

# BUILDING A LOCK-IN AMPLIFIER FOR A NETWORK SENSOR SYSTEM

Catherine Reynoso (Computer Science), Hampton University

Advisors: Dr. Jay Zemel and Carlos Lopez

## ABSTRACT

Our project is part of an effort to network an array of sensors to monitor the flow distribution in a duct. It involves the fusion of several powerful software-based information extraction instruments (nodes) with a set of pyroelectric sensors. The pyroelectric anemometer sensors function by responding to local changes in the flow of gas. The program, which will be later installed into the microcontroller to carry out the computations for the system is referred to as the lock-in amplifier. Our project specifically focuses on the development of the lock-in amplifier for one of these independent nodes.

## 1. INTRODUCTION

In signal processing the desired signal is usually associated with noise. The goal of our computerized experiment is to develop a program (lock-in amplifier) that will extract the signal by averaging it for a sufficient length of time so that the random noise is greatly reduced. In the long run, this lock-in amplifier will be used in auto-correlation computations to obtain the amplitude of an analog voltage signal generated by the pyroelectric anemometer sensor in response to the gas flow.

The research reported in this document is merely a portion of a larger project's beginning stages to create a network of highly intelligent nodes that will individually involve the integration of a lock-in amplifier with a microcontroller. This integration process is facilitated by first running the application in software and debugging it as far as possible -- a better method than trying to figure out the malfunction of the application after wiring the hardware (PICMicro) with the code. Therefore, the scope of our research pertained to the experimentation of the lock-in amplifier programmed in C that we developed for the PIC 16C73 microcontroller.

C programming language was selected for this project mainly due to the time constraint of 10 weeks and fast-approaching project deadlines. I was already familiar with the basic programming concepts of OOP, so C was the easiest language to learn. C By Example,<sup>1</sup> my main source of reference is highly recommended. It covers the features of the C language in a clear and logical manner through the use of easy-to-follow examples.

The program will run on the PIC (Peripheral Interface Controller) 16C73, which

belongs to the PIC16/17 microcontroller (MCU) family. The devices within this series are low-cost, high-performance, 8-bit microcontrollers<sup>2</sup>. We choose the PIC 16C73 for several reasons. First, it is a Reduced Instruction Set Computer (RISC), which means that it is programmable with only a few instructions. Second, it has low power requirements and a rich peripheral set (including an analog to digital converter). Moreover, the PIC series 16/17 MCUs have a distinctive processor architecture. Unlike other microcontrollers, they use a "Harvard Architecture," in which the program and data are accessed from separate memories using separate buses. Separating program and data buses is an advantage because it improves bandwidth and serves as a protection against failure.

In many ways, the PIC 16/17 MCU family has one of the best-optimized processors for small assembly language application codes that must be very fast<sup>3</sup>. Currently it is one of the most complete and full-featured microcontroller series available on the market. "I have been working with the PICMicro (it is also commonly known as the "PIC") for over seven years now and I am continually amazed at its capabilities and usefulness in simple applications," says Myke Predko, author of various microcontroller books<sup>4</sup>.

The success of the PICMicro also comes from Microchip's excellent no-cost software development tools, "MPLAB." MPLAB is one of the best "Integrated Development Environments" available for any microcontroller<sup>3</sup>. It is a suite of tools that gives PICMicro users the flexibility to edit, compile, and debug from a single user interface. The MPLAB system supports in-circuit emulators, device programmers, and a C-compiler in an easy-to-use graphical environment. This tool is relatively fast, and the only cost is the time to download it.

## 2. PROJECT GOAL

Our goal was to develop, test, and evaluate the effectiveness of a distributed noise cancellation algorithm, which will eventually be used to program a PIC microcontroller. In particular, the project objectives were to:

- Study and gain an understanding of my advisors' ongoing project - to develop a network of intelligent nodes that measure the flow of gas.
- Concentrate our research on the development and implementation of a distributed noise cancellation algorithm written in C.
- Experiment using different amplitudes of the noise signal.
- Analyze the algorithm based on its noise reduction capabilities and robustness.
- Optimize the algorithm for performance and complexity.

