4-30-2015

Distributed Wireless PZT Fiber Composite Sensor System Development and Demonstration: Wireless Data Acquisition Development

Ethan Paul Hopping

Distributed Wireless PZT Fiber Composite Sensor System Development and Demonstration: Wireless Data Acquisition Development

by

Ethan Paul Hopping

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma or Certificate

to

The Honors College

of

The University of Alabama in Huntsville

30 April, 2015

Honors Capstone Director: Dr. Gang Wang

_____

Student Date

_____

Director Date

_____

Department Chair Date

_____

Honors College Dean Date

Table of Contents

Abstract

Composite materials are finding broader application in aerospace structures due to benefits such as superior stiffness and light weight relative to conventional materials. These new materials have yet to endure the test of time through an entire product life cycle, as compared to well understood conventional materials. Additionally, policy decisions to extend the operational lifespans of current aerospace vehicles well beyond initial design requirements are becoming increasingly common. Continuous health monitoring over the lifespan of the structures is needed in order to fully optimize these material options for the next generation of vehicles.

In this honors project, a wireless PZT fiber composite (WPFC) sensor concept was developed to provide distributed sensing capability to measure structurally deformed shapes. The data acquisition portion of this project demonstrated prototyping of the data acquisition system and associated software components with a development board. An embedded system using an Atmel Microcontroller was used to develop a compact data acquisition system capable of greater than 1 kHz sampling rate and measurement from up to 8 piezoelectric sensors. Performance of the DAQ was analyzed through comparison to measurements made using a Tektronix oscilloscope. Due to time constraints, the wireless portion of the project was not implemented, but the author has provided preliminary suggestions for wireless solutions to consider as future development work.

Acknowledgements

This honors project was developed through the Adaptive Structures Laboratory in the Mechanical and Aerospace Engineering Department at the University of Alabama in Huntsville. As an honors student, I had the privilege to work closely with the Adaptive Structures Team, including Felix Ewere, Kevin Gilbert, Chris Hill, and Dr. Gang Wang. This represented a great learning opportunity so see some of the concepts taught in academic curricula, such as the advanced composites course, applied in a research setting.

Special thanks goes to Felix Ewere for being available in a project support and advising role to answer the questions about lab equipment, sensor fabrication, and testing. Felix also supported the project by performing FEA modelling in Console to determine the natural frequencies of the test structure.

Most importantly, thanks to Dr. Wang for his leadership and guidance over the course of the project.

Project Overview

Structural health monitoring is a topic of significant interest in the aerospace industry. A need for lighter vehicles to maximize fuel efficiency and endurance has led to increasing reliance on composite structures in both aviation and space sectors of the industry. Health monitoring for these composite structures differs significantly from procedures for life cycle monitoring of aluminum structures. In particular, many composites do not undergo strain hardening or necking before failure, as is seen in metal structures. Consequently, strain before failure is more difficult to identify in composite structures, and specialized procedures are required to make sure composite aerospace structures are within structural operating limits. Preventative maintenance is frequently required to ensure structures operate within acceptable factors of safety.

As the cost of embedded systems continues to decrease, new solutions have become available that make continuous structural health monitoring of composite structures possible. Adoption of continuous structural heath monitoring would enable a transition from preventative maintenance to a maintenance schedule that is informed by data. That is, the decision to replace a component can be based on measurements of the strain and cyclic loading of the component over its lifetime, instead of a maintenance schedule that is based on average part lifespan and a factor of safety. The purpose of this honors project is to provide a demonstration of a cost effective, wireless structural health monitoring system, using piezoelectric sensors and a data acquisition system (DAQ) developed from low-cost microcontrollers. This text focuses on the design and validation of the DAQ.

The development of the demonstration project was divided into two components: sensor fabrication and DAQ development. Markus Murdy, another honors student, served as lead on sensor fabrication and was responsible for sensor integration and mounting of the sensors on test hardware.

Ethan Hopping was responsible for development of the DAQ and validation of instrumentation performance. A graphical overview of the demonstration project concept is provided in Figure 1.

