THE DESIGN, FABRICATION, AND TESTING OF LOW-POWER MEMS RELAYS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Leanna Alisa Pancoast

May 2020

ABSTRACT

This thesis covers the design, fabrication, and testing of low-power electrostatically actuated MEMS relays. These relays can be used to enable ultra-low power operation of a sensor node while waiting for a signal of interest, significantly increasing the battery life. Electrostatic MEMS relays are fabricated on silicon-on-insulator wafers using a one-mask process with tools available in the Cornell Nanofabrication Facility (CNF). Tests of the fabricated devices are then processed and automated using Python and Arduino technology to control various test equipment. The resulting MEMS relay is able to be actuated using an external sensor.

BIOGRAPHICAL SKETCH

Leanna Pancoast grew up in Houston, TX where she developed a deep appreciation for good BBQ. From Houston she was able to attend Carnegie Mellon University where she got a Bachelor's of Science in Electrical and Computer Engineering in 2015. After CMU she attended Cornell University to work on a Master's of Science in Electrical and Computer Engineering while working under Amit Lal on MEMS devices. In that process she worked at IBM 2018-2019 on healthcare devices and is currently at NYU Langone working on sensors with MRI team at the Center for Biomedical Imaging (CBI).

ACKNOWLEDGMENTS

I would like to thank and acknowledge Professor Amit Lal for supporting me through the process. Help from the members of the SonicMEMS lab, including but not exclusive to Ved Gund, Alex Ruyack, Visarute Pinrod, Benyamin Davaji. Justin Kuo for being my cubby mate and answering all my questions. Help from those in the Molnar lab including but not exclusive to Sunwoo Lee, Mel Voda, Robin Ying.

I acknowledge Defense Advanced Research Projects Agency (DARPA), NZERO program, for funding this work. This work was performed in part at the Cornell NanoScale Facility, a member of the National Nanotechnology Coordinated Infrastructure (NNCI), which is supported by the National Science Foundation (Grant ECCS-15420819). This work made use of the Cornell Center for Materials Research Shared Facilities which are supported through the NSF MRSEC program (DMR-1120296). This work made use of the Nanobiotechnology Center shared research facilities at Cornell.

TABLE OF CONTENTS

CHAPTER 1 Contents

LIST OF FIGURES

CHAPTER 1 MOTIVATION

Advancing technology has enabled the creation of small, low power, wireless sensor devices that can be deployed across wide areas to sense things for applications such as a building's structural integrity, monitoring a forest, defense of a perimeter, or healthcare [1]. Each of these networks consist of an array of sensor nodes that are deployed across a wide area. A schematic of a typical sensor node is shown in Figure 1 [2].

At its basics, a sensor node consists of a sensor to detect a signal of interest, some analog and digital processing, and a radio or another means to transmit the data. Some sensor nodes can potentially include a wake up trigger to wake up a mostly inactive resource. For example, we can look at one sensor that is common in most houses – a thermostat. For a thermostat, the signal of interest is the temperature of the house, the sensor is a thermometer, the logic or processing is determining if the temperature is

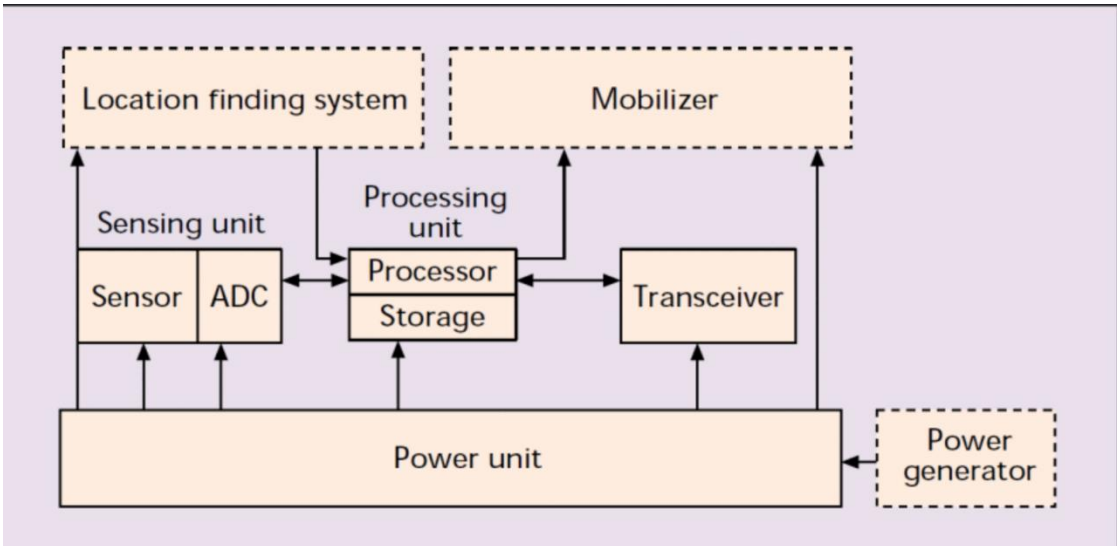


Figure 1 Schematic of a typical sensor node. From Akyildiz 2002

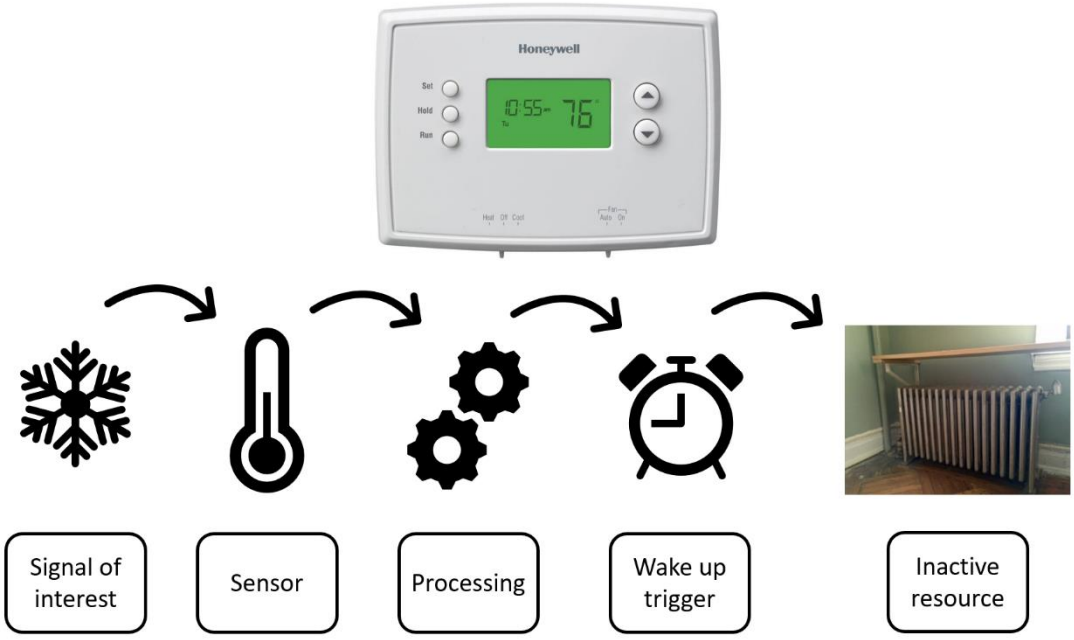


Figure 2**:** Diagram showing thermostat as a sensor node. The lower boxes show the differnt components of a general sensor node. The pictures above show the corresponding feature for a thermostat.

out of a pre-determined range, and a mostly inactive resource that is triggered is either air conditioning or a heater if the temperature goes too high or low respectively.

While a thermostat is in a house with steady access to a power supply, other sensor modules do not always have such power access. As wireless capabilities grow faster than small power modules, the need to extend battery life grows. For example, in order to monitor forest fires, one would need to deploy many of these sensor modules out into the forest. Each node will need some sort of power source, typically a battery, which has a limited life span. To limit the amount of time needed to replace the batteries, one can use energy harvesting, but that is not always reliable as the energy source, such as movement or sunlight, is not always available. A more reliable way to maximize battery life is to minimize the power that is consumed by the sensor node.

To determine the best section to optimize, we take a look at the graph showing the lifetime of a device versus the frequency of the signal of interest in Figure 3 [3]. The lifetime of a sensor module increases as the event frequency goes down, until the event happens less than 5% of the time. Below this frequency, the lifetime remains the same. In times of high frequency events of the signal of interest, power will be consumed by communication and processing of information. In times when the frequency of events of interest is low, power will be consumed by the electronics that are listening for the event of interest. By minimizing the power required to sense the signal of interest, the battery life of the sensor node can be maximized.

A further step to maximize battery life is to minimize the power required to talk back to the central station. While this is not super relevant in the example of forest

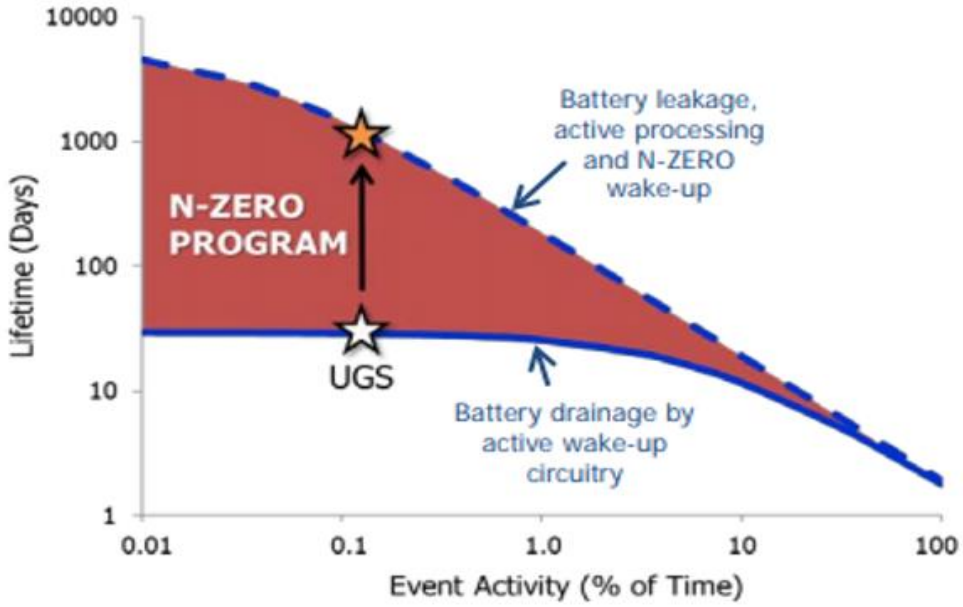


Figure 3: Graph showcasing the lifetime of sensors modules based on the percentage of time the desired activity to sense happens. The grant that funded this research (DARPA N-ZERO) aims to extend the battery life of an unattended ground sensor (UGS) by reducing the power consumed by a sensor module in the asleep but listening state.

fires as the signal of interest is destructive to the sensor, it can be relevant in perimeter defense.

