Building a Seekat Digital-to-Analog Converter

Aric Moilanen
Middle Tennessee State University
University of Michigan Physics REU
PI: Lu Li
August 12, 2019

My goal during my time at the University of Michigan was to build a digital-to-analog converter (DAC) that would control a DC voltage source. In the future, this DAC will be used to apply a gate voltage to samples in a custom-built capacitance bridge. I assembled, tested, and corrected the DAC as well as wrote code to control it.

Background

Professor Li's Group is attempting to measure the density of states of a two-dimensional system. The proposed method to do this is through measuring the sample's capacitance which is described by the equation below.

$$\frac{1}{C} = \frac{1}{C_{geom}} + \frac{1}{C_q} = \frac{1}{C_{geom}} + 1/[e^2 D]$$

C is the total capacitance of the system. The geometric capacitance, C_{geom}, is what we usually think of when considering capacitance and is given by $C_{geom} = \varepsilon \frac{A}{d}$ where ε is the permittivity of the material separating the plates of the capacitor, A is the area of the plates, and d is the distance between the plates. The quantum capacitance, C_q, is much less familiar. This phenomenon occurs when at least one of the plates of the capacitor is made of a low density of states material. In this situation, as the capacitor charges, the negative plate will fill up with electrons occupying high energy states while the positive plate is left with electrons occupying lower energy states. This leads to a change in how the capacitor charges, and thus the capacitance of the sample. The effect of quantum capacitance can be interpreted as that of a second capacitor in series as shown in Figure 1, where the C_q capacitor has a value $C_q = e^2 D$ with D being the density of states. Since the geometric capacitance can be calculated, we can determine the quantum capacitance of our sample and consequently its density of states by measuring the total capacitance of the sample.

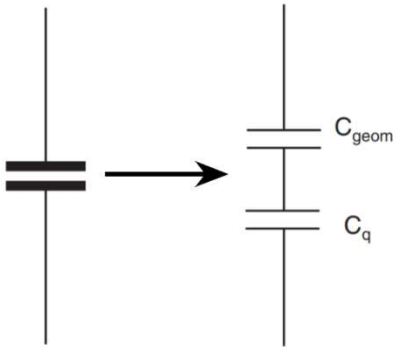


Figure 1: The capacitance of the sample can be represented as two capacitors, C_{geom} and C_q, acting in series [1].

Our tool for measuring the capacitance of our samples is a capacitance bridge. Unfortunately, commercial capacitance bridges lack the frequency and voltage control required for this project, so we must build our own, pictured in its simplest form in Figure 2. In this setup, C is a capacitor of known value, C_S is the sample of unknown capacitance, and V_e and V_b are AC voltage sources. There is also a DC gate voltage, V_g, that is applied to the sample before taking any measurements.

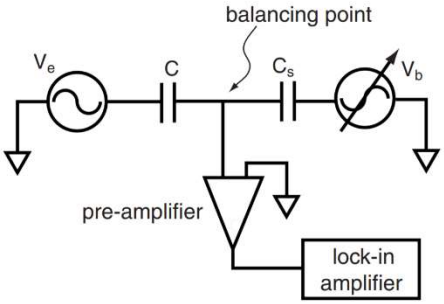


Figure 2: Setup sketch for the capacitance bridge [1].

By varying the phase and amplitude of one AC voltage until null is read at the balancing point, we can determine the capacitance of the sample. We can then adjust the gate voltage and repeat, giving us the capacitance of our sample as a function of the gate voltage. The DAC I built this summer is what will be used to apply the gate voltage to the sample once the rest of the capacitance bridge is complete.

Assembly

The Seekat DAC relies upon two main components: the AD 5764 evaluation board and the Arduino Due. The AD 5764 is the component that performs the actual digital to analog conversion in the Seekat DAC. It is a 16-bit, quad output digital to analog converter with a voltage range of -10V to +10V. Unfortunately, the software for these boards is outdated, so we needed a better way to communicate with them.

