EE447

Term Project Final Report

Emrullah Çelik 2307320

6 February 2022

Contents

Introduction

In this report, my work on the EE447 term project will be explained. In this project, we are getting sound with a microphone, and we find out its frequency and amplitude values. According to dominant frequency and its amplitude motor speed and RGB LED's brightness and color changes. For different parts, if possible, I modified my code from preliminary works. Otherwise, implemented it from scratch.

General Overview of The Code

I divided my code into several subparts.

The subparts are:

- Getting Analog Values from microphone and storing 256 samples
- Configuring DSP library and taking FFT of sound samples and then finding a dominant frequency and its amplitude.
- Rotating the motor according to the detected frequency value.
- Changing direction of the motor in GPIO_PORTF_HANDLER interrupt service subroutine.
- Displaying the relevant information on the LCD screen.
- Generating PWM signals so that LED's brightness changes with amplitude of the detected dominant frequency.
- Changing Amplitude threshold with a potentiometer using ADC module (after adding keypad this part become useless)
- Changing amplitude, low frequency, and high-frequency thresholds with the keypad.

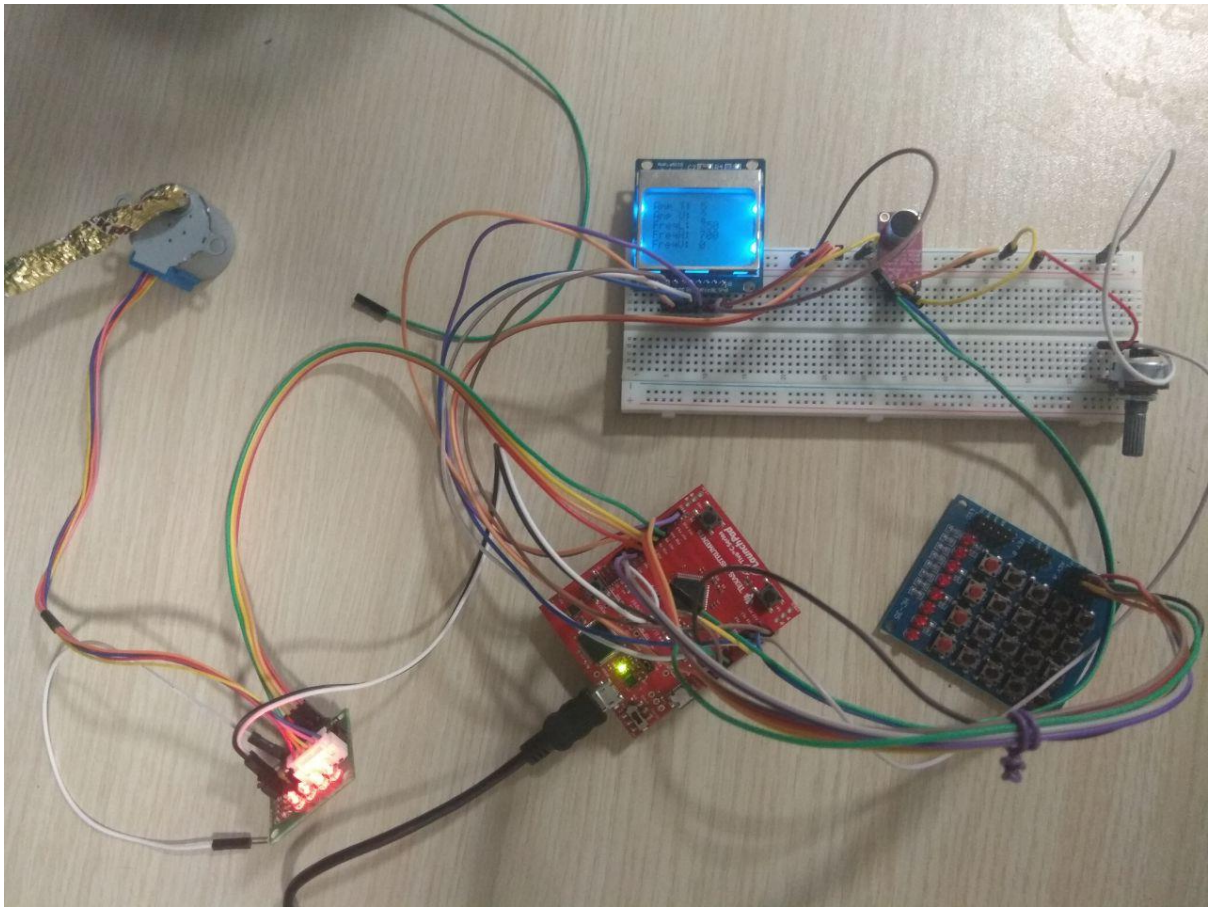


Figure 1. All the parts of the project

Initializations

At the very beginning, I initialized all the ports, timers, ADC, and UART. After initializations, I configured the memory that I am using. Since there is a lot of subroutines and interrupts, I did not want to rely on registers. Therefore, I store all the important information in memory. When needed I load them to register and after my job finishes, I store the last value back to memory. Moreover, some strings are printed at the beginning. These strings are not changing throughout the code, so I just write them at the beginning and never change them again.

Main Loop

In my main loop, I just have a BL to a subroutine in which I update amplitude and frequency values including thresholds. To prevent continuous updates and I made a counter in the main loop so that it just updates every 1 second approximately. Moreover, in my main loop, I read the keypad. While the program is running pressing to K13 makes the system enter into threshold setting mode. It excepts at most three values then waits for which threshold value to change. K14 is for amplitude threshold, K15 is for frequency lower threshold, K16 is for frequency higher threshold. The pressed switch can be observed from the Termite. When K14, K15, or K16 is pressed it writes a new threshold value to the specified field.

