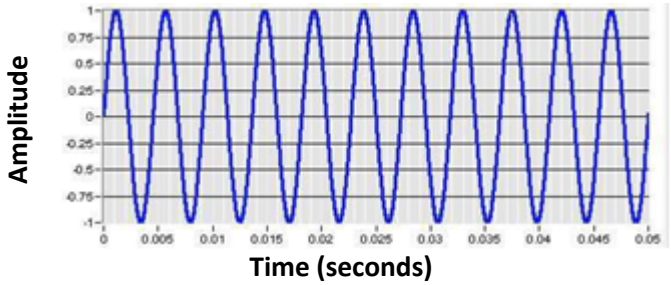
LAB 01

In this lab we will do the following:

1. View and capture voltage produced by your phone (or computer)
2. Build an analog-to-digital converter using an "Arduino" microcontroller

Background:

In lecture, you learned that sound is a vibration of particles in the air. A microphone is a device (transducer) that converts that vibration into voltage. If we sing a pure tone into a microphone and plot the output of the microphone with time, we'd see a sinusoidal pattern as shown in the figure below. We note the amplitude of the voltage is changing as time progresses. We recall that this voltage is continuous with time.



In lecture, you also learned that in order to "digitize" this signal, we need to break it up into discrete segments, that the process of digitization is taking a continuous signal and converting it into a discrete signal that can then be manipulated by a digital computer. A device known as an analog-to-digital converter (ADC or A2D) is the component needed to digitize our signal. An A2D will have a sampling rate: meaning how the x-axis (time) will get partitioned. An A2D will have a quantization value, meaning how the y-axis (voltage) will get partitioned. The A2D will then take a "sample" or measurement of the voltage at regular intervals and produce a digital representation of those samples.

In lab today, we'll take in audio using a microphone and then build our own A2D using an Arduino microcontroller to digitize a continuous audio signal!

As we go to lab, you will do things you've never done before. This is a necessary part of learning. Not everything you do will work perfectly on the first try. This is expected. Included here is an excerpt from Henry Petroski's *Success through Failure: The Paradox of Design* that highlights how we should not fear failure but embrace it and learn from it in design and in our educational experiences. As he quotes Mason, you should "learn to disassociate failures resulting from their [your] attempt to succeed from being failures themselves [yourselves]."

64 SUCCESS THROUGH FAILURE

Prototyping can be viewed as a "sort of three-dimensional sketchpad," with the prototype enabling potential backers and users to see the invention as a tangible thing. Dennis Boyle, a studio leader at the design firm IDEO, further sees the construction of "rapid and rough prototypes" as a means of identifying problems early in the design process, when they are less costly to correct. According to Boyle, if a "project is not generating masses of prototypes, including many that clearly won't fly, something is seriously wrong." The creed at IDEO is thus "Fail early, fail often."[28]

The sentiment is not new, nor is it unique to design. According to Samuel Smiles, the Victorian biographer of engineers who wrote about their overcoming great personal and technical challenges, "We learn wisdom from failure much more than from success. We often discover what will do by finding out what will not do; and probably he who never made a mistake never made a discovery."[29] The American poet James Russell Lowell conveyed a cognate idea by means of a simile: "Mishaps are like knives, that either serve us or cut us, as we grasp them by the blade or the handle."[30] What was true for engineers and poets—and everyone—in the nineteenth century remains true for everyone today. As the design critic Ralph Caplan has written, "The more things change, the more we stay the same."[31]

How individuals react to failure separates leaders from followers, true designers from mere users of things. Professor Jack Matson of Pennsylvania State University believes so strongly in the role of failure in design that he expects students in his Innovative Engineering Design course to fail in order to pass. The course, nicknamed "Failure 101," requires students "to

Success and Failure in Design 65

build and attempt to sell outlandish and frequently useless products," like a hand-held barbecue pit. The most successful students in the course are those who take the most risks and so fail the most. Matson hopes to get them to the point "where students learn to disassociate failures resulting from their attempts to succeed from being failures themselves." He believes that, "Innovation requires that you go beyond the known into the unknown, where there might be trap doors and blind alleys. You've got to map the unknown. You map it by making mistakes." It is not unlike being blindfolded in a labyrinth. Smacking into walls may signal a misstep, but the sum of those missteps defines an outline of the maze. The quicker more mistakes are made, the quicker the maze is mapped. Matson is an advocate of "fast failure."[32]

Whether fast or slow, failure and its avoidance have always been central to the development of designs and their far-reaching influence. Though often considered apocryphal, the familiar story of the standard railroad gauge of 4 feet 8½ inches serves as an example. This odd distance between rails is believed to be rooted in ancient times, when all Roman war chariots came to have that same wheel spacing, which is said to have been established to be no wider than the rear ends of the two horses that pulled a chariot. This width, which prevailed throughout the Roman Empire, ensured that the horses would not pull a too-wide wagon through an opening only wide enough for them. As the standardized chariots ranged throughout the empire, they wore deeper and deeper ruts in the Roman roads, including those in England. So, the development of English wagons incorporated the same wheel width, lest their wheels not ride in the ruts, the path of least resistance and least

ESE 150 – Lab 01: Sampling and Quantizing Audio Signals

Lab – Prelab:

1. What is the relationship between Sampling Frequency and Delay between samples? [provide a symbolic equation] For context, read through Section 3 of the Lab.

