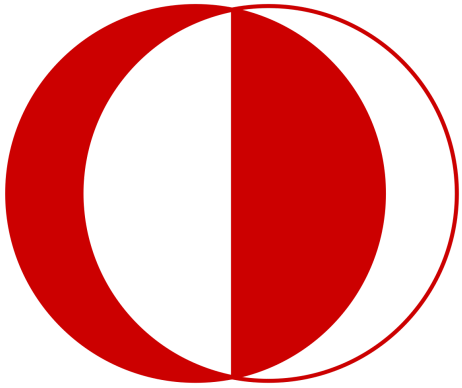


EE447 INTRODUCTION TO MICROPROCESSORS

Term Project:

Audio Frequency Based Stepper Motor Driver

Zeynepnur ŞAHİNEL

2305399

Submit Date: 06.02.2022

TABLE OF CONTENTS

Introduction

This report explains the audio frequency-based stepper motor driver project. The system updates itself according to the captured noise frequency and other user inputs. The project consists of three critical subsystems being audio sampling, frequency detection, and user insterface. In this report, initially overall system description will be made, following this description three subsystems and their internal subsystems are going to be examined. Finally conclusion of the term project and references, appendixes in which results are shown will be given.

Overall System

In this project, a stepper motor drive system based on the frequency of the applied audio signal. The driver system has three subsystems being audio sampling, frequency detection, user interface respectively. Figure 1 shows the overall system diagram.

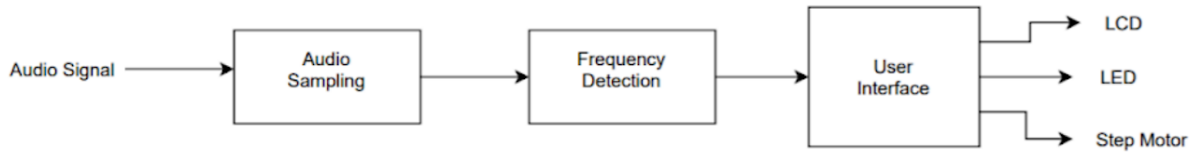


Figure 1. Overall System Block Diagram

Figure 2 shows the overall algorithm. Initially, audio signal created by the microphone module is given as analog input to the board. Using this data 256 samples are taken continously using SysTick handler. After completing the sampling array frequency detection subsytem is started. Here using CMSIS library for DSP, maximum amplitude and the related frequency values are found. Using these values comparisons are made inside the User Interface subsystem. Detailed process is going to be examined at each subsystem section.

1. Wait for SysTick handler to store 256 readings in the array
2. Call the FFT subroutine
3. Compute the complex magnitudes of elements of the frequency domain sequence
4. Find index and magnitude of the dominant (highest-magnitude) frequency component
5. Convert that index to frequency
6. If that magnitude is larger than a threshold, turn on the corresponding LED and update the stepper motor speed
7. If more than a second has elapsed after the last LCD update, show the frequency and magnitude values on LCD
8. Loop back to 1.

Figure 2. Pseudocode for the Overall System taken from Lab Manual.

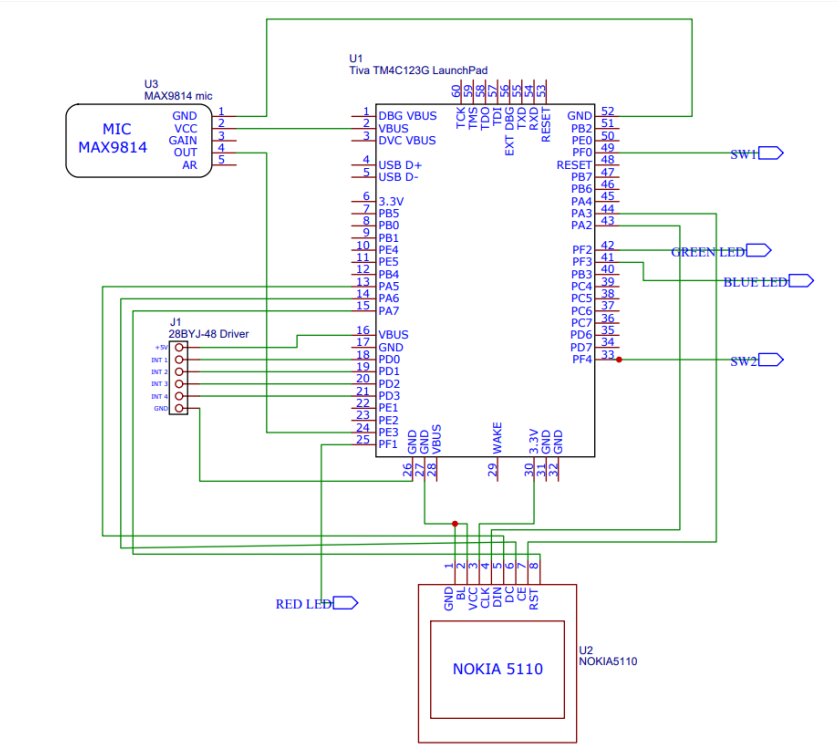


Figure 3. Overall System Pin Diagram

Subsystems

Audio Sampling Subsystem

In this subsystem initially, the audio signal is converted to the electrical signals through the microphone module MAX9814. And this module is used as input to the ADC module in TM4C123. The ADC Module continuously samples the audio signal and stores these samples in an array consisting of 256 elements. For this purpose, ADC is read using SysTick Interrupt Handler.

```
                MOV R11,#256      ;sample number
                LDR R4,=SMPL_ADD
                BL ATD0_PE3_INIT
                BL InitSysTick

start           LDR R1, =ADC0_PSSI ; start sampling
                ; Enable Sequencer 3 in ADC0_PSSI
Sample          LDR R0, [R1]
                ORR R0, R0, #0x08 ; set bit 3 for SS3
                STR R0, [R1]
                B  Sample
```

Figure 4. Taking samples continuously in the main

Figure 4 shows that the sampling routine and branching to the ADC and SysTick initializations. Here, R11 holds the element number of the sample array which is 256. Later R11 is going to be used in the SysTıck_ISR. Moreover, R4 holds the sample address in which memory starts to keep the 256 samples. Then ATD0_PE3_INIT is branched. This subroutine arranges the AIN0 in tiva board. It should be noted that MAX9814 microphone module is connected to the E3 port. Later, Systick is initialized. For this purpose, the InitSysTick file is used which is given in the Lab3 manual. After loading the PSSI address, sampling rouitne starts and continues always.

