# Stop Noise Pollution from Honking

Group DD15

Dimple Kochar 16D070010
Nisha Brahmankar 16D070019
Pratyush Ragini Singh 16D070046

Faculty Mentor- Prof. Pramod Murali

July 26, 2020

## 1 Project Objectives

- To count the number of times a driver uses the horn of the car
- To transmit this data to a server
- To detect if the device is being tampered

## 2 Block Diagram

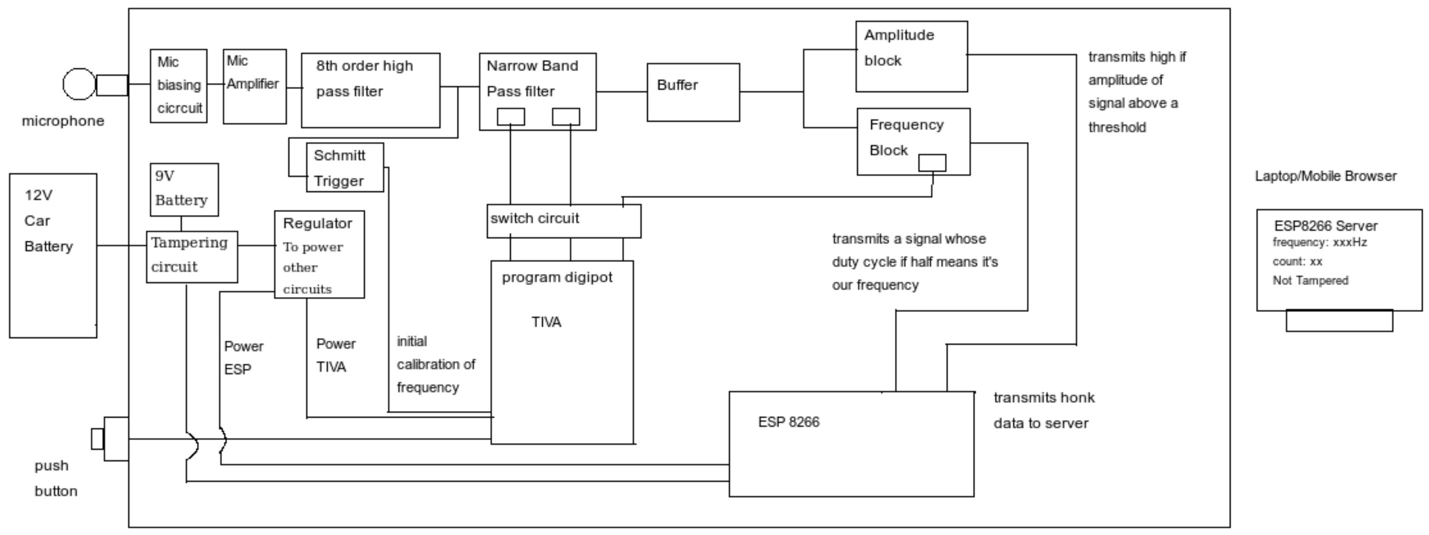


Figure 1: Overall Block Diagram of Project

## 3 Design Approach

- We are implementing this device majorly using hardware. The initial processing of input signal from mic on the device is accomplished by hardware.
- We are detecting the horn based on two things - Amplitude (inorder to avoid aliasing with the horn of same frequency from some other car) and Frequency. These blocks are implemented using hardware.
- For calibration purposes we need to set the value of digital potentiometer and microcontroller(TIVA-C) is used for the purpose.
- For transmission of the data to the server, ESP8266 module is programmed and used to send data.