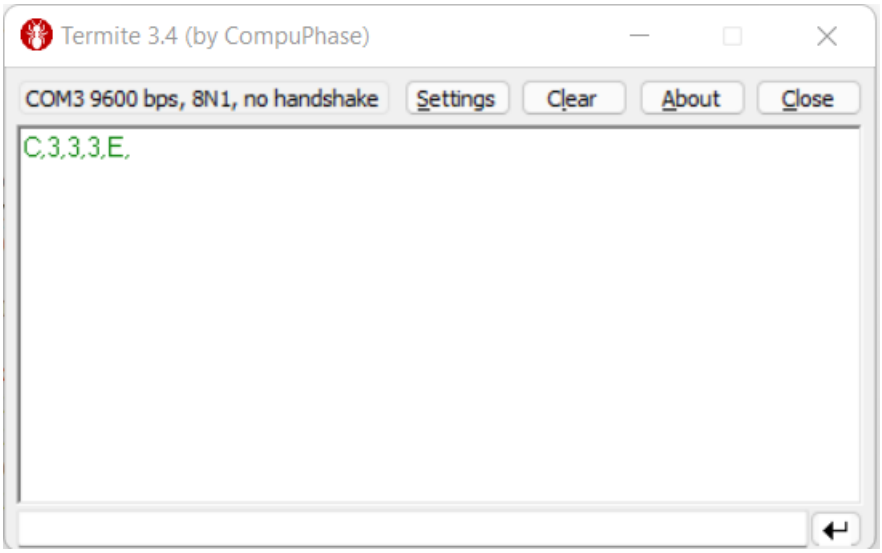


Figure 2. The Sequence for Making Low Frequency Threshold 333

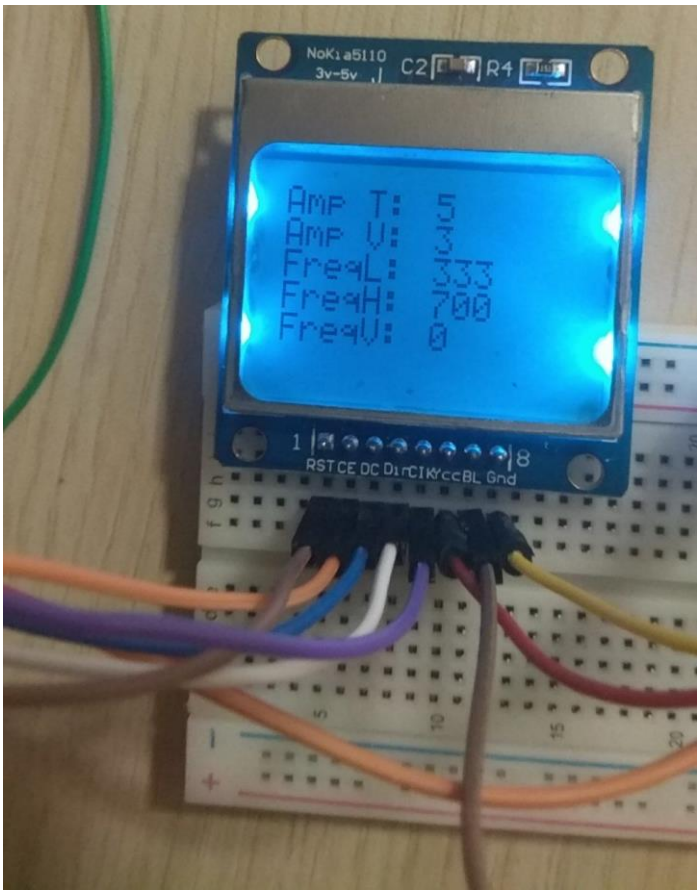


Figure 3. Changing Low Threshold Value to 333

System Tick Subroutine

After configuring the SysTick, I create a subroutine to execute when an interrupt occurs. In this ISR I sample the microphone and potentiometer and store the sampled value in memory. I also count the number of samples gathered. When it reaches 256, I call the take_fft subroutine and find the dominant frequency and its amplitude. I also calculate the speed proportional to frequency in take_fft subroutine. I store all the important information in memory.

In order to sample the microphone, I am using sample sequencer 2 and for the potentiometer, I am using sample sequencer 3. Sampling is done in the ATD_SAMPLING subroutine. This subroutine is called from SysTick ISR. I call take_fft subroutine whenever 256 samples are collected.

GPIO Port F Interrupt

When SW1 or SW2 is pressed an interrupt occurs. In order to configure it, I enabled the Port F to interrupt on NVIC interrupt enable register bit 30. I also set its interrupt priority so that it does not prevent other parts of the code from executing. “SwitchHandler” is my ISR for changing the rotation. At the end of my ISR, I clear the interrupt so that it can enter the interrupt again when it occurs.

Rotating The Motor