Instead, we used an Arduino Due to control the boards using serial peripheral interface (SPI) communication. In order to communicate over SPI, I had to solder 14 pin headers on to both AD 5764s as well as solder a shield for the Due that would allow me to access the required pins for SPI and run power to the AD 5764s. Once this was done, I wired the DAC following the diagram below.

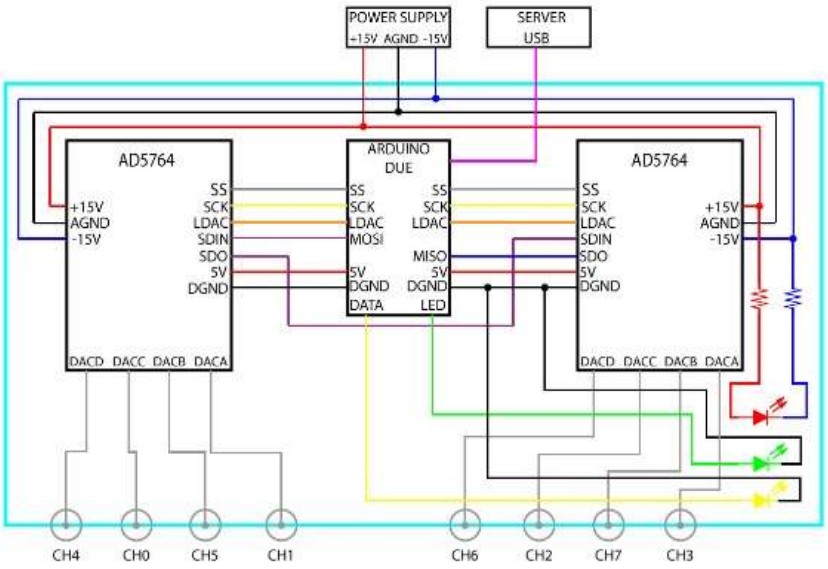
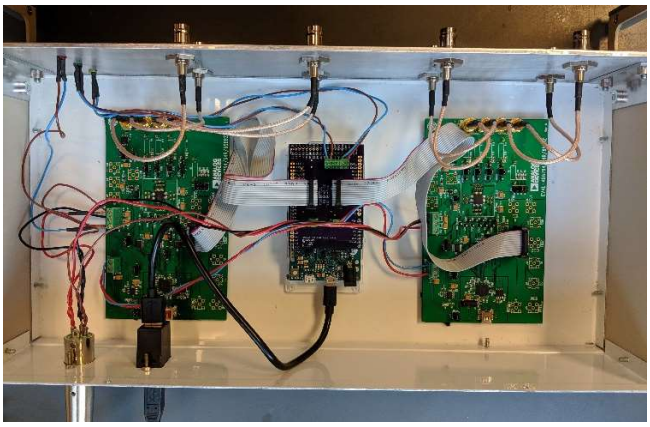


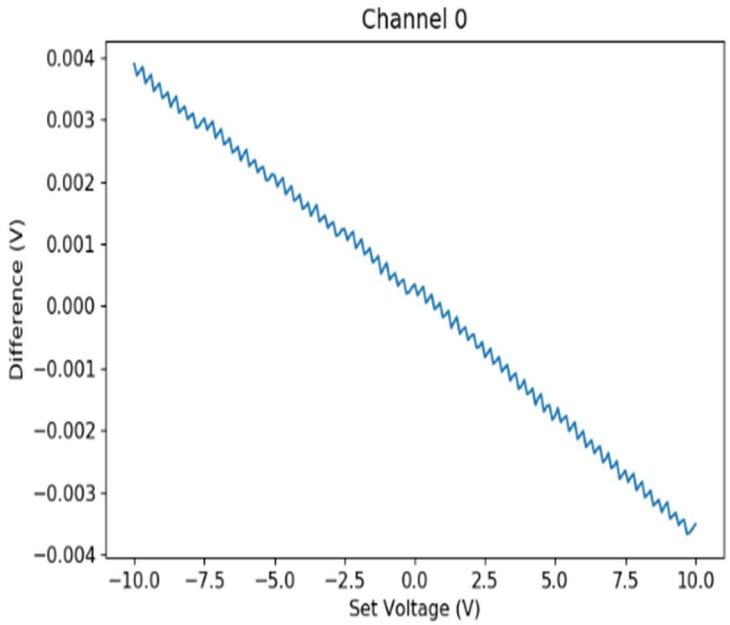
Figure 3: Wiring diagram for the Seekat Digital-to-Analog Converter

