ESE 1500 – Lab 01: Sampling and Quantizing Audio Signals

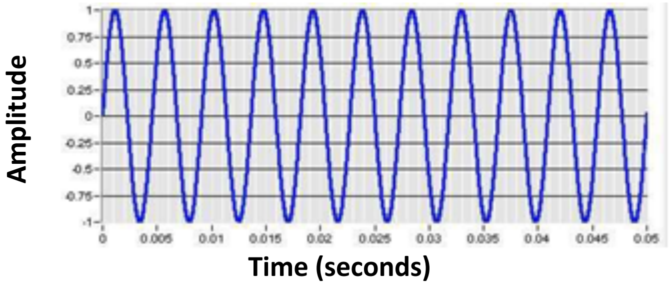
LAB 01

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!

Lab – Prelab:

1. What is the relationship between Sampling Frequency and Delay between samples (period)? [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. Total Delay includes the 18 microseconds delay, while Delay must Add does not include the 18 microseconds delay.

Frequency	**Total Delay between Samples (in microseconds)**	**Delay must add to achieve Total Delay between Samples (in microseconds)**
500Hz		
1000Hz		
5000Hz		
50000Hz		

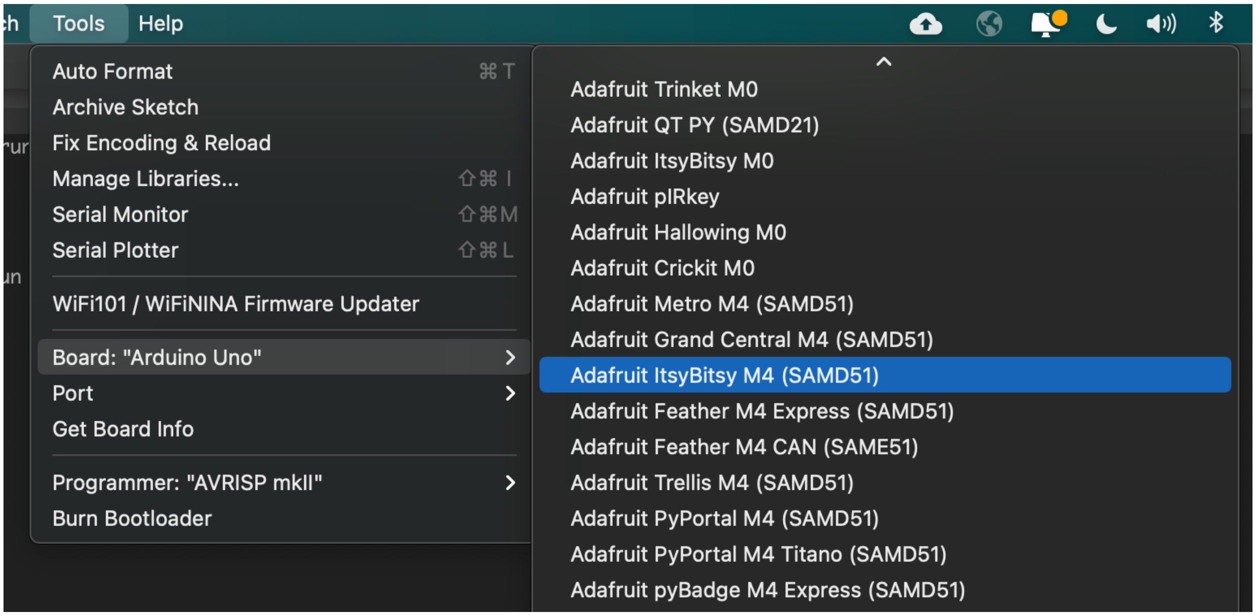
3. Arduino IDE download and setup:

    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


Tools
Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... ⇧⌘I
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Adafruit Trinket M0
Adafruit QT PY (SAMD21)
Adafruit ItsyBitsy M0
Adafruit pIRkey
Adafruit Hallowing M0
Adafruit Crickit M0
Adafruit Metro M4 (SAMD51)
Adafruit Grand Central M4 (SAMD51)
Adafruit ItsyBitsy M4 (SAMD51)
Adafruit Feather M4 Express (SAMD51)
Adafruit Feather M4 CAN (SAME51)
Adafruit Trellis M4 (SAMD51)
Adafruit PyPortal M4 (SAMD51)
Adafruit PyPortal M4 Titano (SAMD51)
Adafruit pyBadge M4 Express (SAMD51)

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.