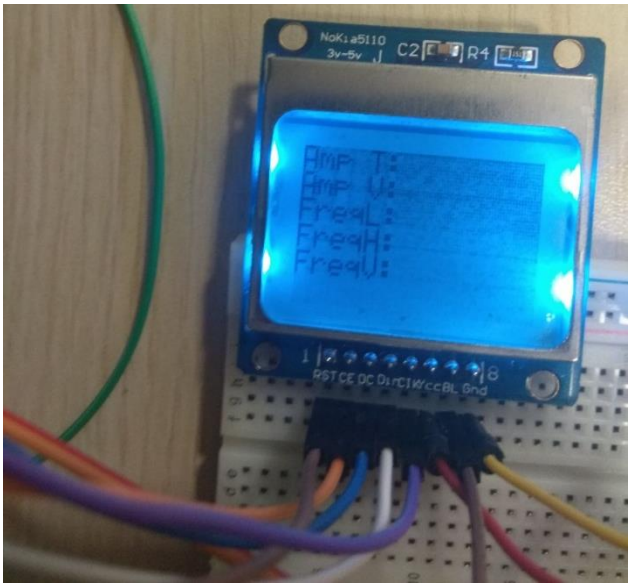
In order to rotate the motor periodic signal is needed. For that purpose, GPTM timer0 is used to create that periodic signal with interrupt. When the interrupt occurs ISR is called. In the ISR I check the rotation direction and update the signals according to rotation direction. I also update the speed of the motor. When the detected frequency is high, I update the TAILR register so that value is small. When the detected frequency is low, I store a big value to the TAILR register so that the motor rotates slowly.

I also update the interrupt priority of timer0 so that the motor does not stop rotation due to other parts of the code.

PWM Signals for LEDs

In order to change the brightness of the LEDs according to dominant frequencies amplitude, I created an interrupt with timer1. I map my magnitude values between 0 and 100. My mapped amplitude value is loaded to the TAILR register during turn on and 100-amplitude is loaded to TAILR during turn off time. That way I generate a PWM signal that changes the brightness of the LED with changing amplitude. I also check amplitude and frequency threshold conditions in this ISR.

LCD Display



Amplitude, frequency low and high thresholds, and current amplitude, frequency values are displayed on the LCD display. To prevent values to change rapidly I made a counter in the main loop so that values update relatively slow.

