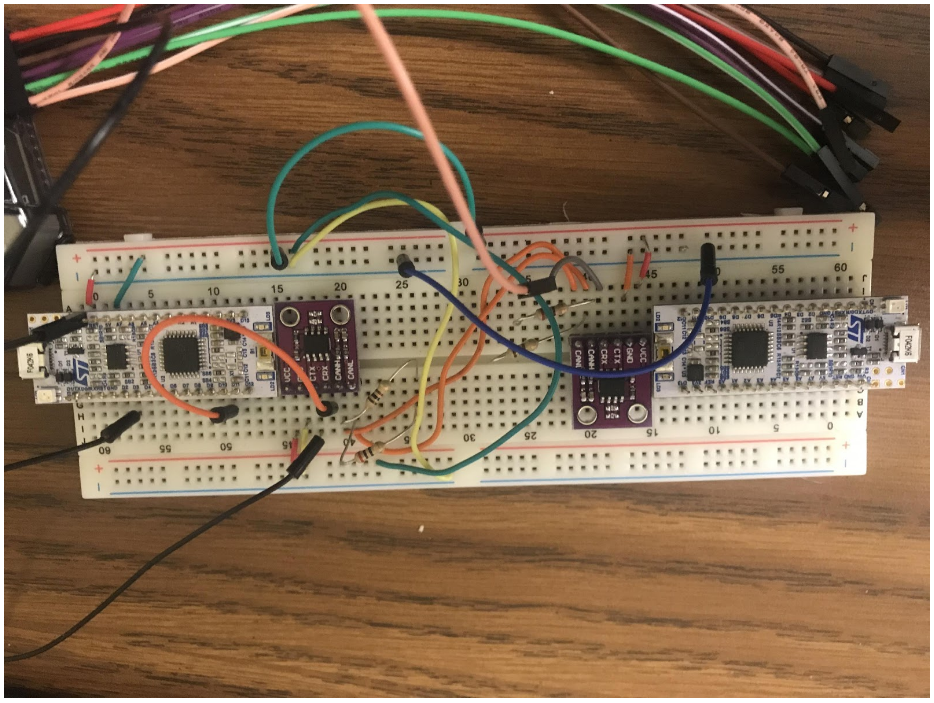


Figure 6. Prototyped CAN Bus circuit with MCP2551 breakout boards and Nucleos.

After this initial circuit testing, we had to verify the assembled circuits of the final sensors with the necessary components on the manufactured PCBs and sensing mechanism. As each of the sensors was hand-assembled, they all had to be individually verified for soldering issues. We started with verifying the power systems of the voltage regulator and reverse polarity protection, which worked successfully. The only issue was the incorrect routing the power LED, which did not impede the use of the circuit. Next was microcontroller assembly verification, which was tested using code verified on the test circuits. As a result, only one out of the batch of 20 sensors had a critical bricking of the microcontroller, with other remaining issues requiring simply reflowing the solder of a few boards. The final component of sensor verification was the sensing itself, which involved stepping through the firmware program to determine faulty rows and columns. It was determined that the driving transistors often had to be reflowed due to the very small nature of the part. This is also where it was noticed that the significant heating of the Velostat would cause a drop in pressure response performance. Later in usage stage, it was discovered CAN Bus errors would occur for certain boards consistently when attached to multiple other sensors in the chain. However, the source of the CAN Bus errors for specific boards was never identified or corrected given only a few of the boards had the problem and only when chained with multiple other boards.

The final testing component was the weight / ADC data collection. The purpose of this testing was to try to find a relationship between the pressure amount and ADC values being returned by the sensor. The actual modeling itself is covered more in the next section, Modeling Results.

The data collection method involved using a jig, as seen in Figure 7, to place specific pressure amounts on specific points of the sensor. This was accomplished by using standard weights, with the assumption of equal distribution of pressure on all points being pressed by the weight. The jig was designed with the points to be able to be configured in quantity and location. For the final dataset, the four center points of the 4x4 grid were activated across a 3 second period. The jig was then rotated 90 degrees and repeated with the same weight value 4 times to cover all possible configurations to account for asymmetry in the jig, which was apparent from initial samples.