Figure 5 demonstrates the SysTick_ISR Flow Chart. SysTick_ISR is called inside the Systick_Handler inside the Startup.s file. It provides a continuous sampling process at every 0.5 ms. Here, before starting to the sampling process RIS, FIFO and ISC register address values are loaded. During the sampling process third bit of the RIS (flag) is checked whether sampling is completed or not. If it is completed result in the FIFO is loaded to the R9. Since the microphone module has a 1.25 DC offset, this offset is substrated. Then R9 is left shifted by 16 bits for the proper FFT process. Afterward, R9 is stored in the memory which is initialized in the main part (R4). Then R4 address pointer is increased by 4 to place new input values. To maintain process at every call, flag must be cleared. Finally R11 is checked whether all 256 samples are taken, if it is completed program goes to FFT_SAMPLE subroutine.

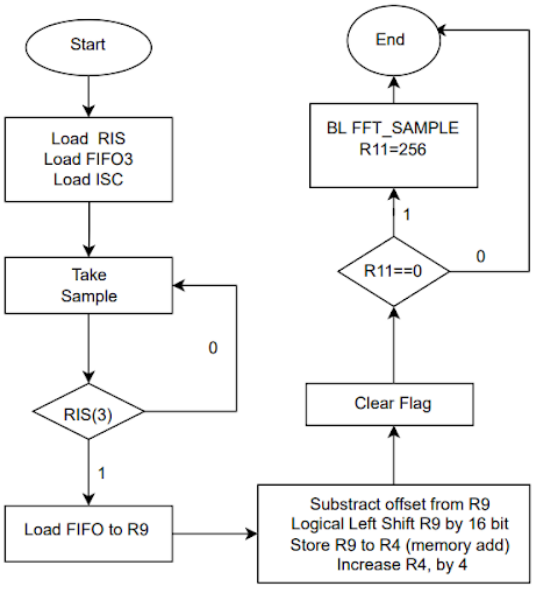


Figure 5. SysTick_ISR Flow Chart

Frequency Detection Subsystem

This subsystem is responsible for the calculation of the frequency using 256 samples array. While the frequency detection FFT (Fast Fourier Transform)is used. To implement FFT, arm_cfft_q15 function is used. Equation 1 shows the DFT (discrete Fourier transform)

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi kn}{N}}, \quad k = 0, 1,...,\text{N-1} \qquad (1)$$

FFT is a computational algorithm that diminishes the computing time and complexity. This transform takes N (N=256 for this case) element sequence input in the time domain, then it converts the sequence into frequency domain with a complex value. In this subsystem after taking FFT of the 256 sample array, complex magnitudes of the elements are computed. This computation is done by the *arm_cfft_q15* function. Then the index having the highest magnitude is chosen and given as an output to the same memory location as the input array.

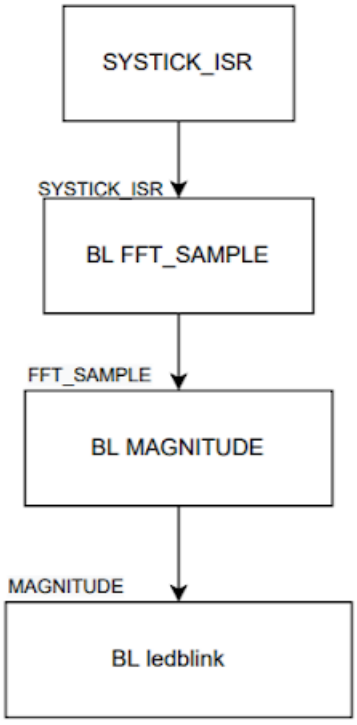


Figure 6. Frequency Detection Subsystem Block Diagram

Figure 6. indicates the branch operations inside the SysTick_ISR. Inside the interrup subroutine FFT is branched. In the FFT_SAMPLE branch with the help of the *arm_cfft_q15,* frequency of the 256 samples are computed in terms of real and imaginary compenents. Fig 7. shows the modified data starting from the sample memory address.