5. Watch the video about breadboards https://www.youtube.com/watch?v=fq6U5Y14oM4
or read about them: https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/all#why-use-breadboards

6. Complete the canvas prelab 1 quiz with your answers to questions 1 and 2 above.

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

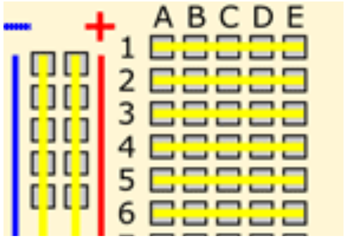
Early during the lab session, a TA will check you off on prelab. Go ahead and start working on the lab. It will take some time for the TAs to get around to all the groups.

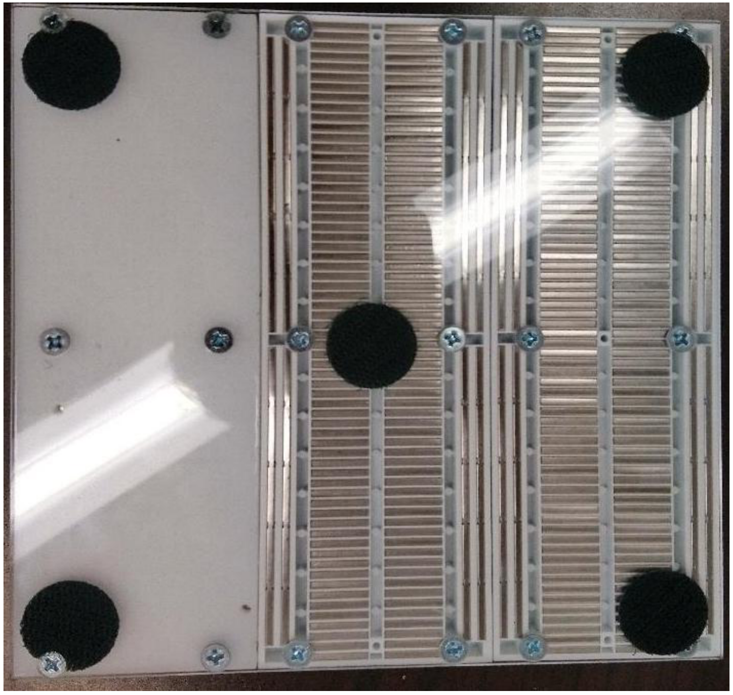
If you have any troubles getting the Arduino and Adafruit tools setup on your laptop, you can use these tools on the computers in Detkin.

Lab Procedure:

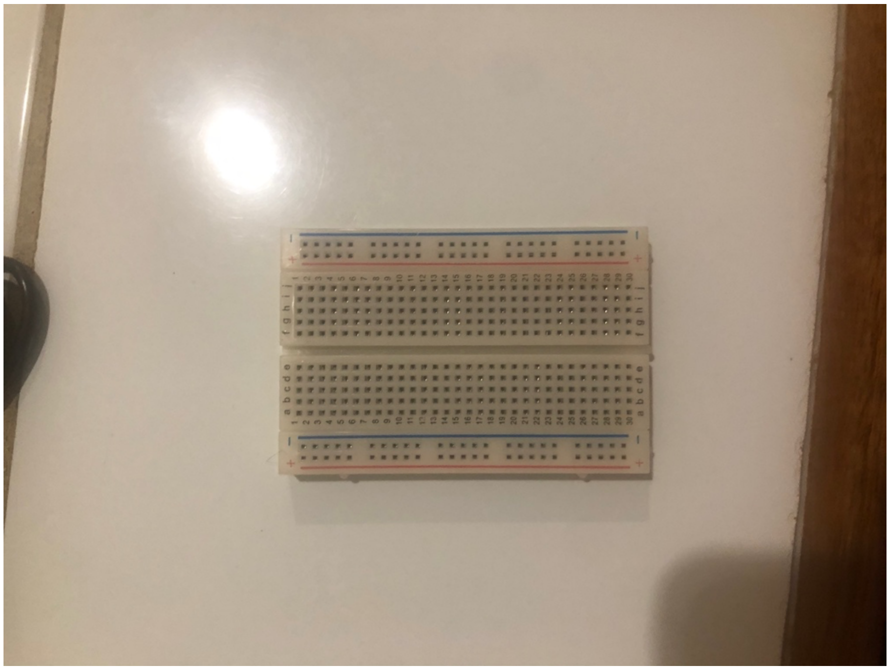
Lab – Section 1: Understanding Your Lab Kit