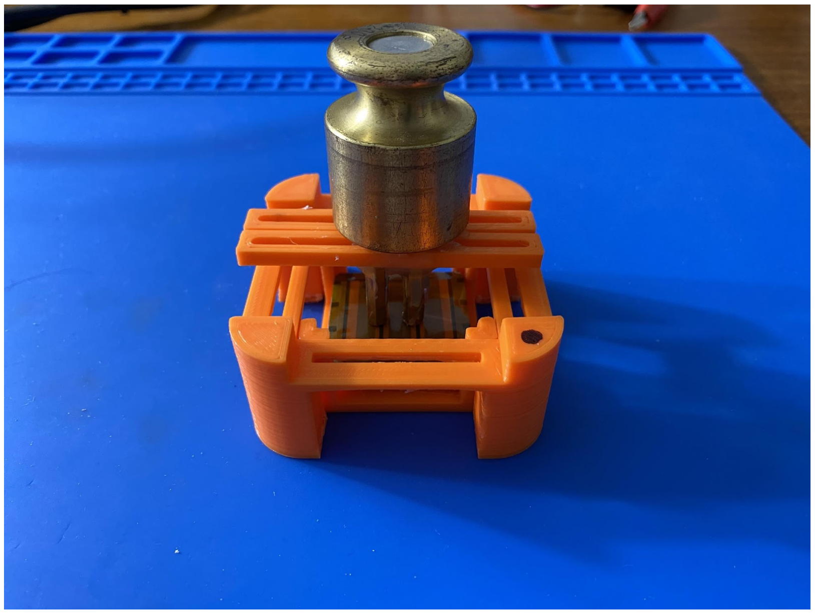


Figure 7. An example of a data collection configuration using the jig and weight

The weight samples were then ranged from around 80 grams (the weight of just the jig) in 50 gram increments up until 500 grams, transitioning over to 100 gram increments for the final duration up to 1kg. More samples were taken at the lower weights due to the presence of the higher dynamic range of the sensor at the lower weight regions. This was then repeated across multiple sensors.

Readings, in millivolts from the voltage divider and ADC, were collected using the record feature of the Tactio software package. Faults with the sensors, such as incredibly high values for no pressure or very low values for high pressure, were also discovered in this testing process and resolved if possible. One of the main sensors used, sensor 20, did appear to have a performance change in the sensing response after continued use. This is most likely due to sharp points of initial sensing methods permanently deforming the Velostat or flex PCB. As mentioned, the Modeling Results section will further explain the results of this sensor data collection.

f) Modeling Results

Our initial research had shown that we could expect a curve of the form $a(x+b)^c + d$, relating pressure to conductance, so we made all our models in terms of conductance, as opposed to resistance. At the onset, we also hoped the value of c would be close to 1, so that we could approximate the model with a linear model. As such we made individual linear models for every sensor. The results for a linear model for one of the 90 degree orientations can be seen below:

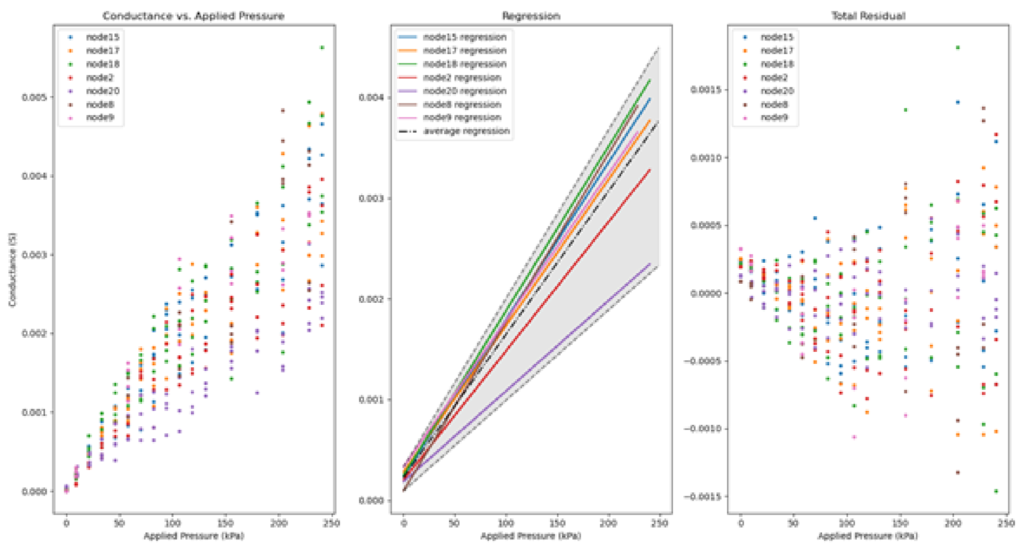


Figure 8. Scatter plot, regression lines, and residual plot for initial linear models.