## 3. BACKGROUND INFORMATION

I had to familiarize myself with several background concepts before starting the project.

### 3.1 Visual Basic

Visual Basic is a programming language used to create graphic user interfaces (GUI) with features that are familiar to Window users<sup>5</sup>. At the completion of the ongoing project, the overall information extracted from the system will be displayed on the PC through the use of a GUI developed in Visual Basic.

### 3.2 Fundamentals of Object Orientated Programming (OPP)

OPP is based on the concept that systems should be built from a collection of reusable components called objects. C, C++, FORTRAN, Ada, and many other programming languages were developed with OOP as their foundation<sup>6</sup>.

### 3.3 C Programming Language

C is one of the most popular programming languages in today's industry. It's effective, portable, and shares many similar features with other high-level programming languages 1.

### 3.4 Review of Math Essentials

Concepts of focus included a review of trigonometry and the fundamental theorem of the integral calculus.

### 3.5 MPLAB

MPLAB is an "Integrated Development Environment" that can be downloaded from the Internet at no cost. It is composed of a collection of tools that construct and debug applications in a project<sup>4</sup>. Once the lock-in amplifier is finalized, MPLAB will be use to program the microcontroller with the lock-in amplifier.

### 3.6 Digital Signal Processing (DSP)

DSP is the transformation of a digital signal to (1) estimate characteristic parameters of a signal or (2) transform a signal into a more desirable form<sup>7</sup>. The objective of our project deals with the latter - the cancellation of noise interference to extract only the desired digital signal.

### 3.7 Microsoft Excel

We used Excel Spreadsheets to model the signals. In this manner, we were able to generate the array index values of the sine, cosine, and noise signals declared at the beginning of the algorithm.

#### 4. EXPERIMENT PHASE AND RESULTS

Before typing the program's code, we developed a block diagram to clear up any confusion about its design. The block diagram made it easier to understand and keep track of the signal's path during the execution of the program.

Once we knew in which direction we wanted to go, we began by generating the values that would be used to model the signals. These values could be easily computed on a calculator; however, for a sample of 384 repeated several times this would take too long. Initially, we used Maple and Matlab, both mathematical-based computer software programs; however, neither one was successful. Both systems made it complicated to generate random numbers whose elements are uniformly distributed in the interval (-Amplitude, Amplitude). After about two weeks, we concluded that Excel would be more user-friendly in achieving this purpose.

Excel easily generated the array index values of the sine, cosine, and noise signals declared at the beginning of the C program. It also made it possible for us to make calculations that would forecast the output of the program. Knowing the correct calculations beforehand was extremely helpful because it allowed us to verify the validity of the program. As a result whenever it was running incorrectly we could trace the problem.

#### 5. PROGRAM COMPUTATION STEPS

After being generated in Excel, the Sin, Cosine, and Noise Signal arrays indexes were filled with their appropriate values (see Appendix for program code). Then several operations occur within a for loop. First, each value in Noise\_Signal array is multiplied by each value in Sine\_Signal array with its corresponding index values. The product of that operation is constantly incremented by each consecutive product, creating a summation of all of the Sine\_Signal values multiplied by the Noise\_Signal values. The same procedure is followed for the Noise\_Signal array and the Cosine\_Signal array. This for loop processes 384 times according to this set number of the discrete points throughout the original signal.

#### 6. CONCLUSIONS

Time constraints have not permitted the goal of our project to be accomplished. The program is not yet completely debugged; however, I will proceed my efforts through my independent research course during the fall 1999 semester at Hampton University. I will continue to have the support of Dr. Jay Zemel and Mr. Carlos Lopez through email.

In order to extract the amplitude of the original signal before being corrupted by noise, a few computations are still needed in our program. First, the program must square the value of Final Result1 and do the same for Final Result2, becoming Sq\_Final\_Result1 and Sq\_Final\_Result2, respectively. Sq\_Final\_Result1 and Sq\_Final\_Result2 need to then be

added together, and the square root of their sum will equal the amplitude of the original signal without noise.

This same algorithm must then be run five times, each using a different experimental amplitude size multiplied by a generated random number and added to the sine curve, producing the Noise\_Signal. The different amplitudes used will be .01, .1, 1, 10, and 100.

