ARDUINO Audio Input using Stethoscope

Study oriented project

Submitted in partial fulfilment of the requirements of
CS F266

by
(T Sai Ruthvik)
ID No- (2013A8TS489G)

Under the supervision of
_____Prof. Sreejith V_____

BITS, PILANI –K K BIRLA GOA CAMPUS

Date _21/03/2017___

TABLE OF CONTENT

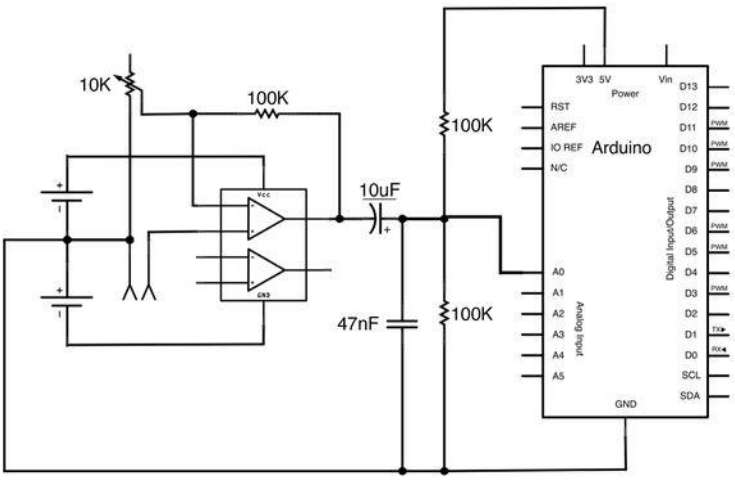
Arduino Audio input using Stethoscope

1. HARDWARE

The amplifier is the first step in the circuit, it increases the amplitude of the signal from around + or - 200mV to + or - 2.5V (ideally). The other function of the amplifier is to protect the audio source (the thing generating the audio signal in the first place) from the rest of the circuit.

The outgoing amplified signal will source all its current from the amplifier, so any load put on it later in the circuit will not be "felt" by the audio source (the microphone element in my case). Do this by setting up one of the op amps in the TL072 or TL082 package in a non-inverting amplifier configuration.

The datasheet of the TL072 or TL082 says that it should be powered with +15 and -15V, but since the signal will never be amplified above + or - 2.5V it's fine to run the op amp with something lower. I used two nine volt batteries wired in series to create a + or - 9V power supply.



Wire up your +V(pin 8) and -V(pin 4) to the op amp. Wire the signal from the mono jack to the non-inverting input (pin 3) and connect the ground pin of the jack to the 0V reference on your voltage supply (for me this was the junction between the two 9V batteries in series). Wire a 100kOhm resistor between the output (pin 1) and inverting input (pin 2) of the op amp. In this circuit I used a 10kOhm potentiometer wired as a variable resistor to adjust the gain (the amount that the amplifier amplifies) of my non-inverting amplifier. Later in this Instructable,

I'll show how you can add an LED indicator to Arduino pin 13 to let you know when you have this pot turned up too high (resulting in clipping of the incoming signal by the Arduino); this way you know when you should turn the pot down and get the signal back in the range you want (amplitude of ~2.5V). Wire this 10K linear taper pot between the inverting input and the 0V reference.

The following equation describes the relative amplitudes of the signal before and after the non-inverting amplifier:

Vout =~ Vin * (1 + R2/R1)
or
Vout/Vin =~ 1 + R2/R1
where R2 is the feedback resistor (between the output and non inverting input), R1 is the resistor to ground, Vout is the amplitude of the outgoing signal (the output from the amplifier), and Vin is the amplitude of the incoming signal (the input to the amplifier)

In this circuit R2 is a 100kOhm resistor and R1 is a 10kOhm potentiometer (variable resistor). By turning the pot you can change the resistance of R1 from 0Ohms to 10KOhms. Here are some example calculations:

When the pot is turned all the way to the left the resistance of R1 is 10kOhms and the ratio of Vout to Vin is about:
1+ 100/10 = 11
A signal coming out of the microphone with an amplitude of 200mV (which is fairly loud on my microphone) will be amplified to:
200mv * 11 = 2200mV = 2.2V
this is right in the range we want (amplitude close to 2.5V without going over)

Turning the pot to its halfway position will give it a resistance of 5kOhms, we can calculate the ratio of Vout to Vin again:
1+ 100/5 = 21
now the amplitude gets multiplied by 21
this is too much amplification for the 200mV signal:
200mV * 21 = 4200mv = 4.2V >> 2.5V
but this amplification would be perfect for a 100mV signal:
100mV *21 = 2100mV = 2.1V =~ 2.5V

Turning the pot farther to the right will keep decreasing the resistance of R1 and increase the amplification (also called gain) of this amplifier theoretically to infinity. Obviously at some point the amplifier will not be able to power a signal with a huge amplitude, but you get the idea. By adjusting the potentiometer you can adjust the gain of the amplifier and tune the sensitivity of the microphone while still keeping it in a range that the Arduino likes.