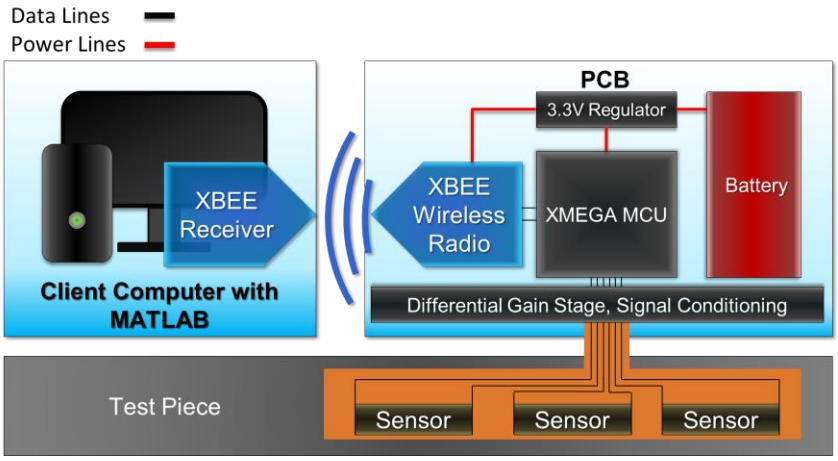

Figure 1: Sensor and data acquisition system overview

The square grey beam in Figure 1 represents the test article to which piezoelectric sensors were mounted. When the test article is deflected, the strain induces a voltage in the piezoelectric sensors. This voltage is measurable, and can be correlated with the magnitude of the deflection of the test article. A microcontroller equipped with an analogue to digital converter (ADC) is used to collect voltage measurements from the sensors. In structural health monitoring applications, this microcontroller and the associated power system would be mounted to the structural component being monitored. The small size of modern embedded systems means that few accommodations would be needed from an integration perspective to add the microcontroller to an existing structure. A radio would be used to transmit voltage measurements to a host computer for analysis. Wireless communications would allow the DAQ to be mounted in difficult-to-access locations within an aircraft, such as the wing box or a helicopter rotor blade. The use of wireless communications would also open up the possibility of using one host machine to collect data from multiple DAQ systems.

Sensor Fabrication and Testing

Sensor fabrication is discussed briefly as it relates to the DAQ. For a more extensive discussion of the sensor fabrication process, see Markus Murdy's text on sensor design and fabrication.

Sensors were manufactured from zirconium titanate (PZT) fibers[1]. This piezoelectric material produces a measurable voltage when strained. The PZT fiber was mounted on an aluminum shim for structural rigidity. A PVC film was used to electrically isolate the PZT from the aluminum shim, and the PZT-wire assembly was cast in a polyester shell. A PZT sensor before casting is shown in Figure 2. Two PZT sensors were fabricated using this procedure and mounted to an aluminum test beam.

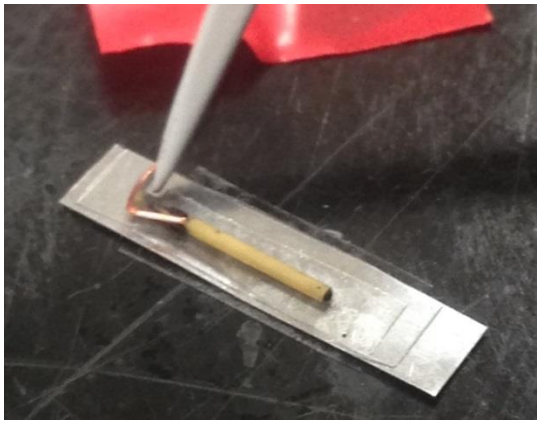

Figure 2: Placing Lead Wires on PZT

Sensor performance was validated independently of the DAQ using a Tecktronix oscilloscope. A compact table shaker was used to drive the aluminum test piece at a specified frequency and amplitude. Sample results from this testing are shown in Figure 3.

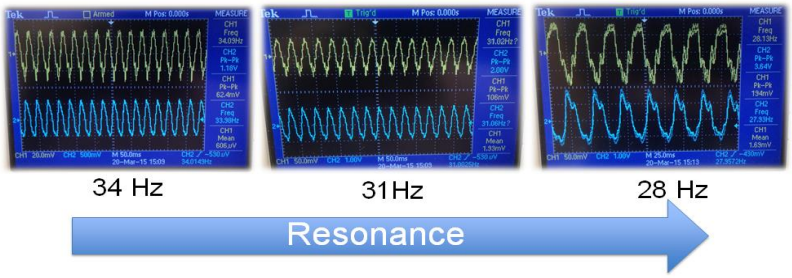

Figure 3: PZT sensor response observed on oscilloscope

The preliminary testing was important for directing DAQ development by establishing an acceptable sampling frequency. For the test piece considered, resonance was found to occur at

[1] http://www.smart-material.com/PZTFiber-product-main.html

approximately 28 Hz. With a capability to sample at over 1 kHz, the microcontroller selected was able to easily measure resonance frequency without aliasing.