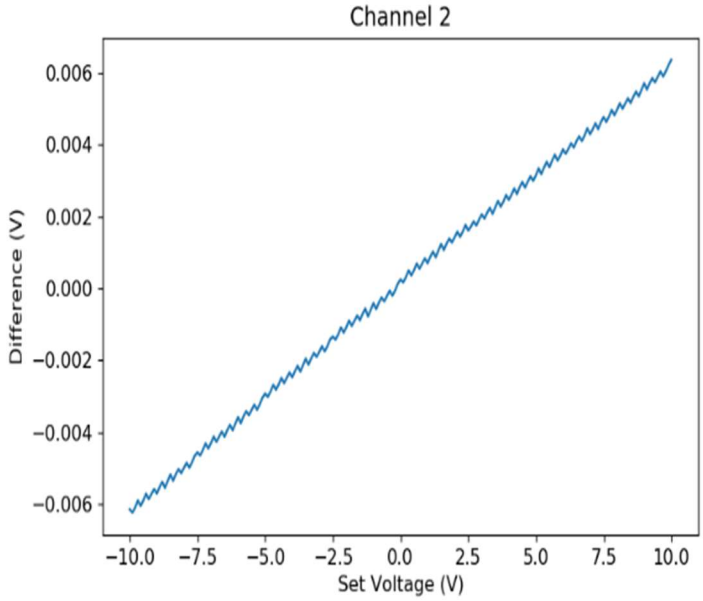
The finished DAC is pictured below, with the two AD 5764s on the left and right, the Arduino in the middle, and the eight voltage outputs at the top.



Testing and Error Correction

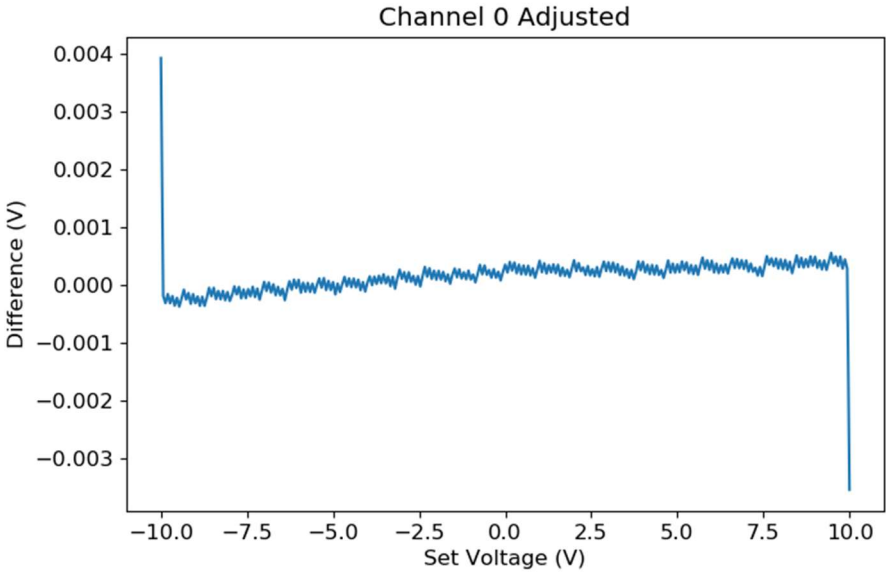
After assembling the DAC, I moved on to testing how much it drifted from its set voltage. We are looking to use this DAC in the range of -1V to +1V, so it needs to be very accurate especially at low voltages. I connected the Seekat to a six decimal digital multimeter and ramped the voltage across the Seekat's full output range. I saved what voltages I had set and what was measured using the multimeter, and then repeated six more times. I then averaged the differences between what was set and measured for all seven runs and produced the plots shown below.