## 7. INCORPORATING THE LOCK-IN AMPLIFIER WITH THE NETWORK OF SENSORS

Once a network of intelligent nodes has been completed, our goal is for the final apparatus of each independent node to function in the following manner:

The pyroelectric anemometer (PA) generates an analog voltage signal as a result of the gas flow. A Conditioning Circuit adjusts the Phase-Gain of the output from one arm of the PA to balance the signal from one electrode to equal the other at zero flow. Buffers connect the Conditioning Circuit to the Differential Instrumentation Amplifier and an Anti-Alias Filter, which gets rid of any interrupting frequencies common to both signals. Then the signal travels to the PIC 16C73 Microcontroller. The Analog-to-Digital Converter located inside the PIC 16C73 Microcontroller converts the analog input signal to an 8-bit digital number. Then the adaptable C program developed for the PIC 16C73 Microcontroller processor core functions as an instrument to run the auto-correlation computations that will obtain the amplitude of the differential PA signal. Hence, using the powerful communication capabilities of these devices, the overall information extracted from the system will be displayed finally on the PC. See PowerPoint Slide #4, "Signal Chain Diagram of Sensor System Independent Nodes," for the graphical representation of the process taken by each independent node within the network.

## 8. ACKNOWLEDGMENTS

I must foremost give recognition to Dr. Jay Zemel and Mr. Carlos Lopez who were my mentors throughout this entire project. It was their idea of creating a network of intelligent nodes that gave birth to this computer-generated experiment. Thank you for giving me the courage to use my mind as well and as honestly as I could and thank you for encouraging me to ask questions whenever necessary.

Next, of course, I must acknowledge the National Science Foundation - sponsors of the SUNFEST-REU program for providing me the opportunity to be a part of this program. This has truly turned out to be a memorable research experience in Philadelphia. I have gained a lot of knowledge regarding research as well as insight into what direction to aim the future of my education.

## 9. REFERENCES

1. Kalicharan, Noel. *C by Example*. New York: Cambridge University Press, 1994.
2. Microchip PIC 16C7X Data Sheet. Microchip Technology Inc., 1995.
3. <http://www.rentron.com/Myke3.htm>, July 1999.
4. <http://www.myke.com/PICMicro>, July 1999.
5. *Visual Basic4 Expert Solutions*. Indianapolis: Que Corporation, 1995.
6. Blaschek, Gunther. *Object Oriented Programming with Prototypes*. New York: Springer-Verlag, 1994.
7. Johnson, Johnny R. *Introduction to Digital Signal Processing*. Prentice-Hall Inc., 1989.

## 10. APPENDIX

### Lock-in Amplifier Algorithm

(Written in C Programming Language)

```
#include <stdio.h>

char Noise_Signal [ ] = {384 values generated in excel};
char Sin_Signal [ ] ={" "};
char Cos_Signal [ ] ={" "};
int Maxindex, Noise_Index, Sin_Index, Cos_Index,
double Final_Result1, Final_Result2, Output1, Output2;

main()
{
    Maxindex = 383;
    Final_Result1 = 0;
    Final_Result2 = 0;
    Sin_Index=0;
    Cos_Index=0;
    for (Noise_Index=0; Noise_Index <= Maxindex; Noise_Index++)
    {
        Output1 = Noise_Signal [Noise_Index] * Sin_Signal [Sin_Index];
        Output2 = Noise_Signal [Noise_Index] * Cos_Signal [Cos_Index];
        //each value in Noise_Signal array is multiplied by
        //each value in Sin_Signal & Cos_Signal array with corresponding index values

        Final_Result1 = Final_Result1 + Output1;
        Final_Result2 = Final_Result2 + Output2;
        //Final_Result1 & Final_Result2 track the summation

        Sin_Index++;
        Cos_Index++;
        //Sin_Index and Cos_Index are incremented each time the loop executes
    }

    printf("RESULTS\n");
    printf("*****\n");
    printf("Sin Value: %f \n", Final_Result1);
    printf("Cos Value: %f \n", Final_Result2);
}
```