Oscilloscope testing also revealed that voltage amplitude varied strongly with sensor location on the test piece. The sensor placed closest to the root of the beam experience voltage amplitudes much higher than those measured by the sensor at the tip of the beam. This is because sensor deflection is higher at the root of the beam than the tip of the beam under the experimental test setup. An important consideration for the DAQ was ensuring the system could accurately sample the voltage signal over small and large voltage ranges.

Data Acquisition System

Due to time constraints, the DAQ was not developed past the prototyping phase to dedicated circuit board development. However, many of the software tools used for sampling and recording data were well developed and tested. The rest of the text is dedicated to a discussion of DAQ prototyping hardware, and some of the software features implemented for sampling voltage from the PZT sensors.

Prototyping Hardware

An Atmel Xmega-A1 Xplained development board was used for software development, testing, and data acquisition[2]. The development board is designed to demonstrate the capabilities of the Xmega 128A1 microcontroller. Some of the technical highlights of this microcontroller include 8 12-bit ADC channels when used as differential channels, 8 timer counter/waveform generation channels, and 78 digital pins. An image of the Xmega development board, and a breadboard for ADC calibration and testing, is provided in Figure 4.

[2] http://www.atmel.com/tools/xmega-a1xplained.aspx

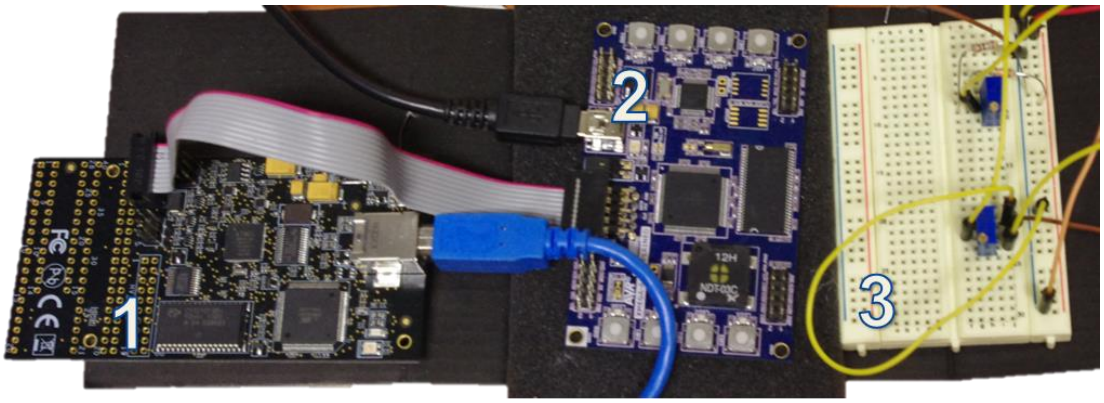

Figure 4: Electrical prototyping setup

In Figure 4, the Xmega is labelled number 2, and the breadboard is number 3. The circuit board labelled number 1 is an AVR Dragon, which is used for programming the Xmega development board. Sensors were attached with a ribbon cable to the ADC pins ported out to the male headers on the Xmega development board.

A preliminary electrical block diagram for the DAQ is provided in Figure 5. Voltage measurements of the PZT sensor response is made with the Xmega's 8 differential ADC channels. The maximum voltage range on these ADC channels is equivalent to VCC/1.6, which comes out to approximately 2 Volts. While the voltage amplitude of the majority of the PZT measurements was below 2 volts, it is easy to imagine circumstances where the PZT voltage range is greater than +/-2 volts. Therefore, the final DAQ was to include variable potentiometers to allow voltages above 2 volts to be stepped down to a range that could be measured by the microcontroller. These variable potentiometers were to be controlled by the microcontroller using SPI.

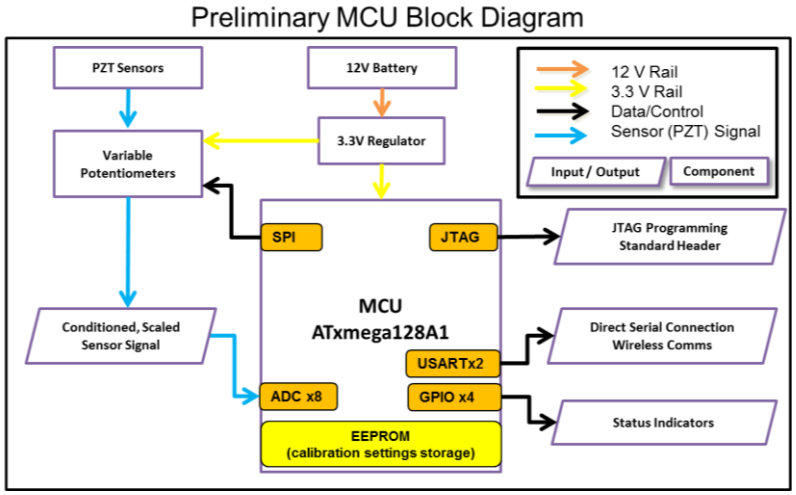

Figure 5: Electrical block diagram

Voltage measurements were transmitted from the microcontroller to the host system USART. During testing, the USART port was directly connected to the host computer over a USB connection. However, this direct connection could easily be replaced with any radio system capable of interfacing with other hardware over USART.