# 4 Subsystems

## 4.1 Calibration

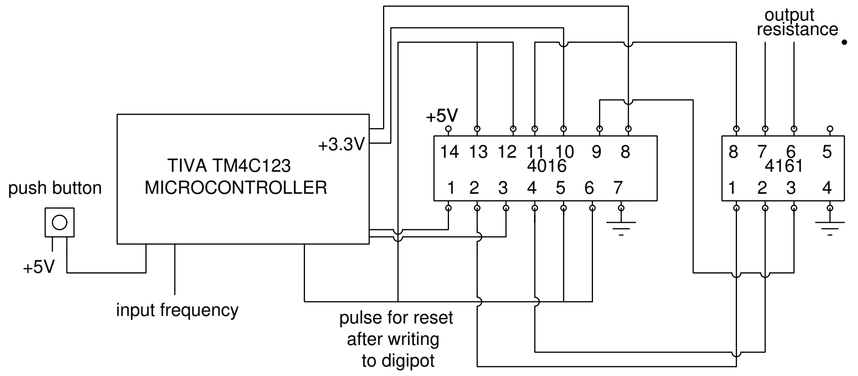


Figure 2: Calibration Block Diagram

Before the device is put to use, we have to calibrate it for the frequency of our horn. For this, we press the push button for 2 seconds, during which the input signal is read by the microcontroller, and frequency is obtained.
Then, the digital potentiometer writes this accordingly to the digital potentiometers. For the written value to reflect, we need to disconnect and reconnect all connections to the digital potentiometer, for which we use a switch.

## 4.2 Mic Amplifier

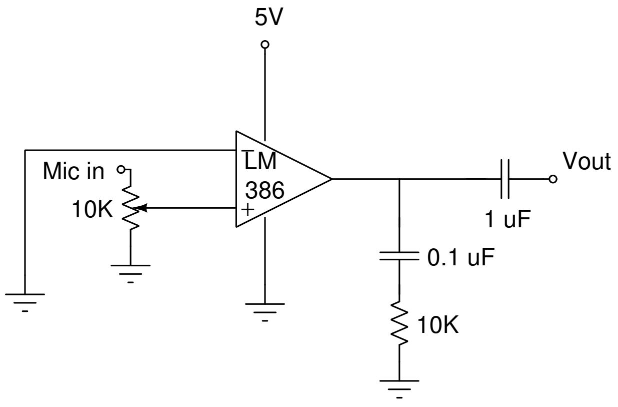


Figure 3: Mic Amplifier Circuit

To get desired signal from mic, an audio jack that separates headphone & microphone signal was used along with biasing circuit to supply power to the mic. Implemented the mic amplifier circuit for amplication of input signal from the mic & tuned it to desired amplitude.

TEST RESULTS: It is an audio amplifier which amplifies 100 mV signal to about 1.5V. This amplification can be controlled by the pot in the initial stage.

## 4.3 High Pass Filter

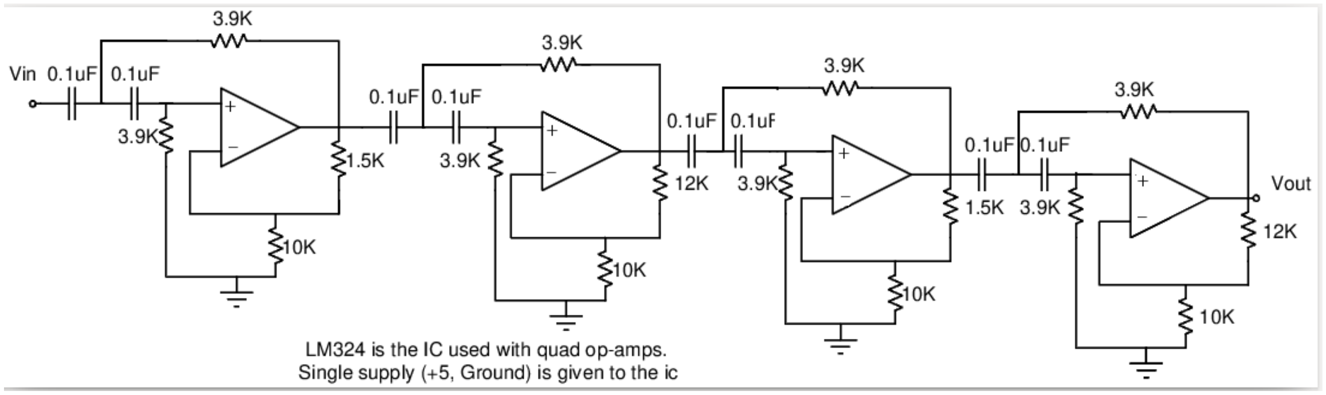


Figure 4: High Pass Filter Circuit

Constructed 8th order Butterworth highpass filter with cutoff frequency of 400Hz.
This is made in order to filter out all the low frequency noise of engine etc.