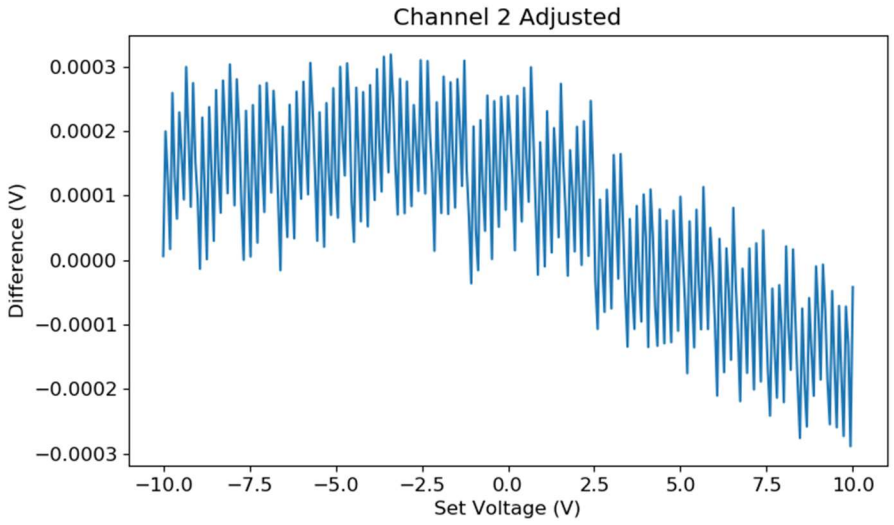




Without any correction, we were already looking at a drift of only a few thousandths of a volt. However, the drift is a very linear error, with the boards controlled by the first AD 5764 (shown in the Channel 0 plot) routinely undershooting and the boards controlled by the second AD 5764 (shown in the Channel 2 plot) routinely overshot. I went back through my data for the seven runs, calculated the factor that each board was undershooting or overshooting by, and applied a linear correction.

After applying the linear correction, I repeated my process outlined previously and produced another set of plots shown below.





The linear correction managed to reduce the error by approximately an order of magnitude, to just a few ten thousandths of a volt. Unfortunately, I could not correct the error right near the voltage limits on the channels that were undershooting, resulting in the sharp tails that can be seen in the Channel 0 Adjusted plot. This is due to the AD 5764s not taking commands to set a voltage less than -10V or greater than +10V.

Code

The Seekat DAC is an open source project, so the code for the Due was already available online. In order to control the Seekat, you only need to pass strings to the Arduino over a serial connection in the following format.

"OPERATION, DATA \r"

For example, the following string,

"RAMP1, 1, -10, 10, 100, 5000 \r"

would ramp channel 1 from -10V to +10V, with a hundred steps in between, and 5000 microseconds between each step. I chose to pass these strings using the Python module PySerial. I also coded in Python a simple text menu to control the Seekat which could perform a variety of tasks including querying the voltage set on each channel, setting the voltage for a given channel, ramping the voltage on a channel, and outputting a variety of waveforms.

References

[1] Li, L. *et al.* Very Large Capacitance Enhancement in a Two-Dimensional Electron System. *Science* **332,**825–828 (2011).

[2] *OpenDACs*, opendacs.com/seekat-with-arduino-due-homepage-2/seekat-illustrated-assembly-instructions/.

Acknowledgements

First, I would like to acknowledge and thank the NSF (grant no. 1852239) for funding this REU program and the University of Michigan for hosting the program. I would also like to thank Professor Myron Campbell, Professor James Lui, and Grace Johnson for organizing the program, and for all the additional effort they put in to make the program exceptional. Finally, I would like to thank Professor Lu Li for allowing me to join his group for the summer, as well as for the mentoring and assistance he provided throughout the program.