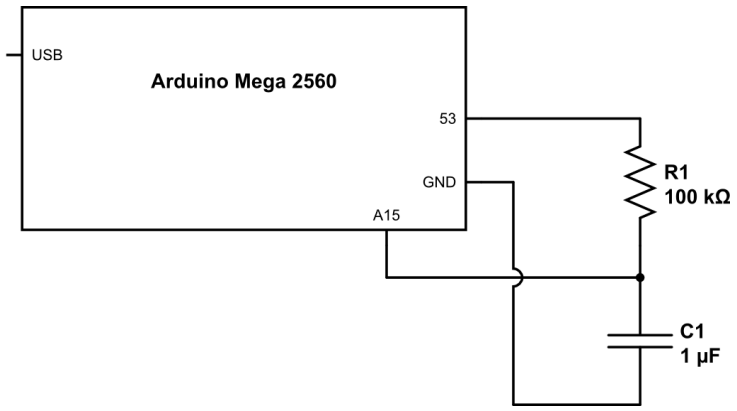
Lab 3 Arduino Capacitance Meter

Objectives:

- Build and investigate an RC discharge circuit.
- Use an Arduino to measure the voltage across a discharging capacitor as a function of time.
- Analyze the data to extract a measured value for the capacitance.

I. Using an unpowered breadboard, your Arduino, and the components listed, construct the circuit shown below. Try to keep the wires reasonably short.



Pin 53 is a digital pin, which we will configure as an output, capable of giving either +5 V (HIGH) or 0 V (LOW). Pin A15 is one of the analog input pins and is capable of measuring analog voltages (0 – 5V).

II. Now we need to construct the code to run the experiment. In pseudo-code, it might look like this:

1. Declare variables
2. `void setup()` – This block tells the Arduino to configure pin 53 as an output and begins the serial connection for displaying text on the serial monitor
3. `void loop()` – This block does all the work, and repeats endlessly until we remove power from the Arduino. It needs to:
    a. Set pin 53 to HIGH to charge the capacitor (perhaps with a small delay to let it fully charge.
    b. Set pin 53 to LOW to begin the discharge
    c. Begin taking time measurements and voltage measurements, storing the values in arrays.
    d. Analyze the data to extract the value of the capacitance.
    e. Print the results to the serial monitor.

Here are some helpful points for each of the elements of the pseudo-code above:

1. Use these keywords to declare variables of the given type.

Keyword	Type
int	Integer, from –32767 to 32767
float	Floating point real number, single precision
double	Same as float on our board
long int	Long integer, +/– 2,147,483,647
unsigned long int	Non-negative long integer

For more data types and other useful programming info, check the Arduino language reference at https://www.arduino.cc/en/Reference/HomePage

2. For our little program, `setup()` only needs to do two things – check out `Serial.begin()` and `pinMode()` in the language reference. For `Serial.begin()`, the only parameter is the speed, so choose at least 9600.

3. Here some hints for each element in `loop()`:
   a) Use `digitalWrite()` to set pin 53 to HIGH. Use `delay(n)` to set a delay of n milliseconds.
   b) Use `digitalWrite()` again.
   c) Use `millis()` to get the current time in ms (the clock rolls over, so it's not connected to true clock time, but successive measurements will give the correct time difference) and `analogRead()` to get the voltage on pin A15. This has to be done in a loop (try a `for` loop), so you have to pick the total number of points to take and the approximate time spacing between points (use `delay()` to set the approximate time delay between points). You can use the index of the loop as the index for your voltage and time arrays. Arrays in C start at index 0.
   d) Now you have to convert the voltage measurements from integer values (0 – 1023) to real voltage values. That range of integers corresponds to the real voltage range 0 – 5 V. Then take the natural log of the real voltage values using `log()`. Just remember that `log(0)` is undefined. A plot of the natural log values (y-axis) vs the time values (x-axis) should yield a straight line with slope equal to –1/RC. To find the best fit slope, you can use this formula: $m = \dfrac{N\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}$. N is the number of data pairs. To find the error in this number due to the scatter in the data, use these formulas:

$$R = \frac{N\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{\sqrt{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}\sqrt{N\sum_{i=1}^{N} y_i^2 - \left(\sum_{i=1}^{N} y_i\right)^2}} \quad \text{and} \quad \sigma_m = \frac{|m|}{R}\sqrt{\frac{1-R^2}{N-2}}$$

   e) Look at `Serial.print()` and `Serial.println()` to print your results (and any debugging notes you need) to the serial monitor.