If a military base wants to know if a car or truck is passing close, then they will have a lot of sensors surrounding the base. A mesh network communication protocol can be used to limit the power needed for communication. The sensors furthest away from the base will need to expend a lot of energy to communicate back to the base. This energy can be lessened by allowing the nodes to communicate to each other, allowing a signal to bunny hop through a sensor network back to base. The distance the signal needs to travel is much less, and thus a smaller signal is needed to communicate. Here the signal of interest is an RF signal, the sensor is an antenna, processing is to determine if the RF signal is from a sister node, and the passive component is a microcontroller to send more information to the next node.

The focus of this work will be on the wakeup trigger for the microcontroller. To minimize leakage current, micro relays will be fabricated to create a wakeup signal from information gathered by a piezoelectric sensor.

The work from this thesis partially lead to the publishing of the following papers:

1. Pinrod et al Zero-power sensors with near-zero-power wakeup switches for reliable sensor platforms

   The near zero-power sensor node solution is presented with piezoelectric sensors and DC tunable threshold electrostatic switches. A sensor suite measuring acceleration, rotation, and magnetic field based on PZT lateral bimorphs is used with NEMS switches for the detection of a desired signal

pattern and generating a wake up trigger. The sensors are capable of detecting physical signals from 5 Hz to 1.5 kHz, with sourcing load capacitances as high as 200 pF. The sensor sensitivities achieved are: 0.1 V/g for the accelerometer, 31 mV/Gauss for the magnetic field, and 0.31 mV/(°/s) for rotation. NEMS switches, with threshold voltages in the mV to 15V range, can combine multiple sensor outputs through multi-gate actuation to detect desired event specific features. Using the sensors, we demonstrate the detection of a portable electrical generator in its different operational modes (On/Off state and the Eco mode).

2. Ruyack et al NEMS Electrostatic Resonant Near-Zero Power Resistive Contact RF Wake-Up Switch with PT FIB Contact

This paper reports a NEMS electrostatic RF wake-up switch capable of detection sensitivity reaching -25 dBm off resonance with sub-pW passive power consumption at less than 1.5V DC operation. The switch utilizes a multielectrode design allowing for a mechanical contact gap to be held just outside thermal oscillations and RF input to physically close the switch. This allows for operation away from pull-in and improves RF sensitivity. A focused ion beam (FIB) patterned platinum-platinum contact point enables the low DC bias requirement and improves contact resistance and longevity. The potential for on-resonance operation provides a pathway forward for improved RF sensitivity in the future.

2.1 MEMS Actuation Mechanisms

There are four main actuation mechanisms for MEMS relays, as shown in Figure 4 [4], [5]. Piezoelectric actuation takes advantage of the properties of piezoelectric materials, such as Lead Zirconum Titanate (PZT) or Aluminum Nitride (AlN). Thermal actuation takes advantage of differing thermal expansion coefficients of different materials to move a beam. Magnetostatics takes advantage of the magnetic properties of different materials. Current running through a coil creates a magnetic field to deflect a magnetic material to one side or another. Electrostatic actuation takes advantage of the electrostatic force that is created between two charged plates to create movement.

Our applications require low power in both the active and passive state, so magnetostatic and thermal actuation are not considered. Piezoelectric actuation could

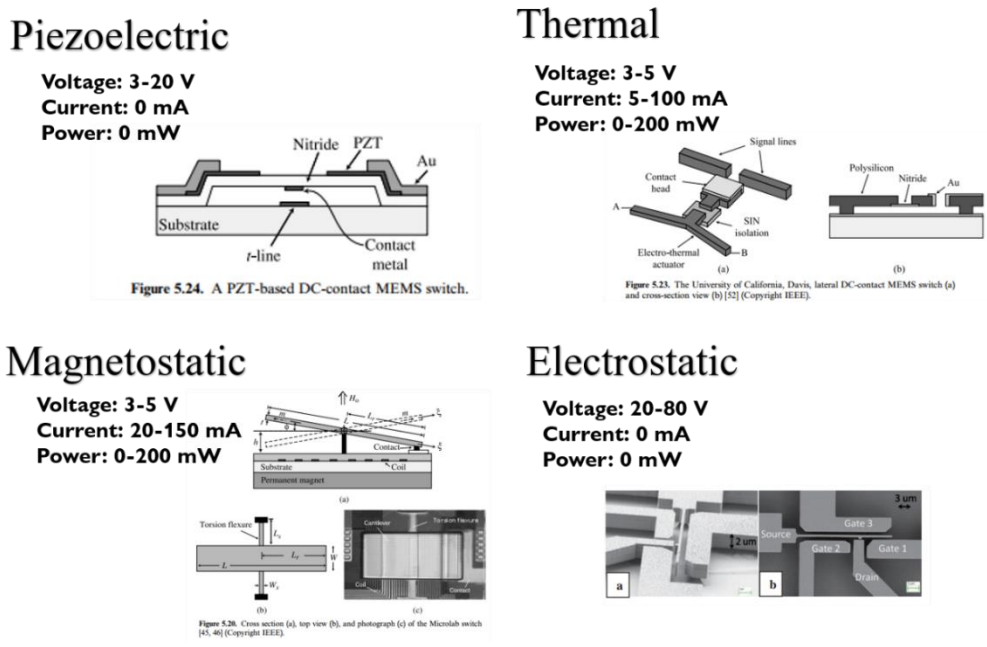


Figure 4: Different MEMS actuation mechanisms and the power requirements for each.

be used, but it is also the basis of the sensors that we want to trigger the switch, such as acceleration. Accelerations could move the piezoelectric switch in unintended ways and create false positives. Electrostatics allow for low power actuation in addition to low sensitivity to external events.

2.2 Electrostatic Actuation Mechanism

Electrostatic actuation has its base on the force between two charges, as given by $F_{es} = \frac{kq_q q_2}{r^2}$, where k is Coulomb's constant, $q_1$and q_2 are the magnitude of the charges, and r is the distance between them [6]. When the charges have the same polarity, there is a repulsive force, and when the charges have opposite polarities, there is an attractive force.

The force can thus be controlled across two metal plates by controlling the voltage that is between them. The basic model for an electrostatic switch is a parallel plate

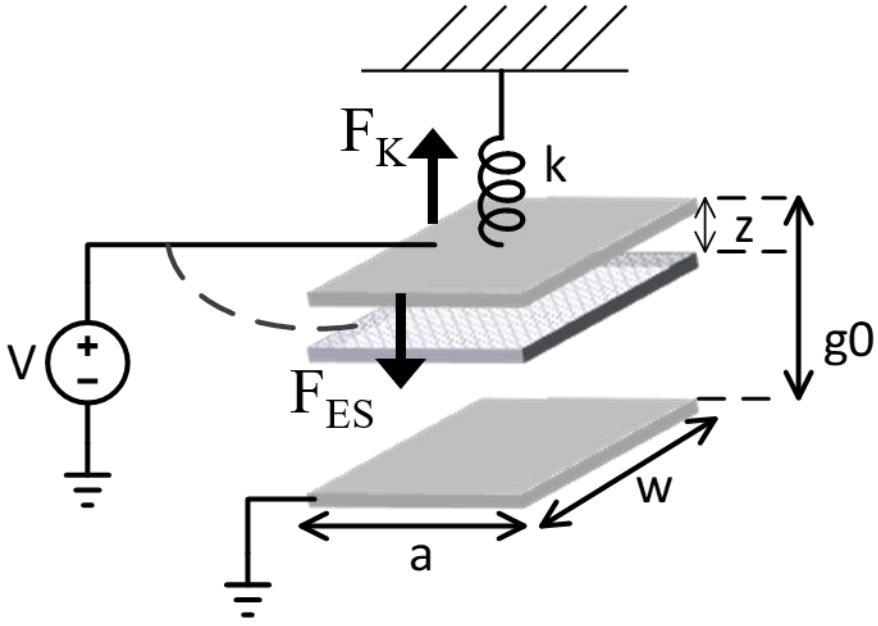


Figure 5: Basic model of an electrostatic MEMS device. A voltage applied between a conductive plate attached to a spring and a static conductive plate.

capacitor with a plate attached to a spring, as shown in Figure 5. The equations of interest with this model are the electrostatic force acting between the two plates at a given voltage, V, and the restoring force of the spring.

The electrostatic force between the two plates at given voltage V is given by $F_{es} = \frac{\epsilon_0 a w V^2}{2(g_0 - z)^2}$, where ϵ_0 is the permittivity of free space at $8.86 \cdot 10^{-15} \frac{F}{m}$, a is the length of one side of the plate, w is the width of the plate, V is the voltage between the two plates, g_0 is the initial gap between the two plates, and z is the displacement of the top plate caused by the electrostatic force between the two plates. The restoring force of the spring is given by $F_k = -kz$, where k is the spring constant and z is the displacement of the top plate.

As the voltage increases, the force attracting the plates together will move the top plate down. If the voltage stays the same, then there will be an equilibrium point where the electrostatic force balances the restoring spring force.

The thing to note here is that the electrostatic force will increase faster than the restoring force due to a quadratic dependence on distance moved versus a linear dependence for each corresponding force. There is an unstable point at $z = \frac{g_0}{3}$. A figure of metric for this type of device is the pull-in voltage, or the voltage at which the plate will reach the pull-in point. The pull-in voltage is given by $V_{PI} = \sqrt{\frac{8 k {g_0}^3}{\epsilon_0 a w}}$.

