Abstract

Over the course of seven months, a study was undertaken to determine if it was possible to take a low memory cost Fast Fourier Transform program and run it on a platform which does not have a lot of memory in total, such as an Arduino. This study stems from the fact that many Fast Fourier Transform programs require a lot of memory to run effectively. Under the direction of Dr. Anna Gilbert, a professor at the University of Michigan, we conducted such a study using a Fast Fourier Transform program which she designed. This program uses less memory and is faster than normal FFT programs because it takes fewer samples while still maintaining the accuracy of the results. The results of the study will help to prove that it is possible to create a Fast Fourier Transform program which uses less than 125 kilobytes of memory while still maintaining accuracy, and that Dr. Gilbert’s Fast Fourier Transform program is a viable substitute for normal FFT programs.

Purpose and Goals

The main goal of this project is to run the Fast Fourier Transform program created by Dr. Anna Gilbert on a low memory processor (Arduino), while still maintaining the original accuracy of the program which is equal to that of other Fast Fourier Transform programs running on computers with a lot of memory. In order to test this, we will first convert Dr. Gilbert’s original code from MATLAB style code into C++, put the code onto an Arduino, test the code using real world data obtained from sensors, and then compare the results to the original MATLAB code as well as a more memory intensive FFT program. This would then prove that Dr. Gilbert’s FFT program is not only able to run optimally on a low memory processor such as an Arduino, something that other FFT programs would never be able to do, but also that the program is theoretically faster and more efficient than the standard FFT program.

Methods

Experimental Supplies:

For our research experiment we were supplied the necessary materials by Dr. Anna Gilbert. We used an open-source micro-controller and processing board made by the company Arduino, specifically we used the Arduino UnoTM and Arduino MegaTM in order to see which board would be able to run the code better. We also were provided all the necessary sensors and connecting wire needed to be able to power the Arduino and take sample data from real world sampling sources. These sensors included but are not limited to: breadboards, connecting wire, Arduino power cable, pressure sensor, and miscellaneous buttons.

References


Initial Results and Data:

We were successfully able to implement Dr. Gilbert’s Fast Fourier Transform code onto an Arduino. The translated final code was all able to fit onto an Arduino UNO, which has 256 kilobytes of storage, and only used up approximately 90% of the maximum data on the microprocessor.

Unfortunately, we ran into some trouble debugging our code when we were trying to make sure that our translated program worked correctly and we were unable to test the Arduino as we intended to in time to get the results in. If we are able to work out all the bugs and if all goes well with the tests, then the data returned by the Arduino should tell us that the frequency of the sound wave created is the same as the frequency we dialed in and shown on the oscilloscope. To prove the accuracy and consistency of the returned data, we ran numerous trials with different frequencies for the sound wave. Below is the outputted data from the Arduino in graph from a few of our trials, as well as the setup which was used during the experiment.

Analyzed Data and Implications:

Even though we were unable to test the accuracy of the code on the Arduino through real world tests, we were able to prove that Dr. Gilbert’s Fast Fourier Transform code is capable of being put onto a low memory processor. In addition to this, our unofficial initial tests of the program so far have shown us that this program should still maintain the same accuracy in its answers as other FFT functions at relatively low frequencies. We are therefore confident that Dr. Gilbert’s code/way for calculating the FFT of a function is a viable substitute for the currently used way of calculating the FFT because it takes up less space and is project- ed to run quicker and maintain a relatively low margin of error even when run on a low memory processor (i.e. an Arduino UNO). This is as expected as Dr. Gilbert first proved that this code was as accurate as the normal MATLAB FFT function and found that it was also faster, to an extent. [1]