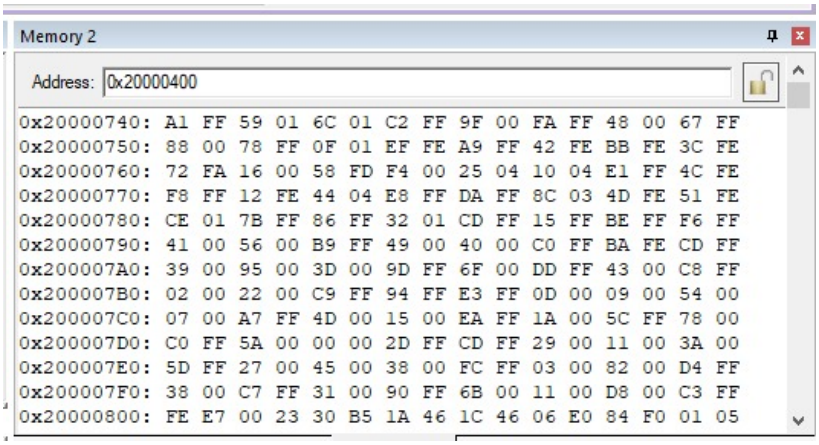


Figure 7. Memory demonstration after FFT is taken

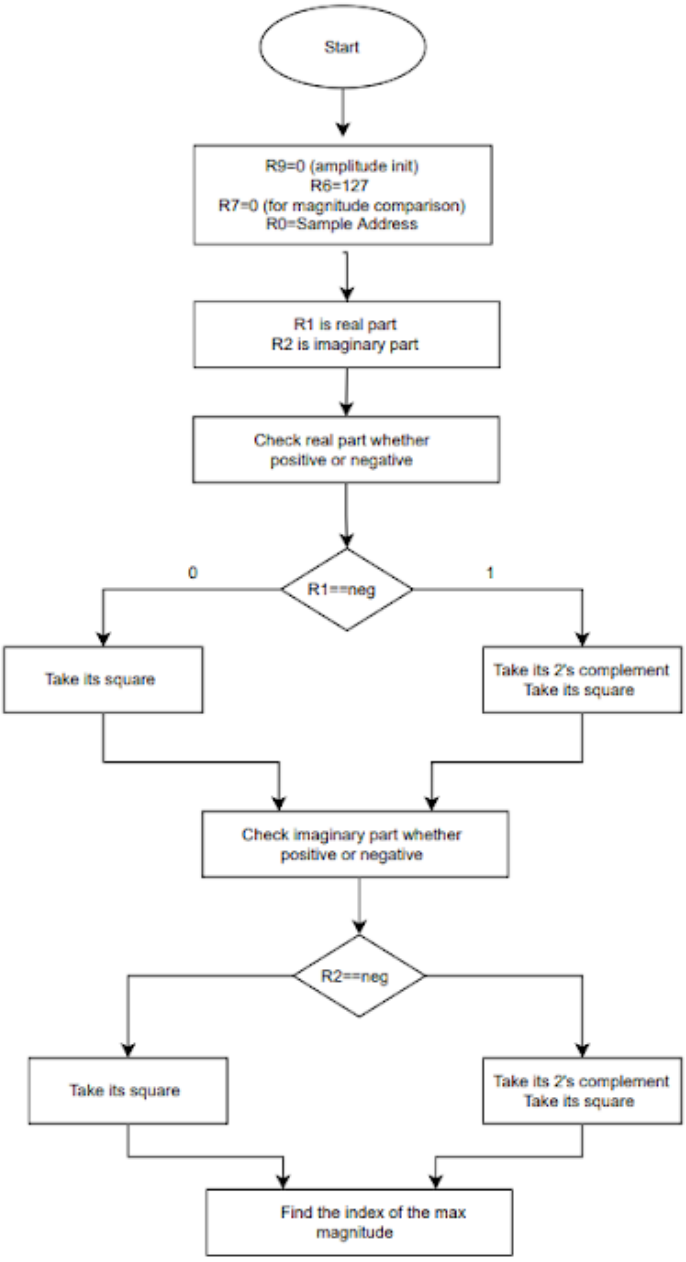


Figure 8.a MAGNITUDE Subroutine first part until finding index of the max amplitude

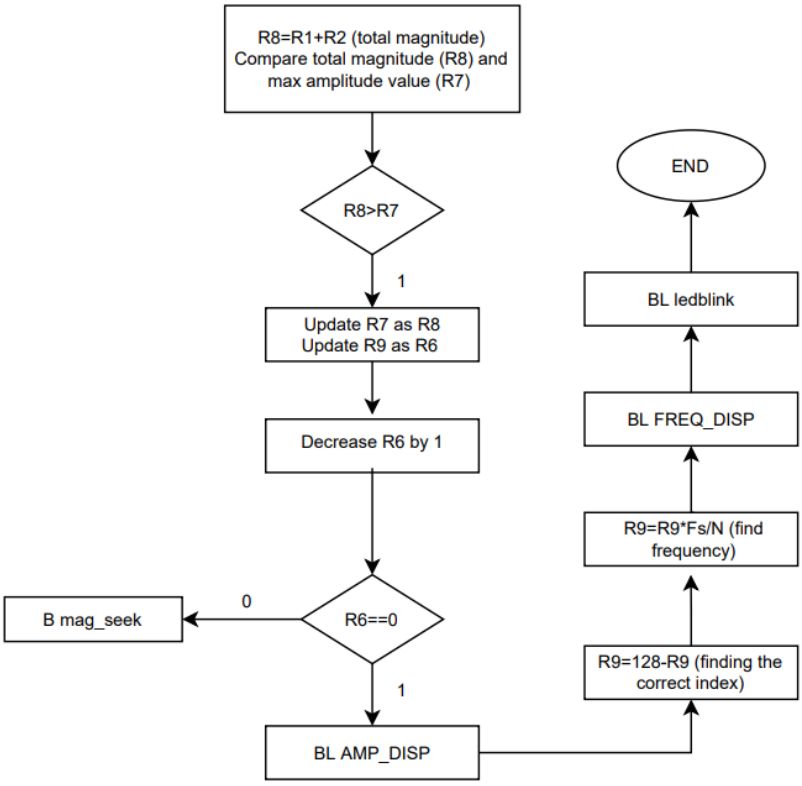


Figure 8b Continue of the MAGNITUDE Subroutine

Fig 8a and 8b shows the MAGNITUDE subroutine. This subroutine is called inside the FFT_SAMPLE subroutine. In the first part of the code, R9 which holds the amplitude index data later is initialized as zero. This initialization should be done for the next 256 sample array since R9 takes the maximum amplitude index of the previous array. R6 is the index counter that is set to 127 due to the fact that FFT is symmetric. Therefore, the middle of the sample array is taken as a start index. Moreover, R7 which is going to hold the maximum magnitude is set as zero initially. Later, real part and imaginary part of each sample are loaded to the R1 and R2 respectively. Then, both R1 and R2 are checked whether they are positive or negative. If they are negative additional 2's complement calculation is done, then the multiplication is applied for the magnitude calculation. After calculating both imaginary and real magnitude part b starts. In part b these two values are added to find the total frequency magnitude of each sample. Then R7 (max value is compared with the total magnitude of each sample. If the total magnitude of the current sample is bigger than the previous max amplitude value, max value is updated with this current value and R9 (max magnitude index) takes the current index value. When all the indexes are scanned, Amplitude Display Subroutine is called. After returning from this subroutine corrected index is calculated since at first, the total index is taken as half. Having calculated the proper index

value, the frequency value is found from the FFT formula which is given in Eq. 1. The result obtained from this equation is used in the Frequency Display subroutine. After returning from this subroutine ledblink is subroutine is called.