Software Design

The DAQ was designed to communicate and interface with a GUI developed for the host computer in Matlab. The host machine could send start-stop commands to the DAQ to initiate data recording, and results from the DAQ were plotted in the Matlab GUI in real-time. Data could be saved from the Matlab interface in an Excel file for post-processing and analysis.

Data Acquisition System

The software for the DAQ was developed in Atmel Studio 6 using the C programming language. Atmel Studio 6 is an integrated development environment for programming and debugging Atmel, ARM, and AVR microcontrollers. A copy of the DAQ development code is provided in the appendix.

The software is structured around two system states: standby and data acquisition. Data acquisition is initiated by a start command sent from the software user interface. The start command enables the ADC and the controlling timer counter module. Samples are taken sequentially at a frequency of 1 kHz from four channels. Data is transmitted over the USART port to the host machine. A unique feature as compared to previous software iterations is that data is sent over the USART port as 16 bit integers as opposed to using the ASCII format. This reduces packet size and allows for faster data transmission and parsing at lower baud rates. This communication protocol change was made with consideration of future wireless applications, which are frequently bandwidth limited.

A unique auto-ranging feature was also implemented in software to address the large variance in peak to peak voltage for sensors placed on the root and the tip of the beam. The ADC on the 128A1 includes an integrated gain stage, which allows differential voltage readings to be scaled by anywhere

from 2x to 64x. The DAQ software was developed to default to highest gain setting when sampling. If the ADC measurements were saturated, gain was gradually reduced until the signal was within a voltage range that was not saturating the ADC. This allows the DAQ to make voltage measurements adequate sensitivity for PZT fibers at both the root and the tip of the beam. The solution is designed to be transparent to the user; however, future work is needed to account for the change in gain from within the Matlab software interface.

Software Interface

A graphical user interface (GUI) was developed in Matlab for data collection and analysis. In the most recent version, this software interface was capable of continuously plotting readings from four separate ADC channels in real time. A screenshot of the GUI with a single voltage trace is provided in Figure 6.

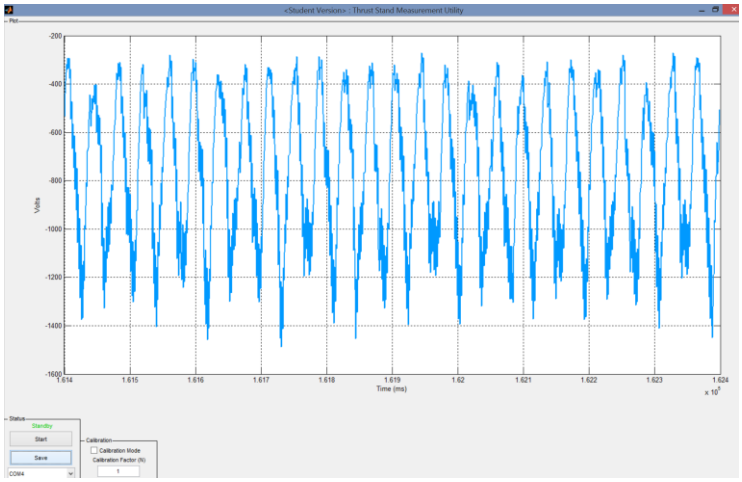

Figure 6: Collecting data in the Matlab interface from a single channel

Controls for the DAQ are provided in the bottom left hand corner of the interface. A drop-down menu is used to select from available serial ports. A start-stop button is used to initiate and terminate data recording, and a save button is provided for exporting data as an excel file. A calibration box is provided for inputting a scaling parameter to convert the raw ADC readings to accurate voltage readings.

Due to the size of the associated Matlab files, the code for the GUI is not included in this report, but can be made available upon request from the author if needed.

Testing and Validation

Validation of DAQ performance was performed by comparing voltage and frequency measurements with measurements made with a Tektronix oscilloscope. Some validation of the performance of the DAQ was provided through this comparison; however, flaws in the sensor fabrication and other technical challenges prevented a high quality comparison between Oscilloscope data and DAQ measurements.

