CG1111 Engineering Principles and Practice I

Sensors and Applications: Acoustic-based Sensors

(Week 9, Studio 2)

Time	Duration (mins)	Activity
0:00	20	Briefing
0:20	60	Activity #1: Ultrasonic Sensor
1:20	70	Activity #2: Designing a Noise Detector using a Microphone
2:30	10	Final discussions and wrap-up

Introduction

In the previous studio, we have examined the working principles of some photoelectric sensors. In this studio, we will be investigating two types of acoustics-based sensors, namely, ultrasonic sensor and microphone, and their use in some practical real-world applications.

Working of the Ultrasound Sensor:

- An ultrasonic sensor works by transmitting a short burst of 40 kHz ultrasonic pulses towards a target and measuring how long it takes for the pulses to be echoed back to the receiver. The measured duration can then be used to calculate the distance.

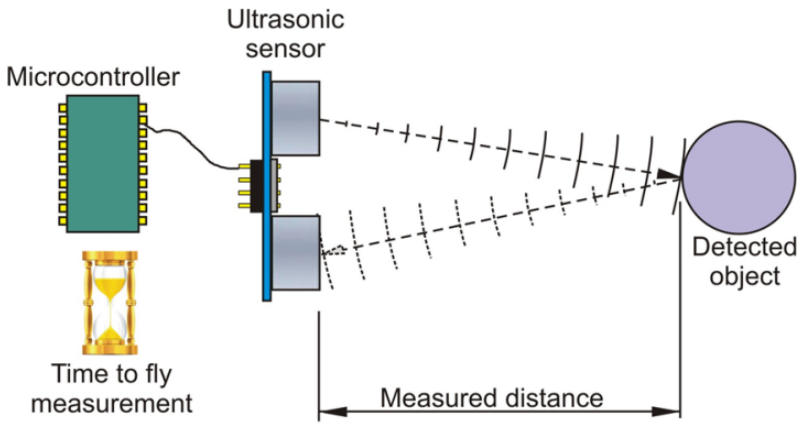


Figure 1. Ultrasonic sensor's working principle.

- In Activity 1, we will be configuring an ultrasonic sensor to work with the Arduino Uno
- The given ultrasonic sensor has 4 pins, namely, 5V, GND, **Trigger and Echo**.
- The ultrasonic sensor sends 8 pulses of 40KHz via the '**Trigger'** terminal.
- The ultrasonic pulses collide an object and leaves in the other direction.
- The '**Echo'** terminal receives the ultrasound pulses and the sensor calculates the distance based on the time of the pulses.
- To perform the time measurement, we can make use of Arduino's **pulseIn()** function:

- Purpose: Measures the duration of a pulse (either HIGH or LOW) on a pin. For example, if its `value` parameter is set to HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing.
- Syntax: `pulseIn(pin, value, timeout)`
- Parameters:
  - `pin`: the number of the pin on which you want to read the pulse. (`int`)
  - `value`: type of pulse to read; either HIGH or LOW. (`int`)
  - `timeout` (optional): the number of microseconds to wait for the pulse to start (`unsigned long`)
- Returns: the length of the pulse (in microseconds), or 0 if `timeout` occurs

Working of the Microphone:

- A microphone is like our human ear. It has a diaphragm like the eardrum that converts sound into an electrical signal.
- Remember that the sound we hear is energy carried by vibrations in the air.
- When you speak, **sound waves** created by your voice carry energy towards the microphone. The **coil**, attached to the diaphragm, moves back and forth as well. The permanent magnets in the microphone produces a magnetic field that cuts through the coil. As the coil moves back and forth through the magnetic field, electric current flows through the coil.

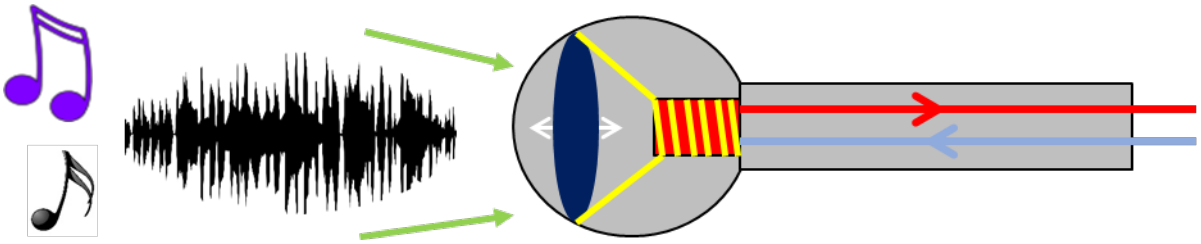


Figure 2. The working principle of a microphone.