User Interface Subsystem

In this subsystem, output relations are determined such as according to the coming frequency magnitude value a led is lightened, stepper motor speed has changed and the results are displayed on. Here, at low frequencies Red, middle frequencies Green and the high frequencies Blue led is on. If no frequency is detected led is going to be turned off. Moreover, stepper motor directions are defined through the switches on the board.

Figure 9. shows the Led Control subroutine. In this routine, initially, R1 and R2 are loaded with maximum amplitude and its frequency value respectively. Later threshold values for the amplitude, low frequency, and upper frequency are set. Later consecutively the comparisons are made. Firstly, the amplitude value is compared with the threshold, if it is lower than this value led is turned off. Then upper and lower frequencies are compared and proper leds are turned on. (For the **results** please look at **Appendix**)

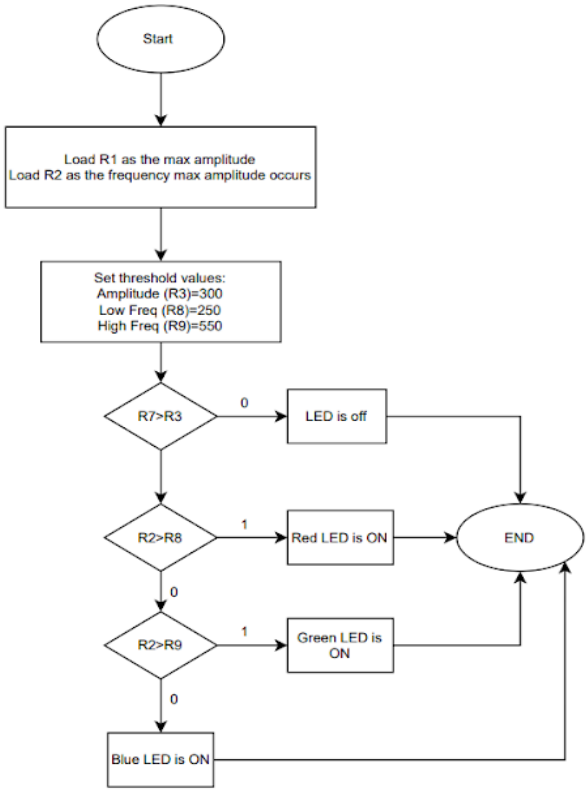


Figure 9. Led Control Subroutine Flow Chart

Fig 10. demonstrates the Motor Control Subroutine. The crucial part of this subrotine is to arrange the TAILR value according to the frequency value. The aim of the project is to drive faster when frequency is higher and vice versa. Therefore a formula is used to calculate the TAILR value. It is in the form y=-mx+n. Here x, y and z represents frequency taken from R7, TAILR value and frequency constant respectively. x can take maximum 1000 Hz values due to the imperfection of the microphone module. Moreover TAILR value can take 2000 as minimum to drive the stepper motor. m is assigned to 14 (0x0e) and n is set to 16000 as max. For instance if the returned frequency value is 1000 Hz, y is 2000 which is the min TAILR value, max speed in other words. Also, in this subroutine according to the rotation bit which is set by the buttons in the Button Control Interrupt Subroutine clockwise or the counter clockwise continous return is provided. This control subroutine is called under the Timer0A_Handler inside the Startup.s file.

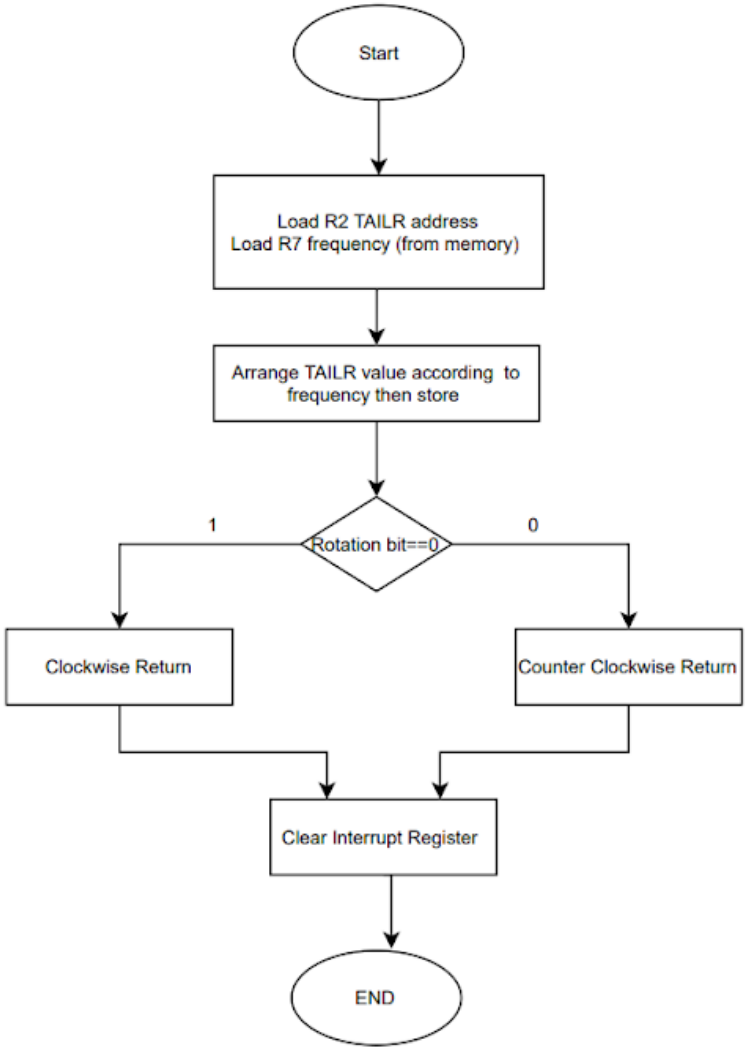


Figure 10. Motor Control Subroutine Flow Chart

Button Check:

Using SW1 and SW2 direction is determined. While using this property, interrupt is used. To enable the interrupt, inside Motor_Init IS, IBE, IEV and IM registers are arranged to sense the switch press. This subroutine is also called from the GPIOPortF_Handler inside Startup.s If the interrupt occurs Button Check is run. In this subroutine according to the RIS value cw or ccw assignments are done. If SW1 is pressed ccw branch is jumped. Otherwise cw branch is operated. In the cw case rotation bit as zero whereas in ccw case this bit is assigned as one. After making these assignments ICR is cleared.

LCD Control:

In this part, Nokia 5110 LCD is used to show the all three thresholds and the changed amplitude and the frequency values. Serial Peripheral Interface (SPI) is used as a communication protocol. Inside LCD_Init module three configuration is done being, GPIO, SPI and Nokia 5110. All of these configurations are detaily examined and commented which are shown in the Fig11a-13b.

1. Enable the clock for GPIOx (RCGCGPIO)
2. Wait until GPIOx is ready (PRGPIO)
3. Configure the CLK, CS (FSS), MOSI (Tx), and MISO(Rx) pins as a digital pin (DEN)
4. Set directions for the pins (DIR)
5. Configure the CLK, CS (FSS), MOSI (Tx), and MISO(Rx) pins for their alternate function (AFSEL)
6. Configure the CLK, CS (FSS), MOSI (Tx), and MISO(Rx) port control pins to route the SSI interface to the pins (PCTL).

Figure 11a. GPIO Configuration taken from the manuel

```
;-----------------PORTA Init-----------------------------------------------

                                LDR   R1,=SYSCTL_RCGCGPIO  ;1-enable PortA
clk
                                LDR   R0,[R1]
                                ORR   R0,R0,#0x01
                                STR   R0,[R1]
                                NOP
                                NOP
                                NOP

                                LDR   R1,=GPIO_PORTA_LOCK   ;2-unlock porta

                                LDR   R0,=0x4C4F434B
                                STR   R0,[R1]
```

```
                                LDR   R1,=GPIO_PORTA_CR
                                MOV   R0,#0xFF
                                STR   R0,[R1]


                                LDR   R1,=GPIO_PORTA_DIR  ;3-set direction as
output
                                LDR   R0,[R1]
                                ORR   R0,R0,#0xEF
                                STR   R0,[R1]

                                LDR   R1,=GPIO_PORTA_AFSEL ;4-function
select
                                MOV   R0,#0x3C
                                STR   R0,[R1]

                                LDR   R1,=GPIO_PORTA_DEN ;5-set porta pins as
outputs
                                MOV   R0,#0xFF
                                STR   R0,[R1]

                                LDR   R1,=GPIO_PORTA_PCTL ;6-set pctl
                                MOV32         R0,#0x00222200
                                ;A2-ssi0clk
                                ;A3-ssi0Fss
                                ;A4-ssi0Rx------not used
                                ;A5-ssi0Tx
                                STR   R0,[R1]
```

Figure 11b. GPIO PortA Initalization inside LCD_INIT Subroutine

1. Enable the clock for SSIx (RCGCSSI)
2. Wait for the SSI peripheral to be ready (PRSSI)
3. Disable the SPI interface (CR1)
4. Set the clock rate of the SPI Clock (CPSR, CR0) accordingly. Please mind the maximum data rate that the LCD is capable of working with.
5. Set the data size to be 8-bits and Freescale mode (CR0)
6. Set the SPI mode (CR0) accordingly.
7. Re-enable the SPI interface (CR1)

Figure 12a. SPI Configuration taken from the manuel

```
;-------------------SSI Init-------------------------------------------------

                                LDR   R1,=SYSCTL_RCGCSSI ;1-set ssi0clk
                                LDR   R0,[R1]
                                ORR   R0,R0,#0x01
```

```
                    STR    R0,[R1]

                    LDR    R1,=SYSCTL_PRSSI

wait                LDR    R0,[R1]          ;2-wait for the SSI peripheral
to be ready
                    AND    R0,R0,#0x01
                    CMP    R0,#0x01
                    BNE            wait

                    LDR    R1,=SSI0_CR1     ;3- Disable the SPI
interface
                    LDR    R0,[R1]
                    BIC    R0,R0,#0x02
                    STR    R0,[R1]

                    LDR    R1,=SSI0_CR0     ;4-Set the clock rate of the
SPI Clock
                    LDR    R0,[R1]
                    MOV  R2,#0xFF3F
                    BIC    R0,R0,R2                 ;clear the all bits except
sph and spio
                    MOV            R2,#0x01C7
                    ORR    R0,R0,R2
                    ;scr =1   scr clock rate
                    ;sph=1    Data is captured on the second clock   edge
transition
                    ;spo=1    A steady state High value is placed on the
SSInClk pin when
                   ;data is not being transferred.
                    ;frf=00   frescale spi frame format
                    ;dss=0111 data size select = 8 bit data
                    STR    R0,[R1]

                    LDR    R1,=SSI0_CPSR   ; clk prescale divisor
                    LDR    R0,[R1]
                    MOV  R0,#20
                    STR    R0,[R1]

                    LDR    R1,=SSI0_CR1   ;7-Re-enable the SPI
interface
                    LDR    R0,[R1]
                    ORR    R0,R0,#0x02
                    STR    R0,[R1]
```