2. Complete the following table (you will need these values for the later part of Section 3), assume that the time in between each sample is the delay plus 18 microseconds needed for the analogRead() function.

Frequency	**Delay**
500Hz	
1000Hz	
5000Hz	
38000Hz	
55000Hz	

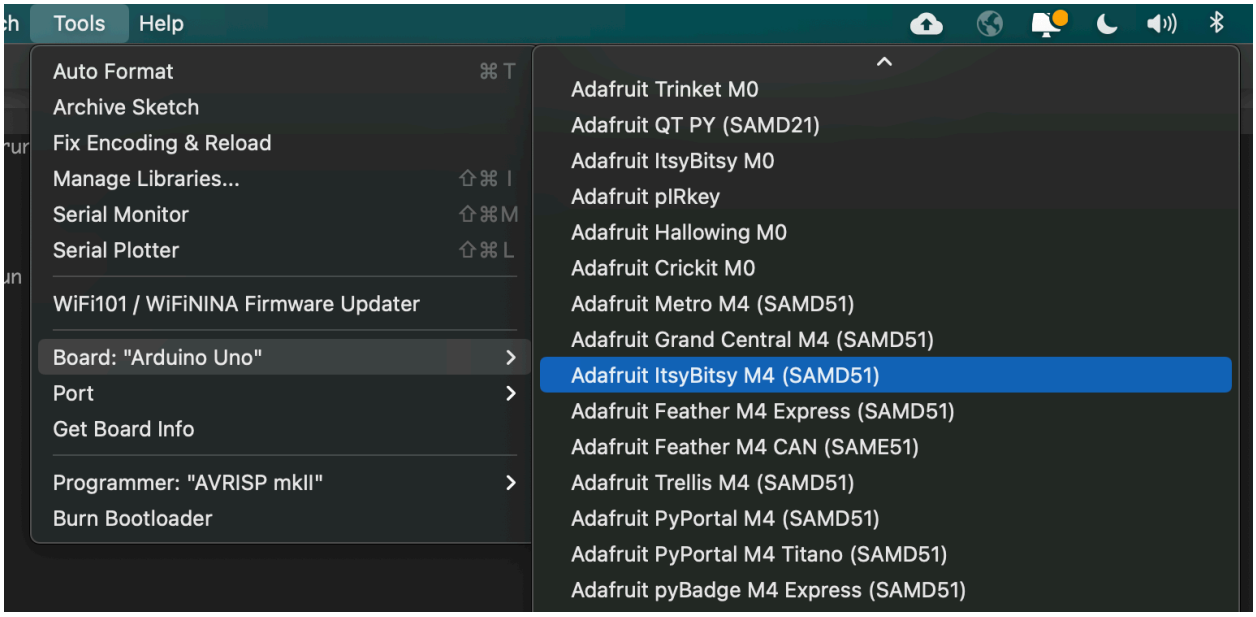
3. Arduino IDE download and setup:

    a. https://learn.adafruit.com/introducing-adafruit-itsybitsy-m4/setup (Complete up until Blink)

        i. You won't be able to run Blink portion until after you get your Lab Kit.

        ii. Once you do have a kit, it is a good initial test.

    b. After you have successfully downloaded all of the necessary board packages, make sure to switch the board type in the Arduino IDE to the "Adafruit ItsyBitsy M4 (SAMD 151)" by navigating to Tools -> Board -> Adafruit ItsyBitsy M4



4. Read through Section 3. Use the Arduino Language reference guide (https://www.arduino.cc/reference/en/) as necessary to understand the operation of the code in Section 3.

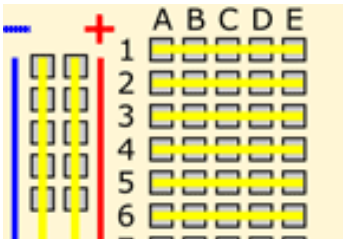
When lab starts, compare your answers with your assigned partner. If your answers differ, discuss amongst yourselves and try to resolve your differences.

Early during the lab session, a TA will check you off on prelab. Go ahead and start working on the lab. It will take some time for the TAs to get around to all the groups.