2.3 Three Terminal Device

The simple capacitor spring model is convenient for understanding how a device will move, but a third terminal is necessary to create an electrical relay that can be compared more as a MOSFET. The springs used in MEMS devices are typically

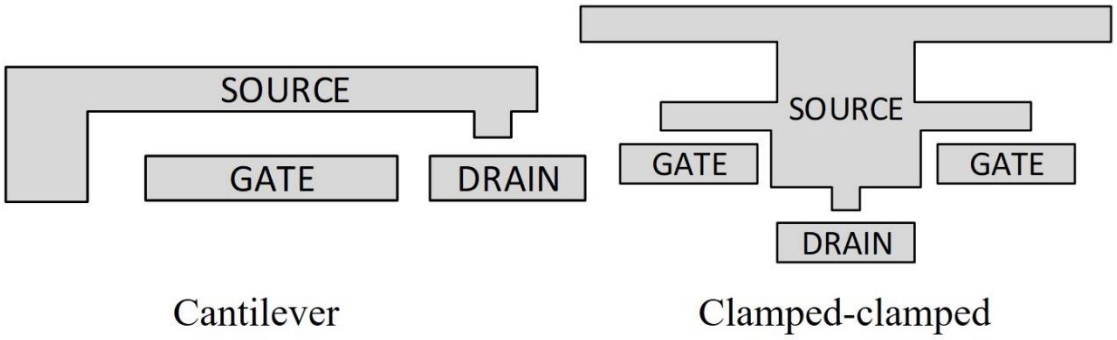


Figure 6 Examples of three terminal MEMS devices

simple mechanical beams, such as a cantilever or a clamped-clamped beam [7]. Two examples of a three terminal relay using these designs are shown in Figure 8. Effective spring constants based on the geometry of the beam and location of the electrostatic forces can be calculated as shown in Figure 7 for cantilever beams. One can look in Kam's Micro-Relay Technology for Energy-Efficient Integrated Circuits [7] for more information on the calculation of these spring constants and mechanical modeling of clamped-clamped, or fixed-fixed beams.

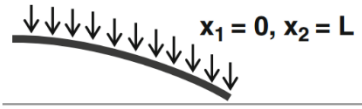
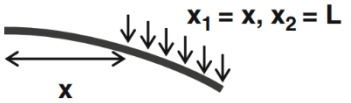
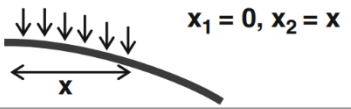
Force distribution	Spring constant
	$k_{eff} = \frac{3EI}{L^3} = \frac{3EWh^3}{4L^3}$
	$k_{eff} = \frac{8EI}{L^3} = \frac{2EWh^3}{3L^3}$
	$k_{eff} = \frac{24EI}{6xL^2+(L-x)^3+2L^3} = \frac{2EWh^3}{6xL^2+(L-x)^3-2L^3}$
	$k_{eff} = \frac{24EI}{x^2(4L-x)} = \frac{2EWh^3}{x^2(4L-x)}$

Figure 7 Different spring constants of cantilevered beams based on force distribution from [7]

2.4 Adding more terminals

High voltages are still required to fully close a device using three terminals. Amponsah *et al.* explored using more terminals to lower the required actuation voltage [5]. A bias is applied to one terminal to bring the device close to closing, and a much smaller voltage can be applied to another terminal to bring the device fully closed. More gates can be added to allow variable bias tuning while a device is out in the field, as shown in Figure 8. The design explored later uses this multiple gate design to lower the turn on voltage of a MEMS relay and enable a piezoelectric accelerometer to switch the MEMS relay to an ON position. The next chapter explores the fabrication process of these relays.

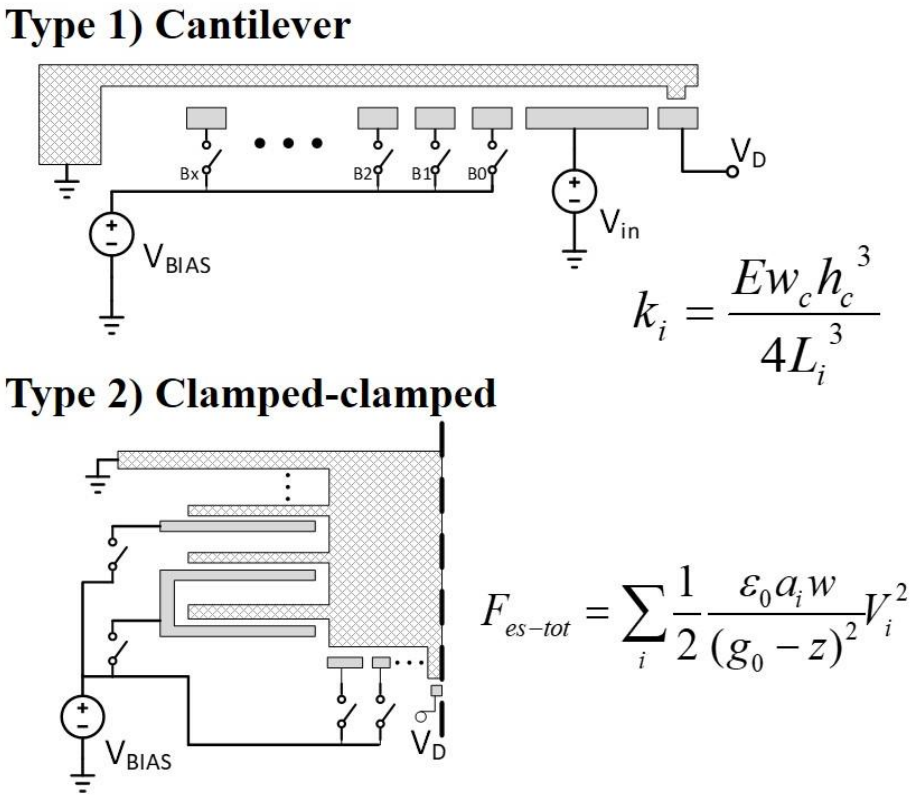


Figure 8: Multiple terminal options of MEMS relays.

CHAPTER 3 PROCESS DEVELOPMENT

3.1 Fabrication Overview

The fabrication of these devices use standard microfabrication techniques, done in the Cornell Nanofabrication Facility (CNF). At its very core, the process has 3 steps to create features- define, etch, and release starting with a silicon on insulator (SOI) wafer that is purchased from a vendor. This process is shown in Figure 9 and based on work previously done by Kwame Amponsah [5]. This SOI wafer consists of a Si device layer (2-5 μm), SiO_2 layer ($2-5\mu m$), and a thicker Si handle layer ($400-600\mu m$). Each step consists of many smaller steps using various tools and processes within the CNF, each of which had to be optimized for the NEMS switches. The following sections discuss an overview of each step.

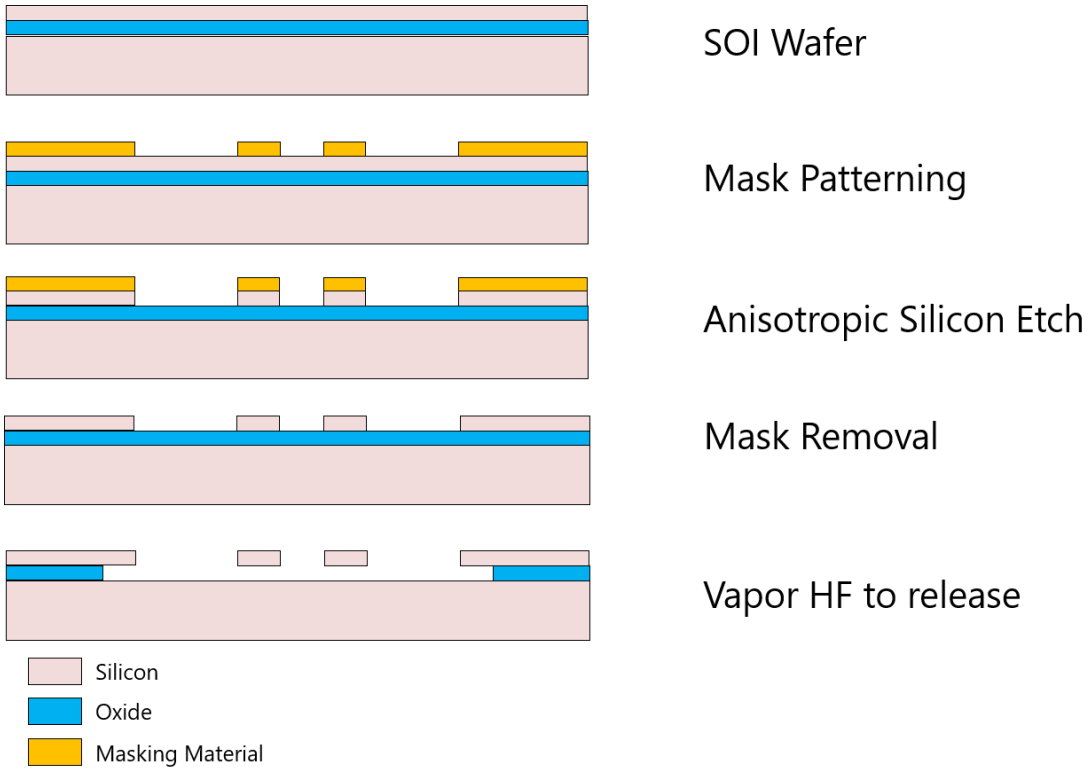


Figure 9: Generalized one mask process flow to create MEMS relays.

3.2 Mask Patterning

Photolithography is used to define the shapes of the devices to be made. A typical photolithography process consists of depositing a light-sensitive polymer onto the wafer, shining light through a mask, and then developing away the unreacted polymer[8]. Examples include contact, proximity, projection, or e-beam lithography. These all share a basic structure as shown in Figure 10. One can achieve different levels of feature precision by varying the type of light source or distance between each of the parts. For the feature sizes required by the design of the devices, deep UV projection lithography was performed using the ASML DUV stepper [9].

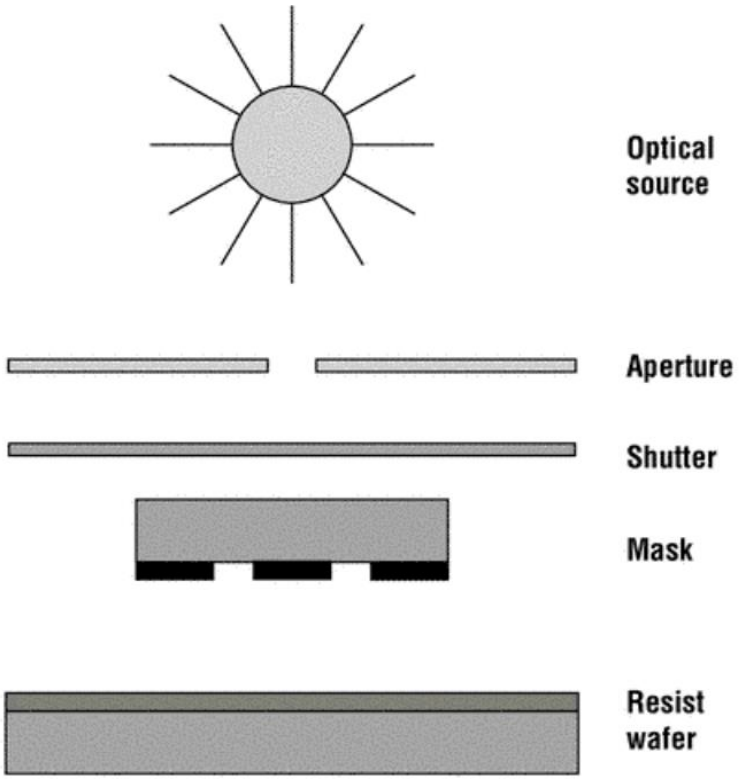


Figure 10 From Cambpell. Schematic of a typical lithographic exposure system.

