Do Microcontrollers Belong in a Sophomore Course on Statics and Dynamics?

Richard E. Haskell, Michael Latcha, and Osamah Rawashdeh
School of Engineering and Computer Science
Oakland University
Rochester, Michigan 48309
Email: haskell@oakland.edu, latcha@oakland.edu, rawashd2@oakland.edu

Abstract

A new core course, *Design and Analysis of Electromechanical Systems*, was taught for the first time in the fall of 2006 at Oakland University. This course is taken by all mechanical, electrical, computer, and industrial and systems engineering majors. During the first five weeks of the course the students learn to program a microcontroller board in C using CodeWarrior. This is made possible by providing a stationery project that contains over 80 assembly language routines that can be called as C functions to perform I/O functions including 7-segment displays, keypad, liquid crystal display, pushbutton switches, real-time interrupts, motor speed control, servo position control, interrupt-driven pulse-width measurement, and serial communication. During the remainder of the course the students learn statics and dynamics and work in groups to design and implement an electromechanical system that uses the microcontroller board programmed in C. This approach has expanded the possibilities in a statics and dynamics course by replacing meter sticks and stop watches with microcontrollers, servos and motors.

1. Introduction

Statics and dynamics used to be a basic engineering course (or courses) taken by all engineering students. A review of electrical and computer engineering programs in Michigan shows that only a small minority of these programs still require a course in mechanics beyond the basic physics course. While all mechanical engineering programs require at least one electrical engineering course that usually covers basic circuit theory and other topics of interest to mechanical engineering students, none of these courses taken by ME majors include any material on programming microcontrollers. But microcontrollers offer unlimited possibilities of automating experiments in a mechanics lab and allowing student to easily measure such properties as the coefficients of friction and the coefficient of restitution.

The School of Engineering and Computer Science at Oakland University has, in its 43-year history, always had a strong engineering core curriculum that is taken by all engineering students. In the fall of 2005 we implemented a new, completely overhauled, core curriculum that consists of five 4-credit courses and one 1-credit course. The last course in this core curriculum is a 4-credit course called *Design and Analysis of Electromechanical Systems*, which contains a 3-hour per week laboratory that is an integral part of the course. A little over half of the course is devoted to traditional topics in statics and dynamics. We sought ways to illustrate mechanics principles in a modern setting that would also provide the students with useful computer skills. Mechanical engineering majors take a new follow-on course in engineering mechanics for which a series of new lab experiments have been developed.[1-3] These experiments lend themselves to computer control, for which the students will now be prepared.

In the new sophomore core course we now spend the first third of the course teaching the students how to program a modern microcontroller in C using CodeWarrior. We had been using the miniDragon+ board from Wytec but moved to their new Dragon12-Plus board in the winter 2008 term, which uses the Freescale MC9S12DG256 microcontroller. We make it easy for the students to program the microcontroller by providing a stationery CodeWarrior project that contains over 80 assembly language routines that can be called as C functions to perform I/O functions including 7-segment displays, keypad, liquid crystal display, pushbutton switches, real-time interrupts, motor speed control, servo position control, interrupt-driven pulse-width measurement, and serial communication. A small *x*-*y*-*z* accelerometer board is interfaced to the microcontroller through its A/D converter port. This accelerometer can be used to measure tilt by measuring the acceleration of gravity. It can also measure inertial acceleration, which can be integrated to obtain velocity and distance measurements. Examples of writing C programs to access all of these I/O functions are given in the book the students use in the course.[4]

This paper will describe our experience in teaching this course for the past three terms and will give examples of labs and student projects that include the design of a digital scale, the measurement of the coefficients of static and dynamic friction, the measurement of the coefficient of restitution of a tennis ball/racquet, and the calculation of projectile velocities using a ballistic pendulum.

Section 2 will describe the operation of the first five weeks of this course in which students learn to program a microcontroller in C. Section 3 will describe the next seven weeks of the course in which students learn statics and dynamics and form teams for their group project. Section 4 will describe some of the group projects that have been done in the past year. Some concluding remarks are given in Section 5.