The next portion of the circuit DC offsets the output from the amplifier. As I explained in step 1, this +2.5V DC offset causes audio signal to oscillate around 2.5V so that it stays within the acceptable range for the Arduino's analog inputs (0-5V). Compare the non dc offset signal is fig 2 with the dc offset in fig 3. Specifically, notice how the signal in fig 3 always stays within the 0-5V range.

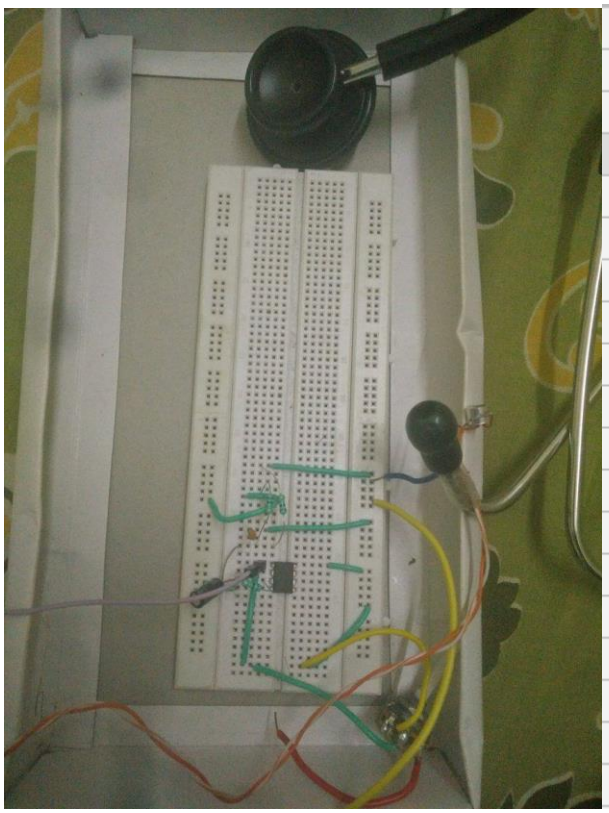
The DC offset circuit has two main components: a voltage divider and a capacitor. The

voltage divider is made from two 100k resistors wired in series from the Arduino's 5V supply to ground. Since the resistors have the same resistance, the voltage at the junction between them equals 2.5V. This 2.5V junction is tied to the output of the amplifier via a 10uF capacitor. As the voltage on the amplifier side of the capacitor rises and falls, it causes charge to momentarily accumulate and repel from the side of the capacitor attached to the 2.5V junction. This causes the voltage at the 2.5V junction to oscillate up and down, centered around 2.5V.

As shown in figs 3-8 and the schematic, connect the negative lead of a 10uF capacitor to the output from the amplifier. Connect the other side of the cap to the junction between two 100k resistors wired in series between 5V and ground. Also add a 47nF capacitor from 2.5V to ground.

Load the following code onto the Arduino. This code reads the voltage of the incoming audio signal using analogRead(A0) as a number between 0 and 1023 and stores it as the variable "incomingAudio."

The sensor data is then wireless transmitted and stored in google spreadsheets in the cloud, which can be accessed by any number of people live.



	timestamp	data
2	5074	319
3	12925	186
4	20313	251
5	27343	267
6	57916	288
7	68964	232
8	77375	304
9	84535	221
10	87964	279
11	96615	0
12	103698	1023
13	112231	24
14	119314	0
15	126640	155
16	133608	1023
17	140754	0
18	148058	0
19	155009	1023

2. SOFTWARE

Now, for the Arduino code to work properly, there are a set of instructions to follow:

1. Create a Temboo account: http://www.temboo.com

2. Retrieve your Temboo application details: http://www.temboo.com/account/applications

3. Replace the values in the TembooAccount.h tab with your Temboo application details

4. You'll also need a Google Spreadsheet that includes a title in the first row of each column that data will be written to. This example assumes there are two columns. The first column is the time (in milliseconds) that the row was appended, and the second column is a sensor value. In other words, your spreadsheet should look like:

Time	Sensor Value

5. Google Spreadsheets requires you to authenticate via OAuth. Follow the steps in the link below to find your ClientID, ClientSecret, and RefreshToken, and then use those values to overwrite the placeholders in the code below.

   https://temboo.com/library/Library/Google/OAuth/

   For the scope field, you need to use: https://www.googleapis.com/auth/spreadsheets

   Here's a video outlines how Temboo helps with the OAuth process:

   https://www.temboo.com/videos#oauthchoreos

   And here's a more in-depth version of this example on our website:

   https://temboo.com/arduino/yun/update-google-spreadsheet

6. Next, upload the sketch to your Arduino Yún and open the serial monitor