Creating a Capacitance Tester using BeagleBone Black

## By

## Klint Mahne

## Tutorial Synopsis

Capacitors are a very integral part of many circuits – from storing voltage to jump a battery, handling inrush current for your motor, the bootstrap in which controls your high side driver, or as a filter helping to transform AC waveforms into a DC state. Although the functionality of the capacitors may change depending on the job assigned to it, the formula that determines how fast it charges stays the same. In this tutorial, we will be utilizing this relationship between time, resistance, and capacitance to determine the value of the capacitor while utilizing the BeagleBone Black to both drive and measure the circuit. This tutorial will also be utilizing the BeagleBone’s ability to run python as the interface for calculating and displaying the information.

If you wish to replicate this tutorial, you will need the following items:

(1) - [small breadboard](https://www.digikey.com/en/products/detail/dfrobot/FIT0096-C/7597070) (1) - [BeagleBone Black](https://www.digikey.com/en/products/detail/beagleboard-by-seeed-studio/102110420/12719590) (2) - [1k Ohm resistors](https://www.digikey.com/en/products/detail/yageo/CFR-25JR-52-1K/11974)

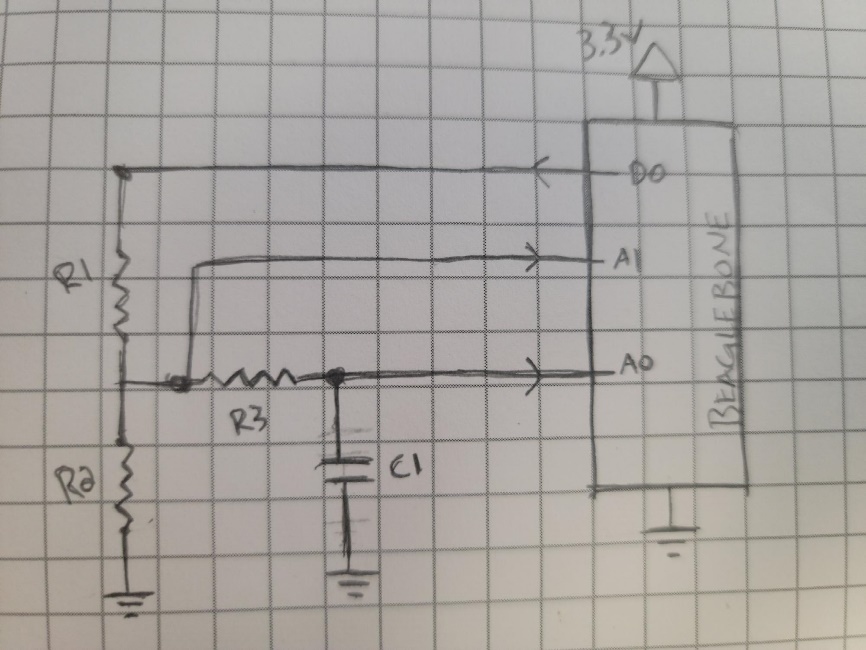
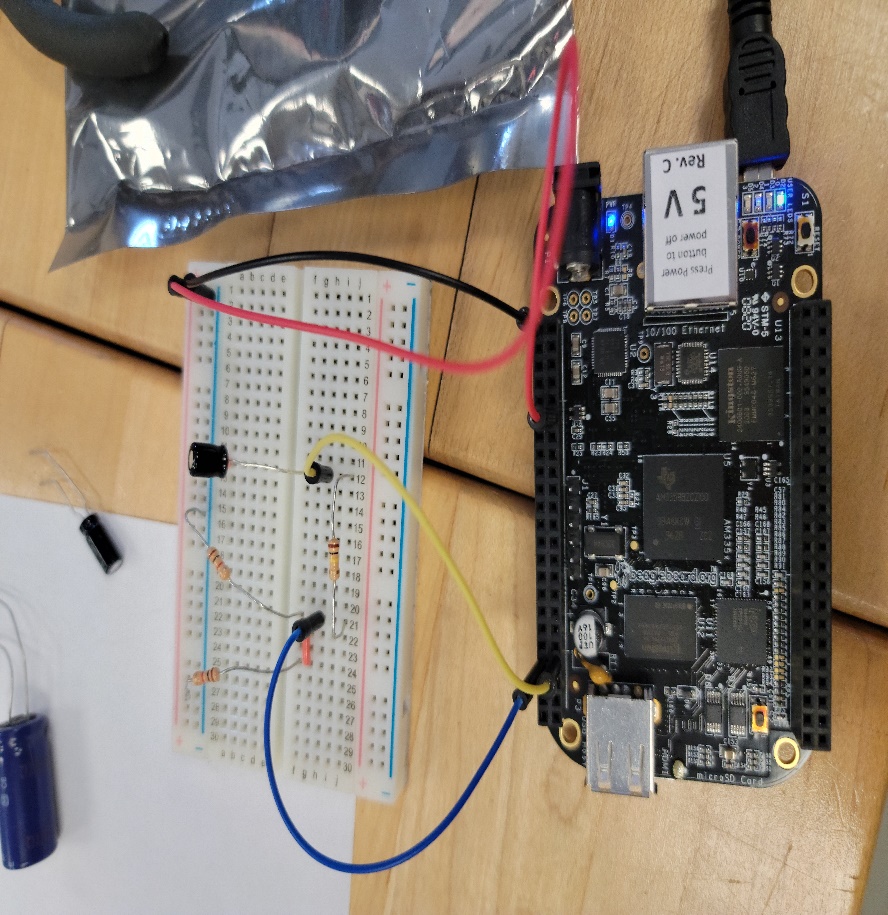
(1) – Resistor of a known size (x) – [Various Capacitors](https://www.digikey.com/en/products/filter/aluminum-electrolytic-capacitors/58?s=N4IgjCBcpgnAHLKoDGUBmBDANgZwKYA0IA9lANogAMIAugL7EBMVALEtCGpFnkaRXAACAK0AxEMTAA6JqIlTpAVnmSQc8WqazVzaQHZdIAMxHj005uKsjrA0ZVWQS6QDYj7p6%2Bnwj%2B5Ua%2BTvA6TmBURmAaCuCOMWCe8UExLEZMYGnRWjZOTHFahrn%2B%2BcTGEU7GljHGJSaJasawtuUxrFVqrIWtTU5KLWpKGb3txEr1xK79E1kTOTGuyWr6MyD6tfBTIPAr8ItSVJvhe%2BDhkalhTHNqYGWRrIestWB9kZOvQ-H6hxuRiJGwmxYgNOuQSmUBt1ylTS9xhI3UL1ybyKgJ%2BFQOZhB1XO2KupUh1VhFWR1X0eJMaOqAOam1YWI6BI6RNaiNaJI6XwcGN69NGzIGrIG7NGnN61K8OLUrkFE2FIAWmy%2Biqox2Wiplq0pag2m22uo18Dl8FFMVgvJAsA14W5EloxAADgAXKAgACqADsAJaOgDy6AAsvhMLgAK4AJ3wIEYIAAtExkFwoI6wyH%2BGRIJQlGokAxiDGOKgkym04IIAxo-GMyAAMKYe2YFDekhh3BCOj0IA) (x) – [Various Jumpers](https://www.digikey.com/en/products/detail/global-specialties/WK-3/5231342)