1. Obtain a breadboard (it's a white board with holes in it in 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.





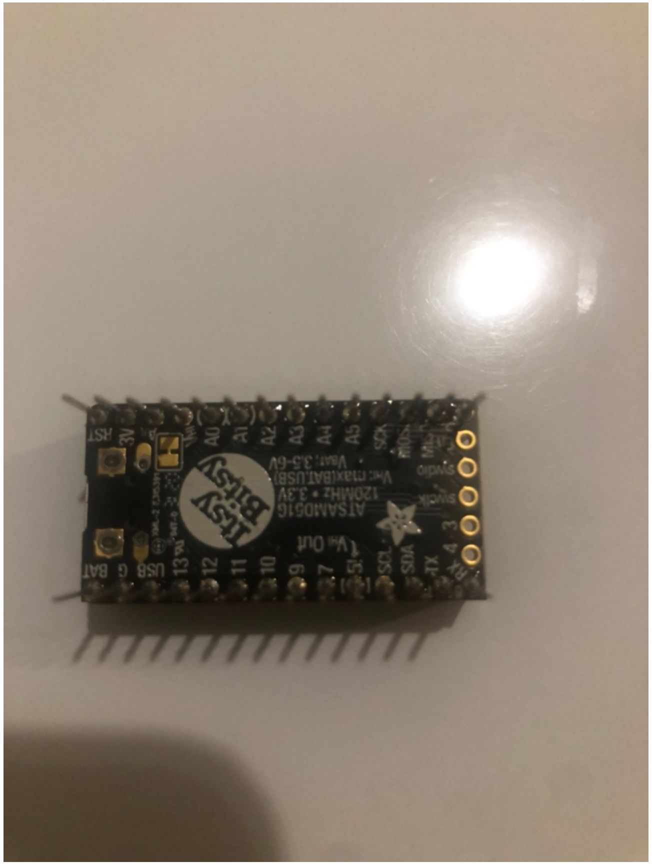
For reference the front of the bread board is:



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 micro USB connector. The wire connectors will allow us to connect different components to the board, and the micro USB connector will allow us to connect the board to your computer.

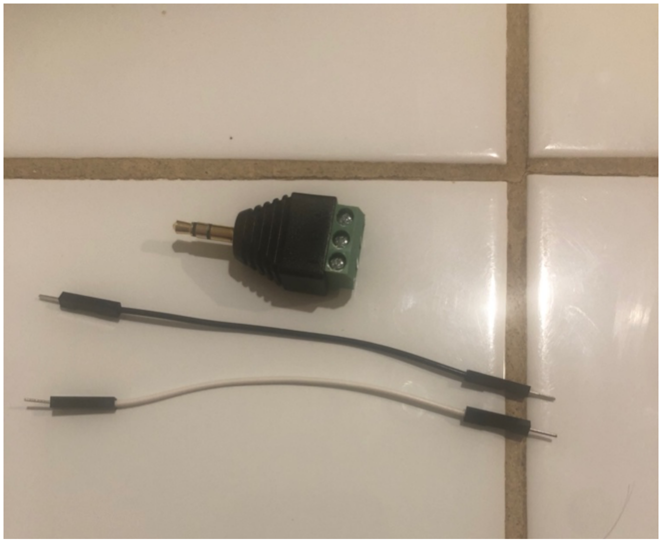
d. Here is picture of the bottom of the Itsy-Bitsy:



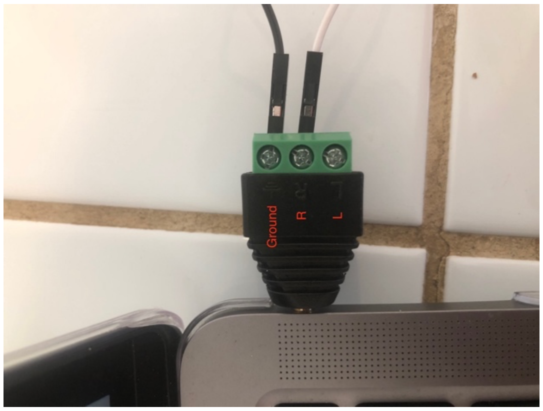
You'll notice labelled male pins. This will connect into our breadboard.