- In Activity 2, we will be designing a noise detector using a microphone. In the process, we will also understand the designing of filtering circuits for sensor interfaces, as well as the working of an envelope detector.
- A microphone is essentially a variable resistor that changes its resistance based on the loudness of the sound.
- A microphone responds to a large range of frequencies, typically ranging from 16 Hz - 16000/18000 Hz. **It is important to identify the working range needed for our system or application**.
- Here, since we are dealing with noise detection, we will limit our frequency range from 100 Hz to 4000 Hz. To achieve this, we need to design a band-pass filter, which is essentially a high-pass filter combined with a low pass filter.
- An envelope detector is an electronic circuit that takes a high-frequency signal as input and provides an output that is the "envelope" of the original signal (see Figure 3).
- The capacitor in the circuit in Figure 3 stores up charge on the rising edge and releases it slowly through a load resistor when the signal falls.

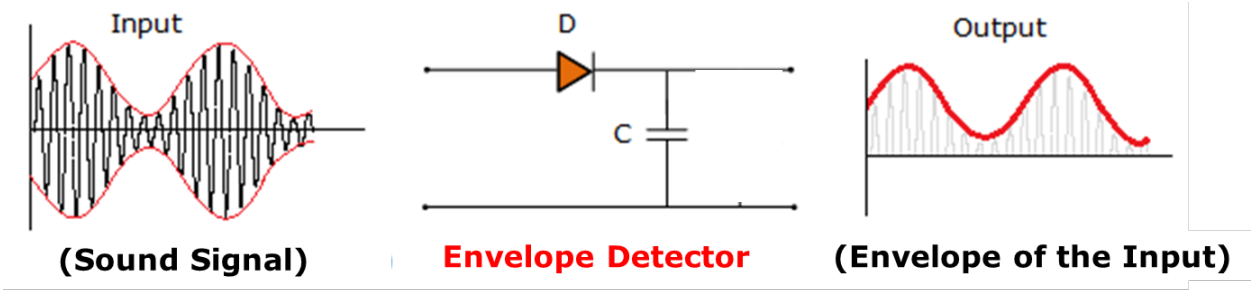


Figure 3. Envelope Detector

Objectives:

- To understand the working of an ultrasonic sensor through observing its working principle
- To understand the working of a microphone
- To be able to design filtering circuits for sensor interfaces
- To understand the working of an envelope detector

Materials:

- Ultrasonic Sensor (HC-SR04)
- Arduino Uno
- Bitscope
- Microphone (AMB-713-RC)
- Microphone Jack
- LM358 x2
- IN4001 diode x1
- LEDs:
  - 1 x Red
  - 1 x Yellow
  - 1 x Green
- Resistors
  - 560 Ω x3
  - 1 kΩ x4
  - 2.2 kΩ x1
  - 10 kΩ x1
  - 15 kΩ x1
  - 100 kΩ x1
  - 390 kΩ x1
  - 2 kΩ variable resistor (code 202)
- Capacitors
  - 100 nF x 1 (Code 104)
  - 100 pF x 1 (Code 101)
  - 2.2 µF x 1 (Electrolytic Capacitor)
- Ruler → 0-100 cm (This is to verify the distance measured by the ultrasonic sensor)

Setups:

A. Bitscope DSO Setup:

a) Ensure that the "GND" pins of the Bitscope are connected to the common ground terminal of the circuit.

b) In the "Channel Controls (7)" panel, change the voltage per division of "CHA" or "CHB" according to the setup requirements.

c) In the "Timebase Control (6)" panel, change the Timebase according to the setup requirements.

d) Change the vertical position for both "CHA" and "CHB" to "Mean".

e) Connect "CHA" and the "AWG" pin to the V_{in} terminal in your circuit.

f) Connect "CHB" to the V_{out} terminal in your circuit. Ensure that you activate "CHB"