## The Idea

A capacitor’s charge follows a specific equation – Vc = Vs \* (1-e(-t/RC)). Quickly running through this we note that this says the voltage of the capacitor is equal to the source voltage multiplied by one minus Euler’s number to the power of negative ones multiplied to time divided by the product of resistance and capacitance. Phew, what a mouthful! The are a few key takeaways from this, though. The product of resistance and capacitance (RC) is called tau. Because of the logarithmic nature of this formula (research Euler’s number if you’re curious about this), the capacitor never reaches 100% of the source voltage. We instead say that a capacitor is fully charged at 5 tau, or 5 times RC. On the logarithmic scale, 5 tau equals 99.6%.

When we look at the equation, we can control the voltages so we do not have to worry about those (you will find out later why source voltage will be approximately half of 3.3v and why we want our capacitor voltage to be just under that). With the understanding that we will consider the capacitor full at 5 tau, we don’t have to worry about the (1-e(x)) portion of the equation since those are constants and not variables. So that leaves us with time, resistance, and capacitance (t/RC). Since this project is all about trying to find the value of a capacitor, we’re going to choose our resistor and input that as part of the calculation. Time is also a variable, but one that we do not know immediately. To solve for this, we will measure how long it takes for the capacitor to reach 99.6% (5 tau) of the source voltage. The best way to do this would be to create a timer through attach a time stamp to a digital pin sending the voltage to the circuit and then creating another timestamp when the voltage over the capacitor reaches the 99.6%, and then subtracting the original start time from the final time. This will give us the time it takes to charge. Once we have the time and the resistance, we can then solve for capacitance by rearranging the equation at 5 tau into C = t /(5 \* R). The next step is to build the simple circuit that can handle this.