2. Learning to Program a Microcontroller in a Statics and Dynamics Course

Some of the course objectives in *Design and Analysis of Electromechanical Systems* include solving kinematic and kinetic dynamics problems involving particles and rigid bodies using Newton's second law, work and energy, and impulse and momentum principles. We have found that introducing microcontroller programming at the beginning of the course makes it possible for the students to design and implement experiments which verify these basic principles.

Thus, the first five weeks of the course cover programming the miniDragon+ (and now the Dragon12-Plus) microcontroller board from Wytec[5] in C using CodeWarrior. The Dragon12-Plus board was chosen because it contains the Freescale MC9S12DG256 microcontroller (preloaded with the serial monitor) that has many I/O features, includes female headers that make it easy to interface to circuits on the built-in protoboard, and includes a 4-digit on-board 7-segment display, four pushbutton switches, eight slide switches, eight LEDs, a built-in hex keypad and LCD display, and a potentiometer connected to the A/D converter. The board also contains a serial dual D/A converter and an H-bridge that is useful for controlling the speed of DC motors. A built-in connector makes it easy to connect a small accelerometer module to the board. CodeWarrior was chosen as the development tool because it allows a stationery project to be included with all student projects that contains the assembly language routines that we wrote as C functions for use in the student programs, a free evaluation copy is available for students to download to their own computers, interfacing to the Dragon12_Plus board through a serial cable

is simple, and the student programs are downloaded directly to flash memory. The Dragon12_Plus board has a *Load/Run* switch that allows the student's programs to run automatically when the reset button is pressed in the *Run* mode. This same method of programming microcontrollers could be used with similar microcontroller boards.

Microcontrollers such as the Freescale MC9S12DG256 are remarkable devices. They contain not only a sophisticated microprocessor with a rich set of instructions and addressing modes, but they also contain built-in RAM, EEPROM, and flash memory as well as numerous useful I/O ports, including parallel I/O, several different types of serial I/O, timers, and A/D converters.

This extensive list of features makes the programming of these microcontrollers, particularly using assembly language, a daunting task. Quick looks at the many textbooks (or the many datasheets) that describe these microcontrollers will confirm this. These books and datasheets spend a great deal of time explaining how to program the many I/O registers to perform the wide variety of different I/O tasks. However, it is usually something fairly simple that one is trying to do; e.g. turn on a light, read a switch, turn on a motor at some speed, read the value of an A/D converter, or measure some time interval. If one just wants to learn how to do these simple things without getting bogged down in the details of the microcontrollers I/O and internal operation, then a different approach is needed.

We do this by providing a stationery CodeWarrior project that contains over 80 assembly language routines that can be called as C functions to perform I/O functions including 7-segment displays, keypad, liquid crystal display, pushbutton switches, real-time interrupts, motor speed control, servo position control, interrupt-driven pulse-width measurement, and serial communication. The students have access to a book that includes twenty-one worked examples of programming the microcontroller in C using these assembly language C functions.

Students attend a 3-hour lab each week. In the first lab they follow a CodeWarrior tutorial to program the first four examples in the book. Listing 1 shows an example that produces a hex counter on the rightmost 7-segment display.

The students are then introduced to basic human interfaces. First, digital inputs to the microcontroller are discussed using switches, in the form of toggle switches, pushbuttons, and a hex-keypad. This is followed by a discussion on how to use the onboard LCD as well as the serial communication interface (SCI) in conjunction with a simple terminal program on a PC. Using the low-level assembly drivers provided, the students then work on Lab 2 where they are to write a set of small C programs that utilize these I/O devices. In one problem they are asked to write a program that identifies on a 7-segment display a button pressed on the hex keypad. In another problem, they are asked to read eight toggle switches as an ASCII character and display it on the LCD screen. In a third problem, bidirectional communication is implemented using the hex keypad and LCD on one side and a PC running a terminal program on the other.