B. Dual Power Supply Setup:

a) The USB breakout cable provides the primary DC voltage source while the Arduino Uno provides the second DC voltage source

b) Connect your Arduino Uno (using the USB cable that came with it) to another USB power adapter that is isolated from the one powering your USB breakout cable. This can be another USB power bank or adapter. **Do not connect the Arduino to the same laptop/PC as the Bitscope and/or the USB breakout Cable**

Note: It is very important that the second power source is isolated from the first. If they are from the same device (e.g., two USB ports on the same power bank or laptop), their grounds (i.e., negative terminals) are connected internally and you will get wrong results!

c) Notice that the Arduino Uno has a pin that is labelled as "5V". This serves as the "+" terminal of your second 5 V source. There are also two ground pins beside it that are labelled as "GND". Either one of this can be used as the "–" terminal of your second Arduino 5V DC source.

d) Connect the '-' of the USB breakout cable to the '+' from the Arduino 5V DC source. This becomes the new GND terminal of the Dual Power Supply setup and your circuit's common ground.

e) Measure the voltage at the '+' of the USB breakout cable with respect to the GND terminal using the digital multimeter and ensure that it is around **+5V**.

f) Measure the voltage at the '-' of the Arduino 5V DC source with respect to the GND terminal using the digital multimeter and ensure that it is around **-5V**.

Note: If you are using a power bank to power up either the Arduino or the USB breakout cable, it may switch off after some time because of low current drawn from the circuit. Be sure to periodically check and switch on the power bank if needed.

C. Measurements using Bitscope:

a) Click on "TRACE" in the "Timebase Control (6)" panel to freeze the waveforms.

b) Turn on "CURSOR" in the "Scope Selectors (2)" panel.

c) To measure the peak-to-peak voltage, click on "CHA"(or "CHB") in the "Channel Controls (7)" panel, then right-click on the brown voltage box in the "Cursor Measurements (5)" panel to select "MAX".

d) Right-click on the dirty-green box directly underneath the brown box and select "MIN".

e) The peak-to-peak voltage of "CHA" (or "CHB") is now shown in the yellow box directly underneath the dirty-green box.

Note: Always verify the V_{in} peak-to-peak using the cursor measurement before going ahead with the activity

---

Activity #1: Ultrasonic Sensor (60 mins)

We will be using the Arduino Uno to configure and characterize our ultrasonic sensor.

Procedure:

1. Connect the ultrasonic sensor to the Arduino Uno as follows:
   - Vcc (Sensor)-------> 5V (Arduino)
   - Trigger ----------> Arduino Pin 12
   - Echo ----------> Arduino Pin 11
   - GND (Sensor)----> GND (Arduino)
2. Connect the Arduino Uno to your laptop/PC (using the USB cable) and open the Arduino IDE. Select the correct COM port in the IDE and choose the board as "Arduino Uno".
3. Copy and paste the code found in "Ultrasound.txt" into a new Arduino sketch. Pay attention to how the ultrasonic pulses are triggered, and how the flight duration of the echo is measured. Save the code, compline the code, upload it to the Arduino Uno, and open the '**Serial Monitor'**.
4. Fix an obstacle (target) at the **10-cm** mark of a ruler. Ensure that the obstacle is at least A4 size.
5. Align your ultrasonic sensor at the **0-cm** mark of a ruler, with the two cylinders on the sensor facing the target obstacle.
6. The duration given in the Serial Monitor is for the total distance the ultrasonic wave travelled. This total distance is twice the distance between the target and the sensor.
7. Fill in Table 1 for different distances between the ultrasonic sensor and the target.
8. You can also see the change in duration as you change the obstacle position on the **'Serial Plotter'**.
9. With the help of Microsoft Excel, determine the relationship between the duration measured for 'd', and the distance 'd' between the sensor and the target.
10. What is the significance of the gradient when you plot Duration vs. Distance? Try to compare it against the theoretical value (e.g., calculated using an online tool).
11. Modify the code to display the estimated distance 'd' (in cm) of the target, instead of the flight duration of the ultrasonic pulses. (Hint: Use the gradient of the plot of Duration vs Distance)

```
#define TIMEOUT 30000
#define WAITING_TIME 1000

/* The ultrasonic sensor (HC- SR04) has 4 pins,Vcc,  Trigger, Echo and GND
 *  Vcc(Sensor)---> 5V (Arduino)
 *  Trigger ------> Arduino Pin 12
 *  Echo ---------> Arduino Pin 11
 *  GND (Sensor)--> GND (Arduino)
*/
#define TRIGGER 12
#define ECHO 11

long duration;

//The following code is run once at the start of execution to setup the different peripherals

void setup()
{
  pinMode(TRIGGER, OUTPUT);
  digitalWrite(TRIGGER, LOW);
  pinMode(ECHO, INPUT);

  // Set up the serial Communication
  Serial.begin(9600);
}
void loop()
{
  //The following code is run repeatedly
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(10);

  duration = pulseIn(ECHO, HIGH, TIMEOUT);
  Serial.print("Duration: ");
  Serial.print(duration);
  Serial.println(" microseconds");

  delay(WAITING_TIME);
}
```