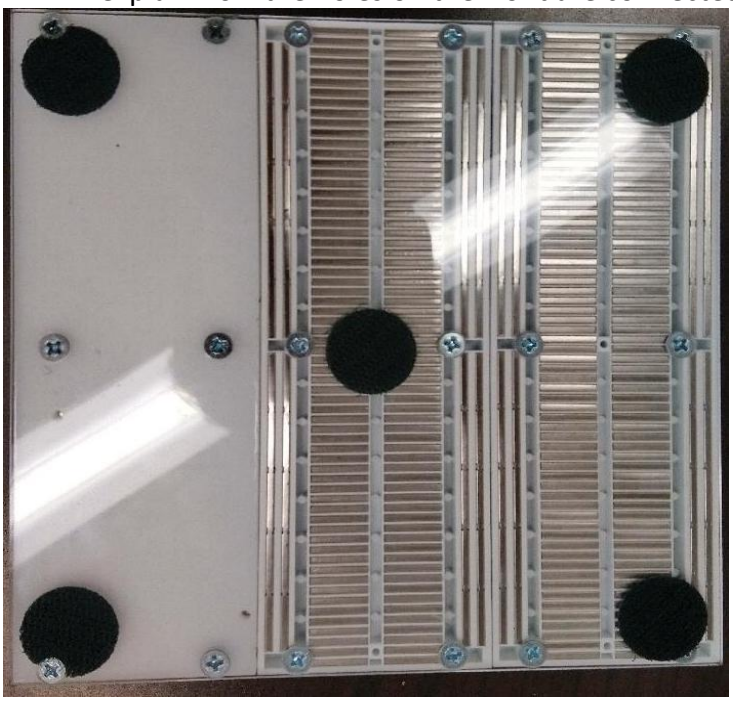
Lab Procedure:

Lab – Section 1: Understanding Your Lab Kit

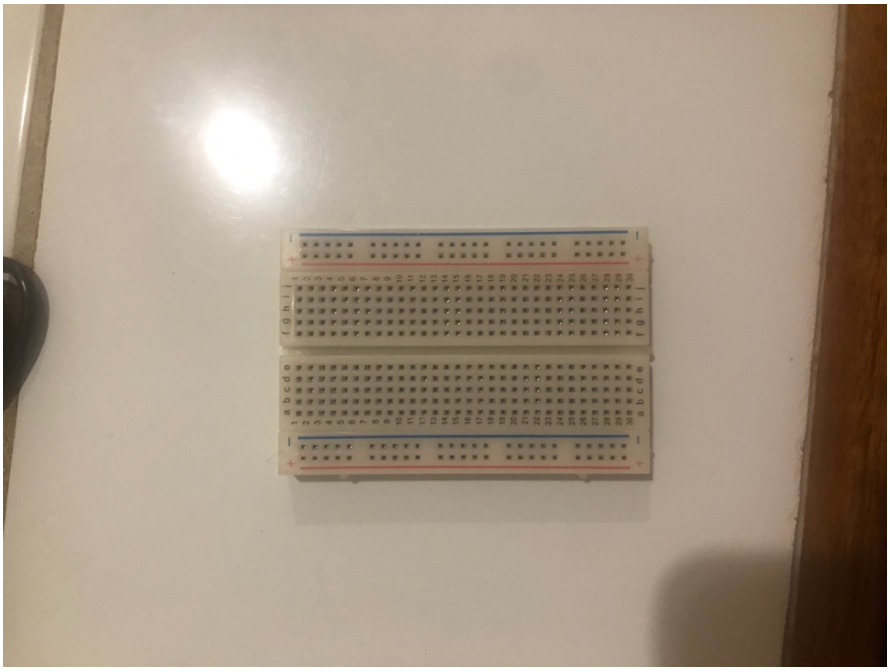
1. Obtain a breadboard (it's a white board with holes in it at your lab kit).
    a. A breadboard allows you to prototype by connecting various electric components when inserted.
    b. Each row is designated with a number (1-30).
    c. The five holes in each row are electrically connected.
    d. The long columns on the sides of the breadboard, marked by a + or -, are the power rails of the board, where each column is electrically connected.



    e. Following is a view of the back of a different (larger) breadboard that may help explain how the holes on the front are connected together.