A schematic of a projection stepper lithography system is shown in Figure 11. The mask for the features to be printed can be several times larger than the features desired to print. For example, a 4:1 printer would allow the mask to have a 4 μm wide line to print down to a 1 μm line on the wafer. This allows for cheaper masks due to larger feature sizes to be cut out. Another feature of the stepper lithography system is the ability to repeat the mask pattern exposure across the wafer. This allows for a single photolithography step to create many copies of the mask. By using the stepper, we were able to create 20 different copies of the switch design on a single wafer.

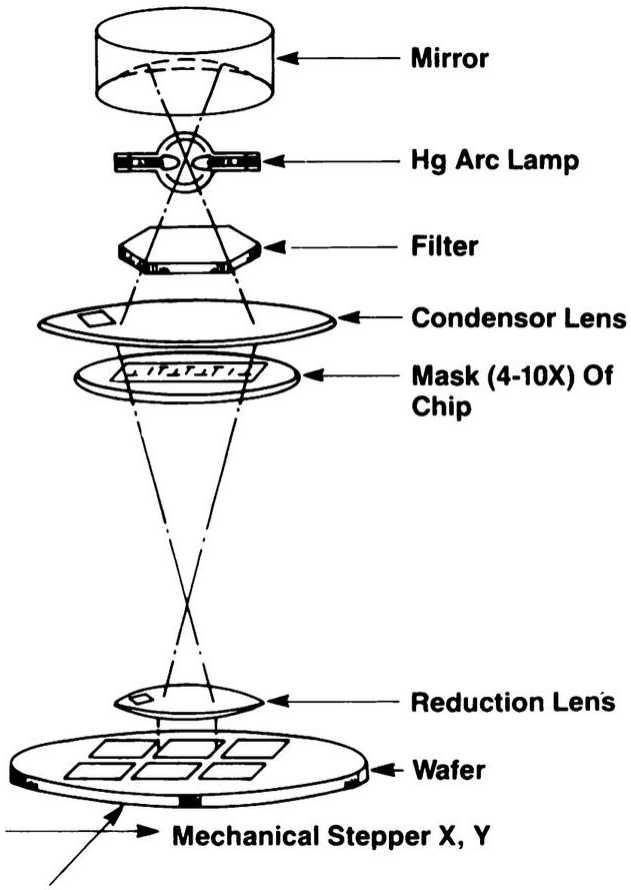


Figure 11 Schematic of refractive optics reduction printer. From Semiconductor Lithography Principles: Practices, and Materials, Moreau

3.3 Etching Features

The current process for device fabrication consists of two main etch steps, one two define the switches in the device layer of an SOI wafer, and another to remove the sacrificial SiO_2 layer beneath the areas that we want to be able to move freely. The first etch to define features in the device layer requires an anisotropic etch through Si. Several methods exist to do this, such as deep reactive ion etching (DRIE) or inductively coupled plasma (ICP) etching [8]. The removal of the SiO_2 layer requires an isotropic etch that will not induce stiction that prevents the devices from being fully released from the substrate [10]. Effects of these etches on different features in the low-power relays designed are explored.

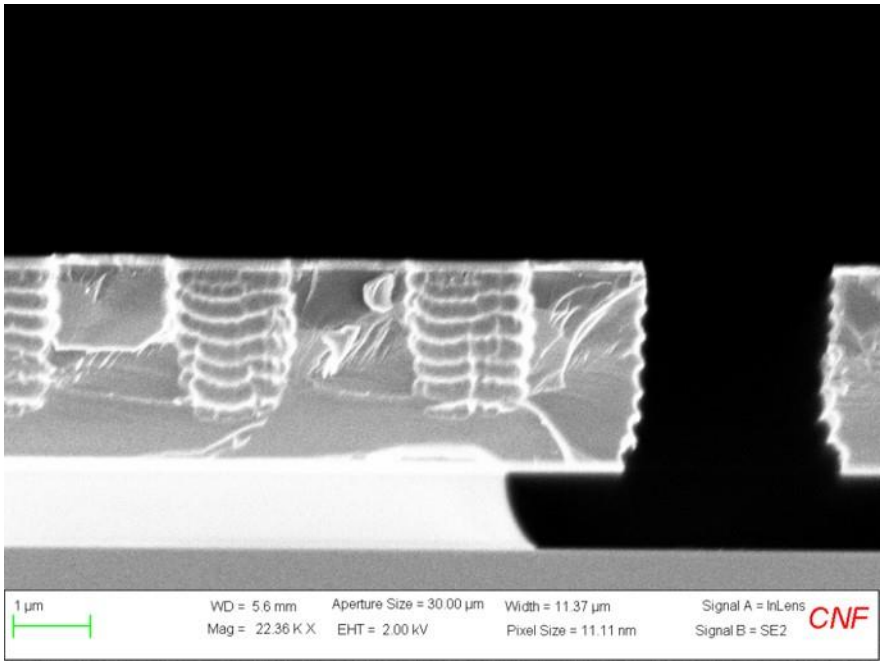


Figure 12: SEM cross section of failed DRIE etch of release holes, after vapor HF etching of SiO2. Able to observe properties of the different etches used.

3.3.1 Silicon Etch

To define desired features in the device layer, an anisotropic Si etch is required. We tried using both DRIE and ICP processes. The etch should leave behind a sidewall that will maximize contact area at the point where the source meets the drain when the switch is in the closed position. The scallops of the DRIE process shown in Figure 12 are not ideal due to a higher contact resistance when there is less area making physical contact for current to run through. An inductively coupled plasma (ICP) tool is used to etch through the silicon isotropically and results in smoother sidewalls as shown in Figure 17. The smoother sidewalls will allow for a lower contact resistance by increasing the surface area of contact of the source and drain of the switches.

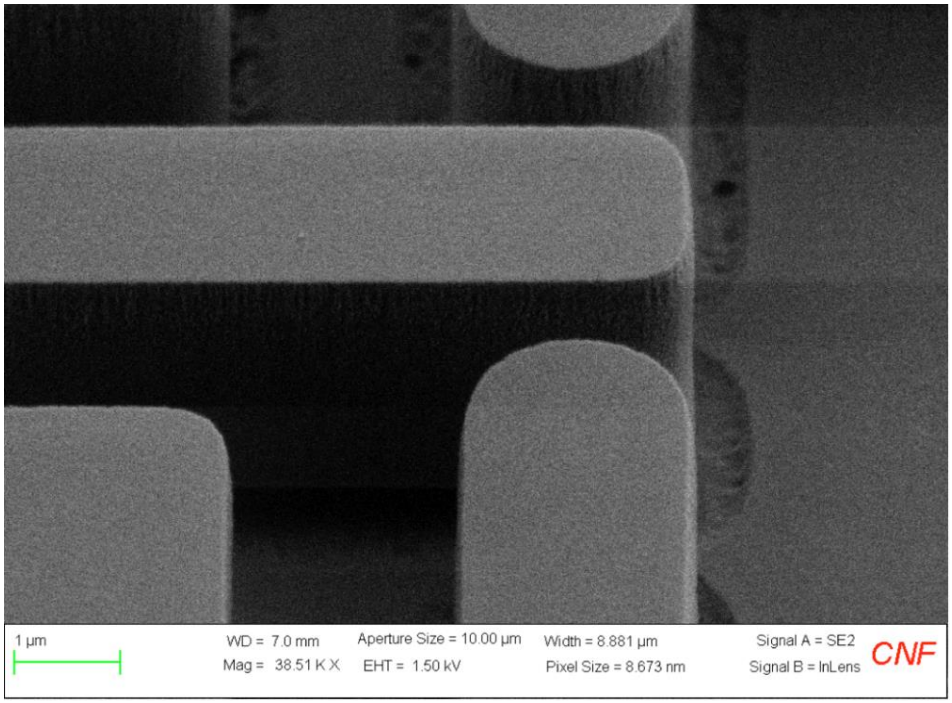


Figure 13 Smooth sidewall of ICP etch through Silicon.

3.3.2 Silicon Dioxide Etch

Once features are defined in the device layer, MEMS require a release mechanism to release beams anchored to the substrates to allow beams to move. If using a liquid etchant, care needs to be taken to prevent adhesion of the released structures to the substrate, or stiction. Critical point drying (CPD) with CO_2 can help prevent stiction during the drying process of a liquid etch [8]. Vapor or dry etches can avoid this issue.

In the current process, silicon dioxide is the release layer on the SOI wafer. Hydrofluoric (HF) acid is commonly used to etch silicon dioxide. The etch rate of SiO_2 can be up to 23,000 $Å/min$ in a concentrated HF solution. A buffered HF solution (BHF) can be used to etch through SiO_2 at a more controlled rate of 230 $Å/min$ [11]. This allows for more handling time to prevent unintended release of anchor structures. Vapor HF on SiO_2 has an etch rate of 660 $Å/min$ and does not have the issue of causing stiction of the released devices due to the surface tension of drying liquids.

3.3.3 Release Holes

To allow for smaller etch times and anchor structures when MEMS devices contain large structures, release holes are put into the large structures. We ran into an issue with these release holes during our exploration of the silicon etch process. Figure 12 shows a cross section of a device after the releasing vapor HF etch and showcases different characteristics of the DRIE etch and the HF etch. The device layer, SiO_2 layer, and part of the handle layer are visible. The left side shows a structure that has

attempted release holes through the device layer. The right side of image shows an area of etch that successfully went through the full device layer.