Evidently, the sensors had well clustered values at the low-end of the pressure spectrum. As the pressure applied increased, the variance across sensors increased drastically. We attribute most of the variance seen to our manufacturing process, as we believe we did not have enough control over factors like the degree of bending the flex pcb had during assembly, or exactly how much adhesive we used in joining the sensor layers.

We did see several of the sensors having similar slopes, and most of them had very small offsets, which was promising at first. The residual plot said otherwise, however. First, we can clearly see the increase in variance in the residual plot. The other concerning matter is the size of these residuals. Some of them reach half the value of its corresponding data point, which would make such a model inadequate for predicting pressure values. Given these results, we decided to put the model building on hold until we could reduce the amount of variance in the system, and focused more on features we could show-off during the expo since we were starting to run short on time.

2) Technical Specifications

Electrical:

Supply Voltage	5V
Operating Voltage	3.3V
Maximum Rated Node Power Consumption	1.25W
Average Node Power Consumption	0.6W
Maximum Bus Supply Current	3A

Table 1. Electrical Specifications

Sensing:

Pressure Ranges	2 - 240 KPa
ADC Quantization Step Size	0.806 µVolts
Spatial Resolution	1.78 points/cm^2
Min Inter-node Sensing Null	10mm
Node Sensing Area	30mm x 30mm

Table 2. Sensing Specifications

Mechanical:

Sense Node Dimensions	40mm x 40mm x 15mm
Mounting Method	Interchangeable Brackets

Table 3. Mechanical Specifications

Interface:

Bus Communication Protocol	CAN 2.0
Maximum Polling Latency	10ms @ 1 node 100ms @ 10 nodes 200ms @ 20 nodes
Unique Node Addresses	255

Table 4. Interface Specifications

Firmware/Software:

Interface Library Language	C++, Python
Source Code Version Control	git
Source Code Distribution	GitHub

Table 5. Firmware and Software Specifications

3) Next Steps

a) Electrical

The electrical system had a few hiccups along the manufacturing process that we would like to address in a future revision. First, the power indicator LED was not connected properly and did not turn on when power was present, so we did not populate it in most boards. Second, the mosfets controlling the sensor rows were hard to solder due to their size and the shape of their pins. A slightly bigger package would be able to fit in the pcb and would prevent us from reflowing as many boards in the future. Next, the external oscillator was not used successfully. It is unclear if this is a firmware or hardware implementation problem, and once we pin it down, we will make the required changes accordingly. Additionally, to facilitate testing in the future, we will likely add test pads connected to a uart line of the mcu. Lastly, using copper tape to make bigger intersections in the grid was too time consuming. To allow for thicker traces in the pcbs, we will need to redesign the rigid board to have buried vias, as currently there is no way to route all the traces without interfering with thicker grid pads. Such a change will likely require a different attachment method for the flex pcb pads as well, but we have not discussed this as of yet.

b) Mechanical

The next steps for the mechanical aspects of Tactio will primarily consist of testing and manufacturing improvements. Various performance tests need to be conducted on the sensor nodes to determine the durability and reliability of the sensor as a whole, as well as upper limits on factors such as applied force. Additionally, environmental tolerances for factors such as vibration, temperature, and humidity need to be determined, tested, and then accounted for in the design if those tolerances are not met. To facilitate the use of the sensors across a wide array of environments, it would also be beneficial to modify the enclosure design to provide basic water and dust proofing. Additionally, the manufacturing process for the mechanical stackup of the sensors could be greatly improved by utilizing technologies such as injection molding for the enclosure.

c) Software

At the end of the semester, the Software system concluded with a working minimum viable product. The application was capable of reliably communicating with the sensors, recording and saving data, and sending commands. However, some areas could be improved. During the testing phase, certain UI bugs were encountered associated with Retina/4K screens. In addition, certain compatibility issues between MacOS/Windows/Linux should be resolved. Finally, some UI features and shortcuts could be improved for a better user experience.