Figure 4. Arduino Code for the Ultrasonic Sensor (From Ultrasound.txt)

Table 1:

Distance ‘d’ (cm)	Duration for ‘2d’ (microseconds)	Duration for ‘d’ (microseconds)
0		
10		
20		
30		
40		
50		
60		
70		
80		
90		
100		

Activity #2: Designing a Noise Detector using a Microphone (70 mins)

- In this activity, we will be building our own Noise Detector using a Microphone.
- The LM358's pin-out is shown in Figure 5. In this activity, we will be using four op-amps. Recall that the LM358 has two op-amps per chip. Hence, we can use just two LM358 chips to design our overall circuit.
- The noise detector is designed progressively by dividing it into the sub-systems as shown in Figure 6.

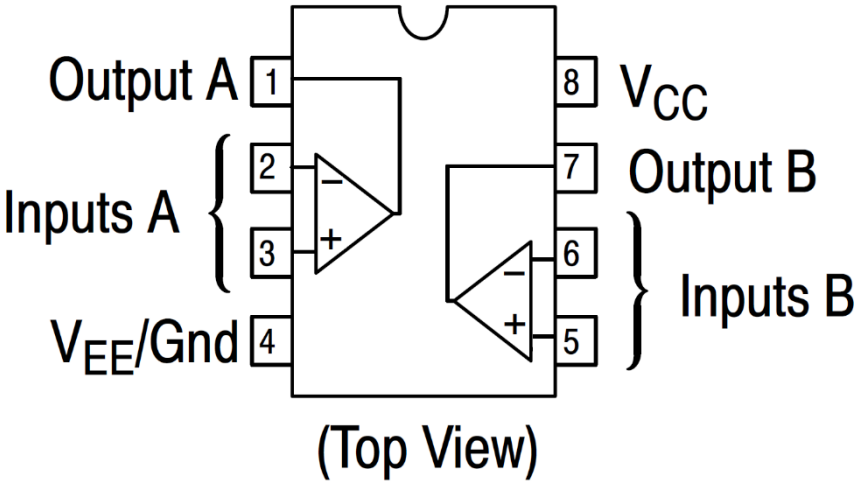


Figure 5. LM358 Pin-Out

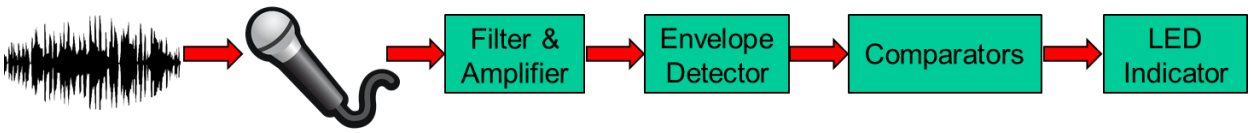


Figure 6. Noise Detector Block Diagram

1. Wire the circuit on your breadboard as per the schematic given in Figure 7.
2. Ensure that you connect the microphone with the correct polarity. Refer to the datasheet of the microphone for the same.
3. Use the dual power setup described above **(Setups: B. Dual Power Supply Setup)** to power up the op-amp. Pin 4 of the Op-Amp should be at -5V and Pin 8 should be at +5V. Pay attention to the polarity of the voltage supply. The "GND" terminal of the Dual Power Supply Setup is the ground terminal of the circuit
4. Calculate the low-pass cutoff frequency, high pass cutoff frequency, and the overall gain for the filter-amplifier combination.