The sides of features going through the device layer show the scallops that are characteristic of the Bosch Process of a DRIE etch. This can be seen as horizontal lines on the release holes, and as the peaks going down the edges of the larger gap. The smaller-gap release holes did not get etched as deeply through the device layer as the main gap. A notching effect seen by the curve at the bottom of the device layer in the right most gap reveals an over-etch of the DRIE as the chemistry of the DRIE etch does not etch through SiO_2 as fast as it etches through Si.

You can see the isotropic nature of the vapor HF etch due to the 'bowl' edge that the oxide has close to the gap. One can imagine if the etch holes were successfully defined through the whole device layer, that the SiO_2 layer underneath would be completely removed and allow for the movement of the feature.

1.3.3.3 Determining Release

The release of the devices requires the removal of a layer that cannot be seen with a typical microscope. Since the devices gap features are so small, the most informative way to inspect the devices is using an SEM. However, due to the nature of the SEM and the electrostatic devices, the devices could get damaged by electron charge buildup. Hence, a non SEM method to determine successful release of devices is desired. There are two methods that were used. One method is to use infrared (IR) imaging to determine the release success. Si is transparent to IR and you can see a shadow of where oxide is gone, such as in Figure 15. Another method used is to use darkfield imaging to look at the undercut. Darkfield optical imaging places a cap on

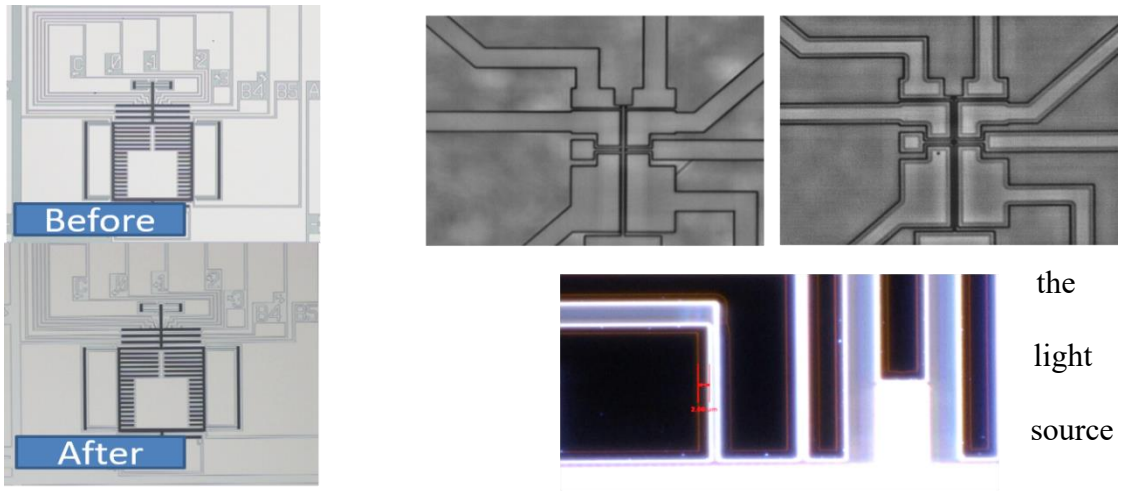


Figure 15: Different ways to determine if MEMS devices have been released with a successful HF etch of the sacrificial SiO2 layer.

of the microscope, only allowing collimated light to hit the sample. Only scattered light is received by the lens and eye. This reveals lines of the overhang of a successful etch.

3.4 Bad Wafer Oxide

Sourcing wafers from a good supplier is important. In Figure 16 you can see microscope images of two device that have gone through a vapor HF release process.

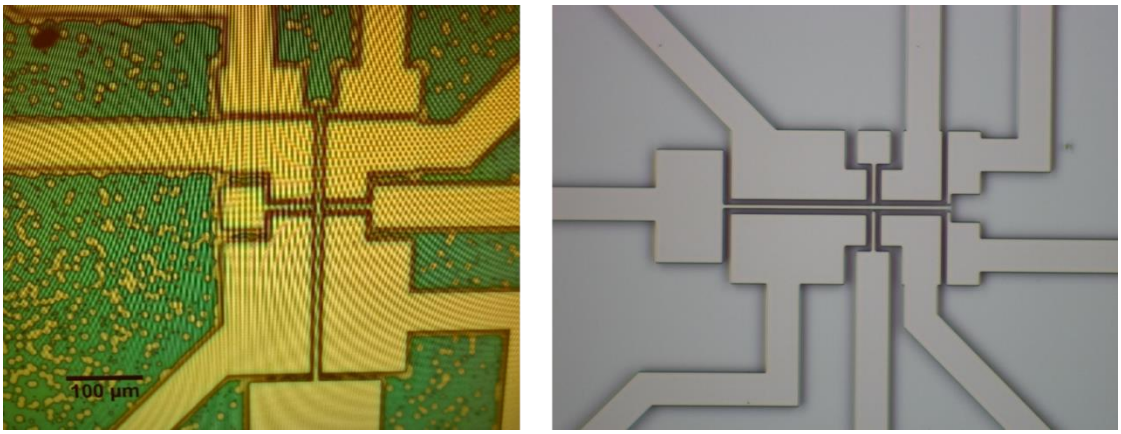


Figure 16: Two microscope pictures of devices after vapor HF etch. Left pictures shows a discontinuous etch of the oxide. Right picture shows uniform coloring across the device and Si underneath.

The color of the silicon is different in each image due to different white balance used by the image capturing software. The image on the left shows a green oxide with many little holes, showing a non-uniform. On the right side, there is no color difference between the top of the device and the etched away areas, confirming that there is exposed silicon in both areas.

3.5 Metal deposition

Work was done to improve contact resistance of the devices by depositing metal onto the contact area of the switch. Conformal evaporation of Pt was done to cover the sidewalls of the device after release. However, the thin layer of Pt also covers the top of the released devices and cause curvature. Thin films that have a different coefficient of thermal expansion than the substrate will induce either a compressive or tensile stress, as shown in Figure 18. Optical profilometry was performed to measure the

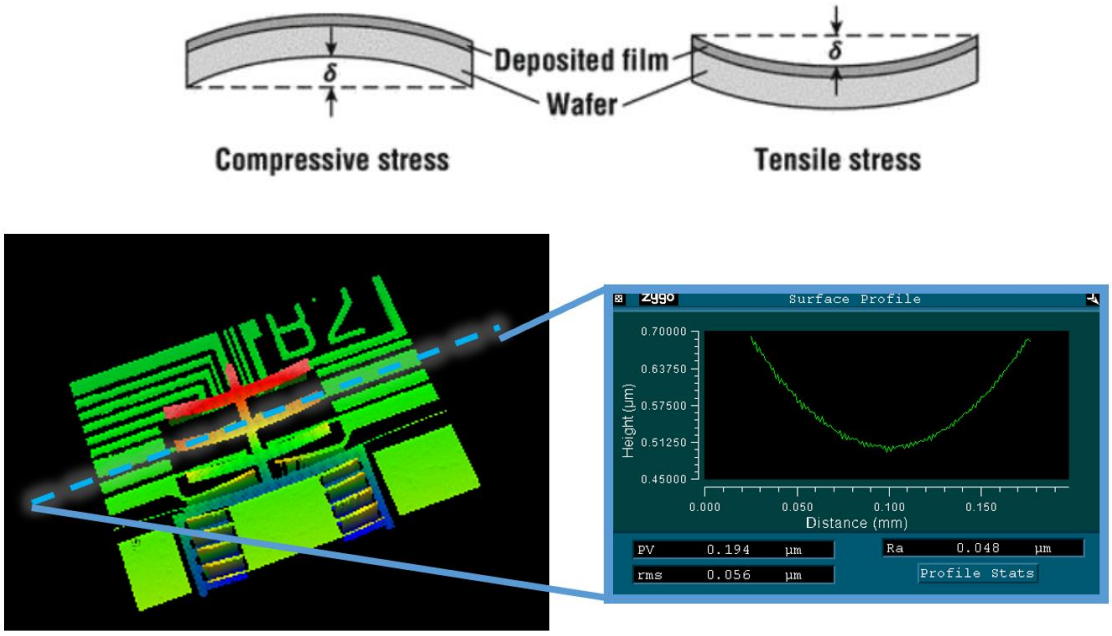


Figure 18 Top image: Direction of curvature caused by stress due to differences in CTE of deposited film and substrate. Bottom image: Optical profilometry result of beams bending after deposition of Pt.

bending of the devices after Pt was deposited. The testing results shown in the next section do not reflect the metal tests. Further work done with metal deposition to decrease the contact resistance of the devices in the closed state can be found in Ruyack et al in [12].

CHAPTER 4 TESTING OF ELECTROSTATIC SWITCHES

4.1 Testing Strategies

Several testing strategies were formed to examine different properties of the device in different stages of development. The following sections explains the tests that were used on the NEMS switches fabricated using the process explained the previous chapter.

4.1.1 Single Gate Test

The single gate test confirms a correct fabrication of the switch and gives information about the quality of the switch. The test requires three probes, for the gate, source, and drain. Voltage V_G is applied to the gate, voltage V_D to the drain, and the source is at ground. V_D remains at a constant voltage while V_G starts at 0 and is slowly increased. The drain current I_D is measured while V_G increases.

When I_D sharply increases, this indicates that the switch has closed. The voltage at which this occurs is called V_{on}, and the current level at this point determines the resistance R_{on}. The gate voltage is then decreased from this V_{on} until the current

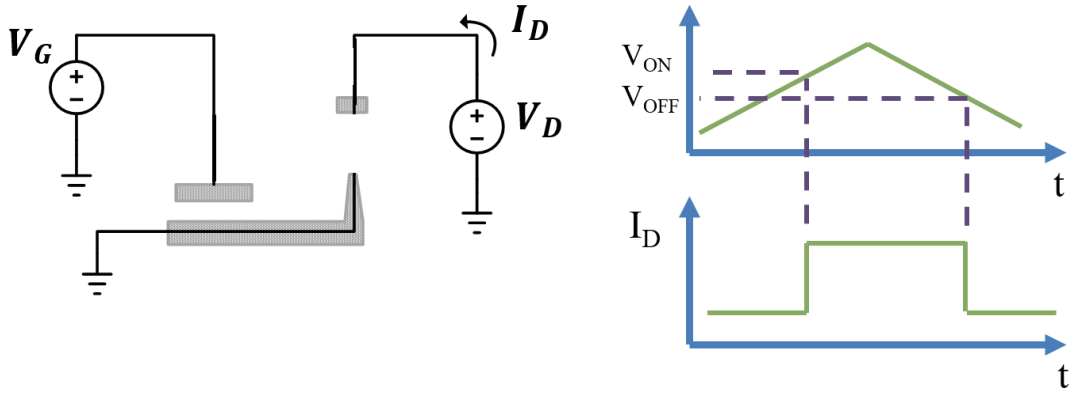


Figure 19: On left: example test setup of a three terminal MEMS relay. On right: Graph example of applied voltage and expected current response over time.

rapidly decreases. The voltage at this sudden decrease is V_{off}. An example schematic and graphs visualizing an ideal test are shown in Figure 19.