A comparison of DAQ and oscilloscope voltage measurements for the test piece is provided in Figure 7. These voltage readings were taken at the same driving frequency, approximately 25 Hz. However, the shaking amplitude was increased between the DAQ test and the oscilloscope test, resulting in higher voltage readings using the DAQ. It was very difficult to line up voltage amplitude between the oscilloscope and DAQ. It is suspected that this is due to electrical connection factors, such as noise in the connectors for the DAQ and poorly insulated wires between the sensor and the DAQ. However, Figure 7 does demonstrate the natural frequency measured using the oscilloscope and DAQ do match.

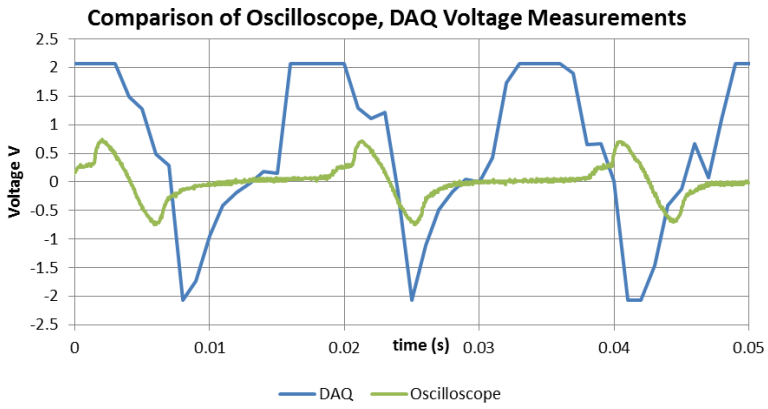

Figure 7: Comparison of oscilloscope and DAQ voltage measurements

The waveform generated by the PZT fibers in the comparison testing was unexpected. Instead of the traditional sinusoidal wave shape, the measured waveform was of a serpentine shape followed by a flat period. The author suspects that the sensors were damaged at some point in the testing, because the earliest tests demonstrated that the sensors were capable of generating sinusoidal waves. The unexpected waveform shape greatly reduced the effectiveness of Fourier analysis for determining the oscillation frequency, since 25 Hz was no longer the dominant frequency. A Fourier transform of waveform data

collected using the oscilloscope is provided in **Error! Reference source not found.**. The natural frequency of 25 Hz is no longer the dominant peak in the frequency domain due to the bizarre waveform. This result also provides an explanation for why the oscilloscope frequently could not identify 25 hertz as the dominant frequency using its onboard Fourier analysis capabilities, due to the presence of multiple peaks of similar amplitudes. Because of this unusual waveform, it was difficult to compare the Oscilloscope and DAQ measurements in later testing.

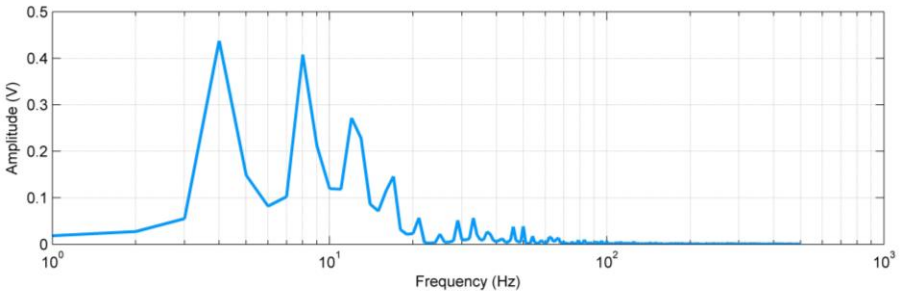

Figure 8: Oscilloscope waveform in the frequency domain

Future Work

There are many opportunities for future work with maturing the project concept and implementation. While software for the DAQ is fairly mature, significant efforts are needed in hardware development. This includes moving the electrical design from a development board to custom printed circuit board (PCB) and radio selection and testing.

A custom circuit board could be fabricated with a much more compact footprint than the development board because only components necessary for operation of the DAQ would be ported out from the microcontroller. A custom PCB would also enable integration of features such as a rechargeable battery into the design.

A key component of this project that was not addressed was radios. The author suggests that the Xbee wifi modules be selected to enable wireless communication. These radios would enable wireless transmission of strain measurements from the DAQ to computer host. The wifi modules are suggested because of the ability to operate at higher bandwidth than the ZigBee protocol and enable additional unique features such as internet data logging.