Listing 1 – Example done in Lab 1

```
// Example 4b: Single Digit 7-Segment Decoder
#include <hidef.h>      /* common defines and macros */
#include <mc9s12dg256.h>     /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dg256b"

#include "main_asm.h" /* interface to the assembly module */

void main(void) {
  int i;
  seg7_enable();        // enable 7-segment display
  led_disable();        // disable LEDs
  while(1){
    for(i = 0; i < 16; i++) {
       seg7dec(i,3);
       ms_delay(500);
    }
  }
}
```

In preparation for Lab 3, binary number encoding standards are discussed. Specifically, the conversion between multi-digit decimal numbers represented as ASCII strings and their binary integer equivalents are examined. Example functions provided to students allow user entry and display of numbers through a serial terminal program and the display of multi-digit integers on the LCD. In Lab 3, the students then use a calculator program that is in their book that enters two decimal numbers from the keypad and displays a running sum on the liquid crystal display when key A is pressed. They modify this program to a) display the result of subtracting the second number from the first number when they press the B key, b) display the result of multiplying two numbers when they press the C key, and c) display the result of dividing the second number into the first number when they press the D key.

Next, the students learn about analog to digital converters (ADCs) using the potentiometer available on the Dragon12-Plus board that is set up as a voltage divider. Examples are provided that display the raw ADC reading as well as the corresponding voltage reading on the LCD. This if followed by examples on pulse width modulation (PWM) and its application to driving small DC motors and servos. In Lab 4, the student are first asked to measure ambient light intensity using a light-sensitive resistor though an ADC and change the brightness of an LED accordingly using PWM. In the second part of Lab 4, they are to implement a system that employs the H-bridge available on the Dragon12-Plus board and allows the adjusting of the speed and direction of a DC motor though a terminal program based textual user interface.

Lab 5 uses two sources of interrupts: a periodic real-time interrupt and an interrupt that occurs when a character comes in the serial (SCI) port. Examples of using these two interrupts separately are given in their book. The students write a program that a) blinks the letters OU in Morse code on the 7-segment display using a real-time interrupt, b) allows the user to type characters from the keypad and have them displayed on the first line of the LCD, and c) displays their name coming from a text file on the PC in through the serial port on the second line of the

LCD. Because of the use of interrupts the three parts of this program appear to be running simultaneously. This example gives the students a good feel for how interrupts are used to generate concurrency in their programs. Listing 2 shows how the two interrupt service routines are written in C.

Listing 2 – Lab 5

```
// Lab 5: Interrupt-Driven Controller: OU
#include <hidef.h>      /* common defines and macros */
#include <mc9s12dp256.h>     /* derivative information */
#include "main_asm.h" /* interface to the assembly module */
#include "queue.h"
#pragma LINK_INFO DERIVATIVE "mc9s12dp256b"
unsigned short dtime;  // delay time
int ix;                // index into states
const int numstates = 12;
const char seg7[] = {
  0x3F, 0x00, 0x3F, 0x00, 0x3F, 0x00,   // O
  0x3E, 0x00, 0x3E, 0x00, 0x3E, 0x00,   // U
};
const char delay[] = {
  0x24, 0x24, 0x24, 0x24, 0x24, 0x24,   // dashes
  0x0C, 0x0C, 0x0C, 0x0C, 0x24, 0x30,   // dot-dot-dash
};
void interrupt 7 handler_rti(){
  dtime--;
  if(dtime == 0){
    seg7_on(seg7[ix],3);      // turn on next display
    dtime = delay[ix];        // get next delay time
    ix++;                     // increment index
    if(ix == numstates){      / after going through all states
      ix = 0;                 //  reset index to 0
    }
  }
  clear_RTI_flag();
}

// SCI0 receive Interrupt Service Routine
void interrupt 20 handler_sci(){
      qstore(read_SCI0_Rx());
}

void main(void) {
  char  c, a;
  char  i, ii;
  PLL_init();          // set system clock frequency to 24 MHz
  led_disable();       // disable LEDs
  seg7_enable();       // enable 7-segment displays
  initq();
  RTI_init();
  ix = 0;           // reset index into states
  dtime = 1;        // start display right away
  lcd_init();           // enable lcd
  keypad_enable();                  // enable keypad

  SCI0_int_init(9600);   // initialize SCI0 at 9600 baud with interrupts
  ii = 0x14;                  // address of line 2 of LCD
  while(1) {
    c = keyscan();            // read keypad
    if(c != 16){
      a = hex2asc(c);         // convert to ascii
      set_lcd_addr(0x00);     // display on 1st line
      data8(a);               //   display on LCD
      wait_keyup();           //   wait to release key
    }
```