Figure 12b. SPI Initalization inside LCD_INIT Subroutine

1. To initialize the Nokia screen first toggle the Reset pin by holding it low for 100ms then setting it high.
2. Send the following commands to initialize the display
   • Set H=1 for Extended Command Mode, V=0 for Horizontal Addressing
   • Set VOP . You may need to sweep values between 0x[B0-C0] for correct operation.
   • Set temperature control value. You may need to sweep values between 0x[04-07] for correct operation.
   • Set voltage bias value as 0x13.
   • Set H=0 for Basic Command Mode
   • Configure for Normal Display Mode
   • Set Cursor to detemine the start address
3. Send data to be displayed.

Figure 13a. Nokia 5110 Configuration taken from the manuel

```
;-----------------Nokia 5110 LCD Config----------------------------------------

                                LDR   R1,=GPIO_PORTA_DATA
                                LDR   R0,[R1]
                                BIC   R0,R0,#0x80   ;set reset pin (A7) low
                                STR   R0,[R1]

                                BL    DELAY100      ;delay 100 msec

                                ORR   R0,R0,#0x80    ;set reset pin (A7) high
                                STR   R0,[R1]

                                BIC   R0,R0,#0x40   ;clear DC(A6) pin
->command
                                STR   R0,[R1]

                                LDR   R1,=SSI0_DR
                                MOV   R0,#0x21      ;power down control db5=1
H=1, V=0
                                STR   R0,[R1]

                                MOV   R0,#0xc0        ;vop6=1
                                STR   R0,[R1]

                                MOV   R0,#0x07     ;temperature control
                                STR   R0,[R1]

                                MOV   R0,#0x13        ;voltage bias 0x13
                                STR   R0,[R1]

                                MOV   R0,#0x20        ;H=0
                                STR   R0,[R1]

                                MOV   R0,#0x0C
```

```
                                STR    R0,[R1]

                                BL DELAY100

                                MOV   R0, #0x40
                                STR    R0, [R1]

                                MOV   R0, #0x80      ; Configure for Normal Display
Mode
                                STR    R0,[R1]
                                BL    DELAY100
                                LDR    R1,=GPIO_PORTA_DATA
                                LDR    R0,[R1]
                                ORR    R0,R0,#0x40    ;Set Cursor to detemine the
start address
                                STR    R0,[R1]
```

Figure 13b. Nokia 5110 Configuration inside LCD_INIT Subroutine

ROW1: In this subroutine AMPLITUDE THR = 300 is displayed.
ROW2: In this subroutine LOW FREQ THR = 250 HZ is displayed.
ROW3:In this subroutine UPPER FRQ THR = 550 HZ is displayed.
ROW4: This row is not used.
ROW5: In this subroutine FREQUENCY is displayed.
ROW6: In this subroutine AMPLITUDE is displayed.

These five row subroutines display the constant data and word. However in the Freq and Amp_Disp subroutines taken amplitude and the frequency values are printed to the screen. Inside these subroutines frequency and the amplitude values are converted to the 3-digit numbers. To print each digit WH_DIGIT subroutine is called.

Conclusion

This project is classified into three subsystems. The first subsystem is the Audio sampling subsystem. Here, an analog data array consisting of 256 samples is sampled continuously with the help of the SysTick Handler. Later frequency of each sample of this array is calculated in the frequency detection subsystem. Initially, the FFT library is used to convert the sample data into real and imaginary components, then using this data maximum amplitude in the array and its frequency value is found inside the Magnitude subroutine. Using these amplitude and frequency values several user interface operations are completed. For instance, compared the obtained frequency and amplitude data to the predetermined thresholds LEDs are turned on or off. Moreover, as a result of these comparisons stepper motor is controlled. To update its speed TAILR value of the Timer0A is updated. To achieve

desired speed y=mx+n formula is used. Using SW1 and SW2 direction of the return is determined, again using the interrupt property of the GPIO Ports. Last but not least, these thresholds and altering the frequency and amplitude values are displayed on the Nokia 5110 Screen using Serial Interface communication protocol. In conclusion, this term project is very instructive to learn how to use Interrupts, System Timer, General Purpose Timers and SPI communication protocol.

