Smart Data Collection and Processing on Arduino

Calculating Fast Fourier Transforms using a Microprocessor

Project Sponser: Dr. Anna Gilbert, Department of Mathematics
Student Researchers: Justin Shetty and James Wich

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 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 affectively. 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, connective wire, Arduino power cable, pressure sensor, and miscellaneous buttons.

References

J. Zou, A. Gilbert, M. Strauss, and I. Daubechies, Theoretical and Experimental Analysis of a Randomized Algorithm for Sparse Fourier Transform Analysis, Journal of Computational Physics, vol. 211, No. 2, 2006, pp. 572--595.

"Arduino - ArduinoBoardUno." Arduino - ArduinoBoardUno. Arduino, n.d. Web. 26 Mar. 2017.

"Arduino - ArduinoBoardMega." Arduino - ArduinoBoardMega. Arduino, n.d. Web. 23 Mar. 2017.

Sine Wave Generator

A device which, when hooked up to a speaker or other device designed to oscillate at some frequency, will cause the device hooked up to it to oscillate in a sinusoidal manner (like a sine wave).

C++

Is an object oriented programming (OOP) language, developed by Bjarne Stroustrup, and is an extension of C language. This type of coding language does not, unfortunately, implement arrays and matrixes like the software MATLAB.

MATLAB

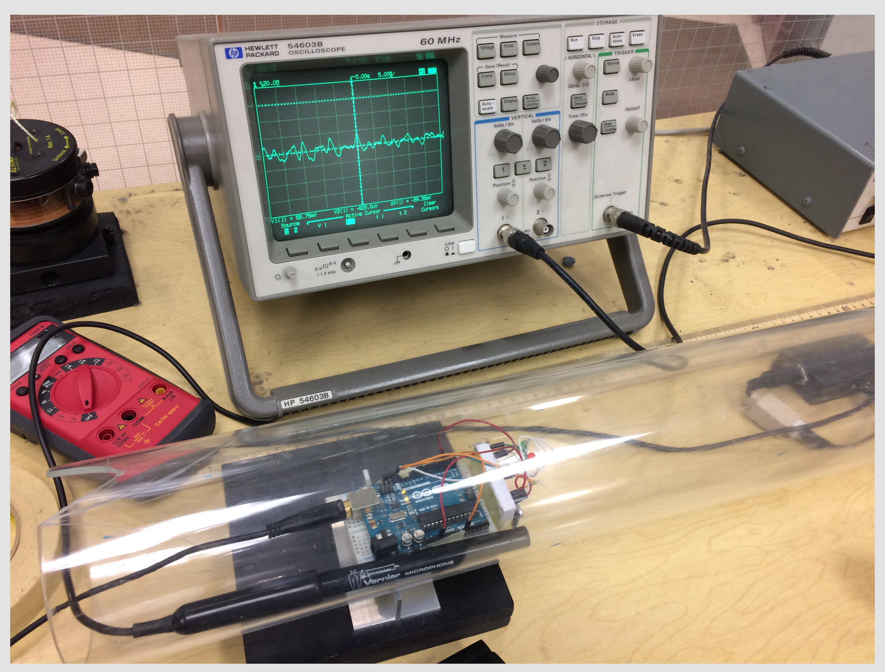
MATLAB stands for **MAT**rix **LAB**oratory and the software is designed around vectors and matrices. This makes the software useful for linear algebra, while also a great tool for solving algebraic and differential equations and for numerical integration.

Arduino/Arduino Uno

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. (Essentially it is a programmable, open-source computing platform which can be used for a number of things.) The company Arduino is an open-source computer hardware and software company which designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control objects in the physical world.

Arduino Mega

The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.



Special thanks to Dr. Gilbert for being an amazing mentor and being very helpful, supportive, teaching, and patient during the entirety of the project.

Code Translation, Board Configuration, and Testing:

In order to prove that Dr. Gilbert's FFT code is able to run on a low memory processor such as the Arduino, we first needed to be able to put the MATLAB code onto the Arduino, which uses the C++ coding language. In order to do this, we first had to translate Dr. Gilbert's base FFT code into C++ code from MATLAB code, but one of the main differences between MATLAB and C++ style coding is that MATLAB code is designed to work with matrices, and C++ is not naturally able to do this. In order to fix this problem, we were required to add a few code libraries and user created functions onto the Arduino which allowed it to essentially work with data structures which mirrored that of MATLAB matrices. We were also required to come up with functions which allowed the Arduino to handle complex numbers as the C++ coding language is not, in its base state, able to work with imaginary/complex numbers such as $\sqrt{(-1)}$. After this was done, and even a little before this was done, an idea was created which shifted the endpoint of our research quite a bit. This idea was conceived out of the fact that Arduinos are able to take data samples from the surrounding environment provided they have the necessary sensors to do so. Thus, we came up with the idea of making the Arduino use a electret microphone amplifier to take data samples from a speaker connected to a sin-wave-generator instead of feeding it data using code. This turned out to be an even better idea once we found out that the code required to make the Arduino take data samples from the environment used a lot less memory than the code required to artificially create the wave data to run the FFT on. It turned out that the only thing we had to do to get this idea off the ground was to buy and install the necessary electret microphone amplifier, put some code in to read the amplitude data from the sensor, and another function to randomly generate times to sample data from the pressure sensor (this is a critical component of Dr. Gilbert's FFT code). The sampling code was also designed to calibrate the sensor to the incoming signal int order to minimize the error in of the sampling. The code put onto the Arduino would then be tested in Randall Physics Laboratory at the University of Michigan using a sine-wave generator, an oscilloscope, and a speaker to generate waves to sample data from with the Arduino.

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 FFT 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 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 projected 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]