Conclusion

A development prototype of a DAQ for collecting stain-mode measurements using multiple PZT sensors was created. An embedded system using an Atmel Microcontroller was used to develop a compact data acquisition system capable of greater than 1 kHz sampling rate and measurement from up to 8 PZT sensors. Performance of the DAQ was analyzed through comparison of strain measurements made using the DAQ with measurements made using an oscilloscope. Prototype sensor defects and other technical challenges made it difficult to match waveforms from the DAQ and oscilloscope exactly. However, the waveforms are similar enough to demonstrate that the DAQ is collecting voltage data and operating nominally. While the software components of the project, including the Matlab interface and DAQ control code were fairly well developed, significant progress is needed in hardware development to move the current prototype to a printed circuit board and enable wireless data transmission.

Appendix

Data Acquisition System Program

```
int16_t ADC4;
int8_t shift=0;
volatile uint32_t t = 0;

void usart_init(void);
void usart_init(void){
                    PORTC.DIRSET |= 0b00001000;   //Set USARTC0 TX to output
                    PORTC.OUTSET |= 0b00000100;   //Set USARTC0 TX high
                    USARTC0.CTRLA |=(1<<5)|(1<<4);               //Enable RX Interrupts with medium priority
                    USARTC0.CTRLB |=(1<<4)|(1<<3)|(1<<2);       //Enable RX,TX,Double Transmission Speed
                    USARTC0_CTRLC = USART_CHSIZE_8BIT_gc;       //8bit character size, unsigned
                    USARTC0_BAUDCTRLA = 34;                                                 //115200 for prototyping
                    USARTC0_BAUDCTRLB = 0;
}

uint8_t ReadSignatureByte(uint16_t Address);
uint8_t ReadSignatureByte(uint16_t Address)
{
          NVM_CMD = NVM_CMD_READ_CALIB_ROW_gc;
          uint8_t Result;
          __asm__ ("lpm %0, Z\n" : "=r" (Result) : "z" (Address));
          NVM_CMD = NVM_CMD_NO_OPERATION_gc;
          return Result;
}

void adc_init(void);
void adc_init(){
          PORTA.DIR =0x00;
          ADCA.CTRLA |= 0b00000001;
          ADCA.CTRLB |= 0b00010000;
          ADCA.REFCTRL = ADC_REFSEL_VCC_gc |0x02;//ADC_REFSEL_INT1V_gc;//ADC_REFSEL_VCC_gc; //|0x02;
          ADCA.PRESCALER = ADC_PRESCALER_DIV32_gc;
          ADCA.EVCTRL |=0b11000101;
          //ADCA.CAL = adc_get_calibration_data(ADC_CAL_ADCA);
          ADCA.CALL = ReadSignatureByte(0x20) ; //ADC Calibration Byte 0
          ADCA.CALH = ReadSignatureByte(0x21) ; //ADC Calibration Byte 1
          //ADCA.CH0.INTCTRL =0b00000011;
          ADCA.CH0.CTRL = 0b00011011;
          ADCA.CH0.MUXCTRL = 0b00000000;
          //ADCA.CH1.INTCTRL =0b00000011;
          ADCA.CH1.CTRL = 0b00011011;
          ADCA.CH1.MUXCTRL = 0b00001001;
          //ADCA.CH2.INTCTRL =0b00000011;
          ADCA.CH2.CTRL = 0b00011011;
          ADCA.CH2.MUXCTRL = 0b00010010;
          ADCA.CH3.INTCTRL =0b00000011;
          ADCA.CH3.CTRL = 0b00011011;
          ADCA.CH3.MUXCTRL = 0b00011011;
}

void tcc_init(void);
void tcc_init(void){
                    //TCC0.CTRLA |=0b00000001;       //Prescalser 8
                    //TCC0.CTRLB |=0b11110000;
                    //TCC0.INTCTRLA |=0b00000011;  //High Level Interrupt
                    //TCC0.INTCTRLB |=0b00000011;
                    ////TCC0.CCA=31999;
                    //TCC0.PER=31999;
                    ////TCD0.CCB=249;
                    ////TCD0.CCC=249;
                    EVSYS.CH0MUX |=0b11000100;
                              TCC0.CTRLA=0b00000100;          //Prescalser 8
                              //TCD0.CTRLB=0b00010000;
                              //TCD0.INTCTRLA=0b00000011;  //High Level Interrupt
                              //TCC0.INTCTRLB=0b11000111;
                              //TCD0.CCA=249;