Listing 2 (cont.) – Lab 5

```
    while(qempty() != 1){     // if queue is not empty
        c = getq();           //  and display on lcd
        set_lcd_addr(ii);     // display on 2nd line
        if(c >= 0x20){        //  printable chars
           data8(c);          // write it to the LCD
           outchar0(c);       // echo it back
           ii++;              // inc address
        }
    }
  }
}
```

3. Statics and Dynamics

The next seven weeks of the course are devoted to mechanics, with laboratory exercises utilizing both MATLAB and the Dragon12-Plus board. For example, in Lab 6 in the Winter 2007 term each group of students built a simple digital scale by cutting out an "L" shaped piece of poster board, as shown in Fig. 1, with a hole at *A* to hang the scale and a counterweight W_2 (large binder clips are used for counterweights). The Freescale MMA7260QT accelerometer (part of the MMA7260Q evaluation board[6]) is used to measure the tilt angle θ. The weight to be measured is W_1.

Taking moments about point A, we find:

$$W_1 = W_2 \frac{L_2}{L_1} \tan\theta \qquad (1)$$

Thus, with the proper selection of W_2, L_1 and L_2, and using the Freescale accelerometer to measure the angle θ, the students are able to construct a scale with a range from 0-2 oz to an accuracy of at least 0.25 oz.

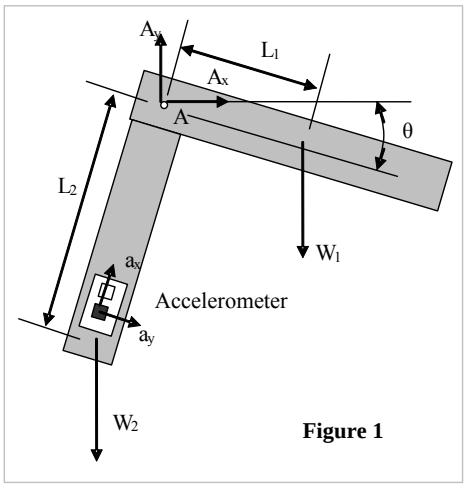

Figure 1 Setup for making a digital scale

In another experiment, the accelerometer module was used to measure the coefficient of static friction using the setup shown in Fig. 2. A small block is placed on a piece of poster board that is taped at one end to the table. The other end of the poster board is slowly lifted until the block just starts to slide down the inclined plane. This occurs at an angle θ where the coefficient of static friction μ_s is given by

$$\mu_s = \tan\theta \qquad (2)$$

This result is easily derived by drawing a free-body diagram of the block.

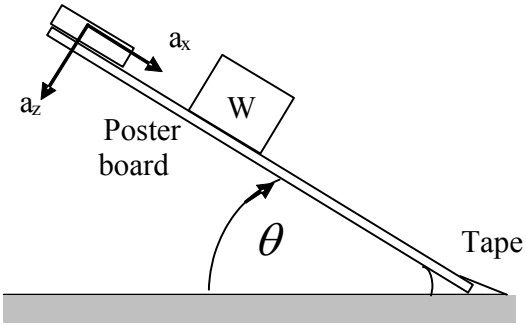

Figure 2 Measuring the coefficient of static friction

In this example, the accelerometer is mounted on the poster board as shown in Fig. 2 where a_x and a_z measure the components of the acceleration of gravity, g. Thus,

$$a_x = g\sin\theta \qquad (3)$$

and

$$a_z = g\cos\theta \qquad (4)$$

From Eqs. (2) – (4) we see that

$$\mu_s = \tan\theta = \frac{a_x}{a_z} \qquad (5)$$

Listing 3 will perform this calculation.

Listing 3 Program to calculate the coefficient of static friction

```
// Calculating coefficient of static friction using an accelerometer
#include <hidef.h>      /* common defines and macros */
#include <mc9s12dp256.h>     /* derivative information */

#include "main_asm.h" /* interface to the assembly module */

#pragma LINK_INFO DERIVATIVE "mc9s12dp256b"
int ax;
int az;
int a0;
int i;
int tan_theta;

void main(void) {
  PLL_init();              // set system clock frequency to 24 MHz
  ad1_enable();             // enable a/d converter 1
  lcd_init();               // enable lcd
  a0 = 0;                   // average 8 values of ax to get a0
```