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)***:

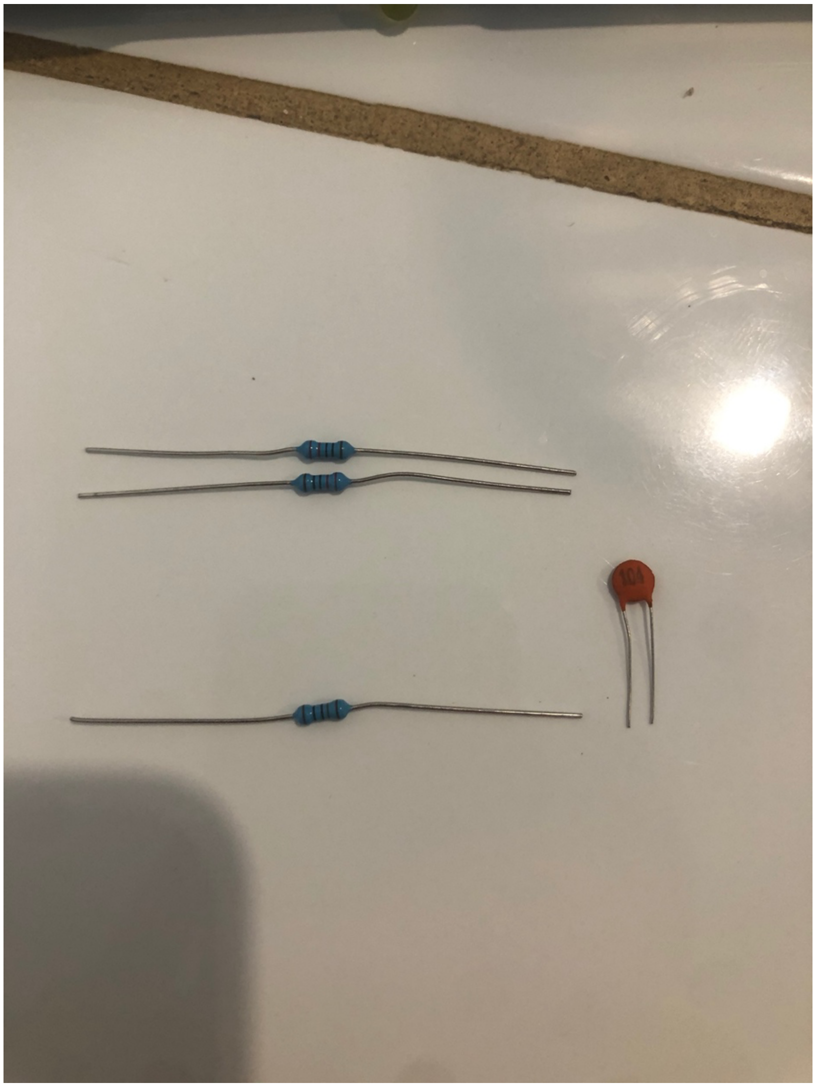
4. Next, grab the male headphone jack, a black wire (common convention used for ground) and any other wire of your choosing:
    a. We will use this headphone jack to sample audio that comes from your laptop, phone, or MP3 player.
        i. Using the same computer as you are using for Zoom as your MP3 player may raise additional complications in audio routing; we recommend you use a different audio source, such as your phone.
    b. ***If your laptop/phone/MP3-player 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 phone or MP3 player.
    g. So far, you should have something similar to the following picture:

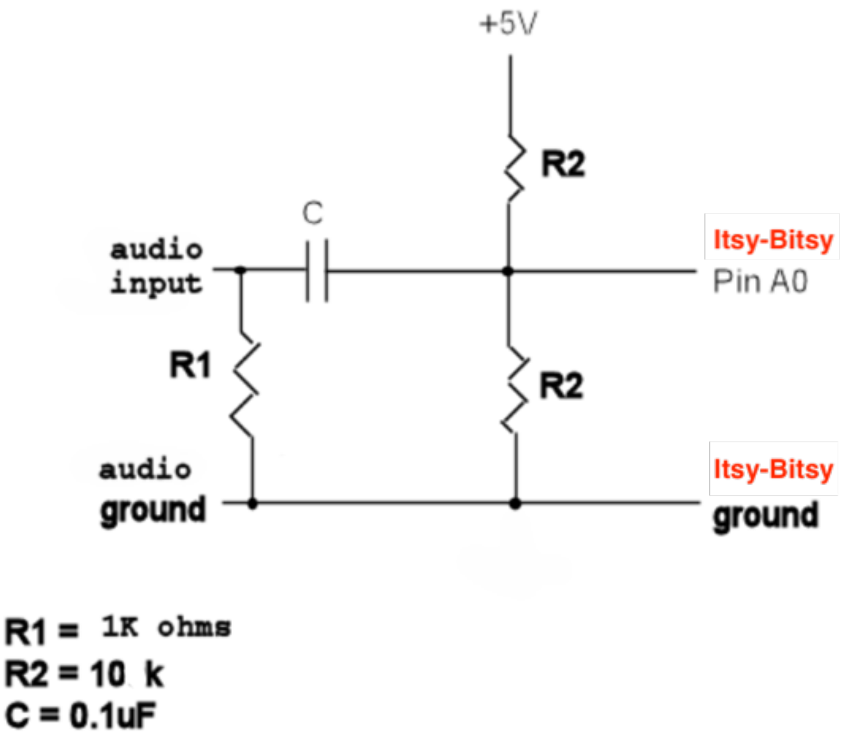


5. Next, grab **2x 10K Ohm**, **1x 1K Ohm** resistor and **1x 100nF** capacitor:
    a. The resistors are the blue objects in the plastic bag. You can determine the value of the resistances by looking at the stamp near the bottom of the holding bands. It is important that you do not lose track of which 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:

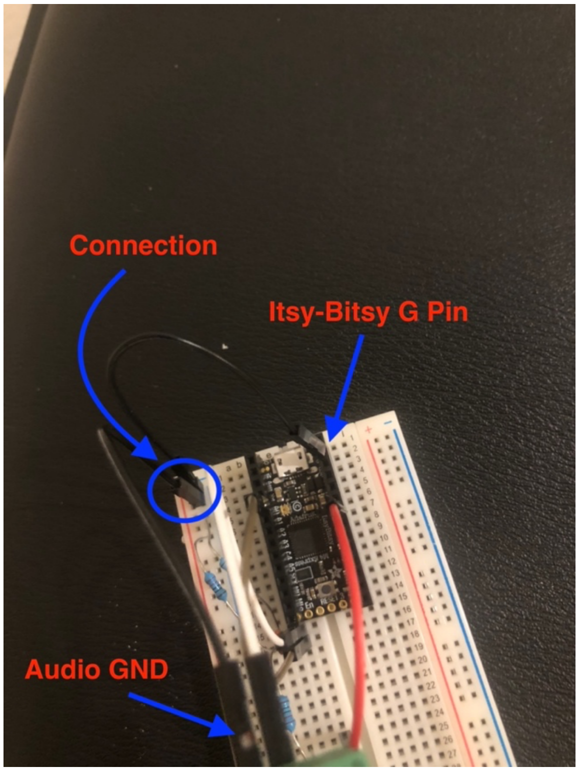

104

<u>Lab – Section 2: Setting Up our Circuit for Sampling:</u>

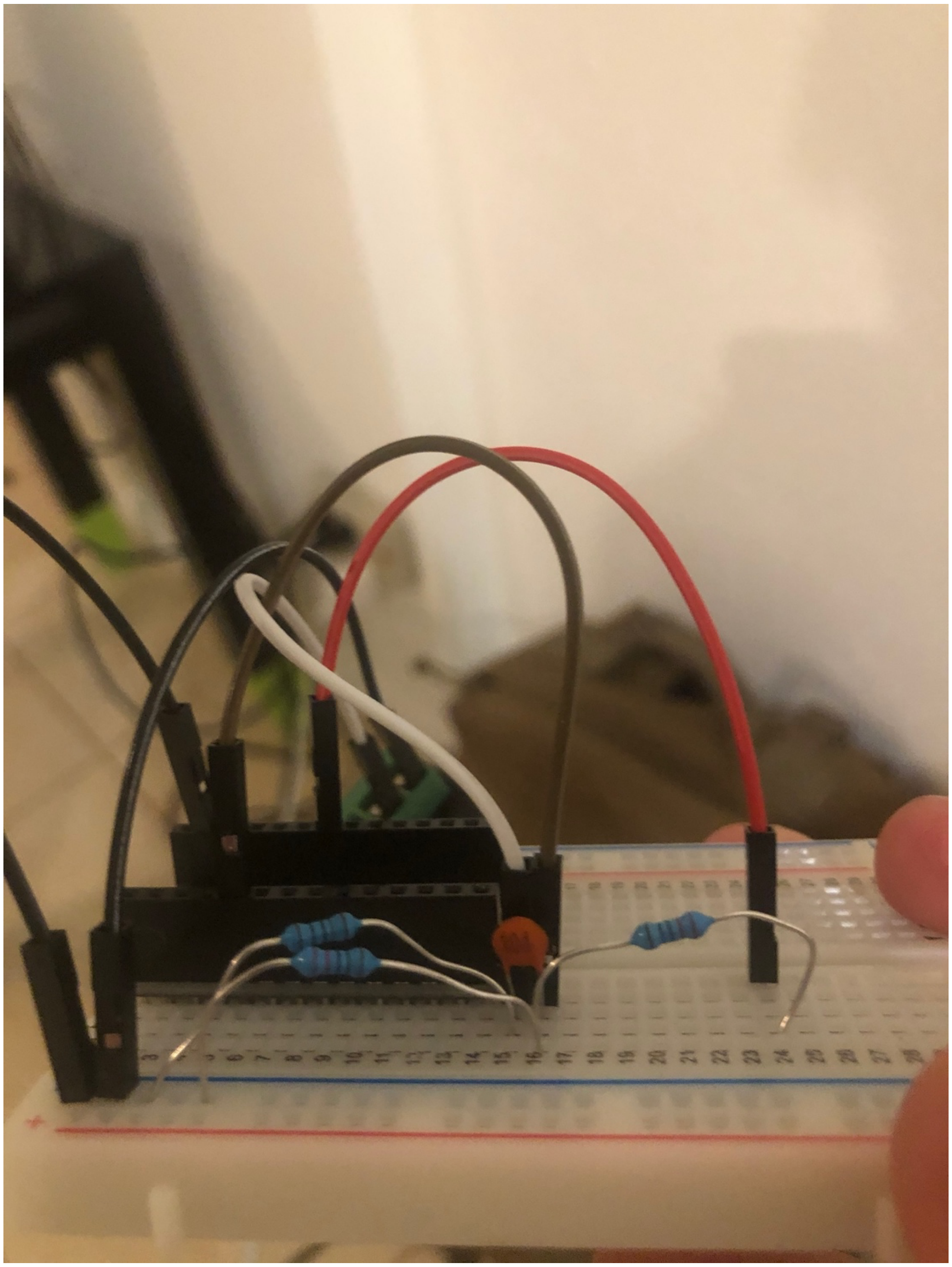
- In this section of the lab you will combine components in the previous section into one final circuit for audio sampling.
- You will effectively be able to *sample* any audio that is played from your device.
- You will implement the following circuit:



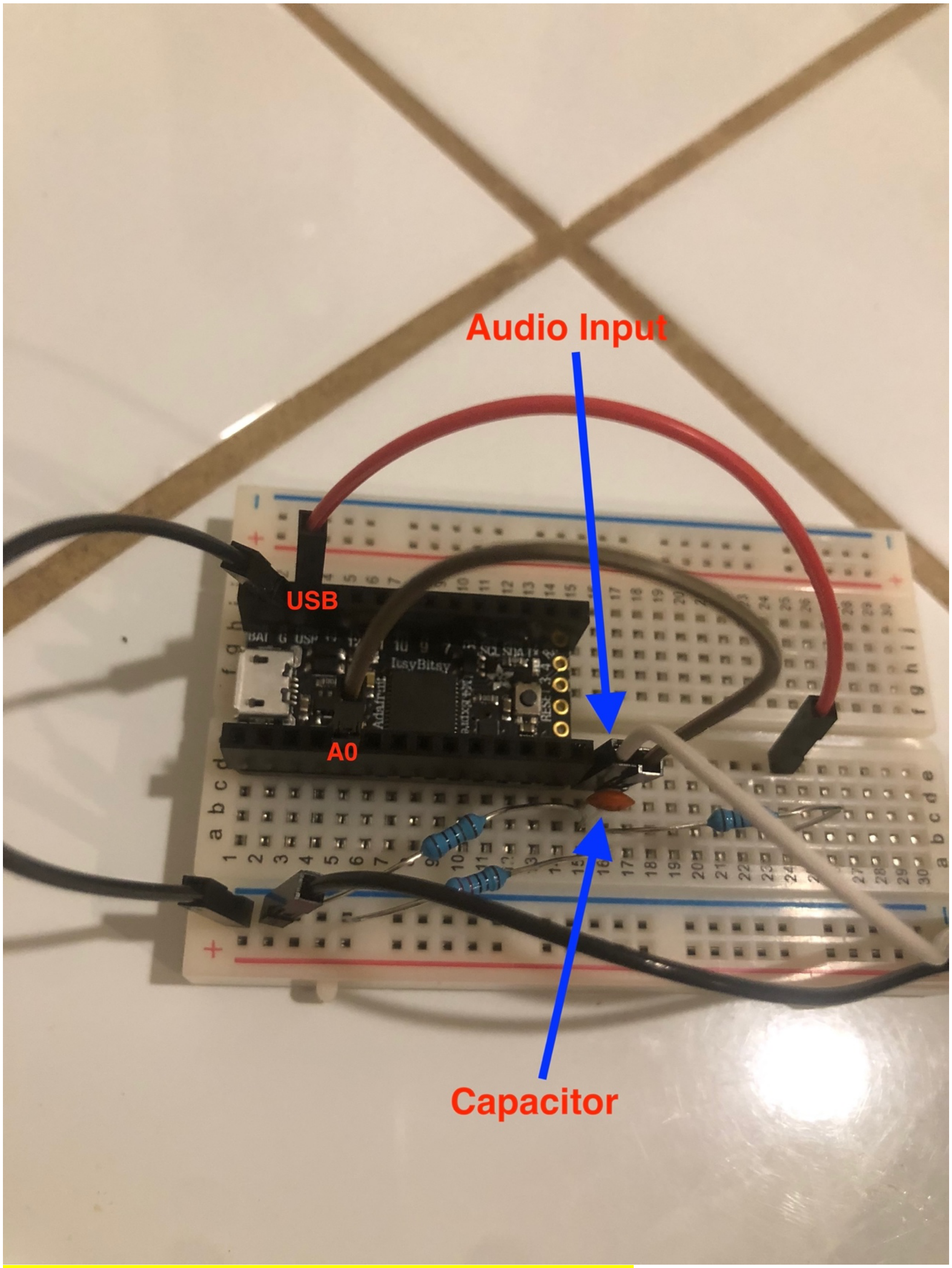
You'll want to have your Itsy-Bitsy straddling both sides of the breadboard. The first step is to connect the Audio ground to the Itsy-Bits ground (pin G). One way to do this is at the negative ground rail on the breadboard:



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
#define MAX_MICROSECONDS 20000
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();
          int time_est=0;
          int i;
          for(i = 0; (i < MAX_SAMPLES) & (time_est<MAX_MICROSECONDS);
      i++) {
            incomingAudio[i] = analogRead(A0); // read input from A0
            delayMicroseconds(2);
      // change line above to change sampling rates; and below

            time_est+=(2+18); // first number (2) should
             // match delayMicrosecond() argument;
             // second is fixed time for analogRead()
          }
          microseconds2 = micros();
          time_elapsed = (microseconds2 - microseconds1);
          int samples = i;

          Serial.print(samples);
          Serial.print(" samples in microseconds: ");
          Serial.println(time_elapsed);
          for(int i = 0; i < 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().