                              //TCD0.CCB=249;
                              //TCD0.CCC=249;
                              TCC0.PER=3999;


}
void USART_SendData(int16_t data);
void USART_SendData(int16_t data ){
          while ( !(USARTC0_STATUS & USART_DREIF_bm));
          USARTC0.DATA = data;
          while ( !(USARTC0_STATUS & USART_DREIF_bm));
          USARTC0.DATA = data>>8;

}

void USART_sendtime(uint32_t time);
void USART_sendtime(uint32_t time){
```

```
		while ( !(USARTC0_STATUS & USART_DREIF_bm));
		USARTC0.DATA = time;
		while ( !(USARTC0_STATUS & USART_DREIF_bm));
		USARTC0.DATA = time>>8;
		while ( !(USARTC0_STATUS & USART_DREIF_bm));
		USARTC0.DATA = time>>16;
		while ( !(USARTC0_STATUS & USART_DREIF_bm));
		USARTC0.DATA = time>>24;
}

void USART_Transmit( unsigned char data );
void USART_Transmit( unsigned char data ){
		while ( !(USARTC0_STATUS & USART_DREIF_bm));		// Wait for empty transmit buffer
		USARTC0.DATA = data;							// Put data in buffer
}

void USART_putstring(char* StringPtr);
void USART_putstring(char* StringPtr){
		while(*StringPtr != 0x00){				//Until null terminator is reached
			USART_Transmit(*StringPtr);			//Send individual character to transmit
			StringPtr++;						//Move to next character
		}
		USART_Transmit('\n');					//Send newline
}


void setUp32MhzInternalOsc(void);
void setUp32MhzInternalOsc()
{
		OSC_CTRL |= OSC_RC32MEN_bm; //Setup 32Mhz crystal

		while(!(OSC_STATUS & OSC_RC32MRDY_bm));

		CCP = CCP_IOREG_gc; //Trigger protection mechanism
		CLK_CTRL = CLK_SCLKSEL_RC32M_gc; //Enable internal  32Mhz crystal


}

ISR(USARTC0_RXC_vect){
		state = USARTC0_DATA;
		if(state==0){
			t=0;
		}
}

ISR(ADCA_CH3_vect){
		if(state==1){
		//char out [100];
//		char out2 [100];
		ADC1=ADCA.CH0RES;
		ADC2=ADCA.CH1RES;
		ADC3=ADCA.CH2RES;
		ADC4=ADCA.CH3RES;
		//sprintf(out,"%ld,%d,%d,%d,%d",t,ADC1,ADC2,ADC3,ADC4);
		//itoa(ADC1,out,10);
		//ltoa(t,out2,10);
		//strcat(out2,",");
		//strcat(out2,out);
		USART_sendtime(t);
		USART_SendData(ADC1);
		USART_SendData(ADC2);
		USART_SendData(ADC3);
		USART_SendData(ADC4);
//		USART_Transmit('\n');
		t++;
		}
}

int main (void)
{

		//sysclk_init();
		//board_init();
		cli();
		setUp32MhzInternalOsc();
		usart_init();
		adc_init();
		tcc_init();
		sei();
		PMIC_CTRL |=(1<<2);
		irq_initialize_vectors();
		cpu_irq_enable();
		PORTE.DIR=0xFF;
		PORTE.OUT=0xFF;
		while(1){
			if(state==1){

				if((ADC1==2047 || ADC1 == -2048)&& ADCA.CH0.CTRL!=0b00000011)
				shift=shift+1;
```

```
                    if(shift == 10){
                              cli();
                              ADCA.CTRLA = 0b00000000;
                              int8_t reg = ADCA.CH0.CTRL;
                              if(reg == 0b00011011){ ADCA.CH0.CTRL = 0b00010111; }//PORTE.OUT=0b01111111;}
                              if(reg == 0b00010111){ ADCA.CH0.CTRL = 0b00010011; }//PORTE.OUT=0b00111111;}
                              if(reg == 0b00010011){ ADCA.CH0.CTRL = 0b00001111; }//PORTE.OUT=0b00011111;}
                              if(reg == 0b00001111){ ADCA.CH0.CTRL = 0b00001011; }//PORTE.OUT=0b00001111;}
                              if(reg == 0b00001011){ ADCA.CH0.CTRL = 0b00000111; }//PORTE.OUT=0b00000111;}
                              if(reg == 0b00000111){ ADCA.CH0.CTRL = 0b00000011; }//PORTE.OUT=0b00000011;}
                              ADCA.CTRLA = 0b00000001;
                              sei();
                    }
                    if((ADC2==2047 || ADC2 == -2048)&& ADCA.CH1.CTRL!=0b00000011)
                    shift=shift+1;
                    if(shift == 10){
                              cli();
                              ADCA.CTRLA = 0b00000000;
                              int8_t reg = ADCA.CH1.CTRL;
                              if(reg == 0b00011011){ ADCA.CH1.CTRL = 0b00010111;}// PORTE.OUT=0b01111111;}