An additional feature that would be particularly useful is a ROS library. Currently, a Python library is provided for communications between a PC and Tactio. Extending this functionality to ROS is a logical next step that further simplifies the integration of Tactio into industrial and research robotics, thus addressing the needs of our target audience in a more effective manner.

d) Firmware

The node firmware is very functional in its current state. The most significant change left to be made is introducing the sensing model to send pressure measurements to the computer rather than ADC values. Further optimization would help the performance of the main data processing loop, though it already performs well.

The network controller needs a moderate amount of work to ship as a product. Its most significant issue is an intermittent hard fault triggered by resetting the controller twice in quick succession. This issue does not occur in normal use as the device is not reset regularly; however, it entirely freezes the device and must be fixed. The network controller firmware also needs to use mutexes more extensively to protect against two threads accessing the same data simultaneously. There are no known major issues with that currently, but more protection could avoid the issues entirely. These are the major issues that need to be fixed in the immediate future. Further exposing configuration to the user would allow it to be integrated into a wider variety of applications.

e) Manufacturing

For this product to be viable, manufacturing methods of the sensor would have to have higher yield rates and much less variance in the performance of the sensor. The most straightforward improvement would be the automated assembly of the components, using a solder paste stencil, traditional pick-and-place method for parts, and reflow oven on the rigid PCB. This would need to be performed at a much higher volume of boards due to often higher initial cost to simply do any PCB part assembly. The boards would still need to be verified, but it would certainly reduce the number of assembly errors. If more compact chip packages that lack access to pins except under the chip, this assembly method would be a necessity.

However, the large cause of issues and the aspect that could not be automated using traditional pick-and-place methods would be the sensing method. As mentioned, a redesign of the flexible PCB and attachment method of the flex to the rigid PCB would certainly assist with this aspect. Using a different method could very feasibly resolve issues such as heating the Velostat during manufacturing/repair, reduce the rate of the flex PCB coming apart from the rigid PCB, facilitate better alignment of the PCBs along with the Velostat, and remove the use of adhesives that reduce performance on the edges of the sensor.

Past the benefits of the changing of the flex PCB mounting method, there is also a need for automating the mounting method. This would most likely involve using a jig for improved board alignment or even having the heating apparatuses needed for connecting the two boards as jig. This would certainly be more justified at a much higher volume production of the sensors, with a redesign of mounting possibly providing a sufficient increase in yield rate and reduction and variance at lower production volumes.

f) Project

The final aspect is the next steps with the Tactio project as a whole. We have seen great success with the project, not only hitting many of our initial design goals but also generating a very compelling product. As a group, we also work well together and have skill sets that are very complementary. If we were to continue with the project, there are a couple of avenues we have identified as possible next steps: a paper publication, licensed hardware, or a crowdfunded hardware. These next steps are not necessarily independent, but they serve as ways to continue with the project.

The paper publication would revolve around generating sufficient material to be able to submit a paper to a robotics or sensing based journal or conference. The licensed hardware would involve reaching out to maker focused board manufacturers and retailers such as Adafruit or Sparkfun with the intention of collaborating to have the sensor possibly produced and sold through them. The final route is a crowdfunded solution of selling the product after raising money through methods such as Kickstarter or Crowd Supply.

All of the next steps mentioned would certainly involve additional revisions of the technical aspects of Tactio. The methods to these revisions include working with a Georgia Tech research lab for publishing a paper and utilizing the Georgia Tech entrepreneurship resources such as Startup Launch and the Inventure Prize for the licensed and crowdfunded approach. There is also the option to self-fund development part time as more of a hobby. All members of the team will either be starting graduate studies or a full-time job in the coming months, so these ideas serve simply as current state musings more than definitive plans. Startup Launch has been applied for pending a decision, and the team will most likely participate in the Inventure Prize Semi-final round that was earned at Capstone Expo.

Appendix A