TEST RESULTS: On giving input of various frequencies, we could see the change in amplitude - decrease for frequencies lower than and near about 400Hz.

## 4.4 Small Filter

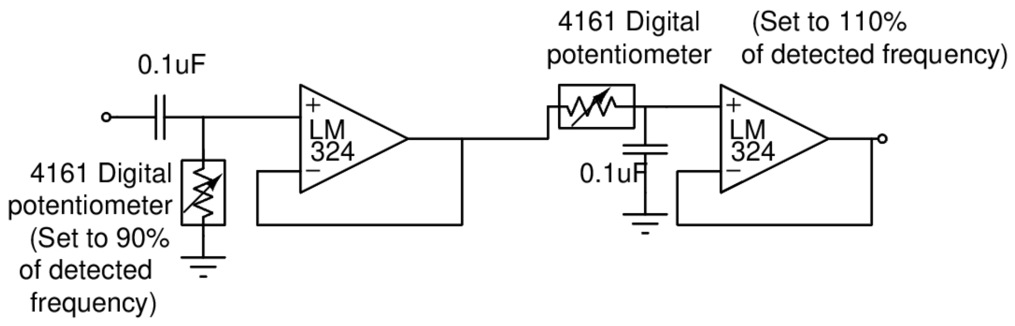

0.1uF
4161 Digital potentiometer (Set to 90% of detected frequency)
LM 324
4161 Digital potentiometer
(Set to 110% of detected frequency)
0.1uF
LM 324

Figure 5: Small Filter Circuit

We then have a narrow band filter for restricting our signal to the frequency of interest, whose potentiometer values were set during calibration.
(Refer Block Digram)

## 4.5 Amplitude Block

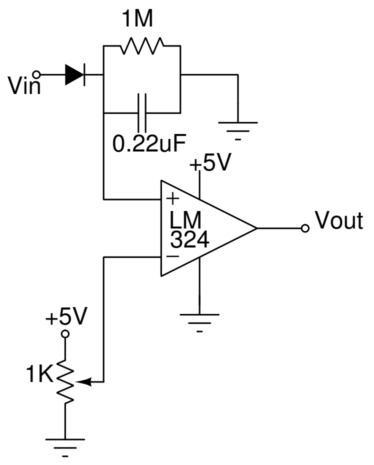


Figure 6: Amplitude Block Circuit

The signal that we receive from the small filter is converted to a DC voltage. If this voltage is above a certain threshold, the signal is detected as our honk.
The threshold has been selected after considering how horns sound in different cars and what the final input amplitude is approximately like to the block.

TEST RESULTS: Obtained accurate results with even 100 mV peak-peak difference in input giving us a 0 V for the lower and 5 V for the higher input.