## The Circuit

The first thing to consider when building this circuit is that the digital pin we use to send voltage to the circuit operates off of 3.3v, whereas the analog read pins we will be using to measure can only accept 1.8v. In essence, we must split the voltage to ensure that we do burn out the analog pins reading the voltages. We can do this using two resistors of the same value, essentially cutting the 3.3v into 1.6v, falling below the 1.8v maximum of the analog pins. The issue we face by doing this, however, is tolerance. If we use resistors of a wide tolerance, we can’t know if the voltage is getting split evenly between them, rendering our reference for source voltage unreliable. The workaround for this, however, is to simply not assume the voltage at the junction is 1.6v and instead actively read the voltage at the junction. If we do this, we get a circuit schematic that looks like the following:   
  
As long as resistors R1 and R2 are of the same value, the 3.3v splits into a safe voltage for the analog pins to read. R3 splits off the junction of R1 and R2, as Kirchhoff’s Voltage Law tells us that the voltage between R3 and ground will be the same as the voltage of R2 and ground. We consider this to be our source voltage, and pin A1 connects to this junction to measure that. The A0 pin connects to the junction of R3 and C1 to measure the voltage over the capacitor. When the reading on A0 reaches 99.6% the value of A1, the capacitor is charged.

In this schematic, I used D0, A1, and A0 as representations of the actual pins. When you are building the circuit, you can use whatever digital output and analog input pins work best for you provided your code reflects this. In my build, I use Row 9 for all for pins. P9\_1 was used for my ground pin, P9\_12 was my digital output, and P9\_39 and P9\_40 served as my analog read pins. It is a simple circuit, but avoid the temptation to put all of the parts in close proximity – it’s much easier to read and test when there is plenty of space to work with.

## The Code

The code was written using the nano command in BeagleBone Black to create a python script. Although this could have been done in C or directly using the shell, I felt that python would be the most appropriate for ease of input without having to compile. Also, honestly, I needed to practice my python and this was an easy way for me to do it. Regardless, this code utilizes the Adafruit\_BBIO and Sleep libraries, so if you are using the code directly be sure that you have these available. Lastly, feel free to alter the code as you wish to fine tune your experiences.

*import Adafruit\_BBIO.GPIO as GPIO*

*import Adafruit\_BBIO.ADC as ADC*

*import time*

*ADC.setup()*

*outPin = "P9\_12"*

*junctionPin = "P9\_39"*

*capPin = "P9\_40"*

*uf = 1000000 // THESE LINES ARE USED LATER TO DETERMINE WHAT UNITS*

*nf = 1000000000 //YOU WANT TO DISPLAY*

*GPIO.setup(outPin,GPIO.OUT)*

*GPIO.output(outPin,GPIO.LOW) //THIS ENSURES PIN IS LOW UNTIL TIMER IS STARTED*

*while(TRUE):*

*print("NOTE:SMALLER CAPS SHOULD HAVE LARGE RESISTORS AND VICE VERSA")*

*resVal = int(input("Please enter numeric value of resistor: "))*

*print("Please input the capacitor into the circuit")*

*Buf = input("Enter 1 when ready to test: ")*

*print("Note: Large Caps with large resistor values will take long to test.")*

*GPUO.output(outPin.GPIO.HIGH)*

*timStart\_1 = time.time()*

*i=1;*

*while(True):*

*juncVolt = ADC.read(junctionPin)*

*capVolt = ADC.read(capPin)*

*i=i+1*

*if capVolt >= juncVolt \* .996:*

*timEnd\_1 = time.time()*

*diff = timEnd\_1 - timStart\_1*

*capValue = ((diff/5\*resVal))\*uf)*

*break*

*elif i>=10000: //THIS INDICATES THAT THE PROGRAM IS STILL ACTIVE*

*print("Charging - Please Wait")*

*i=0*

*GPIO.output(outPin,GPIO.LOW)*

*discharge = diff\*1.1*

*print("Capacitor Value is approximately %d uF"%capValue)*

*print(".")*

*print("..")*

*print("...")*

*print("Please hold: Safety discharging in progress")*

*time.sleep(discharge)*

*print("Capacitor discharged)*

*GPIO.cleanup()*

*break*

From here, you will want to run the program. Using Linux, type python [FileName] and follow the instructions!