Figure 4. The Non-Changing Strings

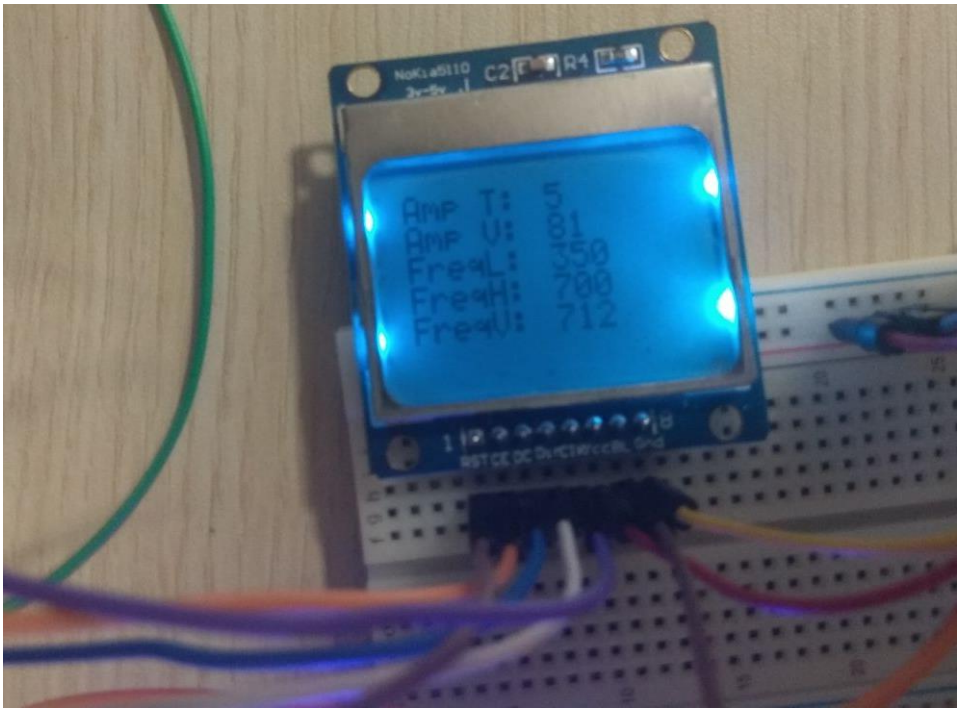


Figure 5. When Code is Running All Values are displayed

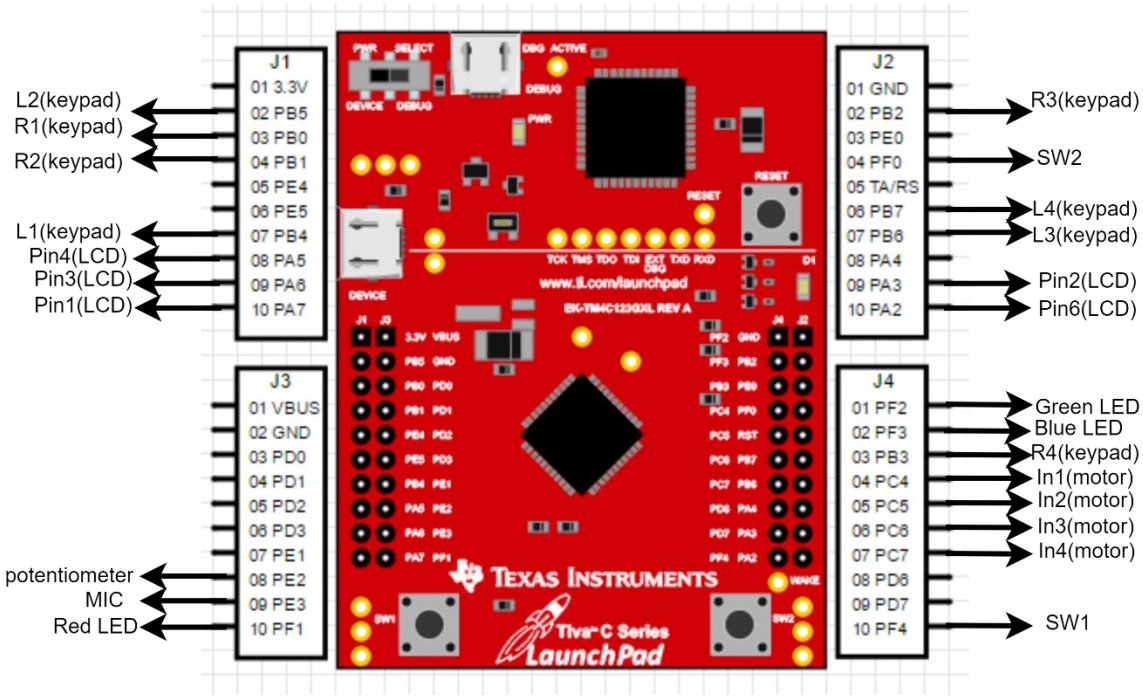


Figure 6. Pin Configuration in My Project