                              if(reg == 0b00010111){ ADCA.CH1.CTRL = 0b00010011;}// PORTE.OUT=0b00111111;}
                              if(reg == 0b00010011){ ADCA.CH1.CTRL = 0b00001111;}// PORTE.OUT=0b00011111;}
                              if(reg == 0b00001111){ ADCA.CH1.CTRL = 0b00001011;}// PORTE.OUT=0b00001111;}
                              if(reg == 0b00001011){ ADCA.CH1.CTRL = 0b00000111;}// PORTE.OUT=0b00000111;}
                              if(reg == 0b00000111){ ADCA.CH1.CTRL = 0b00000011;}// PORTE.OUT=0b00000011;}
                              ADCA.CTRLA = 0b00000001;
                              sei();
                    }
                    if((ADC3==2047 || ADC3 == -2048)&& ADCA.CH2.CTRL!=0b00000011)
                    shift=shift+1;
                    if(shift == 10){
                              cli();
                              ADCA.CTRLA = 0b00000000;
                              int8_t reg = ADCA.CH2.CTRL;
                              if(reg == 0b00011011){ ADCA.CH2.CTRL = 0b00010111;}// PORTE.OUT=0b01111111;}
                              if(reg == 0b00010111){ ADCA.CH2.CTRL = 0b00010011;}// PORTE.OUT=0b00111111;}
                              if(reg == 0b00010011){ ADCA.CH2.CTRL = 0b00001111;}// PORTE.OUT=0b00011111;}
                              if(reg == 0b00001111){ ADCA.CH2.CTRL = 0b00001011;}// PORTE.OUT=0b00001111;}
                              if(reg == 0b00001011){ ADCA.CH2.CTRL = 0b00000111;}// PORTE.OUT=0b00000111;}
                              if(reg == 0b00000111){ ADCA.CH2.CTRL = 0b00000011;}// PORTE.OUT=0b00000011;}
                              ADCA.CTRLA = 0b00000001;
                              sei();
                    }
                              if((ADC4==2047 || ADC4 == -2048)&& ADCA.CH3.CTRL!=0b00000011)
                              shift=shift+1;
                              if(shift == 10){
                                        cli();
                                        ADCA.CTRLA = 0b00000000;
                                        int8_t reg = ADCA.CH3.CTRL;
                                        if(reg == 0b00011011){ ADCA.CH3.CTRL = 0b00010111;}// PORTE.OUT=0b01111111;}
                                        if(reg == 0b00010111){ ADCA.CH3.CTRL = 0b00010011;}// PORTE.OUT=0b00111111;}
                                        if(reg == 0b00010011){ ADCA.CH3.CTRL = 0b00001111;}// PORTE.OUT=0b00011111;}
                                        if(reg == 0b00001111){ ADCA.CH3.CTRL = 0b00001011;}// PORTE.OUT=0b00001111;}
                                        if(reg == 0b00001011){ ADCA.CH3.CTRL = 0b00000111;}// PORTE.OUT=0b00000111;}
                                        if(reg == 0b00000111){ ADCA.CH3.CTRL = 0b00000011;}// PORTE.OUT=0b00000011;}
                                        ADCA.CTRLA = 0b00000001;
                                        sei();
                              }
}
if(state ==0){
          PORTE.OUT=0xFF;
          if(ADCA.CH0.CTRL!=0b00011011){
                    cli();
                    ADCA.CTRLA = 0b00000000;
                    ADCA.CH0.CTRL = 0b00011011;
                    ADCA.CTRLA = 0b00000001;
                    sei();
          }
          if(ADCA.CH1.CTRL!=0b00011011){
                    cli();
                    ADCA.CTRLA = 0b00000000;
                    ADCA.CH1.CTRL = 0b00011011;
                    ADCA.CTRLA = 0b00000001;
                    sei();
          }
          if(ADCA.CH2.CTRL!=0b00011011){
                    cli();
                    ADCA.CTRLA = 0b00000000;
                    ADCA.CH2.CTRL = 0b00011011;
                    ADCA.CTRLA = 0b00000001;
                    sei();
}
          if(ADCA.CH3.CTRL!=0b00011011){
                    cli();
                    ADCA.CTRLA = 0b00000000;
                    ADCA.CH3.CTRL = 0b00011011;
                    ADCA.CTRLA = 0b00000001;
                    sei();
}}}}
```