   Note: you can verify the working of the high pass filter and the low pass filter by following the procedure for Week 9 Studio 1

5. Connect "CHA" to 'V+' and "CHB" to "Vout1" in Figure 7. Select a Timebase of 100 ms/Div and a zoom of 1:2. The combination gives a Timebase of 200ms/Div. Adjust the voltage scale according to each channel's voltage range.

Note: You may see the noise as some disturbances in both 'CHA' and 'CHB' that die down after a few seconds. To see a more continuous waveform for the noise a higher Timebase of 500ms/div or 1s/div is required

6. Ensure that there is some change in both 'CHA' and 'CHB' when the noise increases.

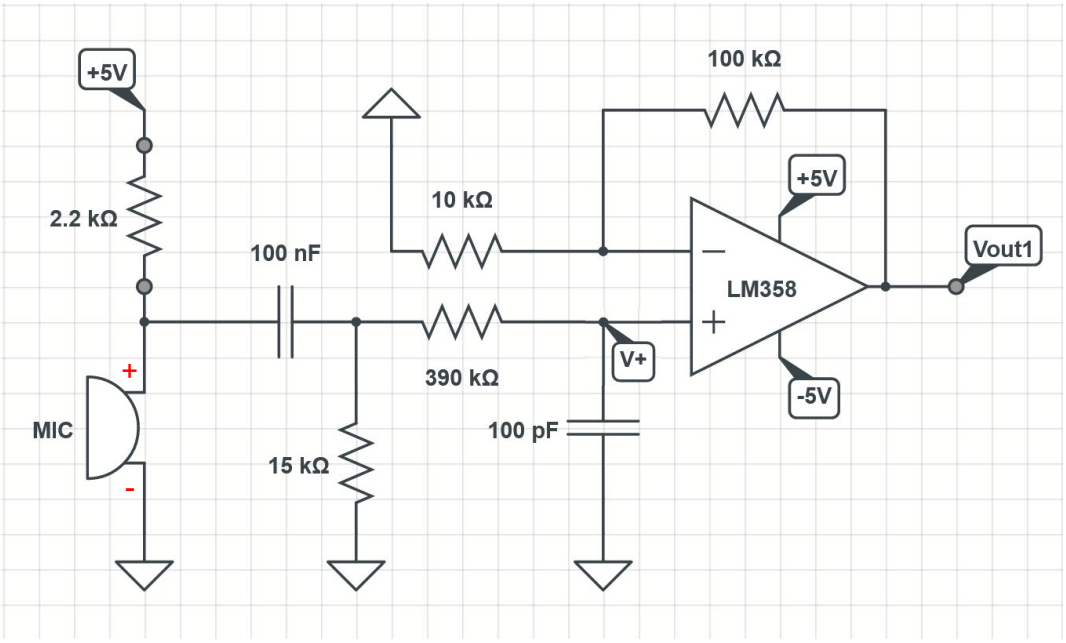


Figure 7. Microphone with the Low Pass Filter + High Pass Filter + Amplifier

7. Next, connect the envelope detector to the output of the filter-amplifier combination as shown in Figure 8. The output of the envelope detector should follow the noise profile and should reduce as the noise reduces.
8. Could you think of a way to modify the responsiveness of the whole circuit? (Hint: you can modify the envelope detector by adding a resistor in parallel to the capacitor. You can experiment with values from 500 Ω to 2 kΩ by using the 2 kΩ variable resistor).