For reference the front of the bread board is:



2. Next, grab the Adafruit Itsy-bitsy
    a. This board will be our micro-controller for the semester!
    b. We can connect various electrical components and your computer to the Itsy-Bitsy:
        i. This allows us to program our board and interface with our digital circuit.
        ii. We will program this board using the Arduino Integrated Development Environment (IDE) which you should have set up during the pre-lab.
    c. Here is a picture of the top of the Itsy-Bitsy:

You should notice *female* wire connectors in addition to the USB type-c connector. The wire connectors will allow us to connect different components to

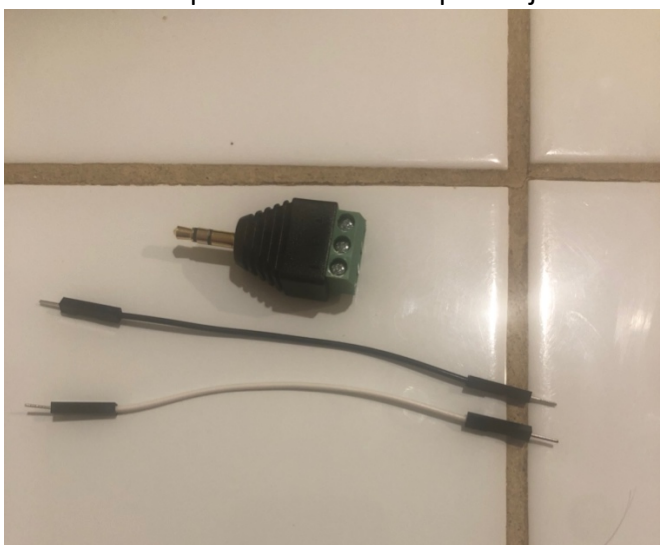
the board, and the USB type-c connector will allow us to connect the board to your computer.

d. Here is picture of the bottom of the Itsy-Bitsy:

You'll notice labelled male pins. This will connect into our breadboard.

3. Grab the USB to micro-USB cord provided in the lab kit
(***If you don't have a STANDARD USB Port on your working computer, please let a TA know)***:

4. Next, grab the male headphone jack, a black wire (common convention used for ground) and any other wire of your choosing:
    a. We will use this headphone jack to sample audio that comes from your computer or phone.
    b. ***If your computer or phone does not have a standard female headphone jack, please let a TA know.***
    c. Below is a picture of the headphone jack and some wires.

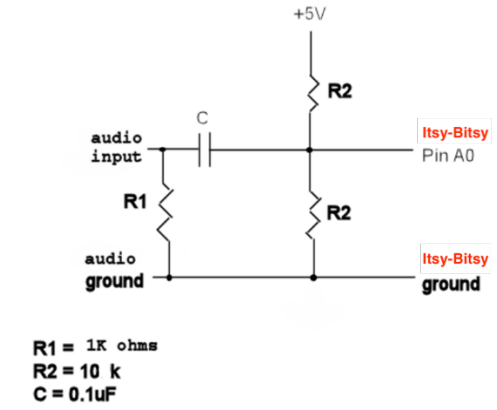


    d. Connect the black wire into the green socket on the headphone jack. This is the socket that is neither Left (L) or Right (R). The symbol you see is often used for ground. Ground is a reference point for any voltage reading. All readings are relative to ground.
    e. Next, connect the non-black wire into either the Left or Right socket. It shouldn't matter which one you choose. This wire will be considered as the audio input.
    f. Finally, connect the headphone jack to your audio source. This could either be your computer or your phone.
    g. So far, you should have something similar to the following picture:

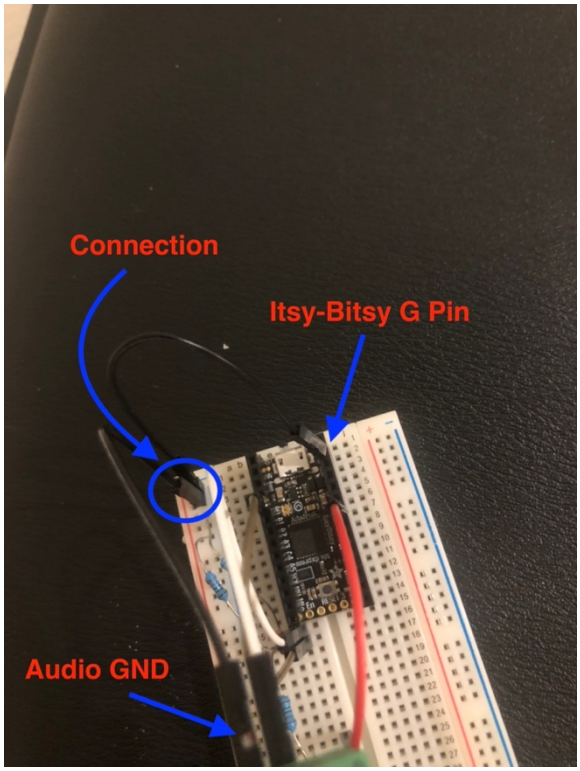
5. Next, grab **2x 10K Ohm**, **1x 1K Ohm** resistor and **1x 100nF** capacitor:
    a. The resistors are the blue objects in the plastic bag. You can determine the value of the resistances by looking at the stamp near the bottom of the holding bands. It is important that you do not lose track of which resistors are which, so that you can reuse them in the future.
    b. The 100nF (aka 0.1uF) capacitor is the red object in the plastic bin with a 104 label.
    c. Here is a picture of the components:

Lab – Section 2: Setting Up our Circuit for Sampling:

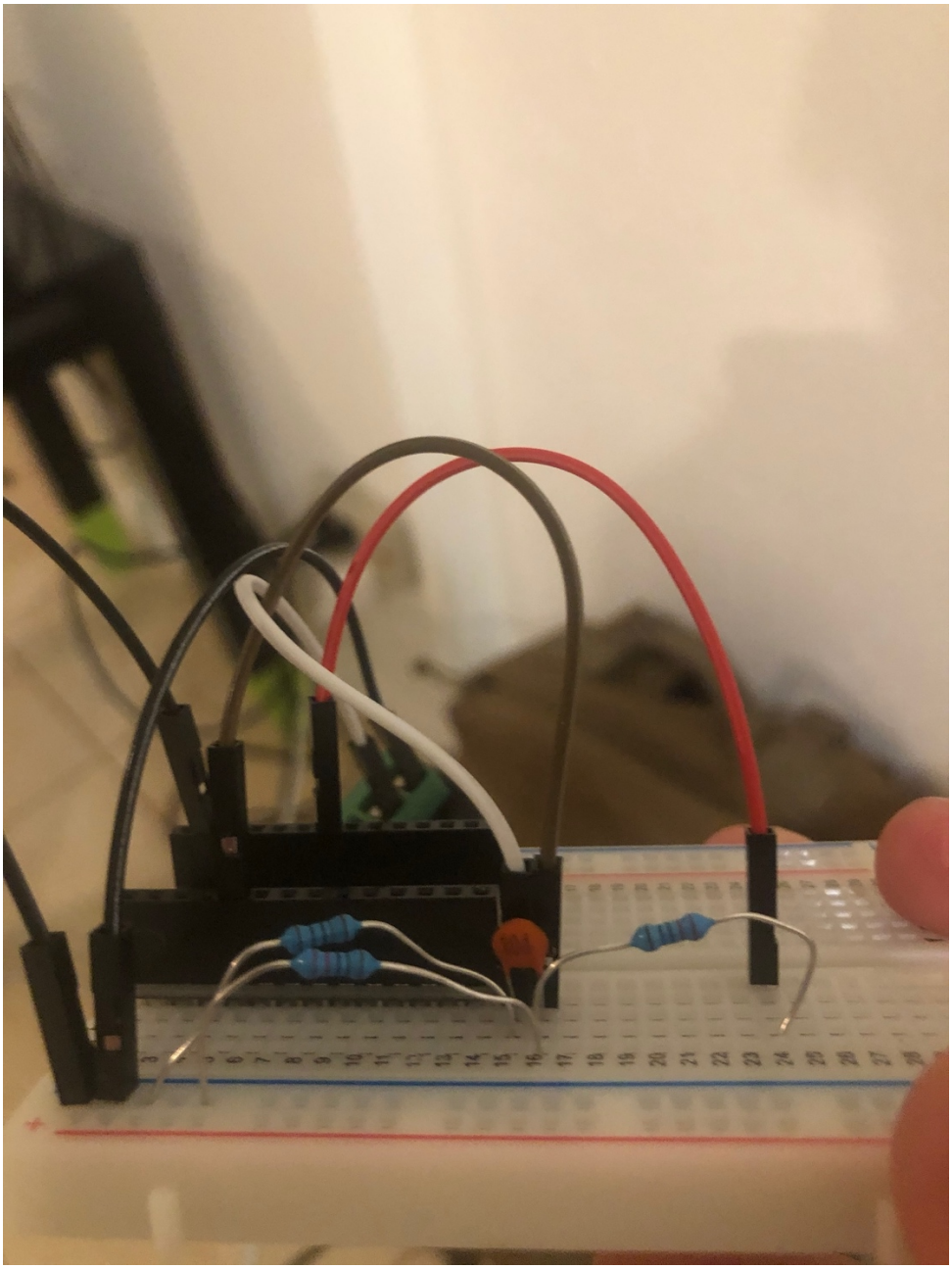
- In this section of the lab you will combine components in the previous section into one final circuit for audio sampling.
- You will effectively be able to *sample* any audio that is played from your device
- You will implement the following circuit:



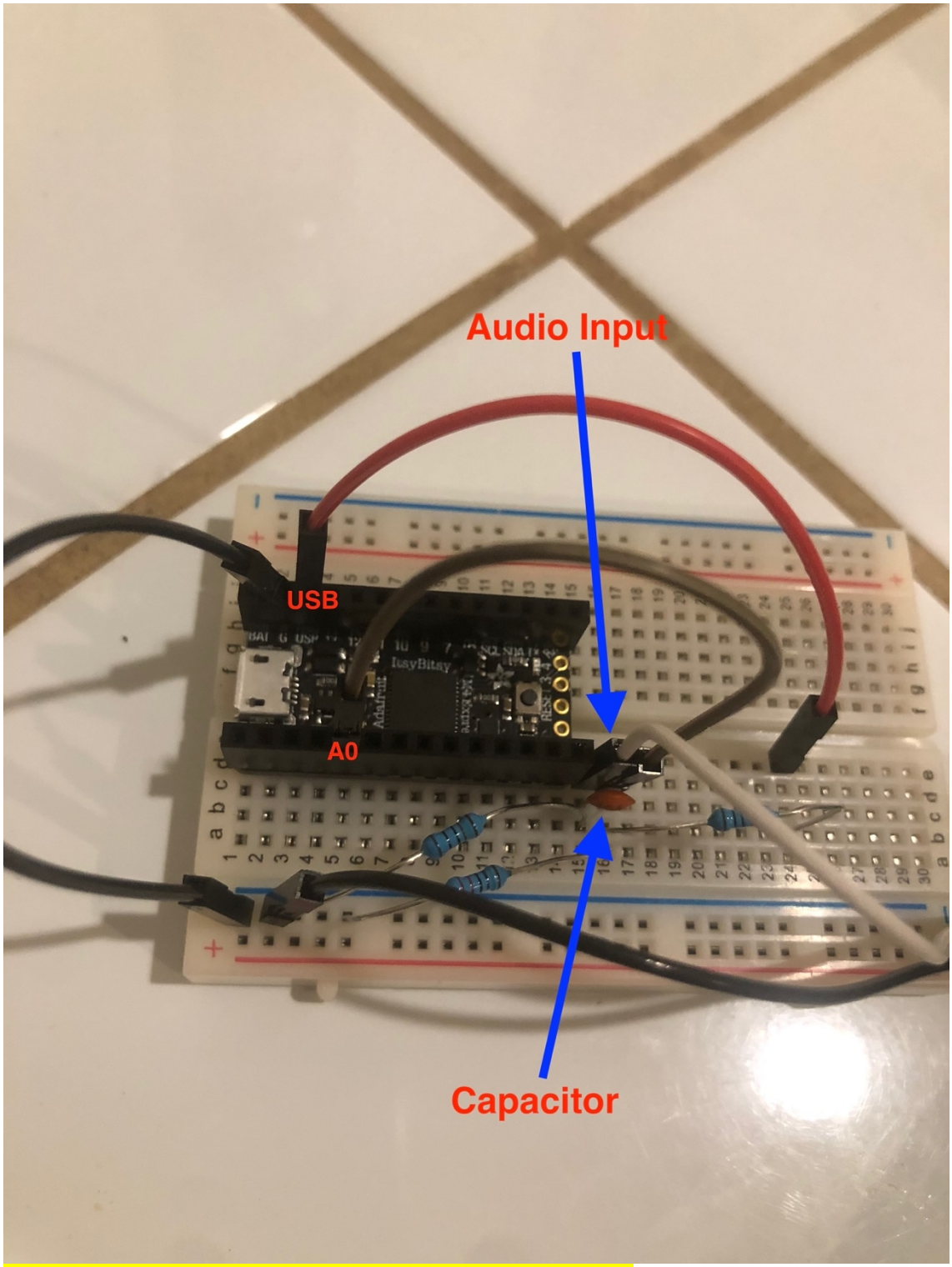
You'll want to have your Itsy-Bitsy straddling both sides of the breadboard. The first step is to connect the Audio ground to the Itsy-Bits ground (pin G). One way to do this is at the negative ground rail on the breadboard:



The next step is to connect the resistors and the capacitors. Both the 1K Ohm and one of the 10K Ohm resistors will connect to the common ground above, but they will go to different ends of the capacitor. It is best to have the capacitor away from any Itsy-Bitsy pins. The other 10K Ohm resistor will connect into the same side of the capacitor as its identical twin. This resistor can go to any other row on the breadboard.



Finally, connect the audio input wire into the same row of the capacitor as the 1kOhm resistor. Connect the other side of the last 10K Ohm resistor to the USB pin of the Itsy-Bitsy, and connect the other side of the capacitor to pin A0. The final product should look like this:



Take a picture of your breadboard circuit for submission!