References

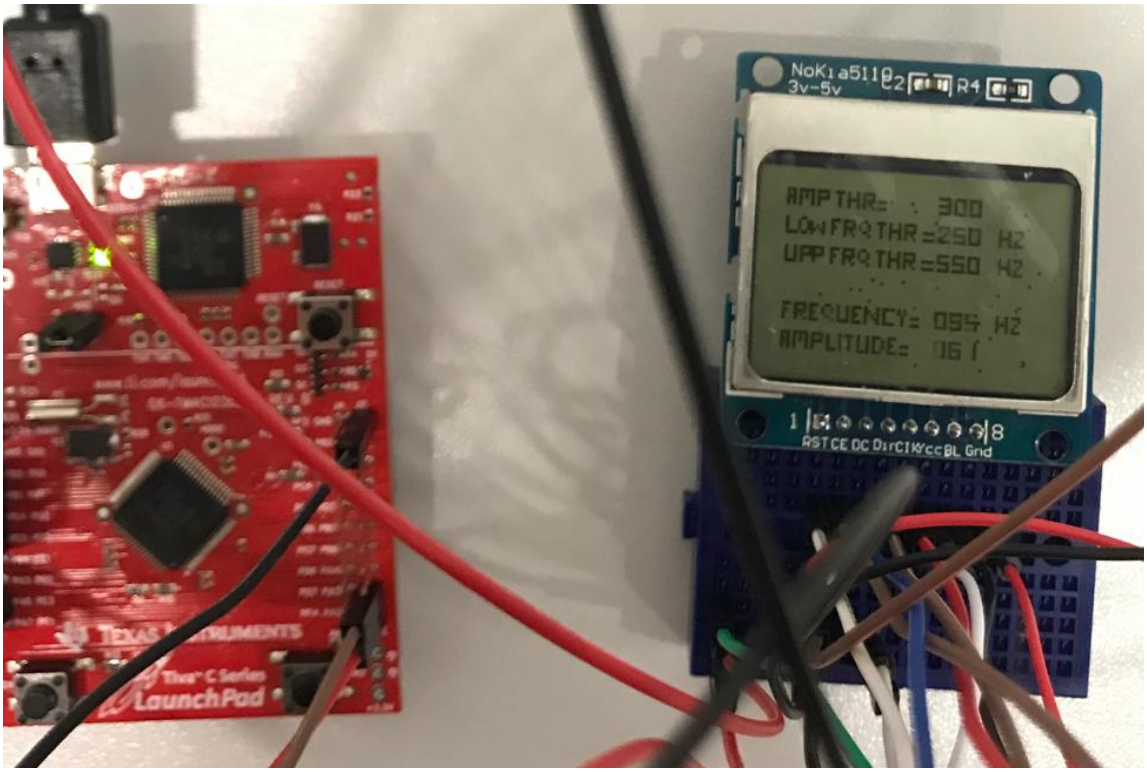
[1] Lab 3, 4, 5 and Term Project Manuels

[2] TI, “Tiva™ c series tm4c123g launchpad evaluation board user’s guide.” http://www.ti.com/lit/ug/spmu296/spmu296.pdf.

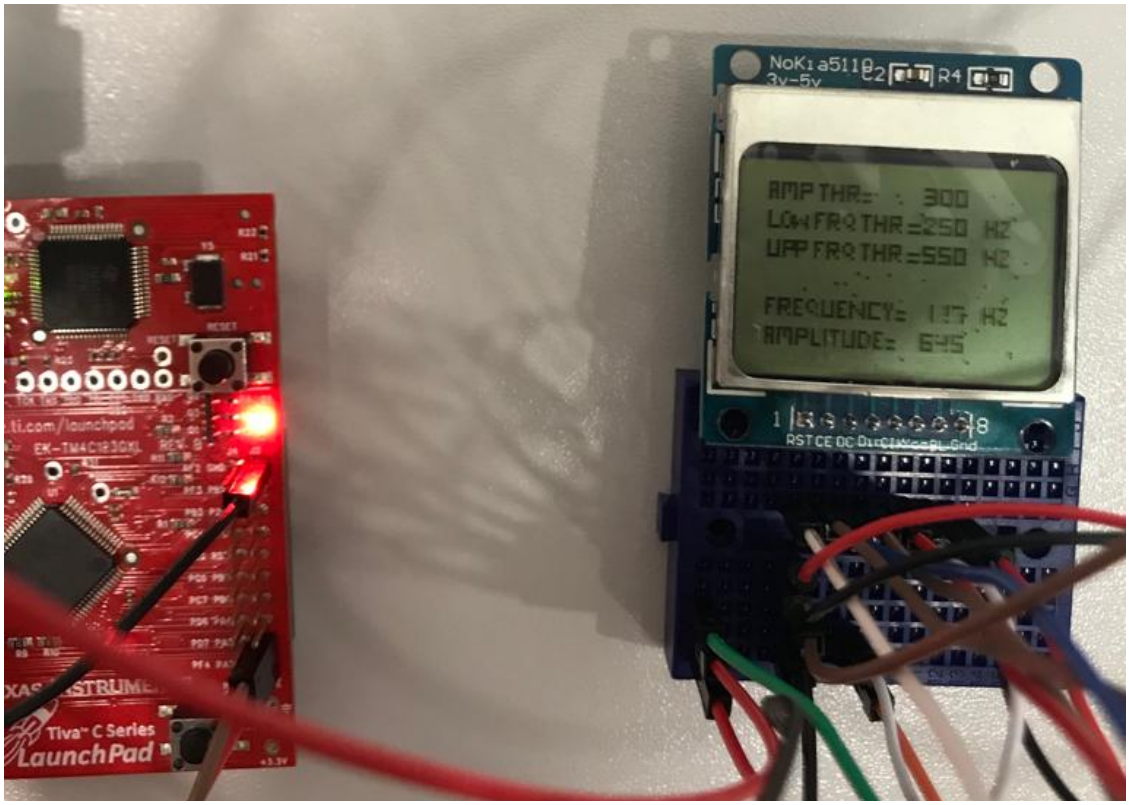
[3] TI, “Tiva™ tm4c123gh6pm microcontroller data sheet.” http://www.ti.com/lit/ds/spms376e/spms376e.pdf.