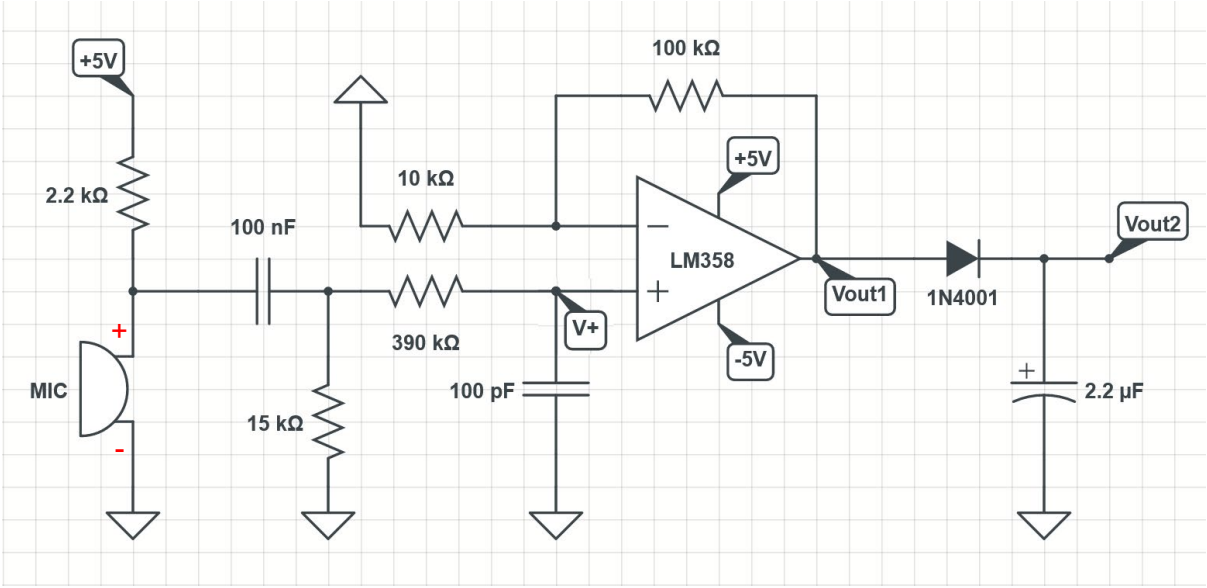


Figure 8. Microphone with the Low Pass Filter + High Pass Filter + Amplifier + Envelope Detector

9. Finally, connect the output from the envelope detector to the comparator circuits with the indicator LEDs as shown in Figure 9. Remember that each LM358 has two op-amps so we can just use two LM358 chips to build the entire circuit. The three comparators have reference voltages of approximately 1.25 V, 2.5V and 3.75 V.

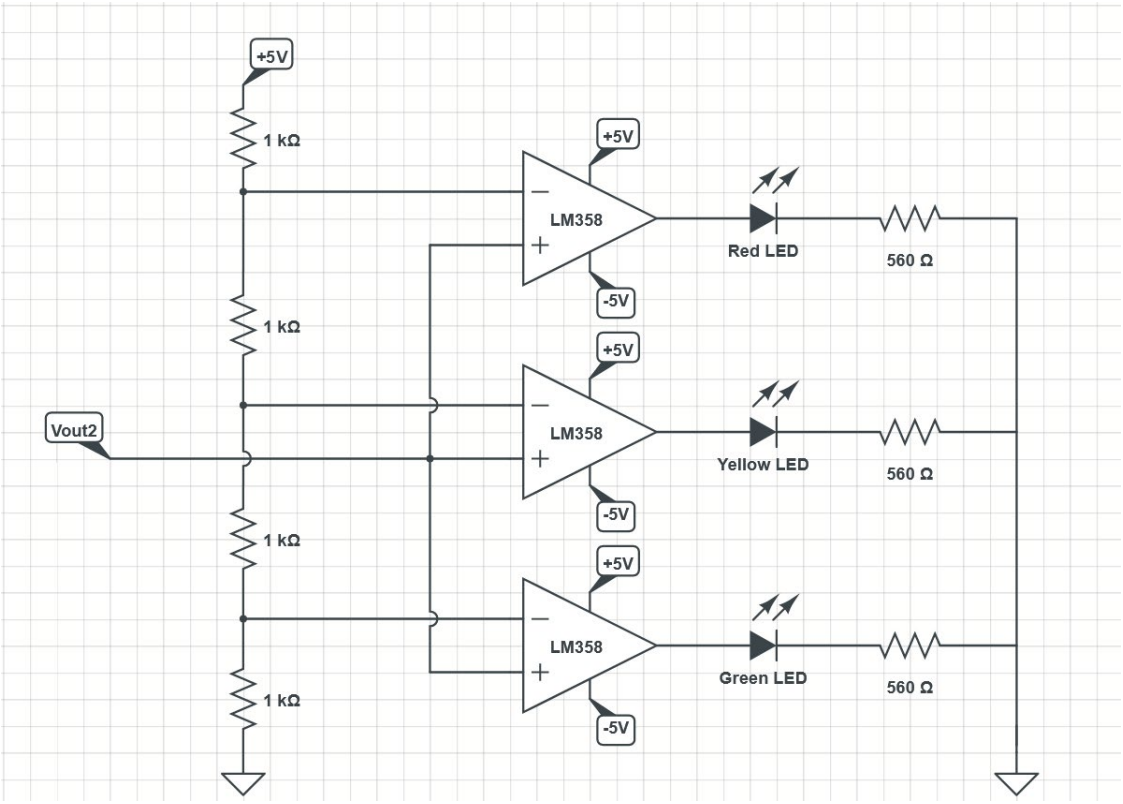


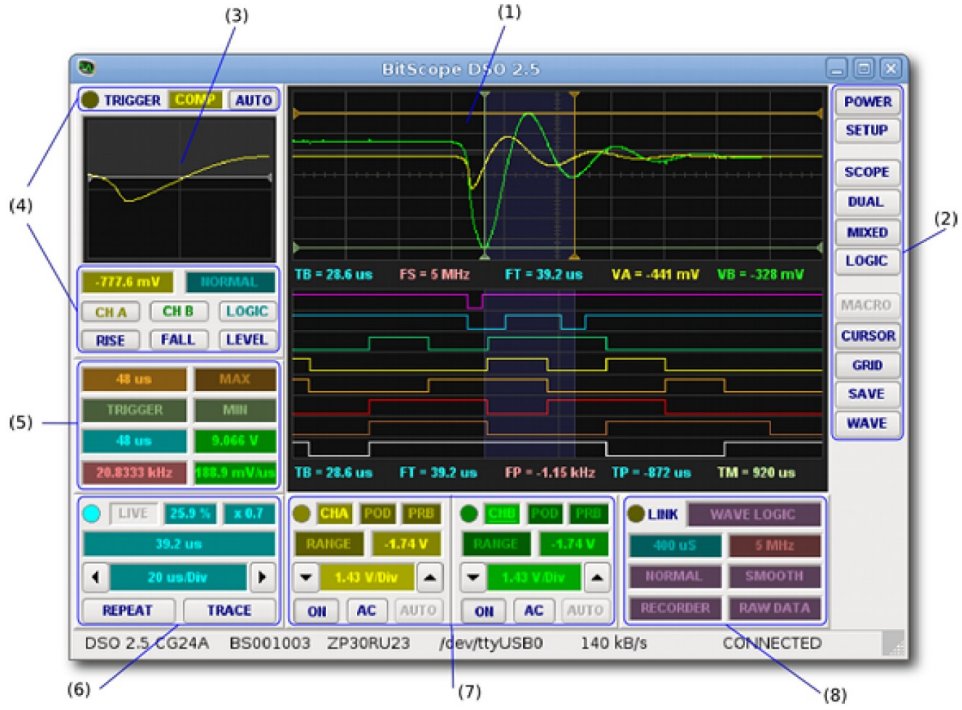
Figure 9. Connecting the Envelope Detector's Output to Comparators and LED Indicators

10. Each comparator outputs a '**high'(**or oscillates to +Vcc**)** when the voltage (which is indicative of the noisiness/loudness of the sound) at the respective op-amp's 'V_+' terminal crosses the corresponding reference voltages (i.e. 1.25 V, 2.5V and 3.75) and switches on the LED connected to it.
11. Thus, green LED implies that the detected sound is the least noisy, green and yellow LED implies moderately noisy, and when all three LEDs light up, it implies very noisy.
12. You can modify the voltage divider resistor values to design your own reference voltages according to the peak voltage of the envelope detector.

Please feel free to ask your TA or the Instructors any questions or doubts you may have.

END OF STUDIO SESSION

APPENDIX: BITSCOPE DSO'S INTERFACE



ID	FEATURE	DESCRIPTION
(1)	Main Display	Waveform, logic and spectrum displays, measurements and cursors.
(2)	Scope Selectors	Virtual instruments, scope tools, presets, cursors, graticule etc.
(3)	Trigger Windows	Shows trigger levels, analog and logic waveforms at the trigger.
(4)	Trigger Controls	Controls trigger setup and displays trigger waveform and data.
(5)	Cursor Measurements	X and Y cursor values, voltage, time and rate measurements.
(6)	Timebase Control	Timebase, Zoom and Time Focus control parameters.
(7)	Channel Controls	Controls input source, range, vertical position and scaling.
(8)	Capture Control	Capture sample rate, duration, frame rate and display modes.