Coursework 1

Pitch detector

Sören Schreiber

K1549360

**Contents**

[**Introduction** 3](#_Toc447204275)

[**Analysis** 3](#_Toc447204276)

[**Design** 3](#_Toc447204277)

[**Implementation** 4](#_Toc447204278)

[**Pitch detector** 4](#_Toc447204279)

[**Fast-Furier-Trasnform** 5](#_Toc447204280)

[**LED control** 5](#_Toc447204281)

# **Introduction**

The first coursework assignment poses the problem of a pitch detection application.

The goal of the application is to specify, design and implement a C-solution using the “dsPIC StarterKit 1”, which can estimate and identify the pitch of an 8-bit signal, sampled at 8 kHz.

This real-time application will visually display the detected pitch using the peripherals (LEDs) provided by the DSP kit.

The proposed solution has been tested by means of a signal generator app on an android phone (using LINE-IN and MIC).

# **Analysis**

The assignment requires that the developed solution is able to gather process and display the results of a pitch detection in real time.

This will be accomplished by estimating the pitch of the original input.

The pitch will be shown in different categories, i.e. LOW, MEDIUM, HIGH.

The signal is specified as a real valued 8-bit signal. This means that the pitch detection will have to be able to deal with the problem of negative frequency in the frequency spectrum derived by the FFT.

The signal is to be sampled at 8000 Hz (8 kHz). Since the signal is real valued half will be a negative frequency which is ignored in this approach. This means the maximum frequency that can be measured will be 4000Hz (4 kHz).

The application will take an audio input using the MIC or LINE-IN peripheral (jumper) and return the result using the 3 LEDs provided by the DSP StarterKit 1.

# **Design**

The project is designed in an object orientated way, meaning most functions will be written as reusable header and source code files.

The first step is the implementation of an audio input reader, which allows transforming an audio signal into a data set.

To sample the audio signal, a frame size needs to be defined. This frame size will and must stay the same everywhere in the program to make sure that no overflow error occur.

For this project the defined frame size is 256. This number allows for a good resolution/precision, while making sure the memory of the chip set is sufficient.

The audio handler data types and sizes are defined and needed in accordance with the functions using them.

The data type for the read audio input will be an array defined as a “fractional” since the FFT requires the input to be in this particular format. It has the size equal to the selected frame size.

The Pitch detector also has variables for “state” and “pitchresult”. Since they contain only numeric values, the data type int has been chosen.

After the audio is sampled it is given to the FFT functions which will transform the data into the frequency domain and detect the pitch by finding the maximum of the power spectral density (PSD). It calculates the pitch by matching the result frequency pin to the value of each pin in the frequency spectrum and returning the result.

The variables given to the FFT are set to be an integer, fractional and fractional complex. This is due to the requirements specified by the FFT functions.

The final step of the pitch detector is to visualize the results of the detection by using the LEDs provided on the chip set.

The frequency spectrum is divided into 5 groups:

* 0-800
* 801-1600
* 1601-2400
* 2401-3200
* 3200-4000

Each group is identified by a predetermined LED combination.

# **Implementation**

The implementation for this project consist out of a number of functions which are used in an object orientated structure. These function control the LEDs, input and DSP functions. The “Pitch detector”-main will only call these functions.

The solution is implemented on a pre-existing project called “Audio Pipe”, which is a simple input to output pipeline. This way the audio reader is set up and functional already.

This pitch detector works in real time. When a signal is inputted into the audio in peripheral, the signal is processed and the results displayed before the next sample is taken by the audio reader.

The following will explain the functionality of each separate file and explain what its purpose is.

## **Pitch detector**

The pitch detector is the main function in the application. It handles the input coming from the audio peripheral and calls functions according to its design.

The audio is read through an “ADCChannelbuffer”, which samples the input with the given frame size and transform it into a fractional data type. This data is then given to the fft functions. The returned pitch level is used in the LED control to display the results.

## **Fast-Furier-Trasnform**

The FFT is the major part of this application. It transforms the given signal into the frequency domain in which the pitch of any given signal can be determined. The FFT function provides a fractional complex dataset for the pitch detector. Since the input signal is expected to be a real signal, the FFT will return an array of data which is set up as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | …. | N/2 | N/2+1 | … | N |
| 0 | …. | w | -w | … | -1 |

To compensate for this problem, the pitch detection method only looks at half of the given data points.

The pitch detector takes the given data and finds the highest value. If there are multiple occurrences of the same value, the position with the highest index will be returned.

This position or pin is then used to determine the approximate frequency/pitch of the signal.

This is accomplished by multiplying the pin number with the value of each pin (calculated by dividing the sampling frequency by the frame size).

After the pitch level is detected it is returned to the pitch detector for further processing.

## **LED control**

The LED control focuses on the LED peripherals provided by the DSP kit. It is responsible for providing a ready state, error state and the control for individual LEDs.

The ready state is called at the start up of the application indicating that it is ready to perform its function.

The error state will only show when an error occurred within the program.

The individual LED control is used to display the pitch level for the user. It is divided into 5 segments of different frequencies:

* GREEN/LOW: 10-800Hz
* GREEN-YELLOW/LOW-MEDIUM:801-1600Hz
* YELLOW/MEDIUM:1601-2400Hz
* YELLOW-RED/MEDIUM-HIGH:2401-3200Hz
* RED/HIGH:<10Hz & >3200

Note: Due to the frame size and the correlating precision, frequencies under 10 Hz will show as a high frequency.