The test is done in a probe station and the voltages and currents are controlled and measured with a Keithley 2400 SMU. Several of these instruments can be controlled together through a GPIB protocol. Python and MATLAB can be used to write programs to control the measurement devices over GPIB (General purpose instrument bus) to automate the testing [13], [14]. This allows for testing over time to be done to determine the repeatability of the switch under test. A GPIB instrument control device

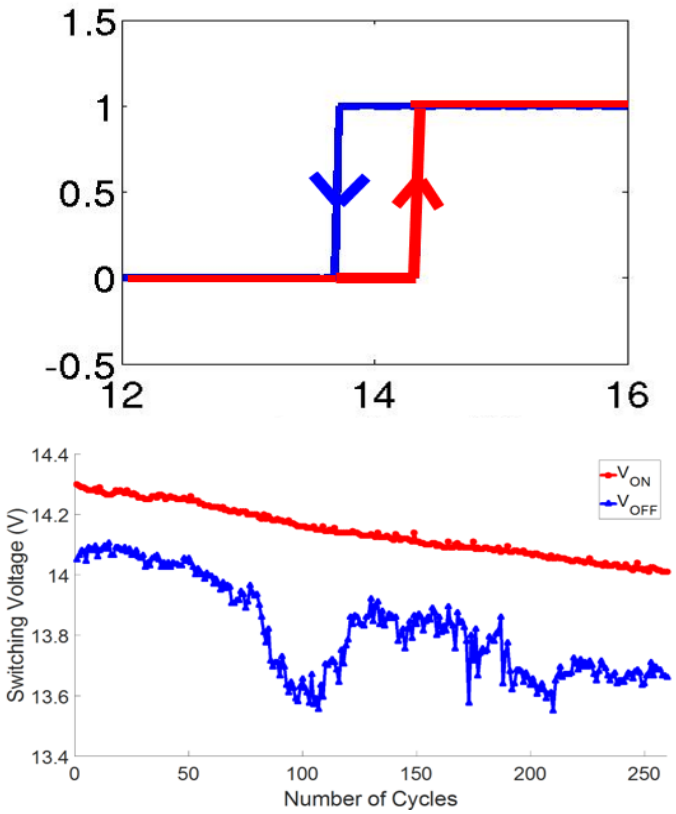


Figure 20: Result of test of three terminal device. Top figure shows the current over one cycle of voltage increase and decrease. Bottom figure shows how the voltage required to move the device closed and the voltage when the device opened again over 250 cycles.

from National Instruments was used to connect the testing instruments to a computer for programming and communication [15].

4.1.2 Single Gate Test Results

After fabricating a device, we can use this single gate test to confirm that it works and note the functionality. Figure 20 shows results of a single time test and a cycle test. The top graph shows I_D as a function of V_G. This single test shows that V_{on} and V_{off} are around 14 V. A cycle test is done around this voltage to determine the reliability of the switch over time. The bottom graph shows V_{on} and V_{off} over 250 cycles.

There are a few possible reasons for the decreasing trend of $\mathrm{V_{on}}$ and V_{off} over time. One reason pertains to a spring softening effect of the switch. The design of the switch means that current will flow through the silicon making up the spring of the switch. As current flows through, the silicon will heat up. As the silicon heats up, the young's modulus decreases, so the turn on voltage decreases. Another reason could be due to mechanical wear on the contact point. Also, as the device is switched on and off the physical contact point at the source-drain will change over time. Large asperities will smoothen out over time through mechanical wear [4]. This means that larger areas will be in contact with each other, allowing higher current to flow more easily.

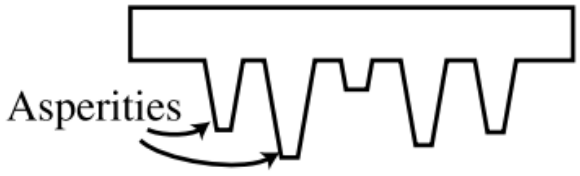


Figure 21 Contact surface area adjusted by asperities, affecting the contact resistance [4].

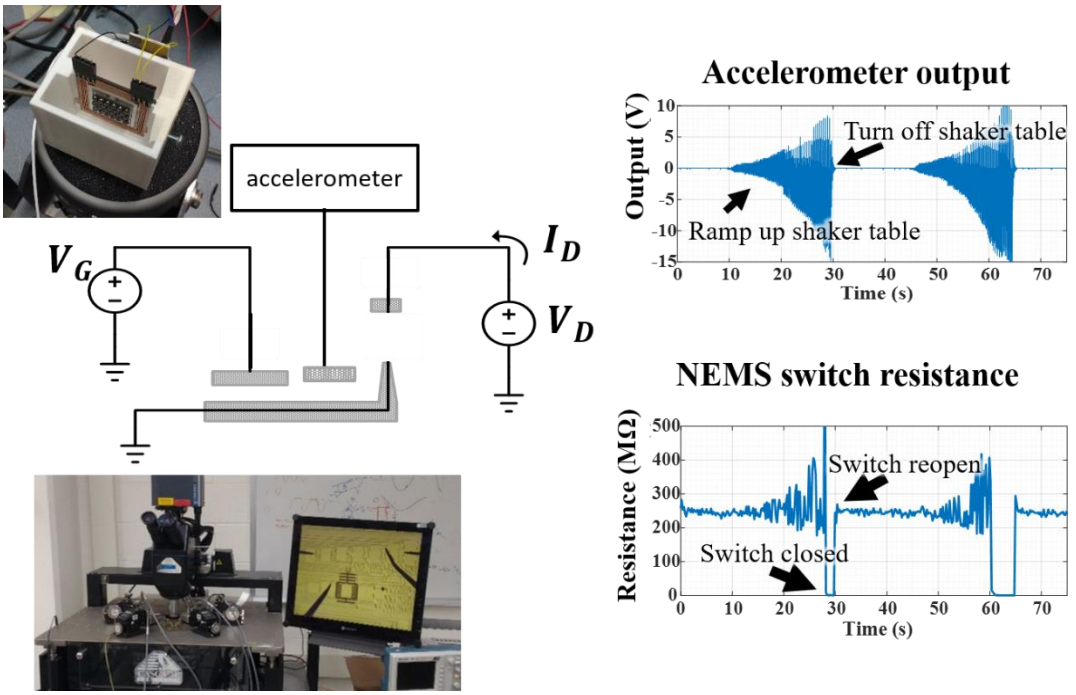


Figure 22: Result of test of three terminal device. Top figure shows the current over one cycle of voltage increase and decrease. Bottom figure shows how the voltage required to move the device closed and the voltage when the device opened again over 250 cycles.

4.2 Actuation by Accelerometer

Once a turn on voltage is found for one gate, we can attach an accelerometer to another gate to fully actuate the device. The test setup is shown in the left side of Figure 22 [16]. A bias voltage of 14 V is applied to one of the gate electrodes of the fabricated device. The output of a zero power sensor accelerometer made from PZT is attached to another gate electrode of the NEMS switch. The source, or main beam of the switch is connected to ground. A small voltage is applied to the drain and current is measured. The accelerometer is attached to a shaker table. The shaker table is controlled through a ramp up of intensity before a sharp cutoff. The output of the accelerometer and the NEMS switch is shown in Figure 22. Once the accelerometer output reaches a certain level, the contact resistance decreases abruptly, indicating a

closed switch. Upon the turning off of the shaker table, the contact resistance increases again, indicating an open switch. This cycle is able to be repeated, showing a device that is able to be cycled on and off with a pre-bias and external sensor.

4.3 Arduino Control Loop

To prevent damage to the device during testing due to high currents running through the device and causing excessive heating, a test utilizing an Arduino was made. A schematic of the test structure is shown in Figure 23. The single gate and external sensor test above have a DC voltage applied to the drain of the switch. This voltage has the potential to cause damage either by arcing between the drain and source as a gate voltage is applied and the source moves closer to the drain, or causing

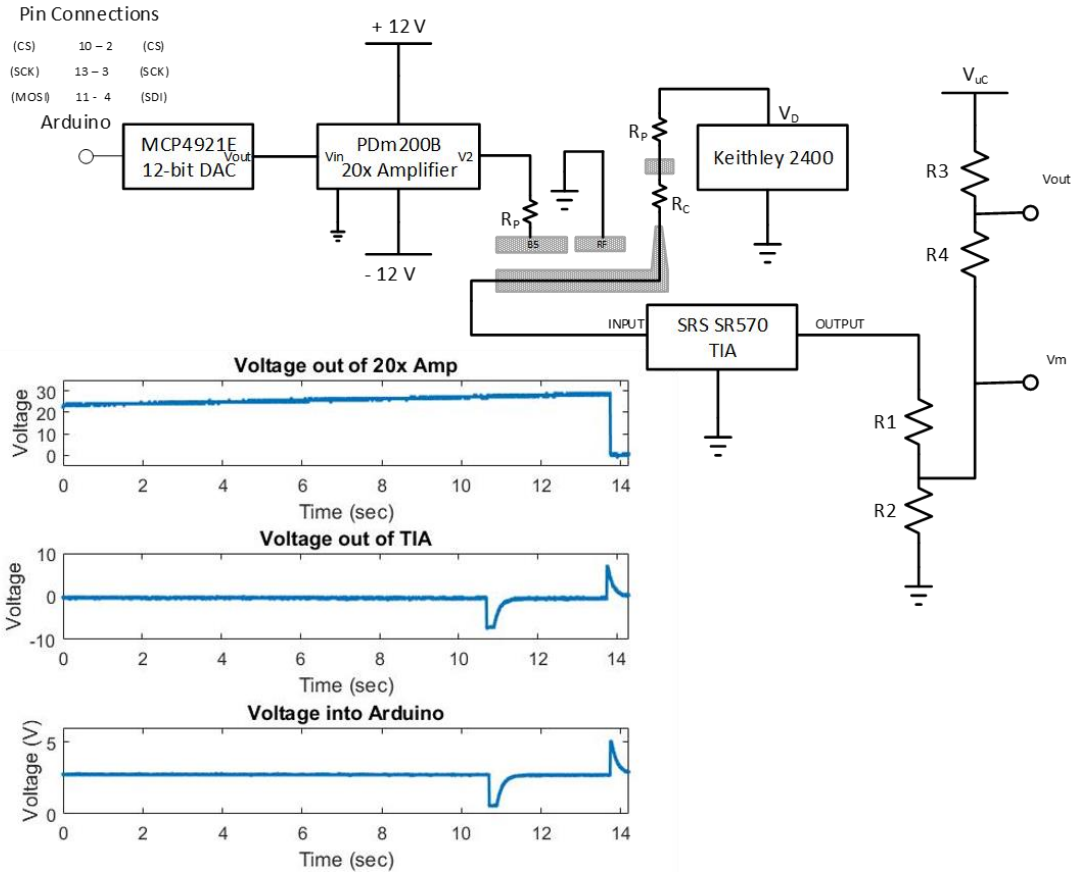


Figure 23: Schematic and result of Arduino control loop test

an increase in current and heat at the area of contact, melting the source and drain together.