Lab – Section 3: Sampling a pure tone using the Arduino

The Itsy-Bitsy is a microcontroller. This basically means it a little computer that has lots of input and output and you can control by writing a simple program (we call them sketches on the Arduino platform)

The goal of this section is to:

- Attach the output of the audio jack to the analog input of the Arduino
- Modify a small Arduino sketch to sample the input from the audio jack
- Output the samples taken over time to the Arduino's terminal output window
- Capture the output
- Plot the output

In this section of the lab, we will use the following code to sample the output of the function generator with the Arduino:

```
#define ADC ADC0
#define MAX_RUNS 1
#define MAX_SAMPLES 10000
int incomingAudio[MAX_SAMPLES];
int startTime;
int run = 1;

unsigned long microseconds1;
unsigned long microseconds2;
unsigned long time_elapsed;

void setup() {
  // put your setup code here, to run once:

  Serial.begin(9600) ;
  while(!Serial);  // wait for serial to be ready
  Serial.println("Hello World;");
  run=0;
  pinMode(A0,INPUT_PULLUP);

  ADC->CTRLA.reg &= 0b1111100011111111; // mask PRESCALER bits
  ADC->CTRLA.reg |= ADC_CTRLA_PRESCALER_DIV16;   // divide
Clock by 16 (original was 64)
   // this PRESCALAR does appear to impact sample
rate...smaller divider ... faster sample
  ADC->AVGCTRL.reg = ADC_AVGCTRL_SAMPLENUM_1 |// take 1 sample
                     ADC_AVGCTRL_ADJRES(0x00ul);// adjusting
result by 0
  ADC->SAMPCTRL.reg = 0x00;      // sampling Time Length = 0
```

```
    Serial.begin(9600) ;
  }

 void loop() {
   // put your main code here, to run repeatedly:

     if(run<MAX_RUNS){
       microseconds1 = micros();
       for(int i = 0; i < MAX_SAMPLES; i++) {
         incomingAudio[i] = analogRead(A0); // read input from A0
         delayMicroseconds(10);
    // change line above to change sampling rates
       }
       microseconds2 = micros();
       time_elapsed = (microseconds2 - microseconds1);


       Serial.print("10,000 samples in microseconds: ");
       Serial.println(time_elapsed);
       for(int i = 0; i < MAX_SAMPLES; i++) {
         Serial.println(incomingAudio[i]);
       }
     run++;
     delay(4000);
     }
}
```

You can download an initial version of this c code from the syllabus.

An Arduino program is called a sketch and is based on C/C++. For the specifics of Arduino programming, you can reference the Arduino Language Reference documentation online. The language is split up into structures, values, and functions. Use this resource to understand the code snippet shown above, both to familiarize yourself with the language and to understand the purpose of the code. Note that the Arduino sketch is comprised of these two functions, setup() and loop(). Effectively, the Arduino sketch behaves like this:

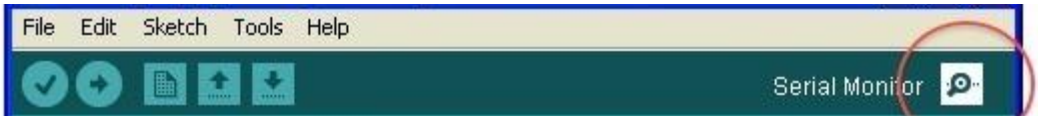
```
main() {

  setup();

  while(true) { loop(); }

}
```

The Arduino sketches are making it easier by just letting you specify the contents of these two functions. The setup() is the one-time code. Here, we use this section to set the speed at

which the Arduino communicates to the host computer over the serial port. The loop() code is the body that is invoked repeatedly, and is usually the main heart of the program. We don't actually want the body code to infinitely repeat, so the run variable is used to ensure the program only executes our logic once. Each time you reset the Arduino (or download new code), it starts over and runs the implicit main routine, re-running setup() then repeatedly executing the contents of loop().

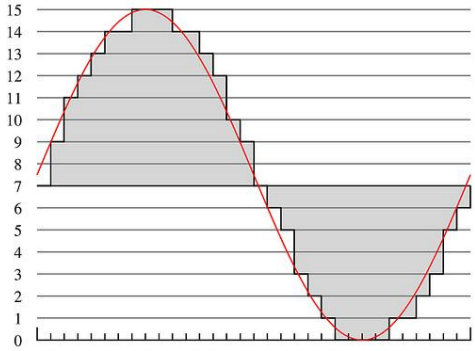
You'll notice that in order to sample the song data, we are conducting an analogRead() on pin A0 and storing the result in an array. Subsequently, we delay by a certain parameter to the delayMicroSeconds() function. ***It Is important to note that in reality, the analogRead() function takes about 18 microseconds to complete. Therefore, the time in between samples*** *(which relates to the sample rate, how?****) is whatever parameter is passed to the delay function plus about 18 microseconds.*** Note that the sketch provided measure the time taken for the sample collection loop and prints the total time taken for 10,000 samples. Use that printed result to calculate the actual sample rate.