Listing 3 (cont.) Program to calculate the coefficient of static friction

```
  for(i = 0; i < 8; i++){
    a0 += ad1conv(0);          // add 8 values of ax
  }
  a0 >>= 3;                   // divide by 8
  while(1) {
    ax = 0;
    az = 0;                    // average 8 values of ax and az
    for(i = 0; i < 8; i++){
      ax += ad1conv(0);        // add 8 values of ax
      az += ad1conv(2);        // add 8 values of az
    }
    ax >>= 3;                 // divide by 8
    az >>= 3;                 // divide by 8
    // calculate coeff of static friction
    tan_theta = 1000*(ax - a0)/(az - a0);
    set_lcd_addr(0x40);           // display on 2nd row of LCD
    write_int_lcd(tan_theta);     // write value in field of 5
    ms_delay(100);                // delay 0.1 seconds
  }
}
```

To calculate the value of $\tan\theta$ in both of the examples above, we must subtract the value corresponding to zero gravity from the measured accelerometer values in the *x* and *z* directions. Let a_0 be the measured accelerometer reading corresponding to zero gravity. We will assume that this value is the same value for both a_x and a_z. That is, a_0 is the a_x value when $\theta = 0$, and is the a_z value when $\theta = 90^o$. We will measure a_0 by measuring a_x when $\theta = 0$. In order to work only with integer values, we multiply Eq. (5) by 1000 before doing the calculation. Thus, the integer that we compute will be

$$1000\tan\theta = 1000\frac{a_x - a_0}{a_z - a_0} \qquad (6)$$

To make the friction measurement, we make sure that the poster board is horizontal and press the reset button on the microcontroller board. This will start the program and calculate a_0 by averaging eight readings of a_x. The poster board is then lifted slowly. The value of $1000\tan\theta$ will continuously be displayed on the LCD.

4. Student Projects

During the last three weeks of the class students work in multidisciplinary teams to design and implement a computer-controlled electromechanical system that demonstrates some statics or dynamics principle. Each group typically includes at least one ME major and at least one EE or CE major. Most projects used a Freescale KIT3109MMA7260QE 3-axis accelerometer.[3] This accelerometer measures the acceleration of gravity as well as inertial acceleration. It is therefore useful for measuring tilt angles as described in Section 3. Table 1 is a list of the student projects done in the last three terms.

Table 1 List of Student Projects

Fall 2006	Winter 2007	Fall 2007
Digital Tape Measure	Remote Control Car	Computer Controlled Catapult
Digital Scale	Dart Gun Chronograph	Accelerometer based Mouse
Real-time Image Tracker	Auto Braking System	Airbag Deployment System
Coefficient of Friction	Ballistic Pendulum Spring	Friction Table
Ballistic Pendulum	Coefficient of Restitution Pendulum	Putterometer
Hovercraft	Fuzzy Logic Balance	Self-Leveling Platform
Coefficient of Restitution	Ping-Pong Launcher	LED Wheel Display
Friction Calculator	Nerf Ball Cannon	Resonant Frequency of Motors
Self-Leveling Platform	Object Averting Vehicle	Dart Board
Gear Ratio Efficiencies	Coefficient of Restitution – Vehicle	Force Constants of Springs
Labyrinth	Centripetal Acceleration Device	Motor with Pulley Reduction

One of the groups in the fall 2006 term built the device shown in Figure 3 to calculate the coefficients of both static and kinetic friction by increasing the slope of an inclined plane using a servo motor until a block starts to slide down the plane. The static coefficient of friction is found by measuring this angle with an accelerometer; the kinetic coefficient of friction is found by measuring the speed of the block with a pair of infrared detectors.

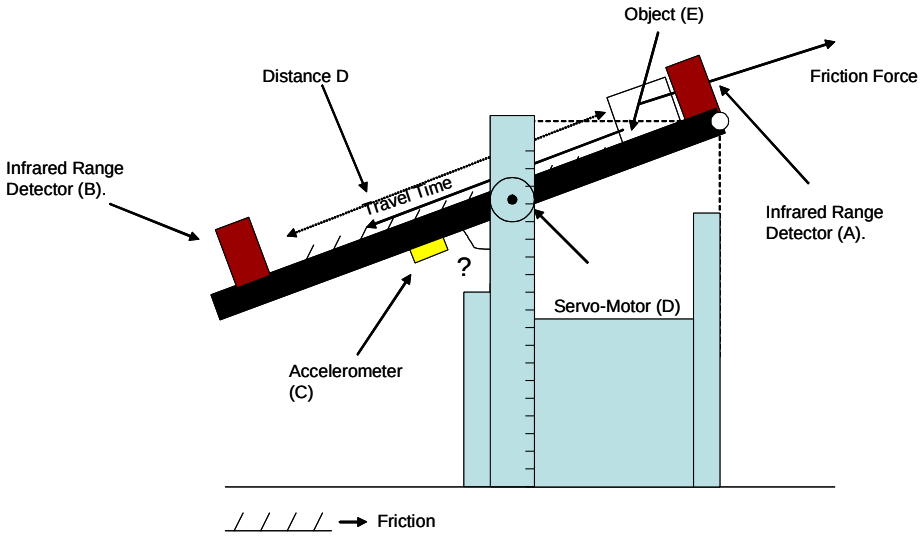

Figure 3 Measuring the coefficients of static and dynamic friction

Figure 4 shows another group's real-time image tracker. They wrote a Visual Basic program on the PC that allowed them to draw a figure on the video monitor and then send the information about this figure to the miniDragon+ board shown in the Fig. 4. The C program on the microcontroller then controlled the two servos so as to have a pen connected to the end arm draw the figure on a piece of paper. This part of their project worked and was very impressive. The group was very ambitious and also tried to solve the inverse problem of moving the arm manually, measuring the acceleration using the accelerometer mounted at the end of the arm, calculating the resulting position of the arm, and sending this data to the PC to be displayed on

the video monitor. Getting accurate, noise-free, accelerometer readings and putting everything together proved to be too ambitious for a 3-week project! However, everyone who witnessed the pen drawing the figure sent from the PC was very impressed.

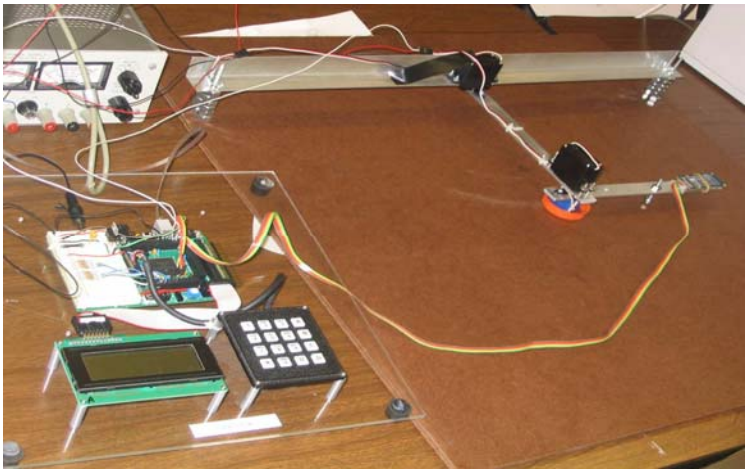

Figure 4 Setup for the real-time image tracker

As a third example from the fall 2006 term, one group measured the coefficient of restitution of a tennis ball bouncing from a tennis racket by attaching an accelerometer to the tennis racket and measuring the time intervals between three adjacent bounces. Acceleration data were collected every 10.24 ms using real-time interrupts and stored in a queue. These data were then sent out the serial port to the PC where a Visual Basic program analyzed the data and calculated the coefficient of restitution. A typical acceleration waveform is shown in Fig. 5 and the measured coefficient of restitution closely matched the published values for new tennis balls.

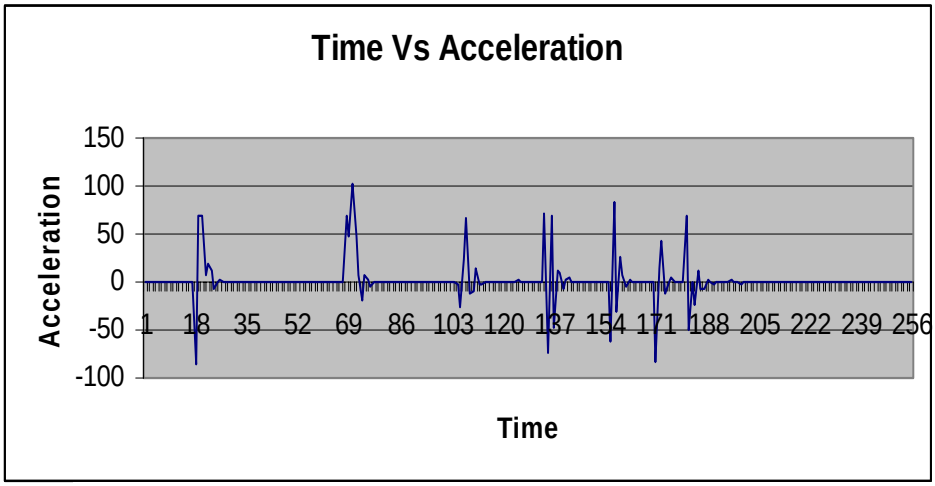

Figure 5 Acceleration data for measuring the coefficient of restitution of a tennis ball

Another example of finding the coefficient of restitution from a project in the winter 2007 term is shown in Fig. 6. The accelerometer was mounted on a vehicle that rode on a frictionless track fabricated by the students. Air was blown out of a series of holes that were drilled in a staggered pattern to get a consistent air pillow over the track. The car was given an initial velocity and then traveled down the track until it collided with a fixed wall. The rubber band on the vehicle absorbed the forces of the collision and then 'sprang' back giving the car a final velocity in the opposite direction. Acceleration data were used to determine the velocities and the coefficient of restitution. Typical data collected in this experiment are shown in Fig. 7.

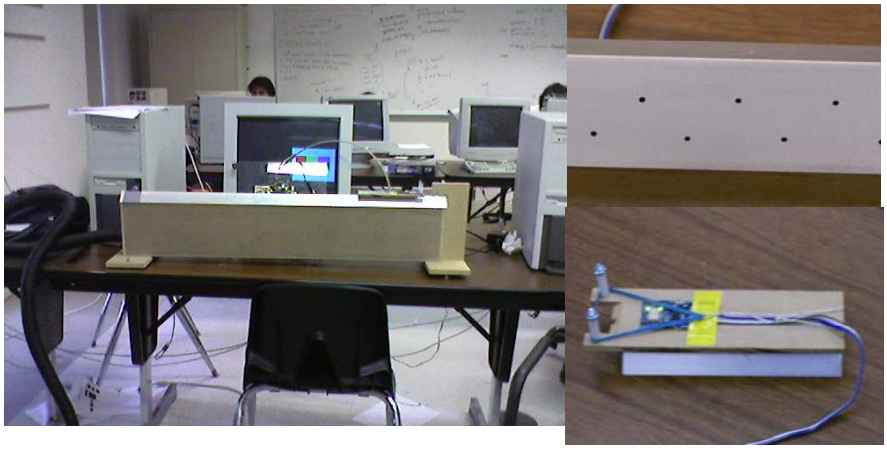

Figure 6 Measuring the coefficient of restitution of a vehicle moving on a frictionless track and colliding with a wall.

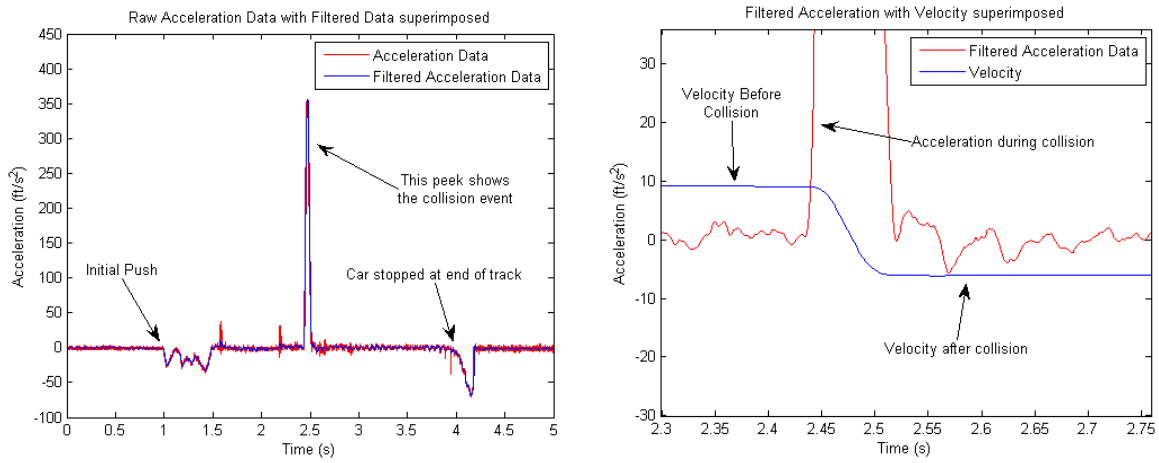

Figure 7 Example of acceleration data collected in the experiment shown in Fig. 6.

In the fall 2007 term one group designed a putterometer by attaching the accelerometer to a golf putter as shown in Fig. 8. The distance the ball travels can be predicted by using the friction coefficient, the momentum principle, and the conservation of energy equation. The accelerometer is used to measure the acceleration of the club and this acceleration is integrated to find the velocity of the club before and after impact with the ball. This change in velocity of the club is related to the initial velocity of the ball, which is used to calculate the distance that the ball will travel. This distance is displayed on an LCD. A separate experiment that rolls the ball down an inclined plane onto the synthetic grass putting surface was used to determine the coefficient of friction. The idea of the putterometer is to train the golfer to use the appropriate acceleration to cause the ball to roll a particular distance.

Figure 8 The putterometer

5. Conclusion

Programming a microcontroller is an essential skill for implementing most useful, modern design projects. It is therefore a skill that all engineering students should have as they enter modern engineering practice. In a new core course at Oakland University all engineering students learn to program a Freescale MC9S12DG256 microcontroller in C using CodeWarrior. Being provided with a stationery project that contains over 80 assembly language routines that can be called as C functions, the students can concentrate on solving a particular design problem using the microcontroller, rather than having to dig into the details of all of the various I/O registers.

Having a microcontroller and accelerometer to use in the lab also enhances the learning of statics and dynamics as demonstrated with the digital scale lab described in Section 3. At the end of the course all groups gave a PowerPoint presentation and demonstration of their project in addition to writing a group project report.

By using a microcontroller and modern engineering tools students get deeply involved in designing experiments that illustrate the principles of mechanics. This deepens their understanding of these principles and leads us to the conclusion that microcontrollers have found a useful home in a sophomore course in statics and dynamics. As an added benefit we noticed last term that for the first time mechanical engineering students in our multidisciplinary senior design course[7] were able to contribute to the programming of microcontrollers and did not have to rely entirely on their computer engineering colleagues.

References

1. Y. P. Chang, "A Slider/Ramp Apparatus Capstone Design Project for a Hands-On Senior-Level Laboratory Design Experience," Proc. ASEE Illinois-Indiana and North Central Joint Section Conference, Indiana-Purdue University, Fort Wayne, IN, March 31 – April 1, 2006.
2. Y. P. Chang, "A Unique String/Pulley System Apparatus Capstone Design Project for a Hands-On Senior-Level Laboratory Design Experience," Proc. ASEE Illinois-Indiana and North Central Joint Section Conference, Indiana-Purdue University, Fort Wayne, IN, March 31 – April 1, 2006.
3. Y. P. Chang, "A Pendulum Impact Apparatus Capstone Design Project for a Hands-On Senior-Level Laboratory Design Experience," Proc. 2008 North Central Section Conference, Wright State University, Dayton, OH, March 29, 2008.
4. Richard E. Haskell, *Learning By Example Using C – Programming the Dragon12-Plus Using CodeWarrior*, LBE Books, Rochester, MI, 2008.
5. Wytec Company, *www.evbplus.com*.
6. Freescale Semiconductor Inc., *www.freescale.com*.
7. Michael A. Latcha, Subramaniam Ganesan, Edward Y. L. Gu, and Richard E. Haskell, "The Melting Pot Approach to Senior Design Part II: Assessment and Improvement," Proc. ASEE NCS Spring Conference, EV-2, Ohio Northern University, Ada, Ohio, April 7-8, 2005.