In this test to find the turn on voltage of the device, the voltage on one gate is stepped up in voltage. Before the next gate voltage is turned on, the drain voltage is turned on briefly, drain current is measured, and then the drain voltage is turned off again. A computer is used to control an Arduino and a Keithley 2400 SMU. The Arduino controls a MCP4921E 12-bit DAC fed through a PDm200B 20x amplifier as the gate voltage [17], . The source is attached to a SRS SR570 transimpedance amplifier (TIA), which is then connected back to the Arduino's DAC. The Keithley 2400 SMU is connected to the computer through a GPIB to USB controller. Python code is used to coordinate the Arduino and SMU measurements using PySerial and PyVISA respectively. Example code of the Arduino controlling the DAC and communicating over serial to the computer is shown the Appendix.

In Figure 23, you can see the result of using capacitive changes instead of DC changes to determine if a switch has closed or not. As the gate voltage increases, there is a spike of current that is read by the TIA. When the gate voltage decreases, there is an equal and opposite spike of current read. This is thought to be due to currents caused by large changes of capacitance as the relay closes and opens.

CHAPTER 5 SUMMARY

In this thesis I have shown the design, fabrication, and testing of low power MEMS relays. These relays can potentially be used in situations to replace power-hungry CMOS technology in order to extend the lifetime of a power source in a sensor node deployed in a sensor network. The fabrication steps were explained in detail, with information on what factors one should consider when choosing one process over another. Multiple test processes of fabricated devices were explained, including a single gate test to test proper fabrication and robustness, a pulsed test to extend device life, and a multi-gate test with an external sensor [16]. The fabricated relays have been shown to have a pre-bias applied and successfully be actuated by an external piezoelectric accelerometer. This type of device can be used in many low power situations requiring a robust sensing mechanism.

This is not the only approach, and others have been working on similar devices. Alex Ruyack *et al* has continued the work and was able to use Focused Ion Beam to deposit Pt onto the tip of the switches to increase the contact conductance of the relay in the ON state [12]. Grogg *et al* has created a curved relay for low-power applications using PtSi as a side metal on the contact point [18].

BIBLIOGRAPHY

[1] H. Fahmy, *Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis*. Springer.

[2] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002, doi: 10.1109/MCOM.2002.1024422.

[3] T. Olsson, "NZERO Broad Agency Announcement." DARPA.

[4] G. M. Rebeiz, *RF MEMS Theory, Design, and Technology*. John Wiley & Sons, Inc, 2003.

[5] K. Amponsah, N. Yoshimizu, S. Ardanuc, and A. Lal, "Near-kT switching-energy lateral NEMS switch," in *2010 5th IEEE International Conference on Nano/Micro Engineered and Molecular Systems (NEMS)*, Jan. 2010, pp. 985–988, doi: 10.1109/NEMS.2010.5592599.

[6] V. Kaajakari, *Practical MEMS*. Small Gear Publishing, 2009.

[7] H. Kam and F. Chen, *Hei Kam•Fred ChenMicro-Relay Technologyfor Energy-EfficientIntegrated Circuits*. Springer.

[8] S. Senturia, *Microsystem Design*. Kluwer Academic Publishers, 2002.

[9] "ASML 300C DUV Stepper | CNF Users." https://www.cnfusers.cornell.edu/node/3 (accessed May 10, 2020).

[10] S. Campbell, *Fabrication Engineering at the Micro- and Nanoscale (The Oxford Series in Electrical and Computer Engineering)*. Oxford University Press, USA, 2008.

[11] K. R. Williams and R. S. Muller, "Etch rates for micromachining processing," *Journal of Microelectromechanical Systems*, vol. 5, no. 4, pp. 256–269, Dec. 1996, doi: 10.1109/84.546406.

[12] A. Ruyack, B. Davaji, L. Pancoast, N. Shalabi, A. Molnar, and A. Lal, "NEMS Electrostatic Resonant Near-Zero Power Resistive Contact RF Wake-Up Switch with PT FIB Contact," in *2019 IEEE 32nd International Conference on Micro Electro Mechanical Systems (MEMS)*, Jan. 2019, pp. 129–132, doi: 10.1109/MEMSYS.2019.8870800.

[13] "PyVISA: Control your instruments with Python — PyVISA 0.0.0 documentation." https://pyvisa.readthedocs.io/en/latest/ (accessed May 14, 2020).

[14] "Working with the GPIB Interface - MATLAB & Simulink." https://www.mathworks.com/help/instrument/working-with-the-gpib-interface.html (accessed May 14, 2020).

[15] "GPIB-USB-HS - National Instruments." https://www.ni.com/en-us/support/model.gpib-usb-hs.html (accessed May 14, 2020).

[16] V. Pinrod *et al.*, "Zero-power sensors with near-zero-power wakeup switches for reliable sensor platforms," in *2017 IEEE 30th International Conference on Micro Electro Mechanical Systems (MEMS)*, Jan. 2017, pp. 1236–1239, doi: 10.1109/MEMSYS.2017.7863640.

[17] "12-Bit DAC with SPI Interface," p. 40, 2004.

[18] D. Grogg *et al.*, "Curved in-plane electromechanical relay for low power logic applications," *J. Micromech. Microeng.*, vol. 23, no. 2, p. 025024, 2013, doi: 10.1088/0960-1317/23/2/025024.