1. Connect your Itsy-Bitsy into the USB on your computer using the USB to micro-USB cable. **If you cannot do so, please notify course staff.**
2. Download the set of sample Tones from the link on the syllabus and install them on your phone or other audio playback device.
3. You will complete the rest of procedure for all of the tones that are given.
4. Pause the song, restart and connect the audio jack to your device without disrupting your circuit.
5. Upload the provided sketch to the Itsy-Bitsy
    a. Open the Arduino IDE
    b. Adjust the settings:
        i. Select Tool -> Serial -> #### (the port will differ depending on your machine, but it should be able to detect the Itsy-Bitsy at your relevant port)
        ii. Select Tool -> Board -> Arduino Uno
    c. Paste the provided code into the IDE.
6. Click the upload button  to upload the code to the Itsy-Bitsy. You will see a message "Done Uploading" once the upload has finished. Once you see this, click to open the Serial Monitor. If you are having trouble uploading even though your board is connected, double press the reset button on the board in order to make the Itsy-Bitsy discoverable by your computers.



7. After a few seconds, a sequence of numbers will be printed into the Serial Monitor. Wait until it is finished printing numbers, then select all, and copy and paste into Excel to save the output. You will need to turn in this data.
    a. Make sure you're seeing numbers that range between 0 and 1023 (not all of them, and maybe not all the way to 1023, but many different values in that range); if you're seeing all 0's or only a couple of different values, things probably aren't connected correctly.
    b. We will use this output for the next lab, so make sure to save the file somewhere (like the shared google drive noted below) you will be able to recover it!
    c. Plotting DATA:

1. In this section, you'll use a spreadsheet (e.g., Excel, Numbers, Google Docs spreadsheet (docs.google.com)) to re-construct your original signal from the samples you collected. Use Excel's plotting capability to produce a plot like the below plot ***for each column*** of your data:
    a. The x-axis should just be the row # (aka the sample #).
    b. The y-axis should be the quantization level: 0->1024 (since you are using the Itsy-Bitsy's *10-bit* quantization data*).*
    c. Use a line plot.
    d. Zoom into your plots to show 3 cycles of waveform.
    e. Give each of the 3 plots a title and label the axes with the appropriate units.



2. Next, create & label 3 new plots:
    a. First use your knowledge of the sample rates, and the time between each sample that they imply, to create 3 new columns that contain the time in seconds of each of the samples.
    b. Next use your knowledge of the voltage range of the Itsy-Bitsy (0v-3.3v) and the quantization levels to create 3 new columns that

contain the voltage of each of the samples. Make sure to take note of the sample rates.

c. Use these new columns to create 3 new plots.
d. The x-axis should now be: **TIME.**
e. The y-axis should now be: **VOLTAGE**
f. Zoom into your plots to show 3 cycles of waveform.
g. Give the plot a title and label the axes with the appropriate units.

3. How close does each plot look to a sine wave? Explain the difference between the plots based on what we've covered so far in the course. [1 paragraph]

8. Modify the sketch to sample at 500Hz. You will need to change the delayMicroseconds function. What is the relationship between sampling frequency and delay (Prelab 1)?
9. Repeat steps 6-7 (page 16) for the 300 Hz wave with the modified code and save the sketch and output. You will need to turn in the modified sketch and data.
    a. Close the Serial Monitor between runs to clear it.
    b. To rerun the sampling code, press the reset button on your Itsy-Bitsy.
10. Modify the sketch to sample at 5000Hz, run on the 300Hz wave, and save the sketch and output. You will need to turn in the modified sketch and data.
11. Before leaving lab, show your data sets (all 3 waves at the default sampling, 300 Hz at 500Hz and 5000Hz sampling) and plots 6c1 and 6c2) to a TA and answer a few questions.
    This is the Lab Exit Check-off.
12. Share your data: *before leaving lab*, make sure both partners have access to the data collected and sketches saved. One way to do this is to setup a shared Google Drive.

PostLab

1. Assuming that no changes are made to the above code, (delay of 10 microseconds), what is the sample rate?

2. Were you able to match the frequency of the monotone waves? Describe how.

3. Plot the 500 Hz and 5000 Hz sampled versions of the 300 Hz wave (if you didn't already plot them in lab).
   a. Describe how plots relate to the plot from the original sample rate
   b. Check the frequencies – do they all have the same frequencies? If not, what frequencies does each have?

HOW TO TURN IN ANSWER TO THE LAB:

- Answer the prelab questions, assemble the data requested, and answer the questions in the lab.
- Upload a word document or PDF containing your informal lab report including
  - Partner's name
  - Prelab answers
  - Wave Plots (highlighted)
  - Arduino sketches (Section 3) (highlighted)
  - Answers to postlab, including plots
  - Make sure all graphs and answers are clearly labelled.
- Separately upload your original sample rate data set samples (3 data sets – 3.6—9) (highlighted)
  - Don't forget to save your data!
- Each lab writeup is individual.
- You can see the grading rubric we are using for the lab on Canvas. Review that to make sure there will be no surprises when your lab is graded.
- Due by Friday 5pm.