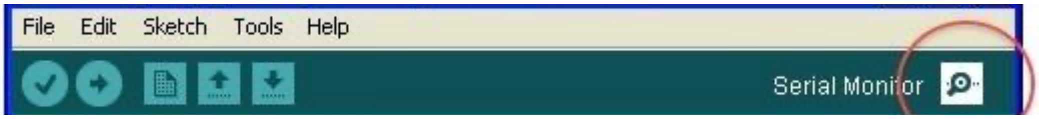
You'll notice that in order to sample the song data, we are conducting an analogRead() on pin A0 and storing the result in an array. Subsequently, we delay by a certain parameter to the delayMicroSeconds() function. ***It Is important to note that in reality, the analogRead() function takes about 18 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.*** (hence the 18 in the line time_est+=(2+18);) Note that the sketch provided measures the time taken for the sample collection loop and prints the total time taken for the set of 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 steps 4—7 for all three 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 OS and specific machine, but it should be able to detect the Itsy-Bitsy at your relevant port; it will be something like /dev/tty# on OS/X and Linux and COM# on Windows)
        ii. Select Tool -> Board -> ItsyBitsy
    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**.

If it complains about ADC0 not being declared or not having enough space, you probably haven't properly configured or selected the ItsyBitsy M4 board.

If it complains about g++ or other tools not being available, you probably haven't installed all the AdaFruit packages.



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 Excel sheet with the raw 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. A few common problems include:
        i. Not connected – The jack isn't plugged in, or the wire from the jack isn't connected to the correct Arduino pin, or some connection in between is missing.
        ii. Not getting audio from your audio source – if your phone or MP3 player isn't actually putting out any sound, you won't see movement. This can be problem if the "song" ends. If using a computer source, make sure the audio port is actually getting output (use the included headphone).
        iii. Not getting a strong enough signal from audio source – some students report that they need to turn the volume all the way up to see the signal.
        iv. Circuit misconfigured – If one of the resistors or the capacitor is not connected properly on either end, that could result in an output that did not move. Not connecting to pin A0 is also a possibility.
    b. 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 through its **quantization level**. Here you are taking the digital samples and reproducing a visual approximation of the original

continuous waveform that you sampled. Use Excel's plotting capability to produce a plot like the below plot ***for each column/tone*** of your data:

a. In Excel, use Insert->Chart

b. The x-axis should just be the row # (aka the sample #).

c. The y-axis should be the **quantization level**: 0->1024 (since you are using the Itsy-Bitsy's *10-bit* quantization data*).*

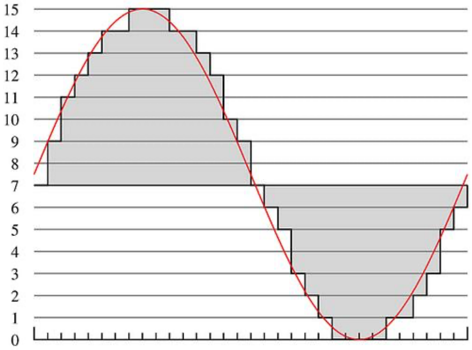
d. Use a line plot.

e. Zoom into your plots to show 3 cycles of waveform.

i. You can achieve this by limiting determining how many data points you choose.

f. Give each of the 3 plots a title and label the axes with the appropriate units.

g. You will need to turn these 3 graphs in.



2. Now we are going to make 3 new plots where we reconstruct our original signal from the samples you collected through its voltage.

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.

i. Hint: We saw numbers that ranged from 0 to 1023; how can we use this to quantize the voltage?

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 argument to the delayMicroseconds function and the value used in update `time_est`. 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 sampling at 500Hz 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 chart 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 7c1 and 7c2) 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 2 microseconds, p14+15), what is the sample rate?

2. For the 300Hz and 500Hz audio waves sampled at the original frequency of 50000Hz, were you able to match the data you collected to the claimed frequency in the file name? 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). Hint: think about how low sampling rate may increase the noise for the connect-the-dots reconstruction.
   a. Describe how these plots relate to the plot from the original (unmodified code) sample rate. Do they look different? How?
   b. What frequencies do the reconstructed sine waves have? [Hint: they may not all appear to have the same frequency as the original, high-sample rate versions.]

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)
  - Answer to question in Section 3 (7c3, highlighted)
  - Answers to postlab, including plots
  - Make sure all graphs and answers are clearly labelled.
- Separately upload your original sample rate data set samples (5 data sets – Section3, 7—10, 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 Monday 3pm.