CHAPTER 6 APPENDIX

6.1 Code - Arduino

```
// This file controls an Arduino Uno to test an
// NZERO NEMS Switch with the DC-TIA test.

// This file will ramp up and down around the turn on voltage
// to find the hysteresis of a given device.

// DAC used is MCP4921E, communicated with SPI

// Thanks to equaliser for a test ze used with a Teensy
// and uploaded to GitHub.
// https://gist.github.com/equaliser/8216031



// Leanna Pancoast 15 Nov 2017

#define DEBUG

#include <SPI.h>

#ifdef DEBUG
#define DEBUG_PRINT(x)     Serial.print (x)
#define DEBUG_PRINTDEC(x)     Serial.print (x, DEC)
#define DEBUG_PRINTLN(x)  Serial.println (x)
#else
#define DEBUG_PRINT(x)
#define DEBUG_PRINTDEC(x)
#define DEBUG_PRINTLN(x)
#endif

// initialize hardware pins
int ledPin = 7;

int slaveSelectPin = 10;

int gatePin = 7;
int sourcePin = A0;
int resetPin = 5;

const byte interruptPin = 2;
volatile byte killState = LOW;
byte printKill = HIGH;

unsigned long previousMillis = 0;

// reset pulse length in ms
int resetLength = 1;

uint16_t ftou(float x){
    return x * (4096 / 5.0);
```

```
}

float utof(uint16_t x){
    return x * (5.0/4096);
}

// initialize testing parameters
uint16_t gate_start = ftou(0/19.5);
uint16_t gate_step = 1;
uint16_t gate_limit = ftou(50.0/19.5);

// delay in ms
const long gate_delay = 200;

uint16_t gateV = 0;

// 2.8 is the 'resting' voltage. Can change to
// a moving average later.
float sourceRest = 2.8;
float vS_thresh = 0.05;

int inByte = 0;

//uint16_t prevV = 0;


float mapf(float x, float in_min, float in_max, float out_min, float
out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
out_min;
}

void setup() {
  // Initialize the Serial to talk to PC
  Serial.begin(115200);
  Serial.setTimeout(50);

  pinMode(slaveSelectPin, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(resetPin, OUTPUT);

  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV2); // 8 MHz on Arduino Uno

  Serial.println("Ready v2.2");

  pinMode(ledPin, OUTPUT);
  write_value(0);

  // setup interrupt things
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), killCode,
FALLING);
```

```
  // give Arduino time before starting
  delay(100);
}

void killCode(){
  killState = HIGH;
}

void resetPulse(){
  digitalWrite(resetPin, HIGH);
  delay(resetLength);
  digitalWrite(resetPin, LOW);
}

void loop() {
  // Wait for python code to say OK to run
  if(killState == HIGH){
    if(printKill){
      Serial.println("stopped");
    }
    digitalWrite(ledPin, LOW);
    printKill = LOW;
  }
  if (Serial.available() > 0) {
    // get incoming byte:
    inByte = Serial.read();

    switch (inByte) {
      case 'p':
        delay(10);
        // Parameters are being input
        // Gate start, gate step, gate end
        if (Serial.available() > 0) {
          gate_start = Serial.read();
        }
        if (Serial.available() > 0) {
          gate_step = Serial.read();
        }
        if (Serial.available() > 0) {
          gate_limit = Serial.read();
        }
        Serial.println("received parameters");
        break;
      case 'b':
        // Begin the sweep
        int worked = 1;
        gateV = gate_start;

        int contLoop = 1;

        digitalWrite(ledPin, HIGH);
        delay(250);
        digitalWrite(ledPin, LOW);
        delay(250);
        digitalWrite(ledPin, HIGH);
```

```
delay(250);
digitalWrite(ledPin, LOW);
delay(250);
digitalWrite(ledPin, HIGH);
delay(250);
digitalWrite(ledPin, LOW);
delay(250);
digitalWrite(ledPin, HIGH);

while(contLoop){
  // check if should stop the process
  if( killState == HIGH ){
    write_value(0);
    contLoop = 0;
    Serial.println("state killed");
    digitalWrite(ledPin,LOW);
    break;
  }

  // check if should continue loop
  if ( Serial.available() > 0 ) {
    inByte = Serial.read();
    // s will be sent in ctrl+c case
    if (inByte == 's') {
      write_value(0);
      contLoop = 0;
      break;
    }
  }

  Serial.println("loop");

  // start ramp up
  Serial.println("start ramp up");
  int ru = rampUp();

  if(!ru){
    // something wrong happened, get out of loop
    Serial.println("ramp up failed");
    digitalWrite(ledPin, LOW);
    break;
  }

  Serial.println("start ramp down");
  // start ramp down
  int rd = rampDown();

  if(!rd){
    // something wrong happened, get out of loop
    Serial.println("ramp down failed");
    digitalWrite(ledPin, LOW);
    //break;
  }
}
```

```
        // exited loop, now set output to 0 V
        write_value(0);
        digitalWrite(ledPin, LOW);
        Serial.println("End program");
    }
  }
}

int rampUp(){
  // will return 1 on success
  // return 0 on not success
  // This function will ramp the gate voltage up until
  // the devices switches or the gate voltage reaches the limit

  // check the source voltage
  float sourceV = getSourceVolt();

  if( abs(sourceRest - sourceV) > vS_thresh ){
    // the switch is shorted from the start
    Serial.print("Shorted from the start with: ");
    Serial.print(sourceV);
    Serial.println(" V");
    return 0;
  }

  // check if out of bounds to ramp up
  if ( gateV > gate_limit){
    Serial.println(gateV - gate_limit);
    Serial.println(gateV);
    Serial.println(gate_limit);
    Serial.println("Open Switch");
    write_value(0);
    return 0;
  }

  // good to start the ramp up
  while( gateV < gate_limit ) {

    // check if should stop the process
    if( killState == HIGH){
      write_value(0);
      Serial.println("state killed in ramp up");
      digitalWrite(ledPin, LOW);
      return 0;
    }

    unsigned long currentMillis = millis();

    // only print out when it increases by a volt
    //prevV = gateV;

    // increase gate voltage
    if(currentMillis - previousMillis >= gate_delay){
      gateV += gate_step;
      write_value(gateV);
```

```
      previousMillis = currentMillis;
      //Serial.print("gate V orig: ");
      //Serial.println(utof(gateV),5);
      //if(gateV - prevV > 1){
        Serial.print("gate V final: ");
        Serial.println(utof(gateV)*19.5,5);
        //Serial.print("Binary: ");
        //Serial.println(gateV, BIN);
        //Serial.println();
      //}
      Serial.print("Source volt: ");
      Serial.println(sourceV);

    }

    // look at source voltage to see if switch
    sourceV = getSourceVolt();

    if( abs(sourceRest - sourceV) > vS_thresh ){
      // then the device has closed!
      Serial.print("Closed at ");
      Serial.print(mapf(gateV,0,4095,0,5)*19.5);
      Serial.print(" V with ");
      Serial.print(sourceV);
      Serial.println(" V on source");

      // don't want device to touch for a long time
      // so need to try to pull it off

      // apply reset pulse
      resetPulse();

      // back off voltage 5 steps
      gateV -= 5*gate_step;
      write_value(gateV);


      return 1;
    }
  }
  Serial.println("Reached limit, open switch");
  return 0;
}


int rampDown(){
  // will return 1 on success
  // return 0 on not success
  // This function will ramp the gate voltage up until
  // the devices opens or the gate voltage goes below 0

  // check the source voltage
  float sourceV = getSourceVolt();

  if( abs(sourceRest - sourceV) < vS_thresh ){
```

```
  // the switch is open from the start
  Serial.print("Open from start of Ramp Down with ");
  Serial.print(sourceV);
  Serial.println(" V on source");
  return 1;
}

// check if out of bounds to ramp up
if ( gateV < 0){
  Serial.println("Not Opening");
  return 0;
}

// good to start the ramp down
while( gateV > gate_step ) {

  // check if should stop the process
  if( killState == HIGH){
    write_value(0);
    Serial.println("state killed in ramp down");
    digitalWrite(ledPin, LOW);
    return 0;
  }

  unsigned long currentMillis = millis();

  // decrease gate voltage if time is correct
  if(currentMillis - previousMillis >= gate_delay){
    previousMillis = currentMillis;
    //Serial.print("gate V orig: ");
    //Serial.println(utof(gateV));
    Serial.print("gate V final: ");
    Serial.println(utof(gateV)*19.5);

    gateV -= gate_step;
    write_value(gateV);
    Serial.print("Source volt: ");
    Serial.println(sourceV);
  }

  // look at source voltage to see if switch
  sourceV = getSourceVolt();

  if( abs(sourceRest - sourceV) < vS_thresh ){
    // then the device has opened!
    Serial.print("Opened at ");
    Serial.print(mapf(gateV,0,4095,0,5)*19.5);
    Serial.print(" V with ");
    Serial.print(sourceV);
    Serial.println(" V on source");
    return 1;
  }
}
Serial.println("Reached 0, closed switch");
return 0;
```

```
}


float getSourceVolt(){
  // This function will return the voltage the Arduino is reading
  // on the given source pin.
  int vS = analogRead(sourcePin);
  // map to get real voltage number
  float vSa = mapf(vS, 0, 1023, 0, 5);
  return vSa;
}

void write_value(uint16_t value) {
  // channel (0 = DACA, 1 = DACB) // Vref input buffer (0 =
unbuffered, 1 = buffered) // gain (1 = 1x, 0 = 2x)  // Output power
down power down (0 = output buffer disabled) //  12 bits of data
  uint16_t out = (0 << 15) | (1 << 14) | (1 << 13) | (1 << 12) |
(value);
  digitalWrite(slaveSelectPin, LOW);
  SPI.transfer(out >> 8);         //you can only put out one byte at
a time so this is splitting the 16 bit value.
  SPI.transfer(out & 0xFF);
  digitalWrite(slaveSelectPin, HIGH);
}
```

6.2 Code – Python

```
# This file works with an Arduino Due to test
# an NZERO RF NEMS Switch with the DC-TIA Test.

# Written by Leanna Pancoast
# 9 Nov 2017
# Cornell University
# Inspiration by the Girino project by Cristiano Lino Fontana

import serial
#from struct import unpack
#import numpy as np
#import logging
#from matplotlib import pyplot as plt
import time
import os.path
import sys

# initial setup things

device_name = 'SOI8-10-M1R-11-E2'

# starting gate voltage
gate_start = 0
# gate step in volts
gate_step = 0.5
# gate limit (max voltage to apply)
```

```
gate_limit = 60

gate_vs = []
source_vs = []


# ******* START SETUP ********
# *** LOGGING ***
# setup logging
#logs_folder_path = "C:\\Users\\MOLNAR_PH330_D1\\Desktop\\NZERO\\TIA
Automation\\logs\\"
#now = time.strftime('%Y%m%d_%H%M%S')
# complete log
#log_fn = "{}{}_{}_log.txt".format(logs_folder_path, device_name,
now)
#data_fn = "{}{}_{}_data.txt".format(logs_folder_path, device_name,
now)

if os.path.exists(log_fn):
    print("you've gone back in time? File already exists")
    sys.exit(0)

# setup logging
#formatter =
logging.Formatter(fmt='%(asctime)s.%(msecs)03d, %(message)s',
datefmt='%m/%d/%Y %H:%M:%S')
#log = logging.getLogger()
#log.setLevel(logging.INFO)
#hdlr1 = logging.FileHandler(log_fn)
#hdlr1.setFormatter(formatter)
#hdlr1.setLevel(logging.INFO)
#log.addHandler(hdlr1)
#
#data = logging.getLogger('data')
#hdlr2 = logging.FileHandler(data_fn)
#hdlr2.setFormatter(formatter)
#data.addHandler(hdlr2)


# *** SERIAL ***
# open serial port
print('Starting Serial port')
#log.info('Starting Serial Port')
stream = serial.Serial( 'COM5',115200,timeout=5)
time.sleep(0.5)
line = ''
while line.find("Ready") == -1:
    line = stream.readline()

print('teehee')
print line


### *** HELPER FUNCTIONS ***
```

```
def checkFail(line):
    if( line.find("killed") != -1):
        print("Kill switch activated")
        return 1

    if( line.find("up failed") != -1):
        print("ramp up failed")
        return 1

    if( line.find("down failed") != -1):
        print("ramp down failed")
        return 1

    if ( line.find("Shorted") != -1 ):
        print("Shorted from start {}".format(stream.readline()))
        return 1
    if( line.find("open") != -1):
        print("Open switch")
        return 1
    if( line.find("stuck") != -1):
        print("stuck switch")
        return 1
    if( line.find("stopped") != -1):
        print("stopped switch")
        return 1
    if( line.find("End") != -1):
        print("End Program")
        return 1

def main():
    global gate_start, gate_step, gate_limit, gate_vs, source_vs,
line

    #### ***** START MEASUREMENT INITIALIZATION *****
    # give the sweep parameters to the Due
    #stream.write('p')
    #stream.write(gate_start)
    #stream.write('{}'.format(gate_step))
    #stream.write(gate_limit)
    #line = stream.readline()
    #print('blah')
    #line = stream.readline()
    #print(line)


    #### ******* STARTING MEASUREMENT *******
    print("** About to test switch {}. Do you want to
continue?\n".format(device_name))
    raw_input("*Press enter to start test\n")
#    log.info("Testing switch")
    # Tell Due to Begin the sweep
    stream.write('b')
    line = stream.readline()
    print('wrote b\n')
```

```
    checkFail(line)

    while(not checkFail(line)):

        if (line.find('Closed') != -1):
            vG = stream.readline()
            vS = stream.readline()
            print( "device closed at {} V with {} Vo".format(vG, vS))

        elif (line.find('Opened') != -1):
            vG = stream.readline()
            vS = stream.readline()
            print( "device closed at {} V with {} Vo".format(vG, vS))


        else:
            #vGa = line
            vGb = stream.readline()
            vS = stream.readline()


            gate_vs.append(vGb)
            source_vs.append(vS)
            print('gate V: {}, source V: {}\n'.format(vGb, vS))

        line = stream.readline()


    print('done!')

    stream.close()
#    hdlr1.close()
#    hdlr2.close()


try :
    main()
except KeyboardInterrupt:
    # set everything to 0 if interrupted with keyboard
    print('closed with ctrl+c')
#    log.info('closed with ctrl+c')
    stream.write('s')
    stream.close()
#    hdlr1.close()
#    hdlr2.close()
```