Appendix

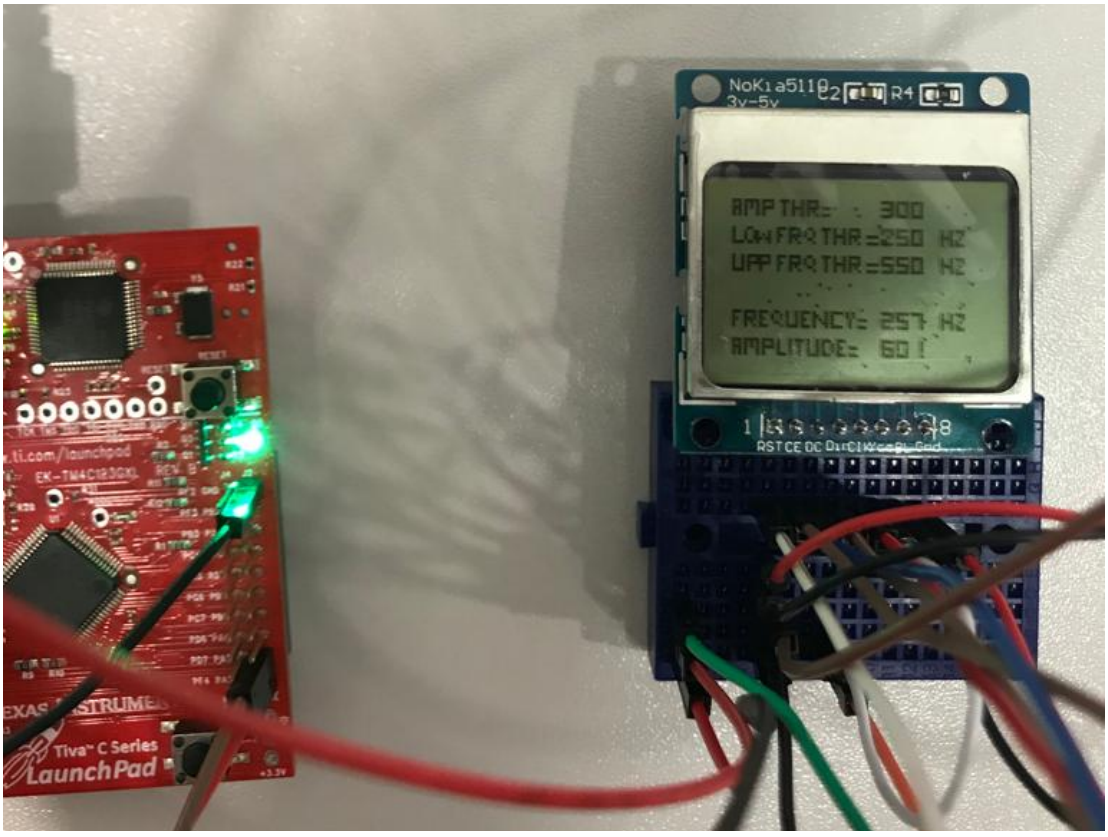
Results:



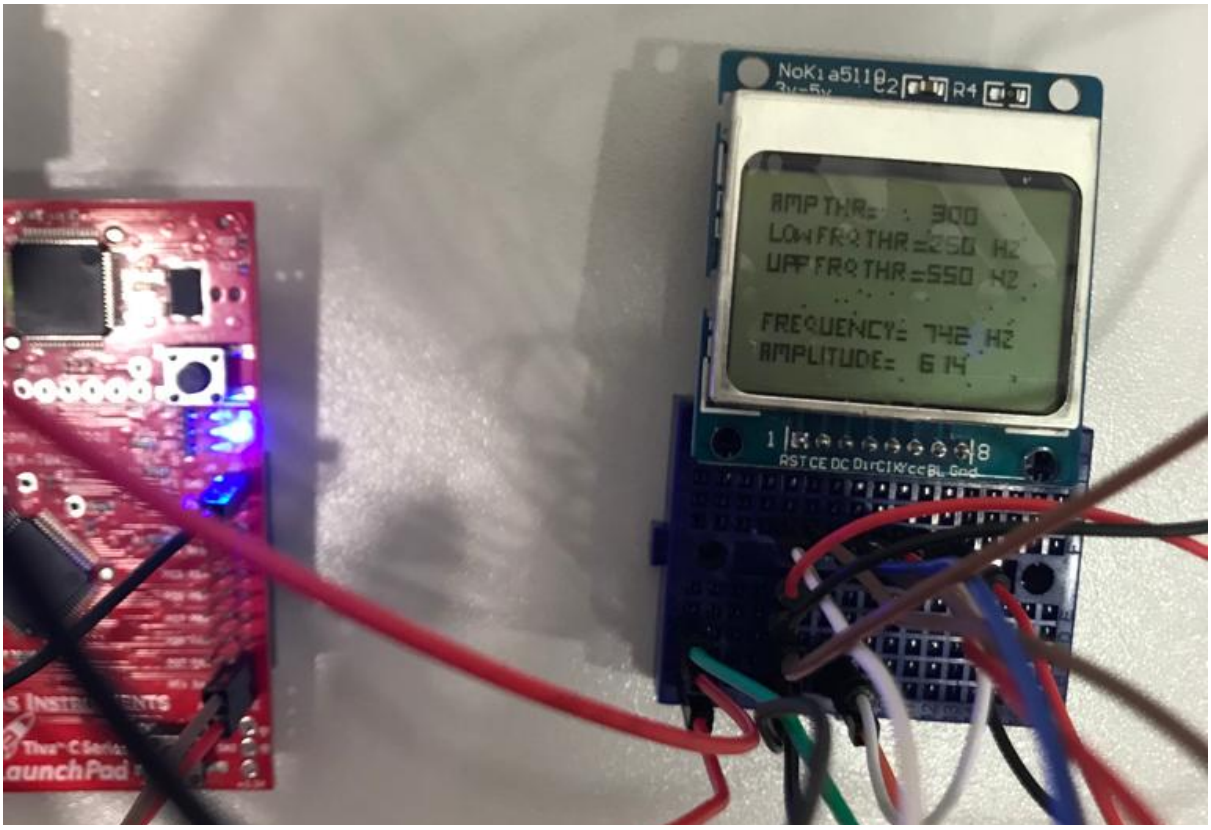
Case 1. Amplitude is lower then amplitude thereshold



Case 2. Frequency is lower than lower frequency threshold and its amplitude is higher than amplitude threshold



Case 3. Frequency is higher than lower frequency and lower then upper threshold and its amplitude is higher than amplitude threshold



Case 4. Frequency is higher than upper frequency threshold and its amplitude is higher than amplitude threshold

Demonstration

Above links shows the demonstration of the motor speed change in addtion to LCD and LEDs.