## 4.6 Frequency Block

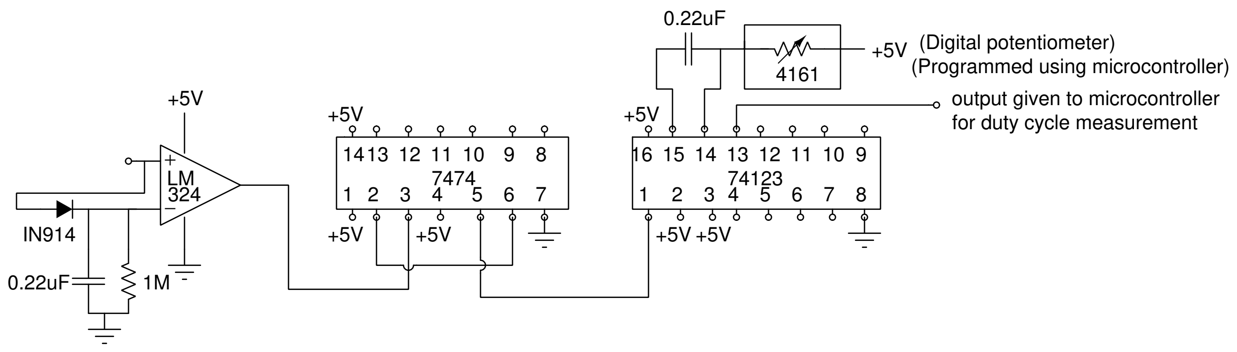

0.22uF
4161
+5V
(Digital potentiometer)
(Programmed using microcontroller)
output given to microcontroller for duty cycle measurement
+5V
LM 324
IN914
0.22uF
1M
7474
74123
+5V +5V

Figure 7: Frequency Block Circuit

- The narrow band signal we receive is converted to a pulse wave of unknown duty cycle of the same frequency.
- Then, using the 7474 IC (D Flip Flop) we convert it to a 50% duty cycle wave with the half the frequency, because 74123 IC requires a nearly 50% duty cycle signal to function.

- The 74123 (monostable multivibrator) outputs a signal as follows:
  1) If input frequency is higher, it gives a >50% duty cycle pulse.
  2) If input frequency is lesser, it gives a < 50% duty cycle pulse. This is measured by a microcontroller which gives a high value if the duty cycle is between 49%-51% (thresholds decided).

TEST RESULTS:
Below are the DSO screenshots for processing input signal of 3kHz:

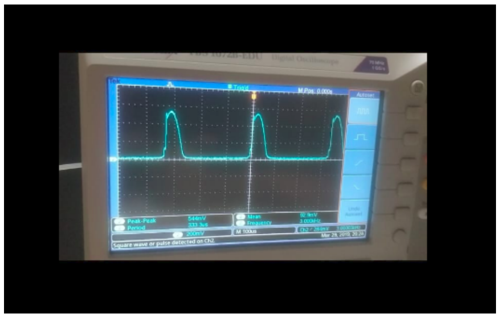


Figure 8: Highpass filter output

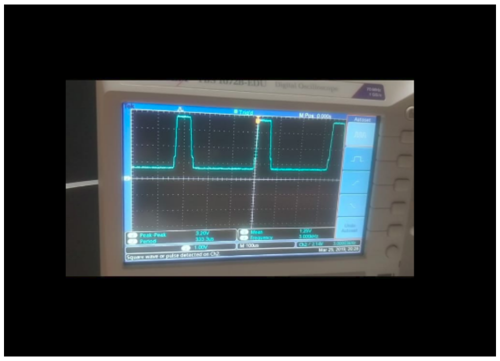


Figure 9: Output at Comparator

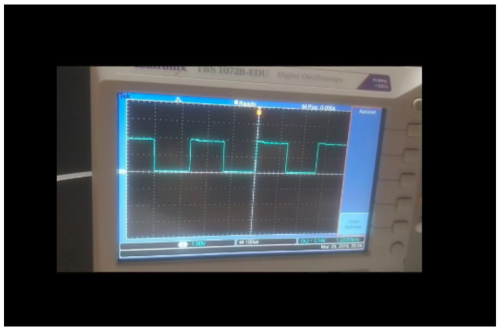


Figure 10: Duty Cycle Correction

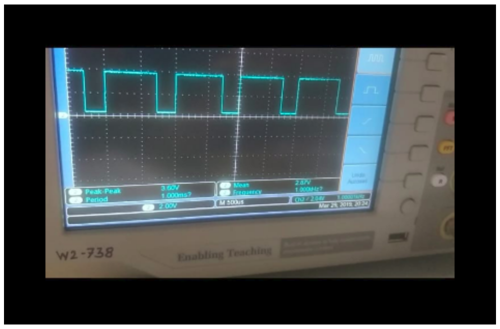


Figure 11: Frequency Comparator output

## 4.7 Tampering

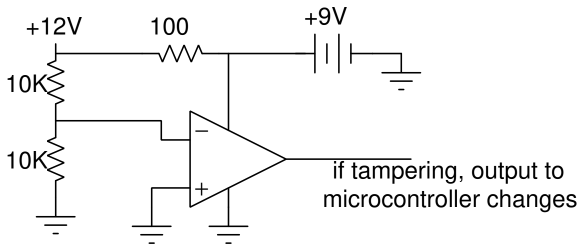


Figure 12: Tampering Circuit

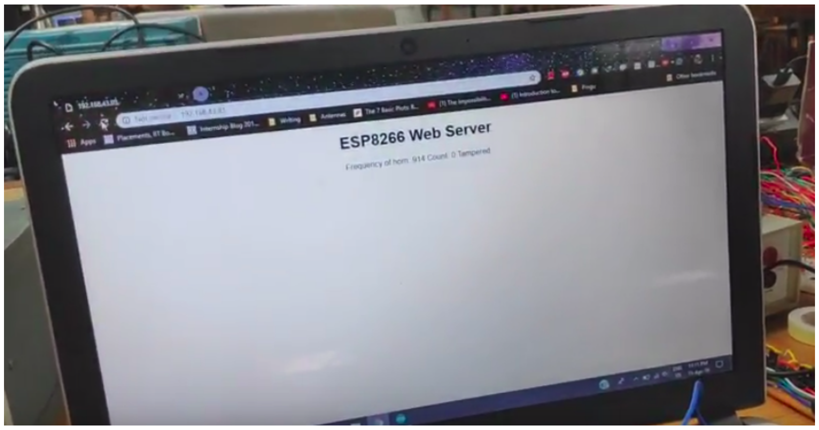
The only way the device can be tampered is when the user cuts the 12V supply from car battery to the device.

We have a separate 9V battery for supply to ESP8266 Module. This battery gets recharged from the car battery.
If the 12V supply wire is cut, ESP sends a signal that our device is tampered with.

## 4.8 ESP8266 Wi-Fi Module

The ESP8266 Module is used to send our data to a web server.
Amplitude block, frequency block and tampering result are given as input signals to the ESP pins. We need a voltage regulator circuit to make sure that these signals are less than or equal to 3.3 V. Based on these signals, data of count and tampering are transmitted.
The ESP server shows our calibrated frequency, total count and tampering result. Count is defined per second which means if horn on for 3 seconds then it will consider it as 3 counts.

TEST RESULTS: Below is the screenshot of ESP Web server.



ESP8266 Web Server

Frequency of horn: 914 Count: 0 Tampered

Figure 13: ESP Web Server

## 5 More Accurate Solution for Frequency Block

The hardware method of frequency block is very complicated and does not give very accurate results. The digital potentiometer values set is not very accurate due to unreliability of SPI communication through TIVA.
To get exact results, we can work out the frequency block using software. Initially the frequency is being measured in both the cases, the difference here is that we compare this frequency with real time frequency using software instead of the very complicated hardware methods. This gives us very accurate results and changes the count after changing the frequency even slightly, say about 10-12 Hz.
The software part is implemented using ESP8266 board and Arduino code.
This method is more reliable for practical purposes. Even after using this method, a lot of hardware is involved in the overall device.

## 6 Problems Faced

- Using the digital potentiometer
  - Desired resistance was effectively written in the digipot, when independent but when it was connected as an equivalent resistance to the frequency block, it stopped changing its resistance value.
- Variation in the behaviour of circuit on breadboard and PCB
  - Same circuit on breadboard works perfectly but there occurs some issue on PCB. Debugging this problem was a challenging task. The reasons for this different behaviour maybe due to printing errors, soldering errors or component damage.
- Unreliability of mic since the one we used was getting damaged easily
  - The condenser mics we initially used from the lab were very unreliable because they get damaged easily. We then used a proper mic, omnidirectional condenser mic which worked properly. We had to use a audio jack that separates mic and headphone signals. Also, mic needs biasing circuit to function correctly.
- Loading of the sub-circuits when assembled together
  - After assembling the circuits, they all behaved differently due to the loading effects, including the 9V battery. To rectify this problem, we